Ph. D. Dissertation

Music similarity based on the joint use of Discrete Riemann metrics and Immune Artificial Systems

Jesús Sanz Marcos

Thesis Advisor: Rafael González Ayestarán

March 2015

A mis padres, Jesús y María Victoria,
y a mi hermana Nuria.

A Sara, mi mujer. A mis hijos, Jesús y Marina.

Today we still yearn to know why we are here and where we came from. Humanity's deepest desire of knowledge is justification enough for our continuing quest. And our goal is nothing less than a complete description of the universe we live in.

Stephen Hawking, A Brief History of Time.

Acknowledgments

I started to work part-time in Music Recommendation in 2000, while I was studying at the university. However, this is my second attempt to write a PhD. The first attempt started in 2003 at the Signal Theory and Communications (TSC) of the UPC. The plan was to build a novel ground based Radar that could take terrain images using the microwave illumination provided by a satellite. This satellite only passed through the interest site during 1 second, every 35 days. We accomplished it and it was the very first time that a civil ground-satellite SAR radar was successfully designed and deployed.

When I started writing the manuscript after releasing the papers and finishing all the experiments, we were given a five day isolated conference to convince us of the low probability that existed of staying in the Academia after publishing our PhD. I think I will regret this forever (*or not*) but this conference opened the Pandora box for me. It also helped that at those days I was living in a hurry between Barcelona and Madrid. More than fifty round trips by train and the same number by plane (when the petrol was not so expensive) to see the girl that a couple of years later became my beloved wife.

So when the only job position in Spain (and in Madrid!) for a Space Radar Engineer appeared, I decided to cancel my scholarship for the PhD and accept the radar job, as it was the job of my life... During the following year and half in Indra Espacio as a Radar engineer I really enjoyed it, looking for stealth submarines using radar technologies in regions of the Earth that I cannot disclose. And I was able at that time to keep writing the PhD manuscript. Some of this sincere acknowledgements are for this first attempt *era*.

First of all, I would like to thank Rafael Ayestarán and Fernando Las-Heras, from the Signal Theory and Communications Department of the University of Oviedo, for the patience, time, real support and helpful interest that they have shown since our initial discussions about my second PhD attempt.

I am in debt with Jordi *Joan* Mallorquí, my final degree project and first Thesis advisor. Thank you for the multi-lateral support and for the trust you deposited on me. I followed your advice once resigning my previous work before accepting the PhD grant and I did not follow your second advice when I renounced to the last year of grant and accepted a position in Madrid. For all that, my most sincere *thank you*.

The first *awesome* person I am deeply grateful to is Pau Prats. He is an outstanding person, persistent, determined, dogged, strong-willed, tireless, indefatigable, resolute, patient, unflagging, staunch, steadfast, untiring, unwavering, unswerving, unshakable, unyielding, insistent and obstinate. He has always been a good teacher and a better colleague. One of the most interesting ideas of the first thesis came to light during a discussion with him.

The first *super*-person I feel like I can not find the words to summarize how appreciative I am is of course Josef Mittermayer from the German Aerospace Agency (DLR). He gave me the opportunity to take part in a real space project and in a very sensitive and challenging problem of the image processing chain for the TerraSAR-X satellite. In addition, while I was working with him, he always managed to find the time to answer my doubts and to discuss some initiatives in the bistatic field. As I know your Spanish is excellent: *mucha suerte y felicidad*. To Alberto Moreira, head of the Radar Institute of the DLR, thank you for giving me so many different opportunities.

I have to thank especially the Signal Theory and Communications (TSC) of the UPC, led by Toni Broquetas. Albert Aguasca helped me in every possible aspect to transform the initial vague idea of a bistatic system in a real hardware prototype which gave this thesis another dimension. Thank you for what you taught me and also for *coupling the DC* of the receiver.

I also thank Paco Dekker (next generation of bistatic researchers) and Xavier Fábregas for your criticism and trust. I had the chance to work together with Ángel Cardame and Lluís Jofre (both from the TSC) in the organization of the Iconic 2005 International Conference. I would like to *transmit* them (as they are teachers of Antenna Theory) that what they taught me has helped me to grow up as a human being more than any science book I could read this last years.

To the young core generation of the Remote Sensing group (both passive and active) at the UPC, I need to say how sorry I am for not having been there at 100% due to my *6 days PRF* trips to Madrid with a duty cycle on 3 days. Pablo Blanco, Sergi Duque, Gerard Margarit, Sandra Monerris and David Navarrete I am a grateful man who will not forget (I have always wanted to use this last

sentence!).

But in 2006, I decided to swap my life. Move to Asturias and build my own Signal Processing Startup. This decision changed my life and I have really lived under much more pressure and stress than my years at Academia or at Indra Espacio. I have worked in more projects than I every thought possible and some of those projects have been really interesting. Mireia Bellot has been my partner in this Music Recommendation adventure. We have worked togeher more than 10 years in the music business: *you are my work-soul-mate*.

I also want to thank my life-long friend Manuel Pérez, who was with me while I was dressing for my wedding and has helped me in every aspect. There are a few more friends: Joan Griso, Jordi Velasco, Clara Aguiló, Òscar Serra, Ricard Lozano, Pablo Balañá, Ricardo Carreras, Luis Aliaga, Alex Posada, Pablo Reyero, Antonio Trias, Vanessa Karjalainen, Eugenia Seco, Sonia Ribas, and *super* Xavier Osó.

Thanks Xavi for letting me be your best-man in your wedding. Thank you Ivan and Carlos for sharing the afternoons and evenings for the last years and thanks Miguel Ropero and Andrea González for being so good friends.

My parents supported me when I decided to *stop working* to do research for a couple of years and now with my own signal processing little company. I hope that after reading this manuscript they will change their mind on the *not working* thing. Thank you for attenuating my humor during the stressful year of the wedding and for obliging me to spend studying one hour at least every day of the year, even though during summers. I will try to keep doing it and transmit with my example the message to the incoming generations, as you did.

Last, but not least, Sara, my dear girl-friend-wife who has suffered more than any one my *concentration* during the last decade. Love you always.

Trying to answer if this stress was worth it from a personal point of view (with my friends and family) or for my research at the university has no sense now. However, might these lines be useful to make sure everyone knows how sincerely grateful I am. I still remember the day when I was 7 years old and my teacher said that my father was at school and that he wanted to talk to me. He just wanted share with my sister and myself that he had finally presented his PhD, after many years of hard work.

The final thought is for Toni Trias, who is my friend, professor, mentor and colleague. It is a quote from Chinese philosopher Lao Tzu (-6th Century) that was used in the Star Wars movie:

When the student is ready the teacher will appear. When the student is truly ready... The teacher will disappear.

Jesús Sanz Marcos
Salinas, May 2015

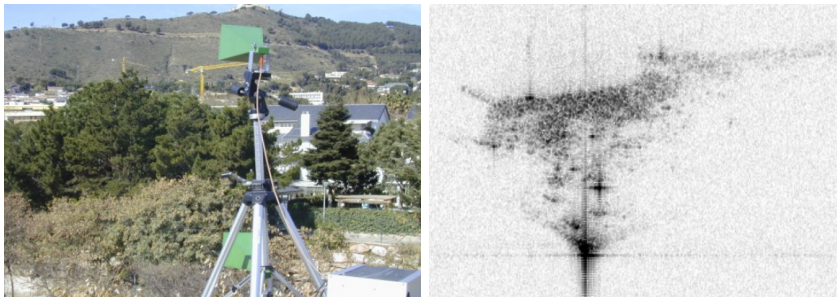


Figure 1: First thesis attempt results: radar pointing to the satellite illuminated area (left) and bistatic reflectivity image first acquisition (right)

Contents

List of Figures **xxiv**

List of Tables **xxv**

List of Symbols and Acronyms **xxvii**

1 Introduction **1**

1.1 Motivation 1
1.2 The problem 2
1.3 The solution 4
1.4 Summary of contributions 6
1.5 Thesis outline 7

2 Music recommendation by signal processing **9**

2.1 What is Music? 9
2.2 Tracks and songs 12
2.3 Pitch and Timbre 13
2.4 Rhythm, Loudness and Harmony 16
2.4.1 Rhythm 16
2.4.2 Loudness 19
2.4.3 Harmony 19
2.5 Ear audition model 20
2.5.1 Preprocessing of sound in peripheral system 20
2.5.2 Mechanical to neural representation 22
2.5.3 Sound perception 22
2.6 Identity and similarity 23
2.7 Descriptor based similarity: First attempts 25

2.7.1 ANALYST1.0 25
2.7.1.1 Timbre Analysis 26
2.7.1.2 Volume 31
2.7.1.3 Rhythm and Tempo 31
2.7.1.4 Noise 33
2.7.2 ANALYST2.0 34
2.7.3 ANALYST3.0 36
2.7.4 ANALYST3.1 37
2.8 Principal Component Analysis applied to descriptors 39
2.9 Descriptor clustering 44
2.10 Raw similarity benchmarks 45
2.11 First patent description 46
2.11.1 Analyzing music and extracting clip 46
2.11.1.1 Clip extraction 48
2.11.2 Recommendation engine 48
2.11.3 User's taste profile and recommendations 50
2.11.4 User's music preference and recommendations 51
2.11.5 Hit prediction 52
2.12 Summary 53

3 State of the art in Music Recommendation 55
3.1 The beginnings 55
3.2 2000 56
3.3 2010 64
3.4 2011 68
3.5 2012 70
3.6 Most interesting lines 71
3.6.1 Based-content Searching 71
3.6.2 Speech Recognition applied to Music 73
3.6.3 Music Identification and Fingerprinting 74
3.6.4 Music Recommendation 76
3.6.5 Artist Similarity 78
3.6.6 People behavior and perception of music 79
3.7 Genre classification 80
3.8 Summary 82

4 Riemann Metrics 85
4.1 Introduction . . . 85
4.2 Feature Selection . . . 86
4.3 Feature extraction . . . 87
4.4 Principal Component Analysis . . . 89
4.5 Independent Component Analysis . . . 90
4.6 Riemann Surface . . . 91
4.6.1 Classification of Riemann surfaces . . . 93
4.7 Discrete Riemann metrics . . . 95
4.7.1 Manifolds . . . 95
4.7.2 Tensor products . . . 96
4.7.3 The Riemannian metrics . . . 97
4.8 Metric Estimation Process . . . 98
4.8.1 Learning set . . . 99
4.8.2 Identical embedding . . . 99
4.8.3 Selection of learning triplets . . . 101
4.8.4 Margin . . . 102
4.8.5 Soft margin . . . 105
4.8.6 Unbiased soft separation problem . . . 108
4.8.7 Hyperplane with bias . . . 109
4.8.8 Example . . . 111
4.9 Building Riemann manifolds with Non Hyperbolic Adjacent Region 112
4.9.1 Gradient and Hessian . . . 115
4.9.2 Reductions. Sammon's method . . . 117
4.9.3 Vector q-Sammon . . . 119
4.9.4 Spherical q-Sammon . . . 120
4.9.5 Vector q-Sammon for the Euclidean case . . . 123
4.10 Analysis of track descriptor data . . . 124
4.10.1 Genres . . . 124
4.10.2 RMS . . . 125
4.10.3 Spectral Centroid . . . 126
4.10.4 Spectral Bandwidth . . . 126
4.10.5 MFCC . . . 127
4.10.6 Onset Long-term descriptors . . . 127

4.10.7 Spectral fluctuation patterns . . . 129
4.10.8 Pitch Class Profile . . . 130
4.11 Euclidean Metric Results . . . 131
4.11.1 Normalization . . . 131
4.11.2 Supervised benchmark similarity database . . . 131
4.11.3 MP3 vs WAV formats . . . 136
4.11.4 Principal component analysis . . . 139
4.12 Riemann metrics Music recommendation algorithm . . . 143
4.12.1 Training database . . . 143
4.12.2 Background on machine learning . . . 144
4.12.3 Description of the algorithm . . . 146
4.13 Results . . . 149
4.14 Summary . . . 149

5 Immune Artificial Systems for Music Recommendation 151
5.1 Introduction . . . 151
5.2 Millisounds . . . 157
5.2.1 Music Information Reduction . . . 157
5.2.2 Phasor motion representation . . . 157
5.2.3 Low encoding quality and low sampling rate signals . . . 162
5.2.4 Innovation detector . . . 164
5.2.5 Extraction of millisounds . . . 167
5.2.6 Vector quantization . . . 170
5.2.7 Negative selection and applied immunology for millisound matching . . . 175
5.3 Applied Immunology . . . 177
5.3.1 Introduction . . . 177
5.3.2 First layer: the Skin . . . 177
5.3.3 Second layer: the medium . . . 178
5.3.4 Immunology layers . . . 179
5.3.4.1 Inborn immune system . . . 179
5.3.4.2 Acquired immune system . . . 180
5.4 Results . . . 181
5.5 Summary . . . 182

6 Long Tail Distribution for Music Popularity 185
6.1 Introduction . . . 185
6.2 Model . . . 189
6.2.1 Scale Free Analysis . . . 190
6.2.2 Parameter Adjustment . . . 191
6.2.3 RE-FIT . . . 192
6.2.4 HRE - Fit . . . 193
6.2.5 Behaviour near $R = 0$ . . . 195
6.2.6 Behaviour at R=∞ . . . 197
6.2.7 Tail Exponent Assymptotic Fit . . . 198
6.3 Mixture of models . . . 199
6.3.1 Sice . . . 200
6.3.2 Reciprocal Mixtures . . . 201
6.3.3 Three parameter Regression . . . 203
6.3.4 Approximate Regression . . . 204
6.4 General Second Order Model . . . 205
6.4.1 Area . . . 206
6.4.2 Tail . . . 207
6.4.3 4+1 Parameter Regression . . . 207
6.4.4 3+2 Parameter regression . . . 208
6.5 Discrete Long Tail model . . . 209
6.5.1 The unit limit axiom . . . 210
6.5.2 The continuous model . . . 212
6.5.3 The second order model . . . 213
6.6 *mElo* algorithm . . . 214
6.6.1 Algorithm description . . . 216
6.6.2 Long Tail Parameters . . . 217
6.6.3 Initialization . . . 217
6.6.4 Methodology . . . 218
6.6.5 Song representation . . . 219
6.6.6 Value propagation Tournaments . . . 220
6.6.7 Evolution in each iteration . . . 222
6.6.8 Algorithm results . . . 222
6.7 Radio algorithm with MELO and similarity music selection . . . 224

6.7.1 Radio journey description 224
6.7.2 Graph implementation 225
6.7.3 Path-finding through track music graph 228
6.7.4 Alternative reduction to playlist 230
6.7.5 Immunology approach to final selection 231
6.7.6 Summary 232

7 Conclusions and future work 235
7.1 Human sensor bandwidth 235
7.1.1 Machine training 236
7.2 Music 237
7.3 What is music and what is not 237
7.4 Unsupervised music classification 238
7.5 Future of music recommendation - long tail boost 238

A 241
A.1 Vorbis audio compression 241
A.1.1 Fidelity measurement and terminology discussion 241
A.1.2 Fidelity, Artifacts and Differences 242
A.1.3 Decode Setup 243
A.1.4 Decode Procedure 243
A.1.5 Floor decode 245
A.1.6 Residue decode 245
A.1.7 Inverse channel coupling 245
A.1.8 Generate floor curve 245
A.1.9 Compute floor/residue dot product 246
A.1.10 Inverse monolithic transform (MDCT) 246
A.1.11 Overlap/add data 246
A.1.12 Cache right hand data 247
A.1.13 Return finished audio data 247

B 249
B.1 MELO Algorithm in Matlab 249

List of Figures

1 First thesis attempt results: radar pointing to the satellite illuminated area (left) and bistatic reflectivity image first acquisition (right) x

1.1 Diagram that depicts a journey in a portion of the music universe. Tracks in red are classic UK hits and tracks in blue are Pop UK hits. The tracks have been organized using the similarity algorithm proposed in this dissertation and applying a Self Organization Map technique to project in two dimensions. Inside every track there is an estimation of the *colour* of each track. Live demo is available at https://tunehog.s3.amazonaws.com/cuban.html. . 4

1.2 Diagram of the proposed solution based on the joint use of descriptor based metrics and an artificial immunology algorithm. . 5

2.1 The piano keyboard diagram below shows the various piano notes C, D, E, F, G, A and B. There's a treble clef, a bass clef and a grand staff. Notice that the C in the treble clef and the C in the bass clef are the same note. This is the point where they cross and meet on the Grand Staff. It is known as Middle C. 14

2.2 Phase detection of the human ear. The human is very insensitive to the relative phase of the component sinusoid. For example, these two waveforms would sound identical, because the amplitudes of their components are the same, even though their relative phases are different. 16

2.3 Violin waveform. A bowed violin produces a saw tooth waveform, as illustrated in (top). The sound heard by the ear is shown in (bottom), the fundamental frequency is the highest peak. The other peaks are the harmonics. 17
2.4 Diagrams of most common meters 18
2.5 Italian terms for describing dynamic levels. 18
2.6 Human ear anatomy 21
2.7 Red line represents Brightness 28
2.8 Green line represents Bandwidht 29
2.9 Frequency filter bank with filters 1-3-5 (top) and 2-4-6 (bottom) 32
2.10 Original audio signal in blue (violin) and envelope in red. The envelope was calculating using $y(n) = max(x(n), y(n-1)) * 0.997$. 33
2.11 A simulated plot of (x_A, y_A). The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented. 41
2.12 A spectrum of possible redundancies in data from the two separate recordings with $X/Y = 2$ (red), $X/Y = 5$ (green) and $X/Y = 15$ (blue). The best-fit line $r2 = kr_1$ for all cases has a slope of 55 degrees. 42

3.1 Several audio segmentation and temporal modeling windows. From [West05] 60

4.1 Riemann sphere 94
4.2 Torus 94
4.3 Coffee mug and torus share same topology, while torus and sphere have different topology 95
4.4 Genres used for calculating the histogram of the descriptors. . . . 125
4.5 RMS mean (left) and variance (right). The color mapping can be found in Fig. 4.4. 126
4.6 Spectral centroid mean (left) and variance (right). The color mapping can be found in Fig. 4.4. 127
4.7 Spectral Bandwidth mean (left) and variance (right) Variables . 127
4.8 Variables 7-12 Mel Frequency Cepstrum Coefficients (MFCCs). Mean (left) and Variance (right) 128

4.9 Variables 7-12 Mel Frequency Cepstrum Coefficients (MFCCs). Mean (left) and Variance (right) 128
4.10 Variable 33 - Onset density 128
4.11 Variables 34-35 Onset Energy. Mean (left) and Variance (right) 129
4.12 Variables 36-37 Aggressiveness (left) Focus (right) 129
4.13 Variables 38-39 Gravity (left) Focus (MaxFluctuations) 130
4.14 Variables 40-41 Bass (left) Focus (BassRatio) 130
4.15 Variables 42-47 Pitch Class Profile. Mean (left) and Variance (right) 130
4.16 Similarity matrix obtained with Euclidean metric using the waveform/uncompressed analysis algorithm 136
4.17 Performance of the Euclidean metric k-NN classification for uncompressed and compressed analysis algorithm 137
4.18 Similarity matrix obtained with Euclidean metric using the MP3 analysis algorithm 138
4.19 Correlation obtained from the Principal Component Analysis for the 65 descriptors 139
4.20 Eigen Vectors obtained from the Principal Component Analysis for the 65 descriptors 140
4.21 Accumulated sum of the eigenvalues. With the first 25 principal components it is possible to reconstruct the original space with an accuracy of 90% 141
4.22 Similarity performance as a function of the number of principal component selected 142
4.23 View in stripes of the 8996 tracks used in the training similarity matrix 143
4.24 Zoom over the first 1k quadrant of the supervised training similarity matrix (Green similar, red different) 144
4.25 Poorly classified pairs during evaluation 145
4.26 Metric average (top) and Metric standard deviation (bottom) when the algorithm is auto-applied to the tracks in the supervised training database 148
4.27 Curvature results 149

5.1 Phagocytic cells. From left ro rigth: 1st image a macrophage (blue) consuming leukemic cells (pink); 2nd image a peritoneal macrophage phagocytosis of E. coli Bacteria; and 3rd image, a neutrophil white blood cells or leukocytes. 153
5.2 An antibody is made up of two heavy chains and two light chains. The unique variable region allows an antibody to recognize its matching antigen 155
5.3 Left: The B lymphocyte activation pathway. B cells function to protect the host by producing antibodies that identify and neutralize foreign objects like bacteria and viruses. Right: The T lymphocyte activation pathway. T cells contribute to immune defenses in two major ways: some direct and regulate immune responses; others directly attack infected or cancerous cells . . . 156
5.4 Nature vs Music 157
5.5 Violin Phasors. Red corresponds to the fundamental tone, while the other plots are the harmonics 158
5.6 Violion phasor evolution with substracted fundamental tone phase 159
5.7 Piano phasor evolution with substracted fundamental tone phase 160
5.8 Time-amplitude Original signal (blue) versus quality degraded signal (red) 161
5.9 Frequency-spectrum power of original signal (blue) versus quality degraded signal (red) 162
5.10 Innovation detection using Linear Prediction error. Blue signal represents the original signal and red signal represents the linear prediction error 166
5.11 Mels 169
5.12 Millisound matrix 170
5.13 One dimensional VQ 171
5.14 Two dimensional VQ 171
5.15 Evolution of the LBG algorithm 174
5.16 Probability of cluster to appear in recommendations of a track of another cluster: unordered (left), ordered by probability (right) . 177
5.17 Percentile probability of a cluster to appear in another cluster seed recommendation 178

5.18 Discarding the track that has the least resemblance to the seed track from a Millisound point of view 179
5.19 List of millisound stripes. The first row corresponds to the seed track and the other 20 are the candidates from the second layer. Red is for the first MFCC, blue is for the ceptral mean extracted MFCC and green is for the MEL coefficients 180
5.20 Zoom on the millisound stripes 181
5.21 Final accuracy achieved with the combination of the Riemann curved metric and the Immunology negative-selection layers . . . 183
5.22 Classification accuracy improvement produced by the immunology layers 183

6.1 Popularity vs Products: The highest popular items are located at the left-hand side of the demand curve, in the *Head*; while the millions of various niche markets items are located on the right-hand side of the same curve, in the *Long Tail*. 186
6.2 Streaming artist segments: the new demand curve. 188
6.3 The graph above shows a plot of the model funtcion for $H = 1$ and $E = 0.5$ and several values of the scale parameter R. 190
6.4 Scale free curves and corresponding HET points for various tail exponents or thicknesses. 192
6.5 $H(R)$, $E(R)$ examples from test data. 195
6.6 Extremal value for parameter R. 196
6.7 Area under the curve 197
6.8 E vs Rinf 199
6.9 Mixture of models 200
6.10 Different RMM for fixed parameters $H = 1$, $R = 1$, $E = 0.5$ . . . 202
6.11 Identity vs Similarity 215
6.12 Hits (white) represented over non-hit tracks (black) over blue background. 219
6.13 Similariby based recursive preferences. Use current MELO values in neighborhoods. 220
6.14 Melo Tournements 221
6.15 Rhapsody music downloads at the head region (rank 0 to 25.000) 222
6.16 Rhapsody music downloads (rank 25.000 to 100.000) 223

6.17 Rhapsody music downloads (rank 100.000 to 800.000) 224
6.18 Origin (blue) and destination (cyan) tracks over a Self Organizing Map produced over the original 65 dimension track-to-track distances. 225
6.19 Graph link distances between tracks (black pixel means no direct link between track *col* to track *row*) 226
6.20 Total distance between tracks 227
6.21 Radio graph with postive and negative score propagation 228
6.22 Radio alternatives with stop points and candidates 229
6.23 Radio alternatives with stop points and candidates 230
6.24 Selection of stop-points using distance score 231

List of Tables

2.1 Similarity results for descriptor benchmark using Euclidean Metric 46

4.1 First half of the normalization values used to calculate the Standard score 132
4.2 Second half of the normalization values used to calculate the Standard score 133
4.3 Number of tracks of each subgenres used during Euclidean metric evaluation 135

6.1 Table caption text 233

List of Symbols and Acronyms

Chapter 4

α_m	Lagrangian unknowns
γ	Lesser margin
$\vec{W}^{(e)}$	Estimated metric
C	Data recordset
D_{ac}	Triplet distance vector
$D_{cc'}$	Original space distance
$d_{cc'}$	Reduced space distance
D	Original space dimension
d	Reduced space dimension
$F\{\vec{x}_c\}$	Sammon's functional
F_D	Feature space
H_{ij}	Hessian
$k_{cc'}$	Attractive interaction of a spring
$K(x_n, x_m)$	Metric kernel
K	Sammon's standarization constant
L	Lagrangian
M	Total number of vectors in the training set
q	Electrical power
S_{ac}	Triplet average vector
$T_{abc}^{(ij)}$	Triplet example definition

V_e	Non Coulombian repulsive potential
X_m	Training vector
$X_m^{(\ell)}$	External supervisor labels
x_n	Data records

Chapter 6

ρ_k	Preference reduction between successive ranks
$\rho_k^{k'}$	Ratio of two different download measurements
A_{sf}	Total download number or area below the curve
A	Total number of downloads or area
$D(r)$	The number of measured downloads of ordered tracks at rank
D_k	Measured download number
E	Thickness of the tail
H	Extrapolated downloads at first rank
R	Hit zone transition
R_{asy}	Variable limit of the partial data selection
r	Rank
x_{HET}	Head-tail-even rank

A*	A star search algorithm
ADC	Audio Description Contest
AIS	Artificial Immune System
ATRAC	Adaptive TRansform Acoustic Coding
BSS	Blind source/signal separation
CD	Compact disc
CHMM	Continuous hidden Markov model
DBSCAN	Density-based spatial clustering of applications with noise
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DIY	Do It Yourself
DNA	Deoxyribonucleic acid
DVD	Digital Versatile Disc
EP	Extended play
FFT	Fast Fourier transform
FLAC	Free Lossless Audio Codec
FP	Fluctuation patterns
HMM	Hidden Markov model
HPCP	Harmonic pitch class profiles
HPSS	Harmonic-Percussion Signal Separation
ICA	Independent Component Analysis
ISMIR	International Society for Music Information Retrieval
LBG	Linde, Buzo, and Gray encoding algorithm
LPC	Linear Prediction Codification
MDCT	Modified discrete cosine transform
MFCC	Mel-frequency cepstral coefficients
MIDI	Musical Instrument Digital Interface
MIREX	Music Information Retrieval Evaluation eXchange
MLT	More Like This
MNIST	Mixed National Institute of Standards and Technology
PCA	Principal Component Analysis

PCM	Pulse Code Modulation
PCP	Pitch Class Profile
RMM	Reciprocal Mixtures
RMS	root mean square
SNR	Signal to noise ratio SNR
SVM	Support vector machine
TREC	Text REtrieval Conference
VQ	Vector quantization
WAV	Waveform Audio Format

Chapter 1

Introduction

1.1 Motivation

The World Wide Web without search was like the Earth without maps. There was a very little chance that you would find what you were looking for. Text search engines provided the required directions to search, find and sometimes discover content. Words, sentences and links of web pages were analyzed, organized and presented in such a manner that it was possible to find information and convert it into knowledge. This scenario has evolved and users are finding themselves lost again. The problem is that users have no idea of what they are looking for. They know, however, that it is there, somewhere in the web.

Music consumers are a real example of this paradigm. Users are not asking anymore for specific content (i.e. *notify me when the next album from Bon Jovi is available*) but are waiting for recommendations (i.e. *find my new favorite band, for my mood this week...*) and they will enforce complex constraints in the near future (i.e. *my partner and friends have to like also my next favorite band as we are going in a road trip together*). In parallel, online stores have increased the size of their collections dramatically approaching or passing already the 50M labeled tracks which complicates the search process but at the same time it creates a richer *universe* of music. The unlabeled content has needed more time to explode but now it is showing much more potential and growth than its commercial counterpart.

Machine learning is being accelerated as computer power, memory, parallel processing capabilities and storage continue to ramp up and prices continue to drop incredibly. The most advance algorithms are being designed mimicking

the architecture of biological neural networks. As an example, on the very competitive MNIST handwriting recognition benchmark, a novel method based on Deep Neural Networks [Ciresan12] has been the first to achieve near-human performance. The era of *Computer Aided Intelligence* has already started.

1.2 The problem

The global problem of music recommendation is clearly explained in [Celma08]. Algorithms that are currently offering music to users tend to recommend tracks that are already quite popular in order to keep a higher predictive accuracy of the recommendation system itself. This approach produces an extreme long tail distribution (1% of all digital tracks account for 80% of all sales in 2007) while at the same time it makes music discovery very unlikely. It is known to the industry that the tolerance of music lovers against bad track recommendations is much lower than with normal web text search. This emotional barrier complicates the trade-off between the usefulness of music discovery and the safe region of music popular hit recommendation. The goal of a *Computer Aided Intelligence* algorithm applied to music recommendation is to provide the right tracks in every moment. There are four basic scenarios:

- The user is browsing a music catalog as if it was searching a library. User might search for music using a complex set of filters such as genre, mood, sells, downloads, similarity, etc and is offered with a list of tracks that might match the search criteria. The user might be willing to confirm his search by listening briefly to the recommended tracks (between 2s and 30s clips). Finally, for the tracks that have been correctly recommended (basically the ones the user likes at that specific moment) a music service can start any of its offered services such as a radio station, downloading the track, looking for other similar tracks, etc.
- The service compiles a list of *right* tracks to be listened by the user at any given time. The recommended playlist is sent to the user pro-actively.
- The service recommends artists, albums, collections, concerts, etc. Basically an abstraction layer on top of a playlist. It can also be expanded to recommending friends or people with similar music interests
- The user wants to start a radio station or it is already listening to a track in radio mode (the track comes to its end and then next track of the playlist has to be played). Which is this *next right track* is what a radio recommendation engine would provide.

This dissertation work does not provide a complete solution to any of the above problems. For this, there are many ways of building a recommendation

algorithm and most of the time it is the result of a combination of completely different techniques. A recommender presents items to users by exploiting user profiles, the items themselves and past relations between items, users and items to users (user's that buy item A also tend to buy item B). Hybrid methods mix demographic and collaborative filtering with context and content-based recommendations (extracting features from the audio tracks) to offer the user with a greater choice. Even the evolution of the mechanical Turk machine produces extremely good playlists as users themselves prepare playlists for other users just to get more reputation from the social network. The problem that the author tries to solve in this dissertation is the kernel of any content-based recommendation system:

How to find a subset of *similar* tracks to a given seed track from a *collection* of tracks by just having access to the *audio recordings*. No metadata and no collaborative filtering, just pure music similarity over million of tracks, and in real-time.

The size of the music catalog (where the algorithms look for similar tracks to the seed track) is increasing by thousands every day as the music collections are getting bigger. At the same time the feature-set produced by analyzing each track is also growing in order to provide more accurate playlists. The audio-music recommendation algorithm has to embrace this growing dataset in its design but at the same time it has to allow for fast recommendations so it can be used in offline or online mode by the industry. The first architectures were based purely in offline calculations were a single graph connected by distance links all tracks of the catalog, therefore there was little need of real-time calculations and brute-force parallelism was used. However, the latest product design require many more of this calculations to happen in real-time, while the user is *using* the service. Recommendations are moving to a user-centric paradigm with instant feedback.

Imagine a service where the user can select an origin track and a destination track and obtain one hour radio experience, which is nice and smooth transition between the two tracks. The algorithm has access to the user music profile and preferences. After every listened track, the algorithm can react in real-time to user feedback (skipped, liked, disliked or simply fully listened) and must find the next track that optimizes user experience but at the same time it must ensure the trajectory on the graph to arrive to its destination. The purpose of this dissertation is to design and build an algorithm to provide this specific service.

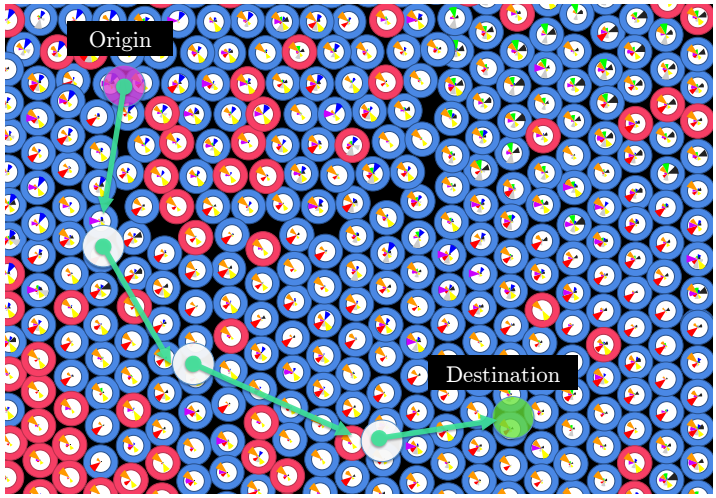


Figure 1.1: Diagram that depicts a journey in a portion of the music universe. Tracks in red are classic UK hits and tracks in blue are Pop UK hits. The tracks have been organized using the similarity algorithm proposed in this dissertation and applying a Self Organization Map technique to project in two dimensions. Inside every track there is an estimation of the *colour* of each track. Live demo is available at https://tunehog.s3.amazonaws.com/cuban.html.

1.3 The solution

The main idea of our solution is to present raw music similarity as a set of layered algorithms of increased specificity. The idea mimics the architecture found in the immune system. For a track to be considered similar to a given seed track it must pass through difference similarity algorithm layers. The top layers are able to narrow down millions of tracks into few thousands with little computing burden. As we move down, the time it takes to examine if a candidate passes to the next layer increases dramatically as the accuracy of the decision.

All tracks are analyzed to obtain an extensive list of features and track descriptions. The aim is to transform the original 4MB MP3 file into something smaller that preserves the necessary information to ensure correct similarity measurements. We have carried out several experiments to demonstrate that it is possible to reduce the full track to a small dataset of 10 seconds of music at very low sampling rate (4kHz) and very low quality encoding (Vorbis at hacked to go at 8kbps) and it is enough for a human to produce highly accurate similarity judgments. A person needs only a second of music to estimate the genre (with an accuracy of 95%). The key is transforming information into knowledge and the first step is reducing the 4MB into a 200kB file.

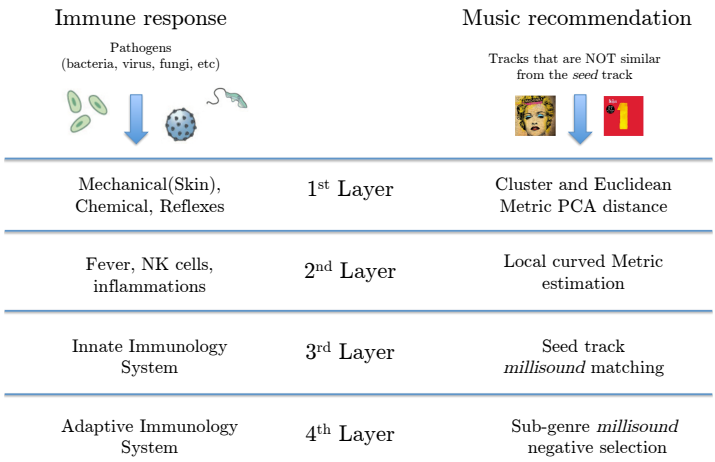


Figure 1.2: Diagram of the proposed solution based on the joint use of descriptor based metrics and an artificial immunology algorithm.

The *first layer* is based on clustering on PCA descriptors that are extracted from the quality-reduced track in order to increase robustness against different audio compression algorithms and sampling frequencies. It behaves as our mechanical self-defense layer. Only the tracks contained in clusters near the seed track are accepted to pass to the next layer, contributing to reduce future calculation requirements. All the calculations are based on the pair track-to-seed and therefore parallel computing approach is natural and simple map reduce techniques can be applied to execute this approach against collections of 100M tracks.

The *second layer* uses feedback from the user and also from previous users for that selected cluster and zone of the music *universe*. The algorithm selects a list of pairs of similar and different tracks that are closed to the seed track using the distance and clusters from the previous layer. Then it applies recursively a metric learning algorithm to minimize the recommendation errors on the pairs of tracks for that particular seed track. At the end it produces a linear sort kernel that describes how two tracks are compared against the seed one to select which of them is most similar to the track. Triangle similarity is introduced for the first time in this phase.

From the *third layer*, the artificial adaptive immune system starts. This system evolved in early vertebrates and allows for a stronger immune response as well as immunological memory, where each pathogen is *remembered* by a signature antigen. The adaptive immune response is antigen-specific and requires the recognition of specific *non-self* antigens during a process called antigen presentation. Antigen specificity allows the generation of responses that are tailored

to specific pathogens or pathogen-infected cells. The ability to mount these tailored responses is maintained in the body by *memory cells*. If a pathogen infects the body more than once, these specific memory cells are used to quickly eliminate it. From an music similarity point of view, all tracks are divided into 50ms millisounds. Each of these millisounds of the seed track acts a possible pathogen and is compared against all the millisounds of the candidate similar tracks that passed the second layer. The goal is to find candidate tracks with very little millisound matches so they are removed from the current list of similar tracks. Given for example a list of 1000 candidate tracks from the 50M original tracks, it is very likely that it contains 100 (a 10%) very similar tracks. However, they are not in the top of the list and the problem is to push good matches by removing false positives.

Finally, the *fourth layer* which is the biggest innovation of this dissertation, basically tries to use millisounds as antigens of negative selection from a huge library of millisounds antigens that has been extracted by analyzing tracks from different genres and subgenres. This library is applied to the seed track itself. The millisound antigens that match are only used to estimate the genre/subgenre but are discarded and only the ones that do not match the seed's millisounds are kept. A *non-self* prototype containing all the non-matched antigens is therefore applied to the candidates to help keep the number of false positives as low as possible.

The four layers can be applied to the radio-trajectory example but also to help promote similar tracks from the long-tail to the current radio experience. To achieve real-time usability of this technology, GPU based calculations are required so the last three layers occur mainly in the graphic card. The radio algorithm can build a set of positive and negative antigens that are used to fine-tune the selection of the *right* next track to be played. For a given user, a decision tree of several tracks ahead can be always be precalculated while the user is listening to the current track so the next track is available without a computer calculation delay.

1.4 Summary of contributions

The main contributions of this Thesis are:

1. A novel user-centric music similarity algorithm based on Riemann's curved metric estimation with hyperbolic control
   (a) Context-based similarity based on Content-based analysis.
   (b) It estimates the metric for a region of the music universe and it is context-based (seed track).

   (c) The algorithm is trained in real time using pairs of similar and different tracks that are compared locally.

   (d) The limits of the curvature are based on a test of hyperbolic of the metric.

2. Artificial Immune system based on layers and millisound non-self prototype creation.

   (a) The system encapsulates four different layers and each layer uses orthogonal algorithm to determine if a track passes to the next layer.

   (b) The layers mimic biological innate and adaptive immune systems. A *non-self* prototype is built based on 50ms segments called millisounds.

   (c) Millisound antigen matching is used to flag false positive and promote good recommendations to the top of the list.

3. A radio playlist algorithm based on trajectories in the music graph using the four layers in real-time proposed in this work applied to an incomplete long tail distribution.

   (a) Expand the popularity values to nearby tracks using a novel approach that maintains a long tail distributed on the predicted data.

   (b) A novel heuristic that selects the best path through the music collection between two tracks.

   (c) Usage of artificial immune system music similarity to real-time playlist generation.

1.5 Thesis outline

This Thesis is structured as follows: chapter 2 introduces the basics of music recommendation based on signal processing, the difference between a track and a song, pitch, timbre, rhythm and also some reference to the ear audition model. It also describes the core of the presented feature extraction algorithm and the raw similarity benchmarks that it achieves. Then, chapter 3 summarizes the state of the art about music recommendation, with detailed study of the evolution from 2000 to present time with special focus of the most interesting research lines. Chapter 4 introduces the Riemann Metrics that are used for the context-content based analysis. The metric estimation process is described in detail together with hyperbolic iterative limit and the improvements achieved in raw similarity are presented. Chapter 5 explains the Immune Artificial System for Music Recommendation and how the different layers of methods are used to transform a collection of 50M tracks into a set of few very similar tracks. Finally chapter 6 compiles all the different techniques together in a working novel Radio

algorithm that produces a playlist with a smooth similarity-based transition between two tracks. Finally, chapter 7 draws some conclusions and discusses open issues and future work.

Chapter 2

Music recommendation by signal processing

2.1 What is Music?

There are several levels to define music, starting from the dictionaries definition, music is often describe as an 'art form', as we can see in some of the below definitions:

- *an art form consisting of sequences of sound in time, especially tones of definite pitch organized melodically, harmonically, rhythmically and according to tone colour* [Collins dictionary]
- *such an art form characteristic of a particular people, culture, or tradition* [Collins dictionary]
- *vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion: the art or science of composing or performing music, a sound perceived as pleasingly harmonious* [Oxford dictionaries]
- *a pattern of sounds made by musical instruments, singing or computers, or a combination of these, intended to give pleasure to people listening to it* [Cambridge dictionaries]

As any art, music expresses particular ideas, feelings, and is intended to produce beauty of form, a pleasant and harmonious sound, but although music

is universal, all those characteristics are relative and subjective, i.e. usually based on culture. So, some music that can be beautiful and pleasant for Western culture, could be not such good for other cultures. More than that, what may be music to one may not be so to another. That fact has been a problem for music recommendation during last decades. With the globalization of the world, and specially since internet era started, people can access to huge amount of music from any part of the world, but not all music fits any culture, and there is no expert people enough to categorize all music.

Fortunately, although music is an art form, it can be also defined and simplified breaking it down into basic elements. So then we can redefine music as *a combination of melody, harmony and rhythm created with a varying degree of loudness and timbre.*

All sounds have pitch, duration, loudness and timbre. Melody, harmony and rhythm are combinations of those elements. When one organizes melody, harmony, Ã¥nd rhythm, and adds varying degrees of loudness and tone qualities, that person is creating music. Some of the music elements are specific to certain type of musical genres, while others are not. But all those elements have allowed people to categorize music and, somehow, find some automated ways to find similarities between two different songs, so let's see some brief definition of all those elements.

Pitch is a single tone, a note. It corresponds to the absolute frequency assigned to a specific note.

Loudness is the sound level, the intensity or volume of that sound.

Timbre or tone quality is the color or specific type of tone color of a sound. It is the specific sound that a certain instrument produces, and it is determined by the unique blend of frequencies of vibrations that compose a particular sound. The same pitch (for instance, the C note) does not have the same timbre if it is played by a piano or by a cello. That is why sometimes timbre is called the tone color of a musical instrument.

A sequence of pitches, a succession of notes, make a distinctive sequence of sound that is known as *Melody*. Melody is often described as the horizontal representation of the structure of a piece of music. It can be conjunct, with smooth sounding and easy to play, or disjunctive, with large pitch jumps and harder to play. Quoting Catherine Schmidt-Jones, from her book *The Basic Elements of Music* [Schmidt09b]: *the melody of a piece of music isn't just any string of notes. It's the notes the catch your ear as you listen; the line that sounds most important is the melody.* The melody can get involved one or more instruments, playing all at once or alternatively.

Harmony is what you get when you have more than one pitch sounding at the same time. It is one of the basic elements of music, although not always present. There are some pieces of music that are only rhythm, with no pitches at all, and some others with a single melody, or just melody with rhythm accompaniment, but if you have more than one pitch sounding at the same time, you have harmony. If you see melody as the horizontal movement of notes, harmony would be the vertical stacking of them.

Rhythm is the element of *time* in music. The way that sounds are placed in time creates the rhythm of music. That way we can talk about the basic rhythm of music, the structural pulse of the music, or about a rhythmic pattern that is repeated throughout the music. In any of those cases, we can take into consideration the specific beat, duration, and tempo of a song to determine the rhythm, and there are even further and more complex aspects under rhythm like syncopation (putting accents *of-the-beat*, when accents fall on weak beats) and tempo changes.

Rhythms are composed by a series of *beats*. A beat is a pulse of sound that is louder that other sounds in the music. Beats are usually organized in groups and form patterns. They are characteristic to drums and other low-frequency instruments like bass guitar for pop and rock music.

Tempo refers to the speed of the beat, and can be measured or described by the number of beats played in a minute (*beats per minute or BPM*) indicating the song's speed. High tempo or BMP values means more beats per minute and faster tracks, while a lower value will result in slower tracks. Tempo is one of main characteristics of a musical genre, as each genre usually has a corresponding tempo range.

As we can see, music can be an extremely complex subject, and at the same time a classic form of creative art that is part of all human cultures.

Why is music so important for all human cultures?

Music combines multiple elements to obtain a result that is pleasant to hear, and that it has deep effect on people physically, spiritually and emotionally. Music causes a mental response. It can cause a range of emotions, from euphoria to deepest sorrow. And music can have effects on the body too, heart rate and skin conductance may change when we listen to some song.

Music has strong connections to the emotional areas of the brain, and it is processed in the brain by interconnected areas. We still do not know a lot about those brain areas, but any damage to them can selectively remove specific aspects of musical appreciation. Although science is still researching about connections between music and brain, the fact that music can trigger powerful

physiological responses is being used for music composers for centuries. And more recently, music is being used to manipulate people's state of mind in several ways, like in Hollywood soundtracks, in business as part of a brand like for example soundtracks for shops, or even to modify human behavior, change our mood and have influence in our purchases, in how we exercise in the gym or to increase the alcohol consumption at clubs.

A century ago, most popular music passed from generation to generation by oral tradition, some music was written to allow people to discover and play it. Only a few decades ago, music was recorded in vinyl. This allowed people to listen to performances and discover new music. As time goes by, better recording ways appeared (cassette, CD), but we still needed experts in music that had to listen to all music to decide which one could fit the taste of some specific users. And then the digital music arrived, and Internet. And the music business and the music fan behavior changed forever. Everyone can access to huge amount of data instantly. The digital music databases are commonly composed by millions of songs. Catalogues cross-cultural, with lots of different musical genres and languages. How could be possible to use the majority part of that huge amount of data? How could we avoid the long tail effect? How can we discover new music among millions and millions of songs without the need of listening to all that music? That is where music recommendation by signal processing has revolutionized the music business and the way that users listen to and discover new music.

2.2 Tracks and songs

Most people use the term *song* to refer to musical works, although that is not always true. Correctly speaking, any musical work that has been created is a *musical piece*. And a song is, in fact, a musical piece, but not all musical pieces are songs. To be a *song*, the musical work must be relatively short and, usually, must have a component of singing. So, for example, an opera would not be considered a song, although it can contain several songs.

For the purpose of music recommendation by signal processing we must take any musical piece into account, so for avoiding misunderstandings we work with *tracks*. The term track came from the old ways of recording music, such as in vinyl and cassettes, but in general it refers to any musical piece on a record, cassette or CD, i.e. it takes into accounts short and long musical pieces, as well as with or without component singing.

Tracks can be digitized in a broad set of formats: WAV, MP3, OGG, and so on. How to decide which is the best audio format to use is a matter of finding the balance between audio quality, minimal lost in audio characteristics and size

of files.

During the research of this thesis, the author has used several types of audio formats, depending on the audio characteristics required at every step of the project. So at the beginning it was decided to use the best format to avoid any lost of data in the signal process, basically the PCM WAVE format to digitize music and as a source format for extracting the musical parameters and characteristics as a base for this dissertation. That ended with the first version of audio descriptors, called ANALYST1.0, that was based on WAV files, with $44.1Hz$ and Stereo (further details in section 2.7). Although at that moment it seemed to be the best option, actually was an issue every now and then for storage purpose, an average size of WAV file for a common musical piece would be around 30-40MB, while some other format like MP3 would use 10 times less space. But that was not the only issue, the time for analyzing tracks and the available sources for obtaining the high quality format of any music content were also a problem. For all those reasons, the original approach was changed and the author started further researching about audio descriptors based on MP3 audio formats (see further information in section 2.7 and subsection 2.7.3).

2.3 Pitch and Timbre

As it has been define in a very brief way in the first section 2.1 of this chapter, *pitch* is a single tone, a note, and it corresponds to the absolute frequency assigned to a specific note, the frequency of the fundamental component in the sound, i.e. the frequency with which the waveform repeats itself. It is a perceptual attribute that allows the ordering of sounds on a frequency-related scale extending from low to high.

Pitches can be described as high or low, based on the frequency of a sound wave, the number of vibrations produces during a given time period. The faster the frequency of vibration, the higher the pitch is. The slower the frequency, the lower the pitch. Frequency is measured in cycles per second or Hertz (Hz). If a sound vibrates at 200 vibrations per second, it has a frequency of 200 Hertz.

Standard Western music notation uses quantized logarithmic scale for arranging notes. Every octave range has 12 notes that are lettered as C, $C\#$, D, $D\#$, E, F, $F\#$, G, $G\#$, A, $A\#$, B, while the octave is indicated with a number following the lettered note (i.e. $C1$, $A4$, etc.). Each note has a nominal fundamental frequency than can be calculated as 440Hz $\times \frac{2^n}{12}$, where $440Hz$ is an agreed-upon anchor point for the tuning (it corresponds with note $A4$), and n varies from -48 to 39 on a standard piano keyboard.

So, everything is about frequencies. For a pure tone there is only one fre-

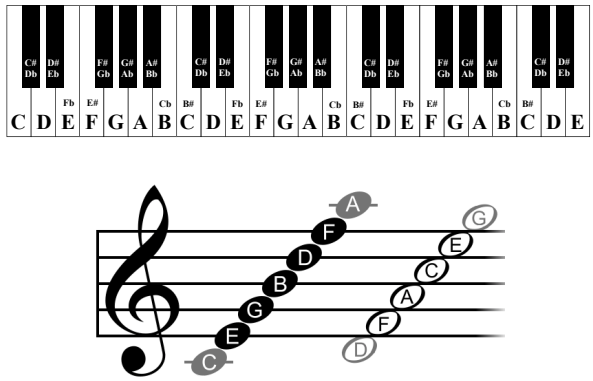


Figure 2.1: The piano keyboard diagram below shows the various piano notes C, D, E, F, G, A and B. There's a treble clef, a bass clef and a grand staff. Notice that the *C* in the treble clef and the *C* in the bass clef are the same note. This is the point where they cross and meet on the Grand Staff. It is known as Middle C.

quency involved but for a complex sound, like in most musical pieces, there are several frequencies involved and the pitch salience depends on the degree to which partials are harmonic. For example, for a harmonic sound the pitch depends on the frequency of the fundamental, but for a complex tones (such a carillon bells) the pitch depends on the frequencies of certain partials and is less highlighted than for harmonic tones.

It is also important to notice that the pitch of a pure tone can be influenced by changing its intensity, duration or even the amount of masking noise, so these are also elements that should be taken into account for music analysis.

It seems to be obvious that if every musical piece is made by a set of pitches, we should consider that element as one of the based ones for music analysis, especially for music recommendation. That is not totally correct. We cannot compare all pitches between two different songs, but we can take into account some range and others concepts based on pitch, like, for instance, brightness (mean frequency or center of mass of all the frequencies) or bandwidth (the spread of frequencies around the mean frequency).

Regarding *timbre*, that is the attribute of sound that allows humans to distinguish one voice or musical instrument from another. The sound of particular musical instruments or a combination of them can be recognized thanks to the perceived timbre. That is the reason why timbre is also known as *tone color*, and also why it can be described in a range from dull to brilliant, from cold to warm, from pure to rich and even from dark to bright.

Each musical instrument or voice produces its own characteristic sound pat-

terns and resultant overtones that gives to the resultant sound a unique tone color or timbre. That is the reason why the same pitch played in two different instruments sounds in different way.

Electric guitar will produce tones which are brilliant and piercing with its upper register, but will produce a rich and dark timbre with lower register. And you can also create a different timbre combining instruments and/or voices. All those details are usually used for composers for creating a special atmosphere with their music.

Musical genres can be also distinguished for special timbre because the musical instruments they use. So, electric guitar and electric bass are usually used in Rock, while an acoustic 12-string guitar is mainly used in Folk or Country Rock, or trumpets and saxophones are more common in Soul.

But what is what makes timbre so unique for each musical instrument? Timbre is a multifaceted attribute that invokes both spectral and temporal sound features. It is determined by the unique blend of frequencies of vibrations that compose a particular sound.

The vibration of sound waves can be complex. Most sounds vibrate at several frequencies at the same time; the additional frequencies are called harmonics. The relative strength of these harmonics helps determine the timbre of a sound.

Hearing is based on the amplitude of the frequencies, and it is insensitive to their phase. The image below shows two time domain waveforms very different that sound identical to human ear. The reason is that they have the same amplitude of frequencies.

The figure 2.3 shows, in the left the waveform displayed on an oscilloscope from the sound of a violin playing the note A below the middle C. The right side of the figure shows how that sound is perceived by human ear: a frequency of 220 Hz, plus harmonics at 440, 660, 880 Hz, etc.

The same note played by another instrument would have a different waveform, however, the ear would hear the same frequency of 220 Hz plus the harmonics, although the amplitude of the harmonics would be different. Since the two instrument would produce the same fundamental frequency for this note, they would sound similar and we would have the identical pitch. But, thanks to the different amplitude of the harmonics, they will not sound identical and we would say they have different *timbre*.

So, we can summarize that the shape of the waveform determines harmonic content, and the perception of the timbre is based on the ear detecting harmonics.

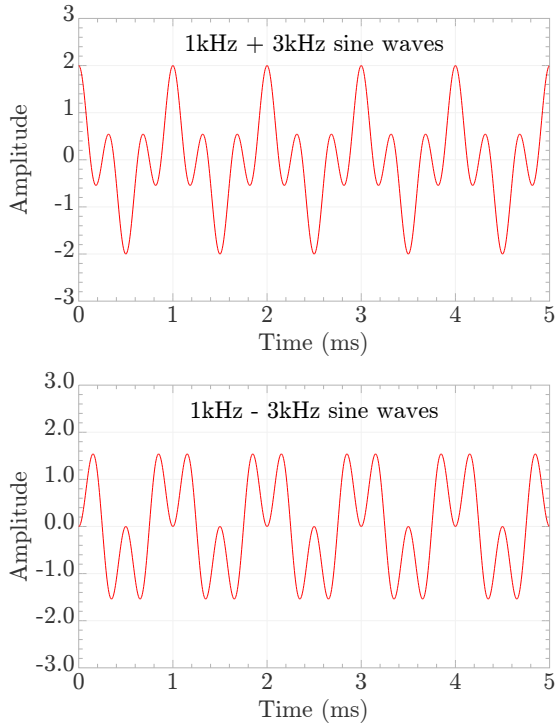


Figure 2.2: Phase detection of the human ear. The human is very insensitive to the relative phase of the component sinusoid. For example, these two waveforms would sound identical, because the amplitudes of their components are the same, even though their relative phases are different.

2.4 Rhythm, Loudness and Harmony

2.4.1 Rhythm

As I already mentioned earlier, Rhythm is the element of *time* in music.

Music moves through time with various levels of duration. The different changes in the duration of pitches create a particular arrangement on note lengths in a piece of music that is what we call rhythm. The rhythm can be regular or irregular, simple or complex, floating or driving.

Rhythm can be defined by its metrics *beat, tempo* and duration. But can also take into account more complex aspects to be defined like syncopation, or tempo changes.

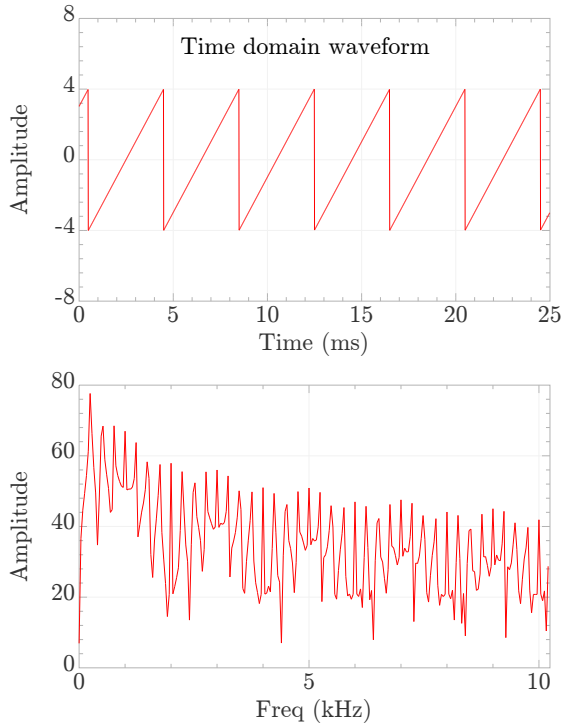


Figure 2.3: Violin waveform. A bowed violin produces a saw tooth waveform, as illustrated in (top). The sound heard by the ear is shown in (bottom), the fundamental frequency is the highest peak. The other peaks are the harmonics.

Beats is a pulse of sound that is louder than other sounds in the music, and it gives music its regular rhythmic pattern. We have a natural tendency to tap our feet or hands to the music, when we do that we are following the structural rhythmic pulse of the music, we are keeping the beat of the song.

Beats are usually organized in groups and form patterns, measures. The notes and rests correspond to a certain number of beats. So we can use beats for defining the meter of a piece of music.

Meter refers to rhythmic patterns produced by grouping together strong and weak beats, and it can be duple (2 beats in a measure), triple (3 beats in a measure), quadruple (4 bits in a measure) and so on. When beats are organized into recurring accent patterns, we have as a results a recognizable meter. The most common meters are diagrammed in Fig. 2.4.

But a piece of music can also be *Non-metric*, without regular groupings or beat patterns, although it is not very common. Or it can also have more than

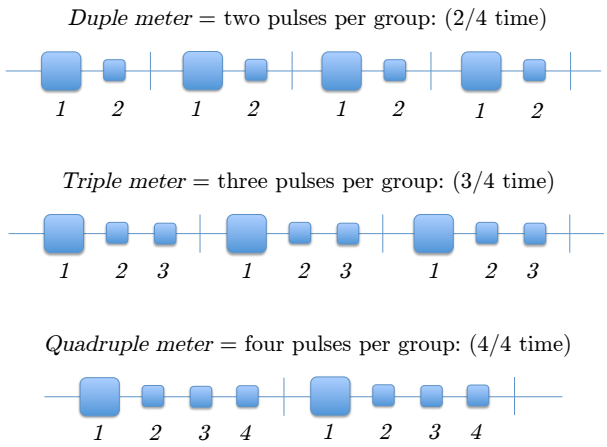


Figure 2.4: Diagrams of most common meters

one independent rhythm or meter happening simultaneously (what is known as *Poly-rhythm*). And it can even use accents falling on weak beats, putting accents *off-the-beat*, between the counted numbers (what is known as *Syncopation*).

Tempo is the speed of the beat, i.e. the speed of the song. It is measured by number of beats played in a minute (Bits Per Minute or BPM). The tempo can be fast, slow, speeding up (Accelerando) or slowing down (Ritardando).

In a stave, the tempo is shown at the beginning with the corresponding Italian word to indicate how slowly or fast the piece should be played (examples: Adagio, Andante, Moderato, etc.). Unless the composer indicates otherwise, the tempo is effective throughout the duration of the music.

Tempo is one of main characteristics of a musical genre, as each genre usually has a corresponding tempo range. The below chart shows tempos from several different Rock & Roll Styles:

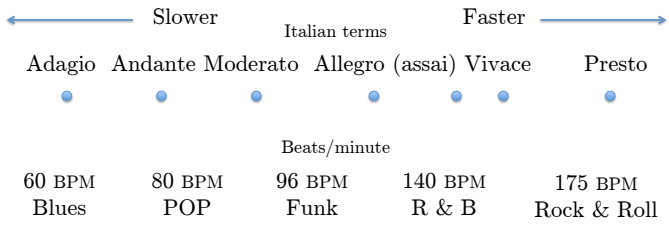


Figure 2.5: Italian terms for describing dynamic levels.

2.4.2 Loudness

Loudness is a measure of the sound wave intensity. It is the amount or level of sound (the amplitude of the sound wave) that we hear. In other words: the volume of the sound.

The degree of loudness or softness of a piece of music, and also changes on them (changes in volume), are called dynamics. They often reflect the intensity.

In classical music the terms used to describe dynamic levels are often in Italian: pianissimo (pp) for very quiet, piano (p) for quiet, mezzo-piano (mp) for moderately quiet, mezzo-forte (mf) for moderately loud, forte (f) for loud and fortissimo (ff) for very loud.

Sound level is often measured in decibels (dB). There is also a characteristic dynamic for some music styles. The dynamic range of an orchestra can exceed $100dB$. If the softest passage is $0dB$, the loudest passages can exceed $100dB$. In electronic equipment the lower limit is the noise level (hissing sound) and the upper limit is reached when distortion occurs. The dynamic range of electronic equipment is also called signal-to-noise ratio.

2.4.3 Harmony

Harmony is what we heard when a combination of notes (or chords) played together. But for better understanding of harmony we should first define what a *melody* is.

In music, melody is the sequence of pitches that usually acts as a theme in a piece of music. It is the tune of a song, something easily recognized and remembered (the tune of some famous songs that you sing or even whistle; that is the melody of the song).

Melodies are built around musical scales, and often repeat throughout a piece of music. They are generally described by its contour as conjunct, which means smooth sounding (easy to sing or play, mostly stepwise), or disjunctive, which means that the melody has large pitch jumps and is harder to play.

We can see melody as the element that focuses on the *horizontal* presentation of pitch, since it is a linear series or sequences of pitches with differing durations moving *horizontally* (one after the other) through time.

Coming back to harmony definition, if melody is the horizontal presentation of pitch, harmony is the *verticalization* of pitch. It is created by playing a group

of notes (simultaneously or as broken chords) behind the melody giving a musical texture. Harmony accompanies and supports the melody.

Pitches are heard simultaneously, and music is arranged vertically. Two simultaneous sounds are heard as an interval, three or more sounding together is a chord (several notes played simultaneously as a *block*).

Harmony is often described in terms of its relative harshness, like *dissonance* (harsh-sounding harmonic combination) and/or *consonance* (smooth-sounding harmonic combination). Dissonant chords produce musical *tension*, although dissonant and consonant chords are terms somewhat subjective.

2.5 Ear audition model

Music is sound, and how we perceive the sound is an important fact to take into account for the music processing.

The human auditory system is a complex multi-level pathway of sound information processing. It can be divided into three main and distinct levels: The preprocessing of the sound in the peripheral system, the conversion of sound signals from mechanical to neural representation, and the last level, the sound perception.

2.5.1 Preprocessing of sound in peripheral system

Sound is a sequence of waves of high and low pressure travelling through air. The perception of sound for humans is limited to frequencies between $20Hz$ to $20{,}000Hz$, although the upper limit decreases with age.

When we hear sound, we are translating movements of air molecules into electrical signals in the brain. This is made possible by the human ear that detects these movements and converts them into neural signals, and by nervous system that converts these signals into what we perceive as sound.

Let's see briefly how this process works, starting by the human ear anatomy description.

The human ear is composed by the *outer ear*, the *middle ear* and the *cochlea* (the inner ear).

The outer ear is composed by the pinna or auricle (the visible part of the

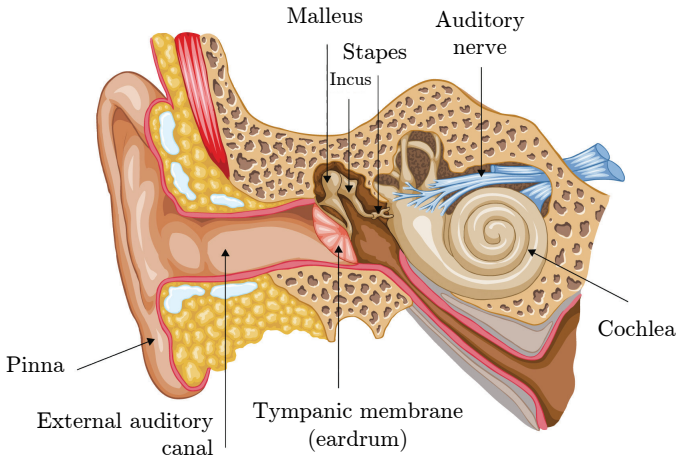


Figure 2.6: Human ear anatomy

ear), and the *ear canal*, a tube about 0,5 cm in diameter that extends about 3cm into the head. The pinna helps us to focus and localize sounds, and collects the sound waves that are transmitted to the middle ear through the ear canal.

The middle ear consists of the tympanic membrane (or eardrum), and a set of small bones (the malleus, the incus, and the stapes). The tympanic membrane is a thin sheet of tissue that vibrates when sound waves struck on it. That vibration is transferred and amplified to inner ear through malleus, incus and stapes.

In the inner ear, the cochlea converts the vibrations to neural impulses. This is the key step between sound and brain. The cochlea is a tube filled of liquid, with 2 mm in diameter and 3 cm in length, which is curled up and looks like a small snail shell. Sound vibrations are transmitted into the cochlea when the stapes moves the oval window. That window is covered by a flexible membrane, which allows the fluid in the cochlea to move back and forth in response to sound vibrations.

Lining the cochlear there is the basilar membrane. This membrane has different structure along the cochlea, so different parts of the cochlea responses to sounds of different pitches. At one end the membrane is narrow and stiff and vibrates in response to high pitches (high frequency detection). The other end is wider and more flexible, and responds to deeper sounds (low frequency detection). In the middle the medium frequency sounds are detected.

Attached to the basilar membrane are two types of tiny hair cells: the outer hair cells and the inner hair cells. These hair cells move when the fluid of the cochlea moves, detecting this way the vibration of the basilar membrane, and

converting that vibration into signals that are sent to the brain.

It is now well known that outer hair cells of the cochlea play an active role in increasing the sensitivity and dynamic range of ear.

Outer hair cells amplify the incoming sound signals, while inner hair cells convert mechanical stimulation to an electrical signal. The vibration opens ions channels, what triggers a series of cellular events that finally generates an action potential in the auditory nerve, sending messages or impulses through the cochlear nerve to the brain.

2.5.2 Mechanical to neural representation

Once the physical or mechanical part of the sound has been translated into nerve impulse, it still needs to travel from ear to brain. That last level of information flow involves as many as 30,000 nerve fibers from the inner ear to the brain.

The nerve impulse travels from first the auditory nerve to the ventral cochlear nucleus, then most signals are projected to the superior olivary complex (which is involved in determining the location of sounds), and then this area sends messages to another structure called the inferior colliculus, which is in the mid brain.

Finally, the information passes through the thalamus and is sent to the temporal lobe of the cerebral complex to decode the information sent from the ears and processing the more complex aspects of sounds.

2.5.3 Sound perception

The last level of the information flow is the sound perception, which could be seen as the psychological aspect or tone. It is the relation between the physics of sounds and their perception, and its study is known as psychoacoustics.

All this system allows us to hearing sounds and to detect several dimensions of them. We can know where the sound comes from, but also we can distinguish the loudness, the pitch and the timbre of that sound. The amplitude of the sound, the difference between high and low-pressure parts of the sound wave gives us the perception of loudness. The frequency of the sound gives us the pitch (the brain recognizes sounds of moderate to high frequency by the portion of the basilar membrane that is moving). And the timbre is represented by the complexity of the sound (number of simpler wave patterns that makes a complex sound).

But music perception produces activity in a much larger area than the ones involved into perceiving sounds. Listening to music produces activation throughout the bilateral fronto-temporal and parieto-temporal brain areas. While a person learns music, there are even more areas of the brain stimulated.

Finally, it is worth to mention that mathematical models exist at all levels of auditory processing. Although many available models are empirical, they often provide a useful framework for applications, such, for instance, the perceptual audio coding.

The perceptual audio coding is a technique for further compressing digital sound by removing frequencies that cannot be perceived by the human ear (frequencies that are too low or too high), or for removing soft sounds that are drowned out by loud sounds (for example, when several musical instruments are playing simultaneously, depending on frequency and volume at any given moment, the sounds from one instrument can cancel out the sounds of another).

MP3, a sound compression tool widely used for music, uses perceptual coding. It employs masking threshold curves in psychoacoustics to significantly reduce the bit width of digital sounds.

2.6 Identity and similarity

As we have seen in previous sections, musical pieces can be described using a very complex set of audio characteristics. Finding the perfect set of audio characteristics that can identify and categorize a musical piece is not a trivial matter. And we can ignore neither the way the music is perceived by human ear system, nor the way that every human being reacts and feels every musical piece. Music is art, and as any art expression, does not create the same effect in all persons. The culture, the musical education and knowledge and lots of other factors can affect the way that a person listens to music and how he perceives the similarities and differences between two musical pieces. All of us have listened some time to a song that has reminded us another song listened in the past, although both pieces were different. How can we identify those similarities? What does make two different songs sound similar although they are not using the same set of notes? That is what musical similarity tries to find out.

But, musical similarity can have different levels of similarity meaning. We can say two covers of the same song are the same song, so they are totally similar. Or, we can find two different songs with fragments that are the same, or total different songs that have some common values and patterns. The first level is what is known as Identity, while the others ones are known as Similarity.

Identity refers to music recognition, to find exact match between two songs, and even between two different versions or covers of the same song. The musical piece is the same, although it can be performed or recorded in different way. Identity is especially useful for mapping a new song to complete and huge music catalog in order to find duplicates in the catalogue, or for identifying a song for finding the corresponding metadata (i.e. album, artist, track name, etc.). There are several well known products that use music identity for helping music lovers to identify that song playing in the radio that they do not know or that they have forgotten the name, or for helping music distributors and aggregators for finding duplicates in their music catalogue. The algorithms behind identity usually use what is known as *acoustic* or *audio fingerprinting*.

In the same way that human fingerprint can identify in a unique way a person; the audio fingerprinting creates a unique representation for each song that is used for finding matching in order to recognize songs into music catalogue.

On the other hand, musical *Similarity* tries to find some common values and patterns between two or more musical pieces. There is none fixed and world wide approved set of parameters that we can use as unique set for finding similarities between songs, we can say that two songs are similar because they use similar melody, or because the instrumentation is the same, or the compositional style, or same timbre, or even the music style.

From human being perspective, we can identify different aspects of musical pieces while listening to them that can give us a way to differentiate them or to find similarities among them, the below list shows some of them:

- Basic characteristics of melody like character and range (smooth contour, jagged contour, wide range, narrow range).
- Basic elements of rhythm: we can identify pulse, tempo, downbeats, and meter (accent, syncopation, groupings of beats, fast or slow, etc.)
- Basic elements of harmony: chord changes and progressions.
- Degrees of loudness.
- Qualities of tone production: we can identify specific instruments; we can differentiate between vocal and instrumentation.
- Basic elements of musical structure and style: We can identify interaction of musical elements and recognize simple phrase patterns, contrast and repetition, clearly perceivable forms like verse-chorus.

But we can also find similarities thanks to less objective features such as mood, music genre, year or decade of the song, etc.

As we can see, musical similarity can be extremely complex to define, but since the arrival of digital music has exposed the music lovers to huge amount of musical pieces, the need for finding similarities among songs in order to help user finding content that he would like has been a priority for music business during last decade and the focus of thousands of researching. Most of those researching use an objective way to find musical similarity, and with that purpose they have created ways for extracting musical features that could be compared. That way we can compare and find objective musical similarities looking to pitch interval similarity, melodic similarity, modulation pattern similarity, timbre similarity, rhythmic pattern similarity, and so on.

2.7 Descriptor based similarity: First attempts

The first attempts of this work started long time ago, with the development of algorithms that could extract and parameterize different audio characteristics of tracks in order to later apply some algorithm to find similarities between songs.

This section is focused on the explanation of the analyst engine evolution; a set of algorithms to calculate audio characteristics of tracks in automated way.

2.7.1 ANALYST1.0

ANALYST1.0 was my first application for extracting audio characteristics of tracks, and I used the output of this application as the basics for my first approach for finding audio similarities.

The ANALYST1.0 engine used tracks with high audio quality format as input data (WAV format) and extracted 12 parameters mainly related to Timbre, Volume, Rhythm, but also others. The complete set of parameters is shown in the below list:

1. Timbre related:
    - Brightness Mean
    - Brightness Standard Deviation
    - Bandwidth Mean
    - Bandwidth Standard Deviation
    - Low Frequency Mean
    - Low Frequency Mean Rang

- Octave

2. Volume
3. Rhythm related:
   - Rhythm
   - Tempo
4. Noise
5. Size

The feature extraction algorithms used in ANALYST1.0 were based on frames analysis. The track was divided into small pieces called frames, where the size of the frame depended on a parameter. For each of the output parameters, each frame was analyzed and calculated the numerical representation of the feature or audio characteristic. That brought a string of numbers for each feature, which was summarized into two numbers (the output parameters) using statistical techniques such mean and variance. The mean indicated the most representative value for a parameter through all the song, and the variance shown the variations of that parameter.

Most features relied on the transformation of the signal from the sample space to the frequency space, and this was implemented as an FFT (Fast Fourier Transform).

A human perception filter model (sometimes an empirical model) was applied for normalizing the frequencies, that way instead of having a physical intensity for each frequency I used the perceptual intensities. The use and type of every model will be stated in the description of each algorithm parameters in the next sections.

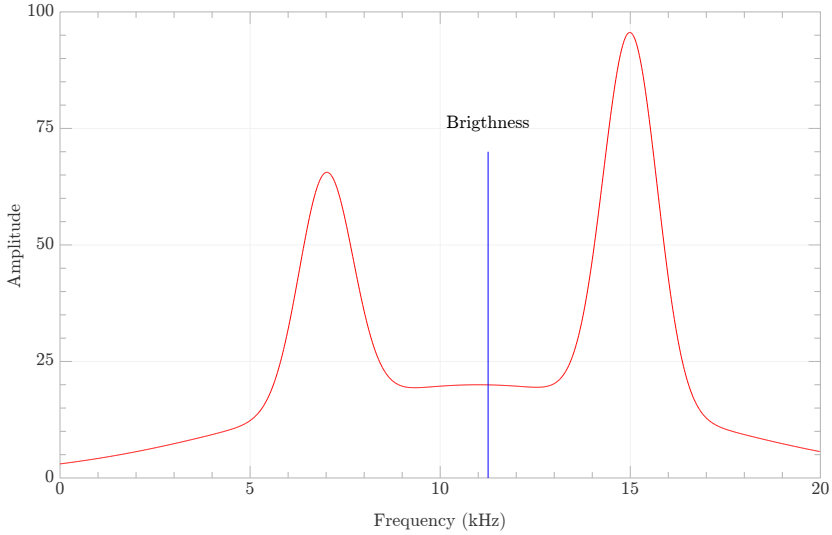
2.7.1.1 Timbre Analysis

As I explained in section 2.3, the timbre gives us the sense of which instrument (or instruments) is being played.

