Universidad de Oviedo

Máster Universitario en Modelización e Investigación Matemática, Estadística y Computación

Trabajo Fin de Máster

Reconocimiento de patrones proyectivo-invariantes mediante redes neuronales de variable compleja

Autor: Pablo José Anta González
Director: D. César Luis Alonso González

Julio, 2015

A mi madre

Indice

Indice de figuras y procedencia

Indice de tablas

0. Objetivos y organización del Proyecto.

Una de las tareas más importantes de la Visión por Computador es el Reconocimiento de Patrones, que consiste en clasificar objetos representados en imágenes a partir de información extraída de éstas. A pesar de su importancia (es indispensable para que los ordenadores "entiendan" las imágenes) y de los esfuerzos invertidos en su desarrollo, sin embargo, aún presenta desafíos. La causa de ello radica en la gran variabilidad que muestra el aspecto de los objetos[1] por cambios en el punto de vista, en la escala, cambios de iluminación, de color, etc; de entre ellos, destacan los primeros, los llamados efectos de perspectiva, por la dificultad que entrañan, y son de los que nos vamos a ocupar en este Proyecto.

Geométricamente, las deformaciones que sufren las figuras por cambios en el punto de vista se describen mediante transformaciones proyectivas, de las cuales en Reconocimiento de Patrones se han empleado fundamentalmente las homografías, en particular las afinidades cuando la distancia de la figura a la cámara es mucho mayor que el tamaño de la figura. El problema es que la teoría de homografías (y la de afinidades, pues es un caso particular) sólo se puede aplicar en la clasificación de figuras contenidas en planos[2]. Entonces, ¿qué hacer si las figuras están incluidas en superficies bidimensionales, pero no planas?

En la literatura, el reconocimiento y clasificación de figuras bidimensionales se resuelve mediante la técnica del image warping, que consiste en reconstruir las superficies bidimensionales que contienen a las figuras. Aunque existen diversas técnicas para hacer esta reconstrucción [113] [109] [69] [26], todas ellas constan de dos fases: 1) se identifican una serie de puntos homólogos en las imágenes 2) a partir de éstos se construye una superficie que cumple algún criterio de optimización. La superficie se puede representar por el producto tensorial de B-splines [113], NURBS [26], etc. En general, estos métodos sólo funcionan bien cuando las imágenes presentan perspectiva afín, lo cual es una limitación; únicamente, el método Schwarps [72] considera proyectividades generales, pero la superficie se obtiene resolviendo un sistema de ecuaciones diferenciales parciales 2D, lo que requiere disponer de bastante información local bidimensional de las imágenes.

Supongamos, y éste es el objeto del Proyecto, que se dispone de un conjunto de curvas, las cuales representan los contornos de unas figuras dibujadas en unas superficies planas orientadas aleatoriamente; a continuación, estas superficies se deforman sin rasgaduras, también de forma aleatoria; lo único que se exige es que al final las figuras no presenten oclusiones al proyectarse en el plano de proyección. La clasificación de estas curvas es un problema de clasificación proyectivo-invariante, pero de la discusión anterior se desprende que en la resolución de este problema no es posible aplicar los métodos tradicionales.

---

[1] En este Proyecto sólo nos ocuparemos de objetos bidimensionales, que a partir de ahora denominaremos figuras.

[2] Las homografías 3D consisten en proyectar sobre un plano β' las figuras planas contenidas en otro plano β; haciendo abstracción de β y β', es decir, identificando ambos planos con un plano genérico, se puede considerar a las homografías una transformación 2D.

La solución aportada en este Proyecto consiste en combinar un detector de puntos dominantes de los contornos de las figuras con una red neuronal de variable compleja que tome como entradas dichos puntos y clasifique las figuras.

La razón de la elección de una red neuronal de variable compleja y no una real, que es lo usual, se debe a la mayor capacidad funcional de las primeras. Problemas que resuelven fácilmente las redes de variable compleja no pueden ser resueltos por sus homólogas reales. En la primera parte del Proyecto se describen las redes neuronales de variable compleja, con sus particularidades y problemas, y se detalla un algoritmo de retropropagación complejo para su entrenamiento.

Los puntos dominantes de una curva son aquellos con un valor de curvatura elevado. Estos puntos tienen mucha importancia en visión por computador, pues se demuestra que son los que captan la atención y permiten, en gran medida, reconocer las figuras. El problema es que su cálculo requiere hacer derivadas, y esto siempre es complicado cuando se trabaja con imágenes. Por lo tanto, en este Proyecto para obtenerlos se ha escogido un método desarrollado por Di Ruberto y Morgera [35] [36] [37] [98], que es simple y no requiere derivar.

El Proyecto se divide básicamente en dos partes: la primera, que abarca los tres primeros capítulos, se dedica al estudio de las redes neuronales, en concreto de los perceptrones, tanto en la versión real como en la compleja; la segunda parte, formada por los dos últimos capítulos, versa sobre las transformaciones proyectivas y la aplicación de las redes neuronales de variable compleja en problemas de reconocimiento de patrones proyectivo-invariantes. Al final del Proyecto se han incluido unos apéndices que resumen algunos conceptos necesarios para seguir los razonamientos del Proyecto.

A continuación, una descripción más detallada de la organización del Proyecto:

- *Capítulo uno*. Se presentan brevemente los principales hitos en el desarrollo de las redes neuronales.

- *Capítulo dos*. Se repasan los conceptos principales de los perceptrones multicapa de variable real.

- *Capítulo tres*. El capítulo se divide en cuatro partes: 1) se estudia la problemática asociada a las redes neuronales de variable compleja, 2) se desarrolla el algoritmo de retropropagación complejo originalmente presentado por Nitta para entrenar al perceptrón multicapa de variable compleja, 3) se analiza la adición de un término momento al algoritmo de retropropagación con la fin de acelerar su convergencia 4) se demuestra que las fronteras de decisión de los perceptrones de variable compleja son ortogonales, lo cual explica la mayor capacidad funcional que tienen estos perceptrones en comparación con sus homólogos reales..

- *Capítulo cuatro*. Se estudian las transformaciones proyectivas, tanto las afinidades como las homografías, así como los invariantes correspondientes a cada uno. Un supuesto esencial de las homografías es que las figuras

proyectadas sean planas; si este no es el caso estamos ante una proyectividad general. La última parte del Capítulo trata brevemente de este tema.

- *Capítulo 5*. Se describen los puntos dominantes y cómo combinarlos con un perceptrón de variable compleja para resolver diversos problemas de proyectividades.

- *Apéndice uno*. Recopila una serie de expresiones útiles para el trabajo con las funciones de activación de las neuronas.

- *Apéndice dos*. Se hace un breve resumen de los conceptos de la Teoría matemática de la Probabilidad necesarios para seguir las argumentaciones del Proyecto.

- *Apéndice tres*. Se hace un breve resumen de los conceptos de la Teoría de la Información necesarios para seguir las argumentaciones del Proyecto.

- *Apéndice cuatro*. Se explica muy brevemente el fundamento del Cálculo de Wirtinger.

- *Apéndice cinco*. Se desarrolla el cálculo de la longitud de arco de curva y de la curvatura invariantes a transformaciones afines.

1. Introducción histórica a las redes neuronales

Las redes neuronales artificiales son sistemas computacionales inspirados en las redes neuronales biológicas, diseñados con el fin de emular la capacidad que tiene el cerebro de resolver ciertas tareas complejas, denominadas del mundo real, en las que la información que se procesa es masiva, distorsionada e imprecisa [90], y para las cuales no resultan adecuados los paradigmas tradicionales de la inteligencia artificial, basados en el tándem lógica booleana-máquina de Von Neuman [90]. Un ejemplo típico de esta clase de problemas es el reconocimiento de patrones. Las redes neuronales, las redes bayesianas, los algoritmos evolutivos, la lógica fuzzy,... son técnicas de procesamiento de información que se engloban bajo el nombre de Soft Computing [90][106].

La teoría de las redes neuronales artificiales comienza en 1943 con la publicación del artículo "A logical calculus of the ideas immanent in neurons activity" [93], en el que McCulloch y Pitts presentan el primer modelo matemático de la actividad de una neurona aislada, así como un estudio de sus capacidades computacionales.

Las neuronas de McCulloch y Pitts tienen las siguientes características [110]:

1) Son unidades procesadoras biestado (activo o inactivo); las sinapsis que las conectan también son biestado (excitado o inhibido).

2) Para que una neurona se active en un instante *t* es necesario que previamente, durante un intervalo temporal fijo denominado tiempo de latencia, se cumplan simultáneamente dos condiciones:

    - Se exciten *n* sinapsis de la neurona, siendo *n* un número fijo que no depende de la actividad previa de la neurona ni de su posición en la red.

    - No se inhiba ninguna sinapsis; en caso contrario, la activación de la neurona sería neutralizada.

3) Los pesos de la neurona son fijos.

4) La excitación de las sinapsis conlleva un cierto retardo, denominado retardo sináptico.

5) Los datos de entrada a la neurona se expresan en un alfabeto {0,1}.

En la figura se muestra un esquema simplificado de la neurona de McCulloch-Pitts:

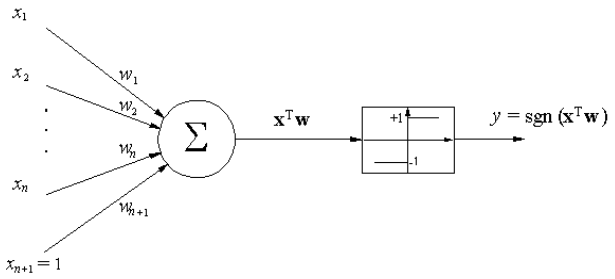


Figura 1.1. La neurona de McCulloch-Pitts (los pesos son fijos)

Lo verdaderamente interesante de este modelo de neurona es que su característica "todo o nada" le permite actuar como una calculadora lógica, es decir, representar cualquier expresión proposicional finita. Por lo tanto, en principio es posible construir cualquier circuito lógico (incluso un ordenador) combinando estas neuronas [148].

El artículo de McCulloch y Pitts tuvo gran influencia en el desarrollo posterior de las redes neuronales, e incluso de los ordenadores basados en el paradigma de Von Neumann; no obstante, es necesario señalar que, desde el conocimiento actual que se tiene de la fisiología de las neuronas, el modelo que plantea es erróneo. Por otro lado, una red neuronal compuesta de neuronas de McCulloch y Pitts tiene un campo de aplicación estrecho debido a la naturaleza binaria de sus neuronas y al valor constante de los pesos. No obstante, este modelo supuso un primer paso decisivo.

Durante los quince años siguientes a la publicación del artículo original de McCulloch y Pitts fueron apareciendo nuevos conceptos y desarrollos que enriquecieron la teoría de redes neuronales artificiales[3]. Uno de los más importantes fue la introducción en 1949 de la regla de Hebb, presentada como una hipótesis del funcionamiento de las sinapsis (las sinapsis que funcionan según la regla de Hebb se denominan hebbianas) por el psicólogo canadiense Donald Hebb en su libro "The organization of behaviour" [61], a partir de los datos de la fisiología del cerebro disponibles en su época. Hebb establece que la transmisión de información entre dos neuronas unidas por una sinapsis (transmisión sináptica) tiene dos características fundamentales [59]: 1) depende de la sincronía en la activación de dichas neuronas, 2) depende de la acción conjunta de ambas neuronas. Por lo tanto, la transmisión de información sináptica es un fenómeno local, pues requiere contigüidad espacio-temporal en la actividad de las neuronas implicadas en la transmisión, y es un mecanismo que precisa de una correlación estadística positiva entre las activaciones de las neuronas presinápticas y postsinápticas. [59] presenta una versión actualizada de la regla de Hebb:

- La fuerza de una sinapsis que conecta dos neuronas se incrementa si éstas se activan simultáneamente.

[3] Todas las redes neuronales discutidas en el Proyecto son artificiales, por lo tanto a partir de ahora se va a prescindir de este adjetivo al nombrarlas, sin que de lugar a confusión.

- En cambio, si las neuronas se activan de forma asíncrona, la fuerza de la sinapsis se debilita o anula (este punto no estaba en la versión original de la regla).

