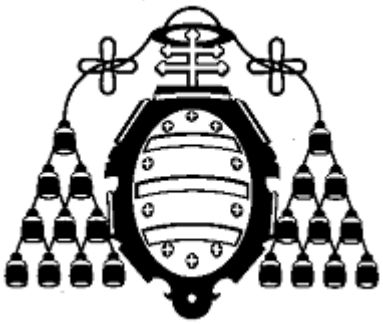
UNIVERSIDAD DE OVIEDO



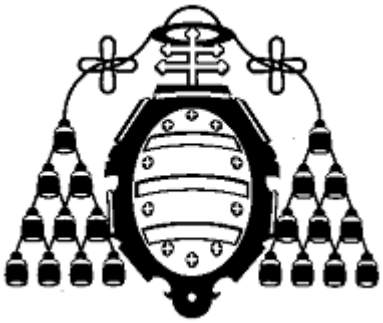
MASTER IN SOFT COMPUTING AND INTELLIGENT DATA ANALYSIS

PROYECTO FIN DE MASTER
MASTER PROJECT

OPINION ANALYSIS IN WEB 2.0

PABLO JOSÉ PÉREZ GÁLLEGO
JULY, 2012

UNIVERSIDAD DE OVIEDO



MASTER IN SOFT COMPUTING AND INTELLIGENT DATA ANALYSIS

PROYECTO FIN DE MASTER
MASTER PROJECT

OPINION ANALYSIS IN WEB 2.0

PABLO JOSÉ PÉREZ GÁLLEGO

JULY, 2012

TUTOR / ADVISOR:
JORGE DÍEZ PELÁEZ
OSCAR LUACES RODRIGUEZ

Abstract

During the last years we are assisting to an intense Web transformation process. It is no longer a mere static information repository but a dynamic system in which users have become the main content contributors. They actively participate in sharing their opinions, thoughts and views about products, events and almost anything in social networks, forums, blogs, etc. With the latest advances in mobile technologies, users can actually interact anytime from anywhere; real time information has become a reality. All these mixture of social networks, discussion groups, forums and blogs are collectively called the *user-generated content*. It has many practical applications and has a potential major value from both the user and business points of view. On one hand, knowing other user opinions is useful when having to take a decision in our daily life. On the other hand, it is an invaluable information source about user preferences and tastes. Due to the large and diverse number of opinion sources, it appears the necessity of systems able to automatically discover, analyze and summarize the expressed sentiment in the so-called user-generated content. *Sentiment analysis* grows out of this need. It focuses on the computational study of people's opinions, appraisals, and emotions toward entities, events and their properties.

In the first three chapters of this document we introduce the problem of sentiment analysis, describing its main characteristics and difficulties, we briefly present the main theoretical background of the realized work, and we provide the reader with an exhaustive literature review, analyzing the previous related works in the literature. Afterwards, we face a sentiment classification problem consisting in learning to classify a series of movie reviews, as positive or negative, in function of the sentiment expressed by the author. In chapter 4 we present the dataset and its main properties, together with all the preprocess steps we have applied to the text movie reviews in order to obtain valuable representations. We also present the methodology we used to execute the experiments and to estimate the performance of the proposed approaches. In chapter 5 we describe our solutions to the problem, we present the details of all the performed experiments and evaluate and discuss the obtained results. As baseline we have reproduced an extensive part of the experiments presented in [Pang et al., 2002]. As follows we propose a series of feature reduction approaches, with the objective of selecting a reduced and representative vocabulary of the movie review domain. Finally, we propose a novel method based on measuring word cooccurrence information in order to obtain a "meaning" representation of the text documents.

Keywords: Sentiment analysis, opinion mining, machine learning, polarity classification, feature selection, SVM, word cooccurrence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Web expansion

During the last years we are assisting to an intense Web transformation process. It is no longer a mere static information repository but a dynamic system in which users have become the main content contributors. They actively participate in sharing their opinions, thoughts and views about products, events and almost anything in social networks, forums, blogs, etc. The popularity of social networks like `Twitter` or `Facebook` has only just started to increase in the last few years. A key reason for its success is that a simple interface is presented to the users, so that they are able to share their sentiments, ideas and experiences in a fast and easy way. Moreover, with the latest advances in mobile technologies, users can actually interact anytime from anywhere; real time information has become a reality. This collaborative and interactive process among users, aimed at sharing information, has received the name of *Web 2.0*.

All these mixture of social networks, discussion groups, forums and blogs are collectively called the *user-generated content* [Liu, 2010a]. It is a new source of information that noticeably represents the word-of-mouth behavior that is present on the Internet. It has many practical applications and has a potential major value from both the user and business points of view. On one hand, knowing other user opinions is useful when having to take a decision in our daily life. We can learn from other user experiences who have already faced the same situation or decision in the past (*wisdom of crowds*). On the other hand, it is an invaluable information source about user preferences and tastes. A key circumstance is that users on the Internet share their opinions by their own free will, without expecting anything in return and without coercion.

However, despite the elevated number of available sources for a user to express its opinions, the existing search engines are still focused on factual information rather than on opinionated (subjective) one. Text information can be broadly categorized into two main types: *facts* and *opinions*. Facts are *objective* expressions about entities, events and their properties. On the other hand, opinions are *subjective* expressions used to describe sentiments or feelings towards entities, events and their properties

[Liu, 2010a]. Facts are whether true or false independently of the individual stating them. In contrast, opinions can not be categorized according to its truth/falseness, and they depend on the author's subjective experience.

Much of the existing research regarding textual information has been traditionally focused on mining and retrieval of factual information, e.g., information retrieval, Web search, etc. These are techniques that work with facts assuming that they are true. Well-known strategies use keywords to represent fact related queries, and then a similarity measure is applied in order to generate a relevance ranking of the retrieved documents. For example, when we are using `Google` to look for factual information, generally it suffices to just look into the top ranked links, since to a greater or lesser extent they all provide us with the same information. On the other hand, when looking for opinions about a particular entity, the top ranked links only represent a little subset of all the opinionated information that is available on the Internet. They do not necessarily represent the global expressed sentiment about the entity. Current search ranking strategy is not appropriate for opinion retrieval/search.

With the latest appearance of opinionated text in the Web, opinion based research has started to grow. Due to the large and diverse number of opinion sources, and the high volume of available information on each one (sparseness), it may be difficult for human users to perform the opinion search task. It is complicated to find relevant sources, extract the passages in which opinions are stated, and finally summarize all the expressed sentiment and organize it into usable forms, so that it help us in decision making. Sometimes useful opinions are "hidden" in large forums surrounded by non important or not related information. It appears the necessity of systems able to automatically *discover*, *analyze* and *summarize* the expressed sentiment in the so-called user-generated content.

Figure 1.1: 2.0 Web representation

1.2 Sentiment Analysis

Sentiment Analysis, also called *Opinion Mining* [Pang and Lee, 2008], is a novel research field that grows out of this need. Sentiment analysis is the computational study of people's *opinions*, *appraisals*, and *emotions* toward entities, events and their properties [Liu, 2010b]. The objective is to determine the text author's attitude, whether (s)he is giving an opinion about an entity/event, expressing an emotional state (s)he is in, or on the other hand trying to cause that state on the text reader.

Information Retrieval (IR) consists in finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [Manning et al., 2008]. In our particular case, the material are the texts composed of unstructured opinions which are useful for answering a opinion based question within the set of opinion documents on the Internet. In the scope of this project, we will consider an opinion as a *positive* or *negative* sentiment expressed by a user on a entity or event, by means of a written commentary.

Over the past few years this problem has attracted the attention of both, research in academia, and applications in business and industry. The principal reasons are that on one hand it is a very challenging problem from the research point of view (leading to a potential high number of publications), and on the other it offers a wide range of practical applications. It has not been until a few years that it became popular because it was complicated to obtain high volumes of opinionated text. This kind of information can be easily retrieved nowadays via Internet. The increase of Machine Learning (ML) methods in Natural Language Processing (NLP) and Information Retrieval tasks has also been a contributive factor.

Sentiment classification can be divided into several specific subtasks [Esuli and Sebastiani, 2005]:

- Determining **subjectivity**, consists in separating texts with factual information from texts with opinions. It is a binary classification task with classes being whether *Objective* or *Subjective* [Wiebe, 2000; Wiebe et al., 2004; Yu and Hatzivassiloglou, 2003].

- Determining **polarity**, from a subjective text, determine whether the expressed opinion is positive, negative or a mixed one [Pang et al., 2002; Turney, 2002; Dave et al., 2003]. It is a binary or multiclass classification task depending on the consideration of the neutral class. Each instance is labeled as *Positive* or *Negative*, and in some problems it is also considered the *Neutral* class.

- Determining the **strength** of polarity, consists in quantifying the degree of the sentiment. It is a multiclass problem with classes usually following a range like: *weakly* positive, *mildly* positive or *strongly* positive. It is also called *affective* classification, when trying to predict users mood, such as happiness, sadness, kindness, sureness and so on [Subasic and Huettner, 2001; Grefenstette et al., 2004; Bollen et al., 2011].

Conceptually, sentiment analysis is a natural language processing or text mining related problem, which can be technically complex. It is a cross-domain problem, in which a series of diverse techniques belonging to different fields, including linguistics, computer science and statistics, have been traditionally applied. Some of the existing techniques range from phrase pattern matching and language usage heuristics, to manual/semi-automatic annotation techniques using information theory measurements to assign scores to words and sentences. Sentiment lexicon generation, whether manually or semi-automatically, has been another important research area. On the other hand, machine learning methods, such as Support Vector Machines (SVM), have been successfully applied to polarity classification problems.

However, during the last years most of the proposed techniques have started to gradually become more and more specific and language dependent. They are strongly focused on natural language processing, aiming at analyzing its single elements, syntactic structures and so on. This is traduced into a higher performance in some specific domains, but the resulting techniques are less applicable in general domains as main drawback.

Precisely, one of the problems affecting this research area is that there is not a well-known and widely accepted set of test problems to measure the effectiveness of the proposed techniques in a general domain. Finding a representative set of data that is correctly labeled is not an easy task. Thus, authors tend to test the performance of their approaches over specific problems with a limited domain. As a consequence, it is difficult to perform comparisons among different strategies and to evaluate the real performance of each solution.

1.3 Principal difficulties

Sentiment Analysis is a challenging problem that presents numerous difficulties inherent to the complexity of natural language processing. Take `Twitter` as example, despite its 140 character length limitation, it is already sometimes difficult for a human to determine the polarity of a single `tweet` due to the lack of context. In general, any task comprising natural language processing presents an elevated number of complications.

The first obstacle we may face is the language diversity that appears on the Internet. Each user generally comments in its mother tongue, and nowadays it is not strange to see that on a same platform (`Twitter`, `Youtube`, etc.), interrelated or question/answer comments may be expressed in different languages. In this context, language dependent techniques may get negatively affected. This situation gets worsened due to the fact that usually people communicating in the same language use different writing styles (jargon). Another common practice in web texts is that people use abbreviations (SMS language) and tend to make a lot of spelling mistakes. As a consequence, identical comments with respect to their semantic meaning may have different textual representations.

A traditional representation of the documents in IR is made by considering each word in the vocabulary by separate. However, when the objective is to analyze the sentiment, that representation suffers from the *lack of context* problem. An isolated word or

term by itself does not suffice to determine its semantic meaning. The same phrase can indicate different sentiment in different domains. For example, the word "*unpredictable*" is a positive description for a movie plot but a negative description for a car's steering abilities [Turney, 2002]. Likewise, even though terms as "*devastating''* are generally seen as negative, in the context of movies or music it might imply an emotional engagement which is usually seen as positive [Mullen and Collier, 2004]. Differences in vocabulary across different domains is also an additional issue.

We can not think about document's polarity as a function of the individual word's sentiment comprising it either. As Turney states in [Turney, 2002], "*the whole is not necessarily the sum of the parts*". Sentiment can be expressed in a more subtle manner. For example, the sentence "*How could anyone sit through this movie?*" does not contain any single word that is obviously negative, but the sentiment it expresses is in fact negative. It is also common a kind of "thwarted expectations" narrative style, in which the writer expects to create an emotive effect by emphasizing the contrast between what he expected and the actual experience. Take into account the next commentary: "*This film should be* ***brilliant****. It sounds like a* ***great*** *plot, the actors are* ***first grade*** *and the supporting cast is* ***good*** *as well, and Stallone is attempting to deliver a* ***good*** *performance. However, it* ***can't hold up****.*" [Pang et al., 2002]. Note that it is mostly composed of naturally positive words, but it expresses a negative opinion about the film. Moreover, the reviewer emphasizes that he expected the film to be a brilliant one. While it would be easy for a human to detect the true sentiment of the review, a classifier using a bag-of-words representation would be biased towards the opposite class, because most of the words being obviously positive. It is also quite common that the concluding sentences of a commentary summarize the opinion expressed in the whole composition.

The potential important contextual effect of *negation* is another core problem [Jia et al., 2009; Wiegand et al., 2010]. These kind of clauses directly affect the polarity of the comments they appear in. Whenever a negation word appears in a sentence, it usually causes the meaning of the sentence to be the opposite of that without the negation. Trying to model its influence is a very challenging task itself and has become a hot research area in the last years. Several language heuristics have been proposed, such as reverting the polarity of the next 5 words appearing after a negation clause, or until a punctuation symbol is found [Das and Chen, 2001; Pang et al., 2002]. They are far from being a definitive solution though. Negation can be presented explicitly (negation clauses) or implicitly with more subtle linguistic patterns that may be language dependent. Another complication is that negation clauses do not necessarily invert the polarity of the following words in certain situations. For example, the clause "*not''* does not modify the polarity of "*cheap*" and "*reliable*" in "*it is not only cheap but also reliable*". Determining the scope of negation clauses is also an important problem [Councill et al., 2010]. Its effect may affect only to the following word, or by contrast to a whole paragraph.

It is usual in many discourse domains for writers to use rhetorical devices or language resources with the objective of evoking an emotional response in the reader. It is very difficult to model these kind of special writing resources in sentiment analysis tasks. Examples of rhetorical devices are irony and metaphor. Irony automatic detection in

texts is a tough problem that remains unsolved [Carvalho et al., 2009]. It mainly affects to negative sentiments that are deliberately conveyed by means of positive sentences.

1.4 Practical applications

A system that is able to successfully discover, analyze and summarize all the valuable information present in the user-generated content has a large number of practical applications. Among them we highlight the following:

- **Businesses and organizations**, useful for market intelligence and online product and service monitoring. It is an invaluable information source of consumer's tastes and preferences. Business usually spends a huge amount of money to find consumer sentiments and opinions.
- **Individuals**, interested in other user opinions when having to purchase a product or using a service. They might be also interested in opinions about political topics. We are no longer restricted to our circle of friends opinions; it exists a global scale on the Internet.
- **Ads placements**, placing user personalized ads in the user-generated content. It would be interesting to announce a product when someone expresses a positive opinion about it. On the other hand, placing an ad from the competitor when someone criticizes a product might also be a good idea.
- **Opinion retrieval/search**, an opinion and sentiment oriented search service. Contrasting with traditional factual oriented search services, it would allow users to perform general searches for opinions. Sentiment and opinion summarization would also be an interesting feature.

1.5 Research goals

The main research goals expected for a master project like this are the following:

- Acquire comprehensive background research of a problem and produce a literature review.
- Develop and demonstrate skills acquired from the taught courses either in the analysis of some theoretical aspect or in the solution of a real practical problem.
- Critically evaluate the obtained results.
- Improve written and verbal communication skills.

In our case we have focused on introducing and describing the main characteristics of the sentiment analysis problem. In Chapter 3 we have produced an exhaustive literature review, analyzing the previous work done on the problem. A taxonomy that allows us to examine the problem with respect to the tasks, features, techniques and domain applications is provided. We have solved a particular sentiment analysis problem consisting in developing a system able to automatically classify a series of movie reviews in function of the sentiment expressed by the author, whether positive or negative. We have proposed a number of solutions, and assessed its performance with respect to a defined performance measure.

Natural language processing is a very challenging research field and an important task within the sentiment analysis problem. Much of the existing research in this field focuses on the NLP subtask, examining syntactic and semantic characteristics of the language. As a result, the proposed techniques tend to be language dependent and less applicable in a general domain. In order to avoid these circumstances, our principal objective is to use machine learning methods in order to generate models presenting an adequate tradeoff between efficiency and complexity, that are applicable in a general domain. We are interested in getting rid of language dependent artifacts, such as phrase patterns, (sentiment) lexicons, and syntactic/semantic constructions. Usually, these kind of techniques are more complex and domain dependent. Furthermore, the application of most of these language dependent techniques requires the usage of external tools, like part of speech taggers, etc. Analyzing and using this sort of NLP techniques remained out of our scope, since machine learning is our core work interest.

One of our goals is to explore diverse methods in order to represent the textual information. We focus in trying to extract the vocabulary that is most likely to help us in the classification task. We start by reproducing an extensive part of the experiments that were presented in [Pang et al., 2002], the first work that explicitly measured the effectiveness of applying machine learning methods to the sentiment analysis problem. As follows we propose a series of feature reduction approaches, with the objective of selecting an adequate subset of words in the vocabulary. We also discuss the application of an ensemble, that is to aggregate a series of different models, and which are the model characteristics an ensemble would get benefited from. All of these steps are performed with the goal of obtaining a reduced and representative vocabulary of the movie review domain in a sentiment analysis task.

An additional important objective consists in designing a novel approach for solving the problem. Our idea consists in representing each document by using the cooccurrence information of the words comprising it. From this cooccurrence information, each word in the vocabulary is represented by a high dimensional vector using Vector Space Models. Words that are close in that high dimensional space, are expected to have a similar semantic meaning. Then we have applied a simple clustering algorithm and obtained a series of *meaning* clusters, i.e., ideally, words that are semantically similar fall in the same cluster. From the cluster information we have created a document "*meaning*" representation as input for the classification algorithms.

Additional and important objectives are learning to use and to apply a scientific methodology, based on empirical and measurable evidence. It is very important for the performed experiments to be repeatable, so other researchers are in position to check the validity of our presented work. The process of creating an structured document describing in detail the problem we are trying to solve, together with the adopted solutions, the followed methodology and the obtained results and conclusions, is a remarkable objective for this master.

Chapter 2

Theoretical background

In this chapter we are going to introduce the main theoretical background of the methods and techniques that have been applied in this project to the sentiment analysis problem. It is not our intention to be exhaustive but to provide the reader with the basis and references in case he/she is not familiar with any of the concepts.

2.1 Machine Learning

Artificial Intelligence (AI) is the branch of computer sciences that aims to make intelligent machines, and in particular intelligent computer programs. It can be defined as the study and design of intelligent agents [Poole et al., 1998]. An intelligent agent is a system that receives percepts from the environment and interacts with it by means of performing actions in accordance to an objective [Russell and Norvig, 2003]. A typical representation of an AI system is presented in Figure 2.1.

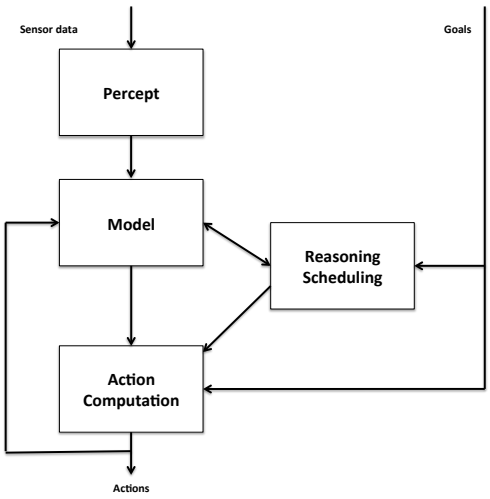


Figure 2.1: Typical representation of an AI system

Machine Learning (ML) is a branch of artificial intelligence which goal is to study and to design algorithms able to generalize behaviors based on empirical data. It is a knowledge *induction process*; from observations of individual instances to broader generalizations. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [Mitchell, 1997].

Data is presented as a series of individual *examples* or instances. A number of *variables* is observed and measured in each example. A major goal is to find relations between the observed variables, that is to capture characteristics of interest of their unknown underlying probability distribution. Thus, it is possible to recognize patterns in the input data and to make intelligent decisions based on the data itself.