From an audio analysis perspective, the timbre can be thought as the main shape of the spectrum (the frequency axis). Based on this distribution, there were several parameters computed: Brightness, Bandwidth, Low Frequency and Octave.

Brightness

The Brightness of a sound gives a global idea of how frequencies are distributed. It represents the centroid (the center of gravity, the center of mass) of the energy or amplitude distributed across the spectrum.



The centroid can represent the frequency mean. Note that mean frequency is often different from main frequency; the main frequency corresponds to the most perceptual frequency in a sound, i.e. the tone.

From a quantitative point of view, the Brightness of a sound gives a global idea of how the frequencies are more or less distributed. From a qualitative perspective, the Brightness of a sound corresponds to the amount of clarity and sharpness.

Since what we are analyzing are mostly songs instead of sounds, extracting the Brightness from the FFT (Fast Fourier Transform) of the whole song would not give us remarkable information, since most songs would have same value. Also, introductory and final parts of a song could have a big effect in the final result. For avoiding those issues, instead of analyzing the whole song, I split up track in windows (frames) and calculate the Brightness frame by frame. That way we have a more local quantification of the parameter, even when later on we apply the mean and standard deviation to have global information of the whole song.

Finally, the perceptual model implemented for Brightness algorithm is based on ear filters theory, and it is a purely mathematical method (not empirical).

The output parameters based on Brightness are the mean (*Brightness mean*

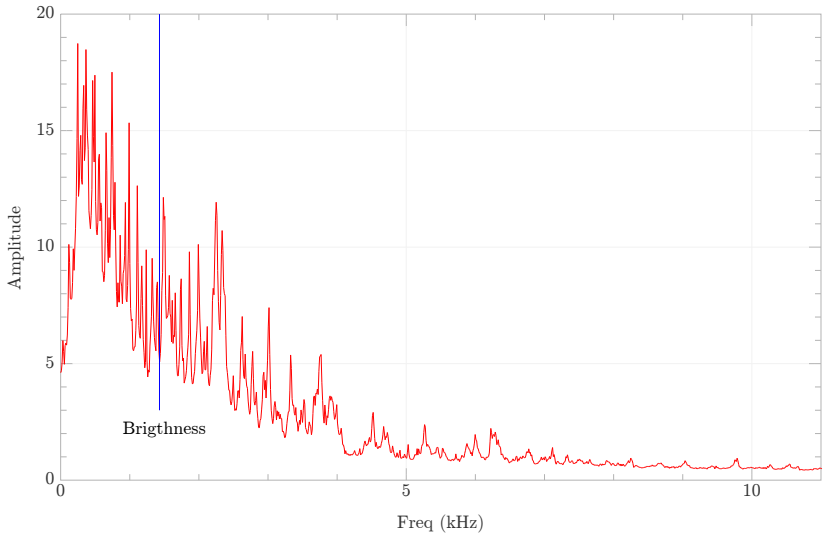


Figure 2.7: Red line represents Brightness

parameter) and the standard deviation (*Brightness standard deviation parameter*) over all the frames of the song.

Bandwidth

The Bandwidth represents the distribution of the spectrum shape across frequencies.

From a qualitative perspective, the Bandwidth of a sound corresponds to the richness; while from the quantitative perspective, the Bandwidth can be defined as the spread of frequencies around the mean frequency or centroid (around the Brightness).

The more frequency richness, the larger Bandwidth value we will have (i.e. larger green light in the image).

As in the Brightness algorithm, the Bandwidth is calculated frame by frame after applying FFT, and it is using a perceptual model based on ear filters theory.

The output parameters based on Bandwidth are the mean (*Bandwidth mean parameter*) and the standard deviation (*Bandwidth standard deviation parameter*) over all the frames of the song.

Low Frequency

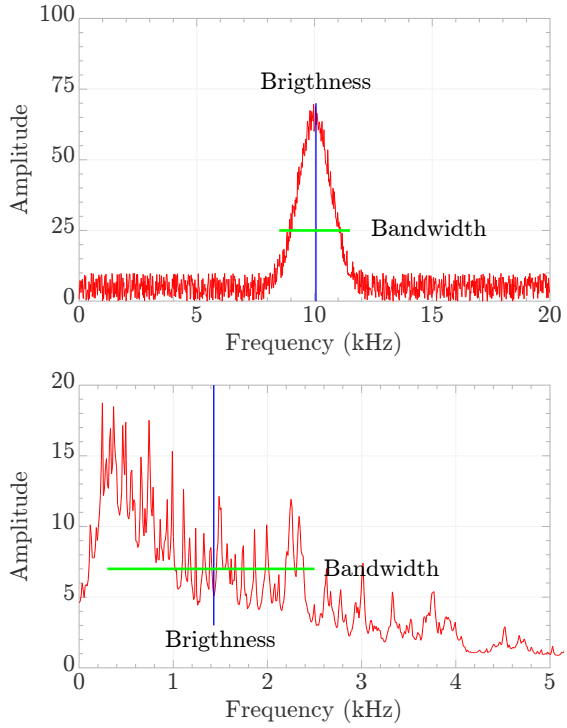


Figure 2.8: Green line represents Bandwidht

Low frequencies are the values close to zero in the horizontal axis of the spectrum (from $40Hz$ to $1KHz$). From qualitative perspective, the average amplitude of those low frequencies can give us an idea about the amount of bass, what it could be an important parameter for distinguishing music styles; for example, rock and rap are more *bass-heavy*, while classical and jazz are more *lighter*.

The perceptual model applied to input data is one of the most complex ones used in the ANALYST1.0 engine. One explanation is that human ear is more sensitive to high frequencies than to low ones, but music tends to use greater amplitudes in the low frequency range. That implies that in order to use the right frequencies that a human ear would hear, we need to attenuate the energy levels of the music before performing frequency analysis, otherwise, low notes would dominate the results and will be far away from human being reaction and point of view.

After performing an FFT to frames, we will have a vector of frequency amplitudes that we will normalize using the following function:

The tonal feature extractor determines frequency amplitudes by performing an FFT on 0.1 sec samples for 16 sec in the middle of the song. The frequency amplitudes A are then normalized using the function:

$$A'_\phi = \frac{A_\phi}{v(\phi)} \tag{2.1}$$

A is the where the normalization factor v is:

$$v(\phi) = \frac{43.066}{75.366 + \phi} + 0.015 \tag{2.2}$$

which is parametrized by ϕ, the frequency of the sample being normalized. v was determined by applying an inverse linear regression against amplitude-versus-frequency data averaged across 100 random songs. Another way to determine v would be to use established models of frequency sensitivity in the human ear. However, determining this function empirically allows us to adapt it to actual music characteristics, which often exhibit some compensation for recording and amplification technology.

The perceptual model implemented here is empirical, based on the study 100 songs stated in the technical article. This perceptual model is interesting for this application because it may take into account amplification, recording, encoding and other effects.

Finally, the two output parameters extracted for the Low Frequency are the below ones:

- Low Frequency Average: the average over all frames of energy in frequencies less than $1000Hz$.
- Low Frequency Average Range: the range over all frames of energy in frequencies less than $1000Hz$.

Octave

As I introduced in section 2.3, when explaining the timbre definition, we hear the fundamental and its harmonics of a sound. These harmonics are distributed across the spectrum representation as multiple frequencies. The parameter Octave gives us a number that describes the index of the most prominent harmonic.

For this audio characteristic, the output parameter is a unique value, the octave with most energy over all frames.

2.7.1.2 Volume

The variations in the volume level of a song can also bring us useful information.

In order to calculate the average change in the threshold value, I used the function described in [Welsh99] and shown in below figure:

$$volume(S) = \sum_{i=2}^{n} \frac{|S_i - S_{\text{i-1}}|}{(n-1)} \tag{2.3}$$

where S is a vector representing the raw sound samples

This function is not the definition of volume, but returns higher values for loud songs and lower ones for quiet ones.

For this case, there is no need of a perceptual model for the algorithm because we are working directly on the time domain.

The output parameter is a unique value; the Volume Mean.

2.7.1.3 Rhythm and Tempo

A rhythmic pulse can be described in terms of its frequency and phase component. The frequency of the pulse in a rhythmic musical signal is the tempo or rate of the rhythm, and the phase of the pulse indicates where the downbeat of the rhythm occurs.

The Rhythm and Tempo features extraction algorithms of the ANALYST1.0 engine are based in the algorithm described in the Scheirer article [Scheirer98], which breaks an input signal into several bands, each representing one-octave ranges, and find the downbeat over the time on each input using banks of resonators. The algorithm allows finding the average tempo of the song, how the tempo varies throughout the song and even the rhythmic complexity of the song.

The algorithm uses *frequency filter bank* for dividing the input signal into six bands (see figure 2.9). For each of these bands, the algorithm calculates the envelope (see figure 2.10) and applies the derivative. Then each of the envelope derivatives is passed on to another filter bank, this time of tuned resonators. In each of those resonator filter bank, the resonator with the resonant frequency that matches the rate of periodic modulation of the envelope derivative will have phase-lock. The outputs of the resonators are examined to see which ones have the phase-locked behavior, and this information is tabulated for each of the band pass channels. All tabulations of all frequency filter bank are summed to arrive to frequency (tempo) estimate for the signal, and use that information back to the peak phase points in the phase-locked resonators to determine the phase of the signal.

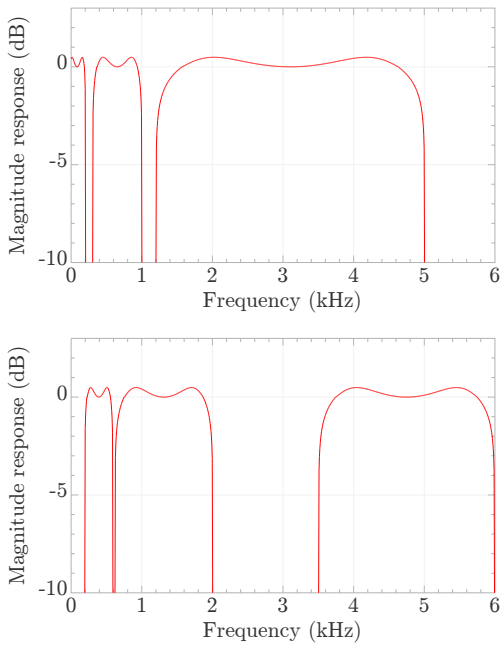


Figure 2.9: Frequency filter bank with filters 1-3-5 (top) and 2-4-6 (bottom)

Human listeners do not need too much broad frequency channels to rhythmically analyze a musical signal. So the first frequency filter bank of the algorithm is implemented using six bands; each band has sharp cutoffs and covers roughly a one-octave range. The lowest band in a low-pass filter with cutoff at $200Hz$. The next four bands are bandpass with cutoffs at 200 and $400Hz$, 400 and $800Hz$, 800 and $1600Hz$, and 1600 and $3200Hz$. The highest band is high pass with cutoff frequency at $3200Hz$.

After the envelope has been extracted and processed for each channel, another filter bank is applied to determine the tempo of the signal, this time a filter bank of comb filter resonators. A comb filter adds a delayed version of the signal to itself, they are often used in reverberations, but they also have properties, which make them suitable for acting as resonators in the phase-locking pulse extraction process. The comb filter with delay T will respond more strongly to a signal with period T than any other, since the response peaks in the filter line up with the frequency distribution of energy in the signal.

For each envelope channel of the frequency filter bank, a filter bank of comb filters is implemented. The output of these resonator filter banks is summed across frequency sub-bands. Examining the energy output from each resonance channel of the summed resonator filter banks, we can determine the strongest periodic component of the signal. The frequency of the resonator with the highest energy output is selected as the tempo of the signal.

All the steps of previous algorithm end with two final output parameters: the Tempo (as the average of the most phase-locked resonator) and the Rhythm (as the average of the second most phase-locked resonator).

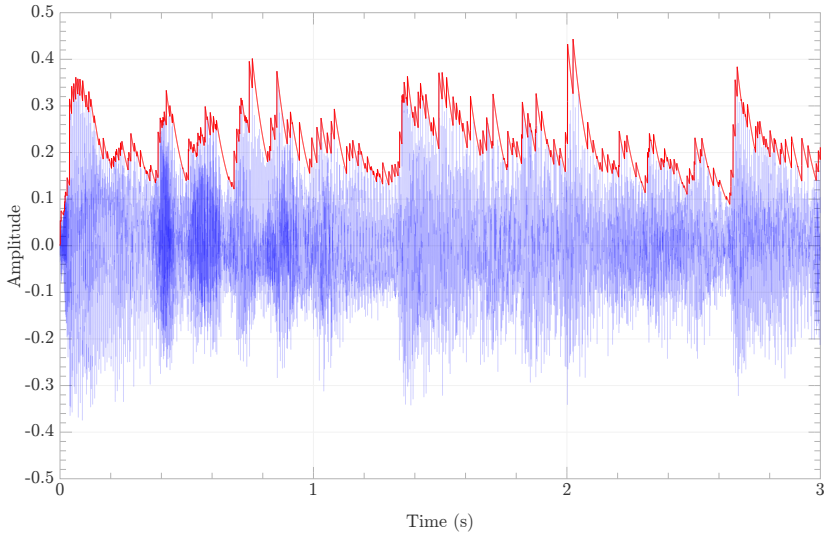


Figure 2.10: Original audio signal in blue (violin) and envelope in red. The envelope was calculating using $y(n) = max(x(n), y(n-1)) * 0.997$.

2.7.1.4 Noise

Although it can sound surprising at first view, the relative noise level in a musical piece can be also used for measuring how similar is that piece to another. If a human being compares a song with pure sound to another with lot of noise it is quite probably that he will not consider them similar.

The noise feature extraction algorithm has been also adopted from the Scheirer article [Scheirer98] and it is based in the Fourier transform.

The Fourier transform of a pure tone (i.e. a perfect sine wave) will have a highly regular structure, with a single spike at the corresponding frequency, while it will not have visible structure for random samples (i.e. highly noise). Noisy sounds will present large number of frequencies in the Fourier transform.

However, we can not count only the number of dominant frequencies because they are not always easy to detect; For example, a violin playing a single tone will have a Fourier transform with a peak at the frequency of the tone, as well as smaller peaks at several different nearby frequencies. Instead of that, algorithm counts peak frequencies by weighting each frequency by the magnitude of the corresponding Fourier coefficient and computes the sum of the magnitudes of the coefficients corresponding to each

dominant frequency (for example, one loud tone with two quieter ones with half the magnitude will count as a total of two peaks). That way, non-dominant frequencies will contribute much less than the dominant ones.

The algorithm does not use any threshold to select the dominant frequency, instead of, it adds up all the Fourier coefficients. At the end, the sum is normalized according to the maximum Fourier coefficient, that way the volume of recording will not affect the result (i.e. songs that have been recorded at a louder volume will not appear more noisy).

Since Fourier transform is taken over a small sample of a song, the output metric for this audio characteristic takes into account only the average noise level over the entire song. The maximum level and the standard deviation of the noise level across the song could be interesting but were not taken into account for this version of the application.

Although ANALYST1.0 engine got quite good results for finding similarities between songs, it was a first approach that shown its limits quite fast. At that moment Music Information Retrieval (MIR) community started to grow and the state-of-the-art algorithms progressed almost every month. In that context, I started several rounds of improvements of the engine for adding some of the latest state-of-the-art algorithms of MIR, as well as for moving to a more modular and portable version of the engine.

The following sections will briefly describe the intermediate versions before I developed the version 3.1 of the analyst engine.

2.7.2 ANALYST2.0

The first improvement was the *ANALYST2.0* engine, which was still using WAV files as input data, although it expanded that description range and added new aspects of the sound such as Harmony and Structure of the song.

The Timbre features were still based on summarizing the shape of the spectrum, but this version calculated new coefficients using the Mel-Frequency-Cepstral-Transform (MFCC) algorithm.

The Noise was also calculated in this next generation of the engine, but this time using a new algorithm called Harmonicity, which measures the degree of harmonics in each song.

The Rhythm features were totally changed to use a new approach based on onset detection, i.e. on the detection of the beginning of each note.

The Volume algorithm evolved into Dynamics module, where parameters related to music production techniques such as compression or silence were computed and added to output parameters.

The fluctuations or changes in amplitude of the spectrum across time were computed with the Spectral Fluctuations algorithm. The purpose was to have an idea of the loudness variations, the bass beat periods, the aggressiveness, etc.

A new aspect of this algorithm was the study and calculation of the Harmony, through a sense of the global Key of the song, as well as other Harmony estimators such as atonality degree, key changes, chord variation ratio, etc.

The other new aspect added to the algorithm was the Structure, which added new parameters for describing the global structure of the song (i.e. verse, chorus, bridge, etc.)

The output result of the engine was a vector of the 69 audio characteristics listed below:

- Timbre related:
  - Timbre related to spectral envelope using weighted statistics of Cepstral Transform: Mean and Variance of the first 13 MFCC
  - Timbre related to statistics on frequency spectrum based on weighted statistics of Spectral: Mean and Variance of the
    - Spectrum centroid
    - Spectrum bandwidth
    - Roll On factor
    - Roll Off factor
    - Spectrum flatness
    - Voice to white
- Noise related:
  - Harmonicity: Mean and Variance of the Noisiness.
- Rhythm related: the Onset Detection
  - Density (ratio)
  - Intensity (Mean)
  - Energy (ratio)
- Volume related: the Dynamics modules.
  - Silence ratio
  - Absolute volume (Mean)
  - Relative volume (flux) (Mean and Var)
  - Envelope flux (Mean and Var)
- Spectral Fluctuation (fluctuation patterns):
  - Max value (ratio)
  - Aggressiveness (ratio)

  - domain of low frequencies (ratio)
  - Gravity (fMod)
  - Spread of Focus

- Harmony
  - Main Key (sin and cos)
  - Chromaticity (Var)
  - Secondary Key distance (radians)
  - Chord complexity (ratio)
  - Chord changes (Density)
  - Far from key / Chord changes (ratio)
  - Chord Major/Minor (ratio)
  - Octave with more energy (ratio)

- Structure:
  - Overall similitude histogram (Centroid and BW)
  - Novelty peaks (Density)
  - Novelty peaks Period histogram (Centroid and BW)
  - Peak Position (Mean)

2.7.3 ANALYST3.0

The extra audio characters of the ANALYST2.0 version were a big step forward, but the limitation of the input format audio files were still an issue that also affects the performance of the engine. With that in mind, and after several feature selection experiments, the first version of the engine based on MP3 encoded data was developed, that was the *ANALYST3.0* engine.

The ANALYST3.0 engine speed up the process by analyzing directly MP3 encoded data, as well as added, removed and modified some features, according to progress in MIR researching and some feature selection experiments. Features that at the end become redundant or useless like Roll On factor, Roll Off factor, Spectrum Flatness, Voice to White ratio were removed, but also other features were replaced to make harmonic description more robust (harmony features were replaced by lower level harmonic features), and some extra ones were added like the Bass ratio.

Finally, after several testing rounds and fine tuning of the engine, I arrived to version *ANALYST3.1*, the most stable version and the one that is worth to mention and explain a bit farther.

2.7.4 ANALYST3.1

The ANALYST3.1 was implemented in C++ and it was the first multi-format version of the analyst. It is dependent on some FFMPEG libraries (libavcodec, libavformat and libavutil) to decode most of audio, but it has been developed to accept most popular audio formats as input files (like MP3, aac, wma, ogg, etc.), although it has been used mostly with MP3 audio tracks.

The algorithm stores the decoded samples in a circular buffer and analyzes them in streaming, what speeds up the execution because the analysis is performed as soon as enough data is decoded. The average processing time for analyzing a three minutes long song is around 5 seconds, versus the average of 20 seconds for analyzing songs with ANALYST2.0.

The algorithm analysis the input data performing frames with no overlap, windows each frame, applies and FFT and than computes several layers of algorithms to the result in order to extract all the audio features. Following is a brief description of the algorithms layers applied.

Mel Frequency Cepstrum Coefficients (MFCCs): after mapping the log amplitudes of the spectrum onto mel scale, a discrete cosine transform is applied to those mel log-amplitudes like if they were a signal. The output data are the mean and variance of the 13 first coefficients of the amplitudes of the resulting spectrum (i.e. 26 output values)

Basic spectral descriptors:

- RMS value. Also mean and variance of the value (2 output values)
- Spectral Centroid: It characterizes an audio spectrum and indicates mass center of the spectrum. Perceptually, it has strong connection with brightness of a sound. Mean and variance of the value as output data (2 output values).
- Spectral bandwidth: It is a measure of how tightly is the spectrum centered around the Spectral Centroid. Sine wav has value zero as Spectral Bandwidth, while white noise has large value. Also mean and variance of the value as output data (2 output values).

Pitch Class Profile (PCP): It measures the intensity of each of the twelve semitones of the tonal scale mapped to a single octave. The output values calculate the mean and variance of those twelve semitones (i.e. 24 output values)

Some extra long-term features are also computed and added to the output data, such as the onset-related features, the Spectral fluctuation patterns and the Bass ratio.

Onset-related features: For each onset found the following values are calculated:

- onset density: number of onsets per second
- onset intensity: mean of the max amplitude reached by every onset

- onset steepness: mean of the max log-amplitude derivative of every onset

Spectral fluctuation patterns: Based on the E. Pampalk's article [Pampalk06], it provides a measure of the amplitude modulation of the loudness per frequency band. In order to achieve that, the Fluctuation patterns (FP from now on) is computed by performing the FFT on the energy envelope of each twelve frequency bands. Every frequency spectrum obtained is weighted by a function modeling the perception of the fluctuation strength. Then, the following values are calculated:

- Gravity: It describes the gravity's center of the fluctuation patterns (FP) of the modulation frequency axis. Low values indicate that the piece might be perceived as slow.
- Focus: It describes the distribution of energy in the FP. In particular, the focus is low if the energy is focused in small regions of the FP, and high if the energy is spread out over the whole FP.
- Bass: the bass is calculated as the sum of the values in the two lowest frequency bands with a modulation frequency higher than $1Hz$
- Aggressiveness: the aggressiveness is measured as the ratio of the sum of values within the frequency bands 2-12 and modulation frequency below $1Hz$ compared to the sum of all. Less aggressive pieces have higher values.
- Max: the maximum fluctuation strength is the highest value in the rhythm pattern. Pieces of music which are dominated by strong beats, have very high values.

Bass ratio: computed as the bass -describe above- compared to the sum of all the frequency bands.

The final output of the ANALYST3.1 engine is a vector with 65 audio parameters values:

- Mean and variance of the (weighted) 13 first MFCCs. 26 values
- Mean and variance of the short-term RMS value. 2 values
- Mean and variance of the (weighted) spectral centroid. 2 values
- Mean and variance of the (weighted) spectral bandwidth. 2 values
- Mean and variance of the (weighted) PCPs. 24 values
- Onset density, intensity and steepness. 3 values
- Spectral fluctuation's aggressiveness, focus, gravity, max, bass and bass ratio. 6 values

2.8 Principal Component Analysis applied to descriptors

Principal Component Analysis (PCA) is one of the simplest and more robust methods to extract relevant information from confusing data sets. It is usually used for dimensionality reduction. The idea is to identify the dependencies and redundancy existing among variables in order to transform a high-dimensional data into a lower-dimensional form, with most meaningful data and without losing too much information. Working with lower dimension is always easier, but also offers a way to see the hidden and simplified dynamics that often underlie it, filtering out the noise.

This method is used in all form of analysis (from neuroscience to computer graphics) because it is simple and a non-parametric method. In the case of Music Information Retrieval (MIR), PCA has been used extensively, especially during first years, because it seemed to be the most logical way to find the hidden relationship among so many audio characteristics.

In the case of this thesis, I also tried this method at the beginning as a way to understand the clouded, unclear and redundant data extracted from the 12 audio characteristics of the ANALYST1.0, and also as a first step to plot the results into 2D and 3D graphics in order to find audio similarities among songs, as well as for visualizing the data.

PCA makes one powerful assumption: linearity. Linearity simplifies the problem in two ways: restricting the set of potential bases, and formalizing the implicit assumption of continuity in a data set.

Any set of n measurement types can be seen as a vector in $n-$dimensional space.

$$\overrightarrow{X} = \begin{bmatrix} x_1 x_2 \dots x_n \end{bmatrix} \tag{2.4}$$

The PCA tries to find another basis, which is a linear combination of the original basis, that best re-expresses the data set.

Let X and Y be $m \times n$ matrices related by a linear transformation P. X is the original data set and Y is the final re-representation of the data set.

$$PX = Y \tag{2.5}$$

Following equation represents a *change of basis*. P is a matrix that transforms X into Y, and Y is the improved data set without correlation.

$$PX = \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix} \begin{bmatrix} x_1 \dots x_n \end{bmatrix} \tag{2.6}$$

$$Y = \begin{bmatrix} p_1 \cdot x_1 \dots p_1 \cdot x_n \\ \vdots \ddots \vdots \\ p_m \cdot x_1 \dots p_m \cdot x_n \end{bmatrix} \tag{2.7}$$

We can note the form of each column of Y as:

$$y_i = \begin{bmatrix} p_1 \cdot x_i \\ \vdots \\ p_m \cdot x_i \end{bmatrix} \tag{2.8}$$

and we can understand that the j^{th} coefficient of y_i is a projection on to the j^{th} row of P. Or, in other words, the y_i is a projection on to the basis of $\{p_1, \dots, p_m\}$. Therefore, the *rows* of P are indeed a new set of basis vectors for representing of *columns* of X

The row vectors $\{p_1, \dots, p_m\}$ in this transformation will become the *principal components* of X

So, the most important part for PCA is finding the best *principal components* for X, or in other words, the best P. But, how can we make a good choice of basis P? Taking into account what is the goal of Y: improving the confusing data set of X

In a linear system, a confusing data set can refer to two factors: noise and redundancy.

In any data set the noise must be low, but it is quite impossible to reduce that factor to zero most of times, so we have to deal with it and minimize it as much as possible. A common measure of the noise is the Signal-to-Noise Ratio (SNR), also known as ratio of variances σ^2

$$SNR = \frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{noise}}} \tag{2.9}$$

High value of SNR ($>> 1$) indicates high precision data (low noise level), while a low SNR indicates noise contaminated data. If we plot two similar measurement types in 2D space, and if we would not have noise at all, all data would be plotted in the same straight diagonal line. The more noise we have, the thicker will be the line, looking like an oval. The SNR measures the *fat* of the oval, which achieve the maximum level when it looks like a perfect circle ($SNR = 1$)

In the case of the music database and their corresponding audio descriptors, the noise can be minimized centering and normalizing the data. With that purpose we need:

$$\bar{X} = 0 (\text{Mean Data} = 0) \tag{2.10}$$

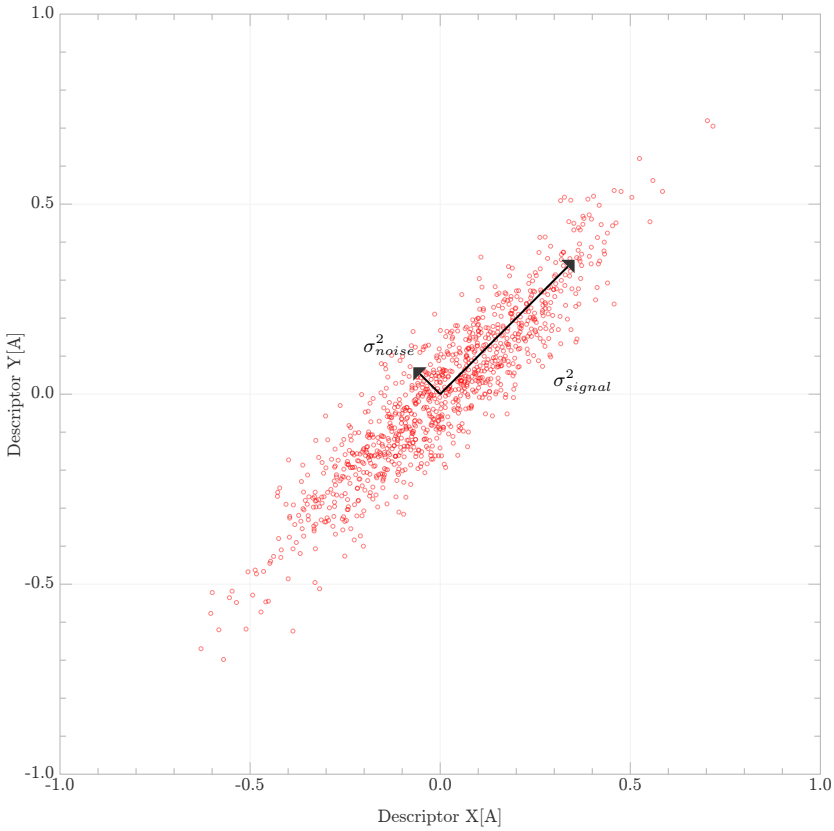


Figure 2.11: A simulated plot of (x_A, y_A). The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented.

$$\sigma_X = \sigma^2 = 1 \text{ (Standard Deviation = Variance = 1)} \tag{2.11}$$

For getting zero $\bar{X}$ (2.10) we have to calculate the mean values (across the entire track database) for each of the audio descriptor, and then centering individual parameters (of each track) using those means. For each descriptor of vector X the new value will be $x_i = x_i - \bar{x_i}$, that way if mean was recalculated with these new values, the mean value would be zero.

For (2.11) we calculate the σ and σ^2 of the normalized data set of previous step, and re-calculate again all values applying $\bar{X}/\sigma^2$ in order to get $\sigma_X = 1$ if it was recalculated using the new values.

On the other hand, redundancy is a more complicated issue, but redundant data is not offering further information to the system, so it is something to be avoided and minimized too. The redundancy can be measured by the correlation between two

variables. If we plot data from two different measurement types in a 2D space, the redundancy will be shown as both measurements plotting over same space of graph, while in a non redundant plot all data will be scatter among all graph space.

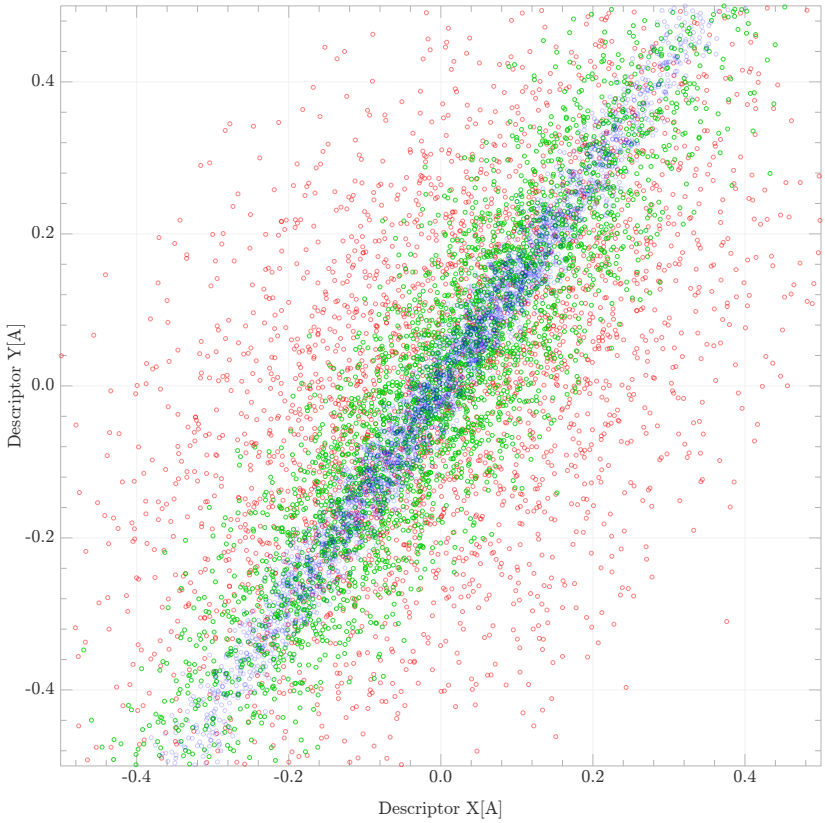


Figure 2.12: A spectrum of possible redundancies in data from the two separate recordings with $X/Y = 2$ (red), $X/Y = 5$ (green) and $X/Y = 15$ (blue). The best-fit line $r2 = kr_1$ for all cases has a slope of 55 degrees.

The figure 2.12 shows a range of possible plots between two arbitrary measurement types r_1 and r_2. Panel (a) represents two recordings with no redundancy, r_1 is entirely uncorrelated with r_2. While panel (c) shows two recordings strongly related, i.e. highly correlated, and with high redundancy. Clearly, in panel (c) it would be more useful to use the linear combination $r_2 - kr_1$ instead of two variables r_1 and r_2 separately. This is the main idea behind the dimensional reduction.

In our case, removing analysis data correlation from the centered and normalized data is mandatory in order to reduce redundant information given by different parameters and obtaining the most unique information from each one. To do this, the system creates a new set of parameters with no correlations, but with all the information included in the parameters analyzed by the analysis tool. These new parameters have no

conceptual meaning (i.e. they are referred to as $parameter_1, parameter_2$, etc; instead of $Volume$, $Tempo$, etc.).

Data correlations are determined using a standard correlation matrix. The standard correlation matrix formula is as follows:

$$V_{\mathrm{ij}} = \frac{1}{M} \sum_{c=1}^{M} Z_i(c) Z_j(c) \tag{2.12}$$

Where V_{ij} is the value of the intersection of column i and row j, M is the number of songs in the database, c is each particular song, Z_i and Z_j are parameter number i and j respectively for each song c.

In essence the correlation matrix multiplies, for each pair of parameters, all values for all registers of data, adding them, and then dividing by the total number of registers.

The correlation matrix is then used to calculate Eigenvectors and Eigenvalues. As the correlation matrix is symmetric, standard Jacobi transformations (of symmetric matrices) are used to compute all Eigenvectors and Eigenvalues. Eigenvectors and Eigenvalues are standard statistical formula used to determine and eliminate correlations between parameters, thereby reducing repeated information and increasing data utility.

Eigenvectors are the key in determining the new uncorrelated parameters from the original parameters, as illustrated in the following formula:

$$\Big[P_1 P_2 \dots P_N\Big] = \Big[O_1 O_2 \dots O_N\Big] \times \begin{bmatrix} E_1 1 E_1 2 \dots E_1 N \\ E_2 1 E_2 2 \dots E_2 N \\ \dots\dots\dots\dots \\ E_N 1 E_N 2 \dots E_N N \end{bmatrix} \tag{2.13}$$

Where P_i (where i goes from 1 to N) are the new parameter values for each song, N is the number of parameters, and also, the number of eigenvectors. O_i (where i goes from 1 to N) are the original parameter values for each song, and E_{ij} (where i and j go from 1 to N) are the values for parameter j of Eigenvector i

Eigenvalues give the amount of uncorrelated information contained in each parameter. Thus, these values are used to weight the importance of parameters (or even discard parameters that contain no or almost no information).

The parameters are now uncorrelated, normalized, orthogonal and weights have been set for each one, so the system can get maximum value from them. These new parameters will replace the old ones that came from the music analysis tool.

2.9 Descriptor clustering

After you have the descriptor vector with main audio information from every track, you need to find some useful pattern among all music database in order to extract relevant information that could help you to find similarities among tracks or some other kind of information. Clustering methods can help on that.

Clustering methods use large multi-dimensional datasets and find groups of objects with some similarity. Objects inside same group have some type of similarity, they are somehow related, while they are different from and unrelated to other groups of same dataset. That brings the user a reduction of the size of the input large database, as well as extracts new information from that database thanks to the new categorization of its elements.

A Clustering is a set of clusters, but depending on the clustering method used or on the type of multi-dimensional data the output number of clustering and even the type of the set can vary. So, we can have partition versus hierarchical clusters, exclusive versus non exclusive clusters, fuzzy versus non fuzzy clustering, partial versus complete, etc. For example, in a partition clustering the division of data is done into non overlapping subsets or clusters, so any object (audio descriptors in our case) can belong only to one cluster; While in a hierarchical clustering a set of nested clusters are organized as a hierarchical tree. On the other hand, in a non-exclusive clustering an object can belong to several clusters, for example for border line objects.

The clusters can be also classified based on the morphology and density of their objects. So we can have well separate clusters, where any object inside a cluster is closer or more similar to every other object of same cluster than to any other object not in the same cluster. Or we can have center-based clusters, where every cluster has a center and any object inside that cluster is closer to that center then to any other center of other cluster. The clusters can be also defined by density of points in a region, or for an objective function (clusters that maximize or minimize an objective function), and so on.

There are a bunch of clustering algorithms, but as Pang-Ning et al. mentioned in their book [IDM05] we could make a main classification with the following ones: Centroid-based clustering (K-means and its variants), Hierarchical clustering, Distribution-based clustering and Density-based clustering.

In the Centroid-based clustering algorithms, every cluster is associated with a *centroid* or center point, and every point or item can be in one cluster, so it is a partition clustering approach. The most well known algorithm of this type is the K-means algorithm and all its variants. K-means algorithm works quite well with standardize input data that creates clusters of same sizes, but it has problems when clusters are of different sizes, densities or even with non-globular shapes, and also it has problems with outliers, but overall, the main problem with most k-means algorithms is that the number of clusters (k) has to be specified in advance.

On the other hand, the Hierarchical clustering does not need to assume any par-

ticular number of clusters; it is a connectivity based clustering that creates a set of nested clusters organized as a hierarchical tree. The algorithm is based on the idea that items are more related to nearby items than to items farther away, so it connects closer items to create clusters. That output hierarchical tree can be represented by a dendrogram. It has better behavior than k-means for outliers and noise, but it shows some bias towards globular clusters. The different algorithms using this approach differ on the way that distances are computed, the linkage criterion and the way to start the algorithm (from single item to whole data set or the opposite).

The Distribution based clustering algorithms are related to statistics. Clusters can be defined as items belonging most likely to the same distribution. One of the main issues with that approach is the over-fitting (when statistical model describes a noise or random error instead of the real and main data), that can appear unless you use complex approaches to the algorithm. This algorithms use the Gaussian mixture models, what implies that it cannot be used with any kind of input data. Those algorithms can produce the output clusters but also a complex model for the clusters than can show the correlation and dependence of attributes.

Finally, Density-based clustering algorithms define a cluster based on density of items in a particular area. Therefore the high density areas will be the final cluster. The most well known algorithm of this category is the DBSCAN. It is an algorithm resistant to noise that can handle clusters of different sizes and shapes.

As we can see, there is no perfect clustering algorithm, finding the right one depends on the characteristics of the input data: type of shape for clustering, number of dimensions of data, amount of noise and outliers, etc. As we will see in the next chapter 3 one of the biggest effort on researching for Music Information Retrieval during last decade has been to find the right algorithm for clustering huge amount of music pieces in order to find clustering and relationships among tracks.

2.10 Raw similarity benchmarks

The first approach of the descriptor based similarity ended with a system based on 65 audio descriptors extracted by ANALYST3.1 engine, which then used a K-NN approach and used the Euclidean Metric over all descriptors in order to find similarities between songs.

The raw similarity benchmark results can be checked in the table 2.1, which shows the obtained similarity results using Euclidean Metric over the 65 descriptors extracted by ANALYST3.1 algorithm.

The test database consisted of 500 tracks divided into 35 families. Using a K-NN approach, for each test track I sorted the remaining test tracks by similarity and then selected the top N. In this set, I count the number of tracks that are from the same family as the seed track and when this is averaged over all the test tracks it produces the classification accuracy.

The results from this benchmark are considered a *glass ceiling* for descriptor based approach and the aim of this dissertation is to improve the algorithm to go beyond this limit.

Top position	Correct classification
Top track	69.7%
Top 2 tracks	67.8%
Top 3 tracks	66.3%
Top 4 tracks	63.2%
Top 5 tracks	60.7%
Top 6 tracks	58.4%
Top 7 tracks	56.7%
Top 8 tracks	54.7%
Top 9 tracks	52.7%
Top 10 tracks	51.6%

Table 2.1: Similarity results for descriptor benchmark using Euclidean Metric

2.11 First patent description

Some of the methods described in previous sections, among other methods were included in my first co-owned patent approved in 2006 with Patent No. US 7,081,759 B2.

The patent describes a method and system for music recommendation. The objects of the invention included the method and system for measuring the audio characteristics of a musical composition, as well as for ordering a collection of digital music based on those characteristics, or for identifying the key similarities between different pieces of music in order to offer music discovery to music shoppers. Further on, the patent also included method and system for creating a user taste profile thanks to determining a preferred musical characteristic profile for a music listener, and also compared new digital music files with historical commercial successful songs in order to find the hit potential of the new song.

Following sections will describe the different modules and targets of the invention with further detail.

2.11.1 Analyzing music and extracting clip

The method and system described in the patent used a digital music database of uncompressed linear PCM (*Pulse Code Modulation*) audio files. The first step of the method included, per each audio file, the extraction of the audio characteristics described in

section 2.7.1, using the *ANALYST1.0* engine. The output of that engine generates a text file with a vector of audio descriptors in numerical form a.k.a. parameter vector.

The physical parameters describe quantifiable characteristics of music that may be mathematically modeled to create a descriptive, electronic *footprint* for each song. The characteristics were identified to be the ones that produced the strongest reaction in testers. In general, the mix of parameters is more important than any individual parameter. As already mentioned in section 2.7, the system analyzes one or more of the following characteristics for each musical composition: brightness, bandwidth, volume, tempo, rhythm, low frequency, noise, octave and how these characteristics change over time, as well as length of audio data.

Each input audio data is divided and analyzed into fragments; the size of every fragment was fixed and selected for optimizing performance over a test sample of songs so as to provide an appropriately representative sample for each parameter of interest. Parameters are measured over all fragments and averaged, so the mean values and standard deviation can be measured among the parameter data so as to develop a profile for the entire song file.

After splitting up the input data into fragments, the method uses Fast Fourier Transform (FFT) techniques to decompose the digital music file into a defined set of FFT coefficients for each fragment. Once those coefficients are established, particular coefficients are chosen in order to calculate a numerical value for each of the parameters of interest in each fragment. In particular, every parameter is quantified as:

- Brightness: Spectral histogram built from FFT.
- Bandwidth: Variance of the spectral histogram built from FFT.
- Volume: Average change in the bit sample amplitudes at lag 1. Lag 1 refers to consecutive beats.
- Tempo: Measure obtained using a Beat Tracker algorithm (based on autocorrelation values). The Beat tracker algorithm calculates how many beats per second are in a fragment.
- Low Frequency: Spectral power weighted with ten inverse frequency.
- Noise: Sum of all the FFT coefficients normalized to the maximum FFT coefficient
- Octave: Temporal mean of Cepstrum (inverse of the logarithmic FFT). The Octave is calculated for each fragment and averaged.
- File Size: Number of bytes of the file (it does not use the FFT).

All the output text files with parameter vectors of all music in database are then loaded to the system in order to execute the algorithms and targets describe in next sections.

2.11.1.1 Clip extraction

A clip should be the most representative fragment of a musical piece, that part of the song that identifies the musical piece. If you already know the song, listening to the clip helps you to identify and recognize the musical piece; if you do not know the song, listening to the clip gives you and idea of the complete musical piece, such as movie trailer for films.

At the time of this patent, Music Industry did not yet establish an automated method to extract the clips, and it was not easy to access to music catalogues at that time, so we had to invent the algorithm too.

Using the same ANALYST1.0 tool used at the beginning of the system, the track is divided in small fragments, each one of several seconds long, and then analyzed separately in order to calculate the parameter vector for each of fragment, like if they were an independent music files from database. The fragment with closest parameter vector to the parameter vector of the entire song is chosen as the most representative one.

The method to select the final clips uses standard mathematical analysis, with unweighted quadratic differences between each section and the whole song.

In order to create a clip of X seconds length, the system selects the fragment starting $X/2$ seconds before the position of the most representative section, and finishing $X/2$ seconds after that position, that way the most representative part reminds in the middle of the clip.

2.11.2 Recommendation engine

There were two different methods in this patent to recommend music to an individual user: one method is to capture a user's own personal taste profile through a music test called *Music Taste Test* (MTT) system (see section 2.11.3 for further information), and the other one is for finding similar songs to a seed song selected by the user. This second method is what in the patent is named as *More Like This* (MLT) system.

The MLT function is implemented through a pattern-recognizing artificial intelligence system that allows a user to receive music recommendations by selecting a song and requesting songs that are mathematically similar to that song. The similarity is based upon the musical characteristics of the song, i.e. based on ANALYST1.0 output data; the parameter vectors.

After all parameter vectors for all files in music database are created and loaded to recommendation engine, various mathematical and statistical procedures are run on the loaded data to ensure that all data is meaningful, and to extract the essential characteristics from each song and its preferences profile. The statistical methods used to improve the quality of the analysis information and subsequent Artifi-

cial Intelligence techniques, included normalization and centered of the system using Zero $\bar{X}$ (Mean Data) and $\sigma_X = \sigma^2 = 1$ (Standard Deviation = Variance = 1) for each parameter separately. The mean value for each of the parameters is calculated across the entire song database, and the individual parameters of individual songs are then centered using those means.

With centered and normalized data, correlations between all analysis parameters are identified and eliminated to reduce redundant information given by different parameters and getting the most unique information from each parameter. This is done, first, through correlation matrix, and secondly through the calculation of Eigenvectors and Eigenvalues.

The standard correlation matrix formula applied is as follows:

$$V_i j = \frac{1}{M} \sum_{c=1}^{M} (Z_i(c) Z_j(c)) \tag{2.14}$$

Where $V_i j$ is the value of the intersection of column i and row j, M is the number of songs in the database, c is each particular song, and Z_i and Z_j are parameter number i and j respectively for each song c.

Basically, the correlation matrix multiplies, for each pair of parameters, all values for all registers of data, adding them, and then dividing by the total number of registers.

The correlation matrix is then used to calculate Eigenvector and Eigenvalues, which are used to determine and eliminate correlations between parameters, thereby reducing repeated information and increasing data utility. Eigenvectors are important to determine the new uncorrelated parameters from the original parameters as shown in form (2.13), while Eigenvalues gives us the amount of uncorrelated information contained in each parameter, i.e. they are used to weight the importance of each parameter. It should be pointed out that weights for various parameters are session-dependent and user dependent. Depending on precision and speed desired, weighting and discarding rules are set for each customer.

For example, in a three parameters example, the parameters P_1, P_2 and P_3 of the output Eigenvalues contain 75%, 23% and 2% of the total amount of information respectively. Depending on the specific customer needs (speed, accuracy, simplicity, etc.) the following weighting rules can be applied:

- Keep all parameters equally weighted (33.33%, 33.33%, and 33.33%)
- Weight all parameters based on the percentages above (75%, 23%, and 2%)
- Discard the last parameter and weight the remaining two as P_1 77% and P_2 23%
- Discard the last parameter and equally weight the remaining 2 (50%, 50%)

After following all these steps, the system has all parameters uncorrelated, normalized, and orthogonal and weights have been set for each one, so the system can get

maximum value from them. These new parameters replace in the system the old ones that came from the music analysis tool.

Standard clustering algorithms are then employed by the system to locate and determine clusters within the entire database, for further usage, especially for music recommendation.

Music Recommendation is based on finding similar song profiles or parameter vectors to a specific song. The system finds matches between vectors, looking for song files that have the most mathematically similar analysis data to the parent song. The look up process consists of searching for similar songs by checking the relative distances between all parameters for each song. It is the difference between parameters that determine if the songs are similar or not for the user. The theory of the recommendation engine is based upon the relative correlation between songs, and the following equation shows the formula applied:

$$C = \sum_{i=1}^{N} (S_p - M_p)^2 \tag{2.15}$$

Where C is the proximity value between the database song and the parent one, N is the number of parameters in the parameter vector, S is the parameter vector of the songs in database, and M is the parameter vector of the parent song or seed. Only songs with a value of C below a predetermined threshold would be treated as close or similar songs. The lower the value of C (the distance), the more similar the two songs are.

2.11.3 User's taste profile and recommendations

One of the ways to recommend music to a user is capturing the user's own personal taste profile through the service *Music Taste Test* (MTT). The MTT is used to determine a user's music preferences, and thereby make personalized song recommendations.

The MTT is a pattern-recognizing artificial intelligence system, a learning process where the user is presented with a number of binary choices between two short audio clips, the user has to choose which one he likes the most. After a series of comparisons, it is possible to generate a profile for that user; the profile is analogous to a song's profile as measured in the analysis phase. That way, songs from the music database that share commonalities to the user profile can be identified as candidates to be preferred songs for that specific user. The more times the user chooses between two songs, the more the system will learn, although the system can offer recommendations without finishing that learning process. By asking binary questions, the system finds the preferred values for each one of the song parameters analyzed. Usually, the average rounds or binary questions needed by the process are around 8-15 rounds to learn the final user's taste profile.

The output of the learning system is the user's taste profile as a personal taste vector, which (as mentioned before) is a parameters vector like any of the song's profile in the music database. In this way, the system only has to apply a normal music recommendation to find similar songs to user's taste profile. Although in this case, we will compare songs against users profiles.

The theory of the recommendation engine is based upon the relative correlation between a user's preferred values and each song. Taking into account that correlation is defined as the relative distance measured as the sum of the squared difference between each parameter. A song is always considered as a whole and, thus, individual parameters are not considered to be matches by themselves. The following equation illustrates the formula:

$$C = \sum_{i=1}^{N} (S_p - V_p)^2 \tag{2.16}$$

Where C is the proximity value between the database song and the user's preferred values, N is the number of parameters in the parameter vector, S is the parameter vector of the songs in database, and V is the parameter vector with the user's preferred values. Only songs with a value of C below a predetermined threshold would be treated as close or similar songs. The lower the value of C (the distance), the more similar the two songs are.

This is precisely the same method used to look up recommendations within MLT function. However, instead of using the chosen song or seed, the system looks at the user's preferred values.

The key point in MTT is the learning system. The first two clips for starting the test are selected randomly but maximizing the dissimilarity in the parameters corresponding to each song. Subsequent selections are performed by using the Eigenvector algorithm, maximizing the information given by the selection. The Eigenvector algorithm eliminates correlated data that cannot be used to distinguish between two songs. For example, if two songs have almost same values on an specific parameter, the selection of either song by the user is likely not due to the specific parameter, so the Eigenvector algorithm will recognize that fact and find another two choices in order to capture the taste of the user regarding all of the variable parameters.

2.11.4 User's music preference and recommendations

The pattern includes a method for creating a user's music preference called *My Personal Preferences* function. This function allows the user to rank several songs while interacting with all the system and other functionality. The ranking has five levels scale: *I love it*, *I like it*, *I neither like nor dislike it*, *I don't like it*, *I hate it.*

The user can add and remove any ranking from their list at any time, but, when requested, recommendations can be made using the current songs and ratings on the list. That list is the user's music preference.

The function, then, weighted the parameter vectors of the user's music preference

based on user's rankings, and are used as input to the system to force the learning from those inputs. That input is grouped in two different groups: liked and disliked songs. And only the songs that are similar among songs from same group are taking into account for final profile or musical taste, the rest of them is removed for avoiding noise. Once that first filter is done, the mean values are calculated for each musical taste (liked and disliked) to determine the corresponding parameter vector (the musical taste vector). The final mean will be biases towards the more liked and the more disliked songs thanks to the higher weights applied to both limits of ranking scale.

After the ranking have been established, the system is ready to recommend songs to the user. Starting from the complete database, all songs in the same cluster where the disliked musical taste vector falls are removed from the possible list of recommended songs.

Finally, the MLT is applied using the liked musical taste vector as input song to find similarities among the database.

This process is similar to the MTT learning process, but instead of system deciding what it wants to learn and asking the appropriate questions, the user selects the songs and forces the system to learn from the user inputs. This is referred to as forced learning.

2.11.5 Hit prediction

The final method presented in the same pattern shows another way of using the same technology. This time instead of finding recommendations, the technology compares songs in order to predict the potential commercial success of a new song, i.e. the hit potential of any song.

In order to achieve that target, the system included reference database with 3.5 millions tracks from major music labels, including releases since 1950. That database was updated weekly with latest releases, and the official charts data were also added and updated weekly in the system.

The first step of the method (after analyzing and standardizing all parameter vectors), was to load all songs into a grid called the music universe. Each song was positioned in that universe according to its mathematical characteristics. Songs with mathematical similarities are positioned very close to one another.

After the topology of the music universe was created, the method identifies all hits of database (those songs with commercial success during last years) and remove all those songs that were not hits during last five years from the universe. The songs remaining (only hits of last five years) are grouped into a limited number of small clusters all over the universe, but with vast spaces between them. Then the data is overlaid with some extra parameters related to commercial success of the music database (total sales, highest chart position, date of release, and other common indicator of commercial success). At that moment, the universe is ready to accept new songs to be analyzed

and to know its hit potential.

Hit songs seem to have common characteristics based on the mathematical analysis. As the market changes, the system reflects such changes by finding new patterns in the hit clusters and applying this to the process. While other factors, such as lyrics, theme and artist, impact the success of a song, it is rare for a song that falls outside of the hit clusters to become a hit.

The system scores a new song according to its similarities with current hit songs. The new song is compared to the hits, by comparing the new song parameter vector to each hit parameter vector, to obtain an affinity. The affinity value is a rating that shows how closely related the mathematical patterns in one song are to another. The lower the affinity value between two songs, the more closely related they are. The greater the number of past hits with lower affinity values to the new song, the higher the hit potential of the new song. At the end, the system takes into account the distance with each song of the hit database, as well as details of the cluster and position in the universe, and gives a final score. Scores above specific threshold indicates a strong or high hit potential of the song; at the moment of the patent a rating of 7.00 or greater (in a 0-10 scale) was considered as hit potential.

The method also calculated some extra ratings such as classical hit potential, where the approach is the same as for recent hits, but using the most outstanding hits since the beginning of the official music charts. Recentness and sales scores were the other two ratings. The first one was related to closeness to the most recent hits (i.e. song closer to hits of last year scores higher than if it is closer to hits of 5 years ago). Finally, the sales scores was based on hits universe and the corresponding sales information.

All the above information was organized and collected in a final report that user can use for determining an appropriate course of action for commercializing the analyzed new song.

2.12 Summary

The goal of this chapter has been to introduce the basics of music recommendation based on signal processing. For that purpose the chapter started with main music concepts definition such as difference between tracks and songs, pitch and timbre, and rhythm, loudness and harmony. It has also added some overview of the human being ear audition model in order to understand how we hear sound and music, as well as to understand the need of applying some human perception filter model to audio descriptors extraction in order to use the perceptual intensities of frequencies instead of the physical ones.

All music recommendation system start with the input data; MIR community has done huge efforts on feature extraction algorithms and on improving the accuracy and the performance of those algorithms. Section 2.7 shows the evolution of the analyst engine until a better balance was found between amount of useful data extracted and

performance in the final version of the engine called ANALYST3.1.

The analyzer presented in this dissertation uses 65 audio descriptors vector for each track as input data to the system. The first attempts ended with a successful system and model for music recommendation based on those audio descriptors, as well as in a K-NN approach and Euclidean Metric over all descriptors in order to find similarities between songs. The co-owned patent approved in 2006 (Patent No. US 7,081,759 B2) with a model and system for music recommendation proves that success.

However, the raw similarity benchmark achieved with that approach and exposed in section 2.10, shown that Euclidean Metric has good accuracy for small distances, but not for bigger ones. Moreover, when the distances increased the order based on distances miss all meaning. The system was good only locally, and for the very close tracks to a seed. A glass ceiling was achieved in the accuracy and new approaches were claiming to break that limit.

MIR community also achieved that limit, the final solution needed a different approach; a set of layered algorithms of increased specificity, emulating the human immune system, is the revolutionary approach and system used on this dissertation and exposed in the following chapters.

Chapter 3

State of the art in Music Recommendation

3.1 The beginnings

The first patents regarding music recommendation appeared in 1995 and they clearly defined the main subject of this dissertation. The Microsoft patent US5616876 starts with a very clear goal:

*A **more like** function allows a subscriber to use a seed song to identify other songs that are similar to the seed song, and to add the new songs to the current playlist*

However, that patents describes that the similarity between songs is based on the subjective content of the songs, as reflected in style tables prepared by editors.

From an academic point of view, the *International Society for Music Information Retrieval* was created in 2000 from a conjunction of loosely-related events occurred in late 1999, such as the three year project OMRAS (Online Music Recognition and Searching) which was a JISC(UK) and NSF(US). The first International Symposium on Music Information Retrieval (whence the acronym ISMIR; web site: http://ciir.cs.umass.edu/music2000/) took place in Plymouth, Massachusetts in October 2000 and there has been a alternating principle each year between the Americas and elsewhere. Since then, the participation in the conference has grown and more than 1200 papers have been presented.

The aim of this chapter is to summarize the evolution of Music Recommendation, as part of Music Information Retrieval, based on the papers presented at ISMIR and also the available patents, as this field is in 2013 a very important subject of commercial research and monetize.

I started to work in Music Recommendation in year 2000, as a seed student at the Spanish company *Grupo AIA, Aplicaciones en Informática avanzada* with a small catalog of about 300 tracks and doing the first successful attempts to analyze tracks and use the obtained music descriptors to search for similar tracks given a seed track. The result of my work and the work of others in that company made possible the acceptance of the US patent US7081579 *Method and system for music recommendation*, which was presented in 2003. I have been actively working in the subject until now, in different companies merged or acquired but from a continuity line with the original work, methods and source code.

3.2 2000

As Uitdenbogerd et al. mentioned in one of the first papers [Uitdenbogerd00] presented at ISMIR , Music Information Retrieval had a longer history than most people realized, with some systems developed in 1960 and with roots in different fields like information retrieval, musicology and music psychology. But, until 2000 decade most of works done for analyzing audio works in automated way were focused on speech recognition. So the first steps and researches for modeling music were using the same approaches and methodologies applied to speech recognition, but for music pieces.

That was the case of one of the first papers presented at first ISMIR in 2000, Logan in [Logan00] applied Mel Frequency Cepstral Coefficients (MFCCs) to modeling music and shown that MFCCs can be useful for speech vs. music discrimination, although it was a first step in that methodology that needed lot of future work on researching for fine tuning the parameters and steps of the algorithm.

In early 2000 all the trends were based on human queries, usually hummed. The input query was a hummed melody that the system had to identify across music catalogs, finding either the perfect match or the most similar one. Since speech recognition field got successful results with audio based methodologies, the first MIR systems were based on them, including the hummed queries identification systems. With that target, Uitdenbogerd et al. [Uitdenbogerd00] started the researching of music identification by creating manually a database with original musical works and a set of covers and similar musical works for each musical work, in order to identify what were the most important aspects of the track that a user uses for matching songs.

Before the digital age, all music catalogs were indexed by artist name and title of the work, very limited tags for searching music, specially if you want to find music with some purpose like music for restaurants, gyms, etc. One of the first papers that talked about extracting perceptual and cognitive information from music was the one presented by Huron [Huron00] in 2000. That paper focused the researching in music summarization and mood characterization. The idea behind music summarization was to find an algorithm that could extract the most significant clip of a song other than choosing the first seconds of the track, while the mood characterization tried to classify the musical works into two-dimensional space based on two factors of mood: valence (happy/anxious) and arousal (calm/energetic).

In 2001, Hofmann-Engl focused his researching [Hofmann01] in the pitch aspect of melodic similarity (although he called it meloton) and the cognitive or perceptual aspects for the similarity model. It was one of the first steps for taking into account other parameters than the ones used for speech recognition, although it excluded properties such as timbre, duration and loudness that were important in further researching. One of the main restrictions of this approach was that the chains to be compared had to have the same rhythm and same dynamic structure.

Another interesting work from 2001 was the Tzanetakis et al. at [Tzanetakis01] for automatic musical genre classification based on a set of features for representing texture and instrumentation. As in other works of these early years of the 2000 decade, they focused their work in small set of parameters and excluded other important ones for future works. The set of features represented the music surface and rhythmic structure of tracks, and trained a statistical pattern recognition classifier using real music database and hierarchy of genres represented as a tree. The set of musical surface features took into account shape and brightness of spectral, the FFT magnitude, the amount of noise and the amount of energy. Also, they started using a Discrete Wavelet Transform (DWT) for calculating an octave decomposition of the signal frequency in order to get information regarding the rhythm. The results of the researching was a good starter for future works, so they had an average accuracy of 62% for some main genres, 74% for voice classification (comparing male voice, female voice and sports announcing) and 86% comparing speech vs. music.

Some of the 2002 papers were still trying to apply speech recognition methods to music similarity. So, in Jin et al. [Jin02] we find a suggestion for improving Hidden Markov Models (HMM) for representation of music and finding similarities. HMM is a statistical model for sequential data that was also used for speech recognition, so it seemed to be logical to apply this methodology for indexing music and finding similarities in melody. The methodology represents every song as a string of states, where every state is a note transition represented by an interval (the difference in pitch between two consecutive notes) and the duration of the note. The methodology also uses probabilities for all possible state changes and a representation in R*-tree for finding matches between songs into a database. Although this methodology was quite effective, it was useful mainly for sung melodies and for small music catalogs (it took 21 seconds for matching query into 9400 songs database).

But not all research were based on speech recognition, there were also other approaches that made research on new audio features for finding music similarities. That was the case, for example, of Aucouturier et al. at [Aucouturier02], where they introduced the timbre similarity measures for comparing music titles with quite good results, around 80% of accuracy. Before this article, most of the works about timbre were done for monophonic sound samples and with the target of finding instrument recognition, but this was one of the first works using full polyphonic music and complex instrumental textures, so the extraction of a global timbre description was not trivial. The system used Mel Frequency Cepstrum Coefficients that were modeled with Gaussian Mixture models. One of their interesting findings were that the system found timbre models useful for finding similarities between same songs (i.e. finding covers and duplicates), but also between different songs but from same artist, or even from same genre. They also suggested that any system should not only use the timbre

measure, but the textual one too, taking into account that user needs to decide if he wants to find similar songs (from same genre and artist if possible) or discover totally new content.

Also in the audio features research field, in 2002, Foote et al. in [Foote02] proposed a method for finding audio similarities using audio characteristics based on rhythm and tempo. The interesting approach of this work was that they were using *beat spectrum* for characterizing rhythm and tempo, so the tempo analysis did not depend on particular attributes such as energy, pitch or spectral features, what means that, in theory, that approach could work for any style of music and could distinguish between different kinds of rhythms with same tempo. That feature was useful for finding rhythmic similarities in a tracks database, and one of the possible uses that they offered was for creating playlist as an automatic DJ. Their experiment got quite well results, between 80 to 96,7% of retrieval precision, but it was done with a very small and limited set of tracks, so further research for accuracy and efficiency in large databases needed to be done.

Feng et al. in [Feng02] tried to avoid pitch tracking of input tracks, using Independent Component Analysis (ICA), a statistic model for revealing hidden factors that underlies sets of random variables, measurements or signals. They mainly extracted the singing channel of the audio track, converted into self-similarity sequence using MFCCs, and then used Recurrent Neural Network (RNN) to remember the sequence in order to be able to find matching between any part of track in the database and the input piece sung by the user. The accuracy on results were quite good, around 79% for matching songs, although they already found that with their small testing database they had troubles for working with musical pieces with too much instrumentation or drums.

During the first years of ISMIR, most of based-content technology research were using MIDI format for tracks database (actually there are still some research using that format), but that type of content is not available for all catalogs, so the trend of 2002 was to use raw Pulse Code Modulation (PCM) audio format with high sound quality (like $44kHz$, stereo). That is what Rauber et al. used in their research [Rauber02], although they transformed the data a posteriori to lower quality and chose parts of the track instead full track to analyze. Rauber et al. proposed an approach for creating a hierarchical organization of tracks following their perceived sound similarity. They used an unsupervised neural network such the Growing Hierarchical Self-Organizing Map (GHSOM) for creating the hierarchical organization. The output of the system was a content-based organization and visualization of audio tracks, where music of similar sound characteristics were mapped together. They mainly use rhythm patterns in various frequency bands as descriptor audio tracks, and applied psycho-acoustic models during feature extraction process. The GHSOM (which uses several layers of SOM) automatically identifies the inherent structure of the music collection.

Maybe one the most crowded research field during 2002 was the music summarization one, an automated way to extract the most representative excerpt of the musical piece. What is also known as *clips* that usually are 30 seconds long. Music similarity methodologies can be used for extracting the most significant part of a song. Cooper et al. in [Cooper02] presented a method that compare different sections of the track

against full track in order to find the most significant part and use it as music summarization. For extracting parameter of the audio they used Mel-Frequency Cepstral Coefficient (MFCC) analysis and then applied Singular Value Decomposition (SVD) for reducing dimensions of the uniformly-spaced bin spectrogram. Finally, for finding the most representative segment of the track, they used a similarity matrix with differences between every segment. The results were quite satisfactory as a first step, and they wanted to continue working on this adding a method to ensure that the most representative section will begin and end at a meaningful segment limits, such as verse/chorus transition.

Another music summarization work was the one presented in 2002 by Hirata et al. at [Hirata02]. They presented a starting point work for using polyphony based for music summarization of piano pieces. The work was a tool for time-span reduction, discovering a piece structure by means of similarity checking, but it needed human beings to interact with the system and let it know which pieces are most important for summarization. The interesting thing was that they applied Generative Theory of Tonal Music (GTTM) and the Deductive Object-Oriented Database (DOOD). GTTM is a theory that Lerdahl and Jakcendoff developed in 1983 for formalizing the listener's intuition; it is based on the idea that removing grace notes from long melody, we obtain a simple similar sounding melody. DOOD framework is based on the idea that things in real world can be represented as the combination of a basic or atomic object and a set of attributes.

And we still have another interesting work in music summarization with Peeter et al. in [Peeters02]. They worked in music audio summary generation but using only signal analysis, without any further information or feedback from users. They considered the audio signal as a succession of states corresponding to the structure of a piece of music. Those states are based in dynamic features representing the time evolution of the energy content in various frequency bands. Those features are segmented in order to create templates, and then the system finds the structure through an unsupervised learning method (K-means and hidden Markov models). The output excerpt was created using a representative example of each state, so it was an automatic generation of sequential summaries (like movie's trailer).

If 2002 was the year of music summarization, I would say that 2003 was the year of segmentation and music classification. Previous works proved that choosing the right audio features is the key point for achieving any target of MIR systems, so there were several research on finding the best audio feature that could summarize the most important data, or even for finding the set of correct audio features. But, every audio feature needs to be extracted from a segment of the track, and choosing the right long of segments or how to segment the track is also a branch of MIR.

Harford in [Harford03] presented a self-organizing neural network for automatic segmentation, learning and retrieval of melodies based on pitch and rhythm. Sheh and Ellis in [Sheh03] worked on chord sequences segmentation and recognition because they thought that chord sequences are a description that captures much of the character of a piece in a compact form, at the same time that allows system to segment the input musical piece into time intervals that correspond to specific words. They based their work on speech recognition tools for audio features, and hidden Markov models

(HMMs) trained with Expectation-Maximization (EM) for sequence recognition. In their small and bias testing set (only 20 songs from same artists) they got accuracy around 75%. The interesting part was that they were able to extract complex musical information such as recognizing chords and finding time alignments, although other systems have proved to be more accurate and more useful for MIR purposes.

Later, in 2005, West and Cox in [West05] studied the different length segmentation for calculating audio features and their influence to automatic audio classification. After testing different lengths (23ms, 200ms, 1sec, 10 sec and whole file), they realized that they were all non-optimal and created a new segmentation methodology based on onset detection. Using short audio frames (like 23ms) across a whole piece tends to drive the distributions from each class of audio towards the center of the feature space, which reduces the ability to separate of the classes. In the other hand, using long sliding windows (like 1 second) that are highly overlapped allows a classification scheme that models multiple distributions for a single class of audio, but they complicate the distributions because windows are capturing several different musical events, as well as it includes a very large amount of redundant information (for example a single sound can contribute to 80 or more feature vectors). All those problems can be solved using a segmentation based on an onset detection technique. The essential idea of this technique is that peaks in the positive differences in the signal envelope correspond to onsets in the audio stream. Every onset can be defined as a musical event. Figure 3.1 shows several audio segmentation and temporal modeling windows. Their work proved that the onset detection based segmentation had the highest accuracy on music classification results

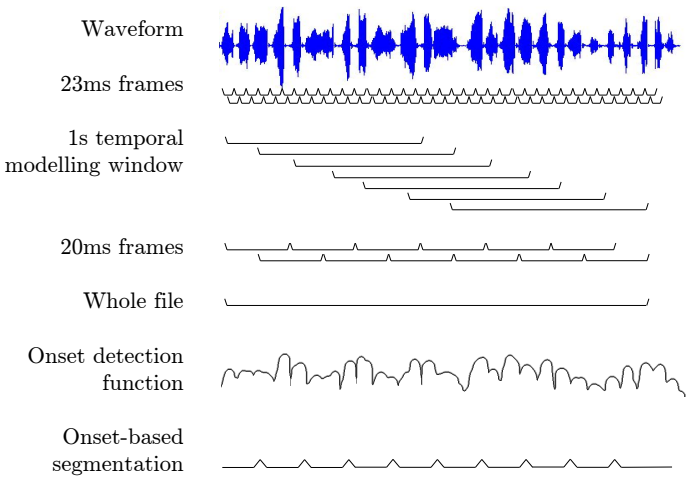


Figure 3.1: Several audio segmentation and temporal modeling windows. From [West05]

Also in 2005, and regarding the music segmentation, Chai and Vercoe in [Chai05] made a researching in tonal analysis, and created an HMM-based approach (Hidden Markov Models) for segmenting musical signals by key change and identifying the key of each segment. They used only classic piano music for their researching. Although

their approach was interesting, the effort for creating the learning database was too high for being able to be extrapolated to huge music catalog (they used only 10 classical musical pieces).

The first attempts for finding similarities between songs with automated way were using small set of audio features or were even focused in only one or two selective characteristics of an audio signal. Allamanche et al. [Allamanche03] in 2003 studied a set of 1000 features (or subsets of main features) and chose the 20 most representative and useful for music similarity purpose. Those final candidates included SFM (Spectral Flatness Measure) and SCF (Spectral Crest Factor) as indicators of how flat or peaky is the power spectral density, Normalized Loudness as a band-wise measure of the perceived sound intensity, MFCC (Mel Frequency Cepstral Coefficient) for representing a signal's spectral envelope shape, and Delta Log-Loudness as the band-wise difference over time of the logarithm os specific loudness. Their results shown that they achieve accuracy around 64% on music similarity.

McKinney et al. in [McKinney03] studied four different audio features sets for genre classification (low-level signal properties, mel-frequency spectral coefficients and two sets based on perceptual models of hearing). For classification they used a standard Gaussian framework. Their accuracy on genre classification was around 74%. And Parry and Essa in [Parry03] proposed a similarity metric based on rhythm that matches rhythms that share the same beats regardless of tempo or identicalness. The target of that work was to create a tool for finding the best transition between multiple audio sources, a feature very useful for DJs and for creating mixes. They also said that methodology could be used for finding content in audio catalogs, although they did not explain how to use it or if the had good results enough in that field.

But, as I mentioned, 2003 edition of ISMIR also brought lots of efforts in music classification, specially for genre classification. This is one of the most popular targets for MIR because almost all large music databases fail on genre classification. They are classified depending on culture things (world music for US catalog can include Spanish rock music, while for a Spanish catalog that music would be classified as Rock, but it would include Indian or Chinese music in the World category), but also, the main issue, with such a large amount of data, and growing every day, it can not longer lies into human tagging.

For mentioning some of the paper of 2003, Dixon et al. in [Dixon03] compared two methods for genre classification of a specific subset of music standard and Latin ballroom dance music. Both methods were for extracting periodicity from audio tracks, in order to find timing patterns and then recognize the style of music. One method performed onset detection and processed all the detected onsets, while the second method used inter-correlation on the amplitude envelopes of band-limits versions of the signal. They found that second method was the best one, with a maximum accuracy of 80%, although their results were clearly limited to the style of the study.

Li et al. in [Li03] worked in emotion detection in music as a multi-label classification problem (i.e. same musical piece could appear in several classes simultaneously), and they use Support Vector Machines (SVM) for its implementation, although they did not have very good results. Chinthaka Maddage et al. in [Chinthaka03] also created

an automated hierarchical music classification based on SVM for classifying music into vocal, instrumental and mixed. They achieved an accuracy around 85% but the main drawback was the high computational complexity.

Later research still used the SVM. Meng and Shawe-Taylor in [Meng05] investigated two models for modeling short time features for genre classification; the multivariate Gaussian model and the multivariate auto-regressive model. And used the SVM approach for the automatic genre classification using the short time features. Their results had around 44% of accuracy with a data set of 11 different music genres (while the human performance was around 52% for same data set). The common time feature for a human to genre classification is of the order to a quarter of a second up to several seconds. Short time features such as MFCCs are usually done with 10-50ms time windows of the audio signal.

In spite of all good researching on MIR, in 2005 collaborative filtering had better accuracy results than any based-content technology for finding music similarities. That year, Stenzel and Kamps in [Stenzel05] bridged the performance gap between collaborative and content-based similarity measures. They presented an approach to improve a content-based system by training a collaborative model with content-based sound features. Their results were very satisfactory.

In 2004, the Music Technology Group (MTG) (from Universitat Pompeu Fabra (UPF), in Barcelona, Spain) organized the Audio Description Contest (ADC 2004) as part of the ISMIR 2004 conference. It was such a successful event, that next year (2005), The 6th edition of ISMIR run in conjunction with the 1st Music Information Retrieval Evaluation eXchange (MIREX 2005) [Downie05], that can be considered a direct descendant of ADC. Since then and up to current days, ISMIR and MIREX have run in parallel.