Aunque la regla de Hebb nace del estudio de la fisiología del cerebro, en el fondo enuncia una observación de sentido común: fenómenos correlacionados positivamente están efectivamente relacionados entre si (obviamente, no se consideran las correlaciones espurias). La importancia de esta regla es que establece un mecanismo para modificar la fuerza de las sinapsis; es decir, para modificar lo que en la teoría actual de redes neuronales son los pesos de la red (recordemos que en el modelo de McCulloch y Pitts los pesos eran fijos). Basándose en este principio, a lo largo de los años cincuenta diversos investigadores fueron desarrollando nuevos modelos de redes neuronales. Por ejemplo, Uttley demostró en 1956 que una red neuronal con sinapsis variables podía aprender a hacer clasificaciones binarias [130].

El siguiente hito en la evolución de las redes neuronales lo constituyó el Perceptrón, desarrollado por Frank Rossenblatt en 1957 en el Laboratorio Aeronáutico de Cornell [116]. El Perceptrón original estaba formado por tres capas de neuronas: una capa de *N* neuronas de entrada, las cuales recogían la información procedente de un conjunto de sensores denominado retina (que no formaba parte de la red), una capa de *M* neuronas de asociación y una capa de salida. La tarea impuesta a la red era aprender una relación binaria

$$d : \{-1,1\}^N \rightarrow \{-1,1\}$$

a partir de un conjunto de pares de ejemplos $\left(\vec{x}_p, d\left(\vec{x}_p\right)\right)$.

En la versión original del Perceptrón, la conectividad entre la retina y la capa de entrada era determinista y no-adaptativa, y aleatoria entre las restantes capas. La siguiente figura muestra un esquema de esta red:

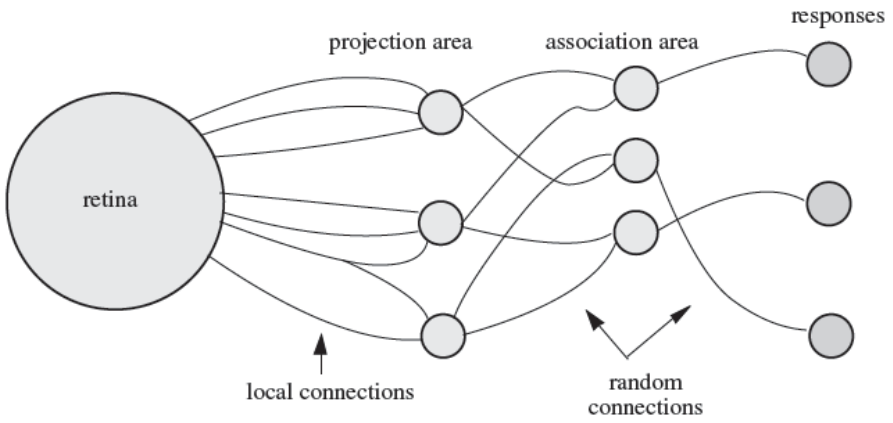


Figura 1.2. El Perceptrón clásico de Rossenblatt

El Perceptrón de Rossenblatt fue simplificado por Minsky y Papert [94]. El funcionamiento del nuevo modelo se ilustra en la figura 1.3: un conjunto de sensores escanea diversas regiones de una imagen, y esta información se pasa a las funciones

binarias ϕ_h, denominadas funciones predicado (representadas en la figura por los paralepípedos); las salidas de estas funciones se combinan linealmente en la neurona Ω, y el resultado se toma como entrada de una función binaria Ψ, de cuya salida se infiere si en la figura hay una cierta forma geométrica (por ejemplo, una X) o no.

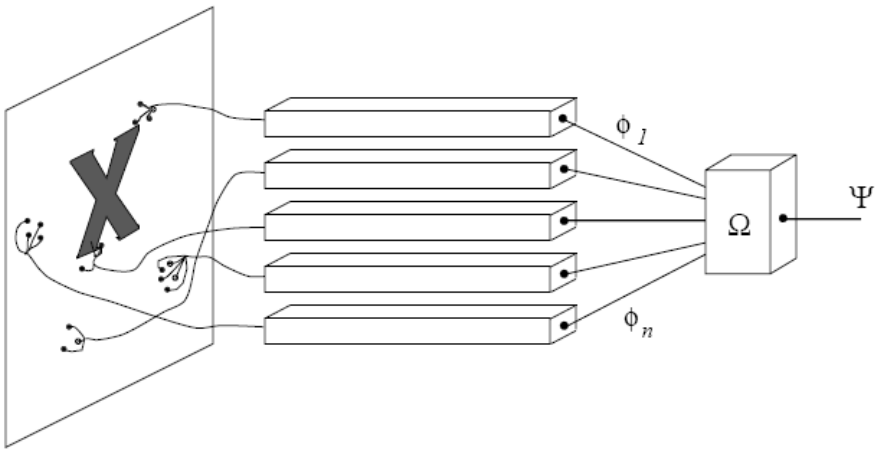


Figura 1.3. El Perceptrón simplificado de Minsky y Papert.

La versión actual del Perceptrón prescinde de la retina, sus entradas pueden tomar cualquier valor y los pesos no son fijos. Esquemáticamente:

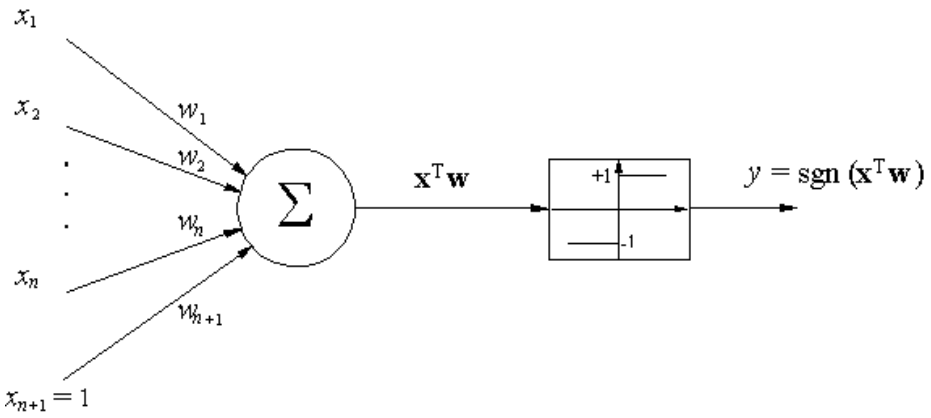


Figura 1.4. El Perceptrón (los pesos son variables)

Podemos observar que el esquema del Perceptrón es similar al de la neurona de McCulloch y Pitt. La diferencia está en que en el perceptrón los pesos pueden variar, mientras que en la neurona de McCulloch y Pitt no. El entrenamiento del Perceptrón consiste en ajustar los pesos mediante el siguiente algoritmo de aprendizaje [78]:

1. Los pesos se inicializan con valores aleatorios.

2. Se selecciona un ejemplo de entrada $\vec{x}$.

3. La red calcula una salida y a partir de $\vec{x}$. Si $y \neq d\left(\vec{x}\right)$ se modifican los pesos mediante la expresión $\Delta w_h = d\left(\vec{x}\right) \cdot x_h$; si $y = d\left(\vec{x}\right)$ no se modifican los pesos.

4. Continuar en 2. hasta que todos los ejemplos cumplan $y = d\left(\vec{x}\right)$.

El respaldo teórico a la eficacia del Perceptrón descansa en el Teorema de convergencia del perceptrón, que se puede enunciar [78]:

> *Si existe un conjunto de pesos $\vec{w}^*$ que permiten al perceptrón representar la transformación $y = d\left(\vec{x}\right)$, entonces, para cualquier elección inicial de los pesos, la regla de aprendizaje del perceptón converge a una solución (que puede ser igual o no a $\vec{w}^*$) en un número finito de pasos.*

Desde el momento de la publicación del trabajo de Rossenblatt, y en muy breve tiempo, numerosos investigadores fueron presentando nuevos modelos de redes neuronales (Adaline de Widrow y Hoff [139] entrenado mediante Mean Least Square (LMS), Madaline de Widrow [138], etc) que resultaron tener éxito como filtros de señales, ecualizadores, etc, aparte de la ventaja de la sencillez de su implementación en circuitos analógicos o VLSI.

Sin embargo, el panorama favorable al paradigma neuronal se vio frenado a partir de 1969, año en el que los investigadores Marvin Minsky y Seymour Papert publicaron el libro Perceptrons [94], en el cual señalaban los problemas que tenían los perceptrones para resolver tareas de clasificación que fueran más complejas que la simple clasificación lineal: si los perceptrones eran monocapa, eran incapaces de resolver un problema como el XOR exclusivo u operador de comparación; si eran multicapa, en teoría podían resolver este problema, pero en aquel momento no se conocía un algoritmo capaz de distribuir el error (la diferencia entre la salida de la red asociada a un determinado patrón de entrada, y el correspondiente patrón de salida) entre los pesos de las distintas capas durante la fase de entrenamiento, lo cual significaba que en la práctica estas redes no se podían entrenar. Aunque algunos autores afirman que la influencia nefasta de dicho estudio es realmente un mito ([10] considera que el freno al desarrollo de las redes neuronales debe atribuirse no tanto al citado estudio como a la competencia de otras técnicas de la inteligencia artificial que en aquella época resultaron más atractivas, así como al aumento de la potencia de los computadores digitales), lo cierto es que el paradigma neuronal estuvo condenado al ostracismo durante todos los años setenta. Posiblemente, lo más reseñable realizado en esta época sea el trabajo realizado por algunos investigadores con las redes auto-organizadas de entrenamiento competitivo [132] [140].

Sin embargo, esta situación de estancamiento varió a partir de los años ochenta. A continuación se exponen brevemente algunos de los avances más significativos que se han dado desde entonces en el campo de las Redes Neuronales son [59]:

- En 1980 Fukushima presenta el Neocognitrón [46], una red de aprendizaje no supervisado capaz de clasificar objetos en imágenes. Esta red se auto-organiza en un sistema jerárquico de capas.

- En 1980 Grossberg descubre un nuevo principio de auto-organización denominado Adaptive Resonance Theory (ART) [53].

- Hopfield descubre en 1982 el paralelismo existente entre las Redes Neuronales Recurrentes con sinapsis simétricas y el modelo de Ising de la física estadística [68]. Las redes de Hopfield son un ejemplo de redes dinámicamente estables que pueden almacenar información.

- En 1982 Kohonen presenta las redes que llevan su nombre, en las cuales la información se representa en forma de mapas bidimensionales de características [76].

- En 1983 Barto, Sutton y Anderson publican un importante artículo sobre Aprendizaje por Refuerzo (Reinforcement Learning) [18], que es un tipo de auto-aprendizaje (por lo tanto, no confundir con el Aprendizaje Supervisado) en el que los agentes toman decisiones racionales con el fin de maximizar una recompensa acumulable proporcionada por el entorno.

- En 1986 Rumelhart, Hinton y Williams publican su famoso trabajo sobre el algoritmo Back-Propagation de entrenamiento de perceptrones multicapa [114][115]. Este algoritmo resuelve el problema señalado en el libro de Minsky y Papert. Es el algoritmo de entrenamiento más ampliamente utilizado para entrenar a los perceptrones multicapa.

- En 1988 Broomhead y Lowe describen las Redes de Base Radial (Radial Basis Functions, RBF) [25], que constituyen una alternativa a los perceptrones multicapa.

- Aunque el primer artículo que menciona el uso combinado de lógica fuzzy y redes neuronales fue escrito por los hermanos Lee en 1974 [81], es durante los años ochenta que la unión de ambas técnicas cobra importancia. La fusión de las redes neuronales y la lógica fuzzy se puede llevar a cabo de dos formas [106]: 1) reemplazando los valores escalares de la redes neuronales tradicionales por regiones fuzzy (Fuzzy-Neural Network, FNN), 2) potenciando alguna de las características de un sistema fuzzy (velocidad, flexibilidad, etc) mediante la ayuda de una red neuronal auxiliar, actuando ambas técnicas por separado, pero de forma colaborativa, en distintas partes del mismo problema (Neural-Fuzzy System, NFS). Basándose en que las reglas de la lógica fuzzy son diferenciables [107], en 1988 Paul Verbos aplicó por primera vez el algoritmo de backpropagation en una FNN [135]. Desde entonces, en la literatura es posible encontrar una ingente cantidad de ejemplos de la combinación de ambos paradigmas.

- En 1990 se presentan las Máquinas de Soporte Vectorial (Support Vector Machines) [24], de gran utilidad en reconocimiento de patrones.