The core objective of a machine learning algorithm is to *generalize* from its experience. Generalization is the ability of an algorithm to perform accurately on new, unseen examples after having trained on a finite set of examples. It should be able to produce useful outputs when facing unseen examples. The training instances constitute the experience of the learner. Thus, it is very important for the training set to adequately represent the problem we want to solve.

2.1.1 Types of data

The set of input examples that are used during the training stage is the so-called *training data* set. Each example is characterized by a series of variables that receive the name of *features* or *attributes*. These features are used to indicate the main properties of each instance. In supervised learning problems (2.1.2) there exists a special feature called *label* that represents the *class* of the example. There are different learning problems depending on the class type, e.g., classification, regression, etc. On the other hand, in unsupervised learning problems (2.1.3) the label is not used. A classical representation of a learning problem is shown in Table 2.1.

Features					
Feat. 1	Feat. 2	...	Feat. $n-1$	Feat. n	Label (optional)
			Example 1		Example's 1 label
			Example 2		Example's 2 label
			...		...
			Example m		Example's m label

Table 2.1: Classical representation of a learning problem

There are different types of features according to its nature:

- **Nominal scale**: the values are names without a natural order. For example a colour scale: *yellow*, *green*, *red*, *blue*, etc.

- **Ordinal scale**: the values are names but they may be (naturally) ordered. For example a feature representing the quality of an entity: *low*, *medium*, *high*.
- **Cardinal and discrete scale**: the values are numbers and the number of different possible values is finite. For example a feature representing a year in a given range: *1995*, *1996*, *1997*, etc.
- **Cardinal and continuous scale**: the values are numbers (integers or reals) and the number of different possible values is not finite. For example the height (cm) of a human: *172.5*, *187.2*, etc.

A series of examples regarding the different types of features and classes (labels) is presented in Table 2.2.

Features				Class
Sepal length	Sepal width	Petal length	Petal width	Type
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	2.7	5.1	1.9	Iris virginica
Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Cycle time (ns)	Main memory (Kb)	Cache (Kb)	Channels MIN	Performance
125	256	256	16	198
29	8000	32	8	256
480	512	32	0	67
480	1000	0	0	45

Table 2.2: Examples of the different types of features and classes

Training data related problems

One of the most important problems is the so-called *curse of dimensionality*. It affects to high dimensional problems, i.e., when the number of features is very large. The problem is that as the number of features increase, the volume of the input search space also increases very fast. As a consequence, the number of training examples required to ensure that most of the feature value possible combinations are represented is very large. That is to say that an enormous number of training instances is needed for adequately represent the problem. With a fixed number of training samples, the predictive power reduces as the dimensionality increases. This is known as the *Hughes effect* [Hughes, 1968].

Another problem is present when there is a certain amount of bias in the values of the features. This phenomenon receives the name of *noise*, and can be caused by faulty or

imprecise sensors, or by manual errors while capturing or handling the data. A bad representation of the problem may be obtained when choosing the wrong features in order to describe the instances. Some of them may not be relevant, because they do not have influence on the knowledge we want to induct. It could also happen that several features represent the same knowledge about the problem, in this case they are called redundant attributes. Choosing the relevant features for representing the examples in each task is a broadly studied problem in the literature.

2.1.2 Supervised learning

Supervised learning is the task consisting in inferring a function from labeled training data (examples), and then using it to predict unseen examples. Each example is comprised by an input object (typically a vector) and by a an output value (label). During the training stage an hypothesis h or model is learned, such that it assigns the right label from the attribute values. The model is called a *classifier* if the output is discrete, or a *regression function* if the output is continuous. A typical supervised learning representation is shown in Figure 2.2.

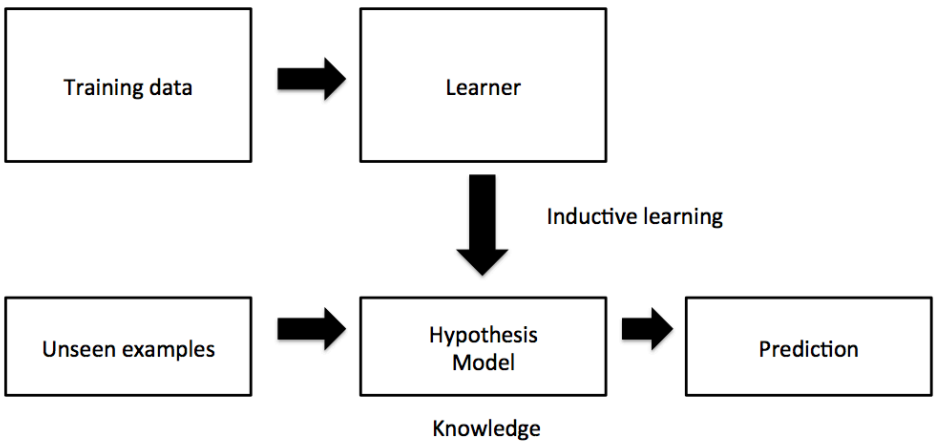


Figure 2.2: Typical representation of a supervised learning procedure

Supervised learning is based on the inductive learning hypothesis:

> "*Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples*"

From a formal point of view, a supervised learning task can be presented in the following general framework. Let $\mathcal{X}$ be an input space, and let $\mathcal{Y}$ be an output space. A learning task is given by a training set $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$ drawn from an unknown distribution $D = Pr(\mathrm{X,Y})$ from the product $\mathcal{X}$ x $\mathcal{Y}$. The aim of such task is to learn an (unknown) objective function $f : \mathcal{X} \rightarrow \mathcal{Y}$, by means of synthesizing a function

$h : \mathcal{X} \rightarrow \mathcal{Y}$ called hypothesis in a space $\mathcal{H}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$. The goal is to find an h such that optimizes the expected prediction performance (or risk) on samples $\mathcal{S}'$ independently and identically distributed (i.i.d.) according to the distribution Pr(X,Y).

In order to assess its performance it is defined a loss function $l : \mathcal{Y}$ x $\mathcal{Y} \rightarrow \mathbb{R}^{\geq 0}$. The performance of a prediction $h(\boldsymbol{x}_i)$ when the true value is y_i, is calculated as l(y,ŷ). The risk (opposite to performance) of the predictions made by a hypothesis h over a concrete sample $\mathcal{S}$ is calculated as:

$$R_{\mathcal{S}}(h) = \frac{1}{|\mathcal{S}|} \sum_{\substack{x_i \in \mathcal{S} \\ y_i \in \mathcal{S}}} l(y_i, h(x_i)) \tag{2.1.2.1}$$

When selecting the adequate hypothesis, it is not appropriate to measure its performance over the training data itself (resubstitution error). It does not represent the real performance of the model over new and unseen data. It is an optimistic measure and the model may adjust too much to the training data, thus negatively affecting its ability to generalize, i.e., what we are looking for. The objective is to find a tradeoff between the model's complexity and its prediction accuracy on the training data. *Occam's Razor* principle is typically put into practice; the simplest model able to explain the training data is the best.

It is said that a hypothesis $h_1 \in \mathcal{H}$ *overfits* to a sample $\mathcal{S}$ if there exists an alternative hypothesis $h_2 \in \mathcal{H}$ such that:

$$R_{\mathcal{S}}(h_1) < R_{\mathcal{S}}(h_2) \text{ y } R_D(h_1) > R_D(h_2) \tag{2.1.2.2}$$

It is said that a hypothesis $h_1 \in \mathcal{H}$ *underfits* to a sample $\mathcal{S}$ if there exists an alternative hypothesis $h_2 \in \mathcal{H}$ such that:

$$R_{\mathcal{S}}(h_1) > R_{\mathcal{S}}(h_2) \text{ y } R_D(h_1) > R_D(h_2) \tag{2.1.2.3}$$

There are different supervised learning tasks depending on the label's nature:

- **Classification**, finite set of nominal or symbolic labels. For example a disease diagnosis application in which each instance is labeled whether "*healthy*" or "*ill*".

- **Ordinal regression**, finite set of ordinal labels. The relative ordering among the different values is significant. For example an application that tries to predict the quality of cow's meat. Each instance is labeled with an integer in a fixed range, so that the higher the label, the higher the quality of the represented individual. When the ordering among the labels is not significant, it is viewed as a standard classification problem.

- **Regression**, labels are whether integer or reals in a continuous scale, i.e., the label set is not finite. For example an application trying to predict the height of an individual in the future.

2.1.3 Unsupervised learning

Unsupervised learning refers to the problem of trying to find hidden structure in unlabeled data. As the instances are not assigned with a concrete label, is not possible to (explicitly[1]) measure the performance of the obtained models. One of the principal objectives is to look for patterns or regularities in the data, so that it is possible to extract any knowledge from the data. Humans can benefit from this knowledge as a way of better understanding the available data. A typical unsupervised learning representation is shown in Figure 2.3.

Clustering is the most famous unsupervised learning task. It is the task of assigning a set of objects into groups (clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.

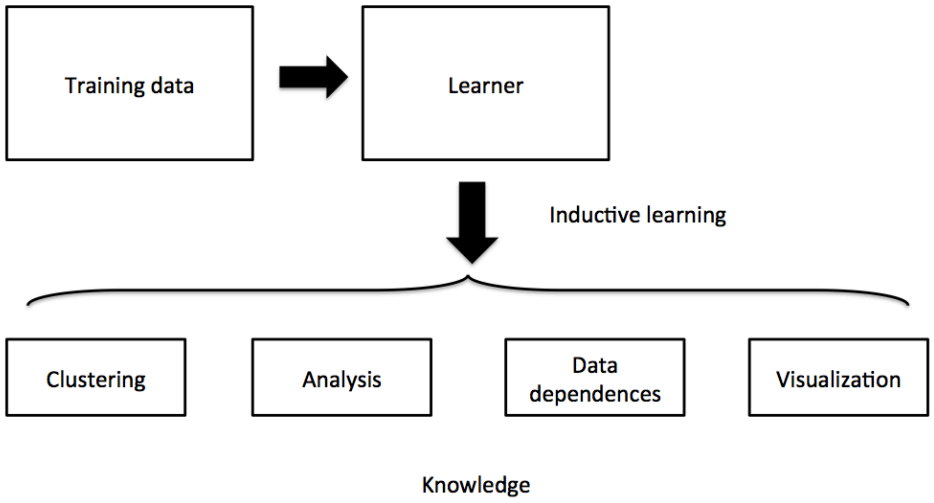


Figure 2.3: Typical representation of an unsupervised learning procedure

2.1.4 Principal applications

Machine learning techniques are suitable, and usually applied in the following contexts:

- Problems in which a human expert is not available. ML techniques are in position to provide us with an explicit knowledge about the problem solving procedure. Decision trees are very useful for this purposes.

- Tasks that humans usually perform, but it is not possible to describe the exact procedure we follow. For example character recognition applications.

[1] There are a series of techniques in order to assess the quality of unsupervised methods, as clustering, which don't take the class into account.

- Problems in which the volume of available data is huge. Humans can not analyze and draw conclusions when the amount of data is large. Data mining techniques are able to find relations among variables in huge databases, which are usually not detectable at a glance.

Machine learning solves problems that can not be modeled, but in whom we have experience (or it is possible to acquire it) on how can them be solved.

2.2 Fundamentals of SVM

Among all the *kernel methods* available nowadays, *Support Vector Machines* (SVM) stand out as one of the most popular and powerful learning machines. They were formally introduced by Vapnik from his studies about statistical learning algorithms [Vapnik, 1995] –where the generalization error was bounded with respect to the complexity of the hypothesis space– the first proposals date from the early nineties [Cortes and Vapnik, 1995; Boser et al., 1992]. These methods have increased their popularity along the years due to their high performance and effectiveness in solving real-world problems. Actually, an important part of the success of SVM methods in industrial applications is the continuous support of AT&T Bell Labs.

Although SVM were initially designed for binary classification problems, they have been generalized to many other machine learning fields, like regression, multi-class classification, ordinal regression and preference learning, etc. Furthermore, the basis for solving more complex problems have been proposed recently, in which the output is even more structured, like graphs and trees, e.g., [Tsochantaridis et al., 2004].

The primary objective of SVM is to obtain efficient models, whose predictions have a high confidence, even when they produce certain errors. Conversely, traditional machine learning approaches have been mainly focused on reducing these errors, i.e., the principle of *Empirical Risk Minimization* (ERM). The SVM approach is quite different, trying to construct models that are structurally correct in order to obtain a higher confidence for future predictions, known as *Structural Risk Minimization* (SRM).

As an example, imagine a plane in a 3D space where there exist two sets of points that are linearly separable in that space. Then, an ERM approach will try to find a specific position of a 2D plane, where the points are separated by it. This position is usually computed taking into account the errors of the model with respect to the known points. However, a SRM approach will try to find the position of the plane that minimizes the risk of wrongs prediction for unknown points, even though it commits some errors over the known ones. In the specific case of SVM, this is usually achieved by maximizing the margin of the solution, which reduces the bounds for generalization errors. Hence, it is not a matter of changing the model itself, but of changing the way it is obtained and, in turn, its properties.

2.2.1 Scalar product

This section is intended to present the key concepts of SVM methods, which includes scalar product, kernel theory and quadratic convex optimization. Hence, we will first define the interpretation of scalar product adopted for this work

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i = \boldsymbol{x}^{\mathsf{t}} \boldsymbol{y}, \tag{2.2.1.1}$$

with three key properties

1. $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$,
2. $\langle \boldsymbol{x}, (\boldsymbol{y} + \boldsymbol{z}) \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$,
3. $\langle a\boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{x}, a\boldsymbol{y} \rangle = a\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.

The associated norm is:

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \quad \Rightarrow \quad \|\boldsymbol{x}\|^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle, \tag{2.2.1.2}$$

and the corresponding metric or distance is:

$$d(\boldsymbol{x}, \boldsymbol{y})^2 = \|\boldsymbol{x} - \boldsymbol{y}\|^2 = \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle = \langle \boldsymbol{x}, \boldsymbol{x} \rangle - 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle + \langle \boldsymbol{y}, \boldsymbol{y} \rangle. \tag{2.2.1.3}$$

Geometric interpretation of scalar product

Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ from a multi-dimensional space $\mathbb{R}^n$, if their norms have value one then their scalar product ranges from +1 (same) to -1 (opposite), taking value zero when they are perpendicular:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\| \cdot cos(\boldsymbol{x}, \boldsymbol{y}) \quad \Rightarrow \quad \langle \boldsymbol{x}, \boldsymbol{y} \rangle = cos(\boldsymbol{x}, \boldsymbol{y}), \quad \text{if } \|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1 \tag{2.2.1.4}$$

Hyperplanes and affine hyperplanes

If we do not consider the length of the vectors, then the scalar product measures their *similarity*. This fact is very important in order to introduce the notion of hyperplanes, which are a subspaces of dimension n-1. These subspaces are defined with only one vector perpendicular to the hyperplane, termed director or weight vector ($\boldsymbol{w}$). Hence, the equation

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0, \tag{2.2.1.5}$$

splits the space in two different regions, namely positive and negative classes:

$$Positive = \{\boldsymbol{x} : \langle \boldsymbol{w}, \boldsymbol{x} \rangle \geq 0\},$$

$$Negative = \{\boldsymbol{x} : \langle \boldsymbol{w}, \boldsymbol{x} \rangle < 0\}.$$

Moreover, an affine hyperplane is an hyperplane translated to a point $\boldsymbol{x}_0$ in the space, i.e., it does not pass through the origin. Equation (2.2.1.5) is then updated as

$$\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0, \tag{2.2.1.6}$$

which also splits the space in two different regions, as depicted in Figure 2.4.

$$Positive = \{\boldsymbol{x} : \langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \geq 0\},$$

$$Negative = \{\boldsymbol{x} : \langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b < 0\}.$$

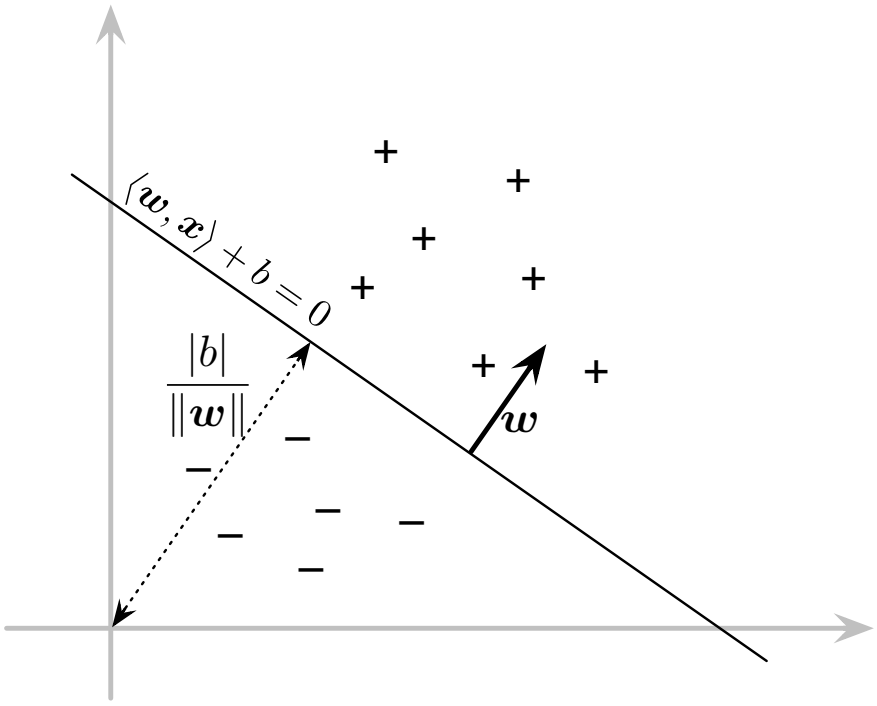


Figure 2.4: Geometric interpretation of hyperplanes and scalar product

Equation (2.2.1.6) can also be represented as

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b \tag{2.2.1.7}$$

Notice also that any function of the family

$$\lambda(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \tag{2.2.1.8}$$

represents the same hyperplane.

Distance to hyperplanes

Let $\boldsymbol{w}$ a director vector of a normalized hyperplane, with norm $\|\boldsymbol{w}\| = 1$, then

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \|\boldsymbol{x}\| \cos(\boldsymbol{w}, \boldsymbol{x}) \tag{2.2.1.9}$$

Therefore, the scalar product is the length of the projection of $\boldsymbol{x}$ in the direction of $\boldsymbol{w}$, i.e., the distance of the point $\boldsymbol{x}$ to the hyperplane determined by $\boldsymbol{w}$ (see Figure 2.5).