The different categories to apply in MIREX shows the evolve of MIR. So, in ADC 2004 there were 4 different categories, while last MIREX edition (MIREX 2012) had 17, most of them audio based-content instead of using symbolic data. Some categories like audio classification, audio melody extraction, audio onset detection or audio key detection have been present in all MIREX editions.

Towards the middle of 2000 decade, the amount of different approaches for audio features extraction, classification methods and targets for MIR were so high, that it was time for evaluating, checking, rating and reconsider some of them before continue creating new approaches. That was the case, for example, of Sigurdsson et al. [Sigurdsson06] in 2006, regarding input audio format. The available types of input audio format have been changing during last decades, and MIR approaches have readjust their technologies for using the new formats. MP3 has been, by far, the preferred audio format for the majority of music listeners, although it was not the preferred audio format for researchers , specially at the beginning of 2000, where researchers tried to find the most accurate audio data, so they preferred lost-less formats like WAV or MIDI. But MP3 become the winner in this field, but, with large amounts of data available in the web, in user collections, etc. the parameter settings, the MP3 coding, usually vary a lot. Sigurdsson et al. studied the influence of those different parameters (bitrate and sampling rates) into MFCCs results. Their main conclusion was that the

widely used subset of MFCCs is robust at bit rates equal or higher than 128 kbits/s, and non-robust with lower bit rates.

Another evaluation paper was the one presented by Fiebrink and Fujinaga, also in 2006. One of the methods applied in several works presented at MIR for improving music classification included the idea of feature selection, specially after the first years of researching in MIR where the audio features vectors grew from only a set of ten to several sets, or even several hundreds. In spite of all efforts in music classification field, most results did not have accuracies higher than 60-70%, so researchers thought that there were over-fitting in the data that could be solved by features selection. Fiebrink and Fujinaga in [Fiebrink06] discussed and demonstrated that this approach was overestimated. According to them using a simple PCA approach with all input audio features can give the system the same or even better accuracy than using feature selection.

But not all work in 2006 were about evaluating and checking previous ones, there were also a bunch of new approaches and ideas. Among all of them, two new trends started to push hard: the use of probabilistic model and the playlist generation.

Flexer et al. [Flexer06] combined the music similarity approach to music genre classification using a probabilistic combination. They trained separated classifiers for the different feature spaces and then combined posterior probabilities to obtain a joint decision. Their testing database has only ballroom dance music styles, but their method could be applied to music categorization, although the accuracy in most of the tested styles was between 50-65%. Temperley in [Temperley06] also used a probabilistic model for his approach, although he used it for melody perception and key detection. And Whiteley et al. in [Whiteley06] used probabilistic model approach to temporal structure in order to inference three musical concepts: the tempo, meter and rhythmic pattern.

Regarding playlist generation, Pauws et al. in [Pauws06] presented an algorithm for creating music playlists in automated way, with some user's preferences and the use of penalty functions and a neighborhood structure. Cunningham et al. [Cunningham06] analyzed how people construct playlists and mixes. As they mentioned in their paper's title, playlist generation is *More of an Art than a Science*, so they support the creation of systems that allow users to set up, get feedback and make decisions about the final playlist. In other words, they advocate for systems that *help* the user for creating playlist, instead of using totally automated methods like the one proposed by Logan [Logan02] in 2002 (more focused on avoiding abrupt transitions by selecting consecutive songs that are closely related like, for example, by mood), or the one proposed by Pampalk et al. [Pampalk05] in 2005 (although they used some user feedback). Oliver and Kreger-Stickles [Oliver06] in 2006, follow that idea, and although they generated playlist in automated way, they took into account the userś physiological response to the music to determine the next song to play. Finally, Pampalk and Gasser [Pampalk06] presented a new playlist generator based on audio similarity measures and some minor user feedback. Later, in 2008, Flexer et al. [Flexer08] proposed a playlist generation system that uses music similarity for selecting songs, but that takes into account smooth transition between start and end songs of the playlist (songs that user has to choose)

If 2006 was the year of evaluation and playlists, 2008 was the year of tagging.

Tags represent a rich source of semantic information that can be useful for text-based music retrieval, recommendation, discovery and visualization. A system using only tags can recommend content, so it is logical that several MIR approaches based their work on tagging or use that information to improve based-content systems. Turnbull et al. [Turnbull08], in 2008, compared five approaches for collecting tags for music; from human participation tagging (surveys, social tags, games) to automatic methods (text mining and auto-tagging audio content). One of their conclusions confirmed that systems based on web documents and social tags are influenced by popularity bias, while content-based auto-tagging systems are not.

Symeonidis et al. in [Symeonidis08] explored the multi-modal behavior of users in social tagging for music, and created a methodology using that information for predicting tagging and recommend content to users. Multi-modal tagging refers to the fact that some track can be tagged by user A with only tag T_1, while another user B can tagged the same item with tags T_1, T_2 and T_3. It is a type of collaborative filtering methodology but applied to automated prediction tagging and then using those results for music recommendations. Interesting approach, specially because it is exploring the social tagging phenomenon, but although it can have some useful uses, it has the problem that can only recommend the already tagged content (i.e. it cannot be used with new content until some user will tag it).

But, 2008 also brought more research and discovery on audio features like in the case of Izumitani and Kashino [Izumitani08] for key and tempo variations detection through self-similarity matrices (using MIDI files), or the paper of Ravelli et al. [Ravelli08] that proposed a different approach for audio indexing in the transform domain, using sparse representations instead of standard transform based coders. The Ravelli et al. method computed mid-level representations for beat tracking (onset detection) and chord recognition (Chromagram). Holzapfel and Stylianou [Holzapfel08] also presented a new approach for onset detection, this time based on the phase spectrum and specifically using the average of the group delay function, what implies that an onset could be detected immediately by locating the positive zero crossings of the phase slope, and also that the system is robust to signals with little percussive content.

Among other papers presented in 2008, there were two of them with fresh new ideas for MIR applications: a music thumbnailer, and a music selection system based on runnerś step frequency. Yoshii and Goto [Yoshii08] presented a way to visualize musical pieces in thumbnail images with based-content meaning. They used audio features and generate compact images corresponding to the audio signals of individual pieces, that way, a user can distinguish audio characteristics from tracks at first view, without the need of knowing the track or streaming the song. On the other hand, Masahiro et al. [Masahiro08] also presented an original system in 2008, an automatic music selection system based on runner's step frequency (derived from an acceleration sensor). Depending on the step frequency, the system selects the most appropriate music with tempo suitable for runner's steps frequency (and changes it when the step frequency changes). Their experimental results shown an accuracy on selecting right music higher than 85%, and they ensure that their system makes running exercises more pleasing.

3.3 2010

In the last editions of ISMIR we can see how the different branches of MIR are refining the approaches, the algorithms, how research mix ideas and fields to improve a few decimals some accuracy or how they come up with surprising ideas for new algorithms. All branches of MIR make steps forward year by year. So, I am going to emphasize in this and the following two sections the best ones for every branch.

One of the issues for automatic music categorization is that not all parts of musical piece have the same importance; so applying them in different way depending on the part of the track can be important. But, there are only a small part of huge databases manually analyzed in that way, dividing tracks into important segments, and they are not even the same depending on the music style. For those reasons, in 2009 MIR community started working on automatic segmentation of musical pieces (the first approaches were done in 2004 to 2008). Instead of researching new audio descriptors for improving music categorization, the idea was to apply the existing ones in different way to different segments, or even using only some parts of musical pieces.

So, in 2010 we find several works on audio-based automatic music structure analysis, all of them with the purpose of dividing an audio recording into temporal segments that correspond to musically meaningful parts and categories. Paulus et al. [Paulus10a] presented an overview of state-of-the-art methods for computational music structure analysis, and then Paulus proposed, individually, a method in [Paulus10b] for labeling music piece structure descriptions using acoustic information. Bimbot et al. in [Bimbot10] also worked in automatic structural segmentation of musical pieces. Eyben et al. in [Eyben10] presented an onset detection method based on auditory spectral features and relative spectral differences using a recurrent neural network. They claim to have high performance and temporal precision for all kinds of music, including complex music mixes. And Kaiser and Sikora [Kaiser10] introduced a method for automatic detection of musical structures in popular music, using non-negative matrix factorization to segment regions of acoustically similar frames in a self-similarity matrix of the audio data. Their results can be applied to explain the structure of the whole music piece.

Holzapfel and Stylianou in [Holzapfel10] use the concept of *parataxis* for finding similarities among songs, what is the same as detecting morphological similarity but for specific traditional eastern Mediterranean music. The methodical description of the structure of the form of musical works is known as morphology of music, and it includes elements like themes, phrases and motives (which are made up of sound characteristics like tonal height, duration, intensity and timbre). Traditional music of Eastern Mediterranean area follows a different morphology than western music, using small melodic phrases which do not follow a specific morphological structure, this is called parataxis. For finding the melodic pattern similarity they used a methodology created for detecting cover versions of songs in western music (Ellis and Polliner in 2007)

And speaking of Western music, not all MIR algorithms can be applied to all type of music, and I am not talking now about genres, but about cultures. Most of ISMIR articles talked about popular Western Music, and they try to use audio features and

algorithms that are, in theory, independent from the type of music, but that not always work. Mak et al. in [Mak10] proved that for Chinese pop music, rhythm and mood related attributes (like tempo and degree of noiseless) and more correlated to human perception in that culture. Instrument and singing style are less relevant, while they are quite relevant in Western music.

In the same way that most of MIR researching started applying methods to music that came from other fields like, for example, speech recognition, in 2010 Chang et al. in [Chang10] applied Compressive Sampling (CS) to music genre classification, achieving significant reduction in the dimensionality of the audio music signals and in computational time, at the same time that has a high accuracy value. Compressible signal can be precisely reconstructed from only a small set of linear measurements, that property is used in what is known as Compressive Sampling (CS) and it has been applied successfully to computer vision, face recognition, phonetic classification and so on. The main benefit of CS is the dramatic reduction in sampling rates, power consumption and computation complexity in digital data.

Moving on to music similarity and music recommendation, Casey and Topel in [Casey10] worked on rhythmic similarity based on timbre features. A set of features based on rhythm and timbre can describe percussive rhythm group similarity, but their work was more focused on rhythm matching of distinctive rhythm groups, identifying and describing rhythm structures that appear frequently in musical databases.

Collaborative filtering methods usually beat based-content methods in music recommendation, but they cannot offer long tail to the user. Last trend in this field is to mix both methodologies. This is what McFee et al. proposed in [McFee10], a method for improving content-based recommendation in the long tail by learning an optimized similarity function from a sample of collaborative filtering data.

Miotto and Orio [Miotto10] also mixed different source of data for music recommendation: acoustic similarity and semantic description. The semantic description is composed by tags describing information about: genre, instrument, acoustic qualities, vocal characteristic, emotion, and usage. They HMM for both representations, in the model each state represents a song, but transitions probabilities depend on acoustic similarity, while observation probabilities represent semantic descriptions. Their results were quite satisfactory, but further testings should be done with untagged long tail content to ensure that the methodology can work for large catalogs.

Karydis et al. [Karydis10] studied the known imperfection of spectral similarity measures for music data: the existence of hubs, orphans items and the distance concentration phenomenon with high dimensionality data space. *Hubs* are items that are close neighbors of many other items although they do not have perceptual similarity, i.e. they are false positives cases. It is an issue that gets worse with larger databases. On the other hand, *orphans* are items that are never similar to others, they rarely become close neighbors of any other item although they possibly have perceptual similarity to a large number of songs. Finally, the *distance concentration phenomenon* is related to the fact that in high dimensionality data space all items tend to at nearly the same distance from each other, so it is hard to identify meaningful nearest neighbors. The dimensionality of the space depends on the number of features used, but it

is a balance issue between less features and easy dimensionality space but poor with results, or more features and then deal with meaning of high dimensionality space.

Flexer et al. in [Flexer10] also studied hubs issue. They compared different databases and music similarity algorithms for identifying the tips for reducing hubness in audio similarity algorithms. Their conclusions were quite interesting, showing that a non-timbre based parametrization of audio similarity has less hubness that standard approach of MFCCs.

Artist labeling and artists similarity tasks have not been very successful in MIR purposes, but Schedl in [Schedl10b] propose the use of music-artist related microblogging posts and services, such as *Twitter*, to perform artist labeling and similarity estimation tasks. The approach uses different text-based indexing models and term weighting measures, and concludes that microblogging is a valid source for musical meta-data.

In the same field, Schedl et al. in [Schedl10a] proposed methods for calculating the artist popularity ranking in a certain country. With that purpose they used four different data sources in the web: page counts from web searching engines, posts on Twitter, shared folder in a file sharing network and play count data from last.fm. They did not find a perfect approach, but their work can be the first step forward for ranking and overall popularity for artists.

Collins in [Collins10] proposed a work for finding artist influences to any band history using two different approaches: collaborative filtering and based-content. Although his results were not outstanding, the idea of using similarities for finding influences to bands was a new and fresh idea.

Regarding music genre classification, Draman et al. in [Draman10] proposed a revolutionary approach: the Artificial Immune System (AIS) domain applied to music classification with a new technique called Modified Immune Classifier (MIC), the newest version of Negative Selection Algorithm (NSA). NSA stresses the antigen recognition process, using two processes at the same time: monitoring (a process for recognizing self/non-self cells by performing the affinity binding) and censoring (the process where antibodies or detectors are randomly generated to match with antigens). MIC eliminates the random process in censoring and defines self cells based on how many classes of data they need to identify and then generate the detectors accordingly.

Rump et al. [Rump10] also worked in genre classification. They observed that the presence of harmonics causes the high MFCCs to be noisy, so they improved the accuracy of MFCC-based genre classification by using the Harmonic-Percussion Signal Separation (HPSS) algorithm to split the signal into two signals: one containing the harmonics, and the other one containing percussion. After that, the MFCCs are calculated for every separated signal, and a multivariate auto-regressive (MAR) model is trained with the improved MFCCs. The best performance achieved was 78.3%, what is a very good result.

Another approach for improving the accuracy on genre classification is selecting the right audio features to be used. That is what Hamel and Eck used [Hamel10] proposed, they use a Deep Belief Network (DBN) for choosing the right audio features to be used

in a genre classification or even in other tagging classification system. The DBN learns abstract representation (high level features) of the input data (all audio features) using unsupervised learning. The output of the DBN is than used as input for supervised learning tasks like music classification.

The automatic tagging of music with descriptive keywords (e.g., genres, emotions, instruments, usages, etc.) based on the content of the song is one of the fields more required by users and music business. Coviello et al. in [Coviello10] propose to use a recent new song model that is based on a generative time series model of the musical content, the dynamic texture mixture (DTM) model, which treats fragments of audio as the output of a linear dynamical system. A DTM takes into account both the acoustics and the dynamics of audio sequences. The acoustics variables encodes the audio features vector, while the dynamics encodes the evolution of the acoustic component over time. They based their model on temporal dynamics and timbre content.

Finally, one of the latest trends on MIR: the music visualization. In the same way that Yoshii and Goto [Yoshii08] proposed in 2008, Chen and Klüber in [Chen10] explored and developed an application for generating thumbnails of musical pieces where the user can see in a visual way information regarding tempo, volume, genre, aggressiveness and bass.

Schnitzer et al. in [Schnitzer10] proposed a method for clustering and visualizing Gaussian music similarity features for computing maps of music collections. Multivariate Gaussian are widely used in automatic music recommendation, however they do not work very well with clustering and visualization methods like SOM. Schnitzer et al. proposed and approach based on the symmetrized Kullback-Leibler centroid of Gaussians to allow the creation of SOM with preservation of original similarity topology, and with less complex computations.

And last, but not least, Stober and Nürnberger presented in [Stober10] a very similar approach to my catalog visualization method of 2003, based on audio similarity, that I called *Music Universe*. They created the MusicGalaxy, an interface for exploring large music collections at track level, using the galaxy metaphor. Every track is represented by a star in the universe and the user is able to navigate through the galaxy. The placement of the stars is computed using multidimensional scaling (MDS) that uses audio features. MDS is a neighborhood preserving projection technique that preserves the distances (dissimilarities) between the objects in the projection. The user is able to explore some part of the galaxy in depth via zoom in, for that purpose the SpringLens technique is applied to that specific part. The SpringLens is a complex overlay of multiple fish-eye lenses that zoom into regions of interest at the same time that compacts the surrounding areas.

3.4 2011

Music business has grown fast during last 10 years, and so the digital music available worldwide, but most of MIR researching are done using very small test and training

databases, some of them with only less than a hundred tracks, some of them with at the most a few set of ten thousands. Bertin-Mahieux et al. presented in ISMIR 2011 [Bertin11] and [Bertin-Mahieux11] for encouraging MIR community to use large databases as the music business is already doing. A Million Dataset.

Most content of the Million Dataset is in MP3 format, but there are still some works based on MIDI files. Abeßer and Lartillot in [Abeser11] presented a method for characterization bass track and drum track of musical pieces instead of using whole piece, and they used MIDI files. And Ahonen et al. in [Ahonen11] presented a compression-based method for measuring similarities, using binary Chromagram extracted from MIDI files and Normalized Compression Distance (NCD). The chroma vector is a 12-dimension representation of notes, stripped from octave information, that are present in a given time frame. In 2011 we also find other works based on chroma vectors, Bandera et al. in [Bandera11] present a fingerprinting method based on chroma vector to match humming input query with real stereo tracks database (with CD quality). The reference database is processed in order to enhance the main voice, and that process does not need to obtain the onset or the exact tone of the sound, what makes it a robust representation for possible imprecise humming or main voice enhancement.

Most of musical genre classification methods use machine-learning approaches using support vector machines with some kernels. Anan et al. in [Hatano11] tried a different approach: a similarity-based learning framework. The similarity-based approach has a theoretical guarantee that one can obtain accurate classifiers using similarity measures under a natural assumption. They use MIDI files.

Content-based vs metadata is one of the latest trends into MIR. Bogdanov and Herrera in [Bogdanov11] studied the possible improvement of adding metadata for genre tags to based-content technology in music recommendation. They first use a content-based approach based on low-level (timbre, temporal and tonal) and inferred high-level semantic descriptions of music, and then add genre metadata and compare results. Their results show that this approach refined improves the music discovery experience for both, the long-tail and the popular music items.

Bryan and Wang in [Bryan11] use network analysis for understanding the dynamic of a complex system such as music influences (artist to artist or genre to genre relationships) based on sample-based music (understanding sample-based music as a musical work that borrows material from another musical source). It is a step forward to understand how artists, music styles, and music itself evolved over time. One of their main results show that modern hip-hop, R&B and electronic music have heavy influence of funk, soul and disco music.

Wang et al. in [Wang11] present a content-based music search system that uses multiple tag filtering with multiple levels of preference in the input query. Tags like genres or male/female singer can be set up with a scroll bar for giving the preference weight (between 0 and 1) for each tag. They call this type query as *Multiple Tags with Multiple Levels of preference* (MTML query). The training database was manually tagged, but the testing one was untagged.

Last trends on MIR try to adapt generic techniques for music recommendation

to specific contexts or users groups. But, the way music is compared seems to vary among individuals and cultures, so it is not an easy task to find the right adaptation. Wolff and Weyde in [Wolff11] studied that concept and presented an approach to use machine-learning techniques for analyzing user data that specifies song similarity.

3.5 2012

And we arrive to last edition of ISMIR. So many works presented during those 12 years, that what Urbano et al. [Urbano12] presented last year was very welcome: a comparison of results among different MIR systems. They compared the real users satisfaction of MIR systems with 100 user testers using 18 different systems of Audio Music Similarity and Retrieval task (AMS), and offer a method for comparing different systems.

Another comparison among different music recommendation algorithms was also done and presented last year by Flexer et al. in [Flexer12], but in this case they studied the *hubness* phenomenon. As I already mentioned in previous sections, in spite of the recommendation algorithm used and the set of audio features extracted, there is always some song or a set of songs that are wrongly classified and close to too much songs of the catalog, the *hub* songs. One of the issues with hub songs is that they appear over and over again in the recommendation lists. But, more important, usually the hub songs are less similar then non-hub songs according to human perception. This is known as hubness phenomenon, and it seems to be caused because of the high dimensionality of the feature space. Flexer et al. made a study of this phenomenon over 17 different algorithms. They found that hub songs are not always the same, that depend on the algorithm and audio features used, but, the most important conclusion was that the phenomenon can be minimized using probabilistic *mutual proximity* (MP) to the distance matrices to re-scale audio similarity distances (hub y is the nearest neighbor of x, but the nearest neighbor of hub y is another point a ($a \neq x$)).

The state-of-the-art genre classification methods in 2012 is mainly based on the SVM and k-NN, but Anan et al. in [Anan12] shown another approach for polyphonic music based on converting musical pieces into binary chroma vector sequences, and then applying the dissimilarity based classification method called TWIST (also their work and methodology from previous works). They claimed to achieved an accuracy of 90% for classical music and Japanese Pop style. Quite impressive results that should be compared with other music styles and input audio formats (they uses MIDI) in order to see how accurate and powerful is that method.

Another approach for automatic genre classification presented last year was Wülfing and Riedmiller [Wulfing12] one. They investigated the use of an unsupervised feature learning methodology, applying a successful methodology from computer vision field to MIR. Although they chose one of the most basics approaches for unsupervised feature learning method, like k-means, one of their interesting conclusions was that convolution extraction of local feature responses can be a good approach for improving the genre prediction, since it has the balance between computational complexity and

accuracy. And, also, that time-frequency features perform better accuracy than frame level features.

In another field, an interesting work is the one by Watson and Regan [Watson12]. They added contextual information to audio features for creating their model of musical mood. The contextual information was gathered in-situ during a users' daily life with data like listener's affective state or listening context. They achieved accuracy of 67% on successfully classification of musical arousal (energy of emotion), and accuracy of 75% on musical valence (positive or negative emotion) when using a combination of audio features and listening context information in the model. The interesting part of their study was that they compared results using only audio features (with accuracy of 59.5% in musical arousal classification and 53% in musical valence), audio features and affective state (60.3% for musical arousal and 60.2% for musical valence) and, finally, musical features and listening context that had the highest accuracy results classification for both, musical arousal and valence.

Finally, I would like to mention the work of Nieto at al. in music summarization. Music summarization was an important field of researching at the beginning of 2000, but after a few tries most of companies of music business decided to use 30 seconds clip of every song starting after first minute. That is useful for most of Western popular music, but not for all type of music. Furthermore, even finding the most representative excerpt of the track, that cannot be the real representation of the full track, so Nieto et al. at [Nieto12] decided to present a method for generating audible summaries of music recordings that really can explain a track through mutually disjoint segments of itself. It would be like a movie trailer but for musical pieces. An interesting approach, although is quite probably that most of music business will not be interested on it, since it is more time consuming than the automated way using nowadays.

3.6 Most interesting lines

As we can see in previous section, music similarity is a large field of MIR research that can imply a bunch of different targets; from speech vs. music discrimination, to music categorization going through identifying a musical piece or even visualize a music catalog. All research done in MIR and papers presented at ISMIR reflect that variety, but most of them go through similar steps in their approaches: characterize the input audio data, apply methods for finding similarities and patterns behind data, and find the result that fits the query or target of the approach, either identify, or classify or find similar tracks. Every of those steps have been studied in depth and different approaches and methods have been applied.

This section will summarize those most interesting lines in ISMIR papers, classifying the information based on target of the systems. The only main line that will not be covered by this section is the genre classification, because there are so many research presented at ISMIR that I prefer to talk about that in next section.

3.6.1 Based-content Searching

Content-based searching engines can be useful for most of music listeners in order to identify content (by hummed query for example) and find recommendations (from the hummed query of from a specific track, artists or genre), but also for musicologists for studying in depth the influence of some artists to other ones, and for music business for using the long tail content of the catalog or even for legal issues like finding copyrights infringements with covers or plagiarisms.

As Typke et al. mentioned in their paper [Typke05] in 2005 (they made a research on existing MIR systems at that moment), until 2005 we could distinguish two main groups of MIR systems for content-based searching, systems for searching notated music (or symbolic data), and systems for searching audio data.

Symbolic data content-based systems

For the notated music some systems used monophonic melodies in order to represented the data by one-dimensional strings of characters, where each character represents one note or one pair of consecutive notes. The matching algorithms find the occurrence of input string into another of the reference database, most of them looking for exact matches with the whole string or with some sub-string of the items in the database. But, for systems that try to find approximate matches, a distance between strings needs to be calculated, and for that purpose dynamic programming and indexing methods (like B-trees for example) have been used in several approaches. Typke et al. [Typke03] in 2003 described a vantage indexing method with transportation distances for measuring melodic similarity, but there also other methods like metric trees or vantage point trees.

With polyphonic music the approaches are different. Instead of using string-based methods we need set-based methods, so we cannot assume that notes are ordered. In those cases we need to assume that events have properties like onset time, pitch and duration. They are still based on notes, but seeing to them as sets. Clausen et al. [Clausen00] in 2000 indexed music by quantifying onset times and by segmenting the music into measure through inverted files. While, Typke et al. [Typke03] avoided the need for quantifying using the triangle inequality for indexing.

Audio data content-based systems

Instead of using notes or any other method that tries to reproduce exactly the concrete description of music, another way of comparing tracks is using abstract representation of audio signals. That is the most common approach in all branches of MIR. This approach extracts the perceptual and cognitive information from music, or at least that most relevant aspects of them. The common way to apply that is segmenting tracks into short frames, and most of times overlapped, that usually are short enough for not including multiple events in the same frame. Frames can have a duration of the order of 25-40 milliseconds. The most common features extracted from that frames can be grouped as perceptual features such as pitch, loudness, beat or as non-perceptual features as the Mel Frequency Cepstral Coefficients (MFCC):

- Pitch: as the fundamental frequency of the spectrum calculated by the Fourier transformation of the frame
- Loudness: computed from the short time Fourier transform, as the square root of the energy.
- Tone: or brightness and bandwidth. Brightness is a measure of the higher-frequency content of the signal, while Bandwidth can be computed as the magnitude weighted average of the differences between the spectral components and the centroid of the short-time Fourier transform.
- Mel-filtered Cepstral Coefficients (MFCCs): computed by applying a mel-spaced set of triangular filters to the short-time Fourier transform, followed by a discrete cosine transform. Mel filtering is a scaling of frequency that take the human ear sensitive property into account (the human ear is sensitive to linear changes in frequency below 1000 Hz, and logarithmic changes above).
- Derivatives: Time differences of all previous features describe the dynamic behavior of sound, and it can be also useful for MIR.

The audio features can be represented as vectors and used for finding similarities between tracks.

These are only a few of the audio features that can be extracted from tracks. The first research were focused in small set of parameters and excluded other important ones that future works reveal to be very important.

The type of abstract representation and the way to compare them distinguish the different proposed works and targets. So, for example, for fingerprinting the feature vector describes short segments of recordings in a way that is robust to noise or bad recordings but highly accurate for finding exact matches between recordings.

Some methods tried to used the methodologies applied to symbolic data based-content to audio data based-content systems. Like Clausen and Kurth using their set-based method converting PCM signals into sets that can be treated the same way they used sets of notes. İzmirli and Dannenberg [Izmirli10], in 2008, investigated the problem of matching symbolic representations directly to audio based representations for applications that use data from both domains.

Most approaches for music similarity include low-level signal features such as MFCCs that contain little semantic information. Some approaches use symbolic representations, which is difficult to obtain from audio signals, but also difficult to have for large amounts of musical pieces. Marolt in [Marolt06] proposed a bridge approach between audio and symbolic domains, using a mid-level melody-based representation that incorporates melodic, rhythmic and structural aspects of music signal.

3.6.2 Speech Recognition applied to Music

As we see in previous sections, MIR inherited several approaches and methodologies from automatic speech recognition. Maybe the most common and popular ones were

the set of features based on mel-frequency cepstral coefficients (MFCCs) and the probabilistic matching method Hidden Markov Models (HMM).

So, we have seen how Logan [Logan00] in 2000 applied the MFCC to speech vs. music discrimination, achieving accuracy as high as 95%, although that accuracy decreases as the number of audio classes increases, specially if we try to use that methodology genre classification. Mesaros and Astola [Mesaros05] in 2005 also applied the 32 MFCC for singer identification and discrimination. A field between speech recognition and music featuring.

One of the successful methods for speech recognition in the past decades has been the use of HMM. The HMM is a statistical model for sequential data and has been used in MIR for indexing music and finding similarities in melody , automatic segmentation and so on, although most of times applying some new algorithm derivative from the original HMM.

That was the case of Jin et al. [Jin02] in 2002, they used an improved approach of HMM, where they represented songs as string of states based on difference in pitch between two consecutive notes and the duration of the note, and represented all possible state changes using R*-tree.

Sheh and Ellis [Sheh03] in 2003 applied the HMM into chord sequences segmentation and recognition, trying to segment the input musical piece into intervals that correspond to specific words. They used HMM trained with Expectation-Maximization (EM) among other speech recognition tools applied to audio features, and achieved their target, extracting complex musical information such as recognizing chords and finding time alignment in musical pieces.

It is worth to mention the GUIDO system created by Hoos et al. [Hoos01] in 2001. HMM is a probabilistic matching method, so its target is to determine probabilistic properties of query pieces and compare them with corresponding properties of database. Audio features like melody interval sequences, pitch sequences and rhythm can be used to calculate Markov chains, where a state can correspond to features like certain pitch, note duration, etc. And the transition probabilities needed for HMM can be applied to each pair of consecutive states, reflecting the numbers of occurrences of different subsequent states. For calculating distances the GUIDOś method uses transition matrices that can be organized as a tree.

Finally, on the variations of HMM is the Continuous HMM (CHMM), specially for speech recognition tasks with large vocabulary and speaker-independent speech. Roger Jang et al. [Roger05] in 2005 studied the use of CHMM for melody recognition via acoustic input (in other words, the query by singing/humming)

3.6.3 Music Identification and Fingerprinting

One of the most popular branches of MIR, specially in the music business, has been the music identification or music recognition. At the beginning the idea was applied

to based-content searching by song input data by user, but over the years, a new and more powerful target was applied, the Audio Fingerprinting.

As Uitdenbogerd et al. [Uitdenbogerd00] mentioned, the first methods tried to find a matching between melodic information query (usually humming or singing) against a database of MIDI format tracks. There were also some research for style information, but again, using MIDI format. Although MIDI format offered a bunch of possibilities, most of the available tracks were not using that format, so the researching had to move to other type of methodology where attributes such as the pitch, loudness, brightness and bandwidth of musical notes were analyzed.

So, in 2001, Hofmann-Engl [Hofmann01] studied the pitch aspect of melodic similarity (meloton) and the cognitive aspects for the similarity model. In 2002, Feng et al. [Feng02] avoided the pitch tracking on song input tracks extracting and analyzing the singing channels of the audio tracks in database and converting them to self-similarity sequences (using MFCCs) to be used on comparisons with input query.

Until 2003 most of efforts for music recognition was based in a system with a front-end for gathering the acoustic input data as vocal melodic queries (like whistling or singing) and transcribe that input into note sequence, and a second part that was the pattern matching back-end that looks for the musical piece that best matches the input sequence. De Mulder et al. in [Mulder03] worked in the first part in 2003, in the front-end, focusing in segmentation and pitch extraction. Their accuracy for transcribing input to notes was 76% for whistling vocal queries and 85% for the sung ones. Orio et al. [Orio03], also in 2003, presented a similar approach but using hidden Markov model (HMM) for modeling the audio features related to the singing voice, and the Viterbi decoding for transcription. And in 2005 we find the work of Roger Jang et al. [Roger05] for melody recognition via acoustic input using Continuous HMM.

On the other hand, the Acoustic or Audio Fingerprinting is a methodology for identifying perfect matching tracks into databases, what has been widely used in music business for identifying duplicates in their catalogs, or for a user to identify right metadata of his catalog when comparing to a reference catalog, or even for recording and excerpt or short fragment of a track using a cell phone in order to identify the track in a reference catalog. This last target is one of the most famous for common users, specially thanks to successful applications like Shazam. While this application works very well, the first target (finding duplicates) still needs to be improved.

The audio fingerprinting method analyzes tracks and associate each of them with a unique key, a fingerprint of the audio, that consumes less space than original recording. The target of the method is to find duplicates of same songs, but the algorithm needs to achieve also three main properties:

- High execution speed
- Robustness to noise and various types of filtering
- Transparency to encoding formats and associated compression schemes

It has been a bunch of audio fingerprinting algorithms presented at ISMIR during

last years, and although Shazam version is by far the most popular one, it is worth to mention some works on this field. Tsai et al. in [Tsai05], where they presented a work for detecting cover versions of songs. For finding those covers they compared main melodies, and for avoiding problems with different accompaniments between covers, they removed the non-vocal portions of the song, extracted the sung notes from the accompanied vocals, and compare the similarities between the sung note sequences. Angeles et al. in [Angeles10] worked on finding inconsistencies in music libraries through acoustic fingerprinting algorithms. And most recently, Bandera et al. [Bandera11] in 2011 presented a fingerprinting method based on chroma vector to match humming input query with real stereo tracks database.

3.6.4 Music Recommendation

Music similarities have been applied to almost all main lines of MIR, because most of its targets are based on finding some pattern or some similarity between songs in order to categorize them. But, this section is dedicated to what is also called Music Recommendation, i.e. finding those musical pieces in the database that are somehow similar to a query or seed track, in order to recommend the user with similar songs to his taste (usually, the query song).

We could say that first approaches came from based-content searching methodologies. When research were trying to find matches between a song query and database is order to identify the musical piece, they realized that looking for perfect melody matching ended most of times with a list of candidates that had some kind of musical similarity to query. Most of users usually like music of same style, so that finding opened doors for further researching on a powerful and important MIR branch, the Music Recommendation.

One of the main problems of music similarity (and even for genre classification) is how to evaluate the accuracy. There is no ground truth, no objective evaluation criteria or standard test sets we can use. Category definitions are subjective, and they change over time, and there lots of musical pieces that are a mix of different styles. What can be a criteria for finding similarities between two songs (like, for example, type of instrumentation, or rhythm, or even if singer is male or female) can not be a criteria for another person. In early 90' the National Institute of Standards and Technology developed a testing and evaluation paradigm for the text retrieval community, called TREC (*Text REtrieval Conference*). Under that paradigm, each text retrieval team had access to a standardized large-scale test collection of text, set of test queries, and also a standardized evaluation of results. Downie in [Downie03] started the first steps for creating an evaluation paradigm similar to TREC but for MIR.

As Feng et al. mentioned in their article [Feng02], until 2002 most of works in this field were based on pitch-tracking and segmenting into notes or into melody contours the acoustic input singing or humming. Such as the case of Jin et al. [Jin02], applying HMM to representation of music and music similarity, although they only used sung music as query and database. Aucouturier et al. at [Aucouturier02] found that timbre models can be useful for finding audio similarities in polyphonic music and complex

instrumental textures and they presented one of the first approaches to use that property to find similarities between different songs but from same artist or genre. They even introduced the idea of distinguish between finding similar songs or discover new ones (although for them that difference was looking for similarities in content from same artists or in full catalog).

But, music recommendation is a broad concept. What does make two songs similar? At the beginning was clear that the melody, but when we talk about recommending songs to a user based on his taste, we know that a music lover is not always listening to the same melody over and over again, but he can be listening to similar style of music. So, music similarity research had to start looking for audio features that could reflect those other similarities, maybe instrumentation, maybe rhythm, or even if it is a male or female singer. The set of possibilities is large. And, we may not forget it, music is an art of form, so it can be somehow subjective to define how similar are two pieces of music.

So, a new trend appear for using and creating further and extended audio features in order to improve music similarity detection. Rauber et al. [Rauber02] in 2002 used rhythm patterns for organizing and mapping a catalog based on similar sound characteristics using unsupervised neural network (GHSOM). Foote et al. in [Foote02] proposed audio features based on beat spectrum for characterizing rhythm and tempo for audio similarity purpose. Their results could could distinguish between different kinds of rhythms with same tempo, and they were one of the first ones to propose an automatic playlist creation based on rhythmic similarities.

As we can see, the first attempts were using small set of audio features or were focused in only one or two characteristic. Allamanche et al. [Allamanche03] in 2003 expanded that vision for audio features and studied a set of 1000 features for choosing the 20 most useful for music similarity. Pickens and Iliopoulos in [Pickens05-1] worked in music similarity field finding non-independent audio features. One of the main problems until that tie was that often the features were single dimensional (i.e. only pitch, or rhythm, etc) or else assumed to be orthogonal. They used the Markov chain models and Markov random fields, creating a feature vector or chains with only changes in pitch (ignoring the duration of the note and the onsets without pitches), then estimating the state transition distribution based on the relative frequency counts of the extracted chains, and then comparing models for finding similar tracks.

The combination of audio characteristics seemed to be the clue for improving accuracy, so different approaches were presented at ISMIR during last decade. But, in spite of all efforts and improvements in music recommendation, around 2005 the research did not found that perfect set of features that could achieve accuracy above 70 or 80%, so new approaches for mixing methodologies arrived. Stenzel and Kamps [Stenzel05] in 2005 mixed collaborative filtering and based-content technologies for audio similarities, using content-based sound features to train a collaborative model. The bridge between both technologies seemed to be quite successful. Symeonidis et al. [Symeonidis08] in 2008 also explored the mix of based-content data with other manual or human being dependent data, tags. They created tagging prediction system and applied it to music recommendation. And, more recently, Bogdanov and Herrera [Bogdanov11] in 2011 added genre tags to based-content technology music recommendation system.

Last trends give the user the power to fine-tune and refine the results of music recommendation. While Wang et al. in [Wang11] decided to let the user chooses the type of filtering to apply to music recommendation system (they used a tag filtering with multiple levels of preference), and, although the training database was manually tagged, the testing one was untagged, so this approach could be, in theory, used for large untagged catalogs. Wolff and Weyde in [Wolff11] explored the fine tuning of generic techniques to user taste, using machine-learning techniques for analyzing user data that specifies song similarity.

In general, music similarity field needs to extract the audio features of every track, and then find a way to compare those features. The simple and first approach placed the feature values into vectors and then calculates an Euclidean distance between points. But Euclidean distance assumes that features are independent, orthogonal and that all features are equally important, what is unlikely to be true with audio features. So, Slaney et al. in [Slaney08] studied in depth metrics for music similarity comparing five different approaches, and propose principled means of finding the appropriate weighting for features. The five approaches studied for Slaney et al. are a good representation of the varied algorithms using Euclidean distance. They grouped them into 2 main types of algorithms:

- **Algorithms based on second-order statistic**s: The algorithm learns a linear transformation of the input space based on second-order statistics of the feature vectors.
  - Whitening algorithm removes an arbitrary scale that the various features might have. It scales all input features equally.
  - LDA (Linear Discriminant Analysis) is used for finding the optimal dimensions to project data and classify it. It rotates data and projects it into a lower-dimensional space for dimensionality reduction. It assumes that data is labeled with two classes, and that each class is distributed with a Gaussian distribution, and also that each class shares the same covariance matrix, what is not always true in music since some artists and albums are more diverse than others.
  - RCA (Relevant Component Analysis) is related to whitening and LDA and it is entirely based on second-order statistics of the input data.
- **Algorithms based on optimization**: They optimize a nearest-neighbor classifier with an objective function that is highly and non-continuous and non-differentiable.
  - NCA (Neighborhood Component Analysis) uses distance to the query point instead of hard decision of identifying nearest neighbors.
  - LMNN (Large Margin Nearest Neighbor), as NCA, is based on an optimization problem, but it mimics the kNN leave-one-out classification error with a piecewise linear convex function.

3.6.5 Artist Similarity

Artist similarity is widely used in collaborative filtering for music recommendation, so it has been several attempts to reproduce that feature using based-content technology, although with less success than with music recommendation. The lack of success in this matter is probably due to the fact that all tracks from same artist are not from same style and instrumentation (specially for those artists with long active artistic career).

In any case, one of the first proposals arrived in year 2000. Ellis et al. [Ellis02] explored the possibility of finding similarities between artists, although they ended with only a possible definition of consistent measures over set of artists to be used as training data for automatic similarity measures based on audio data. Up to date, this is still one of most difficult methodologies, since similarity between artists depends in multiple dimensions: culture, language, genre, instrumentation and so on. Some of those dimensions are based on audio characteristics of their music, but some not. And even taking only audio characteristics into account, there are some artists that have changed significantly their style over the years. So, not everyone agrees when we say that two artists are similar.

Berenzweig et al. [Berenzweig03] in 2003 studied acoustic and subjective approaches for calculating music similarity between artists. They compared MFCC against an intermediate anchor-space of genre classification, and also used subjective techniques in as databases tagged by human beings, playlist, personal collections and web-text mining for evaluating results.

Mandel and Ellis in [Mandel05] used features calculated over entire lengths of songs (instead of short-time features), and Support Vector Machines as classifier, for artist identification purpose.

Finally, in 2011, Bryan and Wang [Bryan11] made a step forward for understanding the music influences based on artist and genre.

3.6.6 People behavior and perception of music

Kim and Belkin [Kim02] presented an interesting research on people behavior for music. They investigated the people music information needs through their perception of music. Most researching were based on finding the best and most accurate system to represent music and create tools and algorithms for finding relationships between musical pieces like similarity, covers, genre classification, etc. The Kim and Belkin approach was different, they thought it could be more interesting and useful to first find the needs of people, and then develop the appropriated tools. And they were right. Although the experiment was quite limited because only used a small set of classical music pieces, the experiment shown that people trend to use the same group of categories when using words for describing music and for searching for music, and those categories are the same in both cases. The experiment also shown that most of common people without music background never use formal features of music informa-

tion, and that their preferred category for searching for music was the one related to occasions and events (like, for example, for celebration, Saturday at art gallery, for a party, etc.)

Lartillot in [Lartillot03] made an interesting approach for discovering musical patterns in automated way. He thought that MIR should emulate the human behavior for finding similarities and listening to music, so he tried to find patterns in musical pieces. Although that idea was quite interesting, it did not have too much influence in the MIR.

3.7 Genre classification

Automatic categorization of tracks into genre, mood or any other category is based mainly in two steps: feature selection and classification. The types of features to extract or the way they are calculated has been changing and evolving during last decade. And also the way and classification method to apply.

One of the first approaches for musical genre classification was Tzanetakis et al. [Tzanetakis01] in 2001. They used a statistical pattern recognition classifier with a small set of parameters for representing music surface and rhythmic structure of tracks, such as shape and brightness of spectral, the FFT magnitude, the amount of noise and the amount of energy. It was a good starting with an average accuracy of 62% for some main genres.

One of the problems with music classification methodologies is the non stable performance in accuracy across different music styles. So that, classical music usually has high accuracy values, while others music styles like R&B show poor values. That caused that several researching focused their works in specific music styles. Such as the case of Dixon et al. [Dixon03] in 2003, who compared two methods for finding timing patterns for recognizing the style of music for a set of Latin ballroom dance music. But, even making the assumption that genre classification method depends on music style, there is still another factor to take into account: music styles differ a lot between Western and non-Western music.

Norowi et al. in [Norowi05] studied the factors affecting automated genre classification, introducing non-western genres to data sets. Their results shown that various factors such as the musical features extracted, classifiers employed, the size of the data set, excerpt length, excerpt location and test set parameters improve classification results. Summarizing their conclusions; the data-set size has to be large, the track length (the excerpt length) does not need to be long (around 30-60 seconds is enough), the track location is important (the first few seconds of a song are crucial genre indicator), the classifier must be chosen according to specific needs to ensure higher results accuracy, and higher cross validation folds may not always attain better results (six-folds seems to be the best one). Cross-validation is a standard evaluation technique in pattern classification, in which the dataset is split into n parts (folds) of equal size. $n-1$ folds are used to train the classifier. The nth fold that was held out is then used to

test it.

As in most of MIR fields, selecting the right audio features has been crucial and an important part of genre classification methodologies. McKinney et al. [McKinney03], in 2003, studied four different audio features sets for genre classification (low-level signal properties, mel-frequency spectral coefficients and two sets based on perceptual models of hearing). Scaringella and Zoia in [Scaringella05] worked on low-level temporal relationships between chunks for genre classification. Their target was to consolidate classification consistency by reducing ambiguities, so they compared 5 different methods taking low-level and short-term time relationships into account to classify audio excerpts into musical genres. Their results shown an average accuracy of almost 70% with a data set of 7 genres, although they clearly shown that some classification algorithms seem more suitable to some particular genres. Wülfing and Riedmiller [Wulfing12] concluded that time-frequency features perform better accuracy than frame level features. Hoffman et al. [Hoffman08] computed timbre similarity using the nonparametric Bayesian model Hierarchical Dirichlet Process (HDP) for discovering the latent structure in MFCC feature data and applied the results to music classification. Their results were compared to other methods like G1 and concluded that HDP achieved better results and performance, as well as avoided the hub issue (G1 models each song's distribution over feature vectors using a single multivariate Gaussian distribution with full covariance matrix. This methodology produces hubs)

During 2000 decade, with the growing type of audio features available, the trend was to select the right ones to avoid over-fitting in the data, although Fiebrink and Fujinaga [Fiebrink06] thought that it was not needed, and that a simple PCA approach could solve that issue.

Pampalk et al. in [Pampalk05-1] combined spectral similarity (which describes aspects related to timbre) with fluctuation patterns (which describe periodic loudness fluctuations over time) and two new descriptors derived thereof (focus and gravity of fluctuation patterns) for genre classification. Their test database used four different music collections (with a total of 6000 tracks) with up to 22 genres per collections. Although they claimed to increase of 14% the average of genre classification performance comparing to latest ISMIR researching, they said that glass ceiling for this field was being achieved. Acoutourier and Pachet already talked about that glass ceiling in 2004, in their article *Improving timbre similarity: How high is the sky?*, presented at *Journal of Negative Results in Speech and Audio Sciences*. According to research done and papers presented during next years, that statement was not true. In 2010, there was still a paper presented to ISMIR that proposed a method for choosing the right audio features for genre classification (Hamel and Eck [Hamel10]).

Tracks from same genre usually shared certain common characteristics like similar type of instruments, similar pitch distribution and rhythmic patterns. Lampropoulos et al. in [Lampropoulos05] presented a system for musical genre classification based on audio features to distinguish musical instrument sources. They proposed Convolutive Sparse Coding (CSC) algorithm applied to several portions of audio signal. They decompose the input audio signal into a number of component signals, each of which corresponds to a different musical instrument source. Then they extracted timbre, rhythmic and pitch features from separated instrumental sources and used it for music

classification. It was an interesting approach that tried to mimic a human listener, who determine the genre of a music piece at the same time that (or maybe also thanks to) identify number of different musical instruments in a complex sound structure. Another important thing of their approach was that they had their best accuracy applying the CSC algorithm to main three parts of the music piece (beginning, middle and the end) instead of the whole track. Not all the same instruments are active throughout all musical piece, so taking into account the three main part of the track can identify better the most important instruments. CSC is an evolution of Independent Component Analysis (ICA) and Independent Subspace Analysis (ISA) methods, both of them are methods that separate individual sources from a single-channel mixture by using sound spectra, but making the assumption of signal independence, what is not always true in musical signals. CSC overcomes that limitation. They used training and testing database with 4 Greek musical genres, and their accuracy achieved around 75%.

If most of research for music similarity or music recommendation were based on unsupervised clustering techniques of music collections, the first years of researching in genre classification were more focused on supervised techniques. The supervised clustering methods start from a given genre classification and then try to map the unclassified content into it using machine-learning algorithms. That approach implies that new genres can not be detected, they will always being classified in one of the given ones (i.e. wrongly classified). In 2005, Flexer et al. in [Flexer05] introduced what they called novelty detection, an automatic identification of new data not covered by training data. For example, if a the training data has 3 different genres, and we try to add a track from a fourth genre, the system would detect it and tagged it as novel data instead of trying to classify it in wrong way. They applied the methodology in genre classification, although it could be used for music similarity also. The accuracy of the system was not bad, but it has too high false positive values in the opinion of the author.

Some of the unsupervised methods applied to genre classification included SVM for automated hierarchical music classification (Chinthaka Maddage et al. [Chinthaka03]), or for modeling short time features for genre classification (Meng and Shawe-Taylor [Meng05]), the k-means for an unsupervised feature learning methodology (Wülfing and Riedmiller [Wulfing12])

The probabilistic model has been also used in this field. Flexer et al. [Flexer06] is one the already mentioned cases, but another one worth to mention is Reed and Lee [Reed06]. They presented in 2006 an algorithm that uses an efficient segmentation of HMM modeling for genre classification, instead of applying the HMM to whole song like previous works. The idea was very logical: if we compare the genre classification with language recognition, applying HMM to whole track is the same that having a single HMM for an entire spoken document, so it should not work. Anyhow, their results were comparable to past solutions on this field, i.e. their accuracy was not higher.

Panagakis et al. [Panagakis08], in 2008, proposed a multilinear framework for automatic genre classification that was inspired by a model of auditory cortical processing. They extract multi-scale spectro-temporal modulation features from the cortical repre-

sentation of sound using three multilinear subspace analysis techniques. This was the first time that spectro-temporal modulation features were used for genre classification, and although they performance right, they did not improve any accuracy achieved in previous works with other methodologies.

Finally, the author would like to stand out, again, the work of Draman et al. [Draman10] proposing Artificial Immune System (AIS) domain applied to music classification. As we will see in further sections this is a revolutionary approach that can suppose a step forward in genre classification

3.8 Summary

It is interesting to see how the evolution of MIR research and paper presented at ISMIR during last 12 years reflect the evolution of music business and music customer needs.

The digital era shook up the music business and forced them to change the industry and the way to catch customers. They are still trying to find the best approach for matching industry and customers needs, but for first time in the history the customer is leading this field.

With large music databases available in internet, and the changes in the user behavior, who has discovered that there is a huge amount of music that can fit his taste, the needs for creating tools that help the user for finding right content and help the industry to offer their content and organize their catalogs are becoming and standard in music world.

Speech recognition needs created a bunch of useful methodologies and a success in that field that made everyone thought that could be also possible to apply that success to music in an easy way. 12 years later MIR research are still trying to find the best tools, or at least those tools that have accuracy enough for being accepted for most past of users.

Starting from music vs. speech discrimination in order to identify that audio content in catalogs that were not useful for music systems, the MIR evolved to creating useful tools for music distributors, such as finding the most representative excerpt of the musical piece. One of the first targets of MIR for helping music distributors were to find right methods for music categorization, and although huge steps forward have been done in that field, currently we are still trying to find a robust system for any number and kind of musical genre or any other categorization. Have we achieved the limit on that? I don't think so, but it is possible that we need a different approach if we want to go further on that.

As we also need to find different approaches for music recommendation. Cold start and long tail issues are fewer issues nowadays thanks to technology and research, but music recommendation still needs to improve the accuracy and performance for large catalogs.

The fact that some industry products that are not based-content have become very popular among the world is not helping MIR in based-content technologies. Products like Pandora or last.fm are both very useful, but cannot sort out the long tail issue or even are scaled to large catalogs. Pandora uses a 600.000 tracks database, and needs music experts working and tagging each piece at the rate of 20 minutes per piece in the best case. How could they extend their method to common aggregator's catalog with more than 20 million songs? On the other hand, last.fm, based on collaborative filtering, can be scaled on condition that every piece of music has to be streamed and tagged by someone. Long tail issue is more than issue in that scene, with large catalogs is quite likely that most of the content will be never used.

Chapter 4

Riemann Metrics

4.1 Introduction

Most of machine learning systems, and in extension most of MIR systems, have to deal with huge amounts of data that are difficult to interpret. Every input item needs to be represented by large amount of variables and parameters, what makes any operation with that data complex. Using large amount of input variables usually adds some data correlation that also makes the output of the system more difficult to understand. Some variables can be important for some type of output, while the less important ones can add some special extra value to the output. How can we find the right variables that will contribute the most to the final target of the system? That is the main goal of feature selection techniques, and it is not a trivial problem as we will see in 4.2 *Feature Selection* section.

Once we find the best features, or the best combination of them that maximizes the target of the system, we will still have a complex output model in most of the cases. Dimensionality reduction methods reduce the high number of dimensions of input data space to a fewer dimensions one, where we can get better performance due to reducing noise, correlation and useless data. Furthermore, we can incorporate domain knowledge into dimensionality reduction method that can help the final target of the system. I will make a short review on main dimensionality reduction techniques in 4.3 *Feature Extraction* section, and then I will make a deeper review on PCA and ICA methods in sections 4.4 *Principal Component Analysis* and 4.5 *Independent Component Analysis*, before moving to Riemann surfaces, because those methods have been applied to this thesis in some of its phases.

After applying dimensionality reduction method we can still find that we have a non-basic final geometry of the output model, but we need to deal and interpret that final geometry in order to achieve the final target of the system. In the case of this dissertation, that final geometry has to be the environment where we can find

audio similarities between tracks, so it is important we can work and interpret them well. After several phases of the researching, I found that manifolds and specifically Riemann surfaces and its associated discrete Riemann metrics could help on my goal. I will briefly explain what is a manifold and a Riemann surface in section 4.6, as well as the Discrete Riemann Metrics in section 4.7, for then going deeper into my implementation and process for Music Recommendation algorithm.

4.2 Feature Selection

A common problem in machine learning is the feature selection, or in other words, how to select the most relevant features for creating the classification model. When using hundreds or even thousands of features from each sample or input data, you can expect to have high level of correlation among different features, what can add huge amount of redundant information, as well as some level of irrelevant and no useful information for some context, therefore selecting the minimum amount of features that contain the maximum information to use in the model is a mandatory target.

Feature selection techniques are a subset of the feature extraction. It is also part of the analysis process, showing which features are important for output prediction, and how they are related.

As Guyon et al. explained in [Guyon03], there are three main benefits of using feature selection techniques:

- improving the prediction by reducing data correlation,
- providing faster and more cost-effective predictors by reducing the measurement and storage requirements as well as reducing training times,
- and providing a better understanding of the underlying process that generated the data, as well as facilitating data visualization.

How to find the best subset of features is not a trivial problem. An exhaustive searching of each possible subset of features is not possible most of times. Furthermore, the main target is to find that subset of variables that together have a good predictive power, and avoid ranking variables based on their individual predictive power. Therefore, you need to apply some of the feature selection algorithms, which mainly can be one of the three following categories: filters, wrappers and embedded methods.

Filters methods select subsets of variables as pre-processing step, without taking into account the target predictor. The selection of variables is made by ranking them with correlation coefficients, measuring the usefulness instead of the error rate for scoring a feature subset. This measure is chosen to be fast to be computed. Common measures include the Mutual Information, Pearson product-moment correlation coefficient, and the inter/intra class distances. Since the predictive model is not taken into account, there are many filters that produce a subset ranking as output, instead of the best feature subset that fits the model.

Wrapper methods are similar to Filters in the searching approach, although Filters evaluate a simple filter while wrappers evaluate against a model. Wrapper methods evaluate subsets of variables based on how useful they are for a given predictor. Although that predictor is used as black box, it scores each subset based on their predictive power. The wrapper methods train a new model for each subset. An exhaustive search of all possible subsets is known to be NP-hard, what makes the search quickly computationally intensive and even intractable, but using efficient search strategies can solve that problem. *Greedy* search strategies can be one of those efficient cases, which iteratively evaluates a candidate subset, then modifies that subset and check if the new subset is better or worse than previous one. The greedy search can include forward selection or backward elimination: forward selection adds variables progressively into larger and larger subsets, while backward elimination starts with variables and progressively removes the least promising ones. Most of times a stopping criterion is used to finish the feature selection technique; a threshold achieved, a maximum run time surpassed, or even a maximum number of iterations done. It does not matter which method is used to finish the algorithm, the subset of features with highest score found up to that point is selected as the final one.

Embedded methods are based on same idea as wrapper ones, but optimized using two-part objective function with a reward for a good fit and a penalty for a large number of variables. They incorporate variable selection as part of the training process, what removes the need of splitting training data into training and validation set, and avoids retraining the system from scratch for every new subset. One popular approach of embedded method is the Recursive Feature Elimination algorithm, commonly used with Support Vector Machines to repeatedly construct a model and remove features with low weights. These approaches tend to be between filters and wrappers in terms of computational complexity. Embedded techniques are embedded and specific to a model.

4.3 Feature extraction

In machine learning the quality of the input data is one of the key points of the process. Using input data with as much information as possible seems to be the best option, but most of times that implies using large input data sets that can add some redundancy to data, as well as make their processing computationally expensive and intensive. Dimensionality reduction can help on that matter by reducing the input data space from high number of dimensions to a fewer dimensions one. We can achieve better performance using features derivative from the original input data, furthermore, we can incorporate domain knowledge into derivative process that can help the target of the predictor. Transforming input data into a lower dimensions space (into set of features) is called feature extraction or feature construction.

The main goal of feature extraction is reducing the amount data by extracting the most relevant information from input data and achieved the best performance and accuracy of the machine learning system. Feature construction can be used for reconstructing data in better way, or for finding the best efficiency on making predictions. Both goals have different approaches: reconstructing data is a unsupervised learning

problem because it is related to data compression, while improving prediction efficiency is a supervised learning problem.

There are several types of dimensionality reduction techniques: clustering, basic linear transforms of the input variables, non-linear transformations, multi-linear subspace learning through tensor representations and other. As an overview of the possibilities, in the following list there is a brief description of some of those dimensionality reductions techniques, for then getting further explanation of the PCA and ICA techniques in the following sections.

- Clustering: The idea of using clustering for feature extraction is to assume that similar features will fall into same cluster, so we can replace that by the corresponding cluster centroid, which becomes a new output feature. Clustering is usually associated with unsupervised learning systems, although it can be improved for feature extraction purposes adding some supervision in the clustering procedure, that way we can obtain more discriminant features.
- Principal component analysis (PCA): Basic linear transforms of the input variables. This method can reduce the input dataset into two dimensions. Other linear transform methods and unsupervised method of feature construction can be Singular Value Decomposition (SVD), LDA. SVD creates a set of features that are linear combination of the original variables. More sophisticated linear transforms can be spectral transforms (Fourier, Hadamard), wavelet transforms or convolutions of kernels.
- Kernel PCA: An extension of PCA that uses kernel methods and non-linear mapping. It does not compute the principal components themselves, but the projections of the input data onto those components.
- Multilinear PCA: An extension of PCA that uses multi-linear subspace learning methodology. It uses multiple orthogonal (linear) transformations, one per each dimension. The main difference with PCA is that instead of reshaping a multidimensional object into a vector, it operates directly on multidimensional objects, what makes multi-linear PCA more efficient because the number estimations is very reduced. This transformation aims to capture as high a variance as possible, accounting for as much of the variability in the data as possible,
- Semi-definite embedding (SDE) or maximum variance unfolding (MVU): Non-linear dimensionality reduction. MVU is also viewed as a non-linear generalization of PCA. Non-linear dimensionality reduction methods can be classified into those that just give a visualization, and those that provide a mapping between two dimension spaces (from high to low dimensional embedding or vice versa). The mapping methods can be also used for feature extraction.
- Multifactor dimensionality reduction: A data mining approach for detecting patterns in the datasets and transform those datasets into reduce and more meaningful structure, a combination of independent variables.
- Multilinear subspace learning (MSL): A dimensionality reduction technique for finding the low-dimensional representation that matches certain set of characteristics. A direct mapping without using vectorization. It uses multi-linear projection, i.e. it maps from high-dimensional tensor space to a low-dimensional tensor space. The term tensor in MSL refers to multidimensional arrays. The

mapping from a high-dimensional tensor space to a low-dimensional tensor space or vector space is named as multi-linear projection. MSL methods are higher-order generalizations of linear subspace learning methods such as PCA, linear discriminant analysis (LDA) and canonical correlation analysis (CCA). In the literature, MSL is also referred to as tensor subspace learning or tensor subspace analysis.

- Nonlinear dimensionality reduction: This method assumes that the data of interest lies on an embedded non-linear manifold within the higher-dimensional space. There are lots of non-linear methods, some of them can provide a mapping between high and low dimensional space embedded (and vice versa), and some of them can be used only for a visualization. Usually the ones that offer mapping are used on feature extraction.
- Isomap: A Nonlinear dimensionality reduction method, and also a low-dimensional embedding method and a representative of isometric mapping methods. It uses a neighborhood graph embedded in the classical scaling, which creates a weighted graph where the method uses geodesic distances instead of Euclidean distances used in basic PCA. That way a manifold structure can be incorporated in the resulting embedding. The geodesic distance is defined by the sum of edge weights along the shortest path between two nodes. Then the top n eigenvectors of the geodesic distance matrix are chosen as coordinates of the new n-dimensional Euclidean space.
- Autoencoder: A neural network trained to map a vector of values to the same vector, i.e. to approximate the identity function. When it is applied to feature extraction, a new layer is added to network for forcing the mapping between high to low-dimensional space (to vector with small number of network units). The method allows the mapping in both directions (from high to low and vice versa).

4.4 Principal Component Analysis

Principal Component Analysis (PCA) is one of the basic linear transforms methods for dimensionality reduction, which allows the extraction of relevant information from confusing data sets. It is one of the simplest and more robust methods for those purposes.

The PCA methodology is based on the assumption of linearity of data, which restricts the set of potential bases, as well as formalizes the assumption of continuity in data set. PCA finds a linear combination of the original basis that identify the dependencies and possible redundancy among original variables; at the same time that makes the dimensionality reduction transforming a high-dimensional data into a lower-dimensional form, keeping the most meaningful data and minimizing data loss. The linear combination is calculated through orthogonal transformations that convert the correlated origin values into a set of values linearly uncorrelated that are called principal components. The final set of principal components is ordered by variance of data (the greatest variance by some projection of the data lies on the first coordinate), and each one is orthogonal to previous components (i.e. uncorrelated among them).

The final number of principal components are always less than or equal to the number of original variables, and all of them are independent if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables. Further explanation of the methodology has been already explained in Chapter 2, section 2.8 *Principal Component Analysis applied to descriptors*

If the data of interest lies into linear structure, PCA can help to find the mapping between higher to lower dimension space. The result is like a projection or shadow of the original object into final low-dimension space. But that does not cover all types of input data structure; when that data lies into non linear structure, we cannot apply a direct mapping using PCA, we need to apply some extension of the method, some nonlinear generalization, like Kernel PCA.

Kernel PCA uses kernel methods for transforming the input data into a high dimensional feature space, for example mapping the input data first to some nonlinear space, and then applies the PCA to find the principal components. The result will not be linear in original data space. Some of the most popular kernels to apply include Gaussian, Polynomial and hyperbolic tangent.

Further nonlinear generalization of PCA is based on principal curves and manifolds for creating nonlinear dimensionality reduction methods constructing and embedded manifold for data approximation, and encoding data using standard geometric projection onto the manifold.

Principal manifolds were introduced by Hastie and Stueltze in 1989. They described them as lines or surfaces passing through the middle of the data distribution. During mid 1990s, Elastic map algorithms were introduced for principal manifold construction in the context of neural network methodology. Elastic map algorithm is based on analogy of principal manifold and elastic membrane, you can construct the principal manifolds generating the grid approximation of the continuous manifold. We will see further explanation of manifolds in section 4.6 *Riemann Surface*.

4.5 Independent Component Analysis

Most of times in audio processing, the input signal is not measured independently, it is not recorded in isolated way, but we have a superposition of signals. For example when you try to record someone speaking on the street, you will probably have environment noise added to the human voice. That problem is known as linear mixture problem. The methodologies for separating the mixed input sources are known as blind source/signal separation (BSS).

Independent Component Analysis (ICA) is an algorithm to solve BSS problems that came from a linear mixture. Their goal is to recover linear mixed signals and their underlying sources without knowledge of the linear mixture.

ICA is broadly used in signal processing and machine learning for filtering signals

and for applying dimensional reduction, considering that identifying the linear mixture (the basis of independent components) provides a way for selecting the individual sources of interest.

For achieving the best results using ICA on mixed signals there are two assumptions needed: the source signals have to be statistically independent of each other; and the distribution of the values in each source signal should be non-Gaussian.

There are three effects of mixing source signals that we may take into account: independence, normality and complexity. although the source signals are independent, their signal mixtures are not; although the source signals do not follow a Gaussian distribution, the sum of independent random variables tends towards a Gaussian distribution; although independent and non-Gaussian signal has a low complexity, any signal mixture has a temporal complexity greater than original source signals. Those effects are taken into account as basic establishment for ICA: *if the extracted signals from a set of mixtures are independent like source signals, or have a non-Gaussian histograms like source signals, or have low complexity like source signals, then they must be source signals.*

The source signals are also called factors, latent variables or *independent components.*

ICA finds the independent components by maximizing the statistical independence of the estimated components. Independence on components are based on the two assumptions mentioned above, so for maximizing the statistical independence of the estimated components, ICA looks for the minimization of mutual information, and the maximization of non-Gaussianity. The minimization of mutual information uses measures like divergence and maximum entropy, while the maximization of non-Gaussianity uses kurtosis and negentropy (negative entropy).

The different ICA approaches and algorithms are based on the different methods you can use for minimizing mutual information among independent components, on the methods for maximizing the non-Gaussianity, as well as on how they define independence of the estimated components. Other differences among algorithms can also come from the preprocessing of data; some uses centering of data subtracting the mean to create a zero mean signal, others uses some basic dimensionality reduction in order to simplify and reduce the complexity of the problem, while others use whitening for ensuring that all dimensions will be treated equally a priori before the ICA methodology starts the process.

4.6 Riemann Surface

Most of the algorithms mentioned in previous sections can not be used or have low accuracy when we have some specific types of surfaces, or some multi-valued functions such as the square root or even the logarithm. Riemann surfaces can help on the study of those type of functions.

Riemann surface is a one-dimensional complex manifold, so accordingly, a two-dimensional real analytic manifold. A two-dimensional real manifold can be transformed in a Riemann surface if, and only if, the surface is directional and metrizable. We can define an directional surface such a surface that has a continuous choice of clockwise rotations on it. So, for example, sphere and torus are directional and will work under complex manifolds, while Möbbius strip or Klein bottle will not. An oriented manifold of (real) dimension two is a two-sided surface. Möbbius strip is a one side surface, if you starts drawing a line in the surface and continues along all strip, you will be back to origin and all surface will be drawn with a line (if two-sides surface you would draw only one side). Complex plane has a natural orientation where the multiplication by i is a counter clockwise rotation, that is the reason why we need directional surfaces will allow complex analysis on them, like the Riemann surfaces. A Riemann surface is an oriented manifold of (real) dimension two together with a conformal structure. That implies that the structure should allow angle measurement on the manifold, the basis of Riemannian metrics.

For complex charts f and g with transition function $h = f(g^{-1}(z))$ we can consider h as a map from an open set $\mathbf{R}^2$ to $\mathbf{R}^2$ whose Jacobian in a point z s just the real linear map given by multiplication by the complex number $h'(z)$. However, the real determinant of multiplication by a complex number α equals $|\alpha|^2$, so the Jacobian of h has positive determinant. Consequently the complex atlas is an oriented atlas.

One of the central objects in complex analysis are the holomorphic functions. An holomorphic function is a function with complex values, with one or more complex variables, that is complex differentiable in a neighborhood of every point in its domain. Being complex differentiable in a neighborhood implies that any holomorphic function is infinitely differentiable and equal to its own Taylor series. The holomorphic functions can be defined between them, and this is one of the main points of Riemann surfaces. Actually, nowadays Riemann surfaces are considered the common and natural setting for studying the holomorphic functions, especially the multi-valued ones.

Every Riemann surface is a surface of two-dimensional real space, but with a complex structure, which is needed for the definition of holomorphic functions.

In topology, a homeomorphism is a continuous function between topological spaces that has a continuous inverse function. Like the Greek origin of the word homeomorphism means, it is related to similar shapes. Homeomorphisms are mappings that preserve all the topological properties of a given space. If two spaces have a homeomorphism between them, they are called homeomorphic, and from topological point of view they are the same. For example square and circle are homeomorphic to each other, but sphere and donut are not. The fist ones can be transformed one on the other by stretching, bending and transforming to the new shape.

We can describe a manifold using an atlas, which is based on charts. A chart for a topological space M is a homeomorphism φ from an open subset U of M to an open subset of Euclidean space. Charts are traditionally recorded as the ordered pair (U, φ). The charts for a topological space are also called coordinate chart or coordinate map. An atlas for a topological space M is a collection of $(U_\alpha, \varphi_\alpha)$ of charts on M such that $\bigcup U_\alpha = M$ In other words, an atlas consist of individual charts that describe individual

regions of the manifold (like Earth and Atlas). A transition map offers a way to of comparing two charts of an atlas.

A complex manifold of complex dimension one (i.e. Riemann surfaces) is a Hausdorff topological space with an atlas. A Hausdorff space is topological space in which distinct points have disjoint neighborhoods, and where any two distinct points of the topological space can be separated by neighborhoods. Their atlas can be defined as follows: for every point $x \in X$ there is a neighborhood containing x homeomorphic to the unit disk of the complex plane (understanding by unit disk the set of points with distance to x is less than 1). The map carrying the structure of the complex plane to the Riemann surface is called a chart. Also, the transition maps between two overlapping charts are required to be holomorphic.

A function $f : M \to N$ between two Riemann surfaces M and N is called holomorphic if for every chart g in the atlas of M, and every chart h in the atlas N, the map $h \circ f \circ g^{-}1$ is holomorphic wherever it is defined. The composition of two holomorphic maps is holomorphic.

Riemann surfaces are given by locally patching charts. Compact Riemann surfaces can be given by polynomial equations inside a projective space. That in conjunction with compactness condition, makes the surface algebraic. This feature of Riemann surfaces allows to study them with either the means of analytic or algebraic geometry.

A Riemann surface is a directional manifold with a countable base, therefore, it is separable and a metric can be applied to it.

4.6.1 Classification of Riemann surfaces

The Riemann surfaces can be classified according to the uniformization theorem: every simply connected Riemann surface is comformally equivalent to one of the following three domains: the Riemann sphere, the complex plane, or the open unit disk. That admits the Riemannian metric of constant curvature (which can be 1, 0, and -1), but also allows to classify the Riemann surfaces as elliptic (positively curved), parabolic (flat, zero curvature), and hyperbolic (negatively curved). The corresponding surfaces are also called elliptic, parabolic and hyperbolic respectively.

Each category will have different genus g. The genus of a connected, directional surface is an integer that represents the maximum number of cuttings along non-intersecting closed simple curves without rendering the resultant manifold disconnected. It is equal to the number of handles on it; It is also equal to the number of holes in the directional surface.

Elliptic Riemann surfaces The surfaces with constant curvature +1. The Riemann sphere $C \cup \infty$ is the only case (see figure 4.1). Its genus is zero.

Parabolic Riemann surfaces The surfaces with constant curvature 0. They can be the plane itself, a cylinder (the punctured plane), or a torus. Torus are the only

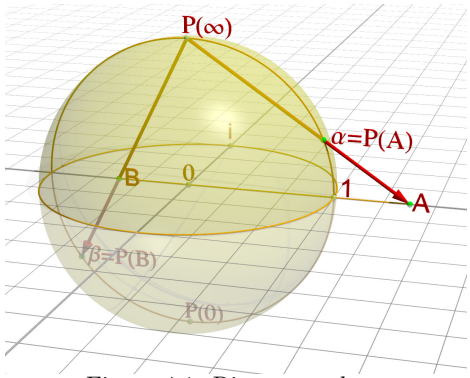


Figure 4.1: Riemann sphere

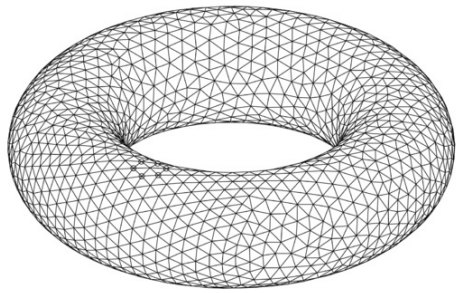


Figure 4.2: Torus

Riemann surfaces of genus one (see figure 4.2).

Hyperbolic Riemann surfaces The surfaces with curvature -1. This is the group with more examples. Hyperbolic Riemann surfaces have genus greater than 1. It is based on unit disk: any simply connected strict subset of the complex plane is biholomorphic to the unit disk, therefore we can describe the local model of any hyperbolic Riemann surface through the open disk with Poincaré-metric of constant curvature -1. Poincaré-metric is the metric tensor describing a two-dimensional surface of constant negative curvature. According to the uniformization theorem above, all hyperbolic surfaces are quotients of the unit disk.

Unlike elliptic and parabolic surfaces, no classification of the hyperbolic surfaces is possible.

When a hyperbolic surface is compact, then the total area of the surface is $4\pi(g-1)$, where g is the genus of the surface.

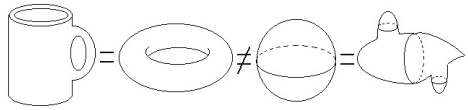


Figure 4.3: Coffee mug and torus share same topology, while torus and sphere have different topology

4.7 Discrete Riemann metrics

4.7.1 Manifolds

Quoting Rowland in [Rowland12], a manifold is a *topological space locally Euclidean*, i.e. around every point, there is a neighborhood that is topologically the same as the open unit ball in $\mathbb{R}^n$. Any object that can be charted is a manifold. The way or how we differentiate between manifolds is the main target of the topology. The properties of a manifold that do not change under continuous deformations make up the manifold's topology. For example, figure 4.3 shows that the surface of a coffee mug with a handle is topologically the same as the surface of the donut or torus, so they share the same topology (one can be continuously stretched into the other), but a torus and sphere are topologically different, therefore we cannot remove the whole of the torus while stretching.

As a topological space, a manifold can be compact or non compact, connected or disconnected, with or without boundary. Usually, open manifold refers to a non-compact manifold without boundary, while closed manifold refers to a compact manifold with boundary.

By default, a manifold has finite dimension n, for n a positive integer.