- La idea de entrenar las redes neuronales haciendo uso de algoritmos genéticos aparece ya en 1975 en el libro “Adaptation in natural and artificial systems” de John H. Holland [67]; no obstante, no es hasta los años noventa que la combinación de ambas técnicas adquiere relevancia. En 1989 Montana y Davis anuncian que han logrado entrenar una red neuronal mediante algoritmos genéticos [96], y señalan que el procedimiento ha dado resultados aún mejores que con un entrenamiento backpropagation. Un detalle importante es que las técnicas de algoritmos genéticos no funcionan bien para optimizar los pesos (este problema se encuentra en la literatura bajo el nombre de Competing Convections Problem [121] [96], o Permutations problem [111]); por lo tanto, es en la evolución de la topología de la red donde los algoritmos genéticos demuestran su poder [137].

- A partir de los años noventa se advierte en la literatura un fuerte crecimiento del número de aplicaciones que combinan las redes neuronales con la lógica difusa y los algoritmos evolutivos (algoritmos genéticos [75], programación genética [145], algoritmos de enjambre [39], etc). En la literatura, a este conjunto de técnicas computacionales se les engloba bajo el nombre de Inteligencia Computacional (Computacional Intelligence) [106].

- Los primeros estudios referentes a Redes Neuronales de Variable Compleja fueron realizados por investigadores rusos durante los años 70 y 80, pero fueron publicados en ruso y tuvieron poca repercusión internacional [5]. Sin embargo, a partir de los años noventa el estudio de las redes neuronales de variable compleja comienza a crecer de forma importante. En [5] se puede encontrar un breve resumen de la evolución de este campo de estudio:

    1. En 1971, Aizenberg, Ivaskiv y Pospelov describen por primera vez una función de activación compleja [6].

    2. En 1991 y 1992, H.Leung y S. Haykin [82], G. Georgiou y C. Koutsougeras [50], presentan de manera independiente un algoritmo de retropropagación de una red neuronal con pesos y funciones de activación complejos. Estos autores demuestran que su algoritmo converge mejor que la versión real.

    3. En 1992, Akira Hirose describe las Redes Neuronales Totalmente Complejas (Fully-Complex Neural Networks) [66] y el Algoritmo de Retropropagación Complejo (Complex Back-Propagation Algorithm) [64].

    4. Tohru Nitta presenta en 1997 un nuevo algoritmo de retropropagación complejo [101]. Es el primer investigador en ocuparse de las neuronas cuaternión [100].

    5. Simone Fiore extiende en 2003 y 2005 el concepto de Aprendizaje Hebbiano a las redes neuronales de variable compleja [42] [43].

    6. S. Buchholz y G. Sommer presentan en 2008 un estudio de la computación neuronal basada en las álgebras de Clifford (que son generalizaciones de los campos complejo y cuaternión) [27].

7. Algunos autores han centrado sus investigaciones en las aplicaciones de las redes neuronales de variable compleja. Por ejemplo, Goh y Mandic han realizado contribuciones al estudio del uso de estas redes como filtros de señales [51], y Md. F. Amin *et al*. las han empleado para resolver problemas de clasificación [12] [13].

2. El perceptrón multicapa de variable real

2.1. Definición de neurona y red neuronal.

La moderna teoría del perceptrón multicapa nace con la interpretación que Minsky y Papert hacen del perceptrón de Rossenblatt. Un perceptrón multicapa es un tipo particular de red neuronal; es necesario, por lo tanto, comenzar con la Teoría de las Redes Neuronales.

A continuación, se formaliza el concepto de neurona [30]:

Definición 1. **Neurona o Unidad Procesadora**

Una neurona o unidad procesadora, sobre un conjunto de nodos N, es una tripleta (X, f, Y), donde X es un subconjunto de N, Y es un nodo de N y $f : \mathbb{R} \to \mathbb{R}$ es una función neuronal (también llamada función de activación) que calcula un valor de salida para Y basado en los valores de los componentes de X:

$$Y = f\left(h\left(\{x_i; x_i \in X\}\right)\right)$$

donde $h\left(\{x_i; x_i \in X\}\right)$ es una función de los valores de las componentes de X y se denomina *actividad de la neurona*. X, Y y f son el conjunto de nodos de entrada, de salida y la función neuronal de la neurona, respectivamente.

Dependiendo de la naturaleza de la función h se distinguen distintos tipos de modelos neuronales. El más típico (es el que se va a utilizar en el resto del trabajo) es el modelo aditivo, correspondiente a:

$$h = h\left(\{x_i; x_i \in X\}\right) = \sum_{x_i \in X} w_i x_i + \theta$$

donde w_i es un peso o factor multiplicador que se aplica a la variable de entrada x_i y θ es un valor umbral, que puede interpretarse como el peso correspondiente a un nodo auxiliar cuyo valor siempre es 1. En el modelo aditivo, *h* se denomina actividad lineal de la neurona.

Los modelos aditivos son adecuados para representar y clasificar conjuntos de datos que cumplen:

a) sus variables representativas (variables de entrada al modelo) no interaccionan entre ellas

b) los bordes de decisión no son cóncavos

c) no existen regiones de decisión inconexas.

Si alguna de las condiciones anteriores no se cumple, el modelo aditivo no es capaz de clasificar los datos. Por ejemplo, en [92] aparece un ejemplo sencillo con dos regiones de decisión inconexas:

Se considera una partición de $\mathbb{R}$ en los conjuntos R_1 y R_2, tal como se muestra en la Figura 2.1. Una función discriminante lineal no puede crear una frontera que sirva como criterio para decidir si un punto de $\mathbb{R}$ pertenece a uno u otro conjunto; en cambio, al proyectar el dominio del problema en $\mathbb{R}^2$, una función cuadrática de la forma $f(x) = w_2 x^2 + w_1 x + w_0$ puede constituir una frontera plausible al elegir adecuadamente los pesos w_0, w_1 y w_2:

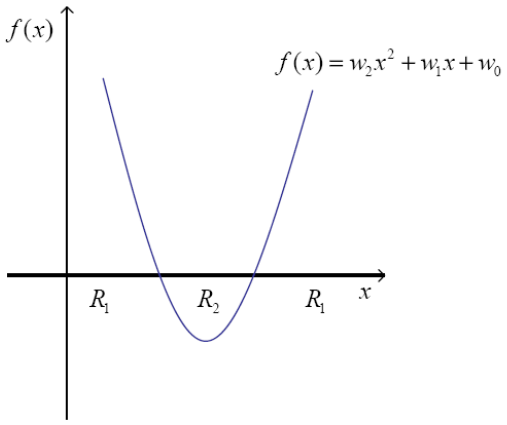


Figura 2.1 Ejemplo de función cuadrática que divide la recta real en dos regiones R_1 y R_2

El criterio de decisión será: $\forall x \in \mathbb{R}: \; x \in R_1 \Leftrightarrow f(x) > 0$.

En realidad, en casos como el del ejemplo, no es necesario acudir a un espacio de entrada de mayor dimensión si en vez de modelos aditivos se usan multiplicativos. El modelo multiplicativo más sencillo tiene la expresión:

$$h = \sum_{i=1}^{m} w_i M_i + \theta$$

donde cada uno de los M_i es un monomio de la forma:

$$M = x_1^{d_1} x_2^{d_2} \ldots x_k^{d_k}$$

siendo d_1, $d_2, \ldots, d_k$ enteros no negativos (a $d_1 + d_2 + \ldots + d_k$ se le denomina orden del monomio). Por lo tanto:

$$h = \sum_{i=1}^{m} w_i \prod_{k^i} x_{k^i}^{d_k^i} + \theta$$

Una neurona cuya actividad responde a la expresión anterior se denomina unidad sigma-pi.

El modelo multiplicativo permite construir polinomios de varias variables de la forma:

$$P = w_0 + \sum_{i=1}^{k} w_i x_i + \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} x_i x_j + \sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{k} w_{ijl} x_i x_j x_l + \ldots$$

mediante redes neuronales.

A partir de la definición formal de neurona se obtiene la definición formal de red neuronal [30]:

Definición 2. Red neuronal artificial

Una red neuronal artificial (RNA) es un par (N,U), donde N es un conjunto de nodos y U un conjunto de unidades procesadoras sobre N, que satisface la siguiente condición: cada nodo $X_i \in N$ tiene que ser un nodo de entrada o de salida de al menos una unidad procesadora de U.

En la Figura 2.2 [30] se muestra un ejemplo de red neuronal de ocho nodos $\{x_1,...,x_8\}$ y cinco unidades procesadoras:

$$U_1 = (\{x_1, x_2, x_3\}, f_1, \{x_4\})$$
$$U_2 = (\{x_1, x_2, x_3\}, f_2, \{x_5\})$$
$$U_3 = (\{x_1, x_2, x_3\}, f_3, \{x_6\})$$
$$U_4 = (\{x_4, x_5, x_6\}, f_4, \{x_7\})$$
$$U_5 = (\{x_4, x_5, x_6\}, f_5, \{x_8\})$$

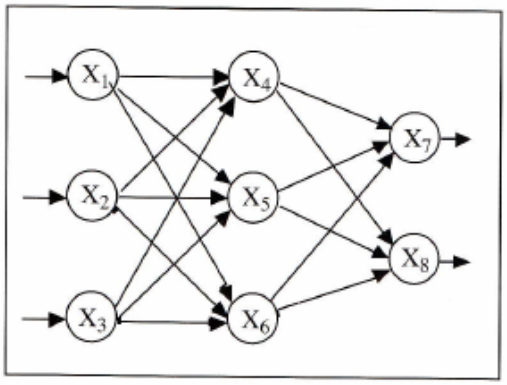


Figura 2.2. Una red neuronal artificial de ocho nodos

En resumen, una red neuronal es un conjunto de unidades procesadoras, denominadas neuronas, que intercambian información entre ellas mediante unas conexiones dirigidas con pesos; en cada neurona, la información procedente de otras neuronas se combina, dando lugar a la actividad (modelo aditivo, modelo sigma-pi,...), la cual, a su vez, es transformada por la función de activación f, generándose así la salida.

2.2 Funciones de activación.

Las redes neuronales pueden emplear diversas familias de funciones como funciones de activación. Los tipos más usuales son:

- **Funciones lineales**, que tienen la expresión:

$$f_c(x) = cx \qquad x \in \mathbb{R}$$

- **Funciones escalón**. Son funciones binarias, cuya salida depende de si el valor de entrada supera o no un determinado valor umbral. Las más típicas son:

  a) La *función signo*:

$$\operatorname{sgn}(x) = \begin{cases} 1 & x < 0 \\ -1 & x \geq 0 \end{cases} \qquad x \in \mathbb{R}$$

  b) La *función escalón*:

$$\Theta(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \qquad x \in \mathbb{R}$$

- **Las funciones sigmoidales**. Son funciones monótonas acotadas no lineales. Las más importantes son:

  a) La *función logística (o función de Fermi)*:

$$f_c(x) = \frac{1}{1+e^{-cx}} \qquad x \in \mathbb{R}$$

  b) La *función tangente hiperbólica*:

$$f_c(x) = \tanh(cx) = \frac{e^{cx} - e^{-cx}}{e^{cx} + e^{-cx}} \qquad x \in \mathbb{R}$$

  Se cumple (demostración en Apéndice 1):

$$y = \frac{1}{1+e^{-cx}} = \frac{1}{2}\left(\tanh\left(\frac{cx}{2}\right)+1\right)$$

A la hora de implementar en un ordenador una red neuronal es conveniente usar el siguiente desarrollo simplificado de $\tanh(cx)$:

$$f_c(x) = \frac{2}{1+e^{-2cx}} - 1$$

pues sólo requiere evaluar una exponencial frente a las cuatro de la expresión original. Esto proporciona ahorro computacional, ya que el cálculo de la exponencial de las librerías matemáticas usuales es lento [122]. En Matlab este desarrollo de $\tanh(cx)$ se denomina $\tan sig(cx)$. En el Apéndice 1 se demuestra que la expresión original de *tanh (cx)* y la simplificada son equivalentes.

Como se verá más adelante, en el entrenamiento de la red neuronal por el método de retropropagación (backpropagation) tiene mucha importancia la derivada de la función de activación. Por ello, resulta útil disponer de una lista con las derivadas de aquellas funciones de activación mencionadas más arriba que sean derivables:

a) Función lineal:

$$f_c'(x) = c \qquad x \in \mathbb{R}$$

b) Función logística

$$f_c'(x) = c f_c(x)\left(1 - f_c(x)\right) \qquad x \in \mathbb{R}$$

c) Función tangente hiperbólica

$$f_c'(x) = c\left(1 - f_c^2(x)\right) \qquad x \in \mathbb{R}$$

(La demostración de estas expresiones está en el Apéndice 1)

2.3. El perceptrón multicapa.

Un perceptrón multicapa es un tipo particular de red neuronal que: 1) tiene sus neuronas distribuidas en capas, 2) todas sus conexiones son hacia delante, 3) todas las neuronas de una capa se conectan con las neuronas de la capa siguiente. Las capas se clasifican así:

- Capa de entrada: formada por aquellas neuronas que reciben información del exterior de la red.