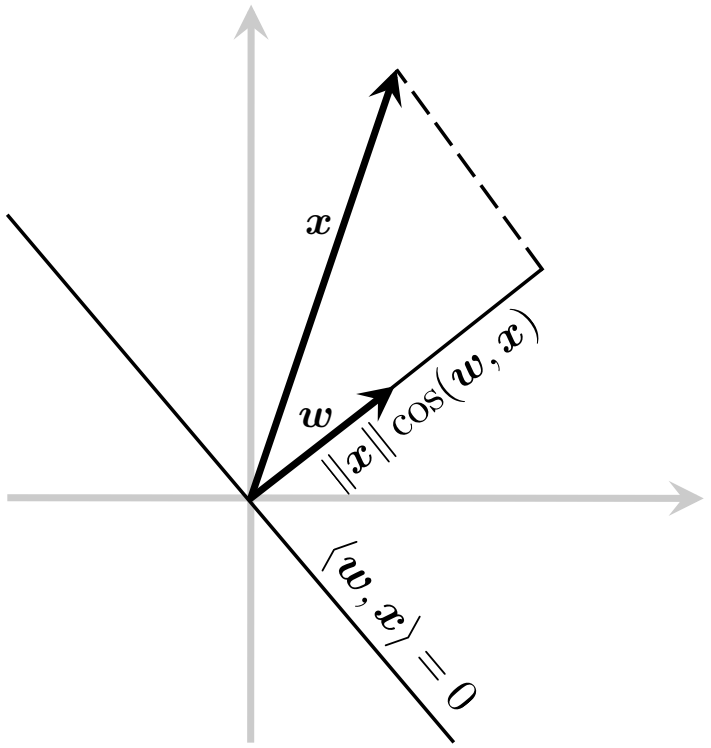


Figure 2.5: Distance to normalized hyperplanes

Generalizing this notion for affine hyperplanes $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, results in the following metric:

$$d(\boldsymbol{w}, b; \boldsymbol{x}) = \frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b\ |}{\|\boldsymbol{w}\|}. \tag{2.2.1.10}$$

Therefore, the split in two regions can be easily obtained through the hypothesis:

$$h(\boldsymbol{x}) = \text{sign}(d(\boldsymbol{w}, b; \boldsymbol{x})) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b). \tag{2.2.1.11}$$

Classification with scalar product

Let a vector space $\mathcal{X}$ endowed with scalar product, and a set of training points $\mathcal{S}$ of size n defined as

$$\mathcal{S} = \{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \{+1, -1\};\ \forall i \in [1, n]\}$$

To classify unseen or test points from $\mathcal{X}$ we can compute the centroids of positive ($\boldsymbol{c}_+$) and negative ($\boldsymbol{c}_-$) examples as follows:

$$\boldsymbol{c}_+ = \frac{1}{n_+} \sum_{i:y_i=+1} \boldsymbol{x}_i \qquad\qquad \boldsymbol{c}_- = \frac{1}{n_-} \sum_{i:y_i=-1} \boldsymbol{x}_i \tag{2.2.1.12}$$

Then, in order to define the affine hyperplane, we compute the director vector $\boldsymbol{w} = \boldsymbol{c}_+ - \boldsymbol{c}_-$. Moreover, we also force the resulting hyperplane to pass through the central point $\boldsymbol{c} = \frac{\boldsymbol{c}_+ + \boldsymbol{c}_-}{2}$ (see Figure 2.6). The hypothesis function can be expressed as follows:

$$\begin{aligned} h(\boldsymbol{x}) &= \operatorname{sign}\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{c} \rangle = \operatorname{sign} \left\langle (\boldsymbol{c}_+ - \boldsymbol{c}_-), \left(\boldsymbol{x} - \frac{\boldsymbol{c}_+ + \boldsymbol{c}_-}{2} \right) \right\rangle \\ &= \operatorname{sign} \left(\langle \boldsymbol{c}_+, \boldsymbol{x} \rangle - \langle \boldsymbol{c}_-, \boldsymbol{x} \rangle - \frac{\langle \boldsymbol{c}_+, \boldsymbol{c}_+ \rangle - \langle \boldsymbol{c}_-, \boldsymbol{c}_- \rangle}{2} \right) \\ &= \operatorname{sign}(\langle (\boldsymbol{c}_+ - \boldsymbol{c}_-), \boldsymbol{x} \rangle + b) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \end{aligned} \tag{2.2.1.13}$$

where

$$b = \frac{\langle \boldsymbol{c}_-, \boldsymbol{c}_- \rangle - \langle \boldsymbol{c}_+, \boldsymbol{c}_+ \rangle}{2} = \frac{\|\boldsymbol{c}_-\|^2 - \|\boldsymbol{c}_+\|^2}{2} \tag{2.2.1.14}$$

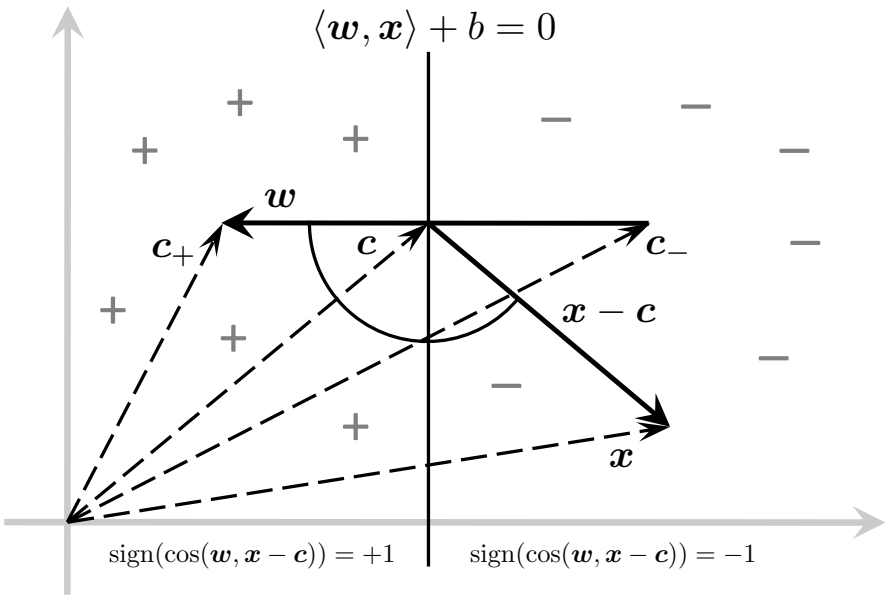


Figure 2.6: Classification with scalar product

However, intuitively, not all the training points should have the same weight. The vectors near the decision boundary should have more relevance in the classification decisions. These *relevant* vectors are the so-called *support vectors*.

2.2.2 Kernel methods

At this point, in order to obtain non-linear solutions, the following step is to take advantage of a kernel function. The key idea is to transform the training points $\boldsymbol{x}_i$ from the input space $\mathcal{X}$ into points $\phi(\boldsymbol{x}_i)$ in a high dimensional space $\mathcal{F}$, which is usually called *feature space*. Then, the assumption is that the training points are linearly separable in this new space. Once defined the transformation function $\phi : \mathcal{X} \to \mathcal{F}$, the classification method based on scalar product can be extended with a kernel function.

Definition 2.1 *A kernel function is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which assigns to each pair of objects from the input space $\mathcal{X}$ a real value obtained through the scalar product of the images of those objects in the feature space $\mathcal{F}$:*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle \tag{2.2.2.1}$$

Obviously, the common core of every kernel method is the kernel function, which is used as the representation mechanism for the input data. This means that even when the input objects do not belong to a vector space, we will be able to apply the algorithm though the implicit kernel transformation, because the scalar product is computed directly in the feature space. This implicit transformation also allows us to apply linear algorithms to obtain non-linear solutions, which results in an extremely powerful feature of these methods.

The kernel function

The kernel functions can be understood as some kind of similarity functions, measuring the similarity between each pair of objects. Where, in this case, the training set $\mathcal{S}$ is represented as a square matrix K of dimension n, which results from the application of the kernel function k to the objects of the input space $\mathcal{X}$:

$$K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j);\ \forall i, j \in [1, n] \tag{2.2.2.2}$$

The application of kernel functions offers several advantages, like the modularity obtained through the isolation of the algorithm with respect to the data representation and even the specific kernel function. The other already mentioned key advantage of kernel functions is the possibility of obtaining complex relations through relatively simple algorithms.

Therefore, thanks to the use of kernels, we do not need to know the image $\phi(\boldsymbol{x})$ for each object, but the result of the scalar product of each pair of objects in the feature space. Actually, in some cases, like the *Gaussian* kernel, the dimension of the feature space is infinite, while the computation of the kernel is relatively simple. In this regard, it essential to guarantee that the kernel function is a valid scalar product in some feature space $\mathcal{F}$.

Definition 2.2 *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is symmetric if*

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{y}, \boldsymbol{x});\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$$

Definition 2.3 *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semi-definite for any set $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ from input space $\mathcal{X}$ and any set of real values $c_1, \cdots, c_n$, with $n > 0$, if*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

From all these definitions, Aronszajn [Aronszajn, 1950] states the following theorem:

Theorem 2.1 *For any function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, symmetric and positive semi-definite, it exists a Hilbert space $\mathcal{F}$ and a function $\phi : \mathcal{X} \to \mathcal{F}$ such that:*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle;\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X},$$

where $\langle \cdot, \cdot \rangle$ represents the scalar product between two points in the Hilbert space $\mathcal{F}$.

Common kernel functions

In this section we will review some of the most common kernel functions for vectorial spaces of the form $\mathcal{X} \subseteq \mathbb{R}^n$.

Linear kernel

As we have introduced in Section 2.2.1, the scalar product is a reasonable approximation for comparing pairs of vectors. Therefore, the linear kernel is the most naive kernel approach, defined in terms of the scalar product:

$$k_L(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{d} x_i y_i, \tag{2.2.2.3}$$

where d is the dimensionality of the input space $\mathcal{X}$. The transformation is this case is the identity function $\phi_L(\boldsymbol{x}) = \boldsymbol{x}$. It is also quite simple to proof that this function fulfills both conditions for kernel functions:

1. It is symmetric:

$$k_L(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle = k_L(\boldsymbol{y}, \boldsymbol{x});\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$$

2. It is positive semi-definite:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k_L(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \left\| \sum_{i=1}^{n} c_i \boldsymbol{x}_i \right\|^2 \geq 0$$

Polynomial kernels

A polynomial kernel function of degree p could be represented as:

$$k_P(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle)^p \tag{2.2.2.4}$$

However, it is commonly used this alternative representation:

$$k_{P'}(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle + r)^p \tag{2.2.2.5}$$

The function ϕ_P associated to Equation (2.2.2.4) transform each input vector in a feature vector with all the monomials of degree p. In the particular case of $\mathcal{X} = \mathbb{R}^2$ and $p = 2$, we have

$$\begin{aligned} k_P(\boldsymbol{x}, \boldsymbol{y}) &= (\langle (x_1, x_2), (y_1, y_2) \rangle)^2 = (x_1 y_1 + x_2 y_2)(x_1 y_1 + x_2 y_2) \\ &= x_1 y_1 x_1 y_1 + x_2 y_2 x_1 y_1 + x_1 y_1 x_2 y_2 + x_2 y_2 x_2 y_2 \\ &= \langle (x_1^2, x_2^2, x_1 x_2, x_2 x_1), (y_1^2, y_2^2, y_1 y_2, y_2 y_1) \rangle = \langle \phi_P(\boldsymbol{x}), \phi_P(\boldsymbol{y}) \rangle; \end{aligned}$$

therefore, we can conclude that $\phi_P(\boldsymbol{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1)$, producing a four-dimensional feature space.

In the example presented in Figure 2.7, we can easily observe the potential of this kind of transformations. In this example, the classification function is non-linear in the input space $\mathcal{X}$. Nevertheless, with an appropriate transformation function like $\phi(\boldsymbol{x}) = \phi((x_1, x_2)) = (x_1^2, x_2)$, we are able to find a linear classification function in the feature space $\mathcal{F}$. Notice also the change in the scale of the horizontal axis (x_1).

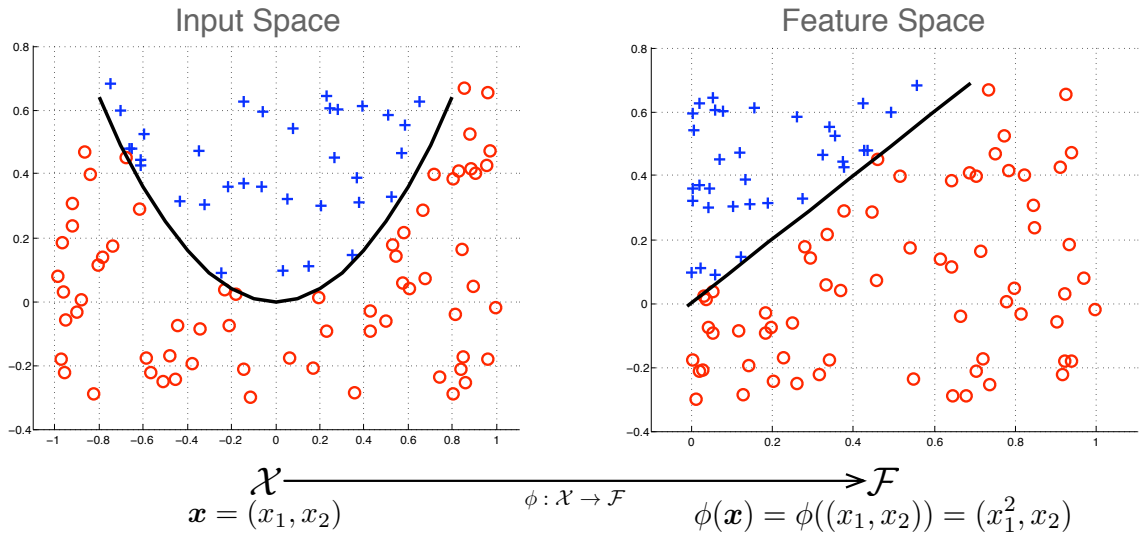


Figure 2.7: Example of non-linear solution in $\mathcal{X}$ with a linear function in $\mathcal{F}$

Gaussian kernel

The Gaussian kernel is also known as *Radial Basis Function* (RBF) kernel, which is often computed as

$$k_G(\boldsymbol{x}, \boldsymbol{y}) = e^{\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{-2\sigma^2}\right)} \tag{2.2.2.6}$$

Notice that in this case we do not know the explicit expression for ϕ_G. This is due to the fact that the feature space for the RBF kernel has infinite dimensions. Moreover, it is also important to emphasize that for polynomial kernels the manual expansion of monomials is computationally very expensive, and even impossible, in most of the cases. In order to overcome this problem we have to take into account that it is not required to know the explicit representation of ϕ, but the procedure to directly compute the scalar product in the feature space $\mathcal{F}$, which is usually known as the *kernel trick* [Aizerman et al., 1964], e.g., as in equations (2.2.2.3), (2.2.2.4), (2.2.2.5) and (2.2.2.6).

2.2.3 Quadratic convex programming

The optimization theory is a branch of mathematics that deals with the characterization of the solution of problems, where a certain objective or cost function f should be optimized, i.e., maximized or minimized. Moreover, these solutions are subject to k constraints g_i that define the feasible region R as

$$R = \{\boldsymbol{w} \in \Omega : g_i(\boldsymbol{w}) \leq 0; \forall i \in [1, k]\}. \tag{2.2.3.1}$$

This section introduces only the essential concepts of *quadratic programming*, a subset of *convex programming*, where the cost function f allows quadratic terms and all the constraints g_i are linear.

Primal problem

There exists not only a collection of efficient strategies, algorithms and implementations to solve optimization problems, but also the necessary and sufficient conditions for a given function to be a solution are formally defined. The general structure of the quadratic optimization problems tackled in this work is:

$$\begin{array}{lll} \min\limits_{\boldsymbol{w}} & f(\boldsymbol{w}) & \boldsymbol{w} \in \Omega, \\ \text{s.t.} & g_i(\boldsymbol{w}) \leq 0 & \forall i \in [1, k]. \end{array} \tag{2.2.3.2}$$

This problem is known as the *primal problem*, in which we try to find the values of the *primal variables*, represented by $\boldsymbol{w}$, that minimize the cost function $f(\boldsymbol{w})$. The most important element of the feasible region defined in Equation (2.2.3.1) is the optimum, which is represented by $\boldsymbol{w}^*$ and that verifies

$$f(\boldsymbol{w}^*) \leq f(\boldsymbol{w}), \quad \forall \boldsymbol{w} \in \Omega.$$

Convexity

The term *convex*, in this case, stands for the fact that there exists only one global optimum, and therefore the search cannot be trapped in a local optimum, as in other machine learning methods like in some *Evolutionary Computation* approaches and in *Artificial Neural Networks*.

Definition 2.4 (Convex set) *A domain or set Ω is convex iif the straight line segment that joins any pair of points $\boldsymbol{x}, \boldsymbol{y}$ of the domain is also contained in that domain:*

$$\Omega \text{ is convex} \Leftrightarrow \forall \boldsymbol{x}, \boldsymbol{y} \in \Omega, \forall \theta \in (0,1) \Rightarrow \theta \boldsymbol{x} + (1-\theta)\boldsymbol{y} \in \Omega.$$

Intuitively, e.g., a solid cube or a solid sphere in $\mathbb{R}^3$ is convex, but any shape with a hollow or dent is not convex. Notice also that in a quadratic programming problem, if the domain Ω is convex, then the feasible region R is also convex because the linear constraints g_i cannot change the convexity of the domain Ω, depicted in the rightmost picture of Figure 2.8.

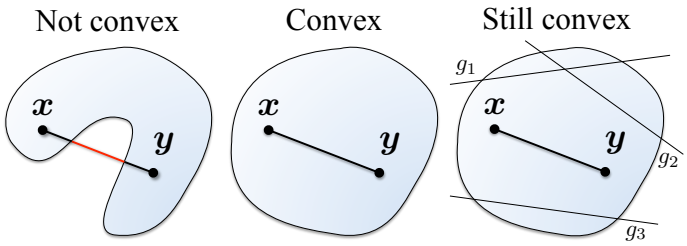


Figure 2.8: Convex domains and linear restrictions

Definition 2.5 (Convex function) *A real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is convex iif the straight line segment that joins the images of any pair of points $\boldsymbol{x}, \boldsymbol{y}$ of its domain $\mathbb{R}^d$ is above the image of any point between $\boldsymbol{x}$ and $\boldsymbol{y}$:*

$$f \text{ is convex} \Leftrightarrow \forall \boldsymbol{x}, \boldsymbol{y} \in \Omega, \forall \theta \in (0,1) \Rightarrow f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}).$$

In other words, a function is convex if and only if the set of points lying on or above its graph is a convex set, as defined in Definition 2.4. The geometric interpretation is presented in Figure 2.9.

Corollary 2.1 *A twice differentiable function f is convex provided its Hessian matrix is positive semi-definite, e.g., a kernel function (see Theorem 2.1).*

Definition 2.6 (Convex problem) *An optimization problem is convex iif the domain Ω, the objective function f and all the constraints g_i are convex.*

Proposition 2.1 *If a function f is convex, then any local minimum $\boldsymbol{w}^*$ is also a global minimum.*

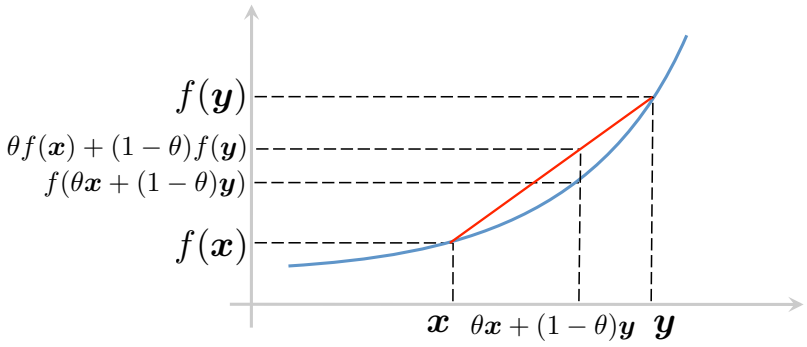


Figure 2.9: Visual interpretation of a convex function

Lagrangian theory

This theory enables the characterization of the solution and provides us a systematic method to obtain it. Initially proposed by Lagrange at the end of the 18th century, it was further developed by Kuhn and Tucker around the middle of the 20th century, introducing the case with inequality restrictions. This theory has two main components: (1) the *Lagrange multipliers* and (2) the *Lagrangian function*, which integrates the restrictions into the objective function.

Definition 2.7 (Lagrangian function) *Given an optimization problem as in Equation* (2.2.3.2)*, with objective function* $f(\boldsymbol{w})$*, defined over the domain* $\Omega \subseteq \mathbb{R}^d$ *and subject to* k *restrictions* $g_i(\boldsymbol{w}) \leq 0$*, the corresponding Lagrangian function is defined as*

$$L(\boldsymbol{w}, \boldsymbol{\alpha}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}), \tag{2.2.3.3}$$

where the coefficients α_i *are termed Lagrange multipliers or dual variables, and must have non-negative values.*

Furthermore, the Lagrange multipliers will, intuitively, show us the relevance of the constraints: the higher the value of the multiplier, the higher the difficulty to fulfill it. The Lagrangian function has, in turn, a *saddle* point with respect to primal and dual variables at the solution.

Duality

Definition 2.8 (Dual problem) *The dual problem of the primal problem, defined in Equation* (2.2.3.2)*, is then reformulated as follows:*

$$\begin{aligned} \max \quad & W(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w} \in \Omega} L(\boldsymbol{w}, \boldsymbol{\alpha}) \\ s.t. \quad & \alpha_i \geq 0. \end{aligned} \tag{2.2.3.4}$$

The key advantage of this dual problem, which usually has simpler constraints than the primal, is that solving the dual we also obtain the solution of the primal, under certain circumstances.

In order to solve the primal problem through its dual, there exists a set of conditions to be fulfilled, known as *Karush-Kuhn-Tucker* (KKT) conditions.