For understanding the concept of manifold we need to define what is a coordinate chart: a coordinate chart on a set X is a subset $U \subseteq X$ together with a bijection

$$\varphi : U \to \varphi(U) \subseteq \mathbf{R}^n \tag{4.1}$$

onto an open set $\varphi(U)$ in $\mathbf{R}^n$. Therefore we can parametrize points x of U by n coordinates $\varphi(x) = (x_1, ..., x_n)$.

Furthermore, and n-dimensional atlas on X is a collection of coordinate charts $\{U_\alpha, \varphi_\alpha\}_{\alpha \in I}$ such that:

- X is covered by the $\{U_\alpha\}_{\alpha \in I}$
- for each $\alpha, \beta \in I, \varphi_\alpha(U_\alpha \cap U_\beta)$ is open in $\mathbf{R}^n$
- the map

$$\varphi_\beta \varphi_\alpha^{-1} : \varphi_\alpha(U_\alpha \cap U_\beta) \to \varphi_\beta(U_\alpha \cap U_\beta) \tag{4.2}$$

is C^∞ with C^∞ inverse.

$F(x_1, ..., x_n) \in \mathbf{R}^n$ is C^∞ if it has derivatives of all orders. In that case, we can say that F is *smooth*.

Two atlases $\{(U_\alpha, \varphi_\alpha)\}, \{(V_i, \psi)\}$ are compatible if their union is an atlas. That means that all the extra maps $\psi_i \varphi_\alpha^{-1}$ must be smooth.

A differentiable structure on X is an equivalence class of atlases.

Taking all previous definitions into account we can now define an *n-dimensional differentiable manifold* as a space X with a differentiable structure. In other words, to prove that something is a manifold, you only need to find one atlas.

If the overlapping of the manifold's charts are smoothly related to each other, the manifolds are differentiable and the are called smooth manifolds. That means that the inverse of one chart followed by the other is an infinitely differentiable map from Euclidean space to itself. More academic definition would be as following: A map $F : M \to N$ of manifolds is a smooth map if for each point $x \in M$ and chart $(U_\alpha, \varphi_\alpha)$ in M with $x \in U_\alpha$ and chart (V_i, ψ_i) of N with $F(x) \in V_i$, the set $F^{-1}(V_i)$ is open the composite function

$$\psi F \varphi_\alpha^{-1} \tag{4.3}$$

on $\varphi_\alpha(F^{-1}(V_i) \cap U_\alpha)$ is a C^∞ function.

The smooth map is continuous in the manifold topology.

A manifold can have more complex structure than a locally Euclidean topology, so we can find manifolds with smooth, complex and even algebraic topology. The range is varied, from a smooth manifold with a metric (what is called **Riemann manifold**), to a complex manifold with a Kähler structure (which is called Kähler manifold).

4.7.2 Tensor products

Vector fields and the derivatives of smooth functions are analytical objects on manifolds. There are more general classes of objects called tensors that can help to define the exterior algebra. The differential forms and the exterior derivative generalize the grad, div and curl of three-dimensional calculus, what makes the basic differential operator on forms.

Let V, W be two finite-dimensional vector spaces over $\mathbf{R}$. The new vector space $V \otimes W$ is the tensor product with the following properties:

- if $v \in V$ and $w \in W$ then there is a product $v \otimes w \in V \otimes W$
- the product is bilinear:

$$(\lambda v_1 + \mu v_2) \otimes w = \lambda v_1 \otimes w + \mu v_2 \otimes w \tag{4.4}$$

$$v \otimes (\lambda w_1 + \mu w_2) = \lambda v \otimes w_1 + \mu v \otimes w_2 \tag{4.5}$$

The tensor product $V \otimes W$ has the universal property that if $B : V \times W \to U$ is a bilinear map to a vector space U then there is a unique linear map

$$\beta : V \otimes W \to U \tag{4.6}$$

such that $B(v, w) = \beta(v \otimes w)$.

4.7.3 The Riemannian metrics

Using Professor Nigel Hitchin definition [MHitchin12], a Riemann metric on a manifold M is a smoothly varying positive definite inner product on the tangent spaces T_x. Quoting Weisstein [Weisstein12], suppose that for every point x in a manifold M, an inner product $< ., . >$ is defined on a tangent space T_xM of M at x. A Riemannian metric on a manifold M is the collection of all those inner products.

An inner product is a bilinear form, so at each point x there is a vector in the tensor product

$$T_x^* \otimes T_x^* \tag{4.7}$$

.

Each coordinate system $x_1, ..., x_n$ defines a basis $dx_1, ..., dx_n$ for each T_x^* in the coordinate neighborhood and the n^2 elements

$$dx_i \otimes dx_j, 1 \leq i, j \leq n \tag{4.8}$$

give a corresponding basis for $T_x^* \otimes T_x^*$

The Jacobian of a change of coordinates defines an invertible linear transformation $J : T_x^* \to T_x^*$ and we have a corresponding invertible linear transformation $J \otimes J : T_x^* \otimes T_x^* \to T_x^* \otimes T_x^*$.

With all the previous definitions we can now say that a Riemannian metric on a manifold M is a section g of $T^* \otimes T^*$ which at each point is symmetric and positive definite.

In a local coordinate system we can write

$$g = \sum_{i,j} g_{ij}(x) dx_i \otimes dx_j \tag{4.9}$$

where $g_{ij}(x) = g_{ji}(x)$ and is a smooth function, with $g_{ij}(x)$positive definite. The tensor product symbol is often omitted so you can simply write:

$$g = \sum_{i,j} g_{ij}(x) dx_i dx_j \tag{4.10}$$

4.8 Metric Estimation Process

Data records are represented as D-dimensional vectors of a feature space F_D. The n-th data record is thus represented by a set of D real numbers $X_n^i; \quad i = 1, 2, ..., D; \quad n = 1, 2, ..., N$; where N is the total number of data records. This representation may come from a direct numeric feature selection method or through a kernelization embedding.

$$x_n \to \phi(x_n) \in F_D \tag{4.11}$$

In this case the data records may have a total or partial symbolic nature and metric properties are established through the kernel

$$K(x_n, x_m) = \langle \phi(x_n) \mid \phi(x_m) \rangle \equiv K_{n,m} \qquad n, m = 1, 2, ..., N \tag{4.12}$$

F_D is then a vector space endowed with a scalar product

$$\left\langle X \mid X' \right\rangle \equiv \sum_{i,j=1}^{D} G_{ij} X^i X'^j \tag{4.13}$$

where G_{ij} is a symmetric $D \times D$ matrix. The origin of coordinates can be chosen at will. If X_0 is the appropriate choice, the correct representative of data record n is $X_n - X_0$ and the scalar product should be written as

$$\begin{gathered} \sum_{i,j=1}^{D} G_{ij}(X^i - X_0^i)(X'^j - X_0^j) = \sum_{i,j=1}^{D} G_{ij} X^i X'^j + \sum_{i=1}^{D} g_i (X^i + X'^i) + b \\ g_i = \sum_{j=1}^{D} G_{ij} X_0^j \quad b \equiv \sum_{i,j=1}^{D} G_{ij} X_0^i X_0^j \end{gathered} \tag{4.14}$$

To fix with this elementary degree of freedom we impose

$$\sum_{n=1}^{N} X_n = 0 \tag{4.15}$$

as a restriction for data records in the training set.

4.8.1 Learning set

We shall now assume that a subset of the whole database has already been labeled. There are thus M data records which have been assigned labels within a label set $\{L\}$ by an external supervisor. We have then

$$X_m^{(\ell)} \quad \ell = 1, 2, ..., L \quad m = 1, 2, ..., M \tag{4.16}$$

as a training set for the best possible metric embedding in R^D.

One may use this supervision to enforce similarity between records belonging to the same label. So two records with the same label should be more similar that each one of them compared with records with different labels. In general, one should select learning triples X_a, X_b, X_c such that one has

$$K(X_a, X_b) > K(X_b, X_c) \tag{4.17}$$

We are using the kernel as the natural measure of similarity of the problem.

4.8.2 Identical embedding

Let us now examine the case where the embedding is the identity

$$X_n \to \phi(X_n) = X_n \tag{4.18}$$

The most direct similarity measure is given by distance defined by

$$\begin{aligned} &K(X_m, X_{m'}) = \|X_m - X_{m'}\|^2 \equiv \langle X_m - X_{m'} \mid X_m - X_{m'} \rangle = \\ &= \sum_{i,j=1}^{D} G_{ij}(X_m^i - X_{m'}^i)(X_m^j - X_{m'}^j) \end{aligned} \tag{4.19}$$

Then if X_a, X_b, X_c is a metrics learning triple, one must have

$$\|X_a - X_b\|^2 < \|X_b - X_c\|^2 \tag{4.20}$$

This can also be written as

$$\sum_{i,j=1}^{D} G_{ij}\left((X_a^i - X_b^i)(X_a^j - X_b^j) - (X_b^i - X_c^i)(X_b^j - X_c^j)\right) < 0$$

$$\begin{aligned} 2\sum_{i=1}^{D-1}\sum_{j=i+1}^{D} G_{ij}\left((X_a^i - X_c^i)\left(\tfrac{X_a^j+X_c^j}{2} - X_b^j\right) + (X_a^j - X_c^j)\left(\tfrac{X_a^i+X_c^i}{2} - X_b^i\right)\right) \\ +2\sum_{i=1}^{D} G_{ii}(X_a^i - X_c^i)\left(\tfrac{X_a^i+X_c^i}{2} - X_b^i\right) \quad < \quad 0 \end{aligned} \tag{4.21}$$

It is then convenient to combine the vectors from the triple as

$$S_{ac} \equiv \frac{X_a + X_c}{2} \quad D_{ac} \equiv X_a - X_c \tag{4.22}$$

Which allow us to write (4.21) as

$$\sum_{i=1}^{D-1}\sum_{j=i+1}^{D} G_{ij}\left(D_{ac}^i(S_{ac}^j - X_b^j) + D_{ac}^j(S_{ac}^i - X_b^i)\right) + \sum_{i=1}^{D} G_{ii}D_{ac}^i(S_{ac}^i - X_b^i) \quad < \quad 0 \tag{4.23}$$

Let us now introduce indexes in a $\frac{D(D+1)}{2}$ dimensional space associated with the $\frac{D(D-1)}{2}$ pairs (ij) of indexes with $i < j$ and the D indexes associated with (ii). Vectors in $\frac{D(D+1)}{2}$ dimensions will be super-scripted with an arrow. For the metric tensor we introduce the vector

$$\vec{W} \equiv G_{(ij)} = (G_{11}, G_{12}, ..., G_{1D}, G_{22}, G_{23}, ..., G_{2D}, ..., G_{D-1D-1}, G_{D-1D}, G_{DD}) \tag{4.24}$$

and for the learning triple X_a, X_b, X_c

$$\begin{aligned} &\vec{T}_{abc} \equiv T_{abc}^{(ij)} \\ &T_{abc}^{(ij)} \equiv D_{ac}^i(S_{ac}^j - X_b^j) + D_{ac}^j(S_{ac}^i - X_b^i) \quad ; \quad i < j \\ &T_{abc}^{(ii)} \equiv D_{ac}^i(S_{ac}^i - X_b^i) \end{aligned} \tag{4.25}$$

Then we can write (4.24) as a scalar product in $R^{\frac{D(D+1)}{2}}$

$$\vec{W} \cdot \vec{T}_{abc} < 0 \tag{4.26}$$

The problem now is to select the most appropriate learning triples and its separation in two classes. According to (4.26) the learned metric $\vec{W}$ could thus be obtained by using a maximum separating hyperplane. This in turn could be treated as a maximal margin Support Vector Machine problem.

4.8.3 Selection of learning triplets

Let us consider the set of vectors $X_m^{(\ell)} \quad m = 1, 2, ..., \#\{\ell\}$ classified as ℓ by the supervisor. We denote by $X_{p(m)}^{(\ell)}$ the vector in $\{\ell\}$ that has the minimal Euclidean distance to $X_m^{(\ell)}$. Let us now consider a third vector in a different class $X_k^{(\ell')}$. Then we propose the learning triplet

$$\left(X_m^{(\ell)}, X_{p(m)}^{(\ell)}, X_k^{(\ell')}\right) \quad if \quad \left\|X_k^{(l')} - X_m^{(\ell)}\right\| > \left\|X_k^{(l')} - X_{p(m)}^{(\ell)}\right\| \tag{4.27}$$

If the condition is not fulfilled, the order of the two first vectors is inverted. Within the class $\{\ell\}$ one can construct the undirected graph T_ℓ which has $1, 2, ..., \#\{l\}$ as the vertex set and the set of minimum distance pairs as its edge set. This graph is necessarily acyclic and is then a non rooted tree over the vectors in $\{\ell\}$. Then (4.27) produces a learning triplet for every edge in T_ℓ combined with all vectors in other classes. The triplet formed by choosing the vector in $\{\ell'\}$ which has the minimal Euclidean distance to the set $\left\{X_m^{(\ell)}, X_{p(m)}^{(\ell)}\right\}$ imposes the most astringent condition. The number of these maximal demanding learning triplets is clearly

$$(\#\{1\}-1)(L-1)+(\#\{2\}-1)(L-1)+...+(\#\{2\}-1)(L-1) = (M-L)(L-1) \tag{4.28}$$

where M is the total number of vectors in the training set.

Notice that T_ℓ can be constructed by selecting an arbitrary vector as root and using a minimal spanning tree conventional algorithm. Each edge on this tree can then be combined with closest vectors in other classes to form the desired triplets. For Euclidean metrics $G_{ij} \equiv \delta_{ij}$ and (4.26) reduces to

$$\vec{W}^{(e)} \cdot \vec{T}_{abc} = \sum_{i=1}^{D} D_{ac}^i \left(S_{ac}^i - X_b^i\right) < 0 \tag{4.29}$$

Therefore it is possible to know in advance the number of correct matches that will occur if the metrics was Euclidean. More importantly one could use (4.29) to estimate an upper bound of the magnitude of separation by evaluating the maximum declassification given in the Euclidean case. It is natural to expect to improve on this with a more general metrics.

Notice that even if the match is perfect in the Euclidean case, using a Support Vector Machine for maximum margin hyper-plane separation one can improve the metrics resolution over the training set.

4.8.4 Margin

A general hyper-plane in D dimensions contain the tips of all vectors X verifying the equation

$$W \cdot X + \theta = 0 = \sum_{i=1}^{D} W_i X_i + \theta \tag{4.30}$$

The distance of a general Z vector to this hyper-plane can be readily found as follows. Since W is the vector perpendicular to the hyperplane, the straight line going through Z and perpendicular to the plane is

$$\frac{X_1 - Z_1}{W_1} = \frac{X_2 - Z_2}{W_2} = ... = \frac{X_D - Z_D}{W_D} = C \tag{4.31}$$

where C is some constant. The intersection point X will verify both this equation and the hyper-plane equation. Therefore

$$X_i = Z_i + CW_i \quad W \cdot X + \theta = W \cdot Z + \theta + CW^2 = 0 \qquad C = -\frac{W \cdot Z + \theta}{W^2} \tag{4.32}$$

At last the distance $d(Z)$ from Z to the hyper-plane can be explicitly computed as

$$d(Z) = \frac{|W \cdot Z + \theta|}{\sqrt{W^2}} \tag{4.33}$$

We may introduce

$$y(Z) \equiv \begin{cases} +1: & W \cdot Z + \theta > 0 \\ -1: & W \cdot Z + \theta < 0 \end{cases} \tag{4.34}$$

and write

$$d(Z) = \frac{y(Z)\,(W \cdot Z + \theta)}{\sqrt{W^2}} = y(Z)\left(\frac{W}{\sqrt{W^2}} \cdot Z + b\right) \qquad b \equiv \frac{\theta}{\sqrt{W^2}} \tag{4.35}$$

Suppose now that we are given a set $\{Z_m\}_{m=1}^{M}$ of training vectors. Define the distance of the hyper-plane to the set as the minimal margin of its points to the hyperplane. Then we propose the problem of finding the hyper-plane with the maximum distance to the set.

Therefore if one wants a maximum margin one could propose the following problem

$$\begin{array}{ll} \min \quad -\gamma \qquad s.a. & \\ y_m(W \cdot X_m + b) \geq \gamma \qquad & W^2 = 1 \end{array} \tag{4.36}$$

So we want the maximum possible value of the margin over the training set. For item m the margin is given by $y_m(W \cdot X_m + b)$ provided the second constraint holds. Then we impose that γ is the lesser margin over the training set in the first constraint and then maximize this quantity. This leads us to the Lagrangian problem

$$\begin{array}{ll} min \quad L(W,b) \equiv -\gamma - \sum\limits_{m=1}^{M} \alpha_m \left(y_m(W \cdot X_m + b) - \gamma\right) + \lambda(W^2 - 1) & \\ \alpha_m \geq 0 \qquad m = 1, 2, ..., M & \end{array} \tag{4.37}$$

Extremal conditions are

$$\begin{array}{l} \frac{\partial L}{\partial W_i} = -\sum\limits_{m=1}^{M} \alpha_m y_m X_m^i + 2\lambda W_i = 0 \quad i = 1, 2, ..., D \\ \frac{\partial L}{\partial b} = -\sum\limits_{m=1}^{M} \alpha_m y_m = 0 \\ \frac{\partial L}{\partial \gamma} = -1 + \sum\limits_{m=1}^{M} \alpha_m = 0 \end{array} \tag{4.38}$$

It is very convenient to introduce a double notation for alphas; $\bar{\alpha}_m \equiv y_m \alpha_m$ so α_m are semi definite positive but $\bar{\alpha}_m$ have the sign according to the class of X_m. The obtained constraints are thus

$$\sum_{m=1}^{M} \alpha_m = 1 \qquad \sum_{m=0}^{M} \bar{\alpha}_m = 0 \tag{4.39}$$

But they are not all. It remains to minimize on λ and to take account of the Kahrun - Tucker conditions. We may substitute the obtained value $W^i = \frac{1}{2\lambda} \sum\limits_{m=1}^{M} \bar{\alpha}_m X_m^i$. Then using the alpha - constraints it is easy to obtain

$$L = -\lambda - \frac{1}{4\lambda} \sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'} \tag{4.40}$$

This is a dual Lagrangian over the multipliers. Therefore one must maximize over it. Notice that $L = -\lambda - \frac{K}{4\lambda} \quad L' = -1 + \frac{K}{4\lambda^2} \quad L'' = \frac{K}{2\lambda^3}$. Therefore the value that

maximizes is $\lambda^* = \frac{1}{2}\sqrt{\sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'}}$. The Lagrangian is now

$$\begin{aligned} & L = -\sqrt{\sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'}} \\ & \sum_{m=1}^{M} \alpha_m = 1 \quad \sum_{m=1}^{M} \bar{\alpha}_m = 0 \qquad \alpha_m \geq 0 \\ & \bar{\alpha}_m \left(\tfrac{1}{2\lambda^*} \sum_{m'=1}^{M} \bar{\alpha}_{m'} X_{m'} \cdot X_m + b \right) - \alpha_m \gamma^* = 0 \end{aligned} \tag{4.41}$$

The value of the Lagrangian at the minimum is

$$L^* = -\sqrt{\sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'}} = -\sqrt{2\lambda^* \gamma^*} = -\sqrt{-L^* \gamma^*} \tag{4.42}$$

Therefore $L^* = -\gamma^*$ as expected. Also if $\bar{W} \equiv \sum_{m=1}^{M} \bar{\alpha}_m X_m$ it follows that $W^* = \frac{\bar{W}}{\sqrt{\bar{W}^2}} = \frac{\bar{W}}{\gamma^*}$ thus confirming the normalization of W^*. Now the value of b is obtained as follows. Let m be an index on the support $\bar{\alpha}_m \neq 0$. Then the margin for X_m must be optimum, i.e.

$$y_m \left(W^* \cdot X_m + b \right) = \gamma^* \quad b = y_m \gamma^* - W^* \cdot X_m = y_m \gamma^* - \frac{1}{\gamma^*} \sum_{m'=1}^{M} \bar{\alpha}_{m'} X_m \cdot X_{m'} \tag{4.43}$$

Sometimes one takes $\bar{W}$ as the weight vector. This is equivalent to multiply W^* by γ^* and the same must be done for b. The dual problem could thus be formulated as follows

$$\begin{aligned} \max \quad & -\sum_{m,m'=1}^{N} \bar{\alpha}_m \bar{\alpha}_{m'} (X_m \cdot X_{m'}) \qquad s.a. \quad \alpha_m \geq 0 \quad m = 1, 2, ..., M \\ & \sum_{m=1}^{M} \alpha_m = 1 \qquad\qquad \sum_{m=1}^{M} \bar{\alpha}_m = 0 \end{aligned} \tag{4.44}$$

After optimization the separating hyper-plane is $W \cdot x + b = 0$ where

$$W = \sum_{m=1}^{M} \bar{\alpha}_m^* X_m \qquad b = y_m \left(\gamma^{*2} - \sum_{m'=1}^{M} \bar{\alpha}_{m'}^* X_m \cdot X_{m'} \right) \qquad \gamma^* = \sqrt{W^2} \tag{4.45}$$

4.8.5 Soft margin

In presence of noise one must allow for declassification. The appropriate formulation in this case is

$$\begin{array}{ll} \min & -\gamma + C \sum_{m=1}^{M} \xi_m \qquad s.a. \\ & y_m(W \cdot X_m + b) \geq \gamma - \xi_m \quad \xi_m \geq 0 \qquad W^2 = 1 \end{array} \tag{4.46}$$

For compliant members one has $\xi_m = 0$ and thus γ is the minimum margin for this subset. There are however non compliant members with $\xi_m > 0$ that do not respect the limit. However this is penalized by the term in C. Obviously for $C \to \infty$ this problem should reduce to the hard margin problem and it is therefore its generalization.

Let us then write the Lagrangian

$$\begin{array}{l} min \quad L(W, b, \gamma, \xi) \equiv -\gamma + C \sum_{m=1}^{M} \xi_m - \sum_{m=1}^{M} \alpha_m \left(y_m(W \cdot X_m + b) - \gamma + \xi_m \right) \\ - \sum_{m=1}^{M} \beta_m \xi_m + \lambda(W^2 - 1) \\ \qquad \alpha_m \geq 0 \quad \beta_m \geq 0 \quad m = 1, 2, ..., M \end{array} \tag{4.47}$$

Conditions for minimum are

$$\begin{array}{l} \frac{\partial L}{\partial W_i} = -\sum_{m=1}^{M} \alpha_m y_m X_m^i + 2\lambda W_i = 0 \quad i = 1, 2, ..., D \\ \frac{\partial L}{\partial b} = -\sum_{m=1}^{M} \alpha_m y_m = 0 \\ \frac{\partial L}{\partial \gamma} = -1 + \sum_{m=1}^{M} \alpha_m = 0 \\ \frac{\partial L}{\partial \xi_m} = C - \alpha_m - \beta_m = 0 \end{array} \tag{4.48}$$

Therefore $W^i \equiv \frac{1}{2\lambda} \sum_{m=0}^{M} \bar{\alpha}_m X_m^i$ and by substitution in the Lagrangian one has

$$L = -\lambda - \frac{1}{4\lambda} \sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} \, X_m \cdot X_{m'} \tag{4.49}$$

The problem is now identical to the hard margin case but the restriction $\alpha_m + \beta_m = C$ imposes the additional constraint $\alpha_m \leq C$.

As before one obtains $\lambda^* = \frac{1}{2}\sqrt{\sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'}}$

Therefore the dual formulation is now

$$\begin{array}{lll} \max & -\sum_{m,m'=1}^{N} \bar{\alpha}_m \bar{\alpha}_{m'} (X_m \cdot X_{m'}) & s.a. \quad 0 \leq \alpha_m \leq C \quad m = 1, 2, ..., M \\ & \sum_{m=1}^{M} \alpha_m = 1 & \sum_{m=1}^{M} \bar{\alpha}_m = 0 \end{array} \tag{4.50}$$

To compute b notice that if ℓ is an index for an item with $0 < y_\ell \alpha_\ell < C$. This means that $y_\ell = +1$ and $\alpha_\ell > 0 \quad \beta_\ell > 0$ and then one has the two constraints $(W \cdot X_\ell + b) = \gamma - \xi_\ell \quad ; \quad \xi_\ell = 0$.

Now if we have a second item ℓ' with $-C < y_\ell \alpha_\ell < 0$ this means $y_\ell = -1$ and again we have the two constraints -$(-W \cdot X_{\ell'} + b) = \gamma - \xi_{\ell'} \quad ; \quad \xi_{\ell'} = 0$

Therefore $b^* = -\frac{1}{2}\{W^* \cdot (X_\ell + X_{\ell'})\}$.

Remember however that normally one wants to leave the normalization of W by taking by defining the separating hyper-plane as $W^* \cdot x + b^* = 0$ with

$$W^* = \sum_{m=1}^{M} \bar{\alpha}_m^* X_m \tag{4.51}$$

This implies a change of scale by a factor of $\frac{1}{2\lambda^*}$. Then b has to take the same scale so the expression $b^* = -\frac{1}{2}\{W^* \cdot (X_\ell + X_{\ell'})\}$ still holds. Also for item ℓ one has in the new scale $\frac{1}{2\lambda^*}(W^* \cdot X_\ell + b^*) = \gamma^*$ which is a valid way to compute the optimal achieved margin.

The dual problem implies solving the optimization problem

$$\begin{array}{ll} \max & \omega \left(\sum_{m=1}^{M} \alpha_m - 1 \right) - \sum_{m,m'=1}^{N} \bar{\alpha}_m \bar{\alpha}_{m'} (X_m \cdot X_{m'}) + \varpi \sum_{m=1}^{M} \bar{\alpha}_m \\ & s.a. \quad 0 \leq \alpha_m \leq C \quad m = 1, 2, ..., M \end{array} \tag{4.52}$$

Where ω, ϖ are the Lagrange multipliers to enforce the constraints. There is a global factor invariance in this Lagrangian. In fact, multiplying by a positive factor $\frac{\varepsilon^2}{2}$ which do not change the type of optimization (minimum or maximum), one has

$$\begin{array}{l} \frac{\varepsilon^2}{2}L = \frac{\omega\varepsilon}{2}\sum\limits_{m=1}^{M}\varepsilon\alpha_m - \frac{1}{2}\sum\limits_{m,m'=1}^{N}\varepsilon\bar{\alpha}_m\varepsilon\bar{\alpha}_{m'}(X_m\cdot X_{m'}) + \frac{\varpi\varepsilon}{2}\sum\limits_{m=1}^{M}\varepsilon\bar{\alpha}_m \\ \quad s.a. \quad 0 \le \alpha_m \le C \quad m = 1,2,...,M \end{array} \tag{4.53}$$

Now if we choose $\varepsilon = \frac{2}{\omega}$ and rename $\varepsilon\alpha_m \to \alpha_m$ we obtain an equivalent problem of the form

$$\begin{array}{l} \max \quad \sum\limits_{m=1}^{M}\alpha_m - \frac{1}{2}\sum\limits_{m,m'=1}^{N}\bar{\alpha}_m\bar{\alpha}_{m'}(X_m\cdot X_{m'}) + \mu\sum\limits_{m=1}^{M}\bar{\alpha}_m \\ \quad s.a. \quad 0 \le \alpha_m \le C^{(new)} = \varepsilon C^{(old)} \quad m = 1,2,...,M \end{array} \tag{4.54}$$

We have lost a multiplier and thus it is impossible to enforce $\sum\limits_{m=1}^{M}\alpha_m = 1$. However, after optimization we can compute $\sum\limits_{m=1}^{M}\alpha_m^* = \varepsilon\sum\limits_{m=1}^{M}\alpha_m^{*(old)} = \varepsilon$ which uses the Lagrange multiplier ε in a deferred way. More simply by computing

$$\alpha_m^{*(old)} = \frac{\alpha_m^*}{\sum\limits_{m'=1}^{M}\alpha_{m'}^*}$$

we have all links to the old problem and its solution. Remember however that now $C = C^{(old)}\sum\limits_{m'=1}^{M}\alpha_{m'}^*$

Now in the primal formulation we notice that if the minimal margin for non slack training set members is fixed to γ^* , one could write

$$\begin{array}{ll} \min \quad -\frac{\gamma^*}{\sqrt{W^2}} + C\sum\limits_{m=1}^{M}\xi_m & s.a. \\ y_m(W\cdot X_m + b) \ge \gamma^* - \xi_m & \xi_m \ge 0 \end{array} \tag{4.55}$$

Notice that now the level for W^2 is fixed by γ^* and the condition $W^2 = 1$ must be abandoned.

Following this approach one could start now from the Lagrangian

$$\begin{array}{l} min \quad L(W,b,\xi) \equiv -\frac{\gamma^*}{\sqrt{W^2}} \\ +C\sum\limits_{m=1}^{M}\xi_m - \sum\limits_{m=1}^{M}\alpha_m\left(y_m(W\cdot X_m + b) - \gamma^* + \xi_m\right) - \sum\limits_{m=1}^{M}\beta_m\xi_m \\ \quad \alpha_m \ge 0 \quad \beta_m \ge 0 \quad m = 1,2,...,M \end{array} \tag{4.56}$$

and the extremal conditions are

$$\begin{array}{l} \frac{\partial L}{\partial W_i} = \frac{\gamma^*}{\left(W^2\right)^{\frac{3}{2}}} W_i - \sum\limits_{m=1}^{M} \bar{\alpha}_m X_m^i = 0 \\ \frac{\partial L}{\partial b} = - \sum\limits_{m=1}^{M} \bar{\alpha}_m = 0 \\ \frac{\partial L}{\partial \xi_m} = C - \alpha_m - \beta_m = 0 \end{array} \tag{4.57}$$

Notice that

$$\begin{array}{l} W_i = \frac{\left(W^2\right)^{\frac{3}{2}}}{\gamma^*} \sum\limits_{m=1}^{M} \bar{\alpha}_m X_m^i \\ W^2 = \frac{\left(W^2\right)^3}{\gamma^{*2}} \sum\limits_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'} \qquad \frac{1}{\left(W^2\right)} = \frac{1}{\gamma^*} \sqrt{\sum\limits_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'}} \end{array}$$

$$W_i = \sqrt{\gamma^*} \left(\sum_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} X_m \cdot X_{m'} \right)^{-\frac{3}{4}} \sum_{m=1}^{M} \bar{\alpha}_m X_m^i \tag{4.58}$$

Substitution in the Lagrangian leads now to

$$\begin{array}{ll} \max & L(\alpha) \equiv \gamma^* \sum\limits_{m=1}^{M} \alpha_m - 2\sqrt{\gamma^*} \left(\sum\limits_{m,m'=1}^{M} \bar{\alpha}_m \bar{\alpha}_{m'} (X_{m'} \cdot X_m) \right)^{\frac{1}{4}} \\ & 0 \leq \alpha_m \leq C \end{array} \tag{4.59}$$

4.8.6 Unbiased soft separation problem

Suppose that one has a training set of vectors $X_m \quad m = 1, 2, ..., M$ which should verify the separation property

$$\sum_{i=1}^{D} W_i X_m^i \geq S > 0 \qquad m = 1, 2, ..., M \tag{4.60}$$

for some vector $\vec{W} \in R^D$. Since one is looking for a hyperplane, the norm of $\vec{W}$ is undetermined. Equivalently by changing the norm of $\vec{W}$ the value of S may be chosen at will. Let us then fix the scale of the problem by imposing

$$S = 1 \tag{4.61}$$

The distance of $\vec{X}_m$ to the hyper-plane is called its margin and it is readily computed to give

$$\gamma_m \equiv \frac{\vec{W} \cdot \vec{X}_m}{\sqrt{\vec{W}^2}} \quad m = 1, 2, ..., M \tag{4.62}$$

Therefore condition (4.60) together with (4.62) is equivalent to say that the margin over the training set verifies

$$\gamma_m \geq \frac{1}{\sqrt{\vec{W}^2}} \quad m = 1, 2, ..., M$$

and therefore

$$\gamma \equiv \min \{\gamma_m\}_{m=1}^{M} \geq \frac{1}{\sqrt{\vec{W}^2}} \tag{4.63}$$

and therefore the *most separating hyperplane* could be obtained by minimizing $\vec{W}^2$ subject to (4.60). This leads us to the minimization problem

$$\min \quad L(W) \equiv \frac{1}{2}\vec{W}^2 \quad s.a. \quad \vec{W} \cdot \vec{X}_m \geq 1 \quad m = 1, 2, ..., M \tag{4.64}$$

This problem may be unfeasible. In general a hyper-plane going through the origin and verifying (4.60) does not exist. There are however some relaxations that transform the problem into a feasible one. The simplest one is to relax the requirement that the hyper-plane goes through the origin.

4.8.7 Hyperplane with bias

As a first relaxation of the original problem, let us propose the Lagrangian

$$\min \quad L(W, b) \equiv \frac{1}{2}\vec{W}^2 + Cb \quad s.a. \quad \vec{W} \cdot \vec{X}_m \geq 1 - b \quad m = 1, 2, ..., M \quad ; \quad b \geq 0 \tag{4.65}$$

where the bias b is a non negative variable that represents a parallel shift of the hyper-plane to achieve feasibility. Obviously one must chose the minimal possible value for b to relax the original problem as slightly as possible. This is achieved by introducing a cost term in the Lagrangian to encourage small values for b. Let us also introduce multipliers to transport the constraints into the Lagrangian. This leads us to

$$\begin{aligned} &L(W,b) \equiv \tfrac{1}{2}\vec{W}^2 + Cb - \sum_{m=1}^{M} \alpha_m \left(\vec{W}\cdot\vec{X}_m - 1 + b\right) - \beta b \\ &\alpha_m \geq 0 \quad m = 1,2,...,M \quad ; \quad \beta \geq 0 \end{aligned} \tag{4.66}$$

The minimization conditions are

$$\begin{aligned} &\tfrac{\partial L}{\partial \vec{W}} = \vec{W} - \sum_{m=1}^{M} \alpha_m \vec{X}_m = 0 \\ &\tfrac{\partial L}{\partial b} = C - \beta - \sum_{m=1}^{M} \alpha_m = 0 \\ &\beta b = 0 \qquad \alpha_m \left(\vec{W}\cdot\vec{X}_m - 1 + b\right) = 0 \qquad m = 1,2,...,M \end{aligned} \tag{4.67}$$

One may substitute these conditions into the Lagrangian to obtain

$$L = -\frac{1}{2}\sum_{m,m'=1}^{M} \alpha_m \alpha_{m'} \vec{X}_m \cdot \vec{X}_{m'} + (C-\beta)b - (b-1)\sum_{m=1}^{M} \alpha_m \tag{4.68}$$

which can also be worked out using (4.67) to obtain

$$L = \sum_{m=1}^{M} \alpha_m - \frac{1}{2}\sum_{m,m'=1}^{M} \mathrm{K}_{mm'}\alpha_m\alpha_{m'} \qquad \mathrm{K}_{mm'} \equiv \vec{X}_m \cdot \vec{X}_{m'} \tag{4.69}$$

An alternative form is

$$\begin{aligned} &L = \tfrac{1}{2}\left(C(1+b) - \beta\right) \\ &\alpha_m \left(K_{mm'}\alpha_{m'} - 1 + b\right) = 0 \quad m = 1,2,...,M \\ &\beta b = 0 \\ &\alpha_m \geq 0 \qquad \beta \geq 0 \end{aligned} \tag{4.70}$$

The problem could be solved using the following algorithm:

- Chose a set $\{S\}$ as the support for the $\alpha's$, i.e. $\alpha_m = 0 \quad \alpha_m \notin \{S\}$
- Find non null $\alpha's$ by solving $\sum_{m'\in\{S\}} \mathrm{K}_{mm'}\alpha_{m'} = 1 - b \qquad m \in \{S\}$
- On this system $b = 0$ if the resulting $\alpha's$ are non negative. In such a case $\beta = C - \sum_{m=1}^{M} \alpha_m$ and this results into a feasible point if $\beta \geq 0$ and the Lagrangian takes the value $L = \frac{1}{2}\left(C - \beta\right)$. If not, it will be necessary to chose b as the minimum positive value that makes the $\alpha's$ in the support non negative. Then $\beta = 0$ and the Lagrangian takes the value $\frac{1}{2}C(1+b)$

- Choose the support set that *maximizes* L
- Notice that re-normalizing the Lagrangian by dividing by $C/2$ and the Lagrangian values would be $1 - \frac{\beta}{C} = \frac{1}{C}\sum_{m=1}^{M}\alpha_m \quad if \quad b = 0$ and $1 + b \quad \beta = 0$ and then the solution will always minimize the value for b.
- When the system $\sum_{m' \in \{S\}} \mathrm{K}_{mm'}\alpha_{m'} = 1 \qquad m \in \{S\}$ does not have a solution with non negative $\alpha's$ one has to use a positive b to obtain

$$\begin{aligned} &\mathrm{K}(\alpha^{new}) = 1 \\ &\alpha_m^{new} = \alpha_m^{old} - b \sum_{m' \in \{S\}} \mathrm{K}^{-1}_{mm'} \end{aligned} \tag{4.71}$$

4.8.8 Example

Let's define this example:

$$\begin{aligned} &X_1 = (-1,-1), X_2 = (1,2), X_3 = (-2,-1) \\ &\mathrm{K} = \begin{pmatrix} 2 & -3 & 3 \\ -3 & 5 & -4 \\ 3 & -4 & 5 \end{pmatrix} \end{aligned} \tag{4.72}$$

In the first iteration $S = \{1\} \quad 2\alpha_1 = 1 - b$ we obtain a solution: $\alpha_1 = \frac{1}{2} \quad b = 0 \quad \beta = C - \frac{1}{2} \quad L = 1 - \frac{C-1/2}{C} = \frac{1}{2C} \leq 1$

In the second iteration $S = \{2\} \quad 5\alpha_2 = 1 - b$, the solution is: $\alpha_2 = \frac{1}{5} \quad b = 0 \quad \beta = C - \frac{1}{5} \quad L = 1 - \frac{C-1/5}{C} = \frac{1}{5C} \leq 1$

In the third iteration $S = \{3\} \quad 5\alpha_3 = 1 - b$, with solution: $\alpha_3 = \frac{1}{5} \quad b = 0 \quad \beta = C - \frac{1}{5} \quad L = 1 - \frac{C-1/5}{C} = \frac{1}{5C} \leq 1$

Fourth case:

$$\begin{aligned} &S = \{1,2\} \quad \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \frac{1}{C}\begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix}\begin{pmatrix} 1-\xi_1 \\ 1-\xi_2 \end{pmatrix} \\ &\xi_1 = \xi_2 = 0 \quad \alpha_1 = \frac{8}{C} \quad \alpha_2 = \frac{5}{C} \quad \alpha_3 = 0 \\ &\xi_3 = 0 \\ &\beta_3 = \frac{1}{3} \quad \beta_2 = \frac{1}{3} - \frac{5}{C} \quad \beta_1 = \frac{1}{3} - \frac{8}{C} \qquad L = \frac{13}{C} \end{aligned} \tag{4.73}$$

Fifth case:

$$S = \{1,3\} \quad \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 1-b \\ 1-b \end{pmatrix} \qquad \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 3(1-b) \\ -(1-b) \end{pmatrix} \tag{4.74}$$

Sixth case:

$$S = \{2,3\} \quad \begin{pmatrix} 5 & -4 \\ -4 & 5 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 1-b \\ 1-b \end{pmatrix} \qquad \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 1-b \\ 1-b \end{pmatrix} \tag{4.75}$$

$$b = 0 \quad C - 2 = \beta \quad L = \frac{2}{C} \leq 1 \tag{4.76}$$

Seventh case:

$$S = \{1,2,3\} \quad \begin{pmatrix} 2 & -3 & 3 \\ -3 & 5 & -4 \\ 3 & -4 & 5 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = (1-b) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{4.77}$$

$$b = 1 \quad \alpha_1 = -3\alpha_3 \qquad \alpha_2 = -\alpha_3 \quad \textit{impossible} \tag{4.78}$$

The maximum solution is 4, independently of C.

$$\vec{W} = \left(8 \begin{pmatrix} -1 \\ -1 \end{pmatrix} + 5 \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) = \begin{pmatrix} -3 \\ 2 \end{pmatrix} \qquad = norm = \begin{pmatrix} -\frac{3}{\sqrt{13}} \\ \frac{2}{\sqrt{13}} \end{pmatrix} \tag{4.79}$$

The problem is feasible as $b = 0$.

4.9 Building Riemann manifolds with Non Hyperbolic Adjacent Region

Sammon's method for inter-dimensional preservation of distances proposes to minimize a functional such as

$$F\left\{\vec{x}_c\right\} \equiv \frac{1}{K} \sum_{\substack{c',c \in \{C\} \\ c'<c}} \frac{\left(D_{cc'} - d(|\vec{x}_c - \vec{x}_{cW}|)\right)^2}{D_{cc'}} \tag{4.80}$$

In previous expression we have a data set $\{C\}$ represented by d-dimensional vectors

$$\vec{x}_c : c \in \{C\} \qquad \vec{x}_c \in R^d \qquad \vec{x}_c \equiv (x_1^{(c)}, x_2^{(c)}, ..., x_d^{(c)}) \tag{4.81}$$

In dimensional reduction context d is the dimension of the reduced space. A data immersion into an original D-dimension space is also supposed to exist, the only known data on that space are the distances among members of the set $\{C\}$ which are expressed in a table

$$D_{cc'} = \quad distance \quad c \leftrightarrow c' \quad in \quad R^D \tag{4.82}$$

The distance in the final space has a supposed exclusive function of difference mod of pair of vectors in the set $\{C\}$:

$$d_{cc'} \equiv d(|\vec{x}_c - \vec{x}_{c'}|) \quad ; \quad |\vec{x}_c - \vec{x}_{c'}| \equiv \sqrt{(\vec{x}_c - \vec{x}_{c'}) \cdot (\vec{x}_c - \vec{x}_{c'})} = \sqrt{\sum_{i=1}^{d} \left(x_i^{(c)} - x_i^{(c')}\right)^2} \tag{4.83}$$

In the Sammon's method it is supposed $d(x) = x$ so that the distance in R^d is the Euclidean distance. The function $d(|\vec{x}|)$ used in (4.80) allows a light convenient generalization for our purposes.

Usually, the original space dimension is lower than the final one, $D > d$, although this is not a mandatory requirement in what follows.

The K constant is a standardization that does not affect the properties of the set that minimizes the functional, but if it is selected such as

$$K \equiv \sum_{\substack{c',c \in \{C\} \\ c'<c}} D_{cc'} \tag{4.84}$$

leads to a good leading option for the functional optimal with the focus numeric values

$$F\{C_{perf}\} = 0 \qquad F\{C_{conc}\} = 1 \tag{4.85}$$

Where $\{C_{perf}\}$ is a set of perfect matching between R^D and R^D distances, and $\{C_{conc}\}$ belongs to a trivialized execution in R^d where all dots are gathered in a unique one.

Whith that normalization, it would be possible to use

$$B(\{C\}) \equiv 1 - F(\{C\}) \tag{4.86}$$

as a goodness of fit of the achieved minimum.

It is interesting to carry out the square in (4.83) for getting

$$F\{\vec{x}_c\} = 1 - \frac{2}{K} \sum_{\substack{c,c' \in \{C\} \\ c<c'}} d_{cc'} + \frac{1}{K} \sum_{\substack{c,c' \in \{C\} \\ c<c'}} \frac{d_{cc'}^2}{D_{cc'}} \tag{4.87}$$

In the Euclidean distance case $d_{cc'} = d(|\vec{x}_c - \vec{x}_{c'}|) = |\vec{x}_c - \vec{x}_{c'}|$, the previous expression belongs to

$$\frac{1}{2} F\{\vec{x}_c\} = \frac{1}{2} + \frac{1}{K} \left[- \sum_{\substack{c,c' \in \{C\} \\ c<c'}} |\vec{x}_c - \vec{x}_{c'}| + \frac{1}{2} \sum_{\substack{c,c' \in \{C\} \\ c<c'}} \frac{|\vec{x}_c - \vec{x}_{c'}|^2}{D_{cc'}} \right] \tag{4.88}$$

For the purpose of optimization the standardization and additive global constants are irrelevant. The square bracket contribution has two components with clear physic interpretation. The first one corresponds to the addition of the all potential pairs

$$V_e \equiv -|\vec{x} - \vec{x}'| \tag{4.89}$$

what can be understood as a non Coulombian repulsive potential between the two dots. The second one corresponds to the attractive interaction of a spring with constant

$$k_{cc'} \equiv \frac{1}{D_{cc'}} \tag{4.90}$$

The functional gradient highlights the forces over the dots of $\{C\}$ understood as physic particles.

$$-\frac{K}{2} \frac{\partial F}{\partial \vec{x}_c} = - \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \frac{\vec{x}_{c'} - \vec{x}_c}{|\vec{x}_c - \vec{x}_{c'}|} + \sum_{\substack{c' \in \{C\} \\ c' \neq c}} k_{cc'} (\vec{x}_{c'} - \vec{x}_c) \tag{4.91}$$

There is a contribution related with the repulsive charge with value +1 and an attractive power associated with a spring with stiffness inversely proportional to distance in R^D. Note that the repulsive power is distance independent. Therefore, this is all about an interaction where the global scale is not modified.

In that context, the Sammon's functional minimization belongs to a physical state where the dots are under collective repulsion for avoiding their gathering, which brings the maximum functional value. This repulsion is managed by the springs, which tend to avoid the space between particles in reciprocal proportion to original distances. Short distances correspond to very rigid springs. The balance is granted by optimal adjustment between original and final distances.

The proportion between the docking constants and the repulsive interaction +1 and the elastic interaction $1/D_{cc'}$ is lead by the form of the Sammon's original functional. Variations of this functional will bring dockings with different reasons between both types of interaction. It seems to be interesting to take the q-Sammon interaction into account with a force between particles given by

$$-\frac{\partial F}{\partial \vec{x}_c} \sim -q^2 \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \frac{\vec{x}_{c'} - \vec{x}_c}{|\vec{x}_c - \vec{x}_{c'}|} + \sum_{\substack{c' \in \{C\} \\ c' \neq c}} k_{cc'}(\vec{x}_{c'} - \vec{x}_c) \tag{4.92}$$

which corresponds to the allocation of an electric power q to each of the particles, and which modifies the original proportion of the Sammon's functional. Actually, it is very easy to check that (4.91) corresponds to a functional as follows:

$$F_q\left\{\vec{x}_c\right\} \equiv \frac{1}{q^4 K} \sum_{\substack{c', c \in \{C\} \\ c' < c}} \frac{\left(q^2 D_{cc'} - d(|\vec{x}_c - \vec{x}_{c'}|)\right)^2}{D_{cc'}} \tag{4.93}$$

which still has the focal values (4.85), although now the perfect fitting corresponds to

$$d_{cc'} = q^2 D_{cc'} \tag{4.94}$$

For $q > 1$ the distances in R^d are bigger than the ones in R^D and vice versa when $q < 1$. In both cases, the distance relation is kept. Therefore, q-Sammon is outstanding when the original space scale is unknown or arbitrary and only the relations are significant. In that case, the constant q manages the level of problem global repulsion, which can force a bigger collective distance in the case of $q > 1$. That control over final scale can be desirable in several applications of the method.

4.9.1 Gradient and Hessian

The optimization of (4.92) requires a gradient calculation of the functional, or from physical point of view, of the forces on each particle. It is quite easy to see that

$$
\begin{aligned}
&-\tfrac{q^4K}{2}\tfrac{\partial F_q}{\partial \vec{x}_c} = \\
&= q^2 \sum_{\substack{c'\in\{C\}\\ c'\neq c}} d'(|\vec{x}_c-\vec{x}_{c'}|)\tfrac{\vec{x}_c-\vec{x}_{c'}}{|\vec{x}_c-\vec{x}_{c'}|} - \sum_{\substack{c'\in\{C\}\\ c'\neq c}} d'(|\vec{x}_c-\vec{x}_{c'}|)\tfrac{d(|\vec{x}_c-\vec{x}_{c'}|)}{|\vec{x}_c-\vec{x}_{c'}|}\tfrac{\vec{x}_c-\vec{x}_{c'}}{D_{cc'}}\ \ l
\end{aligned}
\tag{4.95}
$$

In the Euclidean case $d'=1 \quad d(x)=x$ we get back

$$
-\frac{q^4K}{2}\frac{\partial F_q}{\partial \vec{x}_c}_{euclides} = q^2 \sum_{\substack{c'\in\{C\}\\ c'\neq c}} \frac{\vec{x}_c-\vec{x}_{c'}}{|\vec{x}_c-\vec{x}_{c'}|} - \sum_{\substack{c'\in\{C\}\\ c'\neq c}} \frac{1}{D_{cc'}}(\vec{x}_c-\vec{x}_{c'}) \tag{4.96}
$$

It is possible to write the gradient expression in a more compact way as shown below:

$$
-\frac{q^4K}{2}\frac{\partial F_q}{\partial \vec{x}_c} = \sum_{\substack{c'\in\{C\}\\ c'\neq c}} d'(|\vec{x}_c-\vec{x}_{c'}|)\left(q^2-\frac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}}\right)\frac{\vec{x}_c-\vec{x}_{c'}}{|\vec{x}_c-\vec{x}_{c'}|} \tag{4.97}
$$

Note that if we have $d(|\vec{x}_c-\vec{x}_{c'}|)=q^2D_{cc'}$, for all pairs, the force is canceled out given that the functional absolute minimum has been achieved.

The Hessian can be obtained from (4.94) such as following:

$$
\begin{aligned}
&-\tfrac{q^4K}{2}\tfrac{\partial^2 F_q}{\partial x_i^{(c)}\partial x_j^{(c')}} = \tfrac{(x_i^{(c)}-x_i^{(c'')})(x_j^{(c)}-x_j^{(c'')})}{|\vec{x}_c-\vec{x}_{c''}|^2}\\
&\delta_{cc'}\sum_{\substack{c''\in\{C\}\\ c''\neq c,c'}}\left\{\left(d''(|\vec{x}_c-\vec{x}_{c''}|)-\tfrac{d'(|\vec{x}_c-\vec{x}_{c''}|)}{|\vec{x}_c-\vec{x}_{c''}|}\right)\left(q^2-\tfrac{d(|\vec{x}_c-\vec{x}_{c''}|)}{D_{cc''}}\right)-\tfrac{d'^2(|\vec{x}_c-\vec{x}_{c''}|)}{D_{cc''}}\right\}\\
&+\delta_{cc'}\delta_{ij}\sum_{\substack{c''\in\{C\}\\ c''\neq c,c'}}\left(q^2-\tfrac{d(|\vec{x}_c-\vec{x}_{c''}|)}{D_{cc''}}\right)\tfrac{d'(|\vec{x}_c-\vec{x}_{c''}|)}{|\vec{x}_c-\vec{x}_{c''}|}\\
&-\bar{\delta}_{cc'}\tfrac{(x_i^{(c)}-x_i^{(c')})(x_j^{(c)}-x_j^{(c')})}{|\vec{x}_c-\vec{x}_{c'}|^2}\\
&\left\{\left(d''(|\vec{x}_c-\vec{x}_{c'}|)-\tfrac{d'(|\vec{x}_c-\vec{x}_{c'}|)}{|\vec{x}_c-\vec{x}_{c'}|}\right)\left(q^2-\tfrac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}}\right)-\tfrac{d'^2(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}}\right\}\\
&-\bar{\delta}_{cc'}\delta_{ij}\left(q^2-\tfrac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}}\right)\tfrac{d'(|\vec{x}_c-\vec{x}_{c'}|)}{|\vec{x}_c-\vec{x}_{c'}|}
\end{aligned}
\tag{4.98}
$$

The corresponding expressions in the Euclidean case are given by replacing $d(|x|)\leftrightarrow |x| \quad ; \quad d'(|x|)\leftrightarrow 1 \quad ; \quad d''(|x|)\leftrightarrow 0$.

For a more compact notation we can translate

$$h_{ij}^{(cc')} \equiv \left(\frac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}} - q^2\right)\frac{d'(|\vec{x}_c-\vec{x}_{c'}|)}{|\vec{x}_c-\vec{x}_{c'}|}\left\{\delta_{ij} - \frac{(x_i^{(c)}-x_i^{(c')})(x_j^{(c)}-x_j^{(c')})}{|\vec{x}_c-\vec{x}_{c'}|^2}\right\}$$
$$+\frac{(x_i^{(c)}-x_i^{(c')})(x_j^{(c)}-x_j^{(c')})}{|\vec{x}_c-\vec{x}_{c'}|^2}\left\{\frac{d'^2(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}} + \left(\frac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}} - q^2\right)d''(|\vec{x}_c-\vec{x}_{c'}|)\right\} \tag{4.99}$$

therefore we can write

$$-\frac{q^4K}{2}\frac{\partial^2 F_q}{\partial x_i^{(c)}\partial x_j^{(c')}} = \begin{vmatrix} h_{ij}^{(cc')} & : & c \neq c' \\ -\sum\limits_{\substack{c''\in\{C\}\\ c''\neq c,c'}} h_{ij}^{(cc'')} & : & c = c' \end{vmatrix} \tag{4.100}$$

That expression shows clearly the relation

$$\sum_{c'\in\{C\}}\frac{\partial^2 F_q}{\partial x_i^{(c)}\partial x_j^{(c')}} = 0 \tag{4.101}$$

Omitting the global constant $q^4K/2$ we can write the significant part of the Hessian like

$$\mathrm{H}_{ij}^{(cc')} \equiv \frac{\partial^2 F_q}{\partial x_i^{(c)}\partial x_j^{(c')}} = \begin{vmatrix} -h_{ij}^{(cc')} & : & c \neq c' \\ \sum\limits_{\substack{c''\in\{C\}\\ c''\neq c,c'}} h_{ij}^{(cc'')} & : & c = c' \end{vmatrix} \tag{4.102}$$

Analogously for the gradient

$$\Gamma_i^{(c)} \equiv \frac{\partial F_q}{\partial x_i^{(c)}} = \sum_{\substack{c'\in\{C\}\\ c'\neq c}} d'(|\vec{x}_c-\vec{x}_{c'}|)\left(\frac{d(|\vec{x}_c-\vec{x}_{c'}|)}{D_{cc'}} - q^2\right)\frac{x_i^{(c)}-x_i^{(c')}}{|\vec{x}_c-\vec{x}_{c'}|} \tag{4.103}$$

4.9.2 Reductions. Sammon's method

It is possible to consider a reduced problem in which we want to optimize the functional over a subset of the total of variables. In that case is possible to use the expressions of the previous section and specialize them to the subset of interest.

Let's suppose for example that we want to optimize the functional regarding a selected dot, keeping fixed the coordinates of the rest ones. In that case the rest of variables are the d coordinates of the selected dot c and we have for the gradient

$$\Gamma_i \equiv \frac{\partial F_q}{\partial x_i^{(c)}} = \sum_{\substack{c' \in \{C\} \\ c' \neq c}} d'(|\vec{x}_c - \vec{x}_{c'}|) \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) \frac{x_i^{(c)} - x_i^{(c')}}{|\vec{x}_c - \vec{x}_{c'}|} \tag{4.104}$$

Analogously for the Hessian

$$\begin{aligned} \mathrm{H}_{ij} &\equiv \frac{\partial^2 F_q}{\partial x_i^{(c)} \partial x_j^{(c)}} = \sum_{\substack{c' \in \{C\} \\ c' \neq c}} h_{ij}^{(cc')} \\ h_{\substack{ij \\ c \neq c'}}^{(cc')} &\equiv \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) \frac{d'(|\vec{x}_c - \vec{x}_{c'}|)}{|\vec{x}_c - \vec{x}_{c'}|} \left\{ \delta_{ij} - \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{|\vec{x}_c - \vec{x}_{c'}|^2} \right\} \\ &+ \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{|\vec{x}_c - \vec{x}_{c'}|^2} \left\{ \frac{d'^2(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} + \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) d''(|\vec{x}_c - \vec{x}_{c'}|) \right\} \end{aligned} \tag{4.105}$$

In those expressions the focal dot is c and it has been omitted as superscript in the gradient and Hessian symbols. These are only keeping the indexes of the interests $i, j = 1, 2, ..., d$.

There is a second reduction if we take into account the local optimization in relation to just one coordinate $x_i^{(c)}$ of just one dot. In that case, the gradient analogous is the focal partial derivative:

$$\Gamma \equiv \frac{\partial F_q}{\partial x_i^{(c)}} = \sum_{\substack{c' \in \{C\} \\ c' \neq c}} d'(|\vec{x}_c - \vec{x}_{c'}|) \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) \frac{x_i^{(c)} - x_i^{(c')}}{|\vec{x}_c - \vec{x}_{c'}|} \tag{4.106}$$

While from the Hessian only survives the second derivative regarding the focal coordinate i of the focal dot c

$$\begin{aligned} \mathrm{H} &\equiv \frac{\partial^2 F_q}{\partial x_i^{(c)2}} = \sum_{\substack{c' \in \{C\} \\ c' \neq c}} h_{ii}^{(cc')} \\ h_{\substack{ii \\ c \neq c'}}^{(cc')} &\equiv \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) \frac{d'(|\vec{x}_c - \vec{x}_{c'}|)}{|\vec{x}_c - \vec{x}_{c'}|} \left\{ 1 - \frac{(x_i^{(c)} - x_i^{(c')})^2}{|\vec{x}_c - \vec{x}_{c'}|^2} \right\} + \\ &+ \frac{(x_i^{(c)} - x_i^{(c')})^2}{|\vec{x}_c - \vec{x}_{c'}|^2} \left\{ \frac{d'^2(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} + \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) d''(|\vec{x}_c - \vec{x}_{c'}|) \right\} \end{aligned} \tag{4.107}$$

These last expressions en the Euclidean case are reduced to

$$\Gamma^{(e)} \equiv \frac{\partial F_q}{\partial x_i^{(c)}} = \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left(\frac{|\vec{x}_c - \vec{x}_{c'}|}{D_{cc'}} - q^2 \right) \frac{x_i^{(c)} - x_i^{(c')}}{|\vec{x}_c - \vec{x}_{c'}|} \tag{4.108}$$

$$\begin{aligned} \mathrm{H} \equiv \frac{\partial^2 F_q}{\partial x_i^{(c)2}} &= \sum_{\substack{c' \in \{C\} \\ c' \neq c}} h_{ii}^{(cc')} \\ h_{\substack{ii \\ c \neq c'}}^{(cc')} &\equiv \frac{1}{D_{cc'}} - \frac{q^2}{\left|\vec{x}_c - \vec{x}_{c'}\right|} \left(1 - \frac{\left(x_i^{(c)} - x_i^{(c')}\right)^2}{\left|\vec{x}_c - \vec{x}_{c'}\right|^2} \right) \end{aligned} \tag{4.109}$$

In the Sammon's method the Newton-Raphson direction is followed for the optimization of the coordenate $x_i^{(c)}$. In accordance with the previous equations this implies an update policy at each iteration.

$$\begin{aligned} x_i^{(c)(k+1)} &= x_i^{(c)(k)} - \alpha \frac{\sum\limits_{\substack{c' \in \{C\} \\ c' \neq c}} \left(\frac{\left|\vec{x}_c^{(k)} - \vec{x}_{c'}^{(k)}\right|}{D_{cc'}} - q^2 \right) \frac{x_i^{(c)(k)} - x_i^{(c')(k)}}{\left|\vec{x}_c^{(k)} - \vec{x}_{c'}^{(k)}\right|}}{\sum\limits_{\substack{c' \in \{C\} \\ c' \neq c}} \left[\frac{1}{D_{cc'}} - \frac{q^2}{\left|\vec{x}_c^{(k)} - \vec{x}_{c'}^{(k)}\right|} \left(1 - \frac{\left(x_i^{(c)(k)} - x_i^{(c')(k)}\right)^2}{\left|\vec{x}_c^{(k)} - \vec{x}_{c'}^{(k)}\right|^2} \right) \right]} \\ c &\in \{C\} \quad i = 1, 2, ..., d \end{aligned} \tag{4.110}$$

which belongs to a variation of the Sammon's method that we obviously call q-Sammon. The constant q is the constant of Sammon, which *magic* value is $0.3 \leq \alpha \leq 0.4$ according to Kohonen's recommendation.

4.9.3 Vector q-Sammon

Obviously, it is not needed to consider as variable just one coordinate of a focal dot. If we use the whole vector of the focal dot as a variable we will have to use the first reductions of gradient and Hessian. In other words, introducing the matrix $d \times d$

$$\begin{aligned} \mathrm{H}_{ij} \equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \Bigg\{ & \left(\frac{d(\left|\vec{x}_c - \vec{x}_{c'}\right|)}{D_{cc'}} - q^2 \right) \frac{d'(\left|\vec{x}_c - \vec{x}_{c'}\right|)}{\left|\vec{x}_c - \vec{x}_{c'}\right|} \left[\delta_{ij} - \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{\left|\vec{x}_c - \vec{x}_{c'}\right|^2} \right] + \\ & \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{\left|\vec{x}_c - \vec{x}_{c'}\right|^2} \\ & \left[\frac{d'^2(\left|\vec{x}_c - \vec{x}_{c'}\right|)}{D_{cc'}} + \left(\frac{d(\left|\vec{x}_c - \vec{x}_{c'}\right|)}{D_{cc'}} - q^2 \right) d''(|\vec{x}_c - \vec{x}_{c'}|) \right] \Bigg\} \\ & i, j = 1, 2, ..., d \end{aligned} \tag{4.111}$$

and the second member vector

$$\Gamma_i \equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} d'(|\vec{x}_c - \vec{x}_{c'}|) \left(\frac{d(|\vec{x}_c - \vec{x}_{c'}|)}{D_{cc'}} - q^2 \right) \frac{x_i^{(c)} - x_i^{(c')}}{|\vec{x}_c - \vec{x}_{c'}|} \tag{4.112}$$

The solution of the linear system

$$\begin{array}{l} \sum_{j=1}^{d} \mathrm{H}_{ij} \Delta_j = \Gamma_i \\ i = 1, 2, ..., d \end{array} \tag{4.113}$$

Provides the vector q-Sammon iteration schema

$$\vec{x}_c^{(k+1)} = \vec{x}_c^{(k)} - \alpha \vec{\Delta}_c^{(k)} \tag{4.114}$$

Once the dot c is updated, it needs to update the rest of dots until the phase $k+1$ is completed.

In the Euclidean case, those expressions are simplified to

$$\begin{array}{l} \mathrm{H}_{ij} \equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left\{ \delta_{ij} \left(\frac{1}{D_{cc'}} - \frac{q^2}{|\vec{x}_c - \vec{x}_{c'}|}) \right) + q^2 \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{|\vec{x}_c - \vec{x}_{c'}|^3} \right\} \\ i, j = 1, 2, ..., d \end{array} \tag{4.115}$$

$$\Gamma_i \equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left(\frac{1}{D_{cc'}} - \frac{q^2}{|\vec{x}_c - \vec{x}_{c'}|} \right) \left(x_i^{(c)} - x_i^{(c')} \right) \tag{4.116}$$

The vector method needs the extra calculation of the previously mentioned linear system which, in general, has a very low dimension. However it allows a more efficient grouping when going over dots of the problem.

4.9.4 Spherical q-Sammon

An additional method generalization becomes if the final target space is the sphere with radius of one. That is about first case where the final space is not flat and where there are interest in some type of graphic representation of the set.

We will formally work in $d = 3$, although we will suppose other relations

$$x_1^{(c)} = \sin\theta_c \cos\varphi_c \quad ; \quad x_2^{(c)} = \sin\theta_c \sin\varphi_c \quad ; \quad x_3^{(c)} = \cos\theta_c \tag{4.117}$$

where θ, φ are respectively the polar and azimuth angles over the sphere.

We will get as a distance between dots c, c' the measurement in radians of the angle made up of the two radius to the dots over the maximum circle fixed by c, c' and the origin. Obviously we have

$$d(|\vec{x}_c - \vec{x}_{c'}|) = \cos^{-1}\left(1 - \frac{|\vec{x}_c - \vec{x}_{c'}|^2}{2}\right) \tag{4.118}$$

We also have

$$d'(|\vec{x}_c - \vec{x}_{c'}|) = \frac{1}{\sqrt{1 - \frac{|\vec{x}_c - \vec{x}_{c'}|^2}{4}}} \qquad d''(|\vec{x}_c - \vec{x}_{c'}|) = \frac{|\vec{x}_c - \vec{x}_{c'}|}{4\left(1 - \frac{|\vec{x}_c - \vec{x}_{c'}|^2}{4}\right)^{3/2}} \tag{4.119}$$

Let's take into account the general change to spherical coordinates

$$x_1 = r\sin\theta\cos\varphi \qquad x_2 = r\sin\theta\sin\varphi \qquad x_3 = r\cos\theta \tag{4.120}$$

In the tangent space we have:

$$\begin{array}{l} \frac{\partial}{\partial r} = \sin\theta\cos\varphi\frac{\partial}{\partial x_1} + \sin\theta\sin\varphi\frac{\partial}{\partial x_2} + \cos\theta\frac{\partial}{\partial x_3}\theta \\ \frac{\partial}{\partial \varphi} = -r\sin\theta\sin\varphi\frac{\partial}{\partial x_1} + r\sin\theta\cos\varphi\frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial \theta} = r\cos\theta\sin\varphi\frac{\partial}{\partial x_1} + r\cos\theta\cos\varphi\frac{\partial}{\partial x_2} - r\sin\theta\frac{\partial}{\partial x_3} \end{array} \tag{4.121}$$

Therefore, in the unit sphere:

$$\begin{array}{l} \sin\theta\cos\varphi\frac{\partial}{\partial x_1} + \sin\theta\sin\varphi\frac{\partial}{\partial x_2} + \cos\theta\frac{\partial}{\partial x_3} = 0 \\ \frac{\partial}{\partial \varphi} = -\sin\theta\sin\varphi\frac{\partial}{\partial x_1} + \sin\theta\cos\varphi\frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial \theta} = \cos\theta\cos\varphi\frac{\partial}{\partial x_1} + \cos\theta\sin\varphi\frac{\partial}{\partial x_2} - \sin\theta\frac{\partial}{\partial x_3} \end{array} \tag{4.122}$$

The first equation allows simplifications that lead to

$$x_1\frac{\partial}{\partial x_1}+x_2\frac{\partial}{\partial x_2}+x_3\frac{\partial}{\partial x_3}=0 \quad ; \quad \frac{\partial}{\partial\varphi}=x_1\frac{\partial}{\partial x_2}-x_2\frac{\partial}{\partial x_1} \quad ; \quad \frac{\partial}{\partial\theta}=-\frac{1}{\sqrt{1-x_3^2}}\frac{\partial}{\partial x_3}$$

$$\frac{\partial}{\partial x_1}=\cos\theta\cos\varphi\frac{\partial}{\partial\theta}-\frac{\sin\varphi}{\sin\theta}\frac{\partial}{\partial\varphi} \quad ; \quad \frac{\partial}{\partial x_2}=\cos\theta\sin\varphi\frac{\partial}{\partial\theta}+\frac{\cos\varphi}{\sin\theta}\frac{\partial}{\partial\varphi} \quad ; \quad \frac{\partial}{\partial x_3}=-\sin\theta\frac{\partial}{\partial\theta} \tag{4.123}$$

The three coordinates x_i are not independent and the following relations are verified

$$\frac{\partial x_i}{\partial x_j}=\delta_{ij}-x_ix_j \tag{4.124}$$

That is, the derivative of the coordinates does not make the identity matrix, but the projection over the unit sphere.

For the angular second derivatives, in Cartesian coordinates terms, it is easy to get

$$\begin{aligned}
\frac{\partial^2}{\partial\theta^2}&=\frac{1}{1-x_3^2}\left(\frac{\partial^2}{\partial x_3^2}+x_3\frac{\partial}{\partial x_3}\right)\\
\frac{\partial^2}{\partial\theta\,\partial\varphi}&=-\frac{1}{\sqrt{1-x_3^2}}\left(x_1\frac{\partial^2}{\partial x_2\partial x_3}-x_2\frac{\partial^2}{\partial x_1\partial x_3}-x_1x_3\frac{\partial}{\partial x_2}+x_2x_3\frac{\partial}{\partial x_1}\right)\\
\frac{\partial^2}{\partial\varphi^2}&=x_1^2\frac{\partial^2}{\partial x_2^2}+x_2^2\frac{\partial^2}{\partial x_1^2}-2x_1x_2\frac{\partial^2}{\partial x_1\partial x_2}-x_1\frac{\partial}{\partial x_1}-x_2\frac{\partial}{\partial x_2}\\
&=x_1^2\frac{\partial^2}{\partial x_2^2}+x_2^2\frac{\partial^2}{\partial x_1^2}-2x_1x_2\frac{\partial^2}{\partial x_1\partial x_2}+x_3\frac{\partial}{\partial x_3}
\end{aligned} \tag{4.125}$$

together with the relation of null derivative in radial direction

$$x_1^2\frac{\partial^2}{\partial x_1^2}+x_2^2\frac{\partial^2}{\partial x_2^2}+x_3^2\frac{\partial^2}{\partial x_3^2}+2x_1x_2\frac{\partial^2}{\partial x_1\partial x_2}+2x_1x_3\frac{\partial^2}{\partial x_1\partial x_3}+2x_2x_3\frac{\partial^2}{\partial x_2\partial x_3}=0 \tag{4.126}$$

The equations (4.120) and (4.122) allow the adaptation of the general expressions for gradient and Hessian to the current spherical case. Also the distance function defined at (4.118) must be used in this adaptation.

For example, focusing on the dot c it will be needed to impose

$$\frac{\partial F}{\partial\theta^{(c)}}=\frac{\partial F}{\partial\varphi^{(c)}}=0 \tag{4.127}$$

We can calculate the equations in terms of Cartesian coordinates

$$\frac{-1}{\sqrt{1-x_3^{(c)2}}}\frac{\partial F}{\partial x_3^{(c)}}=0 \qquad ; \qquad x_1^{(c)}\frac{\partial F}{\partial x_2^{(c)}}-x_2^{(c)}\frac{\partial F}{\partial x_1^{(c)}}=0 \tag{4.128}$$

While the Hessian will be

$$\mathrm{H} \equiv \begin{pmatrix} \frac{\partial^2 F}{\partial \theta_c^2} & \frac{\partial^2 F}{\partial \theta_c \partial \varphi_c} \\ \frac{\partial^2 F}{\partial \theta_c \partial \varphi_c} & \frac{\partial^2 F}{\partial \varphi_c^2} \end{pmatrix} = \begin{pmatrix} \mathrm{H}_{11} & \mathrm{H}_{12} \\ \mathrm{H}_{12} & \mathrm{H}_{22} \end{pmatrix}$$

$$\begin{aligned} \mathrm{H}_{11} &\equiv \frac{1}{1-x_3^{(c)2}} \left(\frac{\partial^2 F}{\partial x_3^{(c)2}} + x_3^{(c)} \frac{\partial F}{\partial x_3^{(c)}} \right) \\ \mathrm{H}_{22} &\equiv x_1^{(c)2} \frac{\partial^2 F}{\partial x_2^{(c)2}} + x_2^{(c)2} \frac{\partial^2 F}{\partial x_1^{(c)2}} - 2x_1^{(c)} x_2^{(c)} \frac{\partial^2 F}{\partial x_1^{(c)} \partial x_2^{(c)}} + x_3^{(c)} \frac{\partial F}{\partial x_3^{(c)}} \\ \mathrm{H}_{12} &\equiv -\frac{1}{\sqrt{1-x_3^{(c)2}}} \left(x_1^{(c)} \frac{\partial^2 F}{\partial x_2^{(c)} \partial x_3^{(c)}} - x_2^{(c)} \frac{\partial^2 F}{\partial x_1^{(c)} \partial x_3^{(c)}} - x_1^{(c)} x_3^{(c)} \frac{\partial F}{\partial x_2^{(c)}} + x_2^{(c)} x_3^{(c)} \frac{\partial F}{\partial x_1^{(c)}} \right) \end{aligned} \tag{4.129}$$

In a Newton-Raphson formulation the corrections in the phase $k+1$ are calculated by

$$\begin{pmatrix} \mathrm{H}_{11}^{(k)} & \mathrm{H}_{12}^{(k)} \\ \mathrm{H}_{12}^{(k)} & \mathrm{H}_{22}^{(k)} \end{pmatrix} \begin{pmatrix} \Delta_\theta^{(k+1)} \\ \Delta_\varphi^{(k+1)} \end{pmatrix} = \begin{pmatrix} \frac{-1}{\sqrt{1-x_3^{(c)(k)2}}} \frac{\partial F^{(k)}}{\partial x_3^{(c)}} \\ x_1^{(c)(k)} \frac{\partial F}{\partial x_2^{(c)}} - x_2^{(c)(k)} \frac{\partial F^{(k)}}{\partial x_1^{(c)}} \end{pmatrix} \tag{4.130}$$

and then

$$\theta_c^{(k+1)} = \theta_c^{(k)} - \alpha_\theta \Delta_\theta^{(k+1)} \quad ; \quad \varphi_c^{(k+1)} = \varphi_c^{(k)} - \alpha_\varphi \Delta_\varphi^{(k+1)} \tag{4.131}$$

The Cartesian coordinates in the new phase will be

$$\begin{aligned} x_1^{(c)(k+1)} &= \sin\theta_c^{(k+1)} \cos\varphi_c^{(k+1)} \\ x_2^{(c)(k+1)} &= \sin\theta_c^{(k+1)} \sin\varphi_c^{(k+1)} \quad ; \quad x_3^{(c)(k+1)} = \cos\theta_c^{(k+1)} \end{aligned} \tag{4.132}$$

The constants $\alpha_\theta, \alpha_\varphi$ can be *magic* values or deduced from some rule like for example the Armijo's one or other line search methods for determining the step size in each phase. It is supposed that the direction is always given by the solution of (4.127)

4.9.5 Vector q-Sammon for the Euclidean case

Let's consider the case of vector minimization in the Euclidean case. Let's remind here the outstanding expressions of the vector case.

$$\begin{aligned} \mathrm{H}_{ij} &\equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left\{ \delta_{ij} \left(\frac{1}{D_{cc'}} - \frac{q^2}{\left| \vec{x}_c - \vec{x}_{c'} \right|} \right) \right) + q^2 \frac{(x_i^{(c)} - x_i^{(c')})(x_j^{(c)} - x_j^{(c')})}{\left| \vec{x}_c - \vec{x}_{c'} \right|^3} \Bigg\} \\ i, j &= 1, 2, ..., d \end{aligned} \tag{4.133}$$

$$\begin{aligned} \Gamma_i &\equiv \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left(\frac{1}{D_{cc'}} - \frac{q^2}{\left| \vec{x}_c - \vec{x}_{c'} \right|} \right) \left(x_i^{(c)} - x_i^{(c')} \right) \\ i &= 1, 2, ..., d \end{aligned} \tag{4.134}$$

Fixed the focal dot c, the Newton-Raphson direction is given by the solution of the linear system

$$\begin{array}{l} \sum_{j=1}^{d} \mathrm{H}_{ij}\Delta_j = \Gamma_i \\ i = 1, 2, ..., d \end{array} \tag{4.135}$$

The update of the coordinates of the focal dot is then

$$\begin{array}{l} x_i^{(c)(k+1)} = x_i^{(c)(k)} - \alpha\Delta_i^{(c)} \\ i = 1, 2, ..., d \end{array} \tag{4.136}$$

To obtain gradient and Hessian requires $d(d+3)/2 \cdot \#\{C\}$ operations. That has to be done for all dots, so the total complexity has order $d(d+3)/2 \cdot (\#\{C\})^2$. This is a strict and hard method limitation for big training sets. The natural alternative in those cases is based on calculating the positions of the centroids given by a clusterization method. Afterwards it would be possible to put the related dots to each centroid through a local q-Sammon. Let's remind also the functional that must be minimized in q-Sammon

$$F_q\{\vec{x}\} \equiv \frac{1}{q^4 K} \sum_{\substack{c_1, c_2 \in \{C\} \\ c_1 < c_2}} \frac{\left(q^2 D_{c_1 c_2} - |\vec{x}_{c_1} - \vec{x}_{c_2}|\right)^2}{D_{c_1 c_2}} \tag{4.137}$$

When the dot $x_i^{(c)}$ is updated, the functional variation is

$$\begin{array}{l} \delta F_q^{(c)}\{\vec{x}\} = \\ \frac{2}{q^4 K} \sum_{\substack{c' \in \{C\} \\ c' \neq c}} \left(q^2 - \frac{\left|\vec{x}_c^{(k+1)} - \vec{x}_{c'}\right| + \left|\vec{x}_c^{(k)} - \vec{x}_{c'}\right|}{2D_{cc'}}\right) \left(\left|\vec{x}_c^{(k)} - \vec{x}_{c'}\right| - \left|\vec{x}_c^{(k+1)} - \vec{x}_{c'}\right|\right) \end{array} \tag{4.138}$$

That allows calculating precisely that variation with going over the training set only once.