- Capa de salida: compuesta por las neuronas cuyas salidas son enviadas al exterior de la red.

- Capa intermedia u oculta: formada por neuronas cuyas entradas proceden de neuronas de la red y sus salidas permanecen dentro de la red (son entradas de otras neuronas).

El perceptrón multicapa está constituido por una capa de entrada, una capa de salida y un cierto número de capas ocultas. En el recuento del número de capas no se tiene en cuenta la de entrada. Por ejemplo, un perceptrón de una capa está constituido por las capas de entrada y salida; un perceptrón de dos capas consta de la capa de entrada, de la capa oculta y una capa de salida. Las conexiones de los perceptrones son del tipo conexiones hacia delante (las redes neuronales que cumplen esto se denominan redes feedforward), es decir, las neuronas de una capa se conectan con neuronas de la capa siguiente. Además, en un perceptrón todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente. En general, las redes neuronales pueden tener otros tipos de conexiones: laterales, o conexiones entre neuronas de la misma capa; recurrentes o hacia atrás, en los que las neuronas de una capa se conectan con neuronas de capas precedentes. También existen redes en las que las neuronas de una capa no están conectadas con todas las neuronas de la capa siguiente. La red neuronal de la figura Figura 2.2 es un perceptrón de dos capas.

2.4 El aprendizaje de las redes neuronales.

Las redes neuronales son especialmente hábiles en representar o modelar sistemas, problemas, etc. (en general, entornos) a partir de los datos que generan éstos. Teniendo en cuenta que, según la Teoría del Aprendizaje Automático, el conocimiento es sinónimo de representación, entonces las redes neuronales se pueden considerar sistemas capaces de extraer conocimiento de su entorno. El aprendizaje de la red es el proceso mediante el cual adquiere dicho conocimiento. Existen dos grandes categorías de aprendizaje: supervisado y no supervisado. El primero usa parejas de datos como patrones de aprendizaje: a cada dato de entrada le acompaña su correspondiente dato de salida, y el objetivo del aprendizaje es obtener un mapeo entre los dominios de entrada y salida; en el segundo caso, únicamente hay patrones de entrada (existe un tercer tipo de aprendizaje, híbrido de los dos anteriores, que emplea tanto información supervisada como no-supervisada, y se denomina aprendizaje semi-supervisado). El aprendizaje supervisado se subdivide, a su vez, en dos tipos: clasificación, si el recorrido del mapeo es finito (a los elementos del recorrido se les llama etiquetas de clase), y regresión, si el recorrido es infinito. Las redes neuronales más usuales (los perceptrones entre ellos) emplean el aprendizaje supervisado: a partir de parejas de datos, y mediante un proceso iterativo de ajuste de los pesos, se minimiza una función de error que mide la diferencia de valor entre los patrones de salida y las correspondientes salidas calculadas por la red utilizando los patrones de entrada. Una vez finalizado el aprendizaje, la red guarda su conocimiento del entorno codificado en el valor de los pesos.

La función de error se ha de escoger según el tipo de aplicación que se vaya a dar a la red. Algunas de las funciones de error más comunes son [30]:

- La suma de los cuadrados de los errores (Sum of the Square Errors, SSE), definida por:

$$\sum_{p=1}^{r} \left\| \overrightarrow{b_p} - \widehat{\overrightarrow{b_p}} \right\|^2$$

  donde *r* es el número de patrones de validación, $\overrightarrow{b_p}$ es el vector salida *p*-ésimo calculado por la red a partir del vector de entrada $\overrightarrow{a_p}$, y $\widehat{\overrightarrow{b_p}}$ es el patrón salida *p*-ésimo.

- La raíz cuadrada de la suma de los cuadrados de los errores (Root Square Error, RSE):

$$\sqrt{\sum_{p=1}^{r} \left\| \overrightarrow{b_p} - \widehat{\overrightarrow{b_p}} \right\|^2}$$

- La raíz cuadrada del error cuadrático medio (Root Mean Square Error, RMSE):

$$\sqrt{\frac{\sum_{p=1}^{r} \left\| \overrightarrow{b_p} - \widehat{\overrightarrow{b_p}} \right\|^2}{r}}$$

- El error máximo:

$$\max_{p=1,\ldots,r} \left\| \overrightarrow{b_p} - \widehat{\overrightarrow{b_p}} \right\|$$

Las redes neuronales también se pueden entrenar mediante aprendizaje no supervisado, del cual existen varios tipos (aprendizaje hebbiano, competitivo, feature mapping, etc). Por citar un caso, los mapas auto-organizados de Kohonen usan aprendizaje competitivo. Otra posibilidad es entrenar a las redes neuronales mediante combinaciones mixtas de aprendizaje supervisado-no supervisado, como es el aprendizaje híbrido (la red usa los dos tipos de aprendizaje, pero en capas distintas) y el aprendizaje reforzado (basado en el concepto de condicionamiento por refuerzo: se refuerzan los pesos que aciertan las salidas, y se penalizan los que fallan; el objetivo es minimizar una función de error global que mide el comportamiento de la red). Por ejemplo, las redes de base radial usan aprendizaje híbrido.

Una vez que la red ha sido entrenada es necesario comprobar la bondad de su ajuste con el fenómeno a modelar. Esta comprobación se denomina test de generalización, y se lleva a cabo con la red en fase de recuerdo o ejecución (**Nota**: las redes neuronales tienen dos modos de operación: modo aprendizaje o entrenamiento, y modo recuerdo o ejecución. El primero es el que hemos descrito: la red usa el conjunto de patrones de aprendizaje para evolucionar, ajustando sus pesos y/o estructura; en el modo recuerdo, la red, que ahora tiene fijada su topología y los pesos, calcula las salidas a partir de los datos de entrada). Con el fin de poder llevar a cabo el test de generalización, lo usual es

que el conjunto de patrones disponibles para el aprendizaje de la red se divida en dos grupos: patrones de entrenamiento y patrones de prueba o generalización. El conjunto de patrones de entrenamiento suele dividirse, a su vez, en conjunto de entrenamiento propiamente dicho y conjunto de validación, usado para ajustar el modelo. Esta forma de proceder se denomina validación cruzada (cross-validation).

Es muy importante que el conjunto de prueba no contenga datos usados en la fase de entrenamiento (error conocido como *comprobación sobre el conjunto de entrenamiento*). Suele ser típico utilizar el 80% de los datos para entrenamiento y del 20% restante, usar una mitad para validar y el resto como test de generalización (es un criterio orientativo).

Un problema a evitar cuando se entrena la red, y que se hace patente durante la validación, es el fenómeno del sobreajuste (overfitting): debido a un aprendizaje excesivamente adaptado a los patrones de entrenamiento, el ruido inherente a éstos es incorporado al modelo aprendido, que pierde así generalidad; en consecuencia, la red produce pobres resultados cuando se le suministran datos distintos a los usados para entrenarla. Es decir, una red sobreajustada modela los datos de aprendizaje, pero no el fenómeno que los genera. En las Figuras 2.4. a), b), c) y d) se ilustra el problema del sobreajuste. En la figura a) se muestra un ajuste “perfecto” a los datos de entrenamiento (puntos negros); sin embargo, en b) se hace patente que tal entrenamiento resultó en un sobreajuste, pues la red no se comporta muy bien con los datos de validación (puntos blancos); en c) se muestra el entrenamiento sin sobreajuste; en d) se comparan ambos entrenamientos.

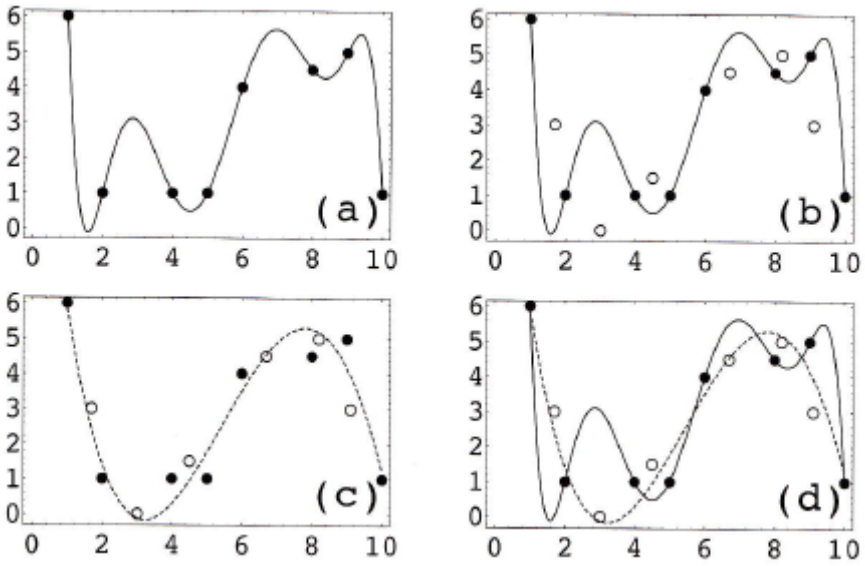


Figura 2.3. Ilustración del problema del sobreajuste

Generalmente, el sobreajuste está asociado a un exceso de parámetros (pesos, bias) de la red. Esto hace que sean preferibles las redes pequeñas a las grandes, supuesto que ambas sean capaces de modelar el fenómeno (Principio de parsimonia, también llamado Principio de la navaja de Occam: las soluciones más sencillas tienen el mejor desempeño). Además, si una red es pequeña es más fácil de entrenar. Es necesario

resaltar que algunos autores discrepan de la utilidad de este principio en el diseño de las redes neuronales [133][52]. De todas formas, existen varios procedimientos para conseguir redes con el tamaño óptimo: técnicas de poda (se parte de una red neuronal de tamaño considerable y se eliminan neuronas, conexiones o capas hasta lograr una red de menor tamaño y desempeño satisfactorio), técnicas de crecimiento (justo lo contrario: se parte de una red muy pequeña y se van añadiendo neuronas, conexiones y capas hasta lograr el tamaño óptimo), etc. El entrenamiento bayesiano normalmente minimiza el problema de sobreajuste [89].

Lo contrario del sobreajuste es el problema de bajoajuste, que consiste en que la red construye un modelo erróneo a partir de los patrones de entrenamiento por alguna de estas razones:

1. El número de neuronas, conexiones o capas es inferior al que se necesita para modelar el fenómeno del que proceden los datos.

2. El número de iteraciones empleadas en el entrenamiento es inferior al requerido para que el error sea suficientemente pequeño.

El primer punto pone de manifiesto la necesidad que tienen las redes neuronales de un cierto grado de "complejidad" para que su aprendizaje resulte satisfactorio. En realidad, cualquier sistema de aprendizaje precisa de un equilibrio óptimo entre complejidad, necesaria para que el sistema disponga de una adecuada capacidad de adaptación a los datos (de lo contrario, daría lugar a bajoajuste), y simplicidad (si no, habría sobreajuste). Se requiere, entonces, disponer de un criterio práctico de complejidad; en teoría de aprendizaje automático se utiliza con este fin la dimensión de Vapnik-Chervonenkis (VC), que mide la capacidad de una determinada familia de funciones $f\left(x,\Theta\right)$, donde f representa a la familia y Θ el conjunto de parámetros, de realizar particiones binarias (esto es, clasificaciones binarias) en el espacio de las variables de entrada. Se define del siguiente modo: dada la familia $f\left(x,\Theta\right)$, la dimensión VC es el cardinal del mayor conjunto de datos para el cual toda partición binaria (dicotomía) es representable por alguna de las funciones de $f\left(x,\Theta\right)$. Un ejemplo sencillo en $\mathbb{R}^2$ [48]: supongamos la familia $f\left(\vec{x},\Theta\right)=w_0+w_1x_1+w_2x_2$, con $\Theta=\left(w_0,w_1,w_2\right)$ (es decir, el conjunto de rectas de $\mathbb{R}^2$); como se advierte en la Fig. 2.5., para un conjunto de datos de cardinal *n=3* es posible representar todas las dicotomías mediante rectas:

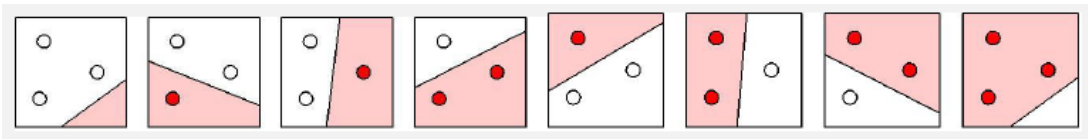


Figura 2.4. Dicotomías separables mediante rectas

Sin embargo, si *n=4* la siguiente dicotomía no se puede representar por una recta:

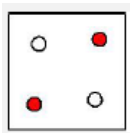


Figura 2.5. Una dicotomía no separable mediante rectas

Por lo tanto, la dimensión VC de $f\left(\vec{x},\Theta\right)=w_0+w_1x_1+w_2x_2$ en $\mathbb{R}^2$ es 3. (**Nota**: la definición de dimensión VC se fundamenta en clasificaciones binarias, pero existen extensiones de este concepto para clasificaciones *n*-arias [21] [123]). Observemos que cuanto mayor es la dimensión VC de una familia de funciones, mayor es su capacidad de representar clasificaciones.