Theorem 2.2 (Karush-Kuhn-Tucker conditions) *Given a primal optimization problem* (2.2.3.2)*, where both the domain* Ω *and the objective function* f *are convex, and the constraints* g_i *are affine functions. The necessary and sufficient conditions for a point* $\boldsymbol{w}^*$ *to be an optimum are the existence of* $\boldsymbol{\alpha}^*$ *such that*

$$\frac{\partial L(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)}{\partial \boldsymbol{w}} = 0 \tag{2.2.3.5a}$$

$$\alpha_i^* g_i(\boldsymbol{w}^*) = 0, \quad \forall i \in [1, k], \tag{2.2.3.5b}$$

$$g_i(\boldsymbol{w}^*) \leq 0, \quad \forall i \in [1, k], \tag{2.2.3.5c}$$

$$\alpha_i^* \geq 0, \quad \forall i \in [1, k]. \tag{2.2.3.5d}$$

The first condition (2.2.3.5a) is directly concluded from the fact that W is defined as the infimum of the Lagrangian function in Equation (2.2.3.4), point where the partial derivative with respect to $\boldsymbol{w}$ must be equal to zero.

The second one (2.2.3.5b), termed *complementary condition*, guarantees that the optimums of both primal and dual are the same: $f(\boldsymbol{w}^*) = W(\boldsymbol{\alpha}^*)$. Hence, *active constraints*, with $g_i(\boldsymbol{w}^*) = 0$, can have a non-zero Lagrange multiplier $\alpha_i^* \geq 0$. While, *inactive constraints*, with $g_i(\boldsymbol{w}^*) < 0$, imply $\alpha_i^* = 0$. This means that the actual number of variables involved may be significantly fewer, i.e., it guarantees the *sparsity* of the solution.

Finally, conditions (2.2.3.5c) and (2.2.3.5d) are the constraints of the primal (2.2.3.2) and dual (2.2.3.4) problems respectively. These conditions guarantee the *feasibility* of the solution.

2.2.4 Binary Classification with SVM

After a brief introduction of the foundations of scalar product classification, kernel methods and convex optimization in the previous sections, we can know define SVMs as linear learning machines that use a dual representation and operate in a kernel-induced feature space $\mathcal{F}$; i.e., the model is a linear function in some feature space, implicitly defined by the kernel.

The first proposals of SVM were initially designed for classification problems with only two classes (see [Vapnik, 1995; Cortes and Vapnik, 1995; Boser et al., 1992]). Therefore, this section will introduce the basics of SVMs as they were originally conceived.

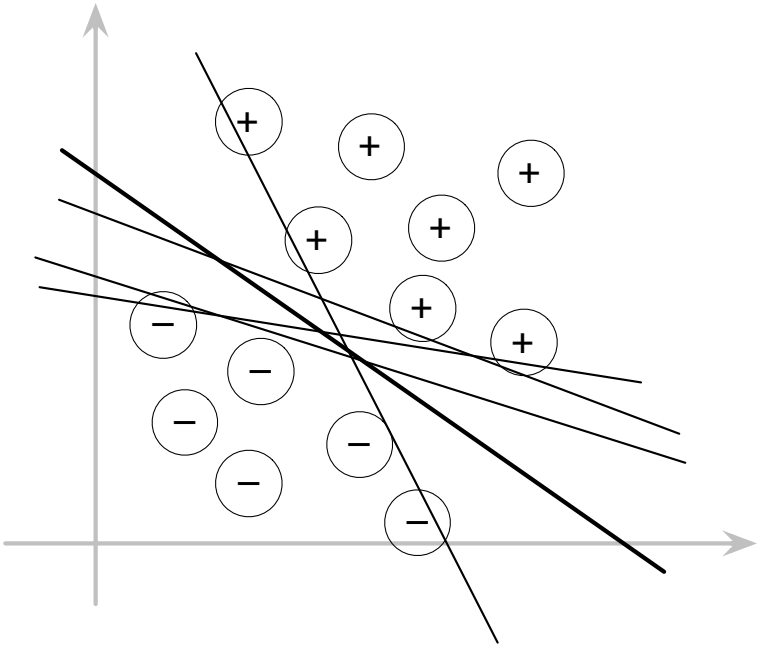


Figure 2.10: Example of multiple solutions for a single problem

Margin maximization

Let a vector space $\mathcal{X} \subseteq \mathbb{R}^d$ and a set of training points $\mathcal{S}$ of size n defined as

$$\mathcal{S} = \{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \{+1, -1\};\ i = 1, \cdots, n\} \tag{2.2.4.1}$$

The binary classification can be solved through a linear function $f : \mathcal{X} \to \mathbb{R}$. This objective function f should assign positive values for the class $+1$ and negative for the class -1, such that:

$$f(\boldsymbol{x}_i) = \begin{cases} >= 0 & \text{if } y_i \in +1 \\ < 0 & \text{if } y_i \in -1 \end{cases} \tag{2.2.4.2}$$

A possible hypothesis function and prediction are:

$$h(\boldsymbol{x}_i) = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b \tag{2.2.4.3}$$

$$y_i' = \text{sign}(h(\boldsymbol{x}_i)) \tag{2.2.4.4}$$

The main objective is to reduce the structural risk (SRM) through the maximization of the margin of the model with respect to the classes. It is straightforward to demonstrate that for a specific training set $\mathcal{S}$ with two linearly separable classes, we can obtain several hyperplanes that classify correctly all the examples. However, as it was stated before, the SVM methods try to minimize the structural risk, rather than the empirical risk.

This means that ERM techniques, like perceptrons, will stop as soon as they find any hyperplane that classifies correctly all the examples. Figure 2.10 shows many possible

solutions of this kind; however, only the wider one has the maximal margin with respect to both classes. Actually, the margin is the same for both. In that picture, the training examples are surrounded by a circle that depicts the possible noise variations, showing that the *non-optimal* solutions are more likely to misclassify them.

Therefore, the primary aim is to search for the hyperplane with the maximal margin among all the possible solutions. The key advantage of SVM is that this search can be performed optimally due to the fact that it is quadratic convex problem, i.e., without local optima but with a global one, as already introduced in Section 2.2.3.

Two kinds of margin

There are two possible formulations of the margin, the *geometric margin* (γ_g) and the *functional margin* (γ_f). In fact, these two margins are closely related, requiring to fix either of both in order to optimize the other. This is because there exists a family of functions which defines the same hyperplane (recall Equation (2.2.1.8) and Figure 2.4).

The functional margin is the minimum difference between applying the function f to the examples of the positive and negative classes:

$$\gamma_f = \min_{+}(f(\boldsymbol{x}_+)) - \max_{-}(f(\boldsymbol{x}_-)) \tag{2.2.4.5}$$

On the other hand, the geometric margin is the distance between the hyperplane and the closer example/s of each class, as depicted in Figure 2.11:

$$\gamma_g = \min_{+}(d(\boldsymbol{w}, b; \boldsymbol{x}_+)) + \min_{-}(d(\boldsymbol{w}, b; \boldsymbol{x}_-)) \tag{2.2.4.6}$$

The key idea is to minimize the risk of overfitting by choosing the maximal margin hyperplane in the feature space. In this way, SVMs control the capacity by increasing the margin, rather than reducing the degrees of freedom. Notice also that we need to fix either of both kinds of margin. Then, for example, if we want to maximize the functional margin we need to restrict the search to normalized hyperplanes, i.e., with $\|\boldsymbol{w}\| = 1$, because otherwise it would be very easy to increase the functional margin by simply increasing λ in Equation (2.2.1.8).

Nevertheless, SVMs implementations usually work with *canonical hyperplanes* in order to maximize the geometric margin. These canonical hyperplanes fix the functional margin, defined in (2.2.4.5), by restricting $\boldsymbol{w}$ and b as follows:

$$\min_{+}(\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + b) = +1 \qquad \max_{-}(\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + b) = -1 \tag{2.2.4.7}$$

From (2.2.1.10) and (2.2.4.6), we have that the geometric margin for any canonical hyperplane is redefined as:

$$\gamma_g = \min_{+}\left(\frac{|\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + b|}{\|\boldsymbol{w}\|}\right) + \min_{-}\left(\frac{|\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + b|}{\|\boldsymbol{w}\|}\right) = \frac{|+1|}{\|\boldsymbol{w}\|} + \frac{|-1|}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|} \tag{2.2.4.8}$$

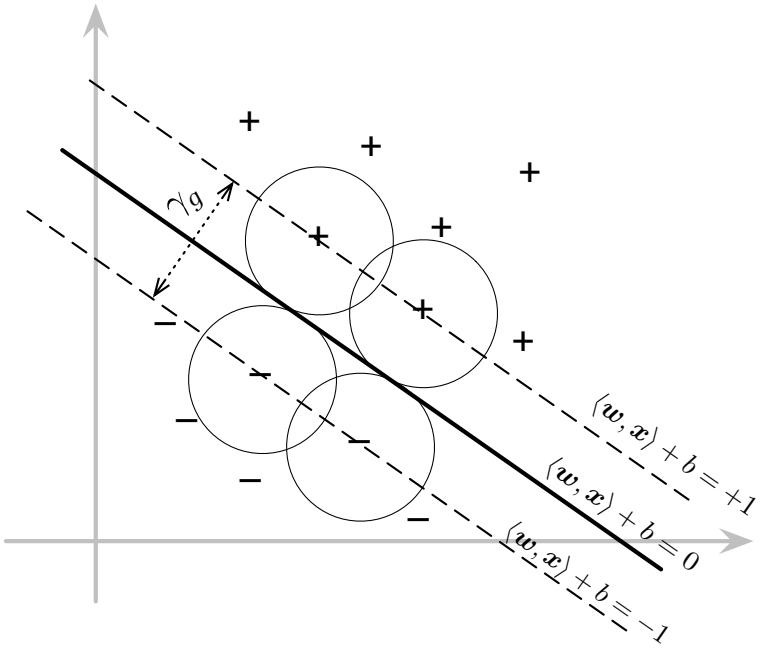


Figure 2.11: Geometric margin maximization

Optimization problem

Therefore, in order to maximize the geometric margin we need to minimize the norm of the weight vector $\boldsymbol{w}$, subject to searching only for canonical hyperplanes. The result is a primal quadratic optimization problem, which is a primal problem as presented in 2.2.3.2:

$$\begin{array}{lll} \min\limits_{\boldsymbol{w}} & \|\boldsymbol{w}\| & \\ \text{s.t.} & (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq +1 & \forall \boldsymbol{x}_i \in +1, \\ & (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq -1 & \forall \boldsymbol{x}_i \in -1. \end{array} \tag{2.2.4.9}$$

Maximal margin classifier

Given that $\|\boldsymbol{w}\|^2 = \langle \boldsymbol{w}, \boldsymbol{w} \rangle$ we can change this version for another equivalent:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w},b} & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle \\ \text{s.t.} & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1, \quad i = 1, \ldots, n \end{array} \tag{2.2.4.10}$$

Where the class $y_i \in \{+1, -1\}$ is used to rewrite the restrictions in a more compact way, and the factor of $1/2$ is used for mathematical convenience. The dual is:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{2.2.4.11}$$

The weight vector can be directly computed as a combination of some training points, called *support vectors*, which are associate with values $\alpha_i > 0$. The term b, which does not appear in the dual, is obtained from Equation (2.2.4.7).

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i \qquad b = -\frac{\max_{-}(\langle \boldsymbol{w}, \boldsymbol{x}_-\rangle + b) + \min_{+}(\langle \boldsymbol{w}, \boldsymbol{x}_+\rangle + b)}{2} \tag{2.2.4.12}$$

Notice also that through the kernel trick we can substitute the dot product in Equation (2.2.4.11) by any kernel function.

Soft margin classifier

In order to provide some flexibility for points that can not be correctly separated, it was introduced a new regularization term within the primal problem [Cortes and Vapnik, 1995; Vapnik, 1995] as follows:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, n. \end{aligned} \tag{2.2.4.13}$$

Where $\boldsymbol{\xi}$ represents the vector of errors committed over each of the examples. The corrsponding dual is:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n. \end{aligned} \tag{2.2.4.14}$$

Notice that the dual problem is almost the same that for the maximal margin classifier, but changing the constraints over the dual variables α_i. Both $\boldsymbol{w}$ and b are calculated as in Equation (2.2.4.12).

2.3 Feature reduction

In machine learning a classification problem consists on a series of labeled examples that are used to induce a model able to classify new instances into a predefined set of classes. Each example is represented with a feature vector that describes its properties and by the corresponding class. Some of these attributes may be redundant or irrelevant and may have a negative impact on the accuracy of the classifier. Choosing the relevant features for representing the objects in each task is a broadly studied problem in the literature [Kohavi and John, 1997; Yang and Pedersen, 1997; Guyon and Elisseeff, 2003].

The three main objectives of feature reduction are:

- Improving the prediction and generalization capability of the classifiers.
- Providing faster and more cost effective predictors.
- Improving the interpretability of the acquired models.

Two key concepts regarding variable reduction are *relevance* and *redundancy*. Learning is harder when we have less/more input variables than needed. Identifying the relevant features for representing the problem helps to avoid overfitting, and usually improves the generalization power of the obtained models. By reducing the input dimensionality the learning process is normally speeded up. It also soothes the effect of the curse of dimensionality, typical of high-dimensional problems. Reducing the number of characteristics to be measured for each instance also reduces data acquisition costs. As regards model's interpretability, a better understanding of the underlying data and the relations between features can be achieved.

Recall that the typical Bag Of Words (BOW) representation that is used when dealing with textual data, usually yields to high dimensional representations. As a preliminary stage, the set of documents is analyzed to extract the corpus. That is a series of terms that constitute the global vocabulary of the data set. Then, a series of characteristics are measured for each term in the vocabulary. In sentiment analysis related problems the number of resulting terms is expected to be very large, and as a consequence, the number of features. That is the reason of the key role that feature reduction has in text related problems, and in particular in sentiment analysis tasks.

There exist two well established strategies to feature reduction:

- Feature reduction by *extraction*, in which the resulting features are not a subset of the original set, but a combination or transformation of the initial ones.
- Feature reduction by *selection*, in which the resulting features are a subset of the initial set.

In this work we have focused on *feature selection*, and the next section is intended to emphasize some of its characteristics.

2.3.1 Feature selection

In the feature subset selection problem, a learning algorithm is faced with the problem of selecting a relevant subset of features upon which to focus its attention, while ignoring the rest [Kohavi and John, 1997]. It can be seen as an optimization problem in which the search space is comprised of all the possible subsets of features, and the objective function consists in maximizing the accuracy of the classifier. The number of feasible solutions is equal to the cardinality of the power set of the original features set minus the empty set, that is $2^n - 1$ solutions with n being the dimensionality of the problem. As the dimension of the search space grows exponentially according to the number of features, an exhaustive search looking for the global optimum solution is not practical from a computational point of view.

If we consider what is evaluated to perform the subset selection, there are two common approaches:

- **Univariate** or individual evaluation, assess individual features and assign them weights according to their degrees of relevance. Usually a rank is produced and the top ranked features are selected. It is efficient for high-dimensional data but does not handle redundant features properly. It's principal drawback is that it does not take into account the relations among features, as they are isolated and then evaluated.
- **Multivariate** or subset evaluation, assess the quality of candidate feature subsets that are produced based on a certain search strategy. Each candidate subset is evaluated with a given measure and compared to the previous best one, replacing it when the evaluation yields to better results. As already hinted, it is not possible to evaluate all the potential feature subsets, so *heuristic* search strategies in the form of *forward* or *backward* selection, or *evolutionary* strategies such as genetic algorithms, are generally applied.

On the other hand, regarding the procedure that is used to evaluate the candidate solutions, there also exist two well-known approaches:

- **Filters**, perform the selection as a preprocessing step based on properties of the data itself and independent of the induction algorithm to be used afterwards. Filter methods are fast but generally produce worse accuracy results because they are not tailored to a specific inducer. Information Gain (IG), or Fast Correlation Based Filter (FCBF) are examples of filters.
- **Wrappers**, a search for an optimal set of features is made using the induction algorithm as a black box. The estimated future performance of the algorithm is the heuristic guiding the search. While wrapper techniques perform better on terms of accuracy, they are computationally expensive. Genetic algorithms have been succesfully applied in this context [Cantu-Paz, 2004; Abbasi et al., 2008].

As it follows we are going to describe in detail one of the filters we have used during this work.

2.3.2 Information gain

Intuitively, information gain is a measure that indicates how informative a feature is when trying to predict the value of the class. Formally, the information gain of a feature F with respect to a class C, can be defined as the difference in the class C's entropy after knowing the values of the feature F.

Entropy is a measure of *impurity*, uncertainty or disorder associated with a random variable. For example, imagine a series of coin tosses. As prior to the toss we do not know if it will come up a head or a tail, the associated uncertainty (entropy) to the throw is maximum. If the coin is composed of two heads or two tails, the associated entropy is 0, as we are in position to predict the outcome of the toss. Most random variables appearing in the real world present an entropy somewhere in the middle.

It can be defined the entropy H of a discrete attribute F with values $\{f_1, ..., f_n\}$ as

$$H(f) = -\sum_{i=1}^{n} p(f_i) \log_b p(f_i), \tag{2.3.2.1}$$

where p is the probability distribution of the variable F and b the logarithm's base. In the case of any $p_i = 0$ the value of the addend $0 \times \log_b 0$ is considered to be 0.

A high entropy value means that the probability distribution of F is uniform (boring), and a low value indicates that F follows a variable distribution (peaks and valleys) otherwise.

Specific conditional entropy $H(C|F = f_i)$ is the entropy of C among those instances in which F has value f_i.

Definition 2.9 *Conditional entropy of an attribute C with respect to other attribute F is calculated as*

$$H(C|F) = \sum_{j=1}^{m} p(F = f_j) H(C|F = f_j), \tag{2.3.2.2}$$

with m being the number of possible values of F and p the probability of F taking the value f_j.

Intuitively, conditional entropy can be understood as the expected number of bits to transmit C if both sides will know the value of F.

Definition 2.10 *Information gain of an attribute F with respect to a class C is calculated as*

$$IG(C|F) = H(C) - H(C|F). \tag{2.3.2.3}$$

Information gain is the number of bits that could be saved, in average, when transmitting C if both ends of the line knew F. It measures the closeness to class C knowing the value of the attribute F. Its domain is on the interval $[0, 1]$, with 1 meaning that F is perfect for predicting C and 0 useless.

Chapter 3

Previous work

Nowadays a huge volume of information is available on the Internet. With the objective of better organizing all this information for the users, a key research area has focused on automatic *text categorization* during the last years. The principal task consists in classifying each document with respect to their subject matter, e.g., *sports*, *politics* and so on. It is called *topic categorization*, and a series of key words or topics are assigned to each document as a mechanism to summarize its content. These kind of techniques work with *facts* (objective information) assuming they are true, independently of the author or the context they appear in.

Nevertheless, with the appearance of online services that allow users to share their sentiments and emotions, the amount of *opinionated* (subjective) information has started to grow quickly. Most of the information available today is subjective rather than objective. Thus, traditional text categorization techniques are no longer appropriate, because opinion is often expressed in a more subtle manner. Nigam and Hurst [2004] reported that only 3% of USENET[1] sentences contained topical information. Moreover, Subasic and Huettner [2001] already showed that web content is well provided with sentiment related information. Sentiment analysis field emerges to deal with this kind of subjective information.

Sentiment analysis attempts to identify and analyze opinions and emotions. Hearst [1992] and Wiebe [1994] were the first authors proposing to analyze texts rich in opinions. Following the idea present in [Abbasi et al., 2008], it can be established a taxonomy trying to describe and summarize the wide range of works in the sentiment analysis field.

This taxonomy is presented in Table 3.1. According to it we can analyze the sentiment analysis problem with respect to the various tasks that are performed, kind of features that are used to represent the information, concrete techniques used to solve the problem and application domains.

[1] USENET is the acronym of **Use**rs **Net**work, a global discussion distributed system on the Internet.