4.10 Analysis of track descriptor data

4.10.1 Genres

This section contains the histogram of each single descriptor extracted in the music analysis process; some of the variables, like the MFCC and the Pitch Class Profiles are

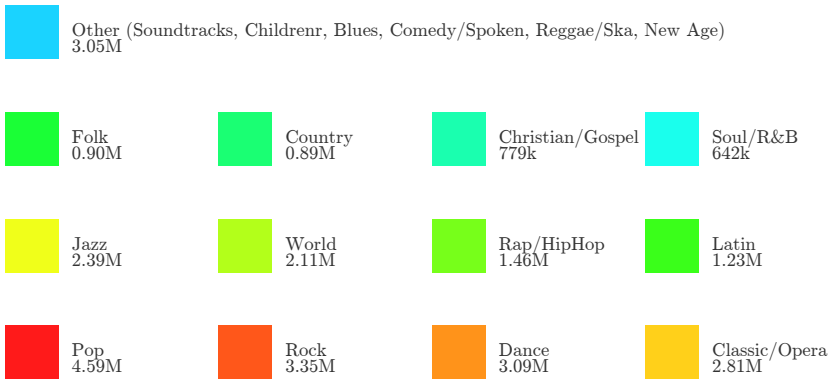


Figure 4.4: Genres used for calculating the histogram of the descriptors.

in vector form and only the first component is displayed.

In order to help visualize the genre separation properties of these descriptors, the histograms are displayed using different colors for each genre. The list of genres selected is shown in Fig. 4.4 together with the number of tracks gathered for each genre. For example, it can be pointed out that Electronic/Dance genres have lower distributions means in most of the histograms.

The main idea is that if two tracks are similar, they should probably belong to the same genre. But the opposite does not apply most of the time and it is not interesting for music playlist generation: two tracks of the same genre tend to make a bad pair when played one after the other. This is why using similarity inside the genre is extremely powerful.

4.10.2 RMS

Speech signals are considered to be slowly time varying signals. A speech signal over a period of say between 10 to 100ms has a characteristic that is fairly stationary. Over a longer period of time, however, the signal characteristics alter to reflect the changes in the speech sounds being spoken. Although music has a larger dynamic range than speech, like speech its characteristics over a short period of time remain stationary. This notion leads to a variety of short-time processing techniques in which short segments of audio signal are isolated and processed as though they were short segments from a sustained audio with fixed properties. This process of segmenting audio signals in to frames is repeated, usually periodically, as often as required. Normally these short segments, which are sometimes referred to as analysis frames, overlap one another. After processing is done on each frame, a single number or a set of numbers may be

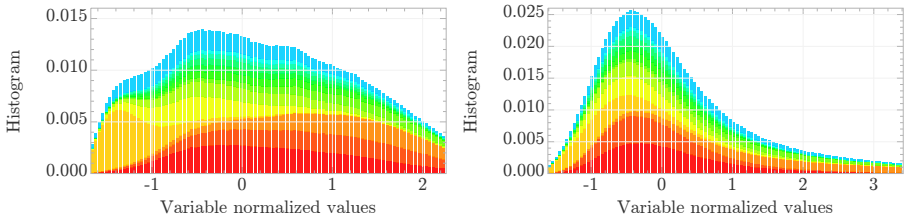


Figure 4.5: RMS mean (left) and variance (right). The color mapping can be found in Fig. 4.4.

obtained. Hence, such processing results in a new time dependent sequence that can serve as representation of the audio signal.

Short-term energy is used in different audio classification problems. In speech signals, it provides a basis for distinguishing voiced speech segments from unvoiced ones. In the case of a very high quality speech, the short-term energy features are used to distinguish speech from silence. Based on their mode of excitation, speech sounds can be classified as: voiced, unvoiced and plosive sounds. Voiced sounds are produced by forcing air through the glottis with the tension of the vocal cords adjusted so that they vibrate in a relaxation oscillation. Unvoiced sounds are produced by forming constrictions at some point in the vocal tract, and forcing air through the constriction at a high velocity enough to produce turbulence.

The histograms of these descriptor can be found in Fig. 4.5.

4.10.3 Spectral Centroid

The spectral centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the *center of mass* of the spectrum is. Perceptually, it has a robust connection with the impression of *brightness* of a sound. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights. It is widely used in digital audio and music processing as an automatic measure of musical timbre. The histograms of these descriptor can be found in Fig. 4.6.

4.10.4 Spectral Bandwidth

Measure of how tightly is the spectrum centered around the spectral centroid. A sine wave has a spectral bandwidth of 0, while a white noise has a large value.

See histograms of these descriptor in Fig. 4.7.

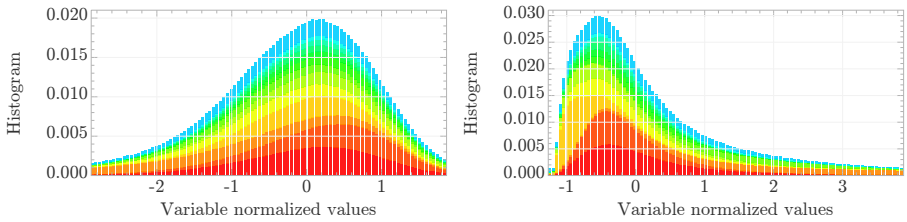


Figure 4.6: Spectral centroid mean (left) and variance (right). The color mapping can be found in Fig. 4.4.

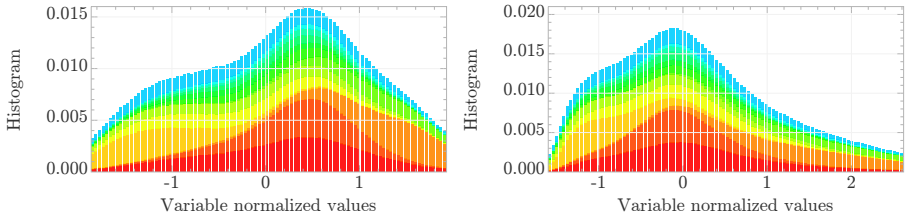


Figure 4.7: Spectral Bandwidth mean (left) and variance (right) Variables

4.10.5 MFCC

MFCCs are short term spectral based features. MFCC features are frequently used by many researchers for speech recognition and it has also been shown that MFCC works well in music/ speech classification problem. Each step in this process of creating Mel Frequency Cepstral Coefficients is motivated by computational or perceptual considerations. The histograms of these descriptor can be found in Fig. 4.9.

4.10.6 Onset Long-term descriptors

For each onset found, the max amplitude reached and the max log-amplitude derivative is computed. The final values used are:

- *Onset density*: number of onsets per second (Fig. 4.10).
- *Onset intensity*: mean of the max amplitude reached by every onset (Fig. 4.11).
- *Onset steepness*: mean of the max log-amplitude derivative of every onset Density, Intensity and Steepness.

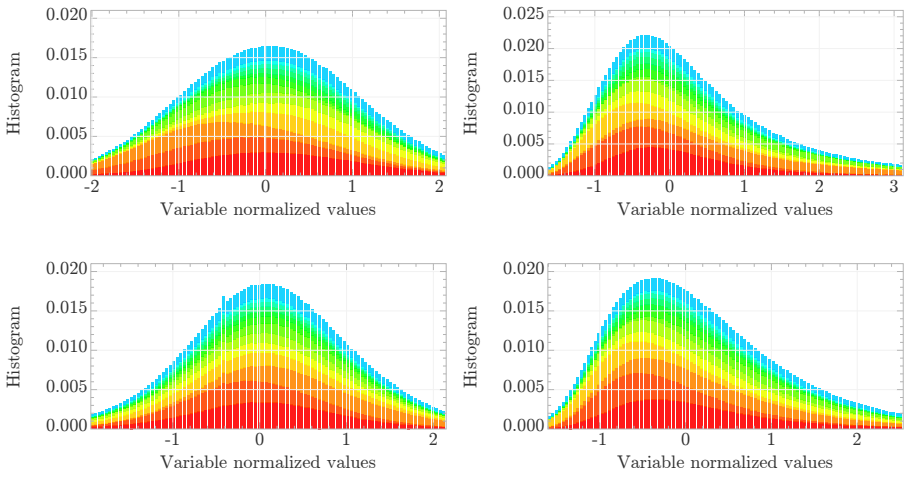


Figure 4.8: Variables 7-12 Mel Frequency Cepstrum Coefficients (MFCCs). Mean (left) and Variance (right)

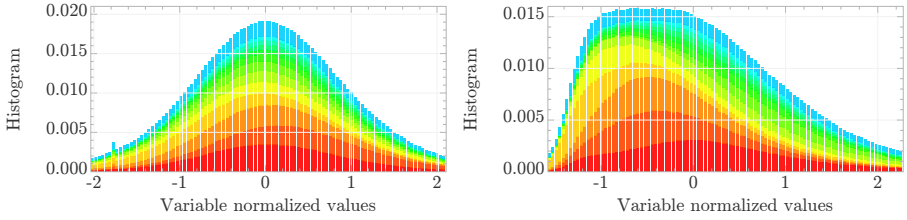


Figure 4.9: Variables 7-12 Mel Frequency Cepstrum Coefficients (MFCCs). Mean (left) and Variance (right)

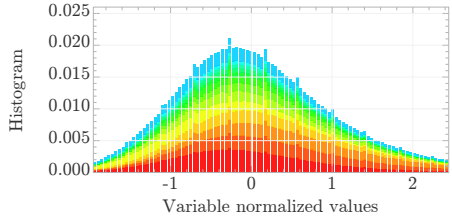


Figure 4.10: Variable 33 - Onset density

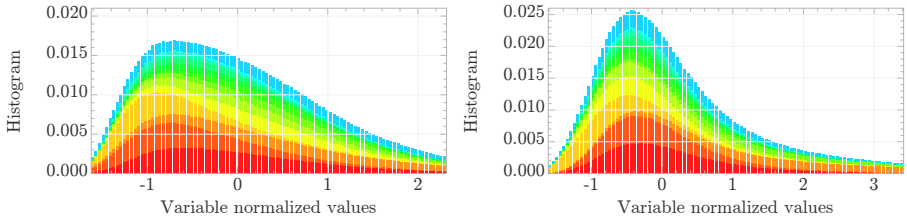


Figure 4.11: Variables 34-35 Onset Energy. Mean (left) and Variance (right)

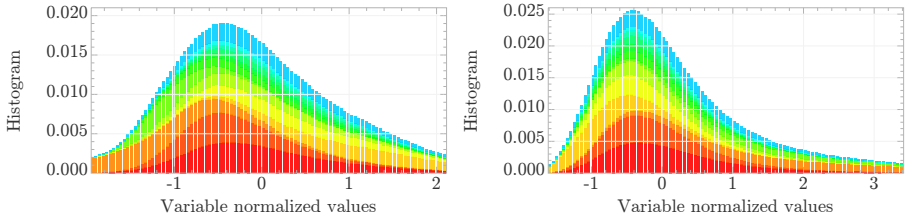


Figure 4.12: Variables 36-37 Aggressiveness (left) Focus (right)

4.10.7 Spectral fluctuation patterns

These descriptors provide a measure of the amplitude modulation of the loudness per frequency band. Fluctuation patterns (FP) is computed by performing a FFT on the energy envelope of each of twelve frequency bands, using 6s-long windows with 3s overlap. Every frequency spectrum obtained is weighted by a function modeling the perception of the fluctuation strength. The following features are extracted:

- gravity: describes the center of gravity of the FP on the modulation frequency axis. Low values indicate that the piece might be perceived as slow.
- focus: describes the distribution of energy in the FP. In particular, the focus is low if the energy is focused in small regions of the FP, and high if the energy is spread out over the whole FP.
- aggressiveness: the aggressiveness is measured as the ratio of the sum of values within the frequency bands 2-12 and modulation frequency below 1Hz compared to the sum of all. Less aggressive pieces have higher values (Fig. 4.12).
- max: the maximum fluctuation strength is the highest value in the rhythm pattern. Pieces of music which are dominated by strong beats have very high values (Fig. 4.13).
- bass: the bass is calculated as the sum of the values in the two lowest frequency bands with a modulation frequency higher than 1Hz (Fig. 4.14).

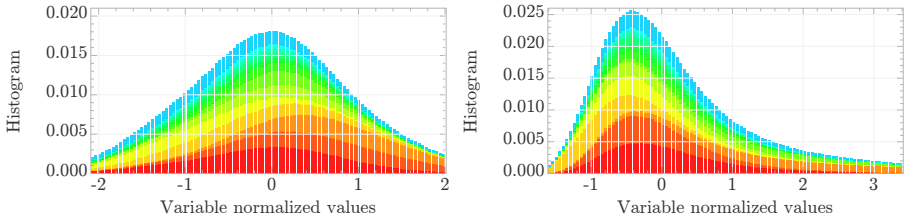


Figure 4.13: Variables 38-39 Gravity (left) Focus (MaxFluctuations)

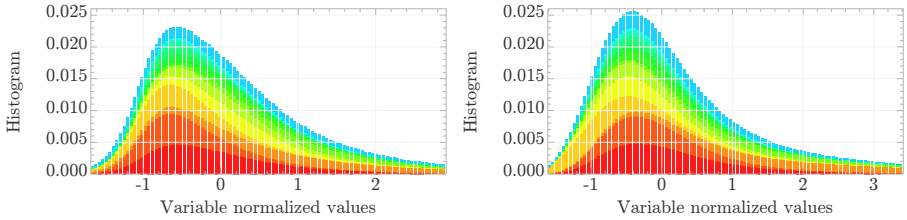


Figure 4.14: Variables 40-41 Bass (left) Focus (BassRatio)

4.10.8 Pitch Class Profile

Harmonic pitch class profiles (HPCP) is a vector of features extracted from an audio signal, based on the Pitch Class Profile descriptor proposed in the context of a chord recognition system. HPCP are enhanced pitch distribution feature which are sequences of feature vectors describing tonality measuring the relative intensity of each of the 12 pitch classes of the equal-tempered scale within an analysis frame. It is also called Chroma. It is in the intensity of each of the twelve semitones of the tonal scale mapped to a single octave. Just the first two histograms of these descriptor to summarize it in Fig. 4.15.

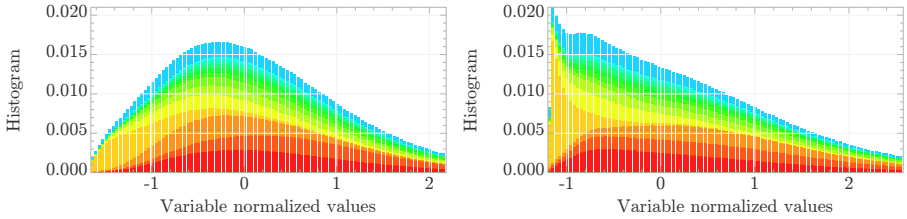


Figure 4.15: Variables 42-47 Pitch Class Profile. Mean (left) and Variance (right)

4.11 Euclidean Metric Results

4.11.1 Normalization

In statistics and applications of statistics, normalization can have a range of meanings. In the simplest cases, normalization of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging. In more complicated cases, normalization may refer to more sophisticated adjustments where the intention is to bring the entire probability distributions of adjusted values into alignment. In the case of normalization of scores in educational assessment, there may be an intention to align distributions to a normal distribution. A different approach to normalization of probability distributions is quantile normalization, where the quantiles of the different measures are brought into alignment.

In statistics, the standard score is the (signed) number of standard deviations an observation or datum is above the mean. Thus, a positive standard score indicates a datum above the mean, while a negative standard score indicates a datum below the mean. It is a dimensionless quantity obtained by subtracting the population mean from an individual raw score and then dividing the difference by the population standard deviation. This conversion process is called standardizing or normalizing.

Standard scores are also called z-values, z-scores, normal scores, and standardized variables; the use of Z is because the normal distribution is also known as the Z *distribution*. They are most frequently used to compare a sample to a standard normal deviate, though they can be defined without assumptions of normality. The z-score is only defined if one knows the population parameters; if one only has a sample set, then the analogous computation with sample mean and sample standard deviation yields the Student's t-statistic.

$$\text{Standard Score: } \frac{X - \mu}{\sigma} \tag{4.139}$$

$$\text{Student's t-statistic: } \frac{X - \bar{X}}{s} \tag{4.140}$$

Tables 4.1 and 4.2 show the normalization values (both mean and standard deviation) used to calculate the Standard score for each descriptor. The descriptors of 30M tracks have been averaged to obtain this normalization tables.

4.11.2 Supervised benchmark similarity database

To evaluate the a priori recommendation accuracy of the selected descriptors, a K-nearest-neighbor (kNN) classification has been use. It is one of the most fundamental and simple classification methods and should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution of the data.

index	Descriptor	Mean	Standard Deviation
0	RMS-Mean	2.674141e3	1.475037e3
1	RMS-Var	8.138944e2	4.649421e2
2	Spectral-Centroid-Mean	6.593846e0	4.105022e-1
3	Spectral-Centroid-Var	3.575492e-1	2.825491e-1
4	Spectral-Bandwidth-Mean	1.617863e0	3.753957e-1
5	Spectral-Bandwidth-Var	1.318431e-1	7.596398e-2
6	MFC0-Mean	4.721864e1	1.380131e1
7	MFC0-Var	2.075286e2	9.399280e1
8	MFC1-Mean	3.959379e0	9.251179e0
9	MFC1-Var	1.563354e2	6.985190e1
10	MFC2-Mean	1.027542e1	5.689372e0
11	MFC2-Var	1.275870e2	5.843014e1
12	MFC3-Mean	1.239127e0	4.923951e0
13	MFC3-Var	9.087221e1	3.672967e1
14	MFC4-Mean	2.821427e0	3.703834e0
15	MFC4-Var	6.839768e1	2.216627e1
16	MFC5-Mean	1.085858e0	3.241086e0
17	MFC5-Var	6.351532e1	1.999761e1
18	MFC6-Mean	4.357739e-1	3.446614e0
19	MFC6-Var	5.903292e1	1.938437e1
20	MFC7-Mean	6.642397e-1	2.684565e0
21	MFC7-Var	5.366766e1	1.773221e1
22	MFC8-Mean	6.877589e-1	2.438237e0
23	MFC8-Var	5.011518e1	1.770173e1
24	MFC9-Mean	6.409841e-1	2.142333e0
25	MFC9-Var	4.541682e1	1.634233e1
26	MFC10-Mean	4.548450e-1	1.893994e0
27	MFC10-Var	4.080434e1	1.430884e1
28	MFC11-Mean	4.832456e-1	1.692496e0
29	MFC11-Var	3.609134e1	1.187996e1
30	MFC12-Mean	6.198565e-1	1.694398e0
31	MFC12-Var	3.245677e1	1.045200e1

Table 4.1: First half of the normalization values used to calculate the Standard score

index	Descriptor	Mean	Standard Deviation
32	Onset-Density	5.024235e0	8.971875e-1
33	Onset-Energy1	7.143283e0	2.344498e0
34	Onset-Energy2	7.867800e0	8.728197e-1
35	Aggressiveness	1.199068e-2	3.723777e-3
36	Focus	2.521377e2	7.533644e1
37	Gravity	5.481006e0	3.283426e-1
38	Max-Fluctuations	6.103916e-2	2.852040e-2
39	Bass	2.496988e-2	6.464097e-3
40	Bass-Ratio	1.819118e-3	2.317415e-4
41	PCP0-Mean	1.993988e2	1.136251e2
42	PCP0-Var	3.062182e5	2.562740e5
43	PCP1-Mean	1.771232e2	1.000335e2
44	PCP1-Var	2.733056e5	2.310185e5
45	PCP2-Mean	1.829966e2	1.057615e2
46	PCP2-Var	2.724597e5	2.298843e5
47	PCP3-Mean	1.916012e2	1.054944e2
48	PCP3-Var	2.935391e5	2.420544e5
49	PCP4-Mean	1.785400e2	9.837348e1
50	PCP4-Var	2.759995e5	2.313447e5
51	PCP5-Mean	1.974951e2	1.082945e2
52	PCP5-Var	2.986644e5	2.432942e5
53	PCP6-Mean	1.805182e2	9.811389e1
54	PCP6-Var	2.810987e5	2.350222e5
55	PCP7-Mean	2.052898e2	1.118076e2
56	PCP7-Var	3.205684e5	2.676035e5
57	PCP8-Mean	2.099002e2	1.131094e2
58	PCP8-Var	3.454165e5	2.999977e5
59	PCP9-Mean	2.089912e2	1.165614e2
60	PCP9-Var	3.524802e5	3.145376e5
61	PCP10-Mean	2.157226e2	1.204980e2
62	PCP10-Var	3.552003e5	3.051712e5
63	PCP11-Mean	1.921801e2	1.091857e2
64	PCP11-Var	3.104003e5	2.674493e5

Table 4.2: Second half of the normalization values used to calculate the Standard score

K-nearest-neighbor classification was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine.

A basic rule in classification analysis is that class predictions are not made for data samples that are used for training or learning. If class predictions are made for samples used in training or learning, the accuracy will be artificially biased upward. Instead, class predictions are made for samples that are kept out of training process. The tracks of Table 4.3 where not included in the normalization process even though their impact will be minimum (500 tracks over 30M tracks).

The input consists of the k closest training examples in the feature space and the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor. *k-NN* is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis. This is basically what the author has carried out with the estimation of local Riemann metric.

A drawback of the basic *majority voting* classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. k-NN can then be applied to the SOM.

All 500 tracks in the supervised database have a subgenre that has been assigned with the intervention of a human expert. Using a simple Euclidean metric k-NN classification, it is possible to see how many of the n most closest tracks to a given seed track belong to the same subgenre than the seed track. Fig. 4.16 shows the output of this

Subgenre	Example size
ambient	19
blues-rock	10
choir-slow	18
clarinet-accordion-jazz-applause	6
crimson-project	4
dance-90s	10
disco	7
flamenco	9
funky-JBs	5
gospel	10
guitar-hero-ballad	10
guitar-voice-french	9
guitar-voice-seu-jorge	6
heavy-metal	17
hip-hop-wu-tan	21
hiromitsu	8
jazz-trio	31
latin-dance	13
opera	26
percussion	14
piano-violin-classical	16
piano-voice-tom-waits	15
pianosolo-classical-19th20th	50
pop	11
pop-dance	8
raggae	45
rock-n-roll	23
rumba	8
sitar-tabla	5
string-quartet	13
symphonies	10
tango-piano-accordion	5
trip-hop-female	16
voice-tambourine	8
yiddish	6

Table 4.3: Number of tracks of each subgenres used during Euclidean metric evaluation

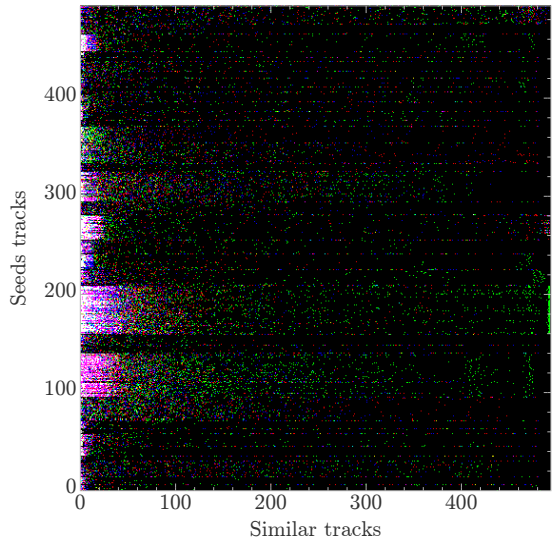


Figure 4.16: Similarity matrix obtained with Euclidean metric using the waveform/uncompressed analysis algorithm

classification. Every row of the matrix represents a single-track recommendation. The similar tracks are placed from left to right with the most similar using Euclidean distance in the left. The black pixels represent tracks with the same subgenre as the seed track. It is possible to see that the Euclidean metric can be used for recommendation as an initial filtering phase but that its accuracy varies greatly for different subgenres. Fig. 4.17 summarizes (in green) the similarity accuracy obtained extracting the music descriptors from Waveform uncompressed audio files. The first tracks returned by the k-NN classification is 80% times of the same subgenre as the seed track. For the second most similar, the accuracy is reduced to 78%.

4.11.3 MP3 vs WAV formats

The first release of the music analysis algorithm was written to extract the information from uncompressed audio files. The common WAV audio format is uncompressed audio in the linear pulse code modulation (LPCM) format. LPCM is also the standard audio coding format for audio CDs, which store two-channel LPCM audio sampled 44,100 times per second with 16 bits per sample. Since LPCM is uncompressed and retains all of the samples of an audio track, professional users or audio experts may use the WAV format with LPCM audio for maximum audio quality. WAV files can also be edited and manipulated with relative ease using software.

Uncompressed WAV files are large, so file sharing of WAV files over the Internet is uncommon. However, it is a commonly used file type; suitable for retaining first

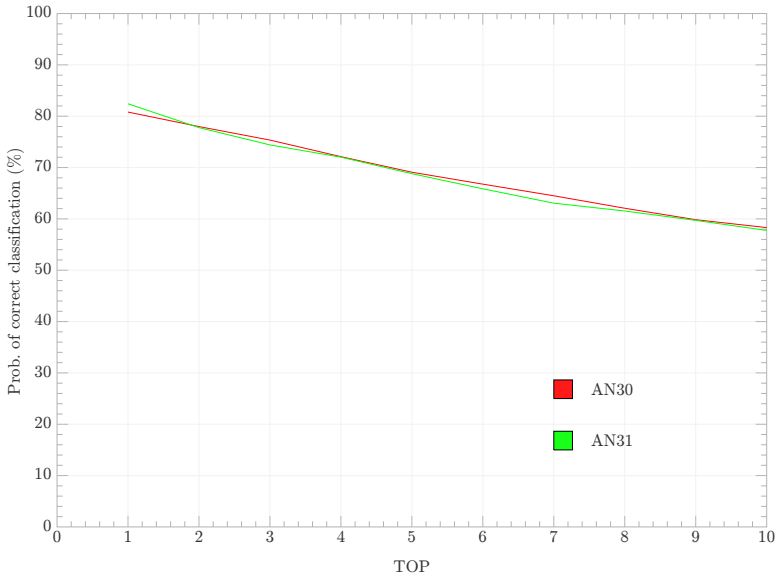


Figure 4.17: Performance of the Euclidean metric k-NN classification for uncompressed and compressed analysis algorithm

generation archived files of high quality; for using it on a system where disk space is not a constraint; or in applications such as audio editing, where the time involved in compressing and uncompressing data is a concern.

More frequently, the smaller file sizes of compressed but lossy formats such as MP3 are used to store and transfer audio. Their small file sizes allow faster Internet transmission, as well as lower consumption of space on memory media. There are also lossless-compression formats such as FLAC.

The usage of the WAV format has more to do with its familiarity and simple structure. Because of this, it continues to enjoy widespread use with a variety of software applications, often functioning as a *lowest common denominator* when it comes to exchanging sound files among different programs.

Compressed audio removes the information that is not significant to our brain. It is important to analyze if this information is also contaminating our descriptors or if it produces any important bias.

What makes MP3 encoding effective as a method of audio data compression is its deviation from the PCM model. As we have seen, in a PCM system the goal is to digitally reproduce the waveform of an incoming signal as accurately as is practically possible. However, it could be argued that the implicit assumption of PCM - namely that the reproduction of sound requires the reproduction of waveforms - is simplistic, and involves a misunderstanding of the way human perception actually works.

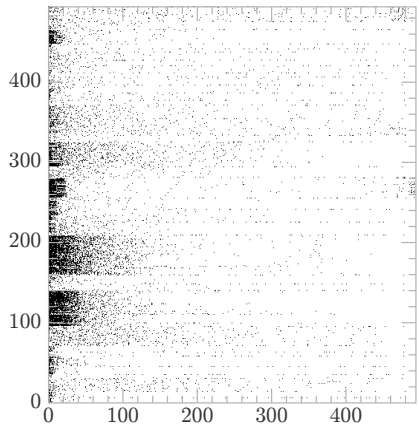


Figure 4.18: Similarity matrix obtained with Euclidean metric using the MP3 analysis algorithm

The fact of the matter is that our ears and our brains are imperfect and biased measuring devices, which interpret external phenomena according to their own prejudices. It has been found, for example, that a doubling in the amplitude of a sound wave does not necessarily correspond with a doubling in the apparent loudness of the sound. A number of factors (such as the frequency content of the sound, and the presence of any background noise) will affect how the external stimulus comes to be interpreted by the human senses. Our perceptions therefore do not exactly mirror events in the outside world, but rather reflect and accentuate certain properties of those events.

The psycho-acoustic model depends upon a particular peculiarity of human auditory perception: an effect known as masking. Masking could be described as a tendency in the listener to prioritize certain sounds ahead of others, according to the context in which they occur. Masking occurs because human hearing is adaptive, and adjusts to suit the prevailing levels of sound and noise in a given environment. For example, a sudden hand-clap in a quiet room might seem startlingly loud. However, if the same hand-clap was immediately preceded by a gunshot, it would seem much less loud. Similarly, in a rehearsal-room recording of a rock band, the sound of an electric guitar might seem to dominate the mix, up until the moment the drummer hits a particular cymbal - at which point the guitar might seem to be briefly drowned out. These are examples of *time-domain* and *frequency-domain* masking respectively. When two sounds occur simultaneously or near-simultaneously, one may be partially masked by the other, depending on factors such as their relative volumes and frequency content.

Masking is what enables perceptual coding to get away with removing much of the data that conventional waveform coding would store. This does not entail discarding all of the data describing masked elements in a sound recording: to do so would probably sound bizarre and unpleasant. Instead, perceptual coding works by assigning fewer

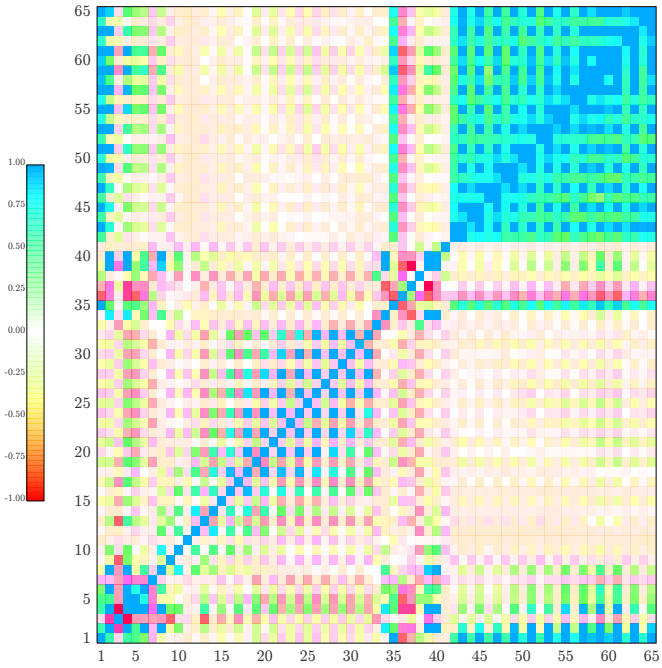


Figure 4.19: Correlation obtained from the Principal Component Analysis for the 65 descriptors

bits of data to the masked elements of a recording than to the *relevant* ones. This has the effect of introducing some distortion, but as this distortion is (hopefully) confined to the masked elements, it will (hopefully) be imperceptible on playback. Using fewer bits to represent the masked elements in a recording means that fewer bits overall are required. This is how MP3 coding succeeds in reducing audio files to be around one-tenth of their original size, with little or no noticeable degradation in sound quality.

The third version of the analyst algorithm works directly from decoded MP3 files. The redundant and masked information was also analyzed by previous versions of the analyst algorithm but as it is shown from Fig. 4.17 the accuracy of both algorithms is very similar, therefore it is possible to conclude that the new generation of the analysis algorithm maintains the performance while increasing processing speed. Processing speed is very important when having to analyze track databases of dozens of millions. Fig. 4.18 displays the similarity matrix for the benchmark track database using the MP3 descriptor analysis algorithm.

4.11.4 Principal component analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into

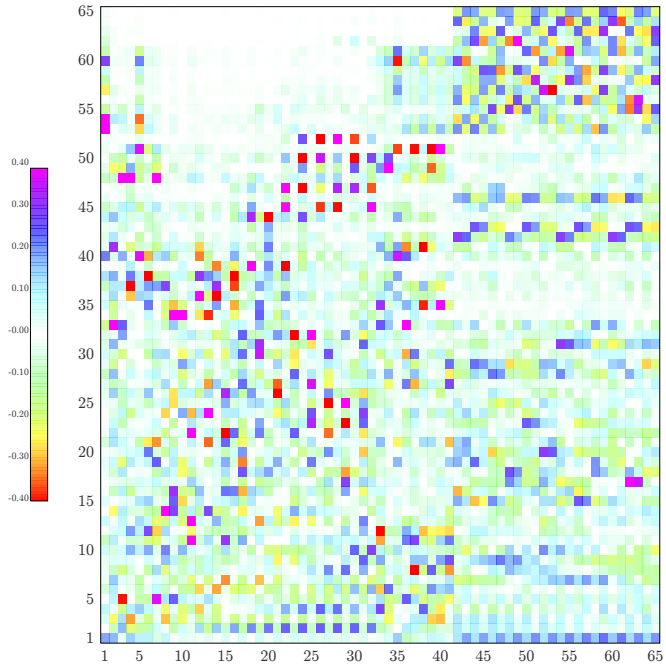


Figure 4.20: Eigen Vectors obtained from the Principal Component Analysis for the 65 descriptors

a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

In probability theory and statistics, a covariance matrix (also known as dispersion matrix or covariance matrix) is a matrix whose element in the i, j position is the covariance between the i-th and j-th elements of a random vector (that is, of a vector of random variables). Each element of the vector is a scalar random variable, either with a finite number of observed empirical values or with a finite or infinite number of potential values specified by a theoretical joint probability distribution of all the random variables. Fig. 4.19 shows the correlation of the 65 descriptors used and Fig. 4.20 the obtained Eigenvectors. High correlation exists between the PCP descriptors but within the MFCC vector.

The main linear technique for dimensionality reduction, the principal component analysis, performs a linear mapping of the data to a lower-dimensional space in such a

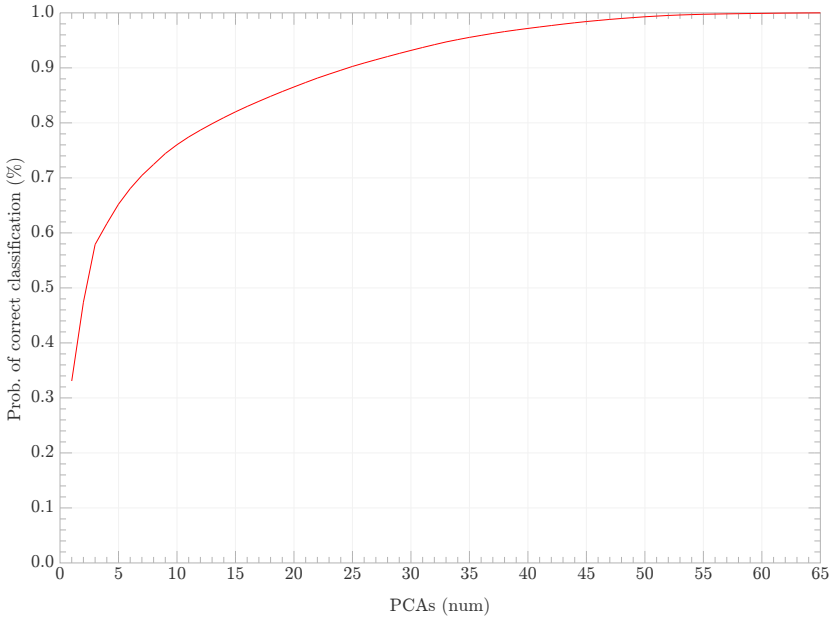


Figure 4.21: Accumulated sum of the eigenvalues. With the first 25 principal components it is possible to reconstruct the original space with an accuracy of 90%

way that the variance of the data in the low-dimensional representation is maximized. In practice, the correlation matrix of the data is constructed and the eigenvectors on this matrix are computed. The eigenvectors that correspond to the largest eigenvalues (the principal components) can now be used to reconstruct a large fraction of the variance of the original data. Moreover, the first few eigenvectors can often be interpreted in terms of the large-scale physical behavior of the system. The original space (with dimension of the number of points) has been reduced (with data loss, but hopefully retaining the most important variance) to the space spanned by a few eigenvectors.

For high-dimensional datasets (i.e. with number of dimensions more than 10), dimension reduction is usually performed prior to applying a k-nearest neighbors algorithm(k-NN) in order to avoid the effects of the curse of dimensionality. The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.

There are multiple phenomena referred to by this name in domains such as numerical analysis, sampling, combinatorial, machine learning, data mining and databases. The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the amount of data needed to sup-

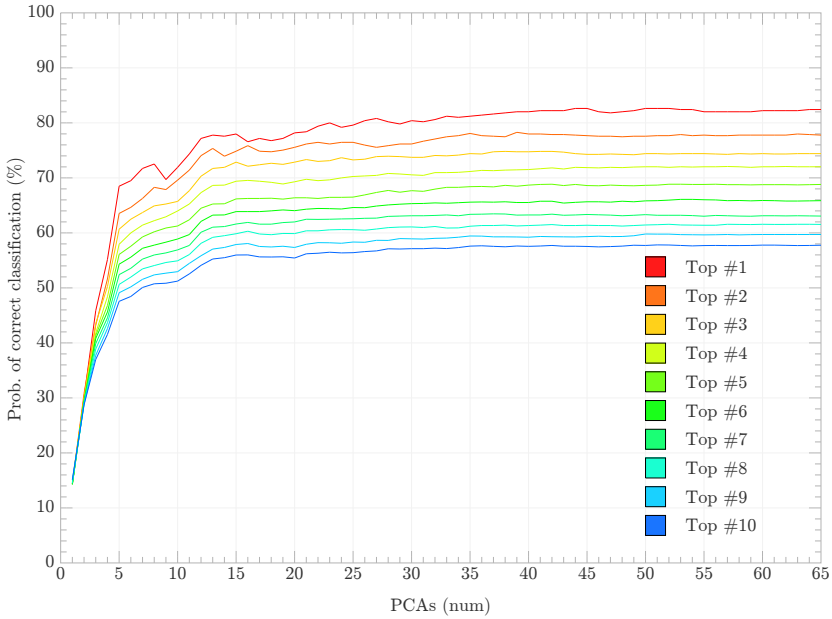


Figure 4.22: Similarity performance as a function of the number of principal component selected

port the result often grows exponentially with the dimensionality. Also organizing and searching data often relies on detecting areas where objects form groups with similar properties; in high dimensional data however all objects appear to be sparse and dissimilar in many ways which prevents common data organization strategies from being efficient.

By looking to the accumulated sum of Eigenvalues (seeFig. 4.20), it can be predicted that PCA will reduce the dimension with a limited impact in the quality of the recommendations. Basically, using the first 25 PCA components, the original space can be reproduced with an accuracy of 90%.

In next chapter the local Riemann metric will be estimated using a PCA space of dimension 25 instead of the original 65. Fig. 4.22 summarizes the effect of the use of a PCA transformation by plotting the achieve accuracy based on the number of PCA used. As stated before, with 25 PCA components a 80% similarity accuracy is obtained for the closest track using k-NN classificator.

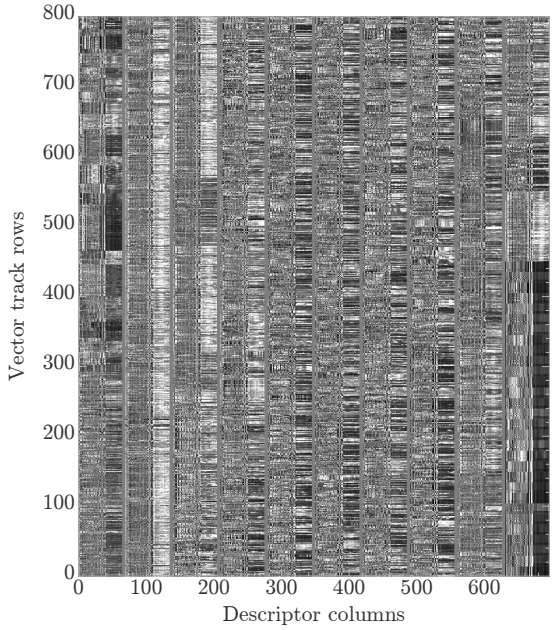


Figure 4.23: View in stripes of the 8996 tracks used in the training similarity matrix

4.12 Riemann metrics Music recommendation algorithm

4.12.1 Training database

The main idea of this algorithm is to be able to use the best metric in each part of the music descriptor space. For example, rythm desriptor might be more useful for similarity measurements for some genres than for others. For the algorithm to capture this variations it is necessary to provide with similarity examples and therefore construct a training database by selected tracks that cover the entire space (kindly called *Music Universe*).

For the average music lover / listener it is very difficult to determine if two tracks are similar or different because it depends on the context. To make decisions more robust it is always a better idea to show to the user three tracks and ask which are the two more similar ones and the two least similar. In other words, if we select one of the tracks as the *seed* or *reference* track, the user must answer which ot the two other tracks is most similar to that reference track. With this scenario, user answers are much more consistent.

During months, we have constructed a music training database consisting on 8996

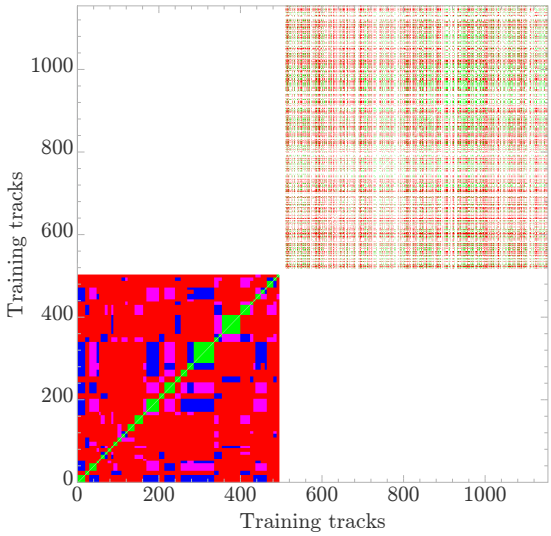


Figure 4.24: Zoom over the first 1k quadrant of the supervised training similarity matrix (Green similar, red different)

tracks. Fig 4.23 shows the 65 descriptors of all those tracks. In order to selected the training tracks, a proprietary clustering algorithm has been used together with some constraints such as trying to keep the genre proportions similar to the global track sample, so we would expect much more similarity pairs of *pop* music than *metal* music. By using the above mentioned *triangular principle* users have been listening randomly to group of 3 tracks and we have stored their supervised similarity scores in a similarity matrix. Fig 4.24 shows a zoom of the first quadrant of this database. The possibles scores where 0 meaning it is the same song, 1 very similar, 2 similar, 3 different, 4 very different.

4.12.2 Background on machine learning

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-

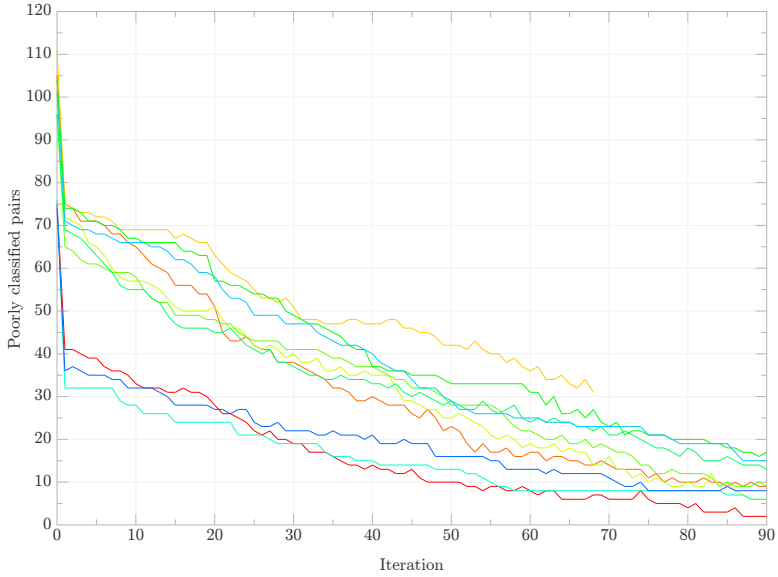


Figure 4.25: Poorly classified pairs during evaluation

linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. As described in previous sections of this chapter, the Riemann metric is estimated using a linear SVM. The curvature matrix DxD is transformed to a vector containing all its elements (therefore of size N^2). In each iteration, the algorithms tries to move through the gradient that increases the performance but at the same time it is verified that the solution does not produce an hyperbolic metric. At ever iteration, the metric can be used to compare two training supervised examples to see how many of the closest similarity pairs are classified together (this means, that a similar pair produces less distance than a different pair). Therefore the algorithm starts with an original number of pairs and tries to minimize the number of wrongly classified pairs. Fig 4.25 shows the classification performance in each iterations for different seeds in different regions of the track descriptor space.

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are

designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $xk(x, y)$ selected to suit the problem.

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes (in our case these two tracks are similar or these two tracks are different), and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p - 1)$ dimensional hyperplane. This is called a linear classifier. There are many hyper-planes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability.

4.12.3 Description of the algorithm

Given a *reference* track and a huge collection of tracks, the goal of this algorithm is to find similar tracks to the reference track within the collection of tracks. At the same time, the similar tracks must be accompanied of a distance measure that can be used to infer the music similarity closeness to the seed track. Basically for distances lower than 1, we expect the tracks to be similar while for differences higher we expect the track to be different to the seed track.

Firstly, the algorithm uses normalized Euclidean Metric to find which tracks of the supervised database are closer and a big portion of the supervised database is used (about 20%). Then Principal Component vectors are estimated locally by subtracting the reference 65 track descriptors to the selected portion of the supervised database and transforming it to a 25 dimension PCA space. Therefore the principal component

analysis produces different Eigenvectors for different reference tracks.

```
Data: Seed track, List of source tracks
Result: Selection of similar tracks with associate distance weight
apply local principal component analysis
find closest supervised pairs;
while true do
    evaluate;
    if all pairs are well classified then
        break;
    end
    solve;
    update;
    if is metric hyperbolic then
        break;
    end
end
calibrate distance;
```

Algorithm 1: Riemann metric optimization algorithm

In this stage the algorithm looks for the closest similar and different pairs in the supervised training database but using Euclidean distance on the 25 local PCA space. Initially the Euclidean metric is used on PCA space and all pairs are compared. An optimum threshold is calculated so the distance of the similar pairs is in average lower than the distance of the different pairs. All the similar pairs that have a distance higher than the threshold plus the different pairs that have a distance lower than the threshold are considered poorly classified pairs. The goal of the algorithm is to find a curvature of the PCA metric that pushes similar pairs below the threshold and different pairs above the threshold.

Fig 1 outlines the high-level diagram of the algorithm in pseudo-code. The final part is to iteratively follow the gradient towards the optimum point that maximizes class separation (between similar and different pairs). However, the updates are performed incrementally to ensure that an hyperbolic metric is not achieved as it would invalidate the original notion of local distance.

Once optimal non-hyperbolic Riemann metric is ready, it is multiplied by a factor so that the similarity threshold is 1. Fig 4.26 shows the average and standard deviation metric components for the supervised database. At this point, the algorithm has produced some local PCA Eigenvectors that can be used in that region of the space together with a curvature for this local PCA space that maximizes the class separation between similar and different tracks.

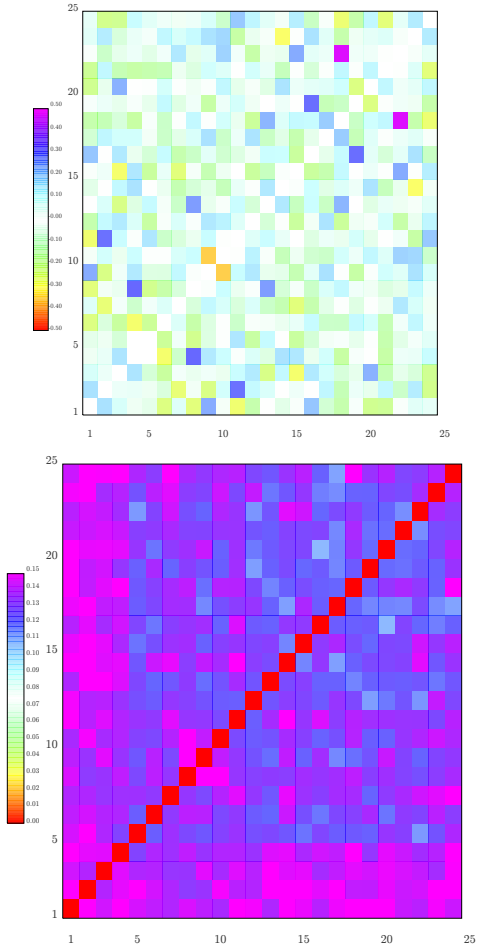


Figure 4.26: Metric average (top) and Metric standard deviation (bottom) when the algorithm is auto-applied to the tracks in the supervised training database

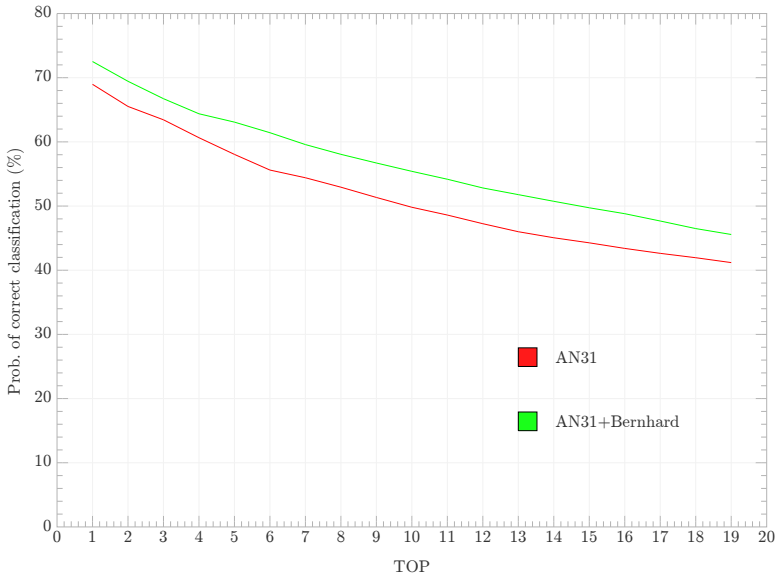


Figure 4.27: Curvature results

4.13 Results

Fig 4.27 shows the performance improvement achieved using the proposed recommendation algorithm. A 5% accuracy increase is obtained over the range of nearer tracks. The Riemann metric estimation algorithm can be used to refine a list of candidate similar tracks produced by the normalized Euclidean distance. The process that calculates locally the Principal Component Analysis together with the Metric optimization iterative algorithm are intensive from a computational point of view compared to a simple Euclidean Distance but the boost in similarity quality that they produce are greatly noticeable.

The improvement is mainly possible a meaningful context is calculated in which the metric can be used locally. The supervised training database can be increased with user feedback scores (likes and dislikes) and re-calculated in real-time so it also creates a framework for personalized music recommendation.

4.14 Summary

In the beginning of 2015, there are more than 100M available tracks through Internet streaming (35M commercial and more than 80M unlicensed). The analyst described in Chapter 2 produces 65 descriptor for each analyzed track. The space of dimension 65 that results is very sparse and similarity (i.e. based on k-nearest neighbors) is only ac-

curate locally. The goal of this chapter has been to present the Riemann metric theory to measure the extension and the accuracy of the music recommendation algorithm.

A basic music recommendation starts with a given seed or reference track that the user selects. Then the algorithm has to find similar tracks within the 100M track collection. In this dissertation, the authors use k-NN based on Euclidean metric to select a big list of initial similar track candidates. Then this list is refined and sorted twice. The first sorting algorithm is based on the Riemann metric (this chapter) while the second and last sorting algorithm is based on an Immunology approach (Chapter 5).

The main idea of the presented algorithm is based on the following assumptions:

- Euclidean Metric produces good recommendations for small distances.
- As the the Euclidean distance between the seed track and a candidate track increases, the accuracy of the recommendation vanishes. More importantly, the given distance cannot be anymore to sort by similarity.
- Some descriptors are more critical in some genres or music styles.

The goal of the algorithm is to estimate the Riemann (curved) metric that will be used to sort a list of candidates by similarity to the seed track. Each seed track produces its own Riemann metric. The produced metric is also calibrated so the threshold between similar and different track is 1. The metric estimation is performed in a reduced PCA space of dimension 25 instead of the 65 original dimension to increase the speed of the algorithm as it has to crawl collections of million of tracks.

The metric is curved incrementally following an optimization gradient and the metric matrix is vectorized in order to find the incremental updates using Support Vector Machines (SVM) technique. To train the SVM a huge supervised database of similar and different tracks is used and the algorithm iteratively reduces bad recommendations (similar tracks having higher distance to the seed than different tracks). The overall improvement of this approach increases a 5% the accuracy of the benchmark recommendations.

Chapter 5

Immune Artificial Systems for Music Recommendation

- Spike 1-2ms
- Frame 40ms a.k.a. *millisound*
- Note 500ms
- Theme 10000ms
- Episode 100000ms
- Song 300000ms

5.1 Introduction

Immunology is the branch of biomedicine that is concerned with the structure and function of the immune system, innate and acquired immunity, and laboratory techniques involving the interaction of antigens with antibodies. Although this seems to be far away from the target of this researching, it has added an amazing approach for improving the accuracy of results.

Using all approaches explained in previous chapters, including sound signal analysis and advanced mathematical physics like the Riemann curvature method; the recommendation algorithms achieved a glass ceiling on the output accuracy. The immunology concept applied to this methodology tries to break that glass ceiling and improve the accuracy removing all those false positive of the results identifying them in the same way that an antibody identifies an antigen with perfect matching of their forms.

But let's first explain more in depth how the immune system works in order to understand the immunology approach applied to this researching.

Human bodies and higher animals are attacked by microorganisms and other particles that can cause them disease, but they have a defense system, the immunology system, which deals with those pathogens and usually distinguishes between their own and foreign agents.

The human immune system defends against infecting and other foreign agents by distinguishing self from non-self (foreign antigens) and manage other protective responses from the principal immunologic cells, the leukocytes.

The immune system can be divided into two main systems: the innate immune system (we born with them) and, in vertebrates, an acquired or adaptive immune system, which grows and consolidates during first years of life and continues evolving during the rest of our lives; As a prove of their importance, only mentioning how easy and often are the infections from burns. For the goal of this researcher, the latter one will be the focus for this chapter; the acquired one.

The adaptive immune system brings a specific answer or action to each specific infectious agent, and it has a specific immune memory that avoids that same infectious agent could cause disease in a second infection. But, before the adaptive immune system acts, the human body has other natural protective barriers such as the skin and the mucous membranes.

The skin is a natural defense acting as impenetrable barrier for most of the microorganisms. It is usually known as the first layer of defense and part of the anatomical barriers to infections. Some pathogens can cross this barrier because they are inoculated by mites, mosquito, bedbugs, etc.

There are parts of the body that are not covered by skin, such as eyes, gastrointestinal tract, respiratory tract or urinary tract. In those cases the human body always has some additional defenses mechanism such as mucous membrane, some fluids and some cilium carpet that help on removing microorganisms.

The second layer of defense is the hostile physiology of the human body, such as the pH (very low in the stomach, slightly acid in skin and vagina), some antimicrobial substances (in tears, sweat, gastric acid, semen, etc.), and even our own native microbiota that avoids colonization in same space from foreign microorganisms (for example bacteria population in mouth, or in intestine).

The third layer of defense is the innate immune system (a.k.a. non-specific immune system), the defense system and all its methods that human body has. If the microorganism or the foreign particles are able to cross the previous layers of defense, then this system starts acting and includes different types of cells and soluble factors for destroying that foreign particle or microorganism, the pathogen. Innate immune system provides immediate defense against infection, but it does not provide long-lasting or protective immunity to the host as the adaptive immune system does. It is mainly and action-reaction system with only one target: destroying the pathogen.

The innate immune system includes two types of components: the humoral immunity and the cell-mediated immunity. Both work together for destroying the pathogen.

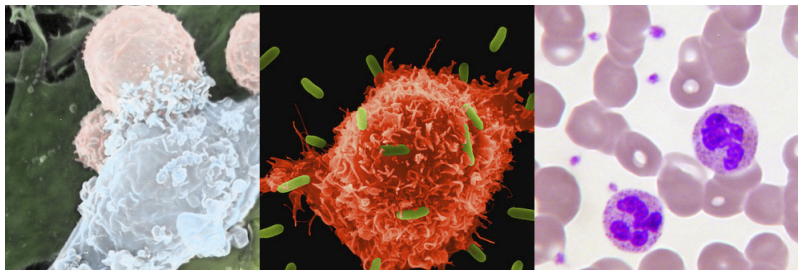


Figure 5.1: Phagocytic cells. From left ro rigth: 1st image a macrophage (blue) consuming leukemic cells (pink); 2nd image a peritoneal macrophage phagocytosis of E. coli Bacteria; and 3rd image, a neutrophil white blood cells or leukocytes.

The humoral immunity components is a complement system of proteins that creates a biochemical cascade to trigger the recruitment of inflammatory cells (inflammation serves to establish a physical barrier against the spread of infection), mark or tag pathogens for destruction by other cells, creating holes in the plasma membrane of the pathogen, etc. The cell-mediated immunity components are the white blood cells, also known as leukocytes. There are several types of leukocytes, each of them with a different function but all of them with same target of identifying and eliminating pathogens:

- Mast cells can be found in connective tissue and in the mucous membranes, when they are activated they release granules with histamine and heparin to help the inflammation process and recruit neutrophils and macrophages.
- Phagocytic cells are recruited for destroying the pathogen by phagocyte, what means they literally ingest (eat), kill and digest the pathogen. Figure 5.1 shows some of those cells. There are several types of phagocytic cells, mainly they can be,
  - Macrophages: the largest ones, they can move outside of the vascular system
  - Neutrophils: the most common and abundant ones. They are granulocytes, and their inside granules contain toxic substances that kill or inhibit growth pathogen
  - Dendritic cells: they are present in tissues and in contact with external environment, one of their roles are to be a bridge between innate and adaptive immune system
- Basophils and Eosinophils are both activated when a pathogen is detected and play a role to help neutrophils by releasing histamine that can defense against parasites or secrete a range of highly toxic proteins and free radicals for killing bacteria and parasites.
- Natural Killer cells (NK cells) destroy any compromised host cell, such as tumor cells or virus-infected cells, but they do not attack the pathogen itself. They act like a cleaning system.

Finally, the fourth layer of defense is the adaptive immune system, which is an evolution of the innate system. While the innate immune system can be found in all classes of plant and animal life, the adaptive immune system is only found in vertebrates, and is the only one that can offer a long-lasting and protective immunity to the host. After initial response from innate immune system to a specific pathogen, the body creates immunological memory that brings the acquired immunity; subsequent encounters with same pathogen will not affect and infect the user, pathogen-specific receptors will identify the already known pathogen.

Similarly as the innate immune system, the adaptive system includes both humoral immunity components and cell-mediated immunity components. The main cells of this system are the B and T cells, which are lymphocytes (a subset of leukocyte) morphologically indistinguishable from one another until after they are activated. B cells play mainly role in humoral immune response, while T cells in cell-mediated immune response.

The humoral response is the interaction between antibodies and antigens. Antibodies can appear in two types of physical forms: a soluble form that is secreted from the cell, and a membrane-bound form. B cells create the soluble antibodies that circulate in the blood plasma and lymph. The soluble antibodies (also known as immunoglobulin) are large Y-shaped proteins used by the immune system to identify and neutralize foreign objects. Antibody can be seen as a lymphocyte receptor. Antigens are also known as antibody generators, since they are any substance that provokes an adaptive immune response through the release of specific proteins called antibodies.

The cell-mediated immunity components of the adaptive immune system include the T cells. There are two types of T cells: the helper T cells, and the killer T cells (or cytotoxic T cells, TC). The first ones help to activate macrophages, killer T cells and B cells, while the the TC induce the death of cells that are infected with viruses or other pathogens.

When a dendritic cell (from innate immune system) phagocytes a pathogen, uses enzymes to chop the pathogen into smaller pieces, called antigens, and then display the non-self antigens on its surface, which the T cells can recognize and bound (as membrane-bound form of the antibody) when passing through the lymph node. In other words, T cells recognize their cognate antigen, while B cell encounters its cognate (or specific) antigen and receives additional signals from a helper T cell. T cells are mobilized either when they encounter a cell such as dendritic cell or B cell that has digested an antigen and is displaying antigen fragment bounds in its surface.

Once the antigen achieves the body, attracts and is bound to a respective and specific antibody. Each antibody binds to a specific antigen, an interaction similar to a lock and key, or as a puzzle analogy.

The part of the antigen that interact (matches) with antibody is called epitopes (see figure 5.2 for a generic example), or antigenic determinants. A very small proportion (less than 0.01%) of the total lymphocytes are able to bind to a particular antigen.

Innate and acquired portions of the immune system work together and not in spite

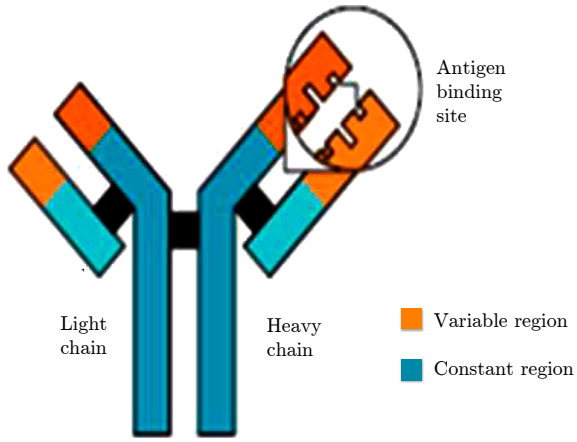


Figure 5.2: An antibody is made up of two heavy chains and two light chains. The unique variable region allows an antibody to recognize its matching antigen

of each other. B and T cells would be unable to function without the input of the innate system. T cells are useless without antigen-presenting cells to activate them, and B cells are crippled without T cell help. Figure 5.3 shows that interaction between both immune systems; a perfect orchestrated music piece for immunity

As we can see from all the previous explanation, the adaptive immune system is a complex system with such as important characteristics as *specificity* to different antigens, *diversity* (the lymphocyte collection is huge, at least 1 billion in human body), with immunological *memory* (the body keeps the memory of each antigen after its first contact), *self limitation* to decline the system activity to basic state after the pathogen has been eliminated, and *discrimination* between body's own cells and foreign invaders.

In this chapter, an artificial immunology approach will be presented for Music Recommendation. The layers of defense are understood as refinements of the system output to sequentially improve the quality of the recommendations. In the same way that human body uses four levels of defense for immunology: skin, hostile physiology, innate immunology, and adaptive immunology; we present four levels for music recommendation that includes (in the same order analogy): Sound Analysis, Riemann metric estimation, Millisound Selection, and Millisound Learning which are metaphorically equivalent to the human immune system. The figure 5.4 shows that analogy.

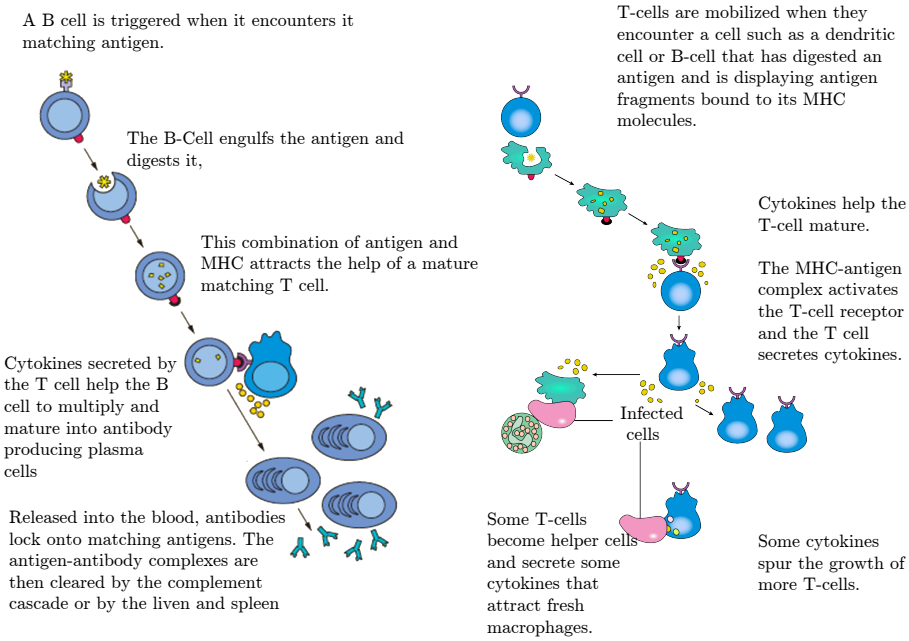


Figure 5.3: Left: The B lymphocyte activation pathway. B cells function to protect the host by producing antibodies that identify and neutralize foreign objects like bacteria and viruses. Right: The T lymphocyte activation pathway. T cells contribute to immune defenses in two major ways: some direct and regulate immune responses; others directly attack infected or cancerous cells

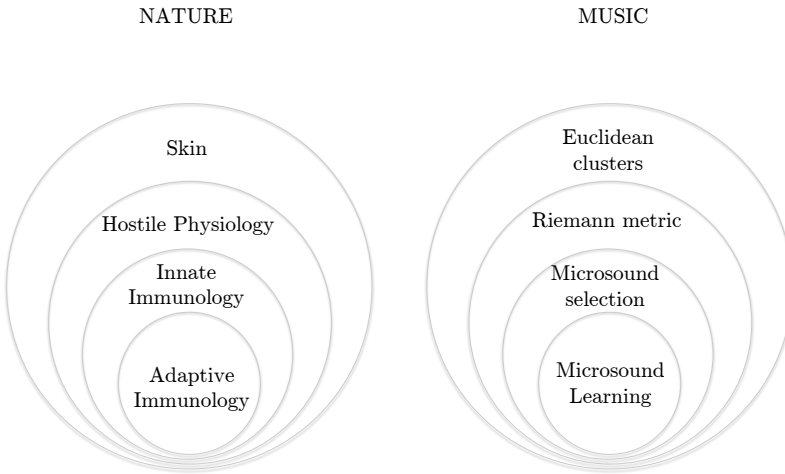


Figure 5.4: Nature vs Music

5.2 Millisounds

5.2.1 Music Information Reduction

Music audio signal contains a huge number of details that enhance the listening experience. Mainly the use of high frequency sampling rates (48kHz or even 96kHz for expert ears) allows extracting as much information as our ears are able to detect. Sometimes, the subtle transient effects at the highest frequency bands produce an overall effect that is only noticeable when listening to the entire track. Music can be thought of as a type of perceptual illusion, much the same way in which a collage is perceived. The brain imposes structure and order on a sequence of sounds that, in effect, creates an entirely new system of meaning. The appreciation of music is tied to the ability to process its underlying structure - the ability to predict what will occur next in the song. But this structure has to involve some level of the unexpected, or it becomes emotionally devoid.

5.2.2 Phasor motion representation

Pitched musical instruments are often based on an approximate harmonic oscillator such as a string or a column of air, which oscillates at numerous frequencies simulta-

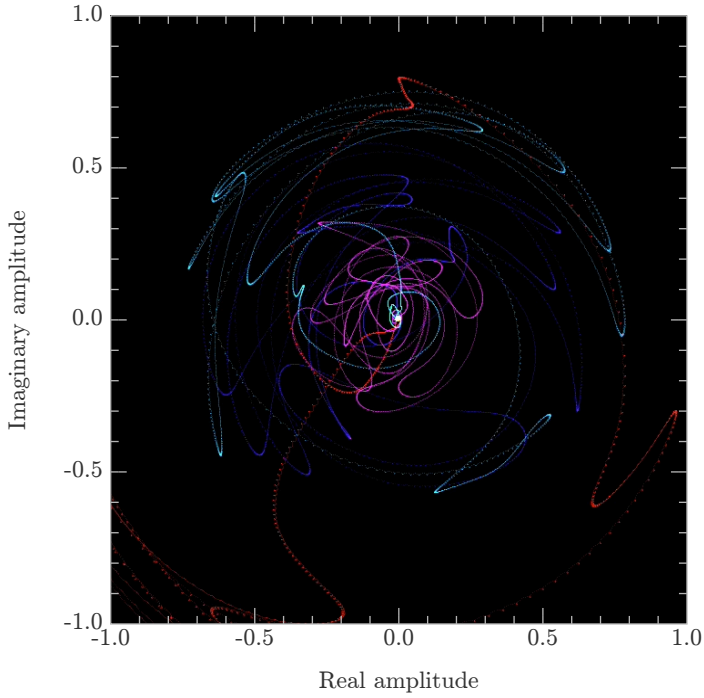


Figure 5.5: Violin Phasors. Red corresponds to the fundamental tone, while the other plots are the harmonics

neously. At these resonant frequencies, waves travel in both directions along the string or air column, reinforcing and canceling each other to form standing waves. Interaction with the surrounding air causes audible sound waves, which travel away from the instrument. Because of the typical spacing of the resonances, these frequencies are mostly limited to integer multiples, or harmonics, of the lowest frequency, and such multiples form the harmonic series.

The musical pitch of a note is usually perceived as the lowest partial present (the fundamental frequency), which may be the one created by vibration over the full length of the string or air column, or a higher harmonic chosen by the player. The musical timbre of a steady tone from such an instrument is determined by the relative strengths of each harmonic.

The relative amplitudes (strengths) of the various harmonics primarily determine the timbre of different instruments and sounds, though onset transients, formants, noises, and inharmonicities also play a role. For example, the clarinet and saxophone have similar mouthpieces and reeds, and both produce sound through resonance of air inside a chamber whose mouthpiece end is considered closed. Because the clarinet's resonator is cylindrical, the even-numbered harmonics are less present. The saxophone's resonator is conical, which allows the even-numbered harmonics to sound more strongly

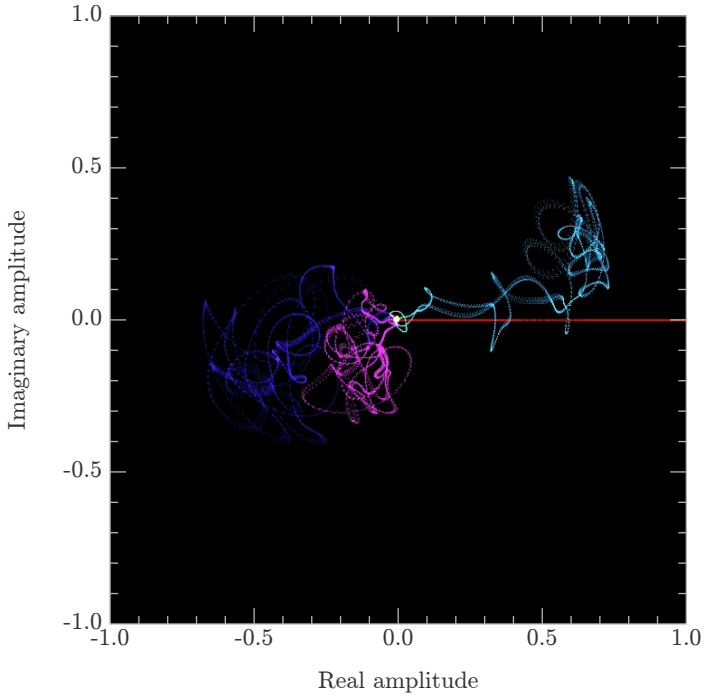


Figure 5.6: Violion phasor evolution with substracted fundamental tone phase

and thus produces a more complex tone. The inharmonic ringing of the instrument's metal resonator is even more prominent in the sounds of brass instruments.