Una red neuronal puede interpretarse como una familia de funciones, cada una de las cuales realiza un mapeo entre el espacio de las variables de entrada y el de salidas (los pesos de la red constituyen los parámetros de dicha familia). Por lo tanto, la dimensión VC es aplicable a las redes neuronales. Macintyre y Sontag demostraron que las redes neuronales feedforward finitas con funciones de activación sigmoidales tenían una dimensión VC finita [88]. La capacidad clasificatoria de la red crece al aumentar el número de pesos (se suele decir que la red tiene mayor "expresividad"). Se demuestra que la dimensión VC de una red neuronal feedforward con funciones de activación sigmoidales y *W* pesos tiene un orden de crecimiento *O* limitado superiormente por $O\left(W^4\right)$ [71]. Se puede encontrar información sobre estos temas en [87].

La combinación óptima entre el ajuste de la red a los datos de entrenamiento y su poder de generalización se mide mediante el riesgo esperado, que se define de la siguiente forma [108]: supongamos que la red entrenada $f\left(\vec{x},\zeta\right)$ representa un mapeo entre $X\rightarrow Z$, el riesgo esperado es

$$R(\zeta)=\int_{(X,Z)}\frac{1}{2}\left\|\vec{z}-f\left(\vec{x},\zeta\right)\right\|dP\left(\vec{x},\vec{z}\right)$$

con *P* la función de probabilidad conjunta entre $\vec{x}$ y z.

Si existe una función de densidad conjunta entre $\vec{x}$ y z, la expresión anterior queda:

$$R(\zeta)=\int_{(X,Z)}\frac{1}{2}\left\|\vec{z}-f\left(\vec{x},\zeta\right)\right\|p\left(\vec{x},\vec{z}\right)d\vec{x},d\vec{z}$$

El riesgo esperado mide la bondad del entrenamiento de la red: cuanto más pequeño sea *R* mejor entrenada estará la red. Generalmente $p(\vec{x},\vec{z})$ no es conocida, y no podemos calcular $R(\zeta)$. Sin embargo, existe la siguiente cota que se cumple con probabilidad $1-\rho$ $(0 \le \rho \le 1)$:

$$R(\zeta) \le R_{emp}(\zeta) + \sqrt{\frac{h \cdot \left(\log\left(\frac{2N}{h} \right) + 1 \right) - \log\left(\frac{\rho}{4} \right)}{N}}$$

donde

- $R_{emp}(\zeta)$ se denomina riesgo empírico: $R_{emp}(\zeta) = \frac{1}{2N} \sum_{i=1}^{N} \left| z_i - f\left(\vec{x}_i, \zeta\right) \right|$, con *N* número de patrones de entrenamiento.
- *h* es la dimensión VC (se cumple $N > h$)
- A la parte derecha de la desigualdad se le llama cota del riesgo, y al segundo término de la cota del riesgo se le llama confianza de VC. La cota del riesgo es independiente de $p(\vec{x},\vec{z})$.

Lo que interesa al entrenar una red neuronal (en general, una máquina de aprendizaje) es obtener una cota de riesgo con el menor valor posible; el problema es que la cota de riesgo depende de *h* y no es posible calcularla en la mayoría de las ocasiones (excepto en el caso de los modelos lineales, en los que se basa la teoría de las Máquinas de Soporte Vectorial).

Por lo tanto, lo único posible es hacer análisis empíricos con las redes neuronales, e intentar extraer algunas conclusiones prácticas. Por ejemplo, MacKay [89] estudia las funciones representadas por perceptrones aleatorios de dos capas (la capa 1 oculta y la capa 2 de salida) con una neurona de entrada y otra de salida. En la figura siguiente vemos un esquema de la red neuronal:

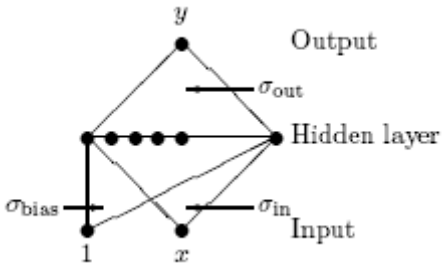


Figura 2.6. Un perceptrón multicapa que consta de una sola neurona de entrada y una sola neurona de salida

Los pesos del perceptrón $w_{lj}^{(1)}$ $l \in \{1\}$ $j \in \{1,...,H\}$, con *H*=número de neuronas de la capa oculta, y $w_{ji}^{(2)}$ $i \in \{1\}$, y los bias $\theta_j^{(1)}$ y $\theta_i^{(2)}$ constituyen variables aleatorias

gaussianas de media 0; $\theta_j^{(1)}$ tiene desviación típica σ_{bias}, $w_{jl}^{(1)}$ tiene desviación típica σ_{in}, y $\theta_i^{(2)}$ y $w_{ij}^{(2)}$ tiene σ_{out} [4]

Claramente, la forma de estas funciones depende de los valores de *H*, σ_{bias}, σ_{in} y σ_{out}. Por ejemplo, en la figura Figura. 2.7. se muestran varias funciones obtenidas usando los valores *H*=400 , $\sigma_{bias} = 8, 6, 4, 3, 2, 1.6, 1.2, 0.8, 0.4, 0.3, 0.2$, $\sigma_{in} = 5\sigma_{bias}$ y $\sigma_{out} = 0.05$ (las abscisas representan los valores de entrada a la red, y las ordenadas los de salida):

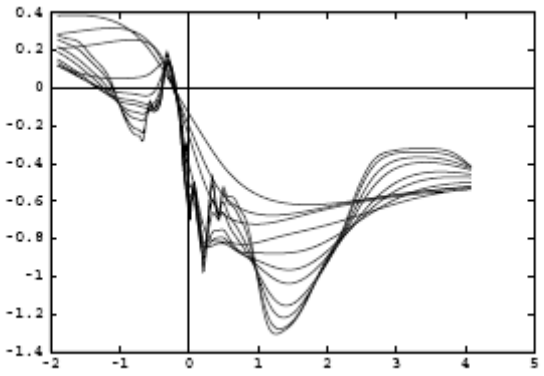


Figura 2.7. Funciones representadas por la red con *H*=400 y parámetros $\sigma_{bias} = 8, 6, 4, 3, 2, 1.6, 1.2, 0.8, 0.4, 0.3, 0.2$, $\sigma_{in} = 5\sigma_{bias}$ y $\sigma_{out} = 0.05$

En general, estas funciones tienen una forma típica:

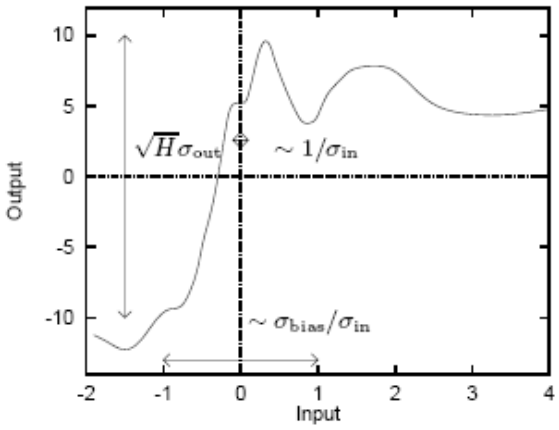


Figura 2.8. Propiedades de una función representada por una red neuronal aleatoria.

[4] A lo largo de este trabajo, un peso $w_{ij}^{(1)}$ denota el peso de la sinapsis que une la neurona l con la neurona j de la capa siguiente. Otros autores usan otras notaciones, por ejemplo Mackay [89].

Se observa:

a) La escala vertical de las funciones, es decir, la diferencia entre sus valores mayor y menor, es $\sqrt{H}\sigma_{out}$.

b) La zona de mayor variabilidad de las funciones tiene una proyección horizontal cuyo ancho es del orden de $\sigma_{bias}/\sigma_{in}$, y la región horizontal con menor variabilidad tiene un ancho del orden de $1/\sigma_{in}$.

c) Al aumentar σ_{bias} , σ_{in} y σ_{out} las funciones se vuelven más complejas y sensibles a los datos de entrada.

Neal [99] muestra que la estructura de estas funciones no depende de *H* si éste este es suficientemente grande; es decir, la complejidad de las funciones no depende del número de parámetros, sino de la magnitud de éstos. Generalmente, al trabajar con datos reales el número de parámetros es grande; resulta razonable, por lo tanto, controlar la magnitud de éstos para limitar la complejidad de la función representada por la red neuronal e independizarla del ruido inherente a los datos (aumentar el poder de generalización de la red). Una forma de lograrlo es emplear un término de regularización en la función de error.

En la literatura sobre redes neuronales a veces se considera que el tamaño óptimo de la capa oculta es $H = 2k + 1$, siendo *k* el número de neuronas de la capa de entrada, basándose en un artículo de Hecht-Nielsen [62]; sin embargo, como se señala en [125], las funciones de activación que emplea este autor en dicho trabajo son mucho más complejas que las sigmoidales utilizadas normalmente, de ahí que, en la práctica sea frecuente que el número de neuronas ocultas necesarias para entrenar correctamente la red no coincida con dicho límite teórico. Así pues, por esta razón, lo más conveniente al diseñar un perceptrón suele ser prescindir de cuestiones teóricas y tomar como guía algunas consideraciones justificadas por la expericiencia; por ejemplo [91]:

- Si *H* coincide con el número de patrones, la red tiende naturalmente a hacer que cada neurona de la capa intermedia se especialice en un patrón, en vez de generalizar. En consecuencia, se debe evitar que *H* coincida con el número de patrones.

- Algunos autores afirman que *H* debe ser como mínimo el 75% del número de neuronas de la capa de entrada.

- En la práctica se ha encontrado que la arquitectura de red más eficaz en la mayor parte de los problemas es la que tiene un número de neuronas ocultas *H* que cumple:

$$\frac{P}{2N} < H < \frac{2P}{N}$$

siendo

P el número de patrones de entrenamiento.

N el número de patrones de la capa de entrada.

2.5 El algoritmo de retropropagación.

El proceso de entrenamiento del perceptrón consiste en obtener el vector de pesos $\vec{w}^*$ que minimiza la función de error $E(\vec{w})$. La forma más usual de entrenar el perceptrón multicapa es mediante el algoritmo de retropropagación (backpropagation), que es un método iterativo basado, a su vez, en el método del descenso del gradiente (gradient descent). En esencia, el algoritmo consiste en un desplazamiento iterativo por el espacio de configuraciones de los pesos, en la dirección del gradiente y en el sentido de su disminución [30].

El algoritmo de retropropagación se puede aplicar a cualesquiera funciones de error, siempre y cuando sean derivables. Casi todas la funciones de error de interés práctico tienen una expresión *E* de la forma

$$E = \sum_{p=1}^{P} E^p(\vec{w})$$

siendo P el número de patrones y $E^p(\vec{w})$ una función de error que mide el comportamiento de la red para el *p*-ésimo patrón (en [23] se puede encontrar una justificación de la forma de *E*). Lo más conveniente es elegir las funciones E^p según el tipo de problema que se desea resolver con la red neuronal. Por ejemplo, la función cuadrática es la función de error más apropiada si la red se emplea para hacer regresiones, y la función de entropía cruzada cuando se usa para clasificar (al final de este capítulo se demuestran estas afirmaciones).