Tasks		
Category	**Description**	**Label**
Classes	Positive/negative sentiments or objective/subjective texts	C1
Level	Document or sentence level classification	C2
Features		
Category	**Examples**	**Label**
Syntactic	Words, POS tag, n-grams, phrase patterns, characters	F1
Semantic	Polarity tags, semantic orientation	F2
Link Based	Web links, send/reply patterns, document citations	F3
Stylistic	Lexical and structural measures of text style	F4
Techniques		
Category	**Examples**	**Labels**
Machine Learning	SVM, naive Bayes, etc.	T1
Link Analysis	Citation analysis and message send/reply patterns	T2
Similarity Score	Phrase pattern matching, frequency counts, etc.	T3
Domains		
Category	**Description**	**Label**
Reviews	Product, movie, and music reviews	D1
Web Discourse	Web forums and blogs	D2
News Articles	Online news articles and web pages	D3

Table 3.1: A taxonomy of the sentiment analysis problem

3.1 Sentiment analysis tasks

Traditionally there have been diverse tasks related with sentiment or opinion analysis. One of the first related research areas consisted in classifying documents according to their *source* or *source style*. Statistic techniques were used in order to detect stylistic variations in the documents [Biber, 1988]. Examples include trying to distinguish among authors, publishers (e.g., *The New York Times* vs. *The Daily News*) [Argamon-Engelson et al., 1998], or even among language styles used by the authors (e.g., high-brow or low-brow). Determining the *genre* of the texts is also another related task; subjective genres as "*editorial*" is one of the possible categories [Karlgren and Cutting, 1994].

As previously stated in section 1.2, one of the principal tasks regarding sentiment analysis is the *objective/subjective* text classification. Different approaches have been proposed both at the document and at the sentence level. The first study proposed by Wiebe [1994] was aimed at tracking the psychological point of view in narrative. In narrative texts, like novels, it was intended to determine whether the text makes reference to beliefs of the characters or to facts from the story, such as landscape or scenery descriptions. An algorithm based on detecting regularities in point of view manipulation is proposed. Afterwards, Wiebe et al. [1999] described a subjective/objective sentence level classifier using the naive Bayes algorithm. The

presence or absence of a series of syntactic clauses (pronouns, adjectives, verbs, adverbs), punctuation symbols and sentence position tags were used as features. An important result reported by Hatzivassiloglou and Wiebe [2000] is that *adjectives* are the most important part of speech regarding a document's sentiment. In [Wiebe, 2000] lexical features were added to the already mentioned presence/absence of syntactic categories. More recently, it was described a document level subjectivity classifier using the k-nearest neighbours algorithm, based on counting the appearances of subjetive words and sentences in each document [Wiebe et al., 2004].

Certain psychological studies discovered measurable associations between a series of words and diverse emotional states [Bradley and Lang, 1999]. Hatzivassiloglou and McKeown [1997] described an unsupervised learning method able to learn positive and negative oriented adjectives with an approximate accuracy of 90%. Later on, Turney [2002] used a little subset ("*excellent*" and "*poor*") of those semantically oriented adjectives, to automatically calculate the semantic orientation of sentences by using pointwise mutual information with the `AltaVista` search engine. Document level polarity classification was then performed by sentence polarity aggregation. A more direct approach was presented in [Pang et al., 2002]. A series of machine learning techniques were directly applied to classify movie reviews into positive and negative.

With respect to the nature of the classes we want to predict (C1), the most common problem is a two class problem involving the *positive* and *negative* classes [Pang et al., 2002; Turney, 2002; Dave et al., 2003]. A variation that has been already discussed consists in a binary classification problem with the classes being objective and subjective [Wiebe et al., 2004]. A closely related problem is affect classification which attempts to classify emotions instead of sentiments. Emotions are expressed with a series of mood states such as happiness, sadness, anger, etc. It is a multiclass classification problem [Subasic and Huettner, 2001; Mishne, 2005].

However, not all the tasks involve carrying out a classification. There exist works in which the goal is not to predict the expressed sentiment as positive or negative, but to predict the numerical rating provided by the reviewer. It is an *ordinal regression* problem that, in general, is more complicated than the binary classification one. Pang and Lee [2005] addressed the problem of "*rating-inference*" as trying to determine an author's evaluation with respect to a multi-point scale (e.g., one to five "stars"). The existence of several different degrees of similarity between class labels significantly complicates the problem. For example, "*three stars*" is intuitively closer to "*four stars*" than to "*one star*". A radically different approach consists in representing the local sentiment flow of the document. Mao and Lebanon [2007] used Conditional Random Fields (CRF) in order to obtain a graphical representation (see Figure 3.1) of the sentiment flow along the document. One of its advantages is that the graphical output is very enjoyable and intuitive for the final user. They also proposed a method to calculate the global sentiment of the document from the local sentiment representations.

A method aimed at mining and summarizing all the customer reviews of a product was proposed by Hu and Liu [2004b]. This summarization task is different from traditional text summarization because they only mine the features of the product on which the customers have expressed their opinions and whether the opinions are

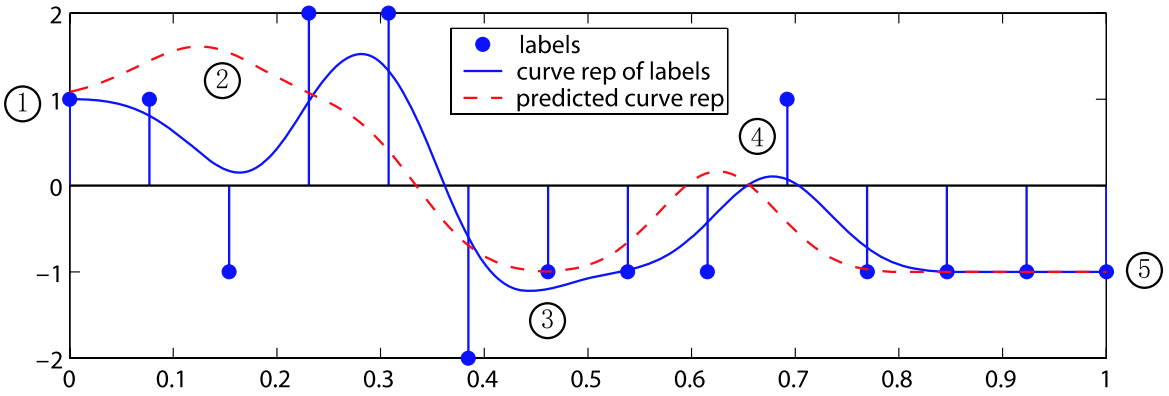


Figure 3.1: Sentiment flow and its smoothed curve representation. The blue circles indicate the labeled sentiment of each sentence. The blue solid curve and red dashed curve are smoothed representations of the labeled and predicted sentiment flows.

positive or negative. The task is basically performed in three steps: 1) mining product features that have been commented by customers; 2) identifying opinion sentences and determining whether they are positive or negative; 3) summarizing the results. An example of a summary concerning a digital camera is presented in Figure 3.2.

*Digital_camera_*1:
Feature: **picture quality**
Positive: 253
<individual review sentences>
Negative: 6
<individual review sentences>
Feature: **size**
Positive: 134
<individual review sentences>
Negative: 10
<individual review sentences>
...

Figure 3.2: An example summary

With respect to the classification level (C2), it can be performed at the document [Pang et al., 2002; Turney, 2002; Mullen and Collier, 2004] or at the sentence level [Pang and Lee, 2004; Mullen and Collier, 2004]. There has also been work on phrase level categorization in order to capture multiple sentiments that may be present within a single sentence [Wilson et al., 2005].

3.2 Sentiment analysis features

According to the Table 3.1, there are four kind of features that have been traditionally used in sentiment analysis works. These include *syntactic*, *semantic*, *link based*, and *stylistic* features. Syntactic and semantic are the most common used ones.

3.2.1 Syntactic features

Among syntactic features it usual to perform a bag of words representation including features like word *presence/frequence*, *n-grams* [Pang et al., 2002; Dave et al., 2003; Abbasi et al., 2008], *character* information [Abbasi et al., 2010], part of speech (POS) tags [Pang et al., 2002; Yi et al., 2003] and punctuation symbols. Additional syntactic features include phrase patterns, which make use of POS tag n-gram patterns [Yi et al., 2003; Fei et al., 2004]. Fei et al. [2004] reported that usually phrase patterns like "n+aj" (noun + positive adjective) represented positive sentiment orientation while "n+dj" (noun + negative adjective) often expressed negative sentiment.

Normalization is also a common practice. It is referred to the observation that many different strings of characters often convey essentially identical meanings. *Stemming* is possibly the most known case of normalization [Dave et al., 2003]. It reduces inflected words to their stem or root form. Using collocations (tags) to substitute *hapax legomena* words (words that appear a single time in a given corpus) is another normalization technique [Wiebe et al., 2004]. Collocations are also used to replace certain parts of fixed n-grams with general word tags, thereby also creating n-gram phrase patterns [Wiebe et al., 2004]. For example, all the bigrams containing a unique adjective followed by the preposition "*as*" are substituted by a special tag.

3.2.2 Semantic features

Semantic features incorporate manual/semi-automatic or fully automatic *annotation* techniques to add polarity or affect intensity related scores to words and phrases. Hatzivassiloglou and McKeown [1997] proposed a method to calculate the semantic orientation (SO) of each word and phrase based on mutual information. It was later extended by Turney [2002], who calculated the score by computing the mutual information between each sentence and the word "*excellent*", and subtracting the mutual information between the same phrase and the word "*poor*". By aggregating the individual scores, a final score for the whole document can be obtained.

A different approach consists in using manual or semi-automatic generated *sentiment lexicons* [Tong, 2001; Fei et al., 2004; Wilson et al., 2005]. Typically a set of seed words is automatically generated and then manually filtered. Polarity and intensity information is then incorporated to each term. However, Pang et al. [2002] reported that human intuition may not be adequate when selecting good indicator features for the sentiment classification task. On one hand, they asked two computer science students to (independently) choose good indicator words for positive and negative sentiments in

movie reviews. On the other hand, they selected a word list based on frequency counts in the entire corpus. After using the resulting lists to classify each review as positive or negative, they found that human based lists performed significantly worse than the frequency count list.

Appraisal groups is another effective technique for annotating words and sentences with semantic scores. `WordNet`[2] is used for generating the initial set of words. After a manual filtering, each expression is manually classified into various appraisal classes. These classes include attitude, orientation, graduation, and polarity of phrases. Afterwards, these taxonomies are combined with standard bag of words features for the final evaluation [Whitelaw et al., 2005]. The authors report a high accuracy over a movie review corpus, outperforming [Pang et al., 2002; Turney, 2002; Mullen and Collier, 2004].

3.2.3 Link based features

Link based features are used to analyze the effect of links and citations in sentiment tasks. A hypothesis is that documents that link to each other or share citations usually express a similar sentiment. Efron [2004] found evidences of opinion web pages heavily linking to each other often shared similar sentiments. In contrast, Agrawal et al. [2003] observed the exact opposite behaviour. They analyzed a USENET newsgroup discussing issues such as abortion and gun control, and noticed that forum replies tended to be antagonistic. Due to the limited usage of link-based features, it is unclear how effective they may be for sentiment classification [Abbasi et al., 2008].

3.2.4 Stylistic features

So far, usage of stylistic features in sentiment analysis tasks has been very limited. These includes lexical and structural attributes such as word length, vocabulary richness measures or special character count appearances. Wiebe et al. [2004] considered hapax legomena for effectively discriminate between objective and subjective classes. They observed a noticeably higher presence of unique words in subjective texts as compared to objective documents. As Wiebe reported, "Apparently, people are creative when they are being opinionated".

Examples of other works having used stylistic features include [Gamon, 2004], in which sentence length was used for predicting sentiments in user feedback data, and [Mishne, 2005], where the number of words per message and per sentence was employed for analyzing blog data. It is still not clear to what extent stylistic features are useful in opinion mining tasks. Moreover, authors like Pang et al. [2002] and Gamon [2004] reveal that relevant features may not be trivial at a glance, and that even though adjectives have been remarked as important throughout the literature, stylistic features may present some hidden structure valuable for the classification task.

[2]WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

3.3 Sentiment analysis techniques

Previously used techniques can be classified into three categories: machine learning algorithms, link analysis methods, and score based approaches.

Existing supervised learning methods can be readily applied to sentiment classification, e.g., *support vector machines* (SVM) [Pang et al., 2002; Dave et al., 2003; Mullen and Collier, 2004], *naive Bayes* [Pang et al., 2002; Dave et al., 2003; Efron, 2004], etc. These kind of techniques are typically used in conjunction with a bag of words representation and syntactic or stylistic features. Support vector machines is, arguably, the most suitable supervised learning algorithm for dealing with textual data. The reason is that SVM with linear kernel gets benefited from the usually high number of features and the sparse representation; classifying in high dimension tends to be linearly separable. In fact, SVM has been reported to outperform other classifiers as naive Bayes in movie review domains [Pang et al., 2002].

Studies using link based features have often utilized link analysis. Efron [2004] used cocitation analysis for sentiment classification of web site opinions while Agrawal et al. [2003] used message reply link structures to classify sentiments in USENET newsgroups. An obvious limitation of link analysis methods is that they are not effective where link structure is not clear or links are sparse.

Score based approaches are generally complementary to semantic features. These techniques usually determine the global sentiment of a message by summing up a series of individual scores previously assigned to words/sentences or phrases. In phrase pattern matching strategies [Yi et al., 2003; Fei et al., 2004], a series of patterns are manually created and labeled as positive or negative. Each time a sentence matches a positive pattern, the global document score is incremented in one unit. The same operation is performed with negative patterns. The global sentiment of each document is whether positive or negative depending on the final obtained sum. Hatzivassiloglou and McKeown [1997], and Turney [2002] use the same procedure but making use of semantic orientation scores for each sentence. Score based methods have also been used for affect analysis where the affect features present within a document are scored based on their degree of intensity for a particular emotion class [Subasic and Huettner, 2001].

3.4 Sentiment analysis domains

Sentiment analysis has been applied in diverse domains including movie and product reviews, web discourse (blogs, forums), news articles, social networks like `Twitter`, stock market applications and so on.

Reviews are usually related to products, movies or music [Morinaga et al., 2002; Pang et al., 2002; Turney, 2002; Hu and Liu, 2004a]. Movie reviews has been reported as a specially difficult domain, since reviewers often not only express their opinions about the movie, but also describe the plot (a great movie with a "*dark*" plot for example), and express their previous expectations about both the movie and actors (typical case of a good actor trapped in a bad movie). It is is difficult to identify the concrete parts

of the review in which the author is actually giving it's opinion about the movie itself. Product reviews are also quite tricky, since a single review can feature positive and negative sentiments about distinct characteristics of the product.

Web discourse analysis comprises the evaluation of blogs, forums and newsgroups [Agrawal et al., 2003; Efron, 2004]. Sentiment is usually assessed with respect to a specific topic, e.g., abortion, politics, gun control and so on. News articles have also been used to evaluate the performance of sentiment analysis approaches [Yu and Hatzivassiloglou, 2003; Yi et al., 2003].

`Twitter` has been used a significant source for obtaining sentiment datasets. Pak and Paroubek [2010] looked for tweets containing "*happy*" and "*sad*" emoticons, and were respectively labeled as positive or negative. Recall that the appearance of a "*happy*" emoticon does not necessarily mean the tweet is actually positive. Other authors have tried to predict the stock market movement by analyzing mood in `Twitter`. Bollen et al. [2011] reported that `Twitter` mood, specially the calmness index in GPOMS[3], helps to predict daily up and down changes in the closing values of the Dow Jones Industrial Average. `Twitter` has also been used for predicting movies box office revenues after the film has been released [Asur and Huberman, 2010].

3.5 Feature selection in sentiment analysis

When using machine learning algorithms in sentiment analysis problems, the document representation is usually performed with a large number of features. Even tough a very diverse kind of features have been proposed throughout the literature, little emphasis have been placed in using techniques to select the most relevant and valuable features in each representation. Having in mind the usually high dimensionality of these problems, we think that feature selection is an important task independently of the chosen representation.

Among the few authors that have applied feature reduction techniques we can highlight Gamon [2004] and Yi et al. [2003]. They both employed log likelihood ratio as measure in order to estimate the predictiveness capability of each feature. The top n ranked features were selected, with n ranging from 1000 to 40,000. Abbasi et al. [2008] proposed a feature selection wrapper based on a genetic algorithm called EWGA (Entropy Weighted Genetic Algorithm). Information gain is used both at the crossover and mutation operators to ensure that enough diversity is present within the population. The performance of each individual, or fitness function, is assessed by estimating the predictive capability of a support vector machine algorithm over unseen data. Each individual encodes the features to be used in the classification task.

[3]GPOMS is the acronym of Google-Profile of Mood States, that measures mood in terms of 6 dimensions (Calm, Alert, Sure, Vital, Kind, and Happy).

3.6 Conclusions

A summarization of the contents that have been presented along this chapter is displayed in Table 3.2. For each work it is shown the application domain, the features and techniques that have been employed, and the usage, or not, of feature selection. At a glance it can be observed that the majority of works utilize syntactic and semantic features, and that feature selection has not been very popular.

Works	**Features**				**Feat. reduction**	**Techniques**			**Domains**		
	F1	F2	F3	F4	Yes/No	T1	T2	T3	D1	D2	D3
Subasic and Huettner [2001]	X	X			No			X			X
Tong [2001]	X	X			No			X	X		
Morinaga et al. [2002]	X				Yes			X	X		
Pang et al. [2002]	X				No	X			X		
Turney [2002]	X	X			No			X	X		
Agrawal et al. [2003]	X		X		No	X	X			X	
Dave et al. [2003]	X				No	X		X	X		
Riloff and Wiebe [2003]		X		X	No	X					X
Yi et al. [2003]	X	X			Yes			X	X		X
Yu and Hatzivassiloglou [2003]	X	X			No	X		X		X	
Efron [2004]	X		X		No	X	X			X	
Fei et al. [2004]		X			No			X	X		
Gamon [2004]	X			X	Yes	X			X		
Grefenstette et al. [2004]	X	X			No			X		X	
Hu and Liu [2004b]	X	X			No			X	X		
Pang and Lee [2004]	X	X			No	X		X	X		
Mullen and Collier [2004]	X	X			No	X			X		
Nigam and Hurst [2004]	X	X			No	X				X	
Wiebe et al. [2004]	X			X	Yes	X		X		X	X
Liu et al. [2005]	X	X			No			X	X		
Mishne [2005]	X	X		X	No	X				X	
Whitelaw et al. [2005]	X	X			No	X			X		
Wilson et al. [2005]	X	X			No	X					X
Pang and Lee [2005]	X				No	X		X	X		

Table 3.2: A previous work summarization

Most sentiment analysis related research is based on the assumption that web comments are written in English. Depending on the considered features, some of the proposed approaches are language dependent. The underlying ideas are of course valid for any language, but it's translation is far from trivial. Take into account approaches based on phrase pattern matching, linguistic constructions, part of speech tags, lexicons, etc.. Some of them require language specific tools in order to work (imagine sentiment lexicons in french or spanish), and other require to be adjusted to the target language by an expert (is is trivial to find equivalences for phrase patterns and constructions between different languages?). On the other hand, there exist other approaches that only need a corpus of text, whatever the language is, to build a model and to predict new comments in that language.

In general, our perception is that much of the existing research is aimed at proposing approaches to solve specific problems in concrete domains. Under this circumstance is complicated to carry out a fair comparison between proposals and to evaluate it's real performance. It is also worth mentioning that there does not exist a well-known and widely accepted set of test problems to measure the effectiveness of the proposed techniques in a general domain.

Chapter 4

Dataset and methodology

This chapter is intended to provide the reader with the general setup configuration we have adopted during the experimentation in this work. We start describing the dataset and its main properties, together with all the preprocess procedures we have applied to the text movie reviews. As follows, we detail the software and hardware environment that has been employed. Finally, we present the methodology we used to execute the experiments and to estimate the performance of the proposed approaches, including evaluation strategies, metrics or loss functions and parameter tuning.

Note that we only describe in this chapter the general configuration that has been employed in all the experiments. Particular characteristics of each technique are commented in chapter 5, which focuses on describing each technique and the obtained results.

4.1 Dataset

In this work it has been used a dataset[1] that represents a real world problem. It is composed of a series of movie reviews extracted from the Internet Movie Database (IMDb) archive of the `rec.arts.movies.reviews` newsgroup. It was firstly introduced in [Pang et al., 2002] and has been broadly used and mentioned in sentiment analysis or opinion mining related literature [Dave et al., 2003; Mullen and Collier, 2004]. Pang et al. [2002] justify the movie review domain selection principally because of the reviewers summarizing its sentiment by means of a numerical rating. Previous results also showed movie reviews to be the most difficult of several domains for sentiment classification [Turney, 2002].