Human ears tend to group phase-coherent, harmonically related frequency components into a single sensation. Rather than perceiving the individual partials, harmonic and inharmonic, of a musical tone, humans perceive them together as a tone color or timbre, and the overall pitch is heard as the fundamental of the harmonic series being experienced. If a sound is heard that is made up of even just a few simultaneous sine tones, and if the intervals among those tones form part of a harmonic series, the brain tends to group this input into a sensation of the pitch of the fundamental of that series, even if the fundamental is not present.

Fig. 5.5 shows the evolution of the fundamental when viewed as a continuous phasor. In physics and engineering, a phasor (a portmanteau of *phase vector*), is a complex number representing a sinusoidal function whose amplitude A, frequency θ, and phase ω are time-invariant. It is a special case of a more general concept called analytic representation. Phasors separate the dependencies on A, θ, and ω into three independent factors. This can be particularly useful because the frequency factor (which includes the time-dependence of the sinusoid) is often common to all the components of a linear combination of sinusoids. The trajectories are chaotic if they are represented without a reference. However, if we use the fundamental tone and we simulate the tendency

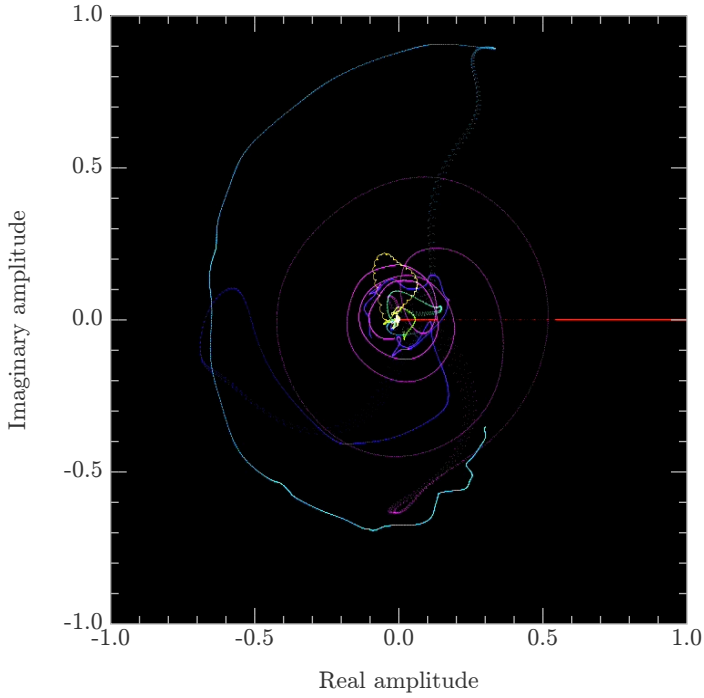


Figure 5.7: Piano phasor evolution with substracted fundamental tone phase

of human ears to group harmonically related frequency components, it is possible to perceive the variations of the different instruments much easily. Fig. 5.6 is the same violin signal as in Fig. 5.5 but the phase of the fundamental tone is subtracted to the harmonics. The timbre characteristics of the violin instrument are even possible to appreciate. Fig. 5.7 uses the same fundamental tone phase subtraction but for a piano instrument. The evolution of the referenced harmonics when viewed as a phasor is radically different even if it is the same 440Hz tone.

Variations in the frequency of harmonics can also affect the perceived fundamental pitch. These variations, most clearly documented in the piano and other stringed instruments but also apparent in brass instruments, are caused by a combination of metal stiffness and the interaction of the vibrating air or string with the resonating body of the instrument.

Music involves subtle violations of timing and, because we know through experience that music is not threatening, these violations are ultimately identified by the frontal lobes as a source of pleasure. The expectation builds anticipation, which, when met, results in the reward reaction. More than any other stimulus, music has the ability to conjure up images and feelings that need not necessarily be directly reflected in memory. The overall phenomenon still retains a certain level of mystery; the reasons behind the thrill of listening to music are strongly tied in with various theories based

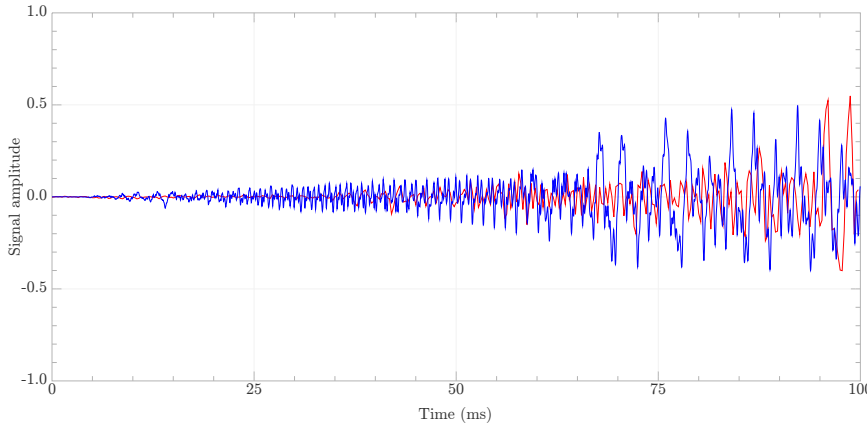


Figure 5.8: Time-amplitude Original signal (blue) versus quality degraded signal (red)

on synesthesia.

Music, though it appears to be similar to features of language, is more rooted in the primitive brain structures that are involved in motivation, reward and emotion. Whether it is the first familiar notes of *The Beatles - Yellow Submarine*, or the beats preceding *AC/DCś Back in Black* the brain synchronizes neural oscillators with the pulse of the music (through cerebellum activation), and starts to predict when the next strong beat will occur. The response to *groove* is mainly unconscious; it is processed first through the cerebellum and amygdala rather than the frontal lobes.

Even if the human brain is not able to trigger neuron spikes at a higher frequency than 1Khz [Javel80], it is able to separate signals by frequency and process them in parallel using delays. The author of this dissertation believe that brain users a harmonic phasor approach to process audio as it is the only way to keep the sampling frequency below 1kHz and be able to explode the phase information of higher frequency ranges. In signal processing, a matched filter is obtained by correlating a known signal, or template, with an unknown signal to detect the presence of the template in the unknown signal. For high frequency signal, a matched filter requires a high sampling rate. However, with the phasor approach and by only taking energy measurements every 1ms (1kHz sampling rate) but with thousands of receivers in parallel with different delays, it would be possible to modulate the incoming high frequency signal into a base band model that could be identified and processed by the human brain. Therefore, a battery of phasor-matched filters would represent a very efficient for human ears to distinguish subtle instruments differences while at the same time preserving the rapid response requirements that the ears as a defensive mechanism must provide.

The concept *millisound* can be introduced at this stage. It basically consists on a very short sound that the authors believe the brain could process independently, as a transient, with its attack time and its decay time. It is in the region of the 50ms and would represent similar evolutions that the ones represented in Figs. 5.5, 5.6 and 5.7.

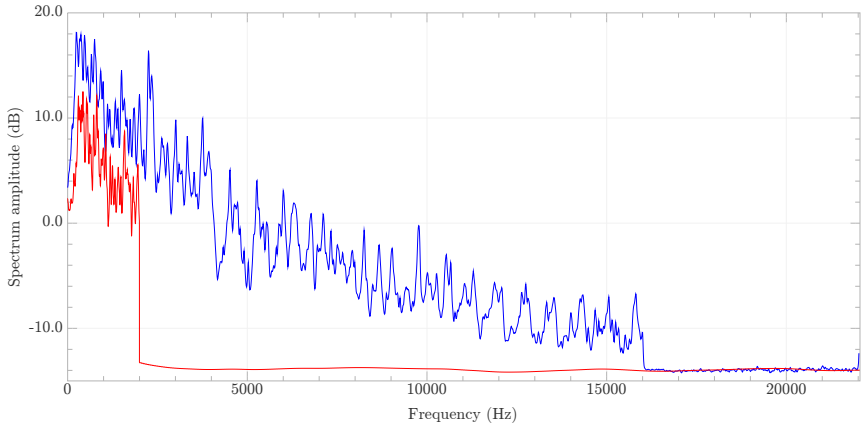


Figure 5.9: Frequency-spectrum power of original signal (blue) versus quality degraded signal (red)

If the brain was specialized in detecting very short millisounds, 50ms represent the typical duration of natural causal sounds in nature, such as the sound of a predator starting to move, which would require immediate detection and immediate reaction. The brain, as any signal processing machine, has to integrate the sound before been able to process it, and we are very used to reacting against very short and loud sounds. *Millisounds* can be viewed as the first layer of signal transformation in which the brain transforms the incoming high sampling rate audio signal into a set of audio and music information carrying objects.

5.2.3 Low encoding quality and low sampling rate signals

Stereophonic sound or, more commonly, stereo, is a method of sound reproduction that creates an illusion of directionality and audible perspective. This is usually achieved by using two or more independent audio channels through a configuration of two or more loudspeakers (or stereo headphones) in such a way as to create the impression of sound heard from various directions, as in natural hearing. Thus the term *stereophonic* applies to so-called *quadraphonic* and *surround-sound* systems as well as the more common two-channel, two-speaker systems. It is often contrasted with monophonic, or *mono* sound, where audio is in the form of one channel, often centered in the sound field (analogous to a visual field). Stereo sound is now common in entertainment systems such as broadcast radio and TV, recorded music and the cinema.

The joint use of Stereophonic sound, high sampling rates together with high bit depths produce the best possible music listening experience. However, for music recommendation there is too much information. Two similar tracks are similar even if recorded in Mono, low bit depth and low sampling rate. This is one of the key concepts of this chapter, as Applied Immunology occurs on tracks that have been downgraded to very low quality. The author has conducted some experiments to find out the limits

of this quality downgrade. A dozen of music lovers were asked to listen to 1000 track segments of increasingly duration and quality. The results of this experiment showed that users were able to estimate genre with a 95% accuracy for tracks with duration of 0.9 seconds, a sampling rate of $4000Hz$ and an encoding rate of $6kbp$.

The selected encoded was the Vorbis encoder as it is a free and open-source software project (see Appendix A.1 for more information). Vorbis produces high fidelity and it is completely free by nature, unencumbered by patents. It is a well-suited replacement for patented and restricted formats like MP3. Vorbis is intended for sample rates from 8 kHz telephony to 192 kHz digital masters and a range of channel representations (monaural, polyphonic, stereo, quadraphonic, 5.1, ambisonic, or up to 255 discrete channels). Given 44.1 kHz (standard CD audio sampling frequency) stereo input, the encoder will produce output from roughly 45 to 500 kbit/s depending on the specified quality setting. Vorbis aims to be more efficient than MP3, with data compression transparency being available at lower bitrates. Vorbis uses forward-adaptive monolithic transform codec based on the modified discrete cosine transform (MDCT). Vorbis uses the modified discrete cosine transform for converting sound data from the time domain to the frequency domain. The resulting frequency-domain data is broken into noise floor and residue components, and then quantized and entropy coded using a codebook-based vector quantization algorithm. The noise floor approach gives Vorbis its characteristic analog noise-like failure mode when the bitrate is too low to encode the audio without perceptible loss. The sound of compression artifacts at low bitrates can be perhaps described as reverberations in an amphitheater or a room.

To prepare tracks for music recommendation, the tracks are processed with the following sequence to obtain very low quality samples:

- Convert track to *MONO*
- Low-pass track with cut off frequency of 2kHz
- Downsample track to 4kHz sampling rate
- Normalize amplitude
- Encode with Vorbis faking 8kHz sampling rate, minimum quality

The decoded files can be used for track identification and also for music similarity but the listening quality is very poor. Fig. 5.8 shows the time domain amplitude information of an original 16kHz bandwidth signal compared to the signal that results after applying the information reduction process described in previous steps. Fig. 5.9 represents the same two signals but in the frequency domain. The total reduction is about a factor of 20. By doing this transformation, the author enforces that there is no superfluous information in the input data for the Applied Immunology music recommendation algorithm. Vorbis algorithm cannot go under 8kHz but it can be achieved by simply changing the original sampling rate label without downsampling the signal. This extra compression produces still audio tracks that can very efficiently be used for recommendation.

5.2.4 Innovation detector

One of the main requirements of a *millisound* is that it must be a causal event, therefore if should not be possible to predict from the last received audio samples. Causality is the relation between an event (the cause) and a second event (the effect), where the second event is understood as a physical consequence of the first. Informally, physicists use the terminology of cause and effect in the same everyday fashion as most other people do. Causal notions are important in general relativity to the extent that to have an arrow of time demands that the universe's semi-Riemannian manifold can change its main rotation, so that *future* and *past* are globally definable quantities. In engineering, a causal system is a system with output and internal states that depends only on the current and previous input values. A system that has some dependence on input values from the future (in addition to possible past or current input values) is termed an acausal system, and a system that depends solely on future input values is an anticausal system. Acausal filters, for example, can only exist as post processing filters, because these filters can extract future values from a memory buffer or a file. Music is about causality and this is easy to understand doing the following experiment: listening a track in reverse produces an extremely weird experience. Most of the signal properties are exactly same (mainly those related to amplitude energy and spectrum components) but the phase information are so hard-coded in human brain that the this reverse-track is generally considered *non natural*.

A *millisound* can be described as a short duration innovation in the incoming signal. A piano note sound is an innovation when it is first received but after it starts to decay it is not an innovation anymore. To build an innovation detector, the author has constructed a simple Linear Predictive Coding (LPC) filter on top of the 4kHz quality degraded signal.

LPC starts with the assumption that a speech signal is produced by a buzzer at the end of a tube (voiced sounds), with occasional added hissing and popping sounds (sibilants and plosive sounds). Although apparently crude, this model is actually a close approximation of the reality of speech production. The glottis (the space between the vocal folds) produces the buzz, which is characterized by its intensity (loudness) and frequency (pitch). The vocal tract (the throat and mouth) forms the tube, which is characterized by its resonances, which give rise to formants, or enhanced frequency bands in the sound produced. Hisses and pops are generated by the action of the tongue, lips and throat during sibilants and plosives.

LPC analyzes the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal after the subtraction of the filtered modeled signal is called the residue.

The numbers that describe the intensity and frequency of the buzz, the formants, and the residue signal, can be stored or transmitted somewhere else. LPC synthesizes the speech signal by reversing the process: use the buzz parameters and the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in speech.

Because speech signals vary with time, this process is done on short chunks of the speech signal, which are called frames; generally 30 to 50 frames per second give intelligible speech with good compression.

The most common representation of the prediction model is

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i) \tag{5.1}$$

where $\hat{x}(n)$ is the predicted signal value, $x(n-i)$ the previous observed values, and a_i the predictor coefficients. The error generated by this estimate is

$$e(n) = x(n) - \hat{x}(n) \tag{5.2}$$

where $x(n)$ is the true signal value. These equations are valid for all types of (one-dimensional) linear prediction. The differences are found in the way the parameters a_i are chosen.

The most common choice in optimization of parameters a_i is the root mean square criterion which is also called the autocorrelation criterion. In this method we minimize the expected value of the squared error $E[e2(n)]$, which yields the equation

$$\sum_{i=1}^{p} a_i R(j-i) = -R(j) \tag{5.3}$$

for $1 \leq j \leq p$, where R is the autocorrelation of signal x_n, defined as

$$R(i) = E\{x(n)\, x(n-i)\} \tag{5.4}$$

and E is the expected value. In the multi-dimensional case this corresponds to minimizing the L_2 norm. The above equations are called the normal equations or Yule-Walker equations. In matrix form the equations can be equivalently written as

$$Ra = -r \tag{5.5}$$

where the autocorrelation matrix R is a symmetric, $p \times p$ Toeplitz matrix with elements $r_{ij} = R(i-j), 0 \leq i, j < p$ the vector r is the autocorrelation vector $r_j = R(j), 0 < j \leq p$, and the vector a is the parameter vector. Another, more general, approach is to minimize the sum of squares of the errors defined in the form

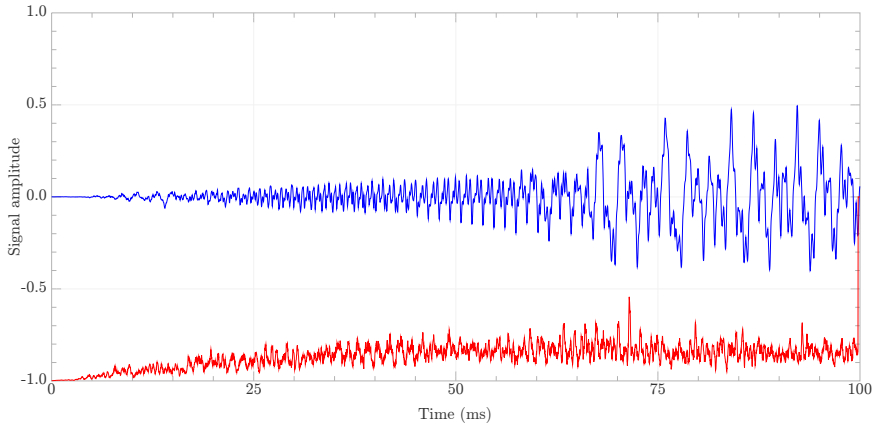


Figure 5.10: Innovation detection using Linear Prediction error. Blue signal represents the original signal and red signal represents the linear prediction error

$$(n) = x\,(n) - \widehat{x}\,(n) = x\,(n) - \sum_{i=1}^{p} a_i x\,(n-i) = -\sum_{i=0}^{p} a_i x\,(n-i) \tag{5.6}$$

where the optimization problem searching over all a_i must now be constrained with $a_0 = -1$. On the other hand, if the mean square prediction error is constrained to be unity and the prediction error equation is included on top of the normal equations, the augmented set of equations is obtained as

$$Ra = [1, 0, \ldots, 0]^T \tag{5.7}$$

where the index i ranges from 0 to p, and R is a $(p+1) \times (p+1)$ matrix. Specification of the parameters of the linear predictor is a wide topic and a large number of other approaches have been proposed. In fact, the autocorrelation method is the most common and it is used, for example, for speech coding in the GSM standard.

Solution of the matrix equation $Ra = r$ is computationally a relatively expensive process. The Gauss algorithm for matrix inversion is probably the oldest solution but this approach does not efficiently use the symmetry of R and r. A faster algorithm is the Levinson recursion proposed by Norman Levinson in 1947, which recursively calculates the solution. Another way of identifying model parameters is to iteratively calculate state estimates using Kalman filters and obtaining maximum likelihood estimates within Expectation-maximization algorithms.

Classical linear prediction specializes to the case where the data points y_β are equally spaced along a line, $y_i, i = 1, 2, ..., N$, and we want to use M consecutive values of y_i to predict an $M+1st$. Stationarity is assumed. That is, the autocorrelation $\langle y_j y_k \rangle$ is assumed to depend only on the difference $|j-k|$, and not on j or k individually, so

that the autocorrelation θ has only a single index,

$$\phi_j \equiv \langle y_j y_k \rangle \approx \frac{1}{N-j} \sum_{i=1}^{N-j} y_i y_{i+j} \tag{5.8}$$

Here, the approximate equality shows one way to use the actual data set values to estimate the autocorrelation components. In the situation described, the estimation equation is

$$y_n = \sum_{j=1}^{M} d_j y_{n-j} + x_n \tag{5.9}$$

This becomes the set of M equations for the M unknown d_j's, now called the linear prediction (LP) coefficients,

$$\sum_{j=1}^{M} \phi_{|j-k|} d_j = \phi_K \quad (k = 1, \ldots M) \tag{5.10}$$

There is an *magical* choice of M, the number of LP coefficients to use. M should be as small as required by the specific problem. There are also round off errors that require to *massage* the complex roots. Linear prediction is especially successful at extrapolating signals that are smooth and oscillatory, though not necessarily periodic. In such cases, linear prediction often extrapolates accurately through many cycles of the signal. By contrast, polynomial extrapolation in general becomes seriously inaccurate after at most a cycle or two. A prototypical example of a signal that can successfully be linearly predicted is the height of ocean tides, for which the fundamental 12-hour period is modulated in phase and amplitude over the course of the month and year, and for which local hydrodynamic effects may make even one cycle of the curve look rather different in shape from a sine wave.

Fig. 5.10 shows the innovation detector used with the Linear Predictive technique and an M factor of 8 using the 4kHz quality downgraded signal. The peaks in the linear prediction error are considered the triggers for the innovations and therefore only the 50ms frame segments that are preceded by a a local maximum of the LP error are considered as millisounds.

5.2.5 Extraction of millisounds

The process of extract the most significant millisounds given an input audio track consists on the following steps:

- Selection of the most representative 30s segment *MONO*

- Innovation detection
- Generation of the DCT transforms on innovation start time stamps
- MEL calculation
- MFCC calculation
- constructing the final millisound payload

The selection of millisounds is not applied over the entire signal but just in a specific 30s segment of it in order to reduce the computational burden related to the a-posterior processing requirements. To find the most representative 30s segment, the signal is analyzed in 30s segments with overlap of 5s. For each segment, a FFT transform is obtained. Then all segments are multiplied with the conjugate version of the other segments (matched filter in time domain) and the segment that has a higher average correlation is selected as the most representative of the signal.

At this point, the innovation detector algorithm is applied over the selected 30s segment to detect where the innovations starts. Then for each innovation the signal is back converted to time domain as Vorbis is storing the data internally as Discrete Cosine Transforms.

Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transform (DFT), a DCT operates on a function at a finite number of discrete data points. The obvious distinction between a DCT and a DFT is that the former uses only cosine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DCT implies different boundary conditions than the DFT or other related transforms.

Vorbis uses an overlapping transform, namely the MDCT, to blend one frame into the next, avoiding most inter-frame block boundary artifacts. The MDCT output of one frame is windowed according to MDCT requirements, overlapped 50% with the output of the previous frame and added. The window shape assures seamless reconstruction.

The modified discrete cosine transform (MDCT) is a lapped transform based on the type-IV discrete cosine transform (DCT-IV), with the additional property of being lapped: it is designed to be performed on consecutive blocks of a larger dataset, where subsequent blocks are overlapped so that the last half of one block coincides with the first half of the next block. This overlapping, in addition to the energy-compaction qualities of the DCT, makes the MDCT especially attractive for signal compression applications, since it helps to avoid artifacts stemming from the block boundaries.

Every millisound consists in a sequence of MDCT frames. These MDCT frames are inversely transformed to time domain to decode the low-quality original signal. The decoding process is described in the Appendix A.1. The millisounds are converted to frequency domain again to obtain the Mel-frequency spectrum. In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

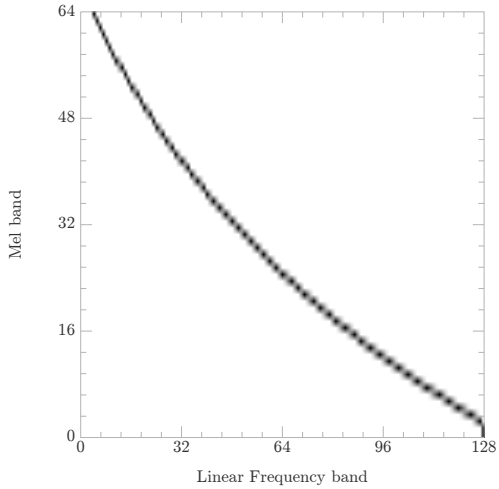


Figure 5.11: Mels

Mel-frequency Cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of Cepstral representation of the audio clip (a nonlinear *spectrum-of-a-spectrum*). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. Fig. 5.11 shows the matrix that can be used to transform from equally spaced frequency to MEL scale. This frequency warping can allow for better representation of sound, for example, in audio compression. The mel scale, named by Stevens, Volkmann, and Newman in 1937, is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by assigning a perceptual pitch of 1000 mels to a 1000 Hz tone, 40 dB above the listener's threshold. Above about 500 Hz, increasingly large intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale. The name mel comes from the word melody to indicate that the scale is based on pitch comparisons.

A popular formula to convert f hertz into m mel is

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{5.11}$$

MFCCs are commonly derived as follows [Xu04]:

1. Take the Fourier transform of (a windowed excerpt of) a signal.

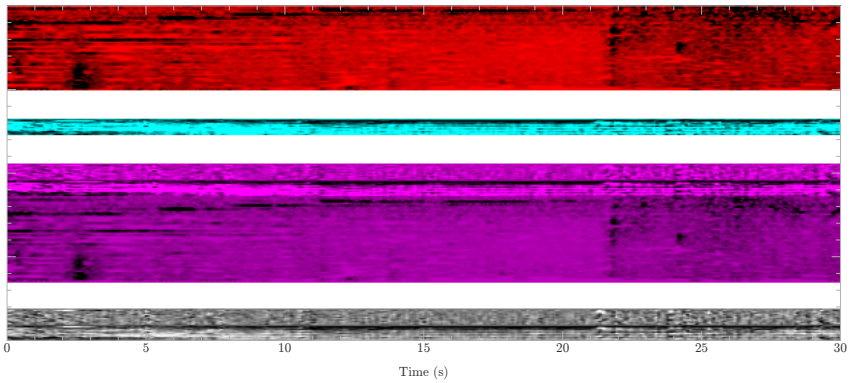


Figure 5.12: Millisound matrix

2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

MFCC values are not very robust in the presence of additive noise, and so it is common to normalize their values in speech recognition systems to lessen the influence of noise. Therefore the MEL original components are kept together with the MFCC components to form the final millisound frame. Fig. 5.12 shows an example of the matrix that is extracted for a given audio track. Every row of the matrix is formed by the MEL and the MFCC coefficients. Every row represents an innovation found in the original audio track. The millisounds are stored uncompressed to save time when they will be compared afterwards. Typical they occupy 10 times more than the Vorbis compressed sound data.

5.2.6 Vector quantization

Every time a millisound is extracted, it is compared to different codebooks of millisounds. For example, the authors have constructed a millisound codebook for all the music genres. The process to obtain one of this genre millisound codebook is based on analyzing thousands of tracks that have been listened by a person and tagged with the appropriate genre and then using vector quantization in each genre codebook.

Vector quantization (VQ) is a lossy data compression method based on the principle of block coding. It is a fixed-to-fixed length algorithm. In the earlier days, the design of a vector quantization (VQ) is considered to be a challenging problem due to the need for multi-dimensional integration. In 1980, Linde, Buzo, and Gray (LBG) proposed

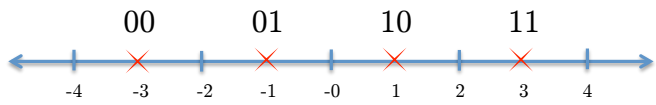


Figure 5.13: One dimensional VQ

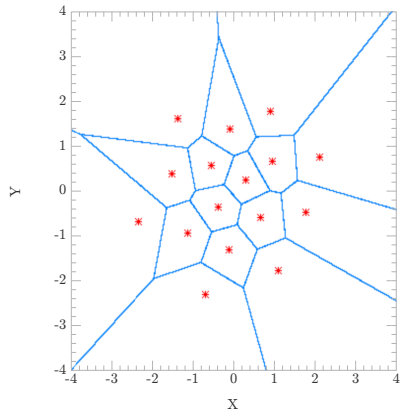


Figure 5.14: Two dimensional VQ

a VQ design algorithm based on a training sequence. The use of a training sequence bypasses the need for multi-dimensional integration. A VQ that is designed using this algorithm are referred to in the literature as an LBG-VQ.

A VQ is nothing more than an approximator. The idea is similar to that of *rounding-off* (say to the nearest integer). An example of a 1-dimensional VQ is shown in Fig. 5.13:

Here, every number less than -2 are approximated by -3. Every number between -2 and 0 are approximated by -1. Every number between 0 and 2 are approximated by $+1$. Every number greater than 2 are approximated by $+3$. Note that the approximate values are uniquely represented by 2 bits. This is a 1-dimensional, 2-bit VQ. It has a rate of 2 bits/dimension.

An example of a 2-dimensional VQ is shown in Fig. 5.14. Here, every pair of numbers falling in a particular region are approximated by a red star associated with that region. Note that there are 16 regions and 16 red stars – each of which can be uniquely represented by 4 bits. Thus, this is a 2-dimensional, 4-bit VQ. Its rate is also 2 bits/dimension.

In the two examples, the red stars are called codevectors and the regions defined by the blue borders are called encoding regions. The set of all codevectors is called the codebook and the set of all encoding regions is called the partition of the space.

The VQ design problem can be stated as follows. Given a vector source with its statistical properties known, given a distortion measure, and given the number of codevectors, find a codebook (the set of all red stars) and a partition (the set of blue lines) which result in the smallest average distortion.

We assume that there is a training sequence consisting of M source vectors:

$$\Gamma = \{x_1, x_2, \ldots, x_M\} \tag{5.12}$$

This training sequence can be obtained from some large database. For example, if the source is a speech signal, then the training sequence can be obtained by recording several long telephone conversations. M is assumed to be sufficiently large so that all the statistical properties of the source are captured by the training sequence. We assume that the source vectors are k-dimensional, i.e.,

$$x_m = (x_{m,1}, x_{m,2}, ..., x_{m,k}) \quad m = 1, 2, ..., M \tag{5.13}$$

Let N be the number of codevectors and let

$$C = \{c_1, c_{2,}, ..., c_N\} \tag{5.14}$$

represents the codebook. Each codevector is k-dimensional, i.e.,

$$c_n = \{c_{n,1}, c_{n,2,}, ..., c_{n,k}\} \quad n = 1, 2, ..., N \tag{5.15}$$

Let S_n be the encoding region associated with codevector c_n and let

$$\mathrm{P} = \{S_1, S_2, ..., S_N\} \tag{5.16}$$

denotes the partition of the space. If the source vector x_m is in the encoding region S_n, then its approximation (denoted by $Q(x_m)$) is c_n:

$$Q(x_m)C_n, \quad \text{if} x_m \in S_n \tag{5.17}$$

Assuming a squared-error distortion measure, the average distortion is given by:

$$D_{av} = \frac{1}{Mk} \sum_{m=1}^{M} \|x_m - Q\,(x_m)\|^2 \tag{5.18}$$

where $\|e\|^2 = e_1^2 + e_2^2 + ... + e_k^2$. The design problem can be succinctly stated as follows: Given Γ and N, find C and P such that D_{av} is minimized. If C and P are a

solution to the above minimization problem, then it must satisfied the following two criteria.

- Nearest Neighbor Condition: This condition says that the encoding region S_n should consists of all vectors that are closer to c_n than any of the other code-vectors. For those vectors lying on the boundary (blue lines), any tie-breaking procedure will do.

$$S_n = \left\{x : \|x - c_n\|^2 \leq \|x - c_{n'}\|^2 \forall n' = 1, 2, ..., N\right\} \tag{5.19}$$

- Centroid Condition This condition says that the codevector c_n should be average of all those training vectors that are in encoding region S_n. In implementation, one should ensure that at least one training vector belongs to each encoding region (so that the denominator in the above equation is never 0).

$$c_n = \frac{\sum_{x_m \in S_n} x_m}{\sum_{x_m \in S_n} 1} \quad n = 1, 2, ..., N \tag{5.20}$$

The LBG VQ design algorithm is an iterative algorithm which alternatively solves the above two optimality criteria. The algorithm requires an initial codebook $C^{(0)}$. This initial codebook is obtained by the *splitting* method. In this method, an initial codevector is set as the average of the entire training sequence. This codevector is then split into two. The iterative algorithm is run with these two vectors as the initial codebook. The final two codevectors are split into four and the process is repeated until the desired number of codevectors is obtained. The algorithm is summarized below.

1. Given Γ. Fixed $\varepsilon > 0$ to be a *small* number.
2. Let $N = 1$ and
$$c_1^* = \frac{1}{M} \sum_{m=1}^{M} x_m \tag{5.21}$$
Calculate
$$D_{av}^* = \frac{1}{Mk} \sum_{m=1}^{M} \|x_m - c_1^*\|^2 \tag{5.22}$$
3. Splitting: For $i = 1, 2, ..., N$, set
$$\begin{aligned} c_i^{(0)} &= (1 + \varepsilon)\, c_i^* \\ c_{N+i}^{(0)} &= (1 - \varepsilon)\, c_i^* \end{aligned} \tag{5.23}$$
Set $N = 2N$.
4. Iteration: Let $D_{av}^{(0)} = D_{av}^*$. Set the iteration index $i = 0$.
I. For $m = 1, 2, ..., M$, find the minimum value of
$$\left\| x_m - c_n^{(i)} \right\|^2 \tag{5.24}$$
over all $n = 1, 2, ..., N$. Let n^* be the index which achieves the minimum. Set

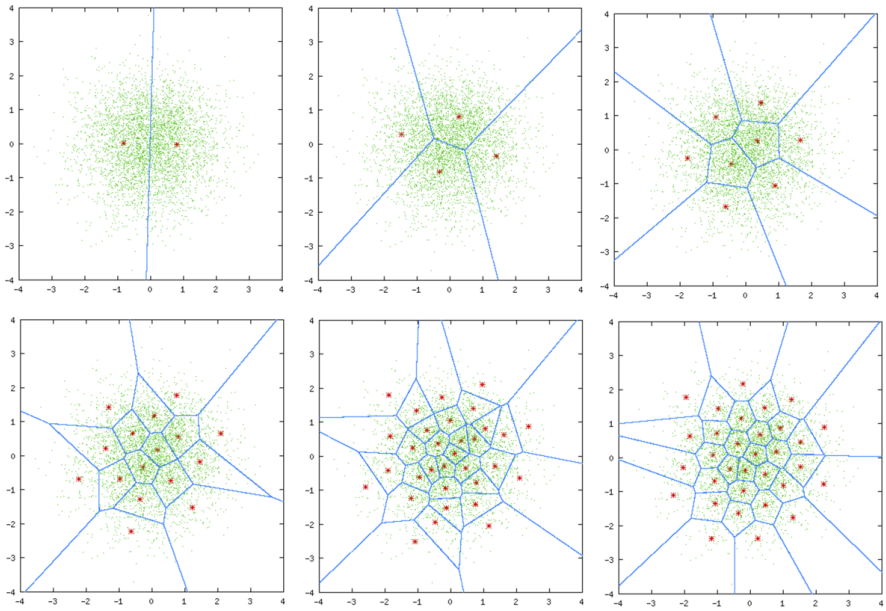


Figure 5.15: Evolution of the LBG algorithm

II. For $n = 1, 2, ..., N$, update the codevector

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_n^{(i)}} x_m}{\sum_{Q(x_m)=c_n^{(i)}} 1} \tag{5.25}$$

III. Set $i = i + 1$.

IV. Calculate

$$D_{av}^{(i)} = \frac{1}{Mk} \sum_{m=1}^{M} \|x_m - Q(x)\|^2 \tag{5.26}$$

V. If $(D_{av}^{(i-1)} - D_{av}^{(i)}/D_{av}^{(i-1)} > \varepsilon$, go back to Step (i). VI. Set $D_{av}^{*} = D_{av}^{(i)}$. For $n = 1, 2, .., N$, set $c_n^{*} = c_n^{(i)}$ sas the final codevectors.

5. Repeat Steps 3 and 4 until the desired number of codevectors is obtained.

Fig. 5.15 shows some iterations of the algorithm running on two dimensional Gaussian source with zero-mean and unit variance (about 4096 training points). This algorithm guarantees a locally optimal solution.

5.2.7 Negative selection and applied immunology for millisound matching

In artificial intelligence, artificial immune systems (AIS) are a class of computationally intelligent systems inspired by the principles and processes of the vertebrate immune system. The algorithms typically exploit the immune system's characteristics of learning and memory to solve a problem.

The field of Artificial Immune Systems (AIS) is concerned with abstracting the structure and function of the immune system to computational systems, and investigating the application of these systems towards solving computational problems from mathematics, engineering, and information technology. AIS is a sub-field of Biologically-inspired computing, and Natural computation, with interests in Machine Learning and belonging to the broader field of Artificial Intelligence.

AIS emerged in the mid 1980s with articles authored by Farmer, Packard and Perelson ([Farmer86]) on immune networks. However, it was only in the mid 1990s that AIS became a field in its own right. Forrest et al. (on negative selection) [Forrest94] published their first papers on AIS in 1994, and Dasgupta conducted extensive studies on Negative Selection Algorithms. Hunt and Cooke started the works on Immune Network models in 1995; Timmis and Neal continued this work and made some improvements. De Castro and Von Zuben's and Nicosia and Cutello's work (on clonal selection) became notable in 2002. The first book on Artificial Immune Systems was edited by Dasgupta in 1999. Currently, new ideas along AIS lines, such as danger theory and algorithms inspired by the innate immune system, are also being explored.

The common techniques are inspired by specific immunological theories that explain the function and behavior of the mammalian adaptive immune system.

- **Clonal Selection Algorithm**: A class of algorithms inspired by the clonal selection theory of acquired immunity that explains how B and T lymphocytes improve their response to antigens over time called affinity maturation. These algorithms focus on the Darwinian attributes of the theory where selection is inspired by the affinity of antigen-antibody interactions, reproduction is inspired by cell division, and variation is inspired by somatic hypermutation. Clonal selection algorithms are most commonly applied to optimization and pattern recognition domains, some of which resemble parallel hill climbing and the genetic algorithm without the recombination operator [Castro02b].
- **Immune Network Algorithms**: Algorithms inspired by the idiotypic network theory proposed by Niels Kaj Jerne that describes the regulation of the immune system by anti-idiotypic antibodies (antibodies that select for other antibodies). This class of algorithms focus on the network graph structures involved where antibodies (or antibody producing cells) represent the nodes and the training algorithm involves growing or pruning edges between the nodes based on affinity (similarity in the problems representation space). Immune network algorithms have been used in clustering, data visualization, control, and optimization domains, and share properties with artificial neural networks [Castro02a].
- **Dendritic Cell Algorithms**: The Dendritic Cell Algorithm (DCA) is an ex-

ample of an immune inspired algorithm developed using a multi-scale approach. This algorithm is based on an abstract model of dendritic cells (DCs). The DCA is abstracted and implemented through a process of examining and modeling various aspects of DC function, from the molecular networks present within the cell to the behavior exhibited by a population of cells as a whole. Within the DCA information is granulated at different layers, achieved through multi-scale processing.

- **Negative Selection Algorithm**: Inspired by the positive and negative selection processes that occur during the maturation of T cells in the thymus called T cell tolerance. Negative selection refers to the identification and deletion (apoptosis) of self-reacting cells, that is T cells that may select for and attack self tissues. This class of algorithms is typically used for classification and pattern recognition problem domains where the problem space is modeled in the complement of available knowledge. For example in the case of an anomaly detection domain the algorithm prepares a set of exemplar pattern detectors trained on normal (non-anomalous) patterns that model and detect unseen or anomalous patterns [Kephart94].

From an audio similarity point of view, the negative selection is used in the last two steps of the recommendation process with the objective to detect tracks that have been *wrongly* considered as *similar* by algorithms in previous layers. The scenario is as follows: the user is trying to find similar tracks given a similar tracks within a context of a specific genre (i.e. *jazz*); To help remove candidates that are not similar to the seed track, the algorithms has been able to pre-process millions of millisounds from all the genres to construct codebooks of millisounds; Given a millisound it can be estimated the probability of appearing in all genres. So for all the candidates in initial similar list, the tracks having the *least* amount of millisounds with high probability to appear in seed track genre (i.e. jazz in the case of the example) will be removed from the list and not presented to the user as similar tracks to the seed track.

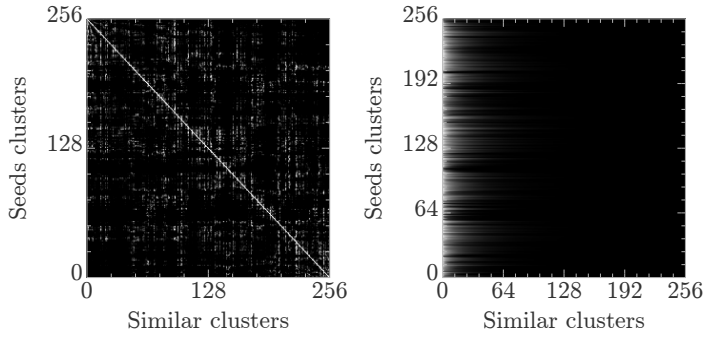


Figure 5.16: Probability of cluster to appear in recommendations of a track of another cluster: unordered (left), ordered by probability (right)

5.3 Applied Immunology

5.3.1 Introduction

When the user selects a seed track, the recommendation algorithm scans the entire collection of tracks and applies a screening mechanism consisting in different layers for all the tracks in the collection. If a track does not pass a layer, it is not considered a candidate to the next layer.

5.3.2 First layer: the Skin

The current implementation extracts 65 music descriptors from each analyzed song. Each descriptor is normalized. In the first layer of the recommendation, the screening process has to be extremely fast from a computation point of view while at the time it must reduce the number of correct similar tracks not being able to pass through the next layer. To perform this task, the 65-dimension space has been clusterized into 256 classes. The task of the first layer is to divide by 20 the tracks that pass to the next layer, so if the initial number is 100 million, we should expect only 5 million to pass to the second phase.

The main concept is the ability to discard track candidates in clusters with the only information of in which cluster the seed track is. To provide such possibility, the Euclidean algorithm has been executed with million of tracks without the clustering filtering and for each cluster, a probability of appearing in another cluster recommendation has been calculated. So basically:

1. Given a track, find its nearest centroid of the 256 classes and determine the cluster centroid c.

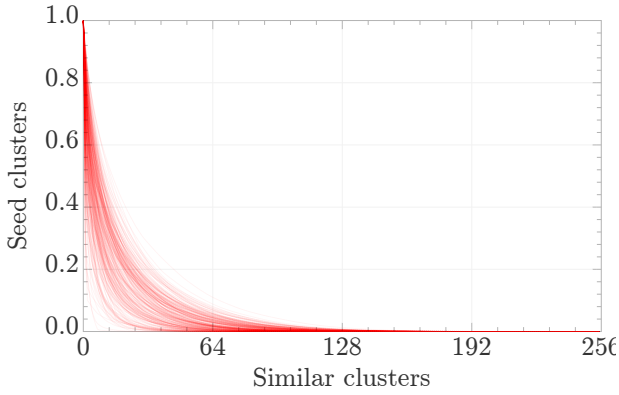


Figure 5.17: Percentile probability of a cluster to appear in another cluster seed recommendation

2. For all tracks in the universe, only pass to the next phase the ones in which the probability of appearing in c cluster recommendation is higher than a threshold.
3. This threshold can be used to select the trade-off between speed and complete scan. One extreme case would be to just look for tracks in the cluster of the seed track and the other extreme would be to look for similar tracks in all the clusters of the space.

Fig. 5.16 shows given a seed cluster (*row*) in which other clusters (*columns*) it is expected to find similar tracks (left) and how the probabilities are arranged when sorted by value for every seed cluster (right). It is possible to notice that the similar tracks are in one fifth of the space so about 80% of the clusters (and therefore the tracks) can be discarded during this first screening layer. The same can be observed when plotting all the cluster percentile probabilities in the same graph (Fig. 5.17). This first layer can help the system downgrade under control if the time to produce a real-time calculation must be reduced (for example, in a peak load moment, when many users are connecting to the system) as it can directly decrease computation time by reducing the number of clusters being allowed to pass to the next layer.

5.3.3 Second layer: the medium

At this stage, the aim is to reduce the current 5 million song list to a more usable 500 similar song list with an average accuracy of 50% for all the songs in the reduced list. The Euclidean metric is used to find the most similar tracks to the seed track and the closest 10 thousand tracks are pre-selected. At this stage, the Riemann curved estimation process is performed for the seed track and the 10k candidates are sorted again using the estimated metric. The 500 tracks more closed to the seed track using the curved metric are allowed to pass to the next layer. Even if we expect a high accuracy in the top part of these 500 tracks, it is expected to decrease quite fast so from this

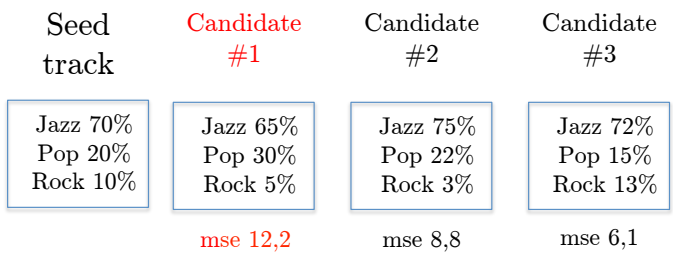


Figure 5.18: Discarding the track that has the least resemblance to the seed track from a Millisound point of view

moment, the task of the third and fourth layers is to remove bad recommendations to promote most similar tracks to the top of the list.

At this stage, it is also possible to filter using other parameters (such as genre, beats per minute, energy, etc.). Classifiers can be implemented in this stage to only let tracks from a specific class to pass to the next layer. For example, the curved metric can be used to classify calm vs. energetic tracks at this stage with an expected accuracy of 90%.

5.3.4 Immunology layers

5.3.4.1 Inborn immune system

The goal of this layer is to discard songs from the 500 similar songs sorted in the previous layer using the local metric loop calculation. First, the seed song is transcoded to a very low quality format that uses an audio perceptual model to discard not significant audio data. Then the seed song is divided into millisounds with an innovation detector. A millisound, as described before, is a segment between 50 and 100 ms that starts with a clear sound/music innovation. The innovation detector uses distances between near autocorrelations to detect changes in music pattern (linear prediction).

Every time a new track is added to the collection of tracks, it is analyzed to obtain the 65 descriptors and at the same time it is encoded with Vorbis and its channel are mixed to obtain a low-quality and low-sampling rate version of it. This simplified signal is processed and a set of millisounds is obtained for the most representative 30s of the original track. About 2500 millisounds vectors are extracted for every track. A millisound frame consists on the following combined and normalized vectors:

- Top 12 Mel-frequency Cepstral coefficients (MFCC)
- Top 12 Mel-frequency Cepstral coefficients with applied Cepstral mean extraction to normalize the channel
- 64 Mel-scale coefficients

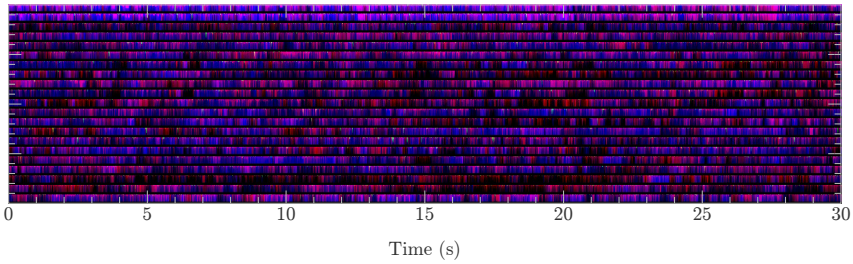


Figure 5.19: List of millisound stripes. The first row corresponds to the seed track and the other 20 are the candidates from the second layer. Red is for the first MFCC, blue is for the ceptral mean extracted MFCC and green is for the MEL coefficients

Almost all tracks in the collection have a *genre* tag associated to them. So when a track is incorporated to the collection, its millisounds are inserted into the millisound codebook of each genre using vector quantization as described before. Each millisound is also independently characterized and the probability of a specific millisound to belong to each genre is calculated. In this third stage, the genre probabilities of the millisound are compared against the genre probabilities of the millisounds of the candidates and the tracks that have millisounds with genre probabilities more different to the seed track are discarded. Fig. 5.18 depicts a simplified examples where the estimated genre probabilities based on Millisounds are used to negatively select the least similar tracks from the list of candidates, which is traduced in most similar tracks being pushed to top positions in the final short list that is presented to the user.

5.3.4.2 Acquired immune system

An epitope, also known as antigenic determinant, is the part of an antigen that is recognized by the immune system, specifically by antibodies, B cells, or T cells. For example, the epitope is the specific piece of the antigen that an antibody binds to. The part of an antibody that binds to the epitope is called a paratope. Although epitopes are usually non-self proteins, sequences derived from the host that can be recognized (as in the case of autoimmune diseases) are also epitopes. In the last layer of the recommendation process, the task is to go further and use an orthogonal method to reduce even more the probability of a wrong recommendation appearing in the final list of tracks presented to the user. The main idea behind is that the millisounds of the seed track will act as epitopes to the millisounds of the candidate tracks that pass screening of the third layer.

For each millisound of a candidate track, the ten most similar millisounds of the seed track are located and for each of the three sections of a millisound frame (12 MFCC, 12 ceptral mean extracted MFCC and 64 MEL) a Euclidean distance is calculated. Fig. 5.19 shows the adaptive millisound algorithm being applied over a list of 20 candidate tracks. More brighter region represent segments of the tracks where there are many millisounds of the original seed track producing a high match (very small Euclidean

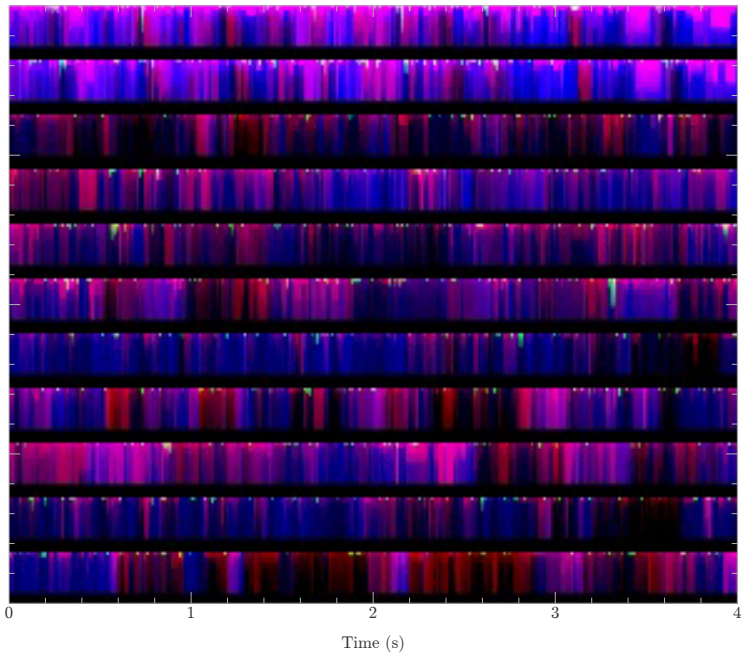


Figure 5.20: Zoom on the millisound stripes

distance).

A zoom of the same figure is available in Fig. 5.20 where more detail can be appreciated. The first stripe is the result of applying the seed epitopes to the seed itself. As it can be observed, the strip is not absolutely bright because instead of choosing the millisound that mostly matches, the algorithm is taking the 10 most similar millisound to provide better response against track variability. The second stripe represents a track that is a version of the original track while it is possible to see that the third track has very little millisound similarity and will most probability be discarded as a similar track.

5.4 Results

This novel Artificial Immunology approach that the author is presenting to help reduce the number of bad recommendations has some specific parameters that must be configured for each specific problem. For example, if the number of tracks in the original collection is very small, it would be recommended to use just the 2nd and 4th layer of this system. If the number of tracks is more than 100M then the first layer is crucial to help reduce the number of candidates that will be examined within the other three layers. The use of millisound to remove bad recommendations can be applied with

many other parameter configurations such as the number of MFCC coefficients. For example, using the MEL scale in the millisound definition does not produce as many benefits as the author originally thought for this specific application but depending on the type of music classifier that is required, their usefulness might increase. To benchmark the classification improvement accuracy produced by the third and fourth layer, the test dataset of chapter 4 has been used. As described before, this test consists on about 500 tracks grouped by 35 very different subgenre families (opera, piano classic, jazz, techno, French pop music). A good classification consists on being able to assign a track to its correct family by just comparing the track against all the tracks of the test set.

The current implementation of the millisound algorithm for negative-selection is using $12 + 12 + 64$ coefficients to describe a millisound. As shown in Fig. 5.21, if the Immunology negative selection is applied over the results obtained with the Riemann curved metric the classification achieved is of 60% over the top ten recommended tracks. Finally, Fig. 5.22 shows that the Immunology approach is providing a maximum 15% classification accuracy increase. If human supervised tags are used in combination with the raw music signal processing, it is possible to boost the classification to more than 90%, which is something that can be accepted by the average user.

However, trying to make a machine decipher the subtle details of music and being able to assert that two tracks are similar is an incredible journey that started just 15 years ago. There is something about music that makes it very personal. For example, we normally use the web to search for different things and we are used to see in the results, entries that do not match our text search. However, when discovering music, a bad recommendation produces a much worse impact than a bad text search result, and it is clear that there are two machines doing a process that was originally a human's only task. The author believes that we take it so personal because music is an open door to our brain.

5.5 Summary

To find similar tracks in a huge collection of music, a novel four layer process has been described. In each layer, a screening process is performed to filter our the list of tracks that go to the next layer. The first layer (which mimics the human skin) uses the original 65 descriptors that have been divided into clusters and only the tracks from similar clusters in a cluster-to-cluster relationship are accepted to pass to the next layer. In the second layer, a curved metric is estimated using the technique described in chapter 4. Finally, a new concept called millisound (an innovation that last about 50ms) is used to detect the tracks with the least similarities from this millisound point of view. This approach is used in the third and four layer to increase a 14% the classification accuracy of the overall music recommendation algorithm.

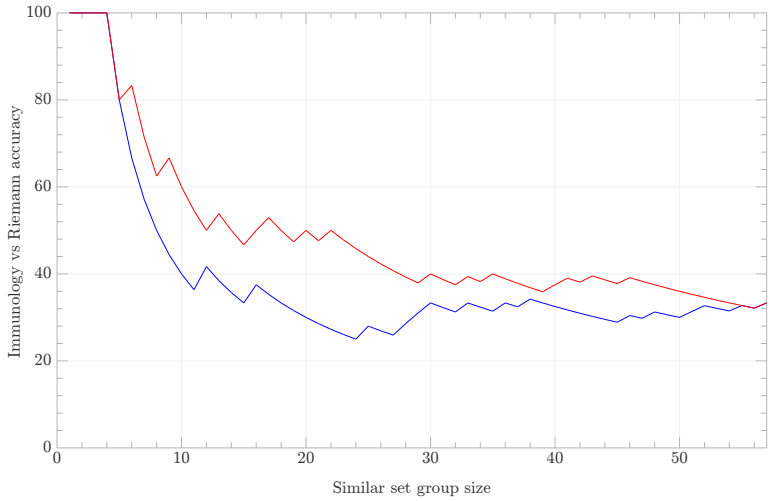


Figure 5.21: Final accuracy achieved with the combination of the Riemann curved metric and the Immunology negative-selection layers

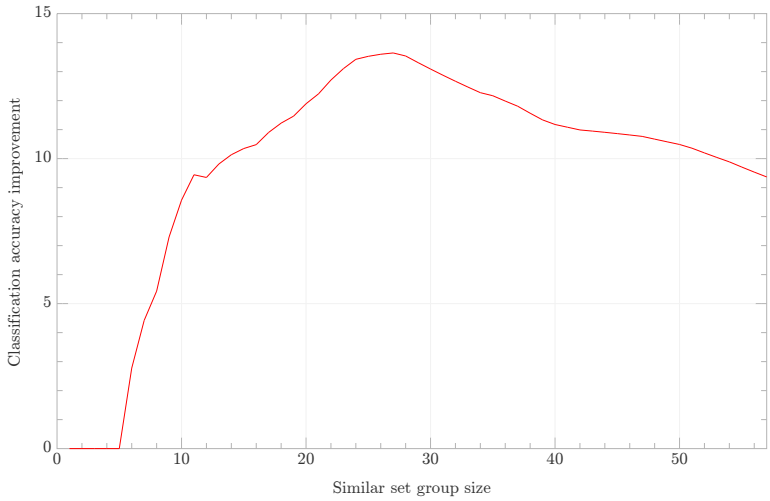


Figure 5.22: Classification accuracy improvement produced by the immunology layers

Chapter 6

Long Tail Distribution for Music Popularity

6.1 Introduction

The Music Industry is being continuously changing during last decades thanks to the appearance of technology and its different digital formats for audio content. CDs were a revolution on the Industry; less space, higher quality, and standing over the time without changes on the quality of the recording. Music Industry changed its main format very quick, in 1983 51.7% of US-based recording were vinyl format (singles, EP and LP), 47.8% were cassette, and only 0.5% were on CD format; 10 years later, in 1993, CD took up most of the market, with almost 65% of recordings using that format, and CD achieved its hit position on 2003 with 95.7% of the market (See table 6.1 for complete data)

However, digital format birth in combination with the personal computers, opened the door to CD burning for storing files into personal computers, where any digital content could be copied and transferred with exactly same quality as the original source. Internet and file sharing and peer-to-peer services like Napster made a huge impact on spreading the word for piracy. Anyone could share and send copies of audio files with the only effort of a mouse click; the piracy appeared as the big monster that was going to kill the industry. We cannot deny the piracy and its bad effects into Music Industry, but, nevertheless, the same reasons that caused the piracy, also caused a changed on music lovers activity, on the way they buy music and how they listen to music. Technology and Internet offered massive catalogs and choices to user for the type of music they could find, all niches exposed to the world. 14 years ago, around 2000, most of peer-to-peer networks offered more than 100 times more songs than any music store. That created the need of niches, and the need of purchase what users really wanted, so they started to demand songs à la carte, where users could buy tracks they

really like rather than the entire album, that made the album sales plummeting, but that also made the user the power to choose and show the real interest for specific songs.

It was a starting point of a new era in the Music Industry, a new era that is still under construction and where everyone is still trying to understand and learn the new rules, although some of them have already become clear, specially the ones about users' demand for music.

Smart technology, including Internet and e-commerce, changed the rules of the game. Physical shops were no longer required, the potential market for almost any type of item become global, and the users become more and more demanding. Two decades ago most of effort was focused on those cases/items that could bring the highest commercial success to the industry: the Best Seller book, the top music star with highest hits of music. The focus was on scoring the few mass-market hits. That behavior created what is known as the Long Tail effect in the industry: For a complete catalog of items, only a few percentage had the maximum number of sales (the head of the distribution, the Short Head in the demand curve), while the rest of content had a very low number of sales (the tail of the catalog, the Long Tail). Chris Anderson, in his book [Anderson06] *The Long Tail: Why the Future of Business is Selling Less of More*, made this demand curve popular, applying that to most of current markets, not only music (see figure 6.1 from his book, which shows the demand curve of popularity vs products).

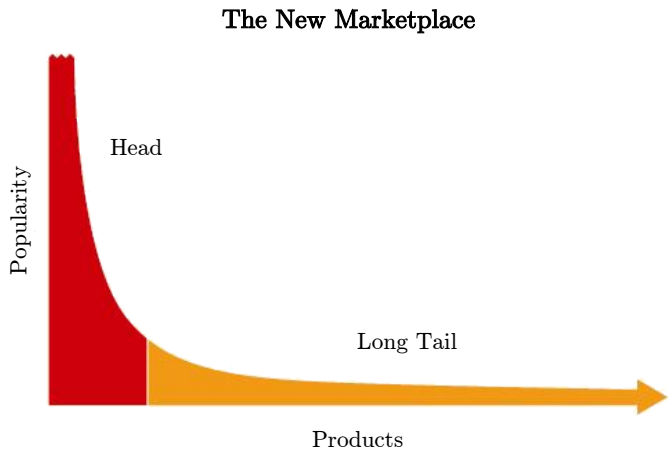


Figure 6.1: Popularity vs Products: The highest popular items are located at the left-hand side of the demand curve, in the *Head*; while the millions of various niche markets items are located on the right-hand side of the same curve, in the *Long Tail*.

Before the Internet era and the online stores, the physical space of warehouse was the main handicap that leads the offer of products. Unless the store had a huge amount

of space, everyone tried to get the most successful items into their catalogs rather than the less popular ones. That was the main reason why Major Labels were also focused on the big music stars, marketing their content as much as they could. But, what could happen if you don't have any limit on the space? If you count all purchases done for the Long Tail, it can bring a huge amount of benefits. More over, what would happen if you do not have to be focused on the mass market, but can achieve multiple niche markets? Any item of your catalog could find the right market and be sold. Those two important things are now possible thanks to Internet and the e-commerce.

Digital format and Internet changed the user behavior, tracks purchase became more popular, but also the complete catalog became exposed and reachable to whole world. Users are switching from album purchasing to song purchasing, but also switching from paying for purchasing to paying for listening to music. New music-based services are becoming more and more popular, either based on radio streaming or on-demand music services. (See table 6.1 for complete data). The Music Industry is moving toward a niche nation instead of the tyranny of the top.

Business, specially in the Music Industry, has to align with the Long Tail of the demand curve. The multiple niche marketing require the attention and can bring more benefit than the the few mass market hits. But that also has associated a change in the way of marketing things. There is no more space for one-size-fits all, now we have a mass market of small niche multitudes. Coming back to the Long Tail demand curve, and quoting his author Chris Anderson, that means that *will no longer rely into the* short head *of the demand curve, into the hits of music, where multiple copies of the same product are sold, but in the* long tail *of the same curve, where millions of various niche markets live*

One show can arrive to millions of people, but also millions of items can arrive to a single person. So we need tools for personalize the experience, to fit any taste, and to help the user to decide and find the right content. It is not only a quantitative change of the market, but a qualitative too.

As Chris Anderson said: *It will be better and more profitable to sell a few copies each into a million niche markets than it will be to try and develop one product which you sell two or three million copies.*

The smart technology also bring more content to catalogs, now an independent artist can make their own recording at home, without the need of expensive studio records. This is becoming a common way of making and marketing your own music. The DIY (Do It Yourself) is increasing the music catalogs by a huge number of items every month. For a common music aggregator company the music catalog available for a countries like US or UK can include more than 25 million songs, most of them from independent artists, only a small portion (3 to 5 million) from Major Label. This is the first time that the Music Industry is getting so many different actors and such a huge amount of content to be niche market and offered to the whole world, no matter how small and niche is that market. The cost of maintaining hits items from Short Head is going to be the same as the cost for maintaining items from which you sell only a few units.

However, the Future Music Forum performed last September 2014 in Barcelona, exposed the fact that music streaming services are changing everything, music industry and user behavior again, and the demand curve for streams per song is moving from the long tail demand curve to a new one where a third type of content is emerging. There is still a short head of hits where a few massive hits dominate the scene over short periods of time, but there are a few items from long tail that users tend to listen over and over again through long period of times, and the rest of content that are not strong enough or demand enough for long term appeal. Mark Mulligan, founder of Midia Research, calls those groups as Hit Machines, Evergreens and Middling Majority (see figure 6.2)

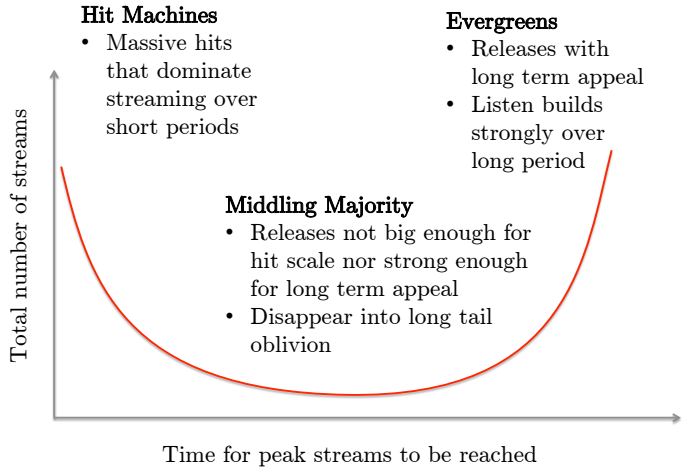


Figure 6.2: Streaming artist segments: the new demand curve.

The new market is promising for users, but it can be also confusing and hard to find your niche (no matter how big or small it is). MIR can help on this new scenario of the Music Industry, helping people to find what they want. Recommendations can be the key of the future, either we use the input from users, or his behavior over the service to learn which songs are similar to the liked ones from that user. User will need to be treated as individual, mass customization will be (is being) mandatory.

Quoting again Chris Anderson, *For many product categories, smart technology is transforming mass markets into millions of small niche markets. Although each of these niche markets may be small, when all the various niches are combined, the volume of business is actually greater than the traditional mass-market successes.*

No matter if the future demand curve will stay as Chris Andersonś format, or with Mark Mulliganś format, the short heads will be there, what means that for huge catalogs we will still see items from short head very well known and very popular for most of the public, while others items that will be totally for niche markets. Identifying and (or) predicting which of those items can be part of the short head can help the

business too. What if we could predict the Long Tail position of any item before it starts being known or sold? Furthermore, what if we can do that from the same moment that a new item arrives to the catalog? That is what the MELO algorithm tries to achieve.

MELO algorithm is a system and method for constructively providing a monetization procedure for a long tail demand curve of market goods, services or contents through a channel such as the Internet or mobile devices, for which there exists a source providing economic scoring (sales, downloads, streaming hours, etc.). Using only the scoring for a few reference items and a quantitative concept of similarity between the items, embodiments provide a procedure that constructively distributes the score from the reference items to the non-ranked ones, yielding the full scoring curve adjusted to a long tail law (power law). In order to build scores for non-ranked items, the method recursively defines relative preferences between items based on their similarity, thus constructing a utility-like function. The preferences are then used within an iterative tournament strategy between the items.

6.2 Model

The model is about some data from Rhapsody downloads per month at December 2005. One proposes here a simple phenomenological adjustment expression given by

$$D(r) = \frac{H}{\left(1 + \frac{r}{R}\right)^{1+E}} \tag{6.1}$$

where $D(r)$ is the number of measured downloads of ordered tracks at rank (position) r starting at $r = 1$. This is a three parameter model which admitt the following interpretations. $H = D(0)$ is the extrapolated download value at 0-rank. It controls the initial heigth of the curve. Notice that $T = \int_0^\infty dr D(r) = \frac{HR}{E}$ is the total number of downloads so one can also think of H as controlling the area below the curve once the ratio R/E is known.

For ranks far deep into the tail one has $\ln D(r) \sim_{r\to\infty} Const - (1 + E) \ln r$ and therefore $1+E$ is the slope of the $\ln D$ vs $\ln r$ in the assymptotic regime. The parameter E controls then the thickness of the tail. Recall that $T = \frac{HR}{E}$ so one must have $E > 0$. The closest E is to 0 the thicker the tail becomes. In fact, for $E = 0$ the area under the curve becomes infinite.

Finally R controls the transition between the *hit* zone $r \sim 0$ and the longtail one $r \sim \infty$. It is a scale parameters that says what is the meaning of the rank being small or big, since the model only depends on the ratio r/R. Its role can be understood from the next graph which has been produced by selecting arbitrarily $H = 1$ and $E = 0.5$ and then plotting the curve $D(r)$ for different values of the R parameter.

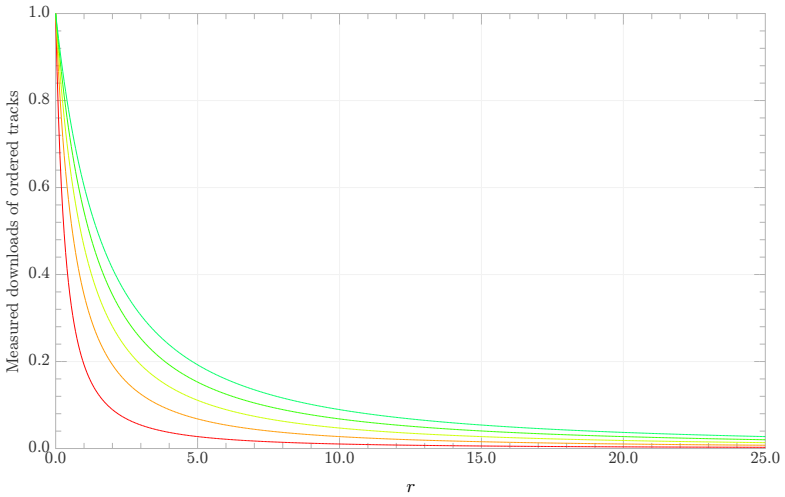


Figure 6.3: The graph above shows a plot of the model funtcion for $H = 1$ and $E = 0.5$ and several values of the scale parameter R.

6.2.1 Scale Free Analysis

In fact the download function D depends not on the absolute rank r but only on the quotient $x \equiv \frac{r}{R}$. Also the H parameter acts in a simple multiplicative way. It is then natural the *scale free* one parameter download function.

$$D_{sf}(x) \equiv (1+x)^{-E-1} \tag{6.2}$$

and recover $D(r)$ with the scale restoration substitution $r = Rx$ followed by dilatation with the factor H. In the graph, the scale free function corresponds to the $R = 1$ curve.

This scale free function only depends on the tail exponent E and so it is characterized by a single numerical quantity. For instance the total download number or area below the curve

$$A_{sf} \equiv \int_0^\infty ds D_{sf}(x) = \frac{1}{E} \tag{6.3}$$

This number defines the total size of the scale free business. This size goes all the way from 0 to ∞ and is controlled by the tail exponent. The lower the value of the exponent E the bigger the business size. It is natural then to define

$$T \equiv \frac{1}{E} \tag{6.4}$$

as a measure of the *thickness* of the size. The size of the business *niche* between relative ranks x_1, x_2 is given by

$$A_{sf}(x_1, x_2) \equiv \int_{x_1}^{x_2} dx D_{sf}(x) = \frac{1}{E}\left(\frac{1}{(1+x_1)^E} - \frac{1}{(1+x_2)^E}\right) \tag{6.5}$$

The *tail* size at x_1 is $A_{sf}(x_1, \infty) = \frac{1}{E}\left(\frac{1}{(1+x_1)^E}\right)$ while the size of the head at this rank is given by $A_{sf}(0, x_1) = \frac{1}{E}\left(1 - \frac{1}{(1+x_1)^E}\right)$.

An interesting relative rank point is the value X_{HET} where head equals tail, $\frac{1}{E}\left(1 - \frac{1}{(1+x_{HET})^E}\right) = \frac{1}{E}\frac{1}{(1+x_{HET})^E}$, which gives us

$$x_{HET} = 2^{\frac{1}{E}} - 1 = 2^T - 1 \tag{6.6}$$

This *head-tail-even* rank depends exponentially on the thickness of the tail. Thick tails have exponentially big x_{HET} while thin ones have small values. Remember that one needs to collect x_{HET} head ranks to equal the corresponding tail. Then a thick tail will require a longer head to balance its heavy contribution to the total size. For a thin tail the few first ranks already balance the full tail contribution.

The following picture represents scale free curves and corresponding HET points for various tail exponents or thicknesses.

6.2.2 Parameter Adjustment

In this section one assumes knowledge of a set of measurements

$$\{r_k, D_k\} : k = 1, 2, \ldots, N \tag{6.7}$$

where D_k is the measured download number at rank r_k. If one assumes that these data can be adjusted by a power law model

$$D(r) = \frac{H}{\left(1 + \frac{r}{R}\right)^{1+E}} \tag{6.8}$$

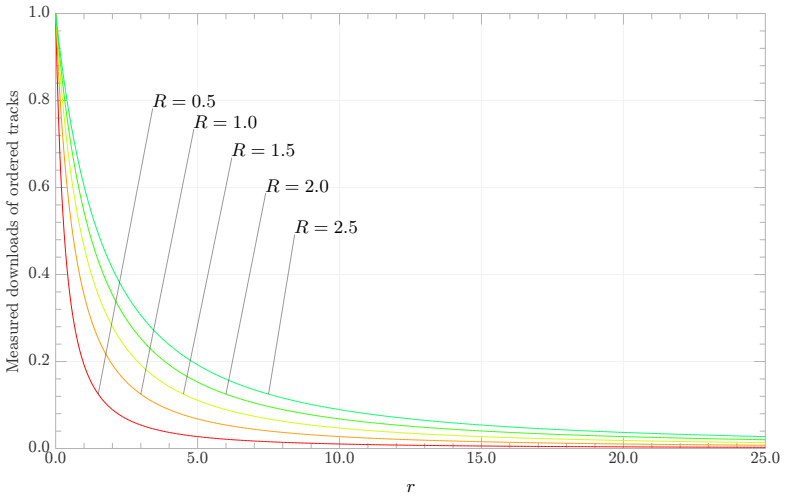


Figure 6.4: Scale free curves and corresponding HET points for various tail exponents or thicknesses.

the problem is how to estimate the value of the parameters H,R,E and provide a measure of the goodness of the fit. In the following subsections one addresses these problems.

6.2.3 RE-FIT

Let us introduce here the ratio of two different download measurements:

$$\rho_k^{k'} \equiv \frac{D_{k'}}{D_k} \tag{6.9}$$

According to the model it should correspond to

$$\frac{D\left(r_{k'}\right)}{D\left(r_k\right)} = \left(\frac{r_k + R}{r_{k'} + R}\right)^{1+E} \tag{6.10}$$

which do not depend on the H parameter. The logarithm of this equation is

$$\ln \frac{D\left(r_{k'}\right)}{D\left(r_k\right)} = 1 + E) \ln \left(\frac{r_k + R}{r_{k'} + R}\right) \tag{6.11}$$

so if we define

$$L_{k,k'} \equiv \ln \frac{D_k}{D_{k'}} \tag{6.12}$$

one shoud have

$$L_{k,k'} \equiv \ln \frac{D\left(r_k\right)}{D\left(r_{k'}\right)} = (1+E) \ln \left(\frac{r_{k'}+R}{r_k+R} \right) \tag{6.13}$$

$$F\left(R,E\right) \equiv \frac{1}{2N} \sum_{(k,k')\in\{sel\}} \left((1+E) \ln \frac{R+r_{k'}}{R+r_k} - L_{k,k'} \right)^2 \tag{6.14}$$

where $\{sel\}$ contains N selected pairs (k,k').