Existen dos modalidades del algoritmo de retropropagación: on-line y batch. En ambos casos, cada peso w_{jk} (peso de la conexión entre la neurona k y la neurona j de la capa anterior) ve modificado su valor proporcionalmente a la menos derivada de un error *Err*:

$$\Delta w_{jk} = -\gamma \frac{\partial Err}{\partial w_{jk}}$$

La diferencia está en que en la versión on-line: $Err = E^p$, y en la versión batch: $Err = E$. Por lo tanto:

$$\Delta w_{jk} = -\gamma \frac{\partial E^p}{\partial w_{jk}} \qquad \text{versión on-line}$$

$$\Delta w_{jk} = -\gamma \frac{\partial E}{\partial w_{jk}} = -\gamma \frac{\partial \sum_{p=1}^{P} E^p}{\partial w_{jk}} \qquad \text{versión batch}$$

En cada iteración del algoritmo de retropropación, a la red neuronal se le suministra un patrón *p* y se calcula el error E^p. Así pues, en la versión on-line del algoritmo, los pesos se actualizan en cada iteración; sin embargo, en la versión batch, es necesario que transcurran *P* iteraciones, lo que se denomina una época, para que la red calcule $E = \sum E^p$ y, por tanto, los pesos se puedan actualizar. La versión on-line requiere que los patrones de entrenamiento se suministren a la red de forma aleatoria.

A continuación se detalla la versión on-line del algoritmo de retropropagación. Cada iteración consta de las siguientes etapas (suponemos una red de dos capas):

1. El patrón de entrada $\vec{x}_p$ se mete en la red y se calcula el valor de salida $\vec{y}_p$. A este proceso de cálculo se le denomina propagación hacia delante.

2. Los pesos de la red se actualizan (los bias se consideran pesos). Antes de explicar cómo se lleva a cabo la modificación de los pesos, recordemos que, para un patrón de entrenamiento *p*, la salida de la neurona *k* de una capa se calcula mediante la expresión:

$$y_k^p = F\left(s_k^p\right) \quad (1)$$

con

$$s_k^p = \sum_j w_{jk} y_j^p \quad (2)$$

siendo F la función de activación (que vamos a suponer, por simplicidad, que es la misma en todas las capas; si no es así, resulta fácil adaptar el algoritmo).

Cada peso w_{jk} se modifica sumándole una cantidad proporcional a la menos derivada del error E^p:

$$\Delta_p w_{jk} = -\gamma \frac{\partial E^p}{\partial w_{jk}}$$

siendo γ un factor de proporcionalidad, denominado tasa de aprendizaje. El subíndice *p* que se ha añadido al símbolo Δ tiene la finalidad de recordar que en la versión on-line la modificación de los pesos se calcula a partir de E^p y no de *E*. Aplicando la regla de la cadena:

$$\frac{\partial E^p}{\partial w_{jk}} = \frac{\partial E^p}{\partial s_k^p}\frac{\partial s_k^p}{\partial w_{jk}}$$

Teniendo en cuenta (2):

$$\frac{\partial s_k^p}{\partial w_{jk}} = y_j^p$$

y definiendo

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p}$$

entonces

$$\Delta_p w_{jk} = \gamma \delta_k^p y_j^p$$

El cálculo de δ_k^p requiere de nuevo el uso de la regla de la cadena:

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p} = -\frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial s_k^p}$$

y por (1):

$$\frac{\partial y_k^p}{\partial s_k^p} = F'\left(s_k^p\right)$$

luego

$$\Delta_p w_{jk} = -\gamma F'\left(s_k^p\right) y_j^p \frac{\partial E^p}{\partial y_k^p}$$

En el cálculo de $\frac{\partial E^p}{\partial y_k^p}$ distinguimos dos casos:

a) La neurona pertenece a la capa de salida (k=o). y_o^p actúa directamente sobre E^p, en consecuencia

$$\boxed{\Delta_p w_{jo} = -\gamma F_o'\left(s_o^p\right) y_j^p \frac{\partial E^p}{\partial y_o^p}}$$

b) La neurona pertenece a la capa oculta (k=h). Podemos calcular el "efecto" de y_h^p sobre E^p a través de su efecto sobre s_o^p, y el de éste sobre E^p; es decir, aplicando la regla de la cadena:

$$\frac{\partial E^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} \frac{\partial s_o^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} \frac{\partial}{\partial y_h^p} \left(\sum_{j=1}^{N_h} w_{jo} y_j^p \right) = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} w_{ho} = -\sum_{o=1}^{N_o} \delta_o^p w_{ho}$$

Por tanto

$$\boxed{\Delta_p w_{jo} = \gamma F_o' \left(s_o^p \right) y_j^p \sum_{o=1}^{N_o} \delta_o^p w_{ho}}$$

3. Se actualizan los pesos w_{jk} a partir de $\Delta_p w_{jk}$:

$$w_{jk} = w_{jk} + \Delta_p w_{jk}$$

4. Se calcula E; el algoritmo finaliza cuando *E* es menor que cierto valor fijado.

En el caso de la versión batch del algoritmo de retropropagación, la cantidad a añadir a los pesos se calcula a partir de *E* mediante la expresión

$$\Delta w_{jk} = -\gamma \frac{\partial E}{\partial w_{jk}}$$

Desarrollando:

$$\Delta w_{jk} = -\gamma \frac{\partial E}{\partial w_{jk}} = -\gamma \frac{\partial \sum_{p=1}^{P} E^p}{\partial w_{jk}} = -\gamma \sum_{p=1}^{P} \frac{\partial \sum_{p'=1}^{P} E^{p'}}{\partial s_k^p} \frac{\partial s_k^p}{\partial w_{jk}} = -\gamma \sum_{p=1}^{P} \frac{\partial E^p}{\partial s_k^p} \frac{\partial s_k^p}{\partial w_{jk}}$$

y teniendo en cuenta las expresiones que se vieron antes:

$$\Delta w_{jk} = -\gamma \sum_{p=1}^{P} \delta_k^p y_j^p = -\gamma \sum_{p=1}^{P} F' \left(s_k^p \right) y_j^p \frac{\partial E^p}{\partial y_k^p} = \sum_{p=1}^{P} -\gamma F' \left(s_k^p \right) y_j^p \frac{\partial E^p}{\partial y_k^p} = \sum_{p=1}^{P} \Delta_p w_{jk}$$

En resumen, en la versión batch

$$\Delta w_{jk} = \sum_{p=1}^{P} \Delta_p w_{jk}$$

Por lo tanto, en el caso de la versión batch del algoritmo de retropropagación, cada actualización de los pesos consta de las siguientes etapas:

1. Se realizan *P* iteraciones (una época), en cada una de las cuales se calcula y guarda el valor $\Delta_p w_{jk}$. El cálculo de $\Delta_p w_{jk}$ se realiza del mismo modo que se describió en la versión on-line, en dos fases: 1) el patrón de entrada $\vec{x}_p$ se propaga hacia delante para obtener la salida $\vec{y}_p$ 2) se calcula $\Delta_p w_{jk}$ usando las expresiones vistas en la versión on-line.

2. Se calcula

$$\Delta w_{jk} = \sum_{p=1}^{P} \Delta_p w_{jk}$$

3. Se actualizan los pesos w_{jk} a partir de $\Delta_p w_{jk}$:

$$w_{jk} = w_{jk} + \Delta_p w_{jk}$$

4. Se calcula E; el algoritmo finaliza cuando *E* es menor que cierto valor fijado.

Existe una tercera modalidad de actualización, denominada semi-batch, en el que los pesos se actualizan tras presentar *k* patrones a la red, siendo $k<P$.

Al final de este capítulo, en las figuras Fig. 2.10. y Fig. 2.11. se muestran en pseudocódigo los esquemas de las versiones on-line y batch del algoritmo de retropropagación, respectivamente.

Es necesario apuntar tres cuestiones referentes a las versiones del método de retropropagación:

- El protocolo on-line requiere que los patrones se suministren a la red de forma aleatoria; de no ser así, en el entrenamiento habría un sesgo a favor del último patrón de entrenamiento, ya que su actualización, por realizarse siempre al final, predominaría sobre las restantes.

- De las dos versiones, solamente la versión batch tiene justificación teórica. La versión batch es la aplicación directa del método del descenso del gradiente en la minimización de la función de error *E*; por el contrario, la versión on-line es un heurístico que funciona en la práctica.
- Existe controversia sobre cuál de los dos métodos de actualización de pesos es mejor. En general, se considera que el protocolo batch es más rápido; sin

embargo, en [124] sus autores señalan que el procedimiento on-line es mucho más eficaz cuando el conjunto de entrenamiento es grande, debido al incremento de la frecuencia de actualizaciones. La razón es simple: puesto que la actualización de los pesos del protocolo batch es la suma de las actualizaciones calculadas con cada patrón, la actualización es relativamente grande, lo cual hace que el algoritmo no se adapte bien a funciones de error complejas, muy irregulares, como sucede cuando los conjuntos de entrenamiento son de tamaño considerable; es decir, el efecto es similar a usar una tasa de entrenamiento demasiado grande. Por el contrario, el entrenamiento online, al no acumular las actualizaciones, posee una mayor capacidad de adaptación a la complejidad de la función. La versión batch tiene el problema adicional de requerir más memoria cuando se implementa computacionalmente.

La función de error *E* se puede modificar añadiendo un término de regularización E_w ($E'\left(\vec{w}\right) = \beta E + \alpha E_w$), que depende de los parámetros (pesos, bias) de la red. La razón para añadir este término es penalizar en el entrenamiento los valores altos de los parámetros, lo cual, como ya se señaló anteriormente, es conveniente en las aplicaciones prácticas. Por ejemplo, se puede usar como término de regularización $E_w = 1/2 \sum_i w_i^2$.

La elección del factor de aprendizaje γ es clave a la hora de poner en práctica el entrenamiento de la red; sin embargo, establecer su valor óptimo es difícil. Si es demasiado pequeño, el algoritmo de retropropagación converge muy lentamente; por el contrario, si su valor es demasiado grande, el algoritmo oscila e incluso puede divergir. Una manera de incrementar el factor de aprendizaje sin llevar a oscilación es incluir un término momento en la expresión del incremento de los pesos [30]:

$$\Delta_p w_{jk}\left(n+1\right) = -\gamma \frac{\partial E^p}{\partial w_{jk}} + \alpha \cdot \Delta_{p'} w_{jk}\left(n\right) \qquad \text{versión on-line}$$

$$\Delta w_{jk}\left(n+1\right) = -\gamma \frac{\partial E}{\partial w_{jk}} + \alpha \cdot \Delta w_{jk}\left(n\right) \qquad \text{versión batch}$$

donde *n* indica la *n*-ésima iteración en la versión on-line, o la *n*-ésima época en la versión batch. α es el factor momento, y es un número entre 0 y 1. La inclusión del término momento hace que la actualización de los pesos no dependa sólo de las características locales de la función de error.

En los dos últimos apartados de este capítulo se van a estudiar las funciones de error que resultan más apropiadas cuando el perceptrón multicapa se entrena para hacer dos tareas estadísticas: 1) regresión y 2) clasificación.

2.6. La función de error cuando el perceptrón se entrena para regresión [23].

El análisis de regresión es un método estadístico que modela la relación de dependencia entre un vector aleatorio $\vec{X}$ (vector independiente), y un vector aleatorio $\vec{D}$ (vector dependiente), a partir de un conjunto de parejas $\left\{\vec{x}^n, \vec{d}^n\right\}$, siendo $\vec{x}^n$ y $\vec{d}^n$ realizaciones de $\vec{X}$ y $\vec{D}$, respectivamente.

Supongamos que la regresión cumple:

1. Existe una función determinista $h_k(\vec{x})$ tal que $D_k = h_k(\vec{x}) + \varepsilon_k$. ε_k es una variable aleatoria denominada error.

2. Los errores ε_k siguen una distribución normal de media cero y desviación típica σ que no depende de $\vec{x}$ ni de *k*:

$$p(\varepsilon_k) = \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left(-\frac{\varepsilon_k{}^2}{2\sigma^2}\right)$$

   Por tanto:

$$p(D_k \mid \vec{x}) = \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left(-\frac{\left(D_k - h_k(\vec{x})\right)^2}{2\sigma^2}\right)$$

   donde $p(\vec{D} \mid \vec{x})$ es la función de densidad de la variable aleatoria $\vec{D}$ condicionada por $\vec{X} = \vec{x}$ (ver Apéndice 2).