The authors only selected reviews where the rating was expressed either with stars or some numerical value. Ratings were then automatically extracted and converted into one of these categories: *positive*, *negative* or *neutral*. Note that reviews have not been manually labeled (after having been read by a human), thus it is possible for some of them to be assigned an erroneous class. A maximum of twenty reviews per author

[1] It can be downloaded from http://www.cs.cornell.edu/people/pabo/movie-review-data/

was established. The final dataset was obtained by randomly selecting 700 positive-sentiment and 700 negative-sentiment documents. The neutral class was not considered in this work. It is a binary classification problem, and with respect to the class the dataset presents a perfect balance. The example distribution is presented in Table 4.1.

Class	Number of documents	%
Positive	700	50
Negative	700	50
	1400	100

Table 4.1: Example distribution in movie review dataset

4.2 Document preprocessing and representation

Our "raw" data is a corpus of natural language text. In order to obtain a suitable representation of the raw text in a computer, we need to apply some linguistic processing to the data [Turney and Pantel, 2010]. First we need to *tokenize* the raw text; that is to extract the *vocabulary* and to decide which *words* or terms are comprising it. Tokenization of English seems very simple at first sight because words are separated by spaces. However we also have to be aware of punctuation symbols like full stops, question or exclamation marks, hyphens and so on (e.g., don't, and/or, state-of-the-art). Note that while separating by spaces is valid for English, is not appropriate for other languages as Chinese. In our case we have extracted the vocabulary from the movie review data by separating words by spaces.

However, by manually examining the obtained vocabulary after the space separation stage, we have found a number of special cases in which the performed separation is not successful. When a punctuation symbol appears right after a word, they are both extracted together, e.g., "*water.*" vs "*water*". Two different terms that represent the same word are created. The same situation appears when text authors emphasize an emotion by repeating several times a single character within a word, e.g., "*gooood*". *Case sensitivity* is also a typical problem in this context. We have implemented a parser in order to overcome these difficulties that is detailed in section 4.2.1.

A second step consists in *normalizing* the extracted words to convert superficially different strings of characters to the same form. The motivation is that we are really interested in the meaning of the words. The most common cases of normalization are converting all words to lower case and stemming. A final stage consists in *annotating* the raw text to mark identical words as being different (e.g., fly as a noun or fly as a verb). The most used annotation method is part-of-speech tagging. As reported by [Kraaij and Pohlmann, 1996], given its nature it is expected for normalization to

increase *recall*[2] and to decrease *precision*[3]. By removing superficial variations, that we consider irrelevant to meaning, we make it easier to recognize similarities, so recall increases. On the other hand, it is expected for annotation to increase precision and to decrease recall, because we have a higher confidence on the detected similarities.

4.2.1 Document parser

We have used `MATLAB` in order to implement a parser that uses regular expressions to perform a series of adjustments to the original text movie reviews. A set of input comments is passed through a number of filters, with each filter carrying out a concrete matching. These matchings are sequentially performed in a pre established order. Each time a matching is successful, the correspondent transformation is performed. Among these filters we highlight the following:

- Filter some punctuation symbols that were extracted together with several words, as there was not a blank space between them (e.g., "t*oday...*", "*(left*", "*good!*").
- Convert all words to lower case.
- Substitute different *entities* by a constant string or *collocation* (tag). For example, we have replaced all *url links* by the string URL, and all the numbers by the string NUMBER.
- Having in mind that often people uses smileys on the Internet to express a sentiment, we have replaced a series of smileys by their corresponding tag: *positive_smiley* (*:),:D*), *negative_smiley* (*:(*, *:/*) or *neutral_smiley* (*:o*).

Note that we have decided not to carry out any annotation task.

4.2.2 Document representation

Once the input documents have been parsed we need to obtain a valid representation as an input for the classification algorithms. We have decided to use a *bag of words* representation, in which each review is represented by a vector with equal size to the number of words in the vocabulary. It does not consider any *order* among words and assumes *independence* among them. Thus, the final data is represented by a *matrix* with as many rows as documents and as many columns as terms in the vocabulary.

Under the hypothesis that people tend to use certain words when expressing their opinions and sentiments, we are going to use word *presence/absence* in each document in order to assign the matrix weights. It is a highly *sparse* representation because most of the terms only appear in a few documents. It is a boolean assignment such that the element a_{jk} in the matrix is assigned a one value if the term k appears in the document j and 0 otherwise.

[2]Recall is the fraction of relevant instances that are retrieved

[3]Precision is the fraction of retrieved instances that are relevant.

On the other hand, a *frequency* representation can be obtained by measuring each term's number of appearances per document. As follows we formally denote both representations. Let $\{t_1, ..., t_k\}$ be a predefined set of k terms comprising the vocabulary. Let $a_k(d)$ be the number of appearances of term t_k in document d. A document's frequency representation is obtained as:

$$d_{freq} = (a_1(d), a_2(d), ..., a_k(d)) \tag{4.2.2.1}$$

Let $a_k^{'}$ be 1 when the term t_k is present on document d and 0 otherwise. A document's presence/abscence representation is obtained as:

$$d_{pres} = (a_1^{'}(d), a_2^{'}(d), ..., a_k^{'}(d)) \tag{4.2.2.2}$$

Recall that in [Pang et al., 2002] it has been already reported that a presence/absence representation yields to better accuracy results than a term frequency representation in sentiment analysis related tasks.

At this point we have utilized a series of traditional IR text analysis techniques in order to determine the *vocabulary*. As a result we have obtained four different representations for the movie review dataset:

- **Unigrams**: words are separated by spaces after the parser stage.
- **Unigrams + NOT**: a heuristic that aims to model the effect of negation is introduced. Whenever a negation clause is detected, all the following words are negated until a punctuation symbol is found. The negation is performed by appending a NOT_ tag to the word. The size of the vocabulary is expected to increase. A list of the considered negation clauses and punctuation symbols is shown in Table 4.2.
- **Unigrams - empty words**: in IR is widely accepted that too much frequent words do not provide information, and that words without semantic meaning (articles, conjuctions, etc.) are not useful. Salton and McGill [1983] report that words appearing in at least 80% of documents do not provide useful information. A list[4] of words without semantic meaning (stop words) is filtered out. The size of the vocabulary is expected to decrease.
- **Unigrams + stemming**: words with different morphology share the same root and are semantically equivalent. A unique word is selected to represent the rest with the same concept. The size of the vocabulary is expected to decrease. We have performed the stemming step by means of the `tm` package in `R`. It uses an implementation of the *Porter* algorithm.

[4]The list has been obtained from http://members.unine.ch/jacques.savoy/clef/englishST.txt

Negation clauses							Punctuation symbols						
not	no	never	nothing	cannot	except	*n't	,	.	;	(	)	:

Table 4.2: Set of negation clauses and punctuation symbols used to determine the scope of them. The asterisk represents any string sequence such that ends with n't.

4.3 Implementation

All the experiments developed in this project has been performed with the support of four core components. From top to bottom, the higher software layer is `The Spider` [Weston et al., 2005], an open-source machine learning toolbox for `MATLAB`. Below, obviously, lays the `MATLAB` software environment itself. The last software layer is offered by `libsvm` library for SVM [Chang and Lin, 2011].

Regarding the hardware configuration, we have executed all the experiments over *Pomar*, an Apple Xgrid cluster from the Artificial Intelligence Center (AIC) of the University of Oviedo.

The Spider machine learning toolbox for MATLAB

This toolbox aims to offer a complete object-oriented environment for machine learning in `MATLAB`. It provides out-of-the-box base learning algorithm and offers plugin capabilities to integrate more algorithms with very little effort.

Apart from these benefits it also includes other machine learning functionalities, like deterministic cross-validation partitioning of data, model selection through grid-search, a complete collection of loss functions for different aims, statistical tests, visual plots and much more.

MATLAB

`MATLAB` (**MAT**rix **LAB**oratory) is a mathematical software that provide users with an Integrated Development Environment (IDE) and its own related programming language (*m* files). Among its main functionalities we can highlight matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces (GUI), and interfacing with programs written in other languages. It is a frequently used software in both universities and research centers. In this work we have used `MATLAB` in its `R2012a` version.

Libsvm: a library for support vector machines

It offers support vector classification (C-SVC, ν-SVC), regression (ε-SVR, ν-SVR) and distribution estimation (one-class SVM) implementations. Some remarkable

features are the support of efficient multi-class classification capabilities, easy-to-use `MATLAB` interface (among other interfaces for alternative machine learning environments and languages), cost-sensitive binary classification for unbalanced data, probability estimates, etc. The version used in this project is `libsvm-mat-2.91-1`.

We have also utilized a special version of `libsvm` called `liblinear`, that is specially well-suited for text classification problems.

Pomar cluster from the Artificial Intelligence Center (AIC)

The hardware layer is an Apple Xgrid cluster, which is composed of 15 nodes or agents. The main node is usually excluded from the experimentations, acting as the main controller of the cluster. The rest 14 agents provide a total CPU power of 396 GHz, divided in 160 parallel processors. In our experiments, each independent task is executed in parallel over one of these processors, as an independent process.

4.4 Performance estimation and metrics

We are interested in the *generalization* ability of the classification algorithms. That is its capacity to successfully predict the class of instances that have not been seen during the training stage. Thereby, it is not appropriate to use the proper training instances to measure the performance of the models. The easier solution consists in splitting the available data into *training* and *test* instances, and then use the former for generating the model, and the latter for measuring its performance. However, this strategy is likely to present a high *variability*, with a strong dependence on the concrete instances that "fall" into the test set.

A well-known solution is the so-called *cross-validation* estimation procedure. It involves partitioning a sample of data into k disjoint subsets (k-fold cross-validation), and measuring the performance of the models in each subset, after having generated each model from the other $k-1$ subsets. The final estimation is performed by averaging the individual model performances, and thus reducing the expected variability.

One of the approaches we are presenting in the following chapter requires to carry out a large number of computations for obtaining the associated representation (cooccurrence word information). Furthermore, this representation directly depends on the concrete instances belonging to the training set. One step consists in calculating the distances among a series of vectors that represent each one of the terms comprising the training vocabulary. Due to the high temporal complexity of this procedure, we realized that it was not possible to apply it as many times as folds in the k-fold cross-validation setup. As a consequence, we decided to perform a simple train/test data split instead of using cross-validation. As we want to conduct a comparison among all the proposed approaches, we have used the same train/test split in all the experiments. 80% of data has been used for training and 20% left for testing. It is also important to remark that the train/test split has been *balanced* in order to keep the proportion of positive versus negative examples.

4.4.1 Parameter tuning

Most of the learning algorithms depend on a series of parameters that are used to adjust certain of its characteristics according to the problem to solve. There are methods that aim to find the most promising values for this parameters. An estimation of the best values, striving for performance over unseen examples, is produced. In this work we are using SVMs with linear kernel, thus the only parameter that require to be tuned is c. We have utilized the so-called *gridsearch* strategy, which uses the training data to compare the performance of the classifier over unseen examples with a number of different values for the parameter c.

Using test data is not appropriate when performing the parameter tuning task. For each target value a cross validation over training data is executed, and as a result, an estimation of the performance over unseen data is obtained. As recommended by Dietterich [1998] we have used 2-fold cross-validation with 5 repetitions. Due to the fact that the only changing condition in the executions is the value of the parameter, differences in the results are explained by the target parameters.

Previously, we have justified ourselves for not using a cross-validation strategy; using it inside the parameter tuning procedure may seem rather incoherent. However, it is important to note that we have applied the high computational cost representation acquisition process to the entire training data. From the obtained representation we have performed the parameter tuning task. The consequence is that training data inside gridsearch runs have been created from both training and test data. Thus, our 2x5 cross-validation estimations are expected to be optimist. Since we only want to compare among different values of a parameter, it is not a problem to get biased estimations.

4.4.2 Performance metrics

There exist a series of functions that are used to assess the performance of the obtained predictions (*loss functions*). Formally, a loss function l is defined as $l : \mathcal{Y}$ x $\mathcal{Y} \rightarrow \mathbb{R}^{\geq 0}$. The performance of a prediction $h(\boldsymbol{x}_i)$ when the true value is y_i, is calculated as $l(\mathrm{y},\hat{\mathrm{y}})$. Note that, in general, any loss function can be applied to any problem, but depending on the concrete characteristics of the problem some of them are considered to be more suitable.

In this work we are dealing with a binary classification problem in which class distribution is perfectly balanced. The dataset is comprised by 700 positive instances and 700 negative ones. In this case, a simple and well-suited loss function to measure the performance of the models is the traditional *accuracy*. As follows we are going to introduce it formally.

Contingency table

Although the contingency table is not a measure itself, it provides the essential values to compute many loss functions in binary classification tests. These values are the *true*

positives (TP), *true negatives* (TN), *false positives* (FP) and *false negatives* (FN).

Table 4.3: Contingency table for binary classification problems

	$y_i = +1$	$y_i = -1$
$h(\boldsymbol{x}_i) = +1$	TP	FP
$h(\boldsymbol{x}_i) = -1$	FN	TN

Typically, by columns we have the expected values of the class (y_i), i.e., positive and negative. Hence, the predicted values, (i.e., $h(\boldsymbol{x}_i)$) are often placed by rows.

Accuracy (ACC)

The classical loss function for binary classification is the *Zero-One Loss* (ZOL), also known as *Class Loss*. This error measure is computed as follows:

$$\Delta_{0/1}(h(\boldsymbol{x}_i), y_i) = [h(\boldsymbol{x}_i) \neq y_i] \tag{4.4.2.1}$$

Applied over the whole set, the zero-one loss measures the proportion of examples that are misclassified. In terms of the contingency table defined in Table 4.3, we compute:

$$ZOL = \sum_{i=1}^{n} \Delta_{0/1}(h(\boldsymbol{x}_i), y_i) = \frac{FP + FN}{TP + TN + FP + FN} \tag{4.4.2.2}$$

This function is the inverse of the classification *Accuracy* (ACC):

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = 1 - ZOL \tag{4.4.2.3}$$

Chapter 5

Experimentation and results

This chapter is intended to explain our approaches to solve the problem introduced in section 4.1, to describe in detail the characteristics of all the experimental setups, and to present the obtained results with respect to the performance measures that were introduced in section 4.4.2. An important objective is to provide the reader with all the required details to repeat the experiments in case it is required. For each one of the presented experiments we are going to describe its:

- Representation and features, that is the approach we have used in order to represent the text commentaries and to extract the features to be used by the learning algorithm.
- Specific experiment conditions, such as the concrete values for the algorithm parameters.
- Obtained results in terms of accuracy, as defined in 4.4.2, and number of utilized features.

Recall that our principal objective is to use machine learning methods in order to generate models presenting an adequate tradeoff between efficiency and complexity (principally with respect to the inherent complexity of natural language processing), that are applicable in a general domain. We are interested in getting rid of language dependent artifacts, such as phrase patterns, (sentiment) lexicons, and syntactic/semantic constructions. Usually, these kind of techniques are more complex and domain dependent. Furthermore, the application of most of these language dependent techniques requires the usage of external tools, like part of speech taggers, etc. Analyzing and using this sort of NLP techniques remained out of our scope, since machine learning is our core work interest.

In this work we prefer to explore diverse methods in order to represent the textual information. We also focus in trying to extract the vocabulary that is most likely to help us in the classification task. We start by reproducing an extensive part of the experiments that were presented in [Pang et al., 2002], the first work that explicitly measured the effectiveness of applying machine learning methods to the

sentiment analysis problem (see section 5.1). As follows we propose a series of feature reduction approaches, with the objective of selecting an adequate subset of words in the vocabulary. We also discuss the application of an ensemble, that is to aggregate a series of different models, and which are the model characteristics an ensemble would get benefited from. All of these steps are performed with the goal of obtaining a reduced and representative vocabulary of the movie review domain in a sentiment analysis task. Finally, we have explored the possibility of representing each word in the target vocabulary with a high dimensional vector. Words that are close in that high dimensional space, are expected to have a similar semantic meaning. Then we have applied a simple clustering algorithm and obtained a series of *meaning* clusters, i.e., ideally, words that are semantically similar fall in the same cluster. From the cluster information we have created a document "*meaning*" representation as input for the classification algorithms.

5.1 Baseline

As already commented, Pang et al. [2002] were the first authors to explicitly measure the effectiveness of applying traditional machine learning methods to the sentiment analysis problem. Machine learning methods have been already reported to successfully perform on traditional information retrieval problems, such as text categorization. Pang et al. [2002] showed that, although sentiment can be expressed in a more subtle manner than traditional keywords, machine learning techniques are also useful when modeling sentiments and opinions.

As baseline they asked two computer science undergraduates to independently choose a list of good indicator words for positive and negative sentiments in movie reviews. Even though both lists seemed intuitively reasonable, only an accuracy of 58% and 64% was respectively obtained over 4.1 dataset[1]. Based on a very preliminary examination of frequency counts in the entire corpus, the authors then selected a list of seven positive and seven negative words. By using this list the accuracy increased to 69%. They concluded that it is worthwhile to explore corpus-based techniques, rather than relying too much on prior intuitions, in order to select good features and to perform sentiment classification in general.

In our baseline we are going to reproduce a series of experiments that were presented in [Pang et al., 2002]. A comparison between two representations (see section 4.2.2), word *frequency* appearance and word *presence/absence*, both at the document level was carried out. They reported that a presence/absence representation yields to better accuracy results than a term frequency representation in sentiment analysis related tasks (82.9% vs 72.8% accuracy using SVM as classifier).

In order to confirm that result we have used a unigram representation (see Section 4.2.2) with both frequency and presence/absence information. At this point we have not performed any filtering, and all the terms in the obtained vocabulary have been taken into account (see Section 4.2 to find out how the vocabulary is extracted). Moreover,

[1] A minimum accuracy of 50% is expected due to the classes being perfectly balanced.

to check the effect of applying traditional IR text analysis techniques in a sentiment analysis classification problem, we have also evaluated the other three representations described in Section 4.2.2. These include Unigrams plus NOT tag, Unigrams plus stop word removal and Unigrams plus stemming. Note these are strategies to extract the vocabulary. As regards the matrix information, for these three representations we have only used presence/absence information, since as it can be seen in Table 5.3, frequency version effectively yields to worse results (we are not considering it anymore in the upcoming experiments).

Another preliminary evaluation we have carried out consists in filtering all terms that do not appear in a minimum number of documents, in this case 5. We have created an object using `spider` that calculates the number of documents each term appears in within the training data. All the words not fulfilling the condition are filtered out both at training and test data.

A Table presenting the characteristics of all the resulting representations is presented in 5.1. As expected, adding the NOT tag increases the number of terms in the vocabulary, while stemming leads to an important decrease in the number of considered words. The empty words method just filters out the words appearing in the stop words list.

Representation	# Features	# Features (after filtering)	# Classes
Unigrams	37156	9460	2
Unigrams + NOT	53628	12036	2
Unigrams - Empty	36617	8954	2
Unigrams + Stemming	24337	7183	2

Table 5.1: Characterization of the four representations before and after the filtering of the words that do not appear in at least 5 documents

We have already discussed why SVM is a well-suited technique for dealing with text data classification problems in Section 3.3. In these preliminary evaluations we have measured the performance of two different SVM implementations: `libsvm` and `liblinear`. The latter is optimized for dealing with high dimensional and linearly separable problems (like the one we are facing), thus we expect it to clearly outperform the former. We have performed parameter tuning on both algorithms by means of `mygridsearch` object in `spider`. The parameter search configuration is shown in Table 5.2.

Folds	Repeats	Learning algorithm	Parameter	Values
2	5	Linear `libsvm` `liblinear`	C	{0.001, 0.01, 0.1, 1, 10, 100}

Table 5.2: Parameter search configuration

The results we have obtained for each representation with both SVM implementations are presented in Table 5.3. The best results are highlighted in bold. As Pang et al.

Representation	All features		5 documents filtering	
	`libsvm`	`liblinear`	`libsvm`	`liblinear`
Unigrams - presence	83.571	**87.5**	83.214	**87.857**
Unigrams - frequency	80.357	81.786	78.929	79.286
Unigrams + NOT	80	82.143	78.571	80
Unigrams - empty	82.5	83.214	82.5	83.571
Unigrams + stemming	82.5	84.643	82.143	83.929

Table 5.3: Accuracy (%) results obtained in baseline experiments

[2002] already reported, word presence/absence information yields to better results than frequency counts, independently of the used SVM implementation and number of features. Frequency model may tend to give a higher weight to those terms being very frequent, but not necessarily useful for the sentiment classification task.