The extrema conditions are

$$\begin{array}{l} \frac{\partial F}{\partial E} = \frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \left((1+E) \ln \frac{R+r_{k'}}{R+r_k} - L_{k,k'} \right) \ln \frac{R+r_{k'}}{R+r_k} = 0 \\ \frac{\partial F}{\partial R} = \frac{1+E}{N} \sum\limits_{(k,k')\in\{sel\}} \left((1+E) \ln \frac{R+r_{k'}}{R+r_k} - L_{k,k'} \right) \frac{r_k - r_{k'}}{\left(R+r_{k'}\right)\left(R+r_k\right)} = 0 \end{array} \tag{6.15}$$

From these equations one has

$$1+E = \frac{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \ln \frac{R+r_{k'}}{R+r_k} L_{k,k'}}{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \left(\ln \frac{R+r_{k'}}{R+r_k} \right)^2} = \frac{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \frac{r_k - r_{k'}}{\left(R+r_{k'}\right)\left(R+r_k\right)} L_{k,k'}}{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \frac{r_k - r_{k'}}{\left(R+r_{k'}\right)\left(R+r_k\right)} \ln \frac{R+r_{k'}}{R+r_k}} \tag{6.16}$$

The second equality provides an equation for the parameter R alone which in principle determines R_{opt}. Once this value has been obtained one has

$$1+E_{opt} = \frac{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \ln \frac{R_{opt}+r_{k'}}{R_{opt}+r_k} L_{k,k'}}{\frac{1}{N} \sum\limits_{(k,k')\in\{sel\}} \left(\ln \frac{R_{opt}+r_{k'}}{R_{opt}+r_k} \right)^2} \tag{6.17}$$

6.2.4 HRE - Fit

Let us consider now a regression functional given by

$$F(H,R,E)=\frac{1}{2N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln\left(1+\frac{r_n}{R}\right)-\ln H\right]^2 \tag{6.18}$$

Computing the derivatives one has

$$\begin{aligned}
&\tfrac{\partial F}{\partial(\ln H)}=-\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln\left(1+\tfrac{r_n}{R}\right)-\ln H\right]\\
&\tfrac{\partial F}{\partial E}=\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln\left(1+\tfrac{r_n}{R}\right)-\ln H\right]\ln\left(1+\tfrac{r_n}{R}\right)\\
&\tfrac{\partial F}{\partial R}=-\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln\left(1+\tfrac{r_n}{R}\right)-\ln H\right]\tfrac{E+1}{R}\tfrac{\frac{r_n}{R}}{1+\frac{r_n}{R}}
\end{aligned} \tag{6.19}$$

so the extrem conditions become

$$\begin{aligned}
&\ln H-(E+1)\left\langle\ln\left(1+\tfrac{r}{R}\right)\right\rangle=\langle\ln D\rangle\\
&\left\langle\ln\left(1+\tfrac{r}{R}\right)\right\rangle\ln H-(E+1)\left\langle\ln^2\left(1+\tfrac{r}{R}\right)\right\rangle=\left\langle\ln D\ln\left(1+\tfrac{r}{R}\right)\right\rangle\\
&\left\langle\tfrac{1}{1+\frac{r}{R}}\right\rangle\ln H-(E+1)\left\langle\tfrac{\ln\left(1+\frac{r}{R}\right)}{1+\frac{r}{R}}\right\rangle=\left\langle\tfrac{\ln D}{1+\frac{r}{R}}\right\rangle
\end{aligned} \tag{6.20}$$

where in the last equation one has written $\frac{\frac{r_n}{R}}{1+\frac{r_n}{R}}=1-\frac{1}{1+\frac{r_n}{R}}$ and dropped the first term which equals the first equation and must be zero. From the first two equations one has

$$\ln H=\frac{\left\langle\ln^2\left(1+\frac{r}{R}\right)\right\rangle\langle\ln D\rangle-\left\langle\ln\left(1+\frac{r}{R}\right)\right\rangle\left\langle\ln D\ln\left(1+\frac{r}{R}\right)\right\rangle}{\left\langle\ln^2\left(1+\frac{r}{R}\right)\right\rangle-\left\langle\ln\left(1+\frac{r}{R}\right)\right\rangle^2}:E+1 \tag{6.21}$$

which determine H, E if R is known. The two first equations determine the functions $H(R)$, $E(R)$ which are depicted in the next figure.

Then the last equation is a relation involving R alone.

$$\begin{aligned}
&\ln H=\tfrac{1}{\left\langle\ln^2\left(1+\frac{r}{R}\right)\right\rangle-\left\langle\ln\left(1+\frac{r}{R}\right)\right\rangle^2}:E+1\\
&\left\langle\tfrac{1}{1+\frac{r}{R}}\right\rangle\left(\left\langle\ln^2\left(1+\tfrac{r}{R}\right)\right\rangle\langle\ln D\rangle-\left\langle\ln\left(1+\tfrac{r}{R}\right)\right\rangle\left\langle\ln D\ln\left(1+\tfrac{r}{R}\right)\right\rangle\right)-\\
&\left\langle\tfrac{\ln\left(1+\frac{r}{R}\right)}{1+\frac{r}{R}}\right\rangle\left(\left\langle\ln\left(1+\tfrac{r}{R}\right)\right\rangle\langle\ln D\rangle-\left\langle\ln\left(1+\tfrac{r}{R}\right)\ln D\right\rangle\right)=\\
&\left\langle\tfrac{\ln D}{1+\frac{r}{R}}\right\rangle\left(\left\langle\ln^2\left(1+\tfrac{r}{R}\right)\right\rangle-\left\langle\ln\left(1+\tfrac{r}{R}\right)\right\rangle^2\right)
\end{aligned} \tag{6.22}$$

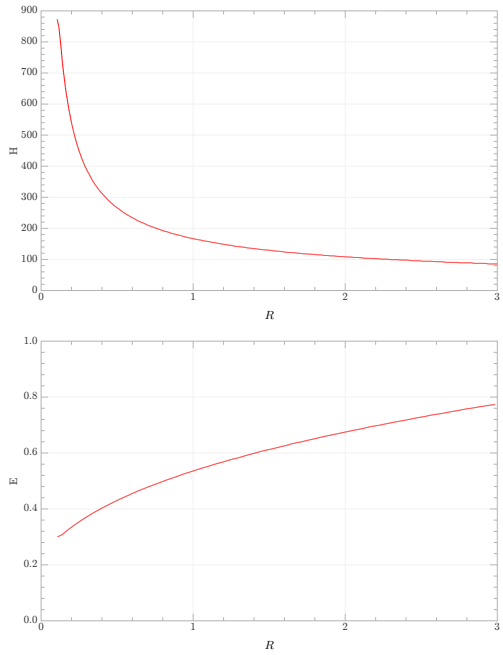


Figure 6.5: $H(R)$, $E(R)$ examples from test data.

The next figure shows that this function is never zero so the least squares problem does not have an extreme value of the parameter R.

The only information contained in the figure is then that the error is a monotonically increasing function of R without any privileged value for it.

6.2.5 Behaviour near $R = 0$

To study this behavior it is convenient to replace all terms involving $\ln\left(1 + \frac{r}{R}\right)$ by $\ln(r + R) - \ln R$ and isolate the contributions coming from the logarithm. This produces no changes in the expression for E.

$$E + 1 = \frac{\langle \ln(r + R) \rangle \langle \ln D \rangle - \langle \ln D \ln(r + R) \rangle}{\langle \ln^2(r + R) \rangle - \langle \ln(r + R) \rangle^2} \tag{6.23}$$

and a $\ln R$ contribution for $\ln H$:

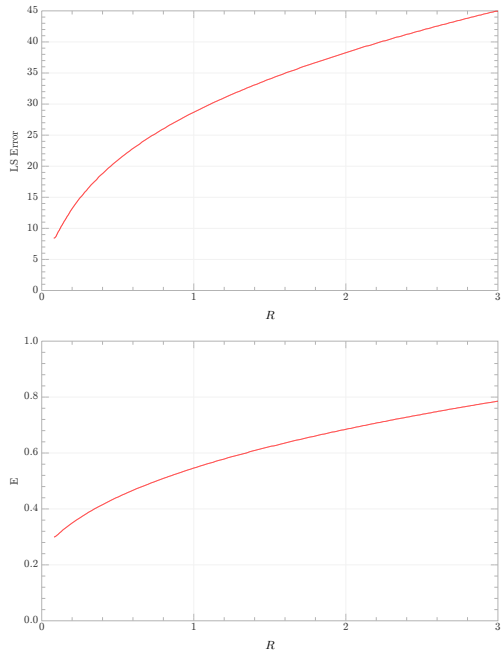


Figure 6.6: Extremal value for parameter R.

$$\begin{aligned}\ln H &= \frac{\left\langle \ln^2(r+R)\right\rangle \langle \ln D\rangle - \langle \ln(r+R)\rangle \langle \ln(r+R)\ln D\rangle}{\left\langle \ln^2(r+R)\right\rangle - \langle \ln(r+R)\rangle^2} + \frac{\langle \ln(r+R)\ln D\rangle - \langle \ln(r+R)\rangle \langle \ln D\rangle}{\left\langle \ln^2(r+R)\right\rangle - \langle \ln(r+R)\rangle^2} \ln R \\ &= \frac{\left\langle \ln^2(r+R)\right\rangle \langle \ln D\rangle - \langle \ln(r+R)\rangle \langle \ln(r+R)\ln D\rangle}{\left\langle \ln^2(r+R)\right\rangle - \langle \ln(r+R)\rangle^2} - (E+1)\ln R\end{aligned} \tag{6.24}$$

Then the behavior at $R = 0$ is easily identified as

$$E_0 + 1 = \frac{\langle \ln r\rangle \langle \ln D\rangle - \langle \ln r\rangle \langle \ln D\rangle}{\left\langle \ln^2 r\right\rangle - \langle \ln r\rangle^2} : R \sim 0 \tag{6.25}$$

and

$$\begin{aligned}\ln H(R) &= \frac{\left\langle \ln^2 r\right\rangle \langle \ln D\rangle - \langle \ln r\rangle \langle \ln r \ln D\rangle}{\left\langle \ln^2 r\right\rangle - \langle \ln r\rangle^2} - (E_0+1)\ln R \\ H(R) &= \frac{K}{R^{E_0+1}} \quad R \sim 0 \quad : K \equiv \frac{\left\langle \ln^2 r\right\rangle \langle \ln D\rangle - \langle \ln r\rangle \langle \ln r \ln D\rangle}{\left\langle \ln^2 r\right\rangle - \langle \ln r\rangle^2}\end{aligned} \tag{6.26}$$

6.2.6 Behaviour at R=∞

It is straightforward to find

$$\begin{array}{l} E+1\frac{\langle r\rangle\langle \ln D\rangle-\langle r\ln D\rangle}{\langle r^2\rangle-\langle r\rangle^2}R:R\sim\infty \\ \ln H=\frac{\langle r\rangle^2\langle \ln D\rangle-\langle r\ln D\rangle}{\langle r^2\rangle-\langle r\rangle^2}R:R\sim\infty \end{array} \tag{6.27}$$

It follows then that

$$\begin{array}{l} E\left(R\right)=\frac{\langle r\rangle\langle \ln D\rangle-\langle r\ln D\rangle}{\langle r^2\rangle-\langle r\rangle^2}R:R\sim\infty \\ H_\infty=\exp\left(\frac{\langle r\rangle^2\langle \ln D\rangle-\langle r\rangle\langle r\ln D\rangle}{\langle r^2\rangle-\langle r\rangle^2}\right) \end{array} \tag{6.28}$$

so $\frac{HR}{E}\rightarrow_{R\rightarrow\infty}Const$, the area under the curve is constant in this limit. as it is shown in the following picture

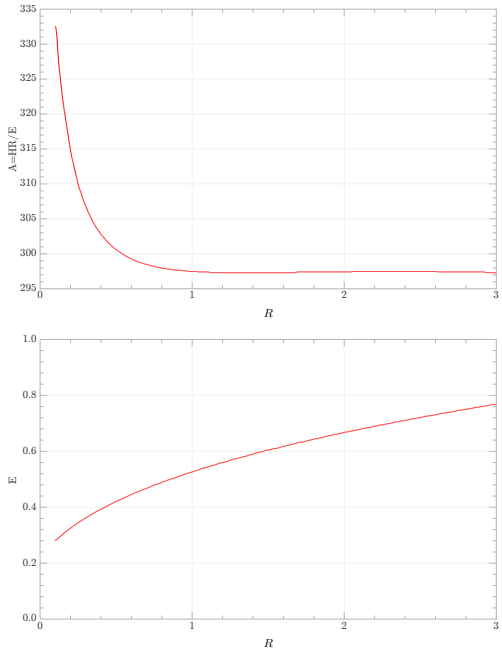


Figure 6.7: Area under the curve

It is remarkable the insensitivity of the area with respect to R from values beyond $R\cong 1$.

$$\begin{array}{l} R\left\langle \frac{1}{r}\right\rangle \ln H-(E+1)\, R\left\langle \frac{\ln r}{r}\right\rangle +(E+1)\, R\ln R=R\left\langle \frac{\ln D}{r}\right\rangle \\ \ln H-(E+1)\,\frac{1}{R}\,\langle r\rangle =\langle \ln D\rangle \end{array} \tag{6.29}$$

6.2.7 Tail Exponent Assymptotic Fit

From the logarithmic form of the model

$$\ln D\left(r\right)=\ln H-(E+1)\ln\left(1+\frac{r}{R}\right) \tag{6.30}$$

it is clear that if one considers only values with

$$\frac{r}{R}\gg 1 \tag{6.31}$$

one may use the approximation

$$\ln D(r)\simeq \ln\left(HR^{E+1}\right)-(E+1)\ln r \tag{6.32}$$

Let us now use only a subset of the measurements that verify

$$r>R_{asy} \tag{6.33}$$

where R_{asy} is a variable limit to obtain the partial data selection. In that case, if one uses linear regression of the pairs $\ln R, \ln D\left(r\right)$ on this subset, one should get an estimate of $E+1$ as the value of the slope and of HR^{E+1} from the exponential of the intercept. One can trust these estimates for values of R which verify

$$R\ll R_{asy} \tag{6.34}$$

The next figure shows the dependence of the estimated values for E arising from the linear regression with data filtered by the size of R_{asy} which is taken as the abscissa variable.

As discussed above the three parameter regression problem has no minimum on the R parameter. The only rule is the lowest, the better. Since the relationship $E-R$ is also monotone over the regression extremes, the same could be said for the E parameter. In fact, the graph shows a minimum value for $E=0.424$ corresponding to $R=0.48$ for a selection value $R_{asy}=25.05$. It is then a possible choice for the best parameter selection. The associated value for the third parameter is $H=264$.

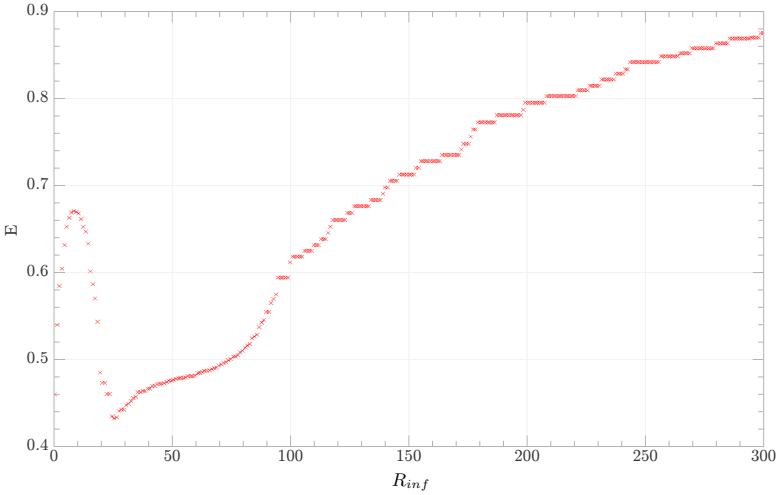


Figure 6.8: E vs Rinf

6.3 Mixture of models

In this section one introduces a model with a new parameter M which can be interpreted as an interpolator between two models of the type discussed before. Explicitly, let us introduce a Mixture Model (MM) as

$$D\left(r\right)=\frac{H\left(1+M\frac{r}{R}\right)}{\left(1+\frac{r}{R}\right)^{E+2}} \tag{6.35}$$

Let us notice that this can be written as

$$D\left(r\right)=\frac{HM\left(1+\frac{r}{R}\right)+\left(1-M\right)H}{\left(1+\frac{r}{R}\right)^{E+2}}=M\frac{H}{\left(1+\frac{r}{R}\right)^{E+1}}+\left(1-M\right)\frac{H}{\left(1+\frac{r}{R}\right)^{E+2}} \tag{6.36}$$

which explains the meaning of the name. For big r/R the behavior is dominated with a tail exponent of value $E+1$, but for low values both models compete to finally build the value H at $r=0$ with weights M and $1-M$ respectively.

This is the simplest generalization of the former described model with polynomial modifications to generate intermediate behavior:

$$D(r) = H \frac{M_0 + M_1\left(1+\frac{r}{R}\right) + \cdots + M_k\left(1+\frac{r}{R}\right)^k}{\left(1+\frac{r}{R}\right)^{E+k+1}} : M_0 + M_1 + \cdots + M_k = 1 \quad (6.37)$$

which again exhibits the asymptotic of an $E+1$-th order tail but with additional structure in the head and middle ranges. The next figure shows a mixture of models with fixed $H = 1$, $R = 1$, $E = 0.5$ and several values of the mixing parameter.

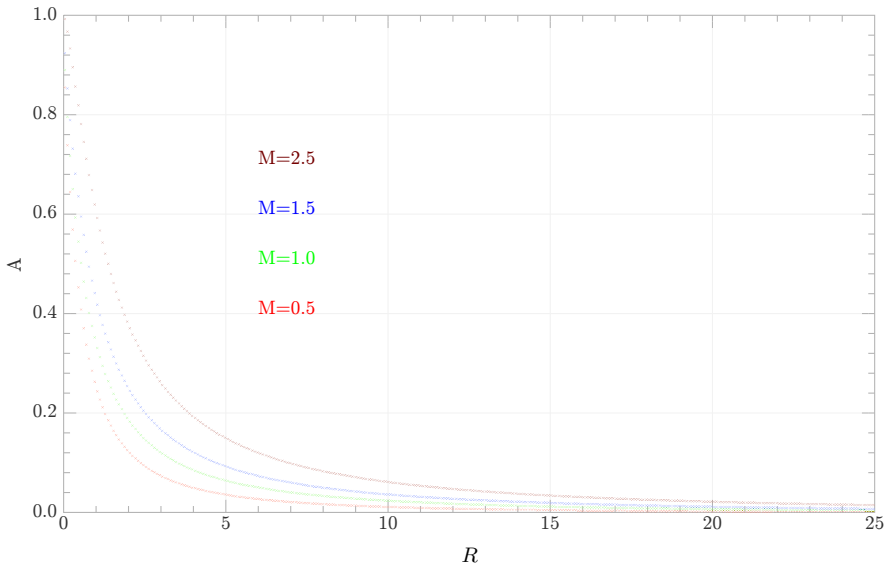


Figure 6.9: Mixture of models

6.3.1 Sice

The total number of downloads or area below the curve is given by

$$\mathrm{A} = \frac{\mathrm{MHR}}{E} + \frac{(1-M)\,HR}{E+1} = \frac{E+M}{E+1}\frac{HR}{E} \quad (6.38)$$

and this size, relative to the unmixed model, is bigger/lower if M is bigger/ lower that one.

In scale free terms with $H = 1$ and $X \equiv \frac{r}{R}$ the total size is simply

$$A_{sf} = \frac{M}{E} + \frac{1-M}{E+1} = \frac{E+M}{E+1}\frac{1}{E} \quad (6.39)$$

so the thickness of the tail is now the mixture of the two thicknesses. The asymptotic behavior is dominated by the exponent E but the thickness of the tail depends on the mixture. Therefore this is a mechanism to obtain a fatter ($M > 1$) or thinner ($M < 1$) tail of a model compatible with a given asymptotic.

In the same scale free terms, the tail size is given by

$$A_{sf}(x_1, \infty) = \frac{1}{E} \frac{1 + M x_1}{E(1 + x_1)^{E+1}} \tag{6.40}$$

so the head is:

$$A_{sf}(0, x_1) = \frac{E + M}{E(E + 1)} - \frac{1}{E} \frac{1 + M x_1}{(1 + x_1)^{E+1}} \tag{6.41}$$

and the x_{HET} abscissa is given by

$$\frac{E + M}{2(E + 1)} = \frac{1 + M x_{HET}}{(1 + x_{HET})^{E+1}} \tag{6.42}$$

which can be solved from the root of

$$f(x_{HET}) \equiv \frac{E + M}{E + 1}(1 + x_{HET})^{E+1} - 2M x_{HET} - 2 = 0 \tag{6.43}$$

which reduces to $2^{\frac{1}{E}} - 1$ when $M \to 1$.

6.3.2 Reciprocal Mixtures

An alternative approach would be the Mixed Model

$$D(r) = \frac{H}{\left(1 + M \frac{r}{R}\right)\left(1 + \frac{r}{R}\right)^E} \tag{6.44}$$

which also has a tail with exponent $1 + E$ and verifies $D(0) = H$ but has an additional parameter for extra structure. Here one has

$$\frac{1}{D(r)} = \frac{\left(1 - M + M + M \frac{r}{R}\right)\left(1 + \frac{r}{R}\right)^E}{H} = M \frac{\left(1 + \frac{r}{R}\right)^{E+1}}{H} + (1 - M) \frac{\left(1 + \frac{r}{R}\right)^E}{H} = \frac{M}{D_{E+1}(r)} + \frac{1 - M}{D_E(r)} \tag{6.45}$$

so it is natural to call it model which is reciprocal to the mixture of the reciprocal models associated with exponents $E+1$ and E. The following figure shows RMMs for fixed $H=1$, $R=1$, $E=0.5$ and several values of the mixture parameter. Notice that the behavior for low(high) M in RMMs is matched by high(low) values in the MMs. Also the decay in RMMs is faster.

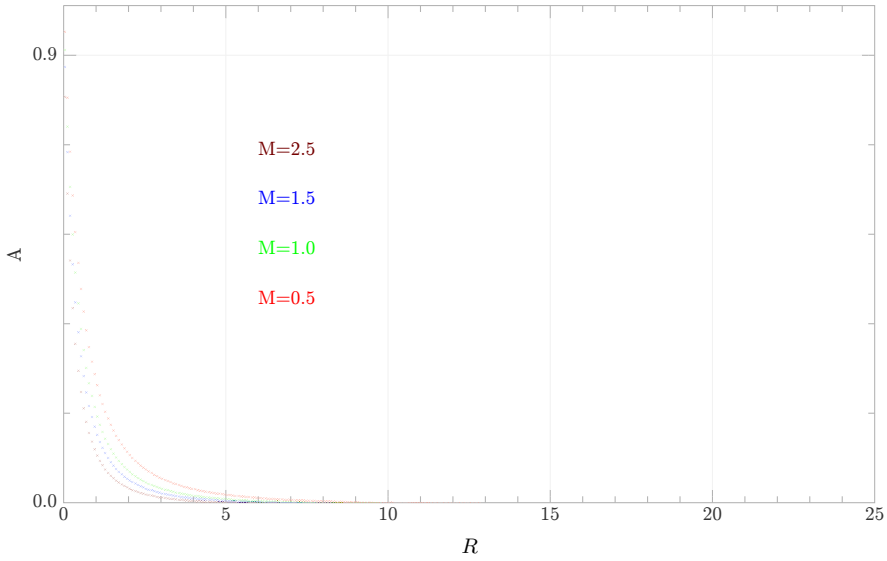


Figure 6.10: Different RMM for fixed parameters $H=1$, $R=1$, $E=0.5$

The scale free version is now

$$D_{sf}(x) = \frac{1}{(1+Mx)(1+x)^E} \tag{6.46}$$

and its area

$$A_{sf} = \int_0^\infty \frac{dx}{(1+Mx)(1+x)^E} = \int_0^\infty \frac{dx}{(1-M+Mx)x^E} = \frac{1}{M}\int_0^\infty dy \frac{y^{E-1}}{1+\frac{1-M}{M}y} \tag{6.47}$$

By using the definition of the hypergeometric function

$${}_2F_1(a,b;c;z) \equiv \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int_0^1 dx \frac{x^{b-1}(1-x)^{c-b-1}}{(1-xz)^a} \tag{6.48}$$

it is clear that

$$A_{sf} = \frac{1}{M}\int_0^1 dy \frac{y^{E-1}}{1-\frac{M-1}{M}y} = \frac{1}{ME}\,{}_2F_1\left(1, E; E+1; \frac{M-1}{M}\right) = \frac{1}{E}\,{}_2F_1\left(1, 1; E+1; 1-M\right)$$
$$= \frac{1}{E}\left(1 + \sum_{n=1}^{\infty} \frac{n!}{(E+1)\cdots(E+n)}(1-M)^n\right) \tag{6.49}$$

For $M \approx 1$ one has the approximation

$$A_{sf} = \frac{1}{M}\int_0^1 dy \frac{y^{E-1}}{1+\frac{1-M}{M}y} \approx \frac{1}{M}\int_0^1 dy \left(y^{E-1} - \frac{1-M}{M}y^E\right) = \frac{1}{E} + \frac{1-M}{E\,(E+1)} + O\left((1-M)^2\right) \tag{6.50}$$

so the tail goes thicker in this case if $M < 1$.

More generally if one introduces $m \equiv \frac{1}{M} - 1 : M = \frac{1}{m+1}$, one has

$$A_{sf} = (1+m)\int_0^1 dy \frac{y^{E-1}}{1+my} = (1+m)\sum_{k=0}^{\infty}(-m)^k \frac{1}{E+k} = \frac{1}{E} - \sum_{k=0}^{\infty} \frac{(-m)^{k+1}}{(E+k)\,(E+k+1)} \tag{6.51}$$

which can be directly obtained from the first representation in terms of the hypergeometric function.

6.3.3 Three parameter Regression

In this section one works on a generalization of the regression proposed in previous sections to find the parameters of a MM. For convenience one will take here a MM of the form

$$D\,(r) = \frac{H\left(1 + M\frac{r}{R}\right)}{\left(1+\frac{r}{R}\right)^{E+2}} = \frac{H\left(1 + (M-1)\frac{\frac{r}{R}}{1+\frac{r}{R}}\right)}{\left(1+\frac{r}{R}\right)^{E+1}} \tag{6.52}$$

Let us then introduce the functional

$$F(H,R,E,M)=\frac{1}{2N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln(1+x_n)-\ln\left(1+(M-1)\,\frac{x_e}{1+x_n}\right)-\ln H\right]^2 \tag{6.53}$$

where one has used $x\equiv r/R$ and ignored the role of R as a variable since one intends to determine H,E,M for R fixed.

One has for the derivatives

$$\begin{aligned}
\tfrac{\partial F}{\partial(\ln H)}&=-\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln(1+x_n)-\ln\left(1+(M-1)\,\tfrac{x_n}{1+x_n}\right)-\ln H\right]\\
\tfrac{\partial F}{\partial E}&=-\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln(1+x_n)-\ln\left(1+(M-1)\,\tfrac{x_n}{1+x_n}\right)-\ln H\right]\ln(1+x_n)\\
\tfrac{\partial F}{\partial M}&=-\tfrac{1}{N}\sum_{n=1}^{N}\left[\ln D_n+(E+1)\ln(1+x_n)-\ln\left(1+(M-1)\,\tfrac{x_n}{1+x_n}\right)-\ln H\right]\tfrac{x_n}{1+Mx_n}
\end{aligned} \tag{6.54}$$

so the minimum conditions are

$$\begin{aligned}
&\ln H-\langle\ln(1+x)\rangle(E+1)=\langle\ln D\rangle-\left\langle\ln\left(1+(M-1)\,\tfrac{x}{1+x}\right)\right\rangle\\
&\langle\ln(1+x)\rangle\ln H-\left\langle\ln^2(1+x)\right\rangle(E+1)=\langle\ln(1+x)\ln D\rangle-\left\langle\ln(1+x)\ln\left(1+(M-1)\,\tfrac{x}{x+1}\right)\right\rangle\\
&\left\langle\tfrac{x}{1+Mx}\right\rangle\ln H-\left\langle\tfrac{x\ln(1+x)}{1+Mx}\right\rangle(E+1)=\left\langle\tfrac{x\ln D}{1+Mx}\right\rangle-\left\langle\frac{x\ln\left(1+(M-1)\,\frac{x}{1+x}\right)}{1+Mx}\right\rangle
\end{aligned} \tag{6.55}$$

The two first equations can be used to express $\ln H$, $E+1$ in terms of M:

$$\begin{aligned}
\ln H&=\frac{\left\langle\ln^2(1+x)\right\rangle\langle\ln D\rangle-\langle\ln(1+x)\rangle\langle\ln(1+x)\ln D\rangle}{\left\langle\ln^2(1+x)\right\rangle-\langle\ln(1+x)\rangle^2}-\\
&-\frac{\left\langle\ln^2(1+x)\right\rangle\left\langle\ln\left(1+(M-1)\,\frac{x}{1+x}\right)\right\rangle-\langle\ln(1+x)\rangle\left\langle\ln(1+x)\ln\left(1+(M-1)\,\frac{x}{1+x}\right)\right\rangle}{\left\langle\ln^2(1+x)\right\rangle-\langle\ln(1+x)\rangle^2}\\
E+1&=\frac{\langle\ln(1+x)\rangle\langle\ln D\rangle-\langle\ln(1+x)\ln D\rangle}{\left\langle\ln^2(1+x)\right\rangle-\langle\ln(1+x)\rangle^2}-\\
&-\frac{\langle\ln(1+x)\rangle\left\langle\ln\left(1+(M-1)\,\frac{x}{1+x}\right)\right\rangle-\left\langle\ln(1+x)\ln\left(1+(M-1)\,\frac{x}{1+x}\right)\right\rangle}{\left\langle\ln^2(1+x)\right\rangle-\langle\ln(1+x)\rangle^2}
\end{aligned} \tag{6.56}$$

and then the last equation determines the value of the M parameter.

6.3.4 Approximate Regression

Let us solve the above equations to the first order in $M-1$. The obtained solution can be used as a starting point for some iterative procedure to obtain the exact values

of the nonlinear problem. The first two equations give

$$\begin{array}{l}\ln H \approx \frac{\left\langle \ln^2(1+x)\right\rangle \langle \ln D\rangle - \langle \ln(1+x)\rangle \langle \ln(1+x)\ln D\rangle}{\left\langle \ln^2(1+x)\right\rangle - \langle \ln(1+x)\rangle^2} - \\ -\frac{\left\langle \ln^2(1+x)\right\rangle \left\langle \frac{x}{1+x}\right\rangle - \langle \ln(1+x)\rangle \left\langle \frac{x\ln(1+x)}{1+x}\right\rangle}{\left\langle \ln^2(1+x)\right\rangle - \langle \ln(1+x)\rangle^2}(M-1) \equiv \ln H_0 - \alpha(M-1)\end{array} \tag{6.57}$$

and

$$\begin{array}{l}E+1 \approx \frac{\langle \ln(1+x)\rangle \langle \ln D\rangle - \langle \ln(1+x)\ln D\rangle}{\left\langle \ln^2(1+x)\right\rangle - \langle \ln(1+x)\rangle^2} - \\ -\frac{\langle \ln(1+x)\rangle \left\langle \frac{x}{1+x}\right\rangle - \left\langle \frac{x\ln(1+x)}{1+x}\right\rangle}{\left\langle \ln^2(1+x)\right\rangle - \langle \ln(1+x)\rangle^2}(M-1) \equiv E_0 + 1 - \beta(M-1)\end{array} \tag{6.58}$$

Multiplying the last equation by M, adding and substracting ones and using the first minimum condition one can write down the equivalent equation

$$\left\langle \frac{1}{1+Mx}\right\rangle \ln H - \left\langle \frac{\ln(1+x)}{1+Mx}\right\rangle (E+1) = \frac{\langle \ln D\rangle}{\langle 1+Mx\rangle} - \left\langle \frac{\ln\left(1+(M-1)\frac{x}{1+x}\right)}{1+Mx}\right\rangle \tag{6.59}$$

which admits the linearized approximation

$$M-1 \approx \frac{\left\langle \frac{1}{1+x}\right\rangle \ln H_0 - \left\langle \frac{\ln(1+x)}{1+x}\right\rangle (E_0+1) - \left\langle \frac{\ln D}{1+x}\right\rangle}{\left\langle \frac{1}{1+x}\right\rangle \alpha - \left\langle \frac{\ln(1+x)}{1+x}\right\rangle \beta + \left\langle \frac{x}{(1+x)^2}\right\rangle \ln H_0 - \left\langle \frac{x\ln(1+x)}{(1+x)^2}\right\rangle (E_0+1) - \left\langle \frac{x\ln D}{(1+x)^2}\right\rangle - \left\langle \frac{x}{(1+x)^2}\right\rangle} \tag{6.60}$$

It is interesting to notice that the nonlinear equation for $M=1$ is equivalent with the vanishing of the R-derivative in the ordinary model that does not have a zero. The introduction of the M -parameter seems then to add the correct degree of freedom to optimize in the intermediate region.

6.4 General Second Order Model

Both mixture models of previous sections can be described by

$$D(r) = \frac{H}{\left(1+\frac{r}{R}\right)^{E+1-W}\left(1+M\frac{r}{R}\right)^{W}} \tag{6.61}$$

for $W = \pm 1$. For other values of W one has a more general four parameter model which produces the logarithmic derivative

$$R\frac{D'(r)}{D(r)} = -\frac{E+1-W}{1+\frac{r}{R}} - \frac{W}{1+M\frac{r}{R}} \tag{6.62}$$

This corresponds to the discrete model

$$\frac{p(k)-p(k+1)}{p(k)} \equiv 1-\phi(k) = \frac{E+1-W}{k+R} + \frac{W}{k+R'} \tag{6.63}$$

which is a proper discrete model if $R, M, W, E > 0$ and $W < E+1$.

For $M \to 1$ the model reduces to the simples one for all values of W. As before one will consider cases were $M \approx 1$, or at least the linearization of the model in terms of $M-1$ with initialization purposes.

Again, one uses the notation $x \equiv r/R$ and one studies all parameters as functions of x.

6.4.1 Area

Let us compute in the scale free model

$$\begin{array}{l} A_{sf} = \int\limits_0^\infty dx \frac{1}{(1+x)^{E+1-W}(1+Mx)^W} = \int\limits_1^\infty dx x^{W-E-1} \frac{1}{(1-M+Mx)^W} \\ = \int\limits_0^1 dy y^{E-1} \frac{1}{(M+(1-M)y)^W} = \frac{1}{M}\int\limits_0^1 dy y^{E-1}\left(1+\frac{1-M}{M}y\right)^{-W} \end{array} \tag{6.64}$$

Let us recall here that ${}_2F_1(a,b;c;z) \equiv \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int\limits_0^1 dx \frac{x^{b-1}(1-x)^{c-b-1}}{(1-xz)^a}$ and then one has

$$A_{sf} = \frac{1}{M^W E} F\left(W, E; E+1; \frac{M-1}{M}\right) \tag{6.65}$$

or by using the transformation

$$F(a,b;c;z) = (1-z)^{-a} F\left(a, c-b; c; \frac{z}{z-1}\right) \tag{6.66}$$

one can write the simpler form:

$$\begin{aligned} A_{sf} &= \tfrac{1}{E} F\left(W, 1; E+1; 1-M\right) = \tfrac{1}{M} \sum_{n=0}^{\infty} \tfrac{(W)_n}{(E+1)_n} (1-M)^n = \\ &= \tfrac{1}{E} \left(1 + \tfrac{W}{E+1}\left(1-M\right) + \tfrac{W(W+1)}{(E+1)(E+2)}(1-M)^2 + \cdots\right) \end{aligned} \tag{6.67}$$

For $M \approx 1$ the area of the model is bigger (lower) than in the simplest case if $W(1-M)$ is positive (negative).

6.4.2 Tail

Also in the scale free model one has:

$$\begin{aligned} &A_{sf}\left(x_0\right) = \textstyle\int_{x_0}^{\infty} \frac{1}{(1+x)^{E+1-W}(1+Mx)^W} = \int\limits_{1+x_0}^{\infty} dx x^{W-E-1} \frac{1}{(1-M+Mx)^W} \\ &= \int\limits_{0}^{\frac{1}{1+x_0}} dy y^{E-1} \frac{1}{(M+(1-M)y)^W} = \frac{1}{(1+x_0)^E} \int\limits_{0}^{1} dy y^{E-1} \frac{1}{\left(M+\frac{1-M}{1+x_0}y\right)^W} \\ &= \frac{1}{M^W(1+x_0)^E} \int\limits_{0}^{1} dy y^{E-1} \left(1 + \frac{1-M}{M(1+x_0)}y\right)^{-W} = \frac{1}{EM^W(1+x_0)^E} F\left(W, E; E+1; \frac{M-1}{M(1+x_0)}\right) \end{aligned} \tag{6.68}$$

or using the hypergeometric transformation as before:

$$A_{sf}\left(x_0\right) = \frac{\left(1+Mx_0\right)^W}{E(1+x_0)^{E-W}} F\left(W, 1; E+1; \frac{1-M}{1+Mx_0}\right) \tag{6.69}$$

To evaluate tails or niche sizes (difference of two tails) one could use a Padé approximation over the series about the origin of z:

$$A_{sf}\left(x_0\right) = \frac{\left(1+Mx_0\right)^W}{E(1+x_0)^{E-W}} \left(1 + \frac{W}{E+1}z + \frac{W\left(W+1\right)}{\left(E+1\right)\left(E+2\right)}z^2 + \cdots\right) : z \equiv \frac{1-M}{1+Mx_0} \tag{6.70}$$

6.4.3 4+1 Parameter Regression

Recalling $D\left(r\right) = \frac{H}{(1+r/R)^{E+1-W}(1+Mr/R)^W}$ one proposes there to minimize the functional:

$$F\left(H,E,W,M;R\right)\equiv\frac{1}{2N}\sum_{n=1}^{N}\left(\ln H-\left(E+1\right)\ln\left(1+x_n\right)-W\ln\frac{1+Mx_n}{1+x_n}-\ln D_n\right)^2 \tag{6.71}$$

This is denoted as 4+1 since the R-parameter is not considered as a degree of freedom but as an additional variable on which all other parameters of the model depend. The derivatives on the true degrees of freedom are given by

$$\begin{array}{l}\frac{\partial F}{\partial(\ln H)}=\frac{1}{N}\sum\limits_{n=1}^{N}\left(\ln H-\left(E+1\right)\ln\left(1+x_n\right)-W\ln\frac{1+Mx_n}{1+x_n}-\ln D_n\right)\\ \frac{\partial F}{\partial E}=-\frac{1}{N}\sum\limits_{n=1}^{N}\left(\ln H-\left(E+1\right)\ln\left(1+x_n\right)-W\ln\frac{1+Mx_n}{1+x_n}-\ln D_n\right)\ln\left(1+x_n\right)\\ \frac{\partial F}{\partial W}=-\frac{1}{N}\sum\limits_{n=1}^{N}\left(\ln H-\left(E+1\right)\ln\left(1+x_n\right)-W\ln\frac{1+Mx_n}{1+x_n}-\ln D_n\right)\ln\frac{1+Mx_n}{1+x_n}\\ \frac{\partial F}{\partial M}=-\frac{W}{N}\sum\limits_{n=1}^{N}\left(\ln H-\left(E+1\right)\ln\left(1+x_n\right)-W\ln\frac{1+Mx_n}{1+x_n}-\ln D_n\right)\frac{x_n}{1+Mx_n}\end{array} \tag{6.72}$$

Since one is interested in values $M\approx 1$, it is natural to introduce $M-1\equiv m$ and replace

$$\frac{1+Mx_n}{1+x_n}=1+m\xi_n:\xi_n\equiv\frac{x_n}{1+x_n} \tag{6.73}$$

The extreme conditions become

$$\begin{array}{l}\ln H-\left\langle\ln\left(1+x\right)\right\rangle\left(E+1\right)-\left\langle\ln\left(1+m\xi\right)\right\rangle W=\left\langle\ln D\right\rangle\\ \left\langle\ln\left(1+x\right)\right\rangle\ln H-\left\langle\ln^2\left(1+x\right)\right\rangle\left(E+1\right)-\left\langle\ln\left(1+x\right)\ln\left(1+m\xi\right)\right\rangle W=\left\langle\ln\left(1+x\right)\ln D\right\rangle\\ \left\langle\ln\left(1+m\xi\right)\right\rangle\ln H-\left\langle\ln\left(1+x\right)\ln\left(1+m\xi\right)\right\rangle\left(E+1\right)-\left\langle\ln^2\left(1+m\xi\right)\right\rangle W=\left\langle\ln\left(1+m\xi\right)lmD\right\rangle\\ \left\langle\frac{1}{1+m\xi}\right\rangle\ln H-\left\langle\frac{\ln(1+x)}{1+m\xi}\right\rangle\left(E+1\right)-\left\langle\frac{\ln(1+m\xi)}{1+m\xi}\right\rangle W=\left\langle\frac{\ln D}{1+m\xi}\right\rangle\end{array} \tag{6.74}$$

6.4.4 3+2 Parameter regression

Let us write again the expression for the proposed model in a symetric way as:

$$D\left(x\right)=\frac{H}{\left(1+x\right)^{W}\left(1+x'\right)^{W'}} \tag{6.75}$$

with tne notation $x \equiv r/R : x' \equiv r/R'$. Let us start by considering R, R' as non regressive degrees of freedom and produce a $3+2$ model given the functional:

$$F\left(H, W, W'; R, R'\right) \equiv \frac{1}{2N} \sum_{n=1}^{N} \left(\ln H - W \ln\left(1 + x_n\right) - W' \ln\left(1 + x'_n\right) - \ln D_n\right)^2 \tag{6.76}$$

i.e.

$$\begin{pmatrix} 1 & -\langle \ln(1+x) \rangle & -\langle \ln(1+x') \rangle \\ -\langle \ln(1+x) \rangle & \langle \ln^2(1+x) \rangle & \langle \ln(1+x)\ln(1+x') \rangle \\ -\langle \ln(1+x') \rangle & \langle \ln(1+x)\ln(1+x') \rangle & \langle \ln^2(1+x) \rangle \end{pmatrix} \begin{pmatrix} \ln H \\ W \\ W' \end{pmatrix} = \\ = \begin{pmatrix} \langle \ln D \rangle \\ -\langle \ln(1+x)\ln D \rangle \\ -\langle \ln(1+x')\ln D \rangle \end{pmatrix} \tag{6.77}$$

$[H, W, W']$ are known as the solutions of a non singular symetric $3x3$ linear system. The two rank scale parameters R, R' can be used to reoptimize the functional or to be determined by other non regressive procedures.

For instance, one can reoptimize against R' . Equating to zero the derivative of the functional leads us to

$$\left\langle \frac{1}{1+x'} \right\rangle \ln H - \left\langle \frac{\ln(1+x)}{1+x'} \right\rangle W - \left\langle \frac{\ln(1+x')}{1+x'} \right\rangle W' - \left\langle \frac{\ln D}{1+x'} \right\rangle = 0 \tag{6.78}$$

which reduces the problem to finding the zero of can be a function of R' for fixed R and with H, W, W' coming from the linear system. This produces an optimal value for R' leaving R as the only non-regressive parameter. Notice that one can always assume $R < R'$ by breaking the symmetry in notation implicit in the model.

$$\left\langle \frac{1}{1+x} \right\rangle \ln H - \left\langle \frac{\ln(1+x)}{1+x} \right\rangle W - \left\langle \frac{\ln(1+x')}{1+x} \right\rangle W' - \left\langle \frac{\ln D}{1+x} \right\rangle = 0 \tag{6.79}$$

6.5 Discrete Long Tail model

Let us assume a set of objects $S_k \in \{S\} : k = 1, 2, \ldots$. The index on the natural numbers implies an order among the objects

$$S_k \prec S_{k'} \Leftrightarrow k > k' \tag{6.80}$$

The order of symbols $\prec$ and $\succ$ can be thought as preferences of some abstract user (man in the sky) over the set. This MITS prefers interms with lowest indexes. According to Von Neumann and Morgensten it is possible to design a preference function $Pref(S)$ which implements the MITS subjective preferences

$$S \prec S_{k'} \Leftrightarrow \text{Pref}(S) > \text{Pref}(S') \tag{6.81}$$

One can define here the numbers

$$p_k \equiv \text{Pref}(S_k) : k = 1, 2, \ldots \tag{6.82}$$

and necessarily

$$p_k > p_{k'} \Leftrightarrow k < k' \tag{6.83}$$

By using a translation, if necessary, all this numbers can be taken as positive and the above scenario is equivalent to the existence of the existence of a monotone decreasing sequence of positive numbers

$$p_1 > p_2 > \cdots > p_k > p_{k+1} > \cdots > 0 \tag{6.84}$$

For an infinite sequence one must also demand

$$\sum_{k=1}^{\infty} p_k = P < \infty \tag{6.85}$$

which requires as necessary condition

$$\lim_{k \to \infty} p_k = 0 \tag{6.86}$$

This is only to have a *normalizable* sequence even in the case of an infinite number of elements.

6.5.1 The unit limit axiom

Let us now introduce the ratios of these numbers as

$$p_{k+1} = \rho_k p_k : k = 1, 2, \ldots \tag{6.87}$$

According to the previous discussion one needs

$$0 < \rho_k < 1 : k = 1, 2, \ldots \tag{6.88}$$

The factor ρ_k expresses the preference reduction between successive ranks. Notice that if the ratios have a limit $\lim_{k\to\infty}\rho_k = \rho < 1$, for ranks higher enough one would have an exponential decay of the form $\rho_k \sim C\rho^k : k \gg 1$. To avoid this asymptotic behavior let us impose the additional condition

$$\lim_{k\to\infty} \rho_k = 1 \tag{6.89}$$

One of the simplest models one may build conforming with the above properties is

$$\rho_k = \frac{k + R - E - 1}{k + R} : k = 1, 2, \ldots \tag{6.90}$$

where one has written the constants of the fraction in this way for convenience. Let us assume $R > 0$ and notice that $1 - \rho_k = \frac{E+1}{k+R}$.Then if one requires

$$-1 < E < R \tag{6.91}$$

one has $0 < \rho_k < 1 : k = 1, 2, \ldots$ and $\lim_{k\to\infty}\rho_k = 1$ as required.

The explicit form of the preference numbers is easily worked out as

$$\begin{array}{l} \rho_{k+1} = \frac{k+R-E-1}{k+R} p_k = \frac{(k+R-E-1)(k+R-E-2)}{(k+R)(k+R-1)} p_{k-1} = \\ = \frac{(k+R-E-1)(k+R-E-2)\ldots(R-E-1)}{(k+R)(k+R-1)\ldots R} p_1 \end{array} \tag{6.92}$$

i.e.

$$p_{k+1} = \frac{\Gamma(R+1)\,\Gamma(k+R-E)}{\Gamma(R-E)\,\Gamma(k+R+1)} p_1 : k = 1, 2, \ldots \tag{6.93}$$

which exhibits the power law behavior

$$p_k \sim \text{const} k^{-E-1} : k \sim \infty \tag{6.94}$$

This can be checkout for instance by introducing the beta function

$$B(a,b) \equiv \int_0^1 dt t^{a-1}(1-t)^{b-1} = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{6.95}$$

which allows us to express

$$\begin{array}{l} p_{k+1} = \frac{\Gamma(R+1)}{\Gamma(R-E)\Gamma(E+1)} \frac{\Gamma(k+R-E)\Gamma(E+1)}{\Gamma(k+R+1)} p_1 = \\ = \frac{p_1}{B(R-E,E+1)} B(k+R-E, E+1) = \\ = \frac{p_1}{B(R-E,E+1)} \int_0^1 dt t^E (1-t)^{k+R-E-1} : k = 1,2,\ldots \end{array} \tag{6.96}$$

To study the behavior at $k \sim \infty$ one may perform the change of variables in the integral $t \to y/k$ which leads us to

$$\begin{array}{l} p_{k+1} = \frac{p_1}{B(R-E,E+1)} \frac{1}{k^{E+1}} \int_0^\infty dy y^E \left(1 - \frac{y}{k}\right)^{k+R-E-1} \\ \sim_{k\to\infty} \frac{p_1}{B(R-E,E+1)} \frac{1}{k^{E+1}} \int_0^\infty dy y^E e^{-y} = \\ = \frac{p_1 \Gamma(E+1)}{B(R-E,E+1)} \frac{1}{k^{E+1}} = \frac{\Gamma(R+1)}{\Gamma(R-E)} \frac{p_1}{k^{E+1}} \end{array} \tag{6.97}$$

6.5.2 The continuous model

The equation $\frac{p_{k+1}-p_k}{p_k} = \rho_k$ may be thought in the continuous model as the differential equation

$$\frac{d \ln p(k)}{dk} = \rho(k) - 1 \tag{6.98}$$

with the solution

$$p(k) = const \exp\left(-\int^k dk' \left(1 - \rho\left(k'\right)\right)\right) \tag{6.99}$$

In the simplest case examined before

$$\frac{d \ln p(k)}{dk} = \frac{-E-1}{k+R} \tag{6.100}$$

and $p(k) = H\left(1+\frac{k}{R}\right)^{-E-1}$ which corresponds to the continuous model examined in other sections with H playing the role of integration constant.

6.5.3 The second order model

A natural generalization will produce a [2/2] rational form for the preferences ratio

$$1-\rho_k=\frac{W}{k+R}+\frac{W'}{k+R'} \tag{6.101}$$

Notice that for $R=R'$ one recovers the simple model described above with parameters $E=W+W'-1,R$.

The continuous version of this model is

$$\frac{d\ln p\left(k\right)}{dk}=-\frac{W}{k+R}-\frac{W'}{k+R'} \tag{6.102}$$

which produces the integrated form

$$p\left(k\right)=\frac{H}{\left(1+\frac{k}{R}\right)^{W}\left(1+\frac{k}{R'}\right)^{W'}} \tag{6.103}$$

which has a tail with exponent $E=W+W'-1$ but resolved into two spectral components with exponents $E_1=W-1:E_2=W'-1$. Returning to the discrete case it is easy to see that

$$\frac{p_{k+1}}{p_k}=1-\frac{W}{k+R}-\frac{W'}{k+R'}=\frac{k^2+\left(R+R'-W-W'\right)k+RR'-WR'-W'R}{\left(k+R\right)\left(k+R'\right)} \tag{6.104}$$

The roots of the numerator are

$$\begin{array}{l}-S\equiv\frac{W+W'-R-R'}{2}-\sqrt{\frac{\left(W+W'-R-R'\right)^2}{4}-RR'+WR'+W'R}\\-S'\equiv\frac{W+W'-R-R'}{2}+\sqrt{\frac{\left(W+W'-R-R'\right)^2}{4}-RR'+WR'+W'R}\end{array} \tag{6.105}$$

and one can write $\frac{p_{k+1}}{p_k}=\frac{(k+S)\left(k+S'\right)}{(k+R)(k+R')}$ which admits the closed form

$$p_k=\frac{\Gamma\left(R\right)\Gamma\left(R'\right)}{\Gamma\left(S\right)\Gamma\left(S'\right)}=\frac{\Gamma\left(k-1+S\right)\Gamma\left(k-1+S'\right)}{\Gamma\left(k-1+R\right)\Gamma\left(k-1+R'\right)}p_1 \tag{6.106}$$

6.6 *mElo* algorithm

The *mElo* algorithm provides popularity scoring for all items in a catalog in order to create a complete long tail distribution of them. mElo is inspired in 2 different sources:

- The Elo rating system is a method for calculating the relative skill levels of players in competitor-versus-competitor games such as chess. It is named after its creator Arpad Elo, a Hungarian-born American physics professor. if you play against someone with higher score than you and you win, you will add more points to you own Elo that if you play against someone with lower score than you. If you loose against some with less Elo than you, you will loose more points that if you loose against someone a priori better (with more points). This makes the rating system self-correcting. A player whose rating is too low should, in the long run, do better than the rating system predicts, and thus gain rating points until the rating reflects the true playing strength. MELO is the application of this idea to a music catalog, but everything within the long-tail framework.
- Theory of Games and Economic Behavior which asserts that there is a function that is able to reproduce the outcomes of a set of pair-wise preferences between the items in the set. Elo rating system: A system initially created for ranking chess players, it is a system for ranking players, for calculating the relative skill levels of players in two-player games such as chess. (Note: Elo comes from the creator name Arpad Elo, so you should not use it with all capital letters ELO because it is not an acronym)

Imagine you have 100 football player candidates from which you do not have any information regarding their skills. How can you know their skills level? Making a tournament between them, but better than that, making that tournament adding some professional football players into the games. That way, after several rounds of the tournament you will find the best players, the ones that will remind competing and playing against the professional ones. The professional players have the highest skill ratings at the beginning of the tournament, while all the new players have zero ratings, but at the end of the tournament, the best new players will have highest ratings closer to professional ones.

As we said, mElo was *inspired* in Elo rating system. One of the main differences between mElo and Elo is the target of scoring distribution. Elo system uses Gaussian distribution as final outcome (see image below), while mElo uses the long tail distribution we mentioned in previous section. In a Gaussian distribution the curve is concentrated in the center and decreases on either side. The most common values are in the middle, while the outstanding and non-common values are in the extreme values.

The other main difference between mElo and Elo is that with media items, is the way to define a game and a winner for that game. In mElo we are using the audio similarity of tracks for that target. mElo (from Media-Elo) uses the Elo rating system concept and this dissertation recommendation algorithm to model the scoring or demand curve for large set of objects. It predicts the relative score of a new item

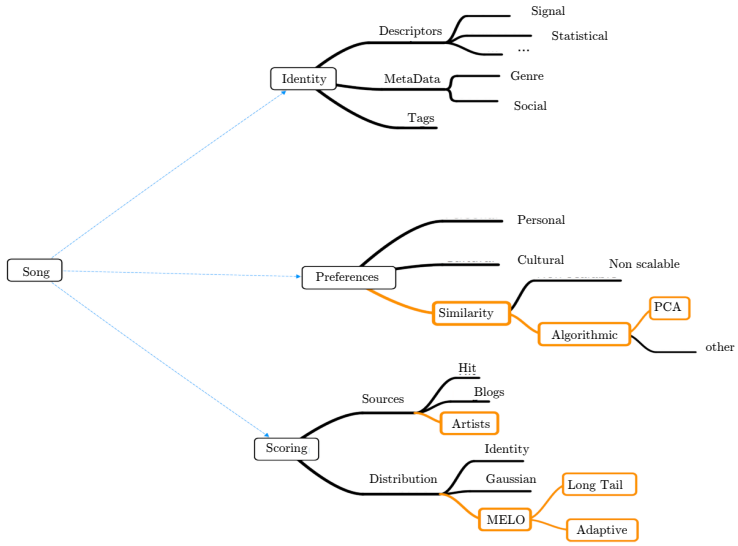


Figure 6.11: Identity vs Similarity

in the universe. 6.11 represents the entire set of information of a given track. mElo algorithm is using the similarity extracted features to infer the popularity of tracks inside the Long Tail as if there was an equal marketing action on all of them and the only key factor was the music.

Using the most popular ones based on charts (or any other popularity or ranking system we want to use), and the audio similarity between tracks, mElo applies a method for obtaining the full distribution of scores for all catalog. mElo uses the part of the catalog for which the scoring in long tail is known (i.e. the ones that are most popular, most downloaded, most streamed or whatever we want to use as a basis for scoring), and predict the scorings of the rest of the catalog in order to have at the end a long tail distribution. mElo is NOT a recommendation algorithm, although it can be used for recommendations. mElo is more a popularity algorithm than a recommendation system.

Imagine that we want to find the best-unknown soccer players in Europe. A strategy would be to start by looking for the best-unknown soccer players in every country and then combine then together. We can ask the best already known players in each country

to help. Basically we do the following:

Data: Set of unknown football players
Result: Selection of the best unknown football players
Randomly divide the set of players in groups;
while *unstable* **do**
  Make the players in each group play against each other in a one-vs-one game.;
  Accumulate the number of wins and losses for each player;
  Repeat again until the score stabilizes;
  Players are scored less randomly and more by their score.
end

Algorithm 2: Divide approach to the identification of the best players

At the end we will have the best known players playing against the best unknown players in the top score groups while we will have the worst unknown players playing against them in the low score groups.

6.6.1 Algorithm description

- The Long Tail model is described by Chris Anderson in his book [Anderson06]. Data plots from this book are used as backgrounds to draw the different curves rating.

- However, the scales of the axes and the values of the critical constants R and E have been estimated from diagrams of the book.

- Rhapsody is an online music store subscription service based in Seattle. On April 6, 2010, Rhapsody relaunched as a standalone company, separate from former parent RealNetworks. The goal of this algorithm is to match the LongTail histograms obtained by the real Rhapsody music store.

- In Rhapsody, the number of downloads (Y axis) ranges from 0 to 200000. That is the maximum value predicted by a song is 200000. In this simulation the Area constant defines the *total number* of downloads. The initial values assigned to the sources of value are adjusted so that the scale goes from 0 to $maxval$. If you choose $maxval = 2000$ there is a simple factor of 100 with the ordinate scale in Rhapsody. The songs that are not sources inherit less with what is determined by locally this choice. Note that this scale is only used to assign the initial values to the sources between 0 and $maxval$.

- The global scale determines the area. When the algorithm iterates, the tracks that are not sources increase their value while the rest will decrease so the total number remains the same.

- The rank of the tracks goes from 0 to 800000. In the simulation of appendix 2, a value of $numSongs = 80000$ is used which will correspond to a factor of 10 with the original Rhapsody download histograms.

6.6.2 Long Tail Parameters

- An essential component of the model is the value of the Long Tail function constants imposed on the final distribution of value. It is equivalent to the parameters of the Gaussian model that Elo requires the distribution value in chess.
- It is assumed that these constants are essentially universal and are related to the dynamics of accessing and download a set (infinity) songs dominated by the knowledge of some elected representatives (hits) using a relational element given by the similarity between songs.
- The constant E is non-dimentional and its value is what fits Rhapsody phenomenon seen and taken here as reference.
- The constant R is proportional to the total number of songs. The proportionality constant is taken here also from the Long Tail phenomenon as shown in Rahpsody.

6.6.3 Initialization

- The aim of this module is to study the possibilities of generating a distribution of emerging value, inspired by the Elo chess on a sample abstract data that act as representatives of songs in the universe.
- There are sources of a priori value that are produced by the selection that DJs do during their programming or track selection. Here the concept DJ is used like any numerical value as the value of a supervised song. Other values could be used such other popularity sources or even user scorings (like an dislikes).
- These sources of value do not cover the complete set of songs. In fact, the reliable sources are possibly a small fraction of the total.
- The initialization is done in two stages. The first is hit identify understood here as the subset of songs for which there is a reliable numerical measure. This measure is normalized between 0 and $maxval$, which are constants that allow to set the scale in this implementation.
- In the second stage every song receives an export value from the hit nearest as in 6.13. That is, using Euclidean distance or the curved distanced described in Chapter 4, the minimum distance between the song and hit (known value) is calculated. Then it is assigned a received value $d_{hit} * exp(-d_{min}/d)$. Here d_{hit} is the known value of the hit and the exponential factor is the attenuation. The value of the constant d in the denominator depends on the scale in which evaluate d_{min}. Notes that the exponential factor attenuation is about $1/7$ to $d_{min} = 2 * d$.
- With this procedure a distribution value for all songs is obtained. It is used as an initial determination of *mElo* for all practical purposes.
- Clearly, many other processes of initial valuation are possible depending on supervised (or not) information available. For example, it would be possible to use an initial division by using metadata supergenre

6.6.4 Methodology

- A valuation $p(n)$ for item in rank n will verify a Long Tail distribution if $p(n+1)/p(n) = 1-(E+1)/(n+R)$ where E, R are the characteristic parameters of the tail. E is the exponent and *R* is the actual origin of the range.
- The valuation of supervised tracks does not follow this relationship and does not generate a Long Tail distribution.
- The extra notion of similarity imposes a new condition on this issue. This condition is *consistency* (continuity). Similar songs should have similar assessments. This is a source of additional information that *corrects* the supervised valuation.
- The Elo notion of ordination in chess is a distribution of values of Gaussian quality and sets a correction algorithm of the Â current rating by the results of official games who normally occur in tournaments.
- The *musical Elo* should impose a power law distribution over the tracks and correct it accordingly. One way is to use the above expression adjusting the model parameters E, R as in the Raphsody downloads data, which are entitled here as universal for the output of the simulation.
- The *musical Elo* will be called *mELO*.
- The information on Euclidean distance is processed approximatelyÂ for speed reasons on a reticle (grid) of square cells. $grid(n, m, k)$ contains the $k-th$ song in the cell n, m. The total number of songs in the cell is $nsongcell(n, m)$. Each song belongs to a single cell and its neighbors songs are considered *closed* by the algorithm. The accumulated Elo is stored in a cell at $meloc(n, m)$.
- The calculation of MELO is performed iteratively by celebration of tournaments. A tournament takes place over a window of the current range $2W+1$. It involves the songs in the range $r-W : r+W$ so the window is specified by the pair (r, W). The r value ranges between $W+1$ and $numSongs - W$.
- Each song takes as valuation the total of the cell where it is contained ($meloc$). With this score, the classification is calculated which determines the new interior range of the selected window.
- Sorting according to previous qualification, the items acquire absolute ranges: $rank = r - W - 1 + k : k = 1, 2, ..., 2W + 1$ so their assigned *mElo* shoud verify: $m(k+1) = m1 * (1-(E+1)/(RR+k)) * (1-(E+1)/(RR+k-1)) * ... * (1-(E+1)/(RR+1))$ which can also be written as $m(k+1) = m1 * Prod(1 : k)(1-(E+1)/(RR+i))$ con $RR = R + r - W - 1$. The product form can be compacted as $P(k)$ and then $m(k+1) = m1 * P(k)$. The only free parameter is m_1 if R and E are considered initial constants. Other alternatives can consider R and/or E as parameters what would have to be re-adjusted during tournaments.
- Then the *mElo* items in the window $r - W - 1 + k, k = 1, 2, ..., 2W + 1$ have determined values ecept the first value given by m_1. It is possible to impose additional conditions on the placement of the window in the total graph. The *mElo* immediately before to the window is $m(r - W - 1)$. By continuity should be $m_1 * (1-(E+1)/(R+r-W-1))$. This can be achieved by multiplying all previous distribution to the window by a factor of continuity $pre = m_1 * (1-(E+1)/(R+r-W-1))/m(r-W-1)$. Ehis also guarantees all values are above the upper window to it. Analogously, we can multiply all the items after

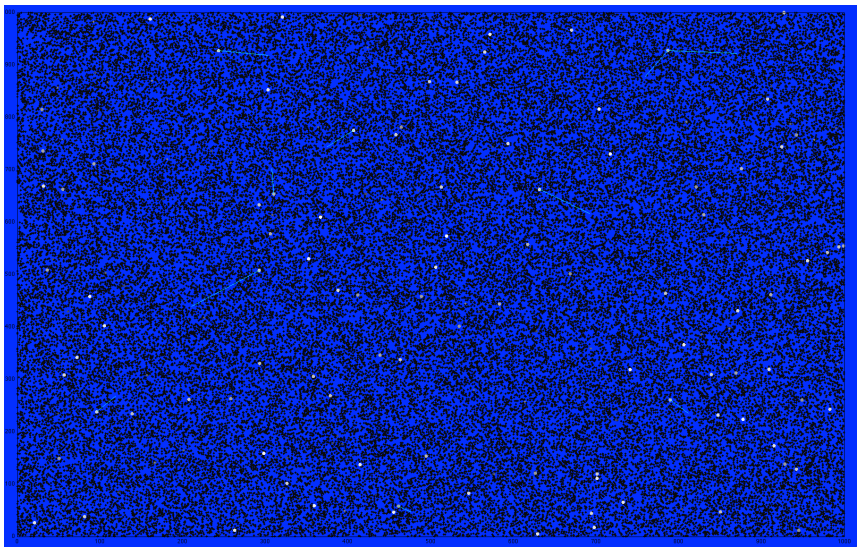


Figure 6.12: Hits (white) represented over non-hit tracks (black) over blue background.

the window by a factor $post = m1 * (1 - (E+1)/(R+r+W+1))/m(r+W+1)$ corresponding to $k = 2W + 2$, imposing the continuity and makes all subsequent lower values the window.

- The value of m_1 is now needed for normalization of the total area. Initially the total area verifies $Apre + AWin + Apost = Area$. With the new *mElo*, the normalization imposes $m_1 * fpre * Apre + m_1 * A'Win + m_1 * fpost * Apost = Area$ where $ApWin$ is the new area of the window. Therefore: $m_1 = Area/(fpre * Apre + A'Win + fpost * Apost)$ ensures standardization and the values in the window along with the anterior and posterior parts are univocally determined.

6.6.5 Song representation

- In this simulation a song has the unique identifier coordinates (x, y) in two dimensions. These values could come from a reduction in major components of a multidimensional model. Its equivalent is the structure that identifies each song the complete set as described in previous chapters (i.e. 65 music track descriptors). Here are randomly assigned as part of the initial process. Fig. 6.12 shows the initial assignation.
- For reasons of efficiency, competition between songs is performed by comparison of local $mELO$ values ($meloc$). By this is meant the following: The set of songs is decomposed into clusters. Each song has a centroid centroid representative who has the $mELO$ an average of songs representing (or a subset thereof). The local $mELO$ of the $song_i$ is defined as the average $mELO$ representative centroid.
- In this simulation clusters are defined simply decomposing total rectangle (X, Y) square cells in pixels NG side. A song then has the authorized representative of

these cells and *meloc* is simply the accumulation of the *mElo*'s of the songs in the cell.

- The information about songs stored in a numSongs x 6 array with the following interpretation: $song(i,1) = x$, $song(i,2) = y$, $song(i,3) = melo$, $song(i,4) = log(melo)$, $song(i,5) = xcell$ and $song(i,6) = ycell$.

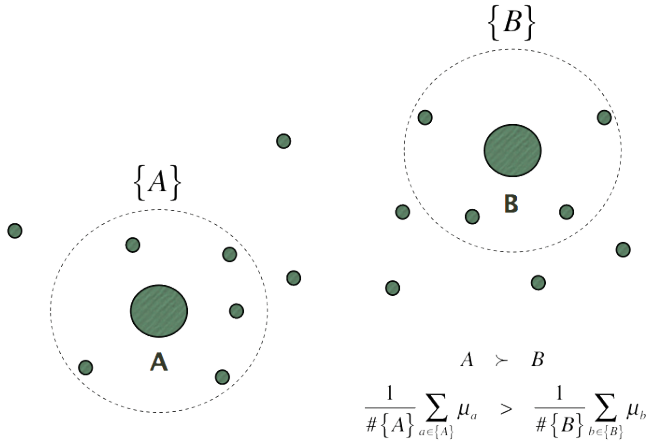


Figure 6.13: Similariby based recursive preferences. Use current MELO values in neighborhoods.

6.6.6 Value propagation Tournaments

- The organization of the songs is kept in an array $rank(1 : numSongs)$ such that $rank(p)$ gives the index in song of the song that presently occupies the position p in total worth ordination the set of songs.
- The rank modification is done by organizing tournaments. Each tournament involves only $2 * W + 1$ songs centered on a position p. The tracks that compete are in positions of rank $p - W, p - W + 1, ..., p + W - 1, p + W$.
- Tournaments organized in seasons. Every season has $numTournaments$ as a number of tournaments to perform. During a season the local melo is not updated. That is, it is assumed that variations in the evaluation can be ignored in the averages used in *meloc*. At the end of season, the *meloc* values are recalculated. Fig. 6.14 shows the window seletion during a tournament.
- This policy is controlled by the size of $numTournaments$. As extreme case, if $numTournaments$ equals to one, *meloc* is recalcultaed after each tournament individually.
- The total number of tournaments is $NT = numTemp * numTournaments$. A song will participate on average in a number of tournaments given by $NT(2W + 1)/numSongs$. Note that in each tournament a particular song can only alter their ranking for $2W + 1$ positions maximum. The maximum shift that a song can

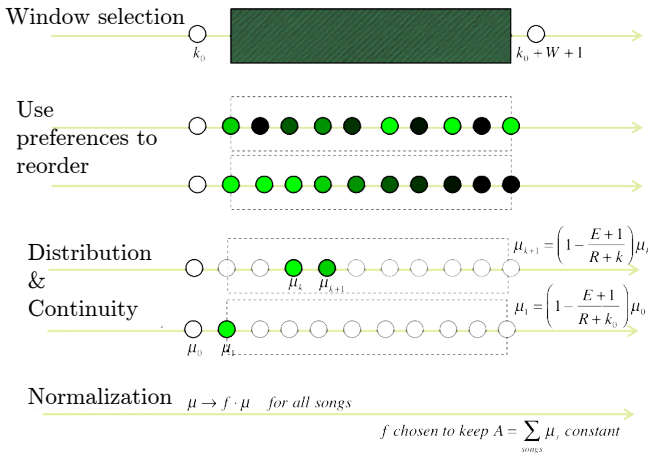


Figure 6.14: Melo Tournements

experience in the ranking to at the end of the season is $maxShift = NT(2W + 1)^2/numSongs$. As with numbers chosen for the simulation, $maxShift = 5010010000/80000 = 625$. Given 80000 songs, it is unlikely that massive shifts may be performed (eg from tail to head) without having a disproportionate luck. However, the stabilization as a Long Tail distribution of appears with these numbers. Therefore if it is necessary to respect approximately the initial ordering (no huge shifts) but spread the values over the Long Tail, the proposed numbers (even smaller windows) are recommended. If it is desired that the re-ordering is more significant, the most important factor is the width of the window as it intervenes quadratically.

- For a problem with 10^7 songs (approximately) with windows of width 10^4, $numTournaments = 10^4$, $numSeasons = 10^3$ the order of magnitude of possible shifts would be $10^14/10^7 = 10^7$ and therefore could be large relatively to the size of the complete assembly. Execution times must obviously play a dominant role as well.

- The main observation from the empirical simulations is that the algorithm converges regardless of the size of the window. If this is large adopts the LongTail form quickly and the rest of the time is spent rearranging songs without changing much the whole way. For smaller windows, mot of the times is used for obtaining the form because the initial order is greatly respected and this determines the progress.

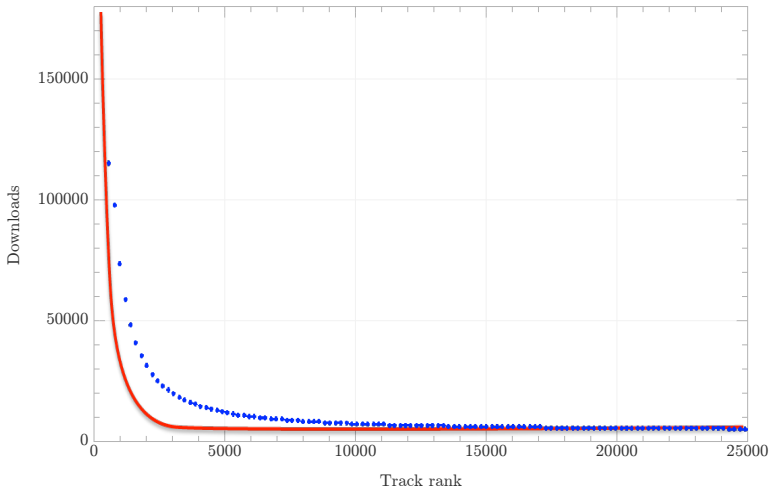


Figure 6.15: Rhapsody music downloads at the head region (rank 0 to 25.000)

6.6.7 Evolution in each iteration

- We assume here that the whole process initialization has been developed. The one that was applied in the simulation is totally arbitrary. Random positions generated x, y for identifiers of songs. About 100 hit songs are declared in a band around $maxval$. The remaining songs are assigned a mElo the exponential decay described above.
- A tournament is selected by determining the central range window r. The $2W+1$ elements in the window are $r - W : r + W$.
- Each season numTournaments played and end of each season refreshes the graph representing the current MELO $song(s, 4)$ contains the logarithm of melo $song(s, 3)$ of the song s. In this initial state $song(s, 3)$ contains no supervised ratings. Therefore a protection is adopted using a threshold strategy. If $song(s, 3)$ is higher than the threshold we consider a decent point and start using the logarithm. For all the rest put the same value. eExperimentally the threshold not seem to affect too much.

6.6.8 Algorithm results

The algorithm has been tested against real data from the Raphsody Downloads Dataset and the results are consistent with the expectations. If we take a look to the Head of the distribution (Fig. 6.15) we observed a slight disagreement with the reference data. However, the region where the Long-Tail starts (between the track ranks 25.000 and 100.000) shows a more accurate modeling. The last section of the distribution (the Tail) shows an even more precise match. The usefulnes of the *mElo* algorithm is that

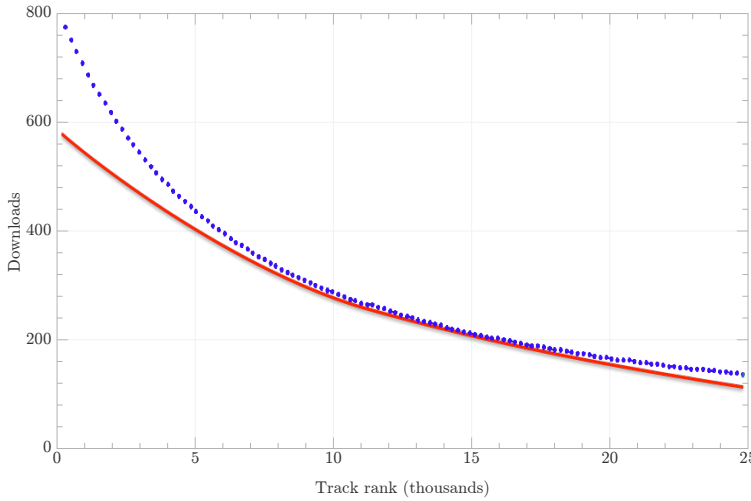


Figure 6.16: Rhapsody music downloads (rank 25.000 to 100.000)

it is able to propagate hitness values to tracks without popularity information and at the same time maintain a real Long Tail distribution.