3. Las variables aleatorias ε_k $k \in \{1, ..., c\}$ son independientes.

Ahora consideremos un perceptrón que modele las funciones $h_k(\vec{x})$; esto es, un perceptrón cuya salida k-ésima sea $h_k(\vec{x})$ para una entrada $\vec{x}$. Es decir, $y_k(\vec{x}; \vec{\omega}) = h_k(\vec{x})$. Las parejas $\left\{\vec{x}^n, \vec{d}^n\right\}$ constituyen los patrones de aprendizaje de la red [5].

[5] Tanto en este apartado como en el siguiente, el número de patrón se va a indicar por *n* en vez de por *p*, como se hizo hasta ahora; la letra *p* pasa a representar probabilidad. La razón de este cambio es evitar confusiones.

Por lo tanto, se tiene:

$$p\left(D_k \mid \vec{x}\right) = \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left(-\frac{\left(y_k\left(\vec{x};\vec{\omega}\right) - D_k\right)^2}{2\sigma^2} \right)$$

donde $y_k\left(\vec{x};\vec{\omega}\right)$ es la salida *k*-ésima de la red para la entrada $\vec{x}$.

Por la independencia de las variables aleatorias $D_k \mid \vec{x}$:

$$p\left(\vec{x},\vec{D}\right) = \left(\prod_{k=1}^{c} p\left(D_k \mid \vec{x}\right)\right) \cdot p\left(\vec{x}\right) = \left(\prod_{k=1}^{c} \frac{1}{\left(2\pi\sigma^2\right)^{1/2}} \exp\left(-\frac{\left(y_k\left(\vec{x};\vec{\omega}\right) - D_k\right)^2}{2\sigma^2} \right)\right) \cdot p\left(\vec{x}\right)$$

que depende de $\vec{\omega}$. Esta expresión nos da el modelo probabilístico asociado a las variables respuesta.

Para buscar los pesos $\vec{\omega}^*$ que logran el mejor ajuste de la red neuronal a los datos, se usa el Principio de Máxima Verosimilitud aplicado a $p\left(\vec{D},\vec{x}\right)$ (ver Apéndice 2). La verosimilitud en este caso tiene la expresión:

$$L = \prod_{n=1}^{N} p\left(\vec{x}^n, \vec{d}^n\right)$$

que se calcula a partir de los patrones $\left\{\vec{x}^n, \vec{d}^n\right\}$ (pues se supone que constituyen una realización de la muestra simple $\left(\vec{X},\vec{D}\right)^n$) (ver Apéndice 2). Por tanto:

$$L = \frac{1}{\left(2\pi\sigma^2\right)^{cN/2}} \prod_{n=1}^{N}\prod_{k=1}^{c} \exp\left(-\frac{\left(y_k\left(\vec{x}^n;\vec{\omega}\right) - d_k^n\right)^2}{2\sigma^2} \right) p\left(\vec{x}^n\right)$$

$\vec{\omega}^*$ corresponde al máximo de *L*. En la práctica se calcula $E = -Ln\,\mathrm{L}$ y $\vec{\omega}^*$ corresponde al mínimo de *E*.

Desarrollando la expresión de *E*:

$$E = -Ln\ L = -Ln\left(\frac{1}{\left(2\pi\sigma^2\right)^{cN/2}} \right) - Ln\left(\prod_{n=1}^{N}\prod_{k=1}^{c} \exp\left(-\frac{\left(y_k\left(\vec{x}^n;\vec{\omega}\right) - d_k^n\right)^2}{2\sigma^2} \right)\right) - \sum_{n=1}^{N} Ln\left(p\left(\vec{x}^n\right)\right)$$

Como $\sum_{n=1}^{N} Ln\left(p\left(\vec{x}^n\right)\right)$ no depende de $\vec{\omega}$ se puede prescindir de este término para calcular $\vec{\omega}^*$. También se prescinde de la constante

$$-Ln\left(\frac{1}{\left(2\pi\sigma^2\right)^{cN/2}}\right)$$

y *E* queda:

$$E = -Ln\left(\prod_{n=1}^{N}\prod_{k=1}^{c}\exp\left(-\frac{\left(y_k\left(\vec{x}^n;\vec{\omega}\right)-d_k^n\right)^2}{2\sigma^2}\right)\right) = -\sum_{n=1}^{N} Ln\left(\prod_{k=1}^{c}\exp\left(-\frac{\left(y_k\left(\vec{x}^n;\vec{\omega}\right)-d_k^n\right)^2}{2\sigma^2}\right)\right) =$$

$$= \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left(y_k\left(\vec{x}^n;\vec{\omega}\right)-d_k^n\right)^2$$

Como σ^2 no influye en el cálculo de $\vec{\omega}^*$ se puede prescindir de ella:

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left(y_k\left(\vec{x}^n;\vec{\omega}\right)-d_k^n\right)^2$$

que es la expresión de la función de error cuadrática. Es decir, el comportamiento óptimo del perceptrón en tareas de regresión se da cuando sus pesos se obtienen minimizando la función de error cuadrática.

Una vez que se conoce $\vec{\omega}^*$ es posible estimar el valor σ por la expresión:

$$\sigma^{*2} = \frac{1}{cN}\sum_{n=1}^{N}\sum_{k=1}^{c}\left(y_k\left(\vec{x}^n;\vec{\omega}^*\right)-d_k^n\right)^2$$

Se ha justificado el empleo de la función cuadrática como función de error de la red neuronal cuando las variables respuesta son gausianas. No obstante, se demuestra que, independientemente de la función de distribución que sigan las variables respuesta, el valor de salida de una red neuronal entrenada con una función de error cuadrática es,:

$$y_k\left(\vec{x};\vec{\omega}^*\right) = \left\langle \vec{D} \mid \vec{x} \right\rangle$$

con $\langle\ \rangle$ promedio [23]. Para que esta expresión se cumpla, el número de patrones de entrenamiento debe ser suficientemente grande.

Recordemos que en el cálculo de la actualización de los pesos de las neuronas de salida por el algoritmo de retropropagación (ya sea en la versión on-line o en la batch) se usa la expresión:

$$\Delta_n w_{jk} = -\gamma F_k^{'}\left(s_k^n\right) y_j^n \frac{\partial E^n}{\partial y_k^n} \qquad k \in \{1,...,c\}$$

Generalmente, en regresión se eligen las funciones sigmoidal y lineal como funciones de activación de las neuronas ocultas y de salida, respectivamente. Por lo tanto, tenemos $F_k^{'}\left(s_k^n\right)=1$ y

$$\frac{\partial E^n}{\partial y_k^n} = y_k^n - d_k^n$$

Entonces, cuando el perceptrón se entrena para regresión, la actualización de las neuronas de salida sigue la expresión:

$$\Delta_n w_{jk} = -\gamma y_j^n \left(y_k^n - d_k^n\right)$$

Los pesos de las neuronas ocultas se modificarían retropropagando estos valores, según se vio en el apartado 2.6.

2.7. La función de error cuando la red se entrena para hacer clasificación. [23]

Supongamos una clasificación de un espacio $\vec{X}$ en *c* clases mutuamente excluyentes C_i $i \in \{1,...,c\}$, y un perceptrón con *c* salidas, entrenado para hacer dicha clasificación. Las entradas de la red van a ser los vectores $\vec{x} \in \vec{X}$. Para modelar la clasificación resulta conveniente representar la pertenencia de los vectores $\vec{x}$ a las clases mediante una codificación 1-a-c: a cada $\vec{x}$ le corresponde un vector $\vec{t} \setminus t_i \in \{0,1\}$, donde $t_i = 0$ si $\vec{x} \notin C_i$ y $t_i = 1$ si $\vec{x} \in C_i$ (Notar que $\sum_{i=1}^{c} t_i = 1$). Por ejemplo, si *c=5* el vector $\vec{x} \in C_3$ se representa por $\vec{t} = (0,0,1,0,0)$.

En el diseño de la red imponemos dos condiciones:

1. La salida y_i se interpreta como la probabilidad [6] de que $\vec{x} \in C_i$. Esta condición introduce dos restricciones a las salidas: *1)* $\sum y_i = 1$ *2)* $y_i \in (0,1)$.

[6] Una forma alternativa de entrenar a la red para tareas de clasificación es considerar que sus salidas son las etiquetas de la clasificación. En este enfoque no se hacen consideraciones estadísticas referentes a la población de los datos; simplemente se espera que, si el conjunto de patrones es suficientemente grande, la red logre el grado adecuado de generalización. Esta forma de proceder resulta útil cuando las

2. La expresión de $\Delta_n w_{jk}$ de las neuronas de salida es similar a la que se ha visto en el apartado de regresión: $\Delta_n w_{jk} = -\gamma y_j^n \left(y_k^n - d_k^n \right)$

La concurrencia de ambas condiciones desaconseja el uso de la función de error cuadrática para el entrenamiento por retropropagación del perceptrón: si se usara una función de este tipo, para que $\Delta_n w_{jk}$ tuviera la expresión exigida en 2., la función de activación de las neuronas de salida debería ser lineal, en cuyo caso no se podría asegurar que $y_i \in (0,1)$. Se debe buscar, por tanto, una combinación de función de error y función de activación de las neuronas de salida que respete las dos condiciones anteriores [7].

Considerando los vectores aleatorios $\vec{X}$ y $\vec{T}$, la clasificación se puede modelar mediante una distribución multinomial:

$$p_{\vec{T}|\vec{X}}\left(\vec{t} \mid \vec{x}\right) = \prod_{i=1}^{c} p_i^{t_i}$$

donde $\vec{p} = \left(p_1, ..., p_c\right) \setminus \sum_{i=1}^{c} p_i = 1$, y depende de $\vec{x}$. Por lo tanto, $p_{T_i|\vec{X}}\left(1|\vec{x}\right) = p_i \quad i \in \{1, ..., c\}$

La red neuronal constituye un clasificador aproximado, cuyas salidas para una entrada $\vec{x} \in \vec{X}$ toman el valor $y_i = p'_{T_i|\vec{X}}\left(1|\vec{x}\right) \quad i \in \{1, ..., c\}$. Es decir[8]

$$p'_{\vec{T}|\vec{X}}\left(\vec{t} \mid \vec{x}\right) = \prod_{i=1}^{c} y_i^{t_i}$$

Como ya se dijo antes, las salidas de la red han de cumplir:

$$\sum_{i=1}^{c} y_i = 1$$

Las funciones de activación de las neuronas de salida deben ser tales que respeten esta restricción. Normalmente se escoge la función de activación softmax:

$$y_i = \frac{\exp\left(s_i\right)}{\sum_{j=1}^{c} \exp\left(s_j\right)} = \frac{1}{1 + \exp\left(-S_i\right)}$$

---

redes neuronales de variable compleja se usan para clasificar, pues en este caso las salidas no son, en general, interpretables como probabilidades.

[7] Se puede demostrar que, si en la clasificación se usa una codificación 1-a-c, la restricción $\sum y_i = 1$ es respetada por la combinación *función de error cuadrática-función de activación lineal de las neuronas de salida* [23], a diferencia de la restricción $y_i \in (0,1)$.

[8] Es importante hacer hincapié en las salidas de la red son las probabilidades y_i y no los t_i.

con

$$S_i = s_i - \ln\left\{\sum_{j\neq i}\exp\left(s_j\right)\right\}$$

Para obtener la función de error de la red neuronal se aplica el principio de máxima verosimilitud a la siguiente función de probabilidad:

$$p''_{\vec{T}|\vec{X}}\left(\vec{t}\mid\vec{x}\right) = \frac{p'_{\vec{T}|\vec{X}}\left(\vec{t}\mid\vec{x}\right)}{p_{\vec{T}|\vec{X}}\left(\vec{t}\mid\vec{x}\right)} = \frac{\prod_{i=1}^{c} y_i^{t_i}}{\prod_{i=1}^{c} p_i^{t_i}}$$

La verosimilitud de esta función de probabilidad se obtiene a partir de una realización de una muestra *N*-dimensional, constituida por *N* parejas $\left\{\vec{x}_n, \vec{t}_n\right\}$ (la realización es el conjunto de patrones de entrenamiento de la red) [9]:

$$L = \prod_{n=1}^{N}\left(\frac{\prod_{i=1}^{c} y_{i,n}^{t_{i,n}}}{\prod_{i=1}^{c} p_{i,n}^{t_{i,n}}}\right)$$

Entonces:

$$E = -\ln L = \sum_{n=1}^{N}\ln\left(\frac{\prod_{i=1}^{c} y_{i,n}^{t_{i,n}}}{\prod_{i=1}^{c} p_{i,n}^{t_{i,n}}}\right) = -\sum_{n=1}^{N}\left(\sum_{i=1}^{c}\ln y_{i,n}^{t_{i,n}}\right) - \left(\sum_{i=1}^{c}\ln \mathrm{p}_{i,n}^{t_{i,n}}\right) =$$

$$= \sum_{n=1}^{N}\sum_{i=1}^{c} t_{i,n}\ln \mathrm{p}_{i,n} - \sum_{n=1}^{N}\sum_{i=1}^{c} t_{i,n}\ln \mathrm{y}_{i,n}$$

Puesto que los valores $p_i \quad i\in\left\{1,...,c\right\}$ son independientes de los pesos $\vec{\omega}$ de la red (además, son desconocidos) se puede prescindir de ellos, de manera que se toma como función de error:

$$E = -\sum_{n=1}^{N}\sum_{i=1}^{c} t_{i,n}\ln \mathrm{y}_i\left(\vec{x}_n;\vec{\omega}\right)$$

donde se ha explicitado que los valores *y* son función de los pesos de la red $\vec{\omega}$ y de $\vec{x}_n$.

[9] En este Apartado el índice de patrón *n* se expresa como subíndice por simplicidad.

A esta función se le suele denominar entropía cruzada (ver Apéndice 3), lo cual es incorrecto: es cierto que presenta la forma de un sumatorio de términos con el aspecto de entropías cruzadas (Apéndice 3, Fórmula (II)), pero en realidad no lo son pues $\vec{t}_n$ es la realización de un vector aleatorio y no una función de probabilidad. En [124] se discute la confusión que hay en la literatura sobre el significado de esta función.

Se puede demostrar que el mínimo de *E* corresponde a $t_{i,n} = \mathrm{y}_i\left(\vec{x}_n;\vec{\omega}\right)$ y su valor es 0.

Recordemos que para calcular la actualización de los pesos de las neuronas de salida por el algoritmo de retropropagación (ya sea en la versión on-line o en la batch) se usa la expresión:

$$\Delta_n w_{ji} = \gamma\, y_{j,n}\delta_{i,n}$$

con $\delta_{i,n} = \partial E^n / \partial s_i$.

Utilizando la regla de la cadena:

$$\delta_{i,n} = \frac{\partial E^n}{\partial s_i} = \sum_{k=1}^{c} \frac{\partial E^n}{\partial y_k}\frac{\partial y_k}{\partial s_i} \qquad i \in \{1,\ldots,c\} \qquad n \in \{1,\ldots,N\}$$

donde $E^n = -\sum_{i=1}^{c} t_{i,n} \ln \mathrm{y}_i$.

Se tiene:

$$\frac{\partial y_k}{\partial s_i} = \frac{\partial\left(\dfrac{e^{s_k}}{\sum_j e^{s_j}}\right)}{\partial s_i} = \frac{e^{s_k}\Delta_{ki}\sum_j e^{s_j} - e^{s_k}e^{s_i}}{\left(\sum_j e^{s_j}\right)^2} = \frac{e^{s_k}}{\sum_j e^{s_j}}\Delta_{kl} - \frac{e^{s_k}}{\sum_j e^{s_j}}\frac{e^{s_i}}{\sum_j e^{s_j}} = y_k\Delta_{ki} - y_k y_l$$

con Δ la delta de Kronecker. Además:

$$\frac{\partial E^n}{\partial y_k} = -\frac{t_{k,n}}{y_k}$$

Luego:

$$\frac{\partial E^n}{\partial s_i} = \sum_{k=1}^{c} -\frac{t_{k,n}}{y_k}\cdot\left(y_k\Delta_{ki} - y_k y_i\right) = \sum_{k=1}^{c} t_{k,n}y_i - t_{k,n}\delta_{ki} = -t_{i,n} + y_i\sum_{k=1}^{c} t_{k,n}$$

Y como $\sum_{k=1}^{c} t_{k,n} = 1$, entonces:

$$\delta_{i,n} = \frac{\partial E^n}{\partial a_i} = y_i\left(\vec{x}_n;\vec{\omega}\right) - t_{i,n} \qquad i \in \{1,...,c\} \qquad n \in \{1,...,N\}$$

Por lo tanto, las neuronas de salida se modifican según la expresión:

$$\Delta_n \omega_{ji} = -\eta y_{j,n}\left(y_{i,n} - t_{i,n}\right) \qquad i \in \{1,...,c\} \qquad n \in \{1,...,N\}$$

Los pesos de las neuronas ocultas se modificarían retropropagando estos valores, según se vio en el apartado 2.6.

En el caso particular de las clasificaciones binarias, con dos clases C_1 y C_2, es preferible que la red conste de una única neurona de salida (por aplicación del Principio de parsimonia). Por ejemplo, si se escoge $y = p'_{\vec{T}_1|\vec{X}}\left(1|\vec{x}\right)$, entonces

$$p'_{\vec{T}|\vec{X}}\left(\vec{t}\,|\,\vec{x}\right) = y^{t_1}\left(1-y\right)^{1-t_1}$$

y la función de error es

$$E = -\ln L = -\sum_{n=1}^{N}\left\{t_n \cdot \ln y\left(\vec{x}_n;\vec{\omega}\right) + \left(1-t_n\right)\cdot \ln\left(1-y\left(\vec{x}_n;\vec{\omega}\right)\right)\right\}$$

siendo *N* el número de patrones de entrenamiento.

Como la salida de la red representa una probabilidad, se escoge como función de activación de la neurona de salida la función sigmoidal

$$y(s) = \frac{1}{1+\exp(-s)}$$

Su derivada es (ver Apéndice 1)

$$y'(s) = y(s)\left(1-y(s)\right)$$

luego, la variación del error al variar *s* tiene la expresión

$$\delta_n = \frac{\partial E^n}{\partial s} = \frac{\partial E^n}{\partial y}\frac{\partial y}{\partial s} = y\left(\vec{x}_n;\vec{\omega}\right) - t_n \quad n \in \{1,...,N\}$$

siendo $E^n = -t_n \cdot \ln y - \left(1-t_n\right)\cdot \ln\left(1-y\right)$.

Entonces, para la neurona de salida

$$\Delta_n \omega_{j,1} = -\eta y_{j,n} \left(y_{1,n} - t_{1,n} \right) \quad n \in \{1, ..., N\}$$

Los pesos de las neuronas ocultas se modificarían retropropagando estos valores, según se vio en el Apartado 2.6.

2.8. Esquemas de las versiones del algoritmo de retropropagación (en pseudocódigo)

En los esquemas, *n* denota el número de pesos ω, $E_{\min}$ al error mínimo que se desea alcanzar, y *P* al número de patrones de entrenamiento $\left(\vec{x}_p, \vec{y}_p\right)$):

Hacer *i=1* hasta *n*
    Inicializar ω_i con valores aleatorios pequeños
Hacer hasta $E < E_{\min}$
    Elegir aleatoriamente *p*
    Hacer *i=1* hasta *n*
        Calcular $\Delta_p \omega_i$ a partir de $\partial E^p / \partial w_i$
    Hacer *i=1* hasta *n*
        $\omega_i = \omega_i + \Delta_p \omega_i$
    Calcular *E*

Figura 2.9. Actualización on-line

Hacer *i=1* hasta *n*
    Inicializar ω_i con valores aleatorios pequeños
Hacer hasta $E < E_{\min}$
    Hacer *i=1* hasta *n*
        Inicializar $\Delta\omega_i = 0$
    Hacer *p=1* hasta *P*
        Hacer *i=1* hasta *n*
            Calcular $\Delta_p \omega_i$ a partir de $\partial E^p / \partial w_i$
            $\Delta\omega_i = \Delta\omega_i + \Delta_p \omega_i$
    Hacer *i=1* hasta *n*
        $\omega_i = \omega_i + \Delta\omega_i$
    Calcular *E*

Figura 2.10. *Actualización en lotes* (batch)

3. Redes neuronales de variable compleja.

3.1. Justificación de la importancia de las redes neuronales de variable compleja.

Las redes neuronales de variable compleja (complex-valued neural network, CVNN) son, como indica su nombre, redes neuronales cuyas entradas, pesos y salidas son números complejos, y cuyas funciones de activación son funciones de variable compleja.

Aunque en un principio pudiera parecer que las CVNNs constituyen una simple extensión de las redes neuronales de variable real (Real-Valued Neural Networks, RVNN), sus particularidades convierten su estudio en todo un nuevo campo de trabajo dentro de la Teoría de Redes Neuronales, que experimenta en la actualidad un gran desarrollo [65].

Las CVNNs surgen de la necesidad de procesar con redes neuronales datos expresados en forma compleja. Obviamente, las redes neuronales de variable real pueden resolver problemas que involucren cantidades complejas, simplemente separando los datos de entrada y salida en sus partes real e imaginaria, y usando ambas como entradas de la red; sin embargo, el uso de CVNNs se justifica por las siguientes ventajas [5][65]:

1. Puesto que las redes neuronales complejas consisten básicamente en "mapeos" en el dominio complejo, pueden procesar de forma natural problemas en los que aparece información compleja, en forma de datos, variables, parámetros, etc. Estos problemas son muy frecuentes en numerosos ámbitos de la Ciencia e Ingeniería; por ejemplo, en procesamiento de la señal (imágenes, sonido, series temporales, señales eléctricas, etc), mecánica cuántica, electromagnetismo, teoría de ondas, etc.

2. Las redes neuronales de variable compleja tienen mayor capacidad funcional que sus homólogas reales, pues son capaces de representar no sólo funciones complejas, sino un mayor número de funciones reales que las redes neuronales reales.

La segunda ventaja es tan importante que es necesario detenerse en ella. Se va a ilustrar con un ejemplo: sea un perceptrón de variable real constituido por una única neurona con función de activación la función escalón o signo:

$$\operatorname{sgn}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

y cuyos valores de entrada $x_1, ..., x_n$ se expresan en el alfabeto con dos símbolos $\{-1,1\}$. Su salida viene dada por la expresión

$$\operatorname{sgn}(\omega_0 + \omega_1 x_1 + ... + \omega_n x_n)$$

donde $\omega_1,...,\omega_n \in \mathbb{R}$ son los pesos de la red y $\omega_0 \in \mathbb{R}$ es el bias. Se debe notar que el perceptrón descrito es funcionalmente idéntico al original de McCulloch-Pitts; la única diferencia se encuentra en que en este último se usaba el alfabeto $\{0,1\}$.

Matemáticamente, el perceptrón constituye una familia de funciones $f(\vec{\omega}): \mathbb{R}^N \to \{-1,1\}$ de la forma:

$$f(x_1,...,x_N) = \text{sgn}(\vec{\omega} \circ \vec{x} + \omega_0)$$

cuyos parámetros son los pesos $\vec{\omega} = (\omega_1,...,\omega_N)$ y ω_0, y donde $\vec{x} = (x_1,...,x_N)$.

Según esta expresión, los valores de $f(\vec{\omega})$ se obtienen en base a una partición del espacio de entrada en dos clases separadas por el hiperplano de ecuación $\omega_0 + \omega_1 x_1 + ... + \omega_N x_N = \vec{\omega} \circ \vec{x} + \omega_0 = 0$. Dependiendo de los valores $\omega_0, \omega_1, ..., \omega_N$, $f(\vec{\omega})$ se concreta en distintas funciones y, en consecuencia, el perceptrón se comporta como distintas funciones binarias. Por ejemplo, si *N=2* y la recta divide el plano de la siguiente forma

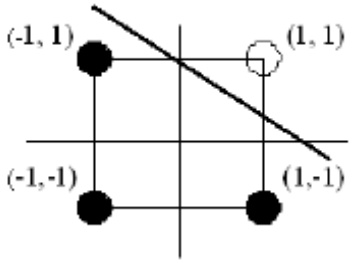


Figura 3.1. La función OR

entonces, f toma valor -1 en $(-1,-1)$, $(-1,1)$ y $(1,-1)$, y 1 en $(1,1)$, lo cual significa que el perceptrón actúa en este caso como la función OR bidimensional

x_1	x_2	x_1 OR x_2
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Tabla 3.1. Función OR bidimensional

Existen 2^{2^N} funciones booleanas de *N* variables. Si *N=2* hay 16 funciones booleanas; si *N=3* hay $2^8 = 256$; si *N=4* hay 65536. El perceptrón con una única neurona puede expresar estas funciones si y sólo si se pueden representar mediante una clasificación binaria en el espacio de las variables independientes, generada por