It seems that NOT tag representation does not help at all, and even makes accuracy decrease despite the higher number of features. We think that such a complex effect as *negation* can not be modeled with such a simple heuristic. Stop word removal does not seem to yield to better results in this particular sentiment analysis problem. Our intuition is that we can not consider the concept of "empty" word to be the same in text categorization and sentiment analysis tasks. While traditional stop word removal lists (like the one we have used) are adequate in IR tasks, they look too much extensive for sentiment related problems. In fact, we can observe that accuracy decreases when filtering these words. In the following experiments we have tried to find a proper subset of real "empty" words for this concrete problem. Stemming also makes the models to perform worse than simply using unigrams themselves. IR traditional text analysis techniques do not give the impression of being as useful as in IR problems when applied to sentiment analysis tasks. The importance of *context* in sentiment analysis could maybe be an explanatory reason.

Regarding SVM implementations, `liblinear` outperforms `libsvm` in all the performed experiments. After having analyzed these results, we have decided in the following experiments to only use unigrams as the strategy to obtain the vocabulary, word presence/absence as feature information and `liblinear` as classification algorithm.

5.2 Looking for an adequate vocabulary: feature selection

In the previous section we have already reduced the cardinality of our target vocabulary by filtering terms that do not appear in at least 5 documents. In Table 5.3 it can be seen that model's performance does not get specially harmed when performing that filtering, even it slightly gets increased in the case of unigrams and stop word removal. However, the subset of words appearing in at least 5 documents is not necessarily an

adequate feature subset in order to learn to predict the sentiment that is expressed in the documents. On one hand, some frequent words may not be relevant within the sentiment class, because of them not having any semantic meaning or not being domain-relevant. On the other hand, not all the low frequent words are necessarily useless, some of them may be helpful in the classification task.

Our objective in this experiment section consists in trying to find the most adequate subset of words to comprise the vocabulary. Note that is a domain dependent task; in the case of this homework we aim to find a representative vocabulary of a movie review domain. However, this approach can be applied in a generic domain by simply wrapping our vocabulary dependent approach into a generic model that is able to discern among different domains. That is a high level classifier able to identify the concrete domain, in order to later select the concrete domain dependent model. In this work we are only focusing on the latter, within a movie review domain.

5.2.1 Information theory measures

At this point we have used a series of feature selection filters (see 2.3.1) coming from the Information theory. It is a better procedure to reduce the dimensionality since we are taking into account the class information. Concretely, we have employed four different information measures in order to build a word ranking. Ideally, the top ranked words are more important when predicting the class. We are interested in obtaining as many models, as diverse as possible. The principal advantage is that we can later perform set operations, like set intersection or set union, from the obtained vocabularies. We are going to briefly describe each measure as follows.

Information gain

Intuitively, information gain is a measure that indicates how important a feature is with respect to a class. It measures the closeness to class c knowing the presence or absence of the term t. In an information retrieval context, information gain of a term t with respect to a class c is calculated as:

$$IG(t,c) = P(t) \cdot P(c/t) \cdot \log \frac{P(c/t)}{P(c)} + P(\bar{t}) \cdot P(c/\bar{t}) \cdot \log \frac{P(c/\bar{t})}{P(c)}$$

Its domain is on the interval $[0, 1]$, with 1 meaning that t is perfect for predicting c and 0 useless.

FCBF

The FCBF (Fast Correlation-Based Filter) algorithm consists of two stages: the first one is a relevance analysis, aimed at ordering the features depending on a relevance score, which is computed as the *symmetric uncertainty* with respect to the class. This stage is also used to discard irrelevant variables, which are those whose ranking score

is below a predefined threshold. The second stage is a redundancy analysis, aimed at selecting predominant features from the relevant set obtained in the first stage. This selection is an iterative process that removes those variables which form an approximate Markov blanket.

Linear or rank correlation

Correlation coefficients measure the strength of association between two variables. In this work it has been used the *Pearson's* linear correlation coefficient, which measures the strength of the linear association between two variables. The sign and the absolute value of a correlation coefficient describe the direction and the magnitude of the relationship between the two variables. Its ranges between -1 and 1. We have used its absolute value, since we are only interested in its magnitude and not in the direction.

Cosine similarity

Cosine similarity is a measure of similarity between two vectors by measuring the cosine of the angle between them. We have calculated the cosine similarity between each term presence/absence vector and the class labels vector. It is often used to compare documents in text mining.

Results

Starting from the original 37156 word vocabulary (see Table 5.1), we have executed each filter so that in each iteration 1000 additional terms are filtered. For example, in the first iteration we rank the 37156 terms in function of the corresponding measure, and then we filter the last 1000 words in the ranking. As a result, we have obtained 37 different set vocabularies for each one of the four filters. Note that we only use the training data when determining the vocabulary. Once we know the words comprising the vocabulary, we just select their corresponding features and carry out the classification task with the original Unigrams presence/absence information. See Table 5.2 as regards `liblinear` parameter tuning configuration.

bad	awful	worst	wasted	mess
also	performance	stupid	wonderfully	dull
laughable	lame	waste	both	unfunny
solid	uninteresting	effective	world	ridiculous
outstanding	boring	true	most	subtle

Table 5.4: List of the 25 top ranked words by the Information Gain filter

A list with the 25 top ranked terms selected by the information gain filter is presented in Table 5.4. Most of the words appearing in the table are *adjectives*. It is in accordance

with some previous intuitions about people tending to use adjectives when expressing their emotions and sentiments [Pang et al., 2002].

# Features	Information gain	FCBF	Linear correlation	Cosine similarity
36156	**87.5**	**87.5**	**87.5**	87.5
35156	86.79	86.79	87.14	**87.86**
34156	87.14	87.14	87.14	87.86
33156	87.14	87.5	87.14	87.86
32156	87.14	86.43	86.43	87.86
31156	86.43	85.71	86.43	87.86
30156	86.43	86.43	86.07	87.14
29156	86.43	86.43	86.43	87.5
28156	86.07	86.79	86.07	87.5
27156	85.71	84.64	85.71	87.14
26156	85.36	86.43	85.71	87.5
25156	85.36	85.71	85.71	87.14
24156	85.71	85.71	85.71	87.5
23156	86.07	85.71	85.36	87.5
22156	86.07	85.71	85.36	87.14
21156	86.07	85.71	85.36	86.79
20156	86.07	85.36	85.36	86.79
19156	86.07	85.36	85	86.79
18156	86.07	85.36	84.64	86.79
17156	86.07	85.36	84.64	86.79
16156	86.43	85.36	84.64	86.79
15156	86.43	85.36	85	85.36
14156	86.43	85.36	85	86.43
13156	86.79	85	85	86.79
12156	86.43	85	85	86.43
11156	86.79	85	84.64	85.71
10156	86.79	85.36	84.64	86.43
9156	86.79	85.36	84.64	86.07
8156	86.79	85.36	85	86.07
7156	85.71	85	85.71	85.36
6156	83.93	83.21	85.36	85
5156	83.57	83.57	85	84.64
4156	83.57	82.86	85	85.36
3156	83.93	82.5	84.64	83.93
2156	82.14	78.93	85	82.14
1156	79.64	78.57	82.86	80.71
156	76.07	73.57	78.21	63.93

Table 5.5: Accuracy (%) results obtained for each filter and number of words comprising the vocabulary

The results we have obtained with each filter in each iteration are presented in Table 5.5. A graphical representation is depicted in Figure 5.1 in order to allow an easier visualization of the results. The principal observation is that the accuracy of the models tends to slightly decrease as the number of utilized features also decreases. However, in general, models using a significant less number of attributes manage to obtain a nearly equal performance as the original model. For example, cosine similarity model is able to successfully predict an 87.5% of test examples with just 23156 features; the original model using 37156 attributes performed the same. We can conclude that feature selection is useful, at a more or less extend depending on the filter, since it provide us with a subset of relevant features for the sentiment classification task, without harming the performance of the predictions. We have gained an insight into a useful word subset for performing sentiment analysis in a movie review domain.

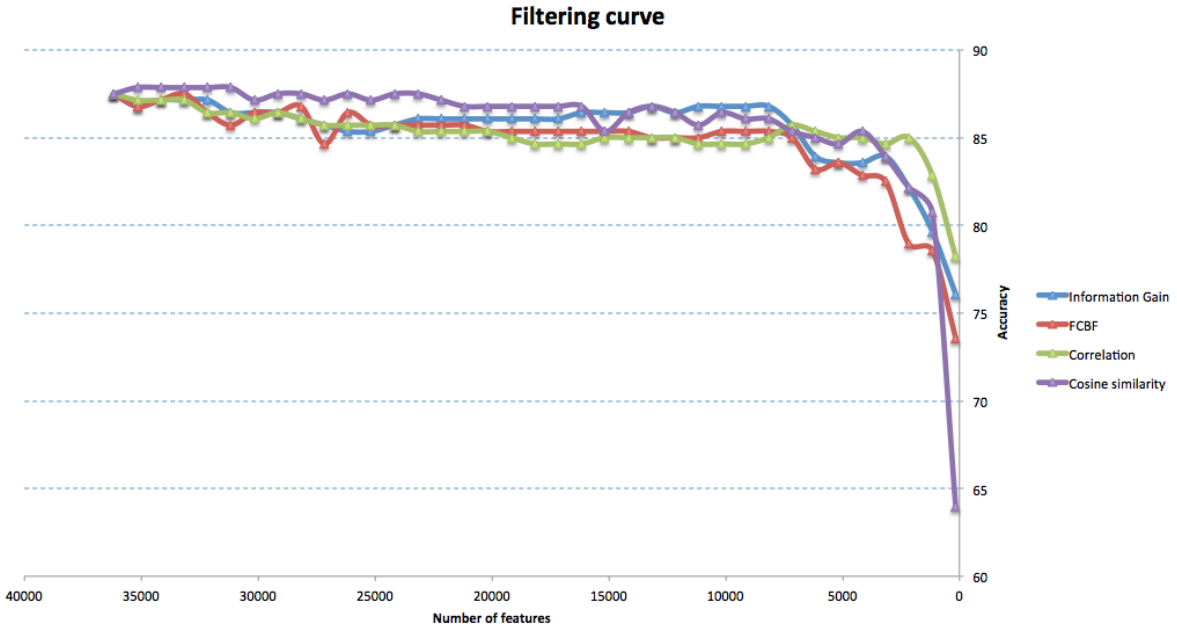


Figure 5.1: Graphical representation of the obtained results for each filter and number of words comprising the vocabulary

5.2.2 Frequency based filtering

In contrast with the previous experiments, in this case we are going to select the most adequate vocabulary by counting word occurrences. We have considered *document frequency*, that is the number of documents in the corpus where the word occurs, independently of the number of times (inside the document). We have taken into consideration *Zipf's law*, which states that words appearing in almost all documents are not informative, and that rare words do not provide much information. Its conclusion is that *intermediate* frequency words are the most informative words. We have accomplished this word reduction task in two separated phases.

Phase 1: Filtering high frequency words

In this experiment we have started by filtering the most frequent words in order to detect the point in which the filtered words start to be useful for the classification task, i.e., the point in which accuracy starts to decrease. Note that this is the same as looking for the concrete set of "empty" words in this particular domain. In fact, we expect to obtain a smaller set than the original "empty" word set that we used in baseline experiments.

Our first try consisted in filtering up to the 5000 most frequent words in groups of 100. A graphical representation of the obtained results is presented in Figure 5.2. It can be observed that the accuracy immediately starts to fall after removing a little number of frequent words. This confirms our initial intuition about the number of real "empty" words in a sentiment analysis related problem being small. Our next movement was intended to zoom into the frequency curve.

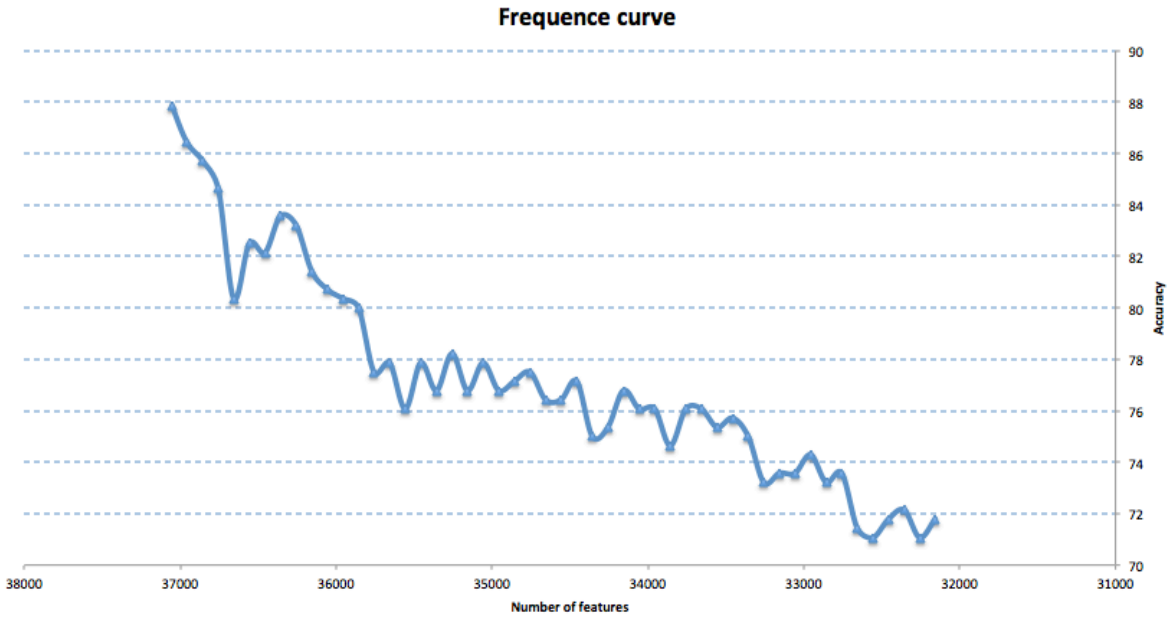


Figure 5.2: Filtering the most frequent terms in the original vocabulary

In order to do so we have carried out a precise filtering of the most frequent words. We started by filtering one by one the most frequent words until we reached 150 terms. The obtained results are depicted in Figure 5.3. In this case the obtained result graphic is not as smooth as in the previous case due to the zoom and the situation of filtering words one by one. From the graphic we have selected three feature points that we considered good, taking into account its neighbour points. These three are highlighted with a red circle in 5.3. Thus, as an output of this first phase we have obtained three different models with an associated vocabulary and accuracy. Table 5.6 shows the selected models together with their assessed performance.

Model	# Features	Accuracy
Point - 37095	37095	88.929
Point - 37058	37058	88.214
Point - 37016	37016	87.143

Table 5.6: Points selected in the precise high frequency experiment

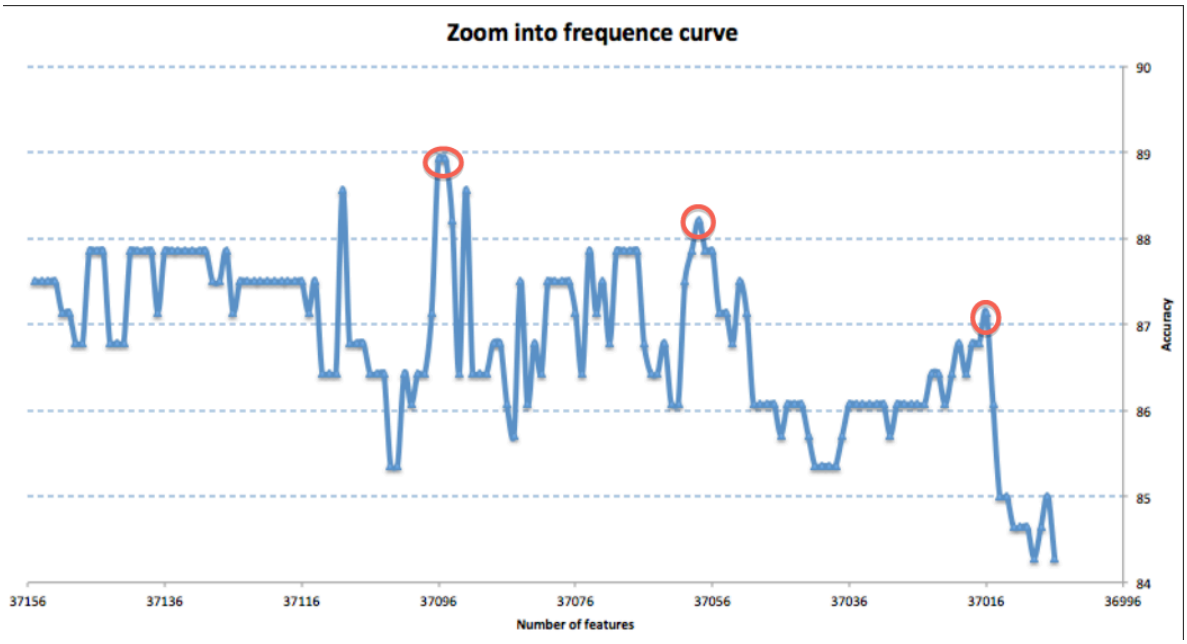


Figure 5.3: A precise filtering of the 150 most frequent terms in the original vocabulary

Phase 2: Filtering low frequency words

From the three selected models, which are already filtered, we are going to repeat the procedure we applied in the information theory measures related experiments. We start filtering the words having a lower frequency count in groups of 1000 until the cardinality of the remaining word set is less than 1000. As a result, for each one of the three selected points we obtain a graphic (Figure 5.4) that is very similar to the graphic (Figure 5.1) we presented in the information theory measures.

If we compare both graphics, we observe that specially Point - 37095 and Point - 37058 perform better than any of the information theory measures. With a reduced number of features (> 1000) an approximately accuracy of 88% is obtained with a simple frequency filtering. If we take into account that correlation and cosine similarity tended to perform better than information gain and FCBF, it is surprising how it looks like the simplest the criterion used to filter, the better the obtained results. It is worth mentioning the good results obtained with Point - 37095 model, which is able to successfully predict 86.7% of test examples with just 2095 features.

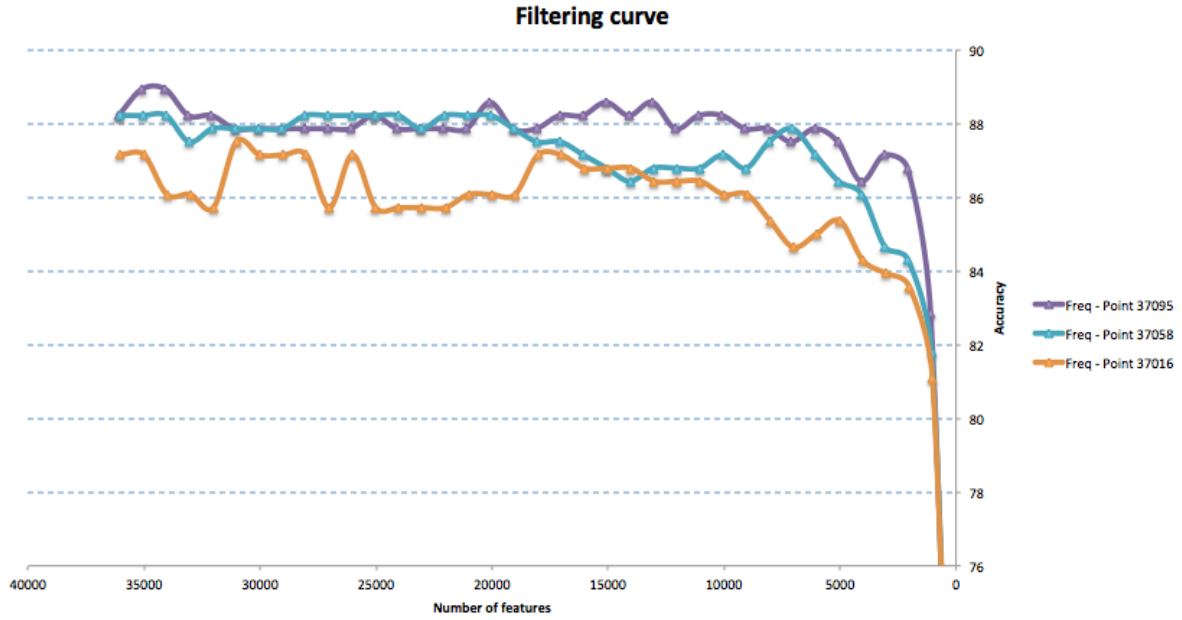


Figure 5.4: Filtering low frequency words after having selected three points in phase 1

5.3 Ensemble: aggregating models

Our next logical move consists in aggregating our 5 generated models by using a voting ensemble. It is expected to perform better than any individual model by itself because the various errors of the models "average out". Ensembles combine multiple hypotheses to form a (hopefully) better hypothesis. For this situation to take place is very important to have enough diversity among the models that are solving the same problem. It is very important for them to make different generalization errors over the test data.