The *mElo* algorithm can be used to refine recommendation and music discovery results. There are many tracks that are similar to tracks that have user exposure and visibility that are in the lowest download ranking positions. With this approach, those tracks could be promoted to the top of the list as their probable *hitness* would have been estimated.

A track needs to be purchased, streamed or downloaded at least once to have some popularity ranking. With mElo we can predict those values, so like the based-content technology that the similarity algorithm proposed in the dissertation is using for calculating the recommendations, a track can be recommended or can have a popularity ranking from the same moment that it is added to the catalog, regardless of the amount of users that have used or asked for that track.

It is also possible to expand the concept of popularity to *user*-centered popularity (territory based, style based). Users could define their own rules for a track to be a hit. And then apply this rules to actual hit tracks and then propagate the value to unknown tracks so the user could have its own personal *mElo* scoring tool.

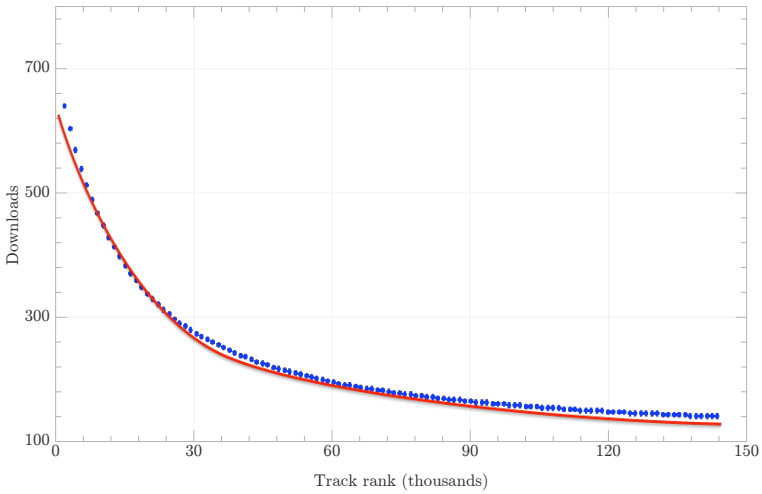


Figure 6.17: Rhapsody music downloads (rank 100.000 to 800.000)

6.7 Radio algorithm with MELO and similarity music selection

6.7.1 Radio journey description

The final goal of this dissertation is to provide a novel method to enhance music listening by carefully selecting the next track to be played. This can be viewed as real-time playlist generation or also dynamic personalized radio. After the current track ends playing, the algorithm is executed to provide the next track to play. At any moment the user can score the current tracks with a *like* or a *dislike*. The algorithm has also information of previous radio sessions and therefore has a list of all the tracks that have been played with user scores. If the user does not score a track but it listens to it entirely, it is also valid information to take into consideration.

Current radio streaming algorithms are based in *influences*. At each moment, the user can add a influence to the radio session. This influence can be a track, an artist or an album. The most basic radio influence algorithm is based on having a *bag of tracks* and whenever the user wants to add a positive influence, the radio algorithm uses similarity technique (like the one presented in Chapter 4) to add similar tracks to the influence track into the set of tracks. The novel approach presented in this dissertation introduces the concept of *radio journey*. The user selects a pair of tracks. The first one is the *origin* track where radio session starts. The second track is the *destination* track where the radio journey end. The goal of the algorithm is to provide a radio streaming session that smoothly transitions from the origin track to the destination time in a certain amount of time. Fig 6.18 shows tracks positioned in a two dimensional grid using a Self Organizing Map (SOM). The SOM tries to maintain distances in high

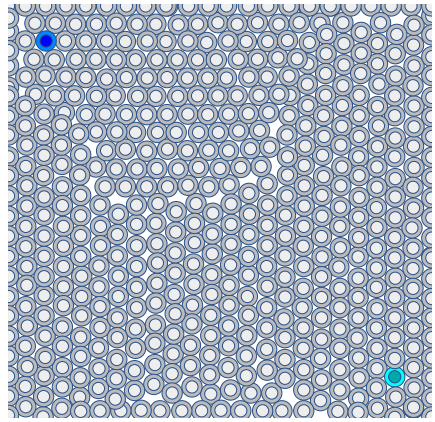


Figure 6.18: Origin (blue) and destination (cyan) tracks over a Self Organizing Map produced over the original 65 dimension track-to-track distances.

and 2D representation. So if two tracks have a relative high distance in the original 65 dimension space, they should also have a high distance in the projected two-dimensional space. The same should occur with very similar (near) tracks in the high dimensional space. Of course the algorithm cannot totally reproduce the high-dimension distances but it produces an approximation and it simplifies the description of the radio journey approach.

6.7.2 Graph implementation

The generalization of this algorithm considers that the user could start a radio journey from the current tracks that is being played. This means tha the algorithm has to be written so it can be used at any time, when new information is available. Therefore, the algorithm will consider as the origin tracks the current track being played and the user is always able to change the *destination* of the radio journey. The only value that changes is the number of tracks pending to be recommended. Therefore in each iteration, the algorithm has to provide a shorter journey. If a track has an average length of 4 minutes, the first playlist generated by the user should contain 15 tracks. Then after the user has listened to the first track, the algorithm is re-executed but this time it has to recommend only 14 tracks to arrive to the destination (of course it should be closer than in the previous iteration). The algorithm is executed at the end of every listened tracks and the only information is that it has to recommend a playlist of smaller length (just one less) and whether the user scored or not the track that has been played.

The entire collection of tracks can be organized in a graph structure using the similarity metric defined in Chapter 4. One of the main parameters of the radio journey is a *mElo* range and the user can modify this range to filter tracks (for example, just selecting the most popular tracks, or just long-tail less visible tracks, etc.). Only tracks that have their *mElo* value within the user selected range are added to the

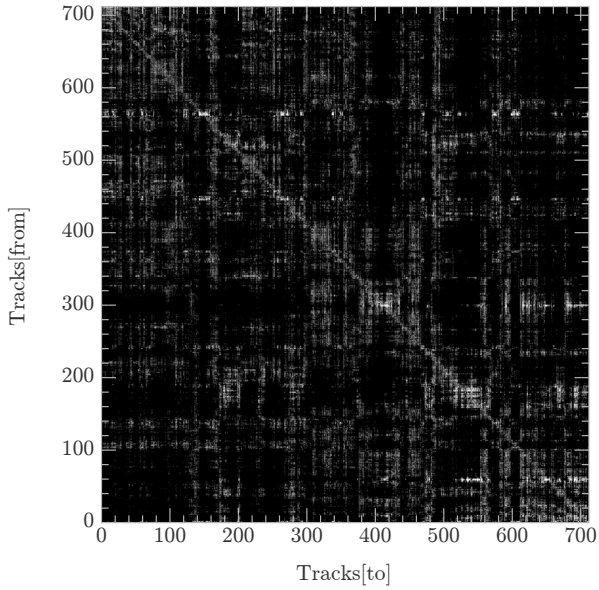


Figure 6.19: Graph link distances between tracks (black pixel means no direct link between track *col* to track *row*)

journey track graph. Therefore the Long-Tail distribution estimation proposed in this dissertation has a direct usage allowing the users to discover music sorted by estimated hit probability, instead of using the very limited existing popularity values.

Each track is assigned a node in the graph and a batch process is launched to search its most similar tracks (using a fast similarity algorithm because it has to be calculated to the entire track set (it can reach 100 million). These similar tracks are used to create links in the graph between themselves and the seed track. The distance of each link is proportional to the distance given by the similarity algorithm. The key aspect at this point is to define a distance threshold at which to stop creating links. Usually, this threshold is dynamic because even if the metric algorithm attempts to normalize the distances, in some regions of the track space the density of tracks varies. Another rule is to do not consider more than a determined amount of similar tracks. Fig. 6.19 shows the graph link matrix that relates two tracks together. The tracks in this graph have been organized so the tracks of the same family are in similar indexes so the diagonal appears wider. There are many combinations of *origin* track to *destination* track that do not have a direct link (they appear in black in Fig. 6.19) and therefore they require intermediate steps to go from one to the other.

Once the graph has been defined with a track assigned to each node and direct links with weighted distance just for similar tracks, it is possible to apply the Floy-Warshall method. This technique is a graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights and also for finding transitive closure of a relation R. A single execution of the algorithm will find the lengths (summed

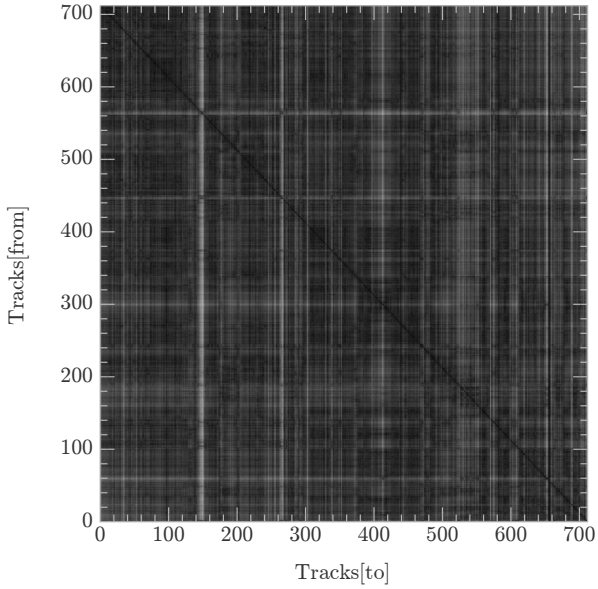


Figure 6.20: Total distance between tracks

weights) of the shortest paths between all pairs of vertices, though it does not return details of the paths themselves. The Floyd-Warshall algorithm compares all possible paths through the graph between each pair of vertices. It is able to do this with $\theta(|V|^3)$ comparisons in a graph. This is remarkable considering that there may be up to $\Omega(|V|^2)$ edges in the graph, and every combination of edges is tested. It does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal. Fig. 6.20 shows the summed distance results (the brighter, the higher the total distance) between each pair of tracks. Bright vertical or horizontal lines represent tracks that are very far away from all the tracks (also called *outlayers*). This matrix can be used to measure the total distance between the origin and destination track and therefore determining the distance jump between tracks to ensure the duration of the playlist session. So if the Floyd-Warshall algorithms determines that between origin track and destination track there is a total distance of 60 units (remember that similarity tries to normalize so single distance lower than one means very similar) and the number of pending tracks is 10, it means that the proposed tracks should have a sequential distance of 6 units.

When the user scores a listened track (positively or negatively) the algorithm has to use this information to improve the journey until the destination track. The distance graph is therefore altered to difficult that the journeys come near tracks that have been scored negatively and tries to favor going through tracks that have been flagged positively. Tracks that are similar to tracks that have been scored positively will increase their probability of appearing in the playlist while at the same time tracks that are similar to negative scored tracks will have lower probability of appearing. Fig. 6.21 shows the propagation effect of the score through the graph.

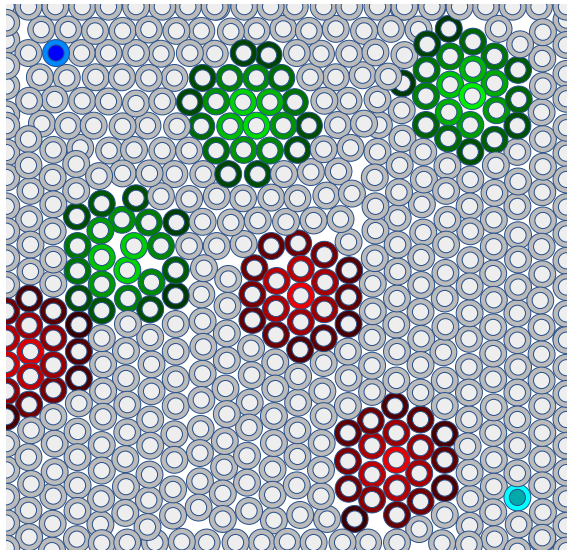


Figure 6.21: Radio graph with postive and negative score propagation

6.7.3 Path-finding through track music graph

To help with this path-finding problem, the more versatile technique is the A* search algorithm. Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute (now SRI International) first described the algorithm in 1968 [Hart68]. It is an extension of Edsger Dijkstra's 1959 algorithm. A* achieves better time performance by using heuristics. Noted for its performance and accuracy, it enjoys widespread use. However, in practical travel-routing systems, it is generally outperformed by algorithms that can pre-process the graph to attain better performance.

A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way.

It uses a knowledge-plus-heuristic cost function of node x (usually denoted $f(x)$) to determine the order in which the search visits nodes in the tree. The cost function is a sum of two functions:

- the past path-cost function, which is the known distance from the starting node to the current node x (usually denoted $g(x)$).
- a future path-cost function, which is an admissible *heuristic estimate* of the distance from x to the goal (usually denoted $h(x)$).

The $h(x)$ part of the $f(x)$ function must be an admissible heuristic; that is, it

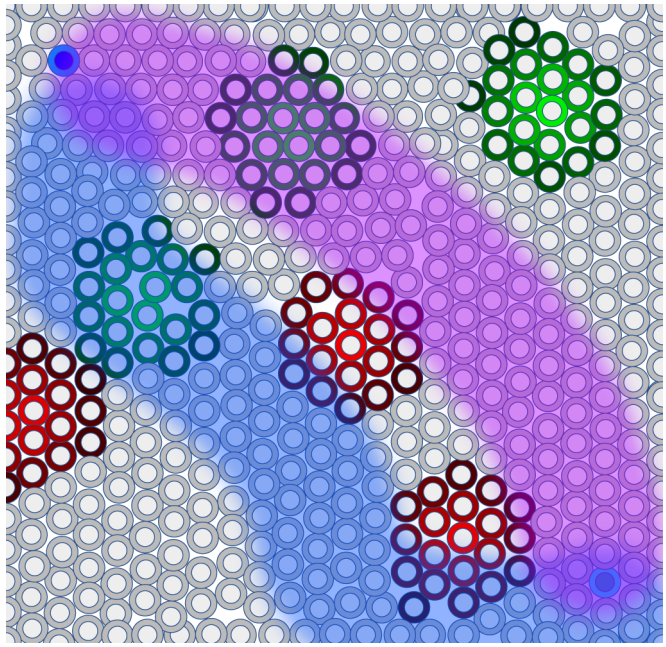


Figure 6.22: Radio alternatives with stop points and candidates

must not overestimate the distance to the goal. Thus, for an application like routing, $h(x)$ might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

If the heuristic h satisfies the additional condition $h(x) \leq d(x, y) + h(y)$ for every edge (x, y) of the graph (where d denotes the length of that edge), then h is called monotone, or consistent. In such a case, A* can be implemented more efficiently-roughly speaking, no node needs to be processed more than once and A* is equivalent to running Dijkstra's algorithm with the reduced cost $d'(x, y) := d(x, y) + h(y) - h(x)$.

Like all informed search algorithms, it first searches the routes that appear to be most likely to lead towards the goal. What sets A* apart from a greedy best-first search is that it also takes the distance already traveled into account; the $g(x)$ part of the heuristic is the cost from the starting point, not simply the local cost from the previously expanded node.

Starting with the initial node, it maintains a priority queue of nodes to be traversed, known as the open set or fringe. The lower $f(x)$ for a given node x, the higher its priority. At each step of the algorithm, the node with the lowest $f(x)$ value is removed from the queue, the f and g values of its neighbors are updated accordingly, and these neighbors are added to the queue. The algorithm continues until a goal node has a lower f value than any node in the queue (or until the queue is empty). (Goal nodes may be passed over multiple times if there remain other nodes with lower f values, as they may lead to a shorter path to a goal.) The f value of the goal is then the length

of the shortest path, since h at the goal is zero in an admissible heuristic.

The algorithm described so far gives us only the length of the shortest path. To find the actual sequence of steps, the algorithm can be easily revised so that each node on the path keeps track of its predecessor. After this algorithm is run, the ending node will point to its predecessor, and so on, until some node's predecessor is the start node. Additionally, if the heuristic is monotonic (or consistent), a closed set of nodes already traversed may be used to make the search more efficient.

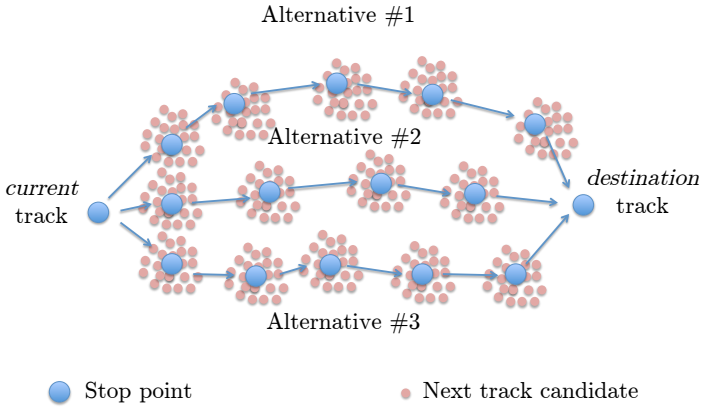


Figure 6.23: Radio alternatives with stop points and candidates

At every A* iteration, the radio journey algorithm uses the total distance measured by the Floyd-Warshall algorithm and the track closeness to a positively or negatively scored track to determine the heuristics and select the closest path. At the end of the A* algorithm, a sequence of tracks is obtained. The journey alternative is constructed by adding the sequence of tracks from the path-finding algorithm and also the most similar tracks to the tracks in the sequence. All the tracks in this alternative are removed from the original graph and the A* is executed again to obtain another alternative. At the end of the process, there are some alternatives to transverse the graph from the origin track to the destination track. Fig. fig:dij14 shows two alternatives track sets that traverse the graph trying to avoid negatively scored tracks and trying to pass near positively scored tracks.

6.7.4 Alternative reduction to playlist

Each alternative is basically a sub-set of tracks that would produce a smooth transition from the planned radio journey. However the number of tracks inside each alternative is hundreds time bigger than the number of tracks to be proposed in the playlist and a selection of the optimal tracks is necessary. The alternatives are reduced independently and will be presented to the user as high level selections. The first alternative calculated by the A* algorithm is the one that will be played and the rest will be presented to the user to give him/her time to see the cover arts, artist names and track titles and change course if required.

The tracks of an alternative are organized in a graph in the similar manner as with the initial global graph and a Dikjstra algorithm is applied to calculate the overall distances between all tracks and origin and destination tracks. Tracks are sorted using the following score:

$$score_i = \frac{Dist(origin, track_i)}{Dist(origin, track_i) + Dist(track_i, destination)} \tag{6.107}$$

As depicted in Fig. 6.24, the score is divided into the number of tracks pending to be played and the closest tracks to each *stop point* is selected. Then the tracks in the alternative set will be grouped by the proximity to the stop points selected tracks. The final task at this point is to select which track present as a candidate for each stop point.

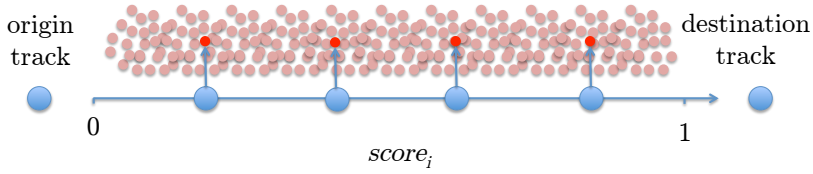


Figure 6.24: Selection of stop-points using distance score

6.7.5 Immunology approach to final selection

The current radio journey algorithm has already taken into consideration scored tracks to change the trajectory over the space and the tracks in each stop point should at this point be less similar to negatively scored tracks and more similar to positively scored tracks. The immunology algorithm is completely orthogonal to the Riemann metric selection used for graph distance and cab be very efficiently applied here. The radio algorithm maintains a database of scored millisounds. Basically, each time the user scores a track, the already pre-calculated millisounds of the track are added to the user millisound profile together with their scores. If a millisound is already in the profile (detected using a simple Euclidean metric of the normalized parameters of the millisound) then it's score is incremented or decremented based on the track score. If the millisound is not in the profile, it is simply added with its initial score.

All the tracks contained in each stop point are evaluated using the 4th layer Immunology approach described in Chapter 5. The millisounds of each track are compared to the millisounds of the user profile and the track with the higher number of positive matches is selected to be the representative of the stop point.

The structure of the algorithm is defined as follows:

Data: User millisound profile, scored tracks (like, dislike)
Data: Origin and destination track
Result: Next tracks to be played to achieve destination, alternative journeys
Use propagated popularity to select the initial track candidates.
Use Riemann metric estimation to fill the graph matrix with tracks and similar tracks;
Use Floyd-Warshall algorithm to calculate the distance between all tracks;
while *not enough alternatives* **do**
    Use A* algorithm to find a trajectory based on music similarity and closeness to scored tracks;
    Build alternative with similar tracks to the tracks in the trajectory;
    Remove the selected tracks from the initial track candidates;
end
foreach *alternative* **do**
    foreach *stop-point* **do**
        Use Millisounds from scored tracks as antigens for the candidates;
        Remove worst candidates by using Artificial Immunology method;
    end
end

6.7.6 Summary

In this chapter, a novel method to propagate track popularity has been presented. A normal catalog consists on a small fraction of the collection being very popular and with very accurate ranking (from album sales, downloads, etc.) while the majority of the catalog (the Long-Tail) remains with very little visibility and inaccurate ranking. The proposed method uses a combination of the Chess game ranking methodology (Elo score) with similarity techniques to propagate the *hitness* for tracks without ranking information. At the same time, the algorithm makes sure that the distribution characteristics of the Long-Tail are maintained.

Finally and as main conclusion of this disseration, a novel radio algorithm is described. User is able to select a destination track and the algorithm proposes a sequence of tracks to be played that maximizes the smoothness of the transitions. This algorithm combines the propagated popularity with the curvature metric estimation described in Chapter 4 and the Artificial Immunology screening method described in Chapter 5 to promote the tracks that are more similar to tracks that the user has scored positively and more different to those negatively scored tracks.

Year	Vinyl	Cassette	CD	Video	Downloads	Subs.	On-Dem.
1983	51.7%	47.8%	0.5%	-	-	-	-
1984	42.6%	55%	2.4%	-	-	-	-
1985	35.8%	55.3%	8.9%	-	-	-	-
1986	26.1%	53.9%	20%	-	-	-	-
1987	18%	53.5%	28.6%	-	-	-	-
1988	11.4%	55%	33.4%	-	-	-	-
1989	5.1%	53.8%	39.3%	1.8%	-	-	-
1990	2.4%	49.4%	45.9%	2.3%	-	-	-
1991	1.2%	41.44%	55.8%	1.5%	-	-	-
1992	0.8%	37.8%	59.5%	1.7%	-	-	-
1993	0.6%	32%	65.3%	2.1%	-	-	-
1994	0.5%	27%	70.6%	1.9%	-	-	-
1995	0.6%	20.6%	77%	1.8%	-	-	-
1996	0.7%	16.7%	80.8%	1.9%	-	-	-
1997	0.6%	13.5%	83.2%	2.6%	-	-	-
1998	0.4%	11.1%	84.9%	3.7%	-	-	-
1999	0.4%	7.6%	89.4%	2.6%	-	-	-
2000	0.4%	4.4%	93.3%	2%	-	-	-
2001	0.4%	2.6%	94.5%	2.4%	-	-	-
2002	0.4%	0.9%	95.1%	3.4%	-	-	-
2003	0.4%	1.7%	95.7%	2.3%	-	-	
2004	0.4%	0.2%	92.8%	4.9%	1.5%	-	
2005	0.2%	0.1%	85.7%	4.9%	4.4%	0.1%	
2006	0.2%	-	79.8%	3.8%	7.2%	1.8%	-
2007	0.2%	-	70.1%	4.6%	12.3	2.2%	-
2008	0.6%	-	62.3%	2.6%	19%	2.5%	-
2009	0.8%	-	55.2%	2.7%	24.5%	2.6%	-
2010	1.3%	-	48.3%	2.5%	31.5%	3%	-
2011	1.8%	-	43.5%	2.71	36.3%	3.5%	0.8%
2012	2.4%	-	35.4%	1.7%	40.3%	5.7%	2.4%
2013	3%	-	30.4%	1.5%	40%	9%	3.3%

Table 6.1: US-based recordings. Data supplied by the Recording Industry of America (RIAA)

Chapter 7

Conclusions and future work

7.1 Human sensor bandwidth

If we want to understand how the human brain works we need to find doors that can let us in. We could try using the human vision, one of our most incredible capabilities but there is big computation problem. First it is estimated that the total pixel resolution is about 560 Megapixels. Assuming a frame rate of 25 captures per second and three bytes per pixel (red, green and blue byes) approximately, it gives a 41 GBytes per second of information. Just to store all the information processed in one day we would need more than 2300 TBytes (16 hours). Maybe in near future, computers may process such a huge amount of data very fast but to give an example just to read all this information with a powerful computer would requires more than a year. So trying to understand how the brain works with the *human vision* door is more complex than expected. What happens if we try with human auditory sensors?

An average human can hear sounds up to 22 kHz. The Nyquist theorem postulates that to sample such a sound, only a sampling frequency of 44 kHz is needed. Considering two ears and 16 bits per sample we would only need 9 GB of data to store all the sounds listened by a human during a day (16 hours). From a computation point of view, it is feasible. Audio compression techniques have reached a 10 times compression of the audio signal without a noticeable loss of quality so storing all the sounds received by a 30 year old person would only require 10TBytes of data. Therefore, trying to enter the human brain by the auditory door is at least, nowadays, not practically impossible.

7.1.1 Machine training

From many points of view, a human can be considered as a very complex system. This system receives inputs (capturing images, listening to sounds, touching things, feeling temperature) and produces outputs (moving muscles, heart beat, etc.). It can also be modeled as complex machine that starts with a cell and some initial building instructions (DNA). The machines grows by reproducing its cells and the inputs gathered by the sensors train the machine itself to perform other tasks. From many points of view, it is a machine designed to build many other machines (organs, brain, skin, etc.) and is also designed to interfere with the environment during its building process so inputs to the system are as important as the instructions. The machine is constantly in a training process, adapting itself to the changing external conditions in order to maximize the survival rate. The brain is an extremely powerful machine that cannot work without initial training. Even after the basic training is finished, it accepts nicely constant training as it has been demonstrated through evolution that it can help survive.

Why does a dog like to play to the throw and catch game? Why do we tend to like to walk and listen to music at the same time? We have a theory that the answer to these questions is basically the same. After that, we can try answer to the question what is music for the human brain. The dog likes so much that game because - and this is this dissertation author opinion - developing hunting capabilities was a huge survival requirement in dog ancestor original habitat. The dog is forced to: first make a prediction of the trajectory of the thrown object, then the dog has to run and therefore develop its muscles and tendons and finally it has to find the object and catch it with precision. From a hunting point of view, these actions are absolutely necessary and the evolution has linked pleasure to the training of the hunting activity for dogs. Because when it comes to the *dog real world*, to know how to hunt makes the difference and training by playing has been selected by evolution as the best training strategy.

The human brain is able of many incredible things from an image processing point of view. For example, just by watching five seconds of a movie the human brain has a very high probability to decide whether the movie was already seen or not. The human vision locates patterns and compares them to the stored patterns to make decisions and we probably are only aware of a very tiny part of the total calculations performed to understand a single visual frame. To achieve such a performance, our brain has an incredible capacity of parallel calculation and this is the key for pattern matching. A dog has to understand the visual environment very precisely to be able to move, run, locate other animals and threads, etc. Its visual brain is therefore similar - from a performance point of view - to our brain. A dog is also able to hear and produce sounds (at different frequencies) but from an Information theory point of view, the bandwidth of the input data for a human and a dog is equivalent. Dogs can be trained to obey to some sounds and are able to use their hearing capabilities to hunt efficiently. Can a dog listen to music then?

The ability of identifying patterns and reproducing them using the voice was one of the key aspects of human evolution. Training to identify these patterns was crucial for the future ability of the human to communicate with other human beings. Speech capabilities were boosted by evolution, basically pattern identification and pattern

reproduction. The *human machine* was able to train its auditory pattern identification systems just by listening to other humans producing sounds and is even able to auto-train itself just by using other sensors to link actions with sounds. So even before a baby is born, its *brain* machine is storing patterns that will be used later to identify future sounds. And exactly the same way a dog finds brain-faked pleasure by training to hunt by playing, a very young human finds pleasure by training the auditory system. Of course, not all types of sounds give pleasure. For example, constant sounds like a tone give the opposite pleasure as well as completely random sounds. But in the middle, there is a trade-off between prediction and innovation that can be viewed as an optimal point for auditory system training.

7.2 Music

Some people think that music is universal. The author considers that enjoying music is a *bonus* feature that our brain gives us but the auditory subsystem was not designed for music at all. Evolution boosted our ability to do sound pattern extraction and sound matching to improve communication and this ability is so powerful that has been able to find pleasure with music. Because music is about sounds with rhythm, harmony and melody. Our brain finds pleasure with some slowly varying rhythms because a rhythmic sound can be predicted easier than a non rhythmic sound. Harmonic sounds are a very special type of sounds where the presence of harmonic tones (at multiple frequencies) can also be predicted or expected. Finally, melody imposes that some sounds are more likely to be listened next than others because they provide a better classification. For this reason, listen to music offers to the human auditory brain an optimal training environment and pleasure can be given back. We tend to enjoy walking and listening to music because both our visual brain and our auditory brain have to be in *pattern* matching mode and depending on the trade-off between predictability and innovation of the music patterns and the visual patterns pleasure can be given back.

In other words, music can help us in the quest of trying to understand how our brain works internally. It is an incredible key to that door. There are several types of sound illusions that give answers to many questions regarding the signal processing inside the auditory system. For example, the Shepard tones are a superposition of sine waves separated by octaves that create the auditory illusion of a tone constantly ascending or descending in pitch. The explanation of such an illusion is that the auditory system has a specific transference function which determines the sensitivity as a function of the input frequency, as all the engineered transducers.

7.3 What is music and what is not

We might think that it is easy to decide when listening to some sounds if it is music or not. A more precise answer is that depending on the sound, it can be very easy or it can be very difficult to define a frontier of what is music and what is not. It also has a very significant dependency on the culture of the person listening to that

sound. Some bird sounds are harmonic and can even be in scale. A cat yell is surely not music unless we collect several cat yells and put them in a melody. So from our point of view, music needs intention to create it and this is what we need to look for when we try to answer to that question. A music sound has to create an environment were the trade-off between predictability and innovation is controlled with intention. If you make a computer play the piano in a purely random way, when you listened to the output sound, you will find harmony rhythm and notes will be on scale, but you will doubt whether is music or not because there is no intention in the melody. As soon as you modify the random way by a very simple melody, the answer to the question turns to: *it's music, but simple music* because there was intention to make it more predictable than purely random.

7.4 Unsupervised music classification

Someone that is used to listen to pop music can easily decide whether two songs that he/she listens for the first time are similar or not. A more practical scenario is presenting to the user three songs and asking the user to decide which pair of them are more similar. The core of the recommendation algorithms presented in this dissertation is to try to extract useful information from these three-songs scenarios and transform it into knowledge. The goal of such extraction of information is to be able to apply that knowledge without human interaction when a unknown song is presented to the system.

7.5 Future of music recommendation - long tail boost

If we put all the available music on a graph and rank each song in order of popularity what you would find is that most people listen to only a small proportion of the available music and after this you have millions of really good songs only listened to by a few people. On our graph this line of lesser listened to songs stretches almost to infinity. This is called the long tail.

This has occurred mainly because of the hegemony of the major record labels that have been able through total control and conditioning focus the consumer's attention on only those songs that they, through money and promotion have convinced them to buy. The Labels have simply not had the resources to promote and publish everyone. This has left a few artists earning mega bucks and millions more barely staying alive even though their music warrants more. How many times have you heard the phrase *in the record business it is not what you know, but who you know.*

Now that music distribution has gone on-line all the ground rules have changed. You can now buy song by song rather than album by album. You can access sites with song streaming, music on-demand as well as buying songs.

Taking the example of those services that let you listen to music for a fee, the vast majority are facing the crude reality that their software for various technical reasons only really lets you listen to the songs at the fat end of the long tail, the so called hits. However as this gets quite boring over time all the on-line services are trying to find an easy way of allowing their customers to discover new music as well as listening to hits. Anyone that can do this, it is believed by the world's top experts, will change the world of music forever as well as making a great deal of money in the process. Well we know for example that the lower end of our demographic from 10 to 28 years really like to listen to new music, whilst at the top end 30 to 60 years will look more for hits and the music they know and love. However, even at this end of the demographic users will move towards discovery of different sounds if this is easy to do and with our patented Music Universe discovery tool it does not get much easier.

The monetary effect on our business apart from winning new users, earning increasingly from advertising revenues, music sales, on-line store and merchandising features and subscriptions is cost saving. For many reasons the cost of music rights from the mainstream labels at the fat end of the tail are much more expensive than those in the long tail. Therefore by leading consumers to the richer music experience to be found by mixing, we will reduce dramatically the cost of music (our biggest potential cost) to our business and therefore make more profits. Monetizing the Long Tail is therefore one of the most important factors for the survival and profitability of on-line music. Many believe that within 15 years there could be 90% of users listening to music from the long tail and only 10% listening to mainstream hits. When this occurs, as it undoubtedly will, the music revolution will have happened and the democratization of the music industry will have occurred.

Appendix A

A.1 Vorbis audio compression

Ogg Vorbis is a fully open, non-proprietary, patent-and-royalty-free, general-purpose compressed audio format for mid to high quality (8kHz-48.0kHz, 16+ bit, polyphonic) audio and music at fixed and variable bitrates from 16 to 128 kbps/channel. This places Vorbis in the same competitive class as audio representations such as MPEG-4 (AAC), and similar to, but higher performance than MPEG-1/2 audio layer 3, MPEG-4 audio (TwinVQ), WMA and PAC.

The bitstream format for Vorbis I was frozen Monday, May 8th 2000. All bitstreams encoded since will remain compatible with all future releases of Vorbis.

A.1.1 Fidelity measurement and terminology discussion

Terminology discussed in this document is based on common terminology associated with contemporary codecs such as MPEG I audio layer 3 (mp3). However, some differences in terminology are useful in the context of Vorbis as Vorbis functions somewhat differently than most current formats. For clarity, then, we describe a common terminology for discussion of Vorbis's and other formats' audio quality.

Objective fidelity is a measure, based on a computable, mechanical metric, of how carefully an output matches an input. For example, a stereo amplifier may claim to introduce less that .01 % total harmonic distortion when amplifying an input signal; this claim is easy to verify given proper equipment, and any number of testers are likely to arrive at the same, exact results. One need not listen to the equipment to make this measurement.

However, given two amplifiers with identical, verifiable objective specifications, listeners may strongly prefer the sound quality of one over the other. This is actually the case in the decades old debate [some would say jihad] among audiophiles involving

vacuum tube versus solid state amplifiers. There are people who can tell the difference, and strongly prefer one over the other despite seemingly identical, measurable quality. This preference is subjective and difficult to measure but nonetheless real.

Individual elements of subjective differences often can be qualified, but overall subjective quality generally is not measurable. Different observers are likely to disagree on the exact results of a subjective test as each observer's perspective differs. When measuring subjective qualities, the best one can hope for is average, empirical results that show statistical significance across a group.

Perceptual codecs are most concerned with subjective, not objective, quality. This is why evaluating a perceptual codec via distortion measures and sonograms alone is useless; these objective measures may provide insight into the quality or functioning of a codec, but cannot answer the much squishier subjective question, *Does it sound good?*. The tube amplifier example is perhaps not the best as very few people can hear, or care to hear, the minute differences between tubes and transistors, whereas the subjective differences in perceptual codecs tend to be quite large even when objective differences are not.

A.1.2 Fidelity, Artifacts and Differences

Audio artifacts and loss of fidelity or more simply put, audio differences are not the same thing.

A loss of fidelity implies differences between the perceived input and output signal; it does not necessarily imply that the differences in output are displeasing or that the output sounds poor (although this is often the case). Tube amplifiers are not higher fidelity than modern solid state and digital systems. They simply produce a form of distortion and coloring that is either unnoticeable or actually pleasing to many ears.

As compared to an original signal using hard metrics, all perceptual codecs [ASPEC, ATRAC, MP3, WMA, AAC, TwinVQ, AC3 and Vorbis included] lose objective fidelity in order to reduce bitrate. This is fact. The idea is to lose fidelity in ways that cannot be perceived. However, most current streaming applications demand bitrates lower than what can be achieved by sacrificing only objective fidelity; this is also fact, despite whatever various company press releases might claim. Subjective fidelity eventually must suffer in one way or another.

The goal is to choose the best possible tradeoff such that the fidelity loss is graceful and not obviously noticeable. Most listeners of FM radio do not realize how much lower fidelity that medium is as compared to compact discs or DAT. However, when compared directly to source material, the difference is obvious. A cassette tape is lower fidelity still, and yet the degradation, relatively speaking, is graceful and generally easy not to notice. Compare this graceful loss of quality to an average 44.1kHz stereo mp3 encoded at 80 or 96kbps. The mp3 might actually be higher objective fidelity but subjectively sounds much worse.

Thus, when a CODEC must sacrifice subjective quality in order to satisfy a user's requirements, the result should be a difference that is generally either difficult to notice without comparison, or easy to ignore. An artifact, on the other hand, is an element introduced into the output that is immediately noticeable, obviously foreign, and undesired. The famous *underwater* or *twinkling* effect synonymous with low bitrate (or poorly encoded) mp3 is an example of an artifact. This working definition differs slightly from common usage, but the coined distinction between differences and artifacts is useful for our discussion.

The goal, when it is absolutely necessary to sacrifice subjective fidelity, is obviously to strive for differences and not artifacts. The vast majority of codecs today fail at this task miserably, predictably, and regularly in one way or another. Avoiding such failures when it is necessary to sacrifice subjective quality is a fundamental design objective of Vorbis and that objective is reflected in Vorbis's design and tuning.

A.1.3 Decode Setup

Before decoding can begin, a decoder must initialize using the bitstream headers matching the stream to be decoded. Vorbis uses three header packets; all are required, in-order, by this specification. Once set up, decode may begin at any audio packet belonging to the Vorbis stream. In Vorbis I, all packets after the three initial headers are audio packets.

The header packets are, in order, the identification header, the comments header, and the setup header. Identification Header The identification header identifies the bitstream as Vorbis, Vorbis version, and the simple audio characteristics of the stream such as sample rate and number of channels. Comment Header The comment header includes user text comments (*tags*) and a vendor string for the application/library that produced the bitstream. The encoding and proper use of the comment header is described in section 5, *comment field and header specification*. Setup Header The setup header includes extensive CODEC setup information as well as the complete VQ and Huffman codebooks needed for decode.

A.1.4 Decode Procedure

The decoding and synthesis procedure for all audio packets is fundamentally the same.

- decode packet type flag
- decode mode number
- decode window shape (long windows only)
- decode floor
- decode residue into residue vectors
- inverse channel coupling of residue vectors

- generate floor curve from decoded floor data
- compute dot product of floor and residue, producing audio spectrum vector
- inverse monolithic transform of audio spectrum vector, always an MDCT in Vorbis I
- overlap/add left-hand output of transform with right-hand output of previous frame
- store right hand-data from transform of current frame for future lapping

Note that clever rearrangement of the synthesis arithmetic is possible; as an example, one can take advantage of symmetries in the MDCT to store the right-hand transform data of a partial MDCT for a 5001% inter-frame buffer space savings, and then complete the transform later before overlap/add with the next frame. This optimization produces entirely equivalent output and is naturally perfectly legal. The decoder must be entirely mathematically equivalent to the specification, it need not be a literal semantic implementation. Packet type decode Vorbis I uses four packet types. The first three packet types mark each of the three Vorbis headers described above. The fourth packet type marks an audio packet. All other packet types are reserved; packets marked with a reserved type should be ignored.

Following the three header packets, all packets in a Vorbis I stream are audio. The first step of audio packet decode is to read and verify the packet type; a non-audio packet when audio is expected indicates stream corruption or a non-compliant stream. The decoder must ignore the packet and not attempt decoding it to audio. Mode decode Vorbis allows an encoder to set up multiple, numbered packet *modes*, as described earlier, all of which may be used in a given Vorbis stream. The mode is encoded as an integer used as a direct offset into the mode instance index.

Window shape decode (long windows only) Vorbis frames may be one of two PCM sample sizes specified during codec setup. In Vorbis I, legal frame sizes are powers of two from 64 to 8192 samples. Aside from coupling, Vorbis handles channels as independent vectors and these frame sizes are in samples per channel.

Vorbis uses an overlapping transform, namely the MDCT, to blend one frame into the next, avoiding most inter-frame block boundary artifacts. The MDCT output of one frame is windowed according to MDCT requirements, overlapped 50% with the output of the previous frame and added. The window shape assures seamless reconstruction.

In the unequal-sized window case, the window shape of the long window must be modified for seamless lapping as above. It is possible to correctly infer window shape to be applied to the current window from knowing the sizes of the current, previous and next window. It is legal for a decoder to use this method. However, in the case of a long window (short windows require no modification), Vorbis also codes two flag bits to specify pre- and post- window shape. Although not strictly necessary for function, this minor redundancy allows a packet to be fully decoded to the point of lapping entirely independently of any other packet, allowing easier abstraction of decode layers as well as allowing a greater level of easy parallelism in encode and decode. Vorbis windows all use the slope function

$$\mathrm{y} = \sin\left(\frac{\pi}{2}\sin^2\left(\frac{x+\frac{1}{2}}{\pi n}\right)\right) \tag{A.1}$$

A.1.5 Floor decode

Each floor is encoded/decoded in channel order, however each floor belongs to a *submap* that specifies which floor configuration to use. All floors are decoded before residue decode begins.

A.1.6 Residue decode

Although the number of residue vectors equals the number of channels, channel coupling may mean that the raw residue vectors extracted during decode do not map directly to specific channels. When channel coupling is in use, some vectors will correspond to coupled magnitude or angle. The coupling relationships are described in the codec setup and may differ from frame to frame, due to different mode numbers. Vorbis codes residue vectors in groups by submap; the coding is done in submap order from submap 0 through $n-1$. This differs from floors which are coded using a configuration provided by submap number, but are coded individually in channel order.

A.1.7 Inverse channel coupling

A detailed discussion of stereo in the Vorbis codec can be found in the document Stereo Channel Coupling in the Vorbis CODEC. Vorbis is not limited to only stereo coupling, but the stereo document also gives a good overview of the generic coupling mechanism.

Vorbis coupling applies to pairs of residue vectors at a time; decoupling is done in-place a pair at a time in the order and using the vectors specified in the current mapping configuration. The decoupling operation is the same for all pairs, converting square polar representation (where one vector is magnitude and the second angle) back to Cartesian representation. After decoupling, in order, each pair of vectors on the coupling list, the resulting residue vectors represent the fine spectral detail of each output channel.

A.1.8 Generate floor curve

The decoder may choose to generate the floor curve at any appropriate time. It is reasonable to generate the output curve when the floor data is decoded from the raw packet, or it can be generated after inverse coupling and applied to the spectral residue

directly, combining generation and the dot product into one step and eliminating some working space.

Both floor 0 and floor 1 generate a linear-range, linear-domain output vector to be multiplied (dot product) by the linear-range, linear-domain spectral residue.

A.1.9 Compute floor/residue dot product

This step is straightforward; for each output channel, the decoder multiplies the floor curve and residue vectors element by element, producing the finished audio spectrum of each channel. One point is worth mentioning about this dot product; a common mistake in a fixed point implementation might be to assume that a 32 bit fixed-point representation for floor and residue and direct multiplication of the vectors is sufficient for acceptable spectral depth in all cases because it happens to mostly work with the current Xiph.Org reference encoder. However, floor vector values can span 140dB (24 bits unsigned), and the audio spectrum vector should represent a minimum of 120dB (21 bits with sign), even when output is to a 16 bit PCM device. For the residue vector to represent full scale if the floor is nailed to -140dB, it must be able to span 0 to +140dB. For the residue vector to reach full scale if the floor is nailed at 0dB, it must be able to represent -140dB to +0dB. Thus, in order to handle full range dynamics, a residue vector may span -140dB to +140dB entirely within spec. A 280dB range is approximately 48 bits with sign; thus the residue vector must be able to represent a 48 bit range and the dot product must be able to handle an effective 48 bit times 24 bit multiplication. This range may be achieved using large (64 bit or larger) integers, or implementing a movable binary point representation.

A.1.10 Inverse monolithic transform (MDCT)

The audio spectrum is converted back into time domain PCM audio via an inverse Modified Discrete Cosine Transform (MDCT). Note that the PCM produced directly from the MDCT is not yet finished audio; it must be lapped with surrounding frames using an appropriate window (such as the Vorbis window) before the MDCT can be considered orthogonal.

A.1.11 Overlap/add data

Windowed MDCT output is overlapped and added with the right hand data of the previous window such that the 3/4 point of the previous window is aligned with the 1/4 point of the current window (as illustrated in the window overlap diagram). At this point, the audio data between the center of the previous frame and the center of the current frame is now finished and ready to be returned.

A.1.12 Cache right hand data

The decoder must cache the right hand portion of the current frame to be lapped with the left hand portion of the next frame.

A.1.13 Return finished audio data

The overlapped portion produced from overlapping the previous and current frame data is finished data to be returned by the decoder. This data spans from the center of the previous window to the center of the current window. In the case of same-sized windows, the amount of data to return is one-half block consisting of and only of the overlapped portions. When overlapping a short and long window, much of the returned range is not actually overlap. This does not damage transform orthogonality.

Data is not returned from the first frame; it must be used to *prime* the decode engine. The encoder accounts for this priming when calculating PCM offsets; after the first frame, the proper PCM output offset is *0* (as no data has been returned yet).

Appendix B

B.1 MELO Algorithm in Matlab

```
%% VALUE PROPAGATION
function ValuePropagation

Area=260000; % Suma de todos los MELO tiene siempre este valor
numSongs=80000; % 10 veces menos que Raphsody
E=0.424; % Long Tail exponent parameter. Sin escala
%E=0.400; % Long Tail exponent parameter (Shorter). Experimentación
%E=0.450; % Long Tail exponent parameter (Longer). Experimentación
R=48.0; % Este valor debe ser 600 para un número total de 1000000.
% Es decir R=600*numSongs/1000000;

% * La información sobre canciones se guarda en un array numSongs x 6 con
% la siguiente interpretación
% * song(i,1)=x   song(i,2)=y Identificador
% * song(i,3)=melo   song(i,4)=log(melo)
% * song(i,5)=xcelda  song(i,6)=ycelda

 song=zeros(numSongs,6);

 %% LOCAL MELO APPROXIMATION
 %
 % * En esta simulación el plano (x,y) es un caudrado 1000x1000. La
 % constante que determina el número de celdas es la longitud NG de cada
 % una de ellas en pixels. Por tanto si NG=10 hay 100*100=10000 celdas en
 % total con lo que el número promedio de canciones por celda es
 % 80000/10000=8. Obviamente esta elección es ampliamente arbitraria y el
 % concepto general equivalente sería la selección del número total de
 % clusters

NG=10; % Longitud del lado del cuadrado del grid en unidades de handles.ax

%% VALUE PROPAGATION TOURNAMENTS
%
```

```
W=50; % Un torneo actúa en una ventana de ancho 2*W+1 canciones
% He probado desde W=30 hasta hasta W=1000
numTemp=50; % Máximo número de temporadas
numTorneos=100; % Número de torneos por temporada


end


%% EVOLUTION CODIGO MATLAB (incompleto)
%
% * Suponemos aquí que todo el proceso de inicialización ha sido
% desarrollado. El que se aplicaba en la simulación es totalmente
% arbitrario. Generaba posiciones aleatorias x,y como identificadores de
% las canciones. Declaraba como hits unas 100 canciones a las que asignaba
% melos en una banda alrededor de maxval. Al resto de canciones se les
% asignaba un melo mediante el decay exponencial descrito antes. Los
% resultados se registraban en la estructura *song*

function Evolucion(song,numSongs,rank,W,numTemp,numTorneos,NG,E,R,Area)

%%% Estructuras
%
% * grid(n,m,i) da el índice de la canción i—sima de la celda (n,m)
% * meloc(n,m) da el melo total de la celda (n,m)
% * nsongcell(n,m) da el número de canciones de la celda (n,m)
% El melo inicialmente se toma de la valoración DJ
% Inicialización de la estructura grid
numxycells=int32(1000/NG);
numcells=numxycells^2;
numsongsxcell=int32(numSongs/numcells);
grid=int32(zeros(numxycells,numxycells,numsongsxcell));
meloc=zeros(numxycells,numxycells);% Total MELO in cell
nsongcell=int32(zeros(numxycells,numxycells));
% Initially MELO is identical with valDJ and contained in song(:,3)
for s=1:numSongs
    x=song(s,1);
    y=song(s,2);
    n=int32(x/NG+0.5);
    m=int32(y/NG+0.5);
    song(s,5)=n; % Putting cell coordinates n,m into memory
    song(s,6)=m;
    nsongcell(n,m)=nsongcell(n,m)+1;
    grid(n,m,nsongcell(n,m))=s;
    meloc(n,m)=meloc(n,m)+song(s,3);
end;
%%% EVOLUCION
%
% * Un torneo se selecciona mediante la determinación del rango central
de
% la ventana: r. Los 2W+1 elementos de la ventana son r—W:r+W.
% * Cada temporada se juegan numTorneos y al final de cada temporada se
% referesca la gráfica representando el MELO vigente
avemel=zeros(2*W+1,1); % Contendrá el meloc de cada item de la ventana
% song(s,4) contiene el logaritmo del melo song(s,3) de la canción s
% En este estado inicial song(s,3) contiene la valoración DJ que no tiene
% mucho sentido para canciones lejanas a los hit. Por ello adoptamos una
```

```
% estrategia con umbral de protección. Si song(s,3) es > umbral la
% consideramos decente como un punto inicial y ponemos su logaritmo en
% song(s,4). Para todo el resto ponemos el mismo valor. Experimentalmente
% no parece depender del umbral
umbral=1.0;
for s=1:numSongs
    if(song(s,3)>umbral)
        x=log(song(s,3));
    else
        x=log(umbral);
    end;
    song(s,4)=x;
end;
song(:,4)=log(song(:,3));
for temp=1:numTemp
    % Inicio de temporada
    display(sprintf('Ejecutando temporada %d...',temp));
    tini=tic;% Control de tiempo por temporada
    for t=1:numTorneos
        % Selección del punto inicial de la ventana. El mínimo es W+1, el
        % máximo es numSongs-W. Recordemos que la ventana tiene sus items
        % en el rango r-W-1+k   k=1,2,...,2*W+1
        r=randi([W+2,numSongs-W-1],1);
        rwin=r-W:r+W; % Indices de la ventana. Recordemos que son rangos
        %dentro del orden vigente. Los índices de las canciones en song son
        swin=rank(rwin);
        for k=1:2*W+1
           s=swin(k);
           avemel(k)=meloc(song(s,5),song(s,6));
        end;
        [~, owin]=sort(avemel,'descend');
        % owin(i) is the value in 1:2*W+1 of the i-th most valuable song so
        % swin(owin(i)) is the corresponding song's index
        % Now let us compute window's melo according to the power law with
        % m1=1
        % mel(owin(i+1))=mel(owin(i))*(1-(E+1)/(R+r-W+i-1)) i=1,...2W
        % The new rank inside the window is:
        swin=swin(owin);
        % Ahora swin(1) es el índice en song de la nueva primera canción
        rank(r-W:r+W)=swin;% Nueva ordenación pasa a rank

        if(r>0) % Continuidad en la cabeza de la ventana
             % El primero por continuidad
            song(swin(1),4)=song(rank(r-W-1),4)+log((1-(E+1)/(R+r-W-1)));
            for i=1:2*W
                song(swin(i+1),4)=song(swin(i),4)+log((1-(E+1)/(R+r-W-1+i)));
            end;
            % Postwindow
            f1=1;
            if(r+W)<numSongs
                s1=rank(r+W+1);
                % Continuity factor for items after the window
                f1=song(swin(2*W+1),4)+log((1-(E+1)/(R+r+W)))-song(s1,4);
            end;
            for i=r+W+1:numSongs
                song(rank(i),4)=f1+song(rank(i),4);
            end;
```

```
        else % Continuidad con la cola
            % Prewindow
            f0=1;
            if(r-W)>1
                % Continuity factor for items before the window
                f0=-song(rank(r-W-1),4);
            end;
            for i=1:r-W-1
                song(rank(i),4)=f0+song(rank(i),4);
            end;
            % El último de la ventana se calcula por continuidad
            song(swin(2*W+1),4)=song(rank(r+W+1),4)-log((1-(E+1)/(R+r+W)));
            for i=2*W:-1:1
                song(swin(i),4)=song(swin(i+1),4)-log((1-(E+1)/(R+r-W-1+i)));
            end;
        end;
       % Exponenciamos los logaritmos con protección de tamaño
        maxlog=8;minlog=-10;
        maxmelo=exp(maxlog); % Selección de máximo valor aproximadamente 3000
        minmelo=exp(minlog);
        for s=1:numSongs
            % Protección de valores excesivamente pequeños o grandes
            if(song(s,4)>maxlog)
                song(s,4)=maxlog;song(s,3)=maxmelo;
            elseif(song(s,4)<minlog)
                song(s,4)=minlog;song(s,3)=minmelo;
            else
                song(s,3)=exp(song(s,4));
            end;
        end;
         % Finalmente normalizamos
        factor=Area/sum(song(:,3));
        song(:,3)=factor*song(:,3);
        song(:,4)=log(song(:,3));
    end;
    % Final de temporada
    display(sprintf('final temporada %d : %1.3f',temp,toc(tini)));
    Paint(handles,100*sort(song(:,3),'descend'));

    %Recomputing of melo in cells (recálculo sólo al final de cada temporada)
    meloc=zeros(numxycells,numxycells);% All local melos to 0
    for s=1:numSongs
        % Coordenadas de la canción
        x=song(s,1);
        y=song(s,2);
        % Celda en que se encuentra la canción
        n=int32(x/NG+0.5);
        m=int32(y/NG+0.5);
        % Acumulación del nuevo melo local
        meloc(n,m)=meloc(n,m)+song(s,3);
    end;
    % Next temporada
end;

end
%
%% MATLAB
```

```
%
% * La función [Y ind]=sort(X,'descend') devuelve Y que es X ordenado
% descendentemente. El vector de índices verifica Y=X(ind), de manera que
% Y(n) da el valor en rango n ya que está ordenado y también ind(n) da el
% índice original en X ya que X(ind(n))=Y(n)
% * Una condición que da un array lógico, como x<2 no puede combinarse con
% otra con operadores lógicos. Entonces x<2&&x>1 es un error. Sin embago la
% intersección puede calcularse primero hallando x1=x>1 luego x2=x<2 y
% produciendo la igualdad y=x1==x2. y es el equivalente de la conjunción
% deseada x<2&&x>1
% * La función int32(x) produce un entero *redondeando*
```

Bibliography

[Anan12] Yoko Anan, Kohei Hatano, Hideo Bannai, Masayuki Takeda, and Ken Satoh. Polyphonic music classification on symbolic data using dissimilarity functions. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Allamanche03] Eric Allamanche, Jürgen Herre, Oliver Hellmuth, Thorsten Kastner, and Christian Ertel. A multiple feature model for musical similarity retrieval. In *ISMIR 2003*, page 2, 2003.

[Abeser11] Jakob Abeßer and Olivier Lartillot. Modelling musical attributes to characterize two-track recordings with bass and drums. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Ahonen11] Teppo E. Ahonen, Kjell Lemström, and Simo Linkola. Compression-based similarity measures in symbolic, polyphonic music. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Angeles10] Bruno Angeles, Cory McKay, and Ichiro Fujinaga. Discovering metadata inconsistencies. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Anderson06] Chris Anderson. *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion, 20200609.

[Aucouturier02] Jean-Julien Aucouturier and Francois Pachet. Music similarity measures: What's the use? In *ISMIR 2002*, page 7, 2002.

[Bimbot10] Frédéric BIMBOT, Olivier LE BLOUCH, Gabriel SARGENT, and Emmanuel VINCENT. Decomposition into autonomous and comparable blocks : A structural description of music pieces. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Berenzweig03] Adam Berenzweig, Dan Ellis, Beth Logan, and Brian Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *ISMIR 2003*, page 36, 2003.

[Bogdanov11] Dmitry Bogdanov and Perfecto Herrera. How much metadata do we need in music recommendation? a subjective evaluation using preference sets. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Bertin-Mahieux11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Bertin11] Thierry Bertin-Mahieux, Matt Hoffman, and Dan Ellis. Million song dataset, 2011.

[Bryan11] Nicholas J. Bryan and Ge Wang. Musical influence network analysis and rank of sample-based music. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Coviello10] Emanuele Coviello, Luke Barrington, Antoni B. Chan, and Gert. R. G. Lanckriet. Automatic music tagging with time series models. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Cunningham06] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 'more of an art than a science': Supporting the creation of playlists and mixes. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 6, 2006.

[Celma08] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[Clausen00] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. Proms: A web-based tool for searching in polyphonic music. In *Ismir 2000*, page 2, 2000.

[Cooper02] Matthew Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. In *ISMIR 2002*, page 5, 2002.

[Chang10] Kaichun K. Chang, Jyh-Shing Roger Jang, and Costas S. Iliopoulos. Music genre classification via compressive sampling. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Chen10] Ya-Xi Chen and René Klüber. Thumbnaildj: Visual thumbnails of music content. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Ciresan12] Dan Claudiu Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[Collins10] Nick Collins. Computational analysis of musical influence: A musicological case study using mir tools. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Castro02b] L. N. De Castro and J. Timmis. Artificial immune systems: A novel paradigm to pattern recognition. In *University of Paisley*, pages 67–84. Springer Verlag, University of Paisley, UK, 2002.

[Casey10] Michael A. Casey and Spencer S. Topel. Timbre-based percussive rhythm classification and retrieval. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 1, 2010.

[Chai05] Wei Chai and Barry Vercoe. Detection of key change in classical piano music. In *Ismir 2005*, page 6, 2005.

[Dixon03] Simon D. lassification of dance music by periodicity patterns. In *ISMIR 2003*, page 20, 2003.

[Castro02a] Leandro Nunes de Castro and Fernando J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Trans. Evolutionary Computation*, 6(3):239–251, 2002.

[Bandera11] Cristina de la Bandera, Ana M. Barbancho, Lorenzo J. Tardon, Simone Sammartino, and Isabel Barbancho. Humming method for content-based music information retrieval. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Downie03] J. Stephen Downie. Toward the scientific evaluation of music information retrieval systems. In *ISMIR 2003*, page 8, 2003.

[Downie05] J. Stephen Downie, Kris West, Andreas Ehmann, and Emmanuel Vincent. The 2005 music information retrieval evaluation exchange (mirex 2005): Preliminary overview. In *Ismir 2005*, page 4, 2005.

[Draman10] Noor Azilah Draman, Campbell Wilson, and Sea Ling. Modified ais-based classifier for music genre classification. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Eyben10] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Ellis02] Daniel P.W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The quest for ground truth in musical artist similarity. In *ISMIR 2002*, page 8, 2002.

[Foote02] Jonathan Foote, Matthew Cooper, and Unjung Nam. Audio retrieval by rhythmic similarity. In *ISMIR 2002*, page 2, 2002.

[Fiebrink06] Rebecca Fiebrink and Ichiro Fujinaga. Feature selection pitfalls and music classification. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 2, 2006.

[Flexer06] Arthur Flexer, Fabien Gouyon, Simon Dixon, and Gerhard Widmer. Probabilistic combination of features for music classification. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 4, 2006.

[Forrest94] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, Rajesh Cherukur, Stephanie Forrest, and Lawrence Allen. Self-nonself discrimination in a computer. In *In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA. IEEE Computer*. Society Press, 1994.

[Farmer86] J. Doyne Farmer, Norman H. Packard, and Alan S. Perelson. The immune system, adaptation, and machine learning. *Physica 22D*, pages 187–204, 1986.

[Flexer05] Arthur Flexer, Elias Pampalk, and Gerhard Widmer. Novelty detection based on spectral similarity of songs. In *Ismir 2005*, page 4, 2005.

[Flexer10] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Tim Pohle. Combining features reduces hubness in audio similarity. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Flexer08] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *ISMIR 2008 – Session 2a – Music Recommendation and Organization*, page 6, 2008.

[Flexer12] Arthur Flexer, Dominik Schnitzer, and Jan Schlüter. A mirex meta-analysis of hubness in audio music similarity. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Feng02] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Popular music retrieval by independent component analysis. In *ISMIR 2002*, page 2, 2002.

[Guyon03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[Hatano11] Kohei Hatano, Yoko Anan, Hideo Bannai, and Masayuki Takeda. Music genre classification using similarity functions. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Hart68] A formal basis for the heuristic determination of minimum cost paths. volume SSC-4, pages 100–107, 1968.

[Harford03] Steven Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. In *ISMIR 2003*, page 2, 2003.

[Hoffman08] Matthew Hoffman, David Blei, and Perry Cook. Content-based musical similarity computation using the hierarchical dirichlet process. In *ISMIR 2008 – Session 3a – Content-Based Retrieval, Categorization and Similarity 1*, page 6, 2008.

[Hofmann01] Ludger Hofmann-Engl. Towards a cognitive model of melodic similarity. In *ISMIR 2001*, page 9, 2001.

[Hamel10] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[MHitchin12] Professor Nigel Hitchin. Differentiable manifolds (notes). Course C3.1b 2012.

[Hirata02] Keiji Hirata and Shu Matsuda. Interactive music summarization based on gttm. In *ISMIR 2002*, page 8, 2002.

[Hoos01] Holger H. Hoos, Kai Renz, and Marko Görg. Guido/mir — an experimental musical information retrieval system based on guido music notation. In *ISMIR 2001*, page 10, 2001.

[Holzapfel08] Andre Holzapfel and Yannis Stylianou. Beat tracking using group delay based onset detection. In *ISMIR 2008 – Session 5c – Rhythm and Meter*, page 6, 2008.

[Holzapfel10] Andre Holzapfel and Yannis Stylianou. Parataxis: Morphological similarity in traditional music. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Huron00] David Huron. Perceptual and cognitive applications in music information retrieval. In *Ismir 2000*, page 2, 2000.

[Izmirli10] Özgür İzmirli and Roger B. Dannenberg. Understanding features and distance functions for music sequence alignment. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Izumitani08] Tomonori Izumitani and Kunio Kashino. A robust musical audio search method based on diagonal dynamic programming matching of self-similarity matrices. In *ISMIR 2008 – Session 5b – Feature Representation*, page 5, 2008.

[Javel80] Eric Javel. Coding of am tones in the chinchilla auditory nerve: Implications for the pitch of complex tones. In *Acoustical Society of America*, page 68, 1980.

[Roger05] Jyh-Shing Roger Jang, Chao-Ling Hsu, and Hong-Ru Lee. Continuous hmm and its enhancement for singing/humming query retrieval. In *Ismir 2005*, page 6, 2005.

[Jin02] Hui Jin and H. V. Jagadish. Indexing hidden markov models for music retrieval. In *ISMIR 2002*, page 5, 2002.

[Kim02] Ja-Young Kim and Nicholas J. Belkin. Categories of music description and search terms and phrases used by non-music experts. In *ISMIR 2002*, page 6, 2002.

[Kephart94] Jeffrey O. Kephart. A biologically inspired immune system for computers. In *In Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139. MIT Press, 1994.

[Karydis10] Ioannis Karydis, Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Looking through the "glass ceiling": A conceptual framework for the problems of spectral similarity. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Kaiser10] Florian Kaiser and Thomas Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Lartillot03] Olivier Lartillot. ï¿¼discovering musical patterns through perceptive heuristics. In *ISMIR 2003*, page 8, 2003.

[Lampropoulos05] Aristomenis S. Lampropoulos, Paraskevi S. Lampropoulou, and George A. Tsihrintzis. Musical genre classification enhanced by improved source separation techniques. In *Ismir 2005*, page 6, 2005.

[Li03] Tao Li and Mitsunori Ogihara. Detecting emotion in music. In *ISMIR 2003*, page 2, 2003.

[Logan00] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Ismir 2000*, page 2, 2000.

[Logan02] Beth Logan. Content-based playlist generation: Exploratory experiments. In *ISMIR 2002*, page 2, 2002.

[Mesaros05] Annamaria Mesaros and Jaakko Astola. The mel-frequency cepstral coefficients in the context of singer identification. In *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, page 4, 2005.

[Marolt06] Matija Marolt. A mid-level melody-based representation for calculating audio similarity. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 6, 2006.

[McKinney03] Martin F. McKinney and Jeroen Breebaart. Features for audio and music classification. In *ISMIR 2003*, page 8, 2003.

[McFee10] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning similarity from collaborative filters. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Mandel05] Michael I. Mandel and Daniel P.W. Ellis. Song-level features and support vector machines for music classification. In *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, page 6, 2005.

[Mak10] Chun-Man Mak, Tan Lee, Suman Senapati, Yu-Ting Yeung, and Wang-Kong Lam. Similarity measures for chinese pop music based on low-level audio signal attributes. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Mulder03] Tom De Mulder, Jean-Pierre Martens, Micheline Lesaffre, Marc Leman, Bernard De Baets, and Hans De Meyer. An auditory model based transcriber of vocal queries. In *ISMIR 2003*, page 2, 2003.

[Miotto10] Riccardo Miotto and Nicola Orio. A probabilistic approach to merge context and content information for music retrieval. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Meng05] Anders Meng and John Shawe-Taylor. An investigation of feature models for music genre classification using the support vector classifier. In *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, page 6, 2005.

[Masahiro08] Niitsuma Masahiro, Hiroshi Takaesu, Hazuki Demachi, Masaki Oono, and Hiroaki Saito. Development of an automatic music selection system based on runner's step frequency. In *ISMIR 2008 – Session 2b – Music Recognition and Visualization*, page 6, 2008.

[Chinthaka03] Namunu Chinthaka Maddage, Changsheng Xu, and Ye Wang. A svm-based classification approach to musical audio. In *ISMIR 2003*, page 2, 2003.

[Norowi05] Noris Mohd Norowi, Shyamala Doraisamy, and Rahmita Wirza. Factors affecting automatic genre classification: An investigation incorporating non-western musical forms. In *Ismir 2005*, page 8, 2005.

[Nieto12] Oriol Nieto, Eric J. Humphrey, and Juan Pablo Bello. Compressing music recordings into audio summaries. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Oliver06] Nuria Oliver and Lucas Kreger-Stickles. Papa: Physiology and purpose-aware automatic playlist generation. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 4, 2006.

[Orio03] Nicola Orio and Matteo Sisti Sette. A hmm-based pitch tracker for audio queries. In *ISMIR 2003*, page 2, 2003.

[Pampalk06] Elias Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, March 2006.

[Paulus10b] Jouni Paulus. Improving markov model-based music piece structure labelling with acoustic information. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Panagakis08] Ioannis Panagakis, Emmanouil Benetos, and Constantine Kotropoulos. Music genre classification: A multilinear approach. In *ISMIR 2008 – Session 5a – Content-Based Retrieval, Categorization and Similarity 2*, page 6, 2008.

[Peeters02] Geoffroy Peeters, Amaury La Burthe, and Xavier Rodet. Toward automatic music audio summary generation from signal analysis. In *ISMIR 2002*, page 7, 2002.

[Parry03] Mitchell Parry and Irfan Essa. Rhythmic similarity through elaboration. In *ISMIR 2003*, page 2, 2003.

[Pampalk05-1] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classificaton. In *Ismir 2005*, page 6, 2005.

[Pickens05-1] Jeremy Pickens and Costas Iliopoulos. Markov random fields and maximum entropy modeling for music information retrieval. In *Ismir 2005*, page 8, 2005.

[Paulus10a] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-based music structure analysis. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 12, 2010.

[Pampalk05] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, page 4, 2005.

[Pauws06] Steffen Pauws, Wim Verhaegh, and Mark Vossen. Fast generation of optimal music playlists using local search. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 6, 2006.

[Reed06] Jeremy Reed and Chin-Hui Lee. A study on music genre classification based on universal acoustic models. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 6, 2006.

[Rump10] Halfdan Rump, Shigeki Miyabe, Emiru Tsunoo, Nobukata Ono, and Shigeki Sagama. Autoregressive mfcc models for genre classification improved by harmonic-percussion separation. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Rowland12] Todd Rowland. Manifold. A Wolfram Web Resource.

[Rauber02] Andreas Rauber, Elias Pampalk, and Dieter Merkl. Using psychoacoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity. In *ISMIR 2002*, page 10, 2002.

[Ravelli08] Emmanuel Ravelli, Gaël Richard, and Laurent Daudet. Fast mir in a sparse transform domain. In *ISMIR 2008 – Session 4c – Automatic Music Analysis and Transcription*, page 6, 2008.

[Scheirer98] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. In *In J. Acoust. Soc. Am. 103:1*, pages 588–601, 1998.

[Schedl10b] Markus Schedl. On the use of microblogging posts for similarity estimation and artist labeling. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Sheh03] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *ISMIR 2003*, page 7, 2003.

[Schnitzer10] Dominik Schnitzer, Arthur Flexer, Gerhard Widmer, and Martin Gasser. Islands of gaussians: The self organizing map and gaussian music similarity features. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Schmidt09b] C. Schmidt-jones. *The Basic Elements of Music*. Orange Grove Books, 2009.

[Stenzel05] Richard Stenzel and Thomas Kamps. Improving content-based similarity measures by training a collaborative model. In *Ismir 2005*, page 8, 2005.

[Stober10] Sebastian Stober and Andreas Nürnberger. Musicgalaxy – an adaptive user-interface for exploratory music retrieval. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 1, 2010.

[Schedl10a] Markus Schedl, Tim Pohle, Noam Koenigstein, and Peter Knees. What's hot ? estimating country-specific artist popularity. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, page 6, 2010.

[Sigurdsson06] Sigurdur Sigurdsson, Kaare Brandt Petersen, and Tue Lehn-Schiøler. Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 4, 2006.

[Symeonidis08] Panagiotis Symeonidis, Maria Ruxanda, Alexandros Nanopoulos, and Yannis Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. In *ISMIR 2008 – Session 2c – Knowledge Representation, Tags, Metadata*, page 6, 2008.

[Slaney08] Malcolm Slaney, Kilian Weinberger, and William White. Learning a metric for music similarity. In *ISMIR 2008 – Session 3a – Content-Based Retrieval, Categorization and Similarity 1*, page 6, 2008.

[Scaringella05] Nicolas Scaringella and Giorgio Zoia. On the modeling of time information for automatic genre recognition systems in audio signals. In *Ismir 2005*, page 6, 2005.

[Turnbull08] Douglas Turnbull, Luke Barrington, and Gert Lanckriet. Five approaches to collecting tags for music. In *ISMIR 2008 – Session 2c – Knowledge Representation, Tags, Metadata*, page 6, 2008.

[Tzanetakis01] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *ISMIR 2001*, page 6, 2001.

[Temperley06] David Temperley. A probabilistic model of melody perception. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 4, 2006.

[Typke03] Rainer Typke, Panos Giannopoulos, Remco C. Veltkamp, Frans Wiering, and René van Oostrum. Using transportation distances for measuring melodic similarity. In *ISMIR 2003*, page 8, 2003.

[IDM05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[Typke05] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. A survey of music information retrieval systems. In *Ismir 2005*, page 8, 2005.

[Tsai05] Wei-Ho Tsai, Hung-Ming Yu, and Hsin-Min Wang. A query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *Ismir 2005*, page 8, 2005.

[Uitdenbogerd00] Alexandra L. Uitdenbogerd, Abhijit Chattaraj, and Justin Zobel. Music ir: Past, present and future. In *Ismir 2000*, page 2, 2000.

[Urbano12] Julián Urbano, J. Stephen Downie, Brian McFee, and Markus Schedl. How significant is statistically significant? the case of audio music similarity and retrieval. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Wang11] Ju-Chiang Wang, Meng-Sung Wu andHsin Min Wang, and Shyh-Kang Jeng. A content-based music search system using query by multi-tags with multi-levels of preference. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 2, 2011.

[Welsh99] Matt Welsh, Nikita Borisov, Jason Hill, Robert von Behren, and Alec Woo. Querying large collections of music for similarity. Technical report, Computer Science Division, University of California, Berkeley, 1999.

[West05] Kris West and Stephen Cox. Finding an optimal segmentation for audio genre classification. In *Ismir 2005*, page 6, 2005.

[Whiteley06] Nick Whiteley, A.Taylan Cemgil, and Simon Godsill. Bayesian modelling of temporal structure in musical audio. In *7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, page 6, 2006.

[Weisstein12] Eric W. Weisstein. Riemannian metric. A Wolfram Web Resource.

[Watson12] Diane Watson and Regan L. Mandryk. Modeling musical mood from audio features and listening context on an in-situ data set. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Wulfing12] Jan Wülfing and Martin Riedmiller. Unsupervised learning of local features for music classification. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, page 6, 2012.

[Wolff11] Daniel Wolff and Tillman Weyde. Adapting metrics for music similarity using comparative ratings. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, page 6, 2011.

[Xu04] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *PCM (3)*, volume 3333 of *Lecture Notes in Computer Science*, pages 566–574. Springer, 2004.

[Yoshii08] Kazuyoshi Yoshii and Masataka Goto. Music thumbnailer: Visualizing musical pieces in thumbnail images based on acoustic features. In *ISMIR 2008 – Session 2b – Music Recognition and Visualization*, page 6, 2008.