In our case we have 4 models resulting from the experiments in Section 5.2.1, and three ones as a result of frequency based filtering. Moreover, for each kind of model, we have a series of points, each one associated to a concrete number of features. We have decided to build two different ensembles, aggregating models that share a similar number of features. As we are looking for diversity in the models comprising the ensemble, we decided to choose models with a small number of features, since we expect the most aggressive the filtering, the most diverse the resulting vocabulary. We also want to evaluate the predictive performance of an ensemble that is built by aggregating a series of small (with small referring to the number of features) models. As for the frequency filtered models, we have decided to only use the best of them to build the ensemble, since all of them share the same vocabulary at a less or more extent (low diversity).

The models that have been used to build both ensembles, together with its associated number of features and assessed performance, are presented in Table 5.7.

On one hand we have used a voting rule to decide the winning class for each instance. Each model "votes" its prediction and the class having the most votes is the ensemble

	Ensemble number			
	1		2	
Model	# Features	Accuracy	# Features	Accuracy
Information gain	8156	86.786	3156	83.929
FCBF	7156	85	3156	82.5
Correlation	7156	85.714	2156	85
Cosine similarity	8156	86.071	4156	85.357
Freq - Point 37095	6095	87.857	2095	86.786

Table 5.7: Information about the models comprising both ensembles

prediction. As the number of models is odd, there is no need for a tie breaker criterion. On the other hand, we have aggregated all the vocabulary within the five models. In mathematical terms, we have performed a union set operation over the five word sets. Afterwards, we have used Unigrams presence/absence information to carry out the classification task. The results we have obtained in both strategies are shown in Table 5.8.

The principal conclusion we can extract from the results is that none of the evaluated ensemble strategies manages to outperform the best of the single models, Frequency - Point 37095. In fact, the ensemble 2 performs worse when voting, and the ensemble 1 when aggregating the vocabulary. We think that a possible explanation for this situation is that worst models tend to make the same prediction mistakes, so a potential good classification made by the best model is lost due to the other models being a majority. As all the models, independently of their individual quality, have the same importance in the voting process, when there exists a low diversity among the worst models, they tend to win the voting when they agree (not on purpose of course) at misclassifying an example. The key reason making our ensemble approach not performing good is that the existing diversity among models is low.

	Ensemble number			
	1		2	
Strategy	# Features	Accuracy	# Features	Accuracy
Voting	-	87.857	-	85
Aggregation	13709	87.5	6116	86.786

Table 5.8: Obtained results in the ensemble experimentation

We have manually examined both ensemble predictions for test examples, and have realized that in most cases a misclassification is obtained, is because all the five models are wrong when predicting the example class. That is to say it is more a problem of our bag of words and presence/absence information representation than of the selected

vocabulary itself. It is possible that our representation does not suffice to successfully model a kind of special reviews that express sentiment in a more subtle manner. Other reasonable, and not mutually exclusive reason, is that some of the reviews may have an incorrectly assigned label. We have manually investigated a number of documents that all the models comprising the ensemble failed to predict. The following is an extract of one of those comments:

> *"after the press screening of moulin rouge i stood in the lobby of the theater listening to the reactions of my friends and colleagues everyone seemed a bit numb understandable after sitting through a barrage of often-incongruous sounds and images a pal of mine simply said that he loved the film and could hardly wait to take his wife to see it another enthusiast immediately began to analyze the production while a woman who flat-out hated the movie gave him the skunk eye when one fellow quietly stated i've never really been a fan of musicals the statement surprised me because even though the story is told almost completely through song i didn't think of it as a musical there is so much going on in moulin rouge that musical seems too small a term to cover it moulin rouge is the kind of creation that sends critics scurrying off to the big tub o' adjectives in search of proper words to describe the experience australian director baz luhrmann the man behind strictly ballroom and romeo juliet fills the heads of viewers with unique camerawork opulent imagery and songs ranging from the sound of music to smells like teen spirit sumptuous and beautiful vulgar and overdone moulin rogue travels through the looking glass while an ethereal stereo loaded with _NUMBER years worth of catchy tunes operates on the random setting."*

In this simple example the reviewer is not even giving its opinion, but describing the various sentiments his/her friends and colleagues had after seeing the movie. Note that we can find opposing sentiments within a single review that are not even held by the same person. This review has been labeled as positive, but we do not think it can be considered as so, since it mixes both positive and negative sentiments.

5.4 Word cooccurrence: meaning classification

One of our objectives for this work consisted in designing a novel approach in order to solve the sentiment analysis problem. Our idea consists in representing each document by using the cooccurrence information of the words comprising it. We have been inspired by Vector Space Models (VSMs), a typically used method in information retrieval tasks (specially in search engines). Its idea is to represent each document in a collection as a point in a space (a vector in a vector space). Points that are together in this space are semantically similar and points that are far apart are semantically distant [Turney and Pantel, 2010].

In our case we are measuring word cooccurrence in order to build a lexical semantic space. Lexical cooccurrence has been established as a useful basis for the construction

of semantic spaces [Lund and Burgess, 1996]. A semantic space is a space, often with a large number of dimensions, in which words or concepts are represented by points; with the position of each point being related to the meaning of the word. Semantic spaces are then useful for examining the relationships among words within them, because these relationships can be quantified by applying distance metrics to the points within the space.

The so-called *distributional hypothesis* is an important basis for our work: words that coccur in similar contexts tend to have similar meanings. If words have similar row vectors in a word-context matrix then they tend to have similar meanings [Deerwester et al., 1990]. As follows we are going to describe the steps we have performed in order to obtain our *"meaning" representation* from word cooccurrence information.

5.4.1 Cooccurrence word matrix generation

We have selected the aggregated vocabulary resulting from ensemble number two as base for this experimentation. We considered this vocabulary to be an adequate one for this process.

A "*window*" representing a span of words is passed over the corpus being analyzed. Note that we only contemplated terms in the corpus belonging to the selected vocabulary. The width of this window can be adjusted. In this work we have evaluated the following window width values: 1, 5 and 10. The higher the window value, the wider the neighborhood that is evaluated. Words within this window are recorded as co-occurring with a strength inversely proportional to the number of other words separating them within the window. Let be "*The horse raced past the barn fell*" an example sentence. The words "*raced*" and "*past*" would receive a maximum cooccurrence value, while "*horse*" and "*barn*" would be considered to co-occur weakly (if the window is wide enough to include them both).

By moving this window over the source corpus in one word increments, and recording at every window movement the cooccurrence values of the words within it, a cooccurrence matrix can be generated. This matrix has n x n dimensions, with n being the cardinality of the vocabulary. Each cell of the matrix represents the summed cooccurrence counts for a single word pair. It is worth mentioning that counts for the "*ab*" pair and counts for the "*ba*" pair are in different cells, i.e., it is direction sensitive. This process produces a matrix in which, for every word in the vocabulary, there is both a row and a column containing relevant information. Rows contain cooccurrence information for words appearing before the word under consideration. Columns contain cooccurrence information for words appearing after the word under consideration.

Table 5.9 shows an example matrix computed for the sentence "*The horse raced past the barn fell*" using a widow width of five words.

	barn	fell	horse	past	raced	the
barn	0	0	2	4	3	6
fell	5	0	1	3	2	4
horse	0	0	0	0	0	5
past	0	0	4	0	5	3
raced	0	0	5	0	0	4
the	0	0	3	5	4	2

Table 5.9: Example matrix for "*The horse raced past the barn fell*" and a 5 width window

5.4.2 Cooccurrence vector generation

From the word cooccurrence matrix we are in position to represent each vocabulary word with a high dimensional cooccurrence vector. We construct each word vector by concatenating its row/column pair, so that given an n x n cooccurrence matrix, a $2n$ cooccurrence vector is available. This $2n$ vector can be conceptualized as representing a word in a $2n$ high dimensional space.

We can use these vectors to explore the existing relationships among words by applying distance metrics within the $2n$ dimensional space. Words that are together in this space are semantically similar and words that are far apart are semantically distant. We have decided to use *cosine similarity* as distance metric, since it is not sensitive to vector magnitude. It is a nice property, since movie review comment length is not normalized. It just measures the cosine of the angle between the two vectors, i.e., direction rather than magnitude.

This is the step presenting an enormous computational cost, from both time and space points of view. Recall that we have to calculate the distance between all the possible combinations of words in the vocabulary. Due to its high temporal cost it was not feasible to perform the calculation in a sequential way. We decided to split it in "little" subtasks (it is possible since each task is independent), and to execute each one of the small subtasks in parallel, by using pomar (see 4.3). As a result we have obtained a matrix D of distances with n x n dimensions.

5.4.3 Building our meaning cluster

At this point we have a matrix that contains information about the distances among all the words comprising the vocabulary. As we have used cooccurrence information in order to calculate these distances, we are in position to applicate the *distributional hypothesis*. Words at a low distance are expected to have similar semantic meaning. From the distance matrix, we have applied a simple clustering algorithm in order to assign words having a similar semantic meaning to the same group, called *cluster*. We have called *meaning cluster* to each one of the obtained groups.

We have used a simple linkage clustering algorithm. The algorithm is composed of the following steps:

1. Iterate over all the words in the vocabulary. In the beginning the number of clusters is equal to 0.
2. For each word w, look for the word t having the higher similarity with respect to w (lowest distance).
3. Let s be the measured similarity between w and t. If s is higher or equal than a given threshold, then apply one of the following cases:
   - If both words are not assigned to a cluster, create a new cluster for them. Increase the cluster counter in one unit.
   - If both words have a cluster assigned, then mix both clusters (set union). Decrease the cluster counter in one unit.
   - Else, assign the word without cluster to the other word's cluster.

   If s is lower than the given threshold, then assign w to the cluster number 0. It is a special cluster in which all the isolated words are assigned.

Ideally, each one of the clusters represents a meaning. We have now to convert our cluster structure to a valid Unigrams representation for the classification algorithm. We want each feature to represent meaning information instead of word/term information. Thus, for each cluster we have aggregated the feature information corresponding to the words comprising the cluster. For the presence/absence representation two different aggregation functions have been evaluated. On one hand, as in this case the information is binary, we have utilized a *maximum* function over all the words in the cluster. That is to say the feature representing the cluster takes a value of 1 when some word in the cluster appears in the document, and 0 otherwise. On the other hand we propose using a *sum* function, that outputs the number of cluster words appearing in the document. As the cluster length is not constant, we have normalized the sum function by dividing its result by the cluster size (number of words comprising it).

As regards the experiments in this section, we have created three cooccurrence matrices with respective windows of one, five and ten. For each resulting distance matrix, we have executed the linkage clustering with a series of thresholds: 0, 0.5 and 0.8. We have used both Unigrams presence/absence and frequency information. Note that it makes no sense to use max aggregation function with a frequency based representation. The obtained results in terms of accuracy and number of utilized features are presented in Table 5.10. The number of features is equivalent to the number of resulting clusters.

Regardless of the configuration, all the experiments using meaning rather than word information have yielded to a significantly lower accuracy. The best result is obtained using a window of 5, frequency information with a normalized sum aggregation and a clustering threshold of 0.5. This model is able to correctly predict 73.21% of test examples. Even though it is far from the performance it has been obtained

	Clustering threshold					
	0		0.5		0.8	
Representation	Acc	# Features	Acc	# Features	Acc	# Features
Presence-1-Max	65,357	400	66,071	376	65,714	179
Presence-5-Max	66,429	289	67,857	263	66,429	152
Presence-10-Max	66,786	205	66,429	199	62,857	120
Presence-1-NormSum	67,857	400	67,857	376	64,643	179
Presence-5-NormSum	68,929	289	66,786	263	70	152
Presence-10-NormSum	65	205	66,071	199	65	120
Frequency-1-NormSum	67,857	400	68,929	376	71,071	179
Frequency-5-NormSum	68,929	289	**73,214**	263	72,5	152
Frequency-10-NormSum	69,643	205	68,929	199	70	120

Table 5.10: Obtained results in the meaning cluster experiments in function of the measured accuracy and number of used features. The representation legend is readed as follows: Unigrams representation-Window-Aggregation function.

in the previous experiments, the model clearly goes beyond the minimum expected performance of 50%. Thas is to say our meaning representation is a valid input for a learner to induce a model that solves the sentiment analysis problem.

As regards the parameters, the best results are in most cases obtained with a window of 5. In general it also seems that better results are obtained when using 0.5 and 0.8 clustering thresholds instead of not using it (0). Contrasting with baseline conclusions, it seems that frequency information is more adequate than presence/absence information when using meaning features rather than word/term features. Intuitively, it seems that a repeating a meaning is more relevant that simply repeating a word.

In order to show the effectiveness of our approach, regarding word meaning similarity within clusters, we are presenting a series of representative cluster examples in Table 5.11. Example number 1 presents words that are nouns and that represent places or locations. Example number 2 is basically comprised of month related words. A series of adjectives that are typically used to express strong emotions are present in example number 3. The remaining examples are pretty self explanatory.

Lets take a look at example number 3. Even though all words belonging to the cluster are adjectives typically used to express strong sentiments, some of them are typically positive and others are negative. Our approach does not take into account the class when performing the clustering task. Thus, within a meaning, it is possible to have words representing opposite sentiments. This could be an explanatory reason for our proposed and novel approach not performing as good as other representations.

Example number	Cluster words							
1	accident crash sequence	apartment mental sign	bank moment store	bar place town	bathroom point vehicle	car restaurant weekend	cell room	competition scene
2	april september	copyright tristar	december	january	july	march	october	ratio
3	beautiful rich	bland sinister	chilling superb	comical talented	gorgeous terrific	magical tragic	nazi	pathetic
4	boy	girl						
5	boyfriend	husband						
6	difficult	easy	hard	impossible				
7	directed	photographed	produced	written				
8	could	may	might	must	should	shouldn't	would	wouldn't

Table 5.11: Representative cluster examples with a window of 10 and a threshold of 0.8

Bibliography

A. Abbasi, H. Chen, and A. Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–34, 2008.

A. Abbasi, S. France, Z. Zhang, and H. Chen. Selecting attributes for sentiment classification using feature relation networks. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

R. Agrawal, S. Rajagopalan, R. Srikant, and Y. Xu. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th international conference on World Wide Web*, pages 529–535. ACM, 2003.

M. Aizerman, E Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

S. Argamon-Engelson, M. Koppel, and G. Avneri. Style-based text categorization: What newspaper am i reading. In Proc. of the AAAI Workshop on Text Categorization, pages 1–4, 1998.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

S. Asur and B.A. Huberman. Predicting the future with social media. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 492–499. IEEE, 2010.

D. Biber. Variation across speech and writing. *Cambridge University Press*, 1988.

J. Bollen, H. Mao, and X.J. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2011.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

M.M. Bradley and P.J. Lang. Affective norms for english words (anew): Instruction manual and affective ratings. *University of Florida: The Center for Research in Psychophysiology*, 1999.

E. Cantu-Paz. Feature subset selection, class separability, and genetic algorithms. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 959–970. Springer, 2004.

P. Carvalho, L. Sarmento, M.J. Silva, and E. de Oliveira. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM, 2009.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

I.G. Councill, R. McDonald, and L. Velikovich. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59. Association for Computational Linguistics, 2010.

S. Das and M. Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of APFA'01 Annual Conference*, pages 37–56, 2001.

K. Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW'03*. ACM, 2003.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. 1990.

T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

M. Efron. Cultural orientation: Classifying subjective documents by cociation analysis. In *Proceedings of AAAI'04 Fall Symposium on Style and Meaning in Language, Art, Music, and Design*, pages 41–48, 2004.

A. Esuli and F. Sebastiani. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 617–624. ACM, 2005.

Z. Fei, J. Liu, and G. Wu. Sentiment classification using phrase patterns. In *Computer and Information Technology, 2004. CIT'04. The Fourth International Conference on*, pages 1147–1152. IEEE, 2004.

M. Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of COLING'04*, page 841. Association for Computational Linguistics, 2004.

G. Grefenstette, Y. Qu, J.G. Shanahan, and D.A. Evans. Coupling niche browsers and affect analysis for an opinion mining. In *In Proceedings of RIAO*. Citeseer, 2004.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

V. Hatzivassiloglou and K.R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of EACL'97*, pages 174–181. Association for Computational Linguistics, 1997.

V. Hatzivassiloglou and J.M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of COLING'00*, pages 299–305. Association for Computational Linguistics, 2000.

M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of AAAI'04 National Conference on Artificial Intelligence*, pages 755–760. AAAI Press/The MIT Press, 2004a.

M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD'04*, pages 168–177. ACM, 2004b.

G. Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, 1968.

L. Jia, C. Yu, and W. Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1827–1830. ACM, 2009.

J. Karlgren and D. Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 1071–1075. Association for Computational Linguistics, 1994.

R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 40–48. ACM, 1996.

B. Liu. *Handbook of Natural Language Processing*, chapter Sentiment Analysis and Subjectivity, pages 1–38. CRC Press, Taylor and Francis Group, 2010a.

B. Liu. Sentiment analysis: A multi-faceted problem. *IEEE Intelligent Systems*, 25(3), 2010b.

B. Liu, M. Hu, and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW'05*, page 351. ACM, 2005.

K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

Y. Mao and G. Lebanon. Isotonic conditional random fields and local sentiment flow. In *Proceedings of NIPS'07*, 2007.

G. Mishne. Experiments with mood classification in blog posts. In *Proceedings of ACM SIGIR 2005 Workshop on Stylistic Analysis of Text for Information Access*. Citeseer, 2005.

T.M. Mitchell. *Machine Learning*, volume 1. McGraw Hill, 1997.

S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. Mining product reputations on the web. In *Proceedings of ACM SIGKDD'02*, pages 341–349. ACM, 2002.

T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP'04*, pages 412–418, 2004.

K. Nigam and M. Hurst. Towards a robust metric of opinion. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 598–603, 2004.

A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC'10*, pages 1320–1326, 2010.

B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL'04 Annual Meeting*. Association for Computational Linguistics, 2004.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL'05 Annual Meeting*, pages 115–124. Association for Computational Linguistics, 2005.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP'02*, pages 79–86. Association for Computational Linguistics, 2002.

D.L. Poole, A.K. Mackworth, and R. Goebel. *Computational intelligence: a logical approach*. Oxford University Press, USA, 1998.

E. Riloff and J. Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP'03*, page 112. Association for Computational Linguistics, 2003.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill College, 1983.

P. Subasic and A. Huettner. Affect analysis of text using fuzzy semantic typing. *Fuzzy Systems, IEEE Transactions on*, 9(4):483–496, 2001.

R.M. Tong. An operational system for detecting and tracking opinions in on-line discussion. In *SIGIR'01 Workshop on Operational Text Classification (OTC)*, 2001.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

P.D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL'02 Annual Meeting*, pages 417–424. Association for Computational Linguistics, 2002.

P.D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010. ISSN 1076-9757.

V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider machine learning toolbox, 2005.

C. Whitelaw, N. Garg, and S. Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

J.M. Wiebe. Tracking point of view in narrative. *Computational Linguistics*, 20(2): 233–287, 1994.

J.M. Wiebe. Learning subjective adjectives from corpora. In *Proceedings of AAAI'00 National Conference*, page 735, 2000.

J.M. Wiebe, R.F. Bruce, and T.P. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of ACL'99 Annual Meeting*, pages 246–253. Association for Computational Linguistics, 1999.

J.M. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin. Learning subjective language. *Computational linguistics*, 30(3):277–308, 2004.

M. Wiegand, A. Balahur, B. Roth, D. Klakow, and A. Montoyo. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68. Association for Computational Linguistics, 2010.

T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP'05*, page 354. Association for Computational Linguistics, 2005.

Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML'97*, pages 412–420, 1997.

J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of IEEE ICDM'03*, page 427. IEEE Computer Society, 2003.

H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP'03*, pages 129–136. Association for Computational Linguistics, 2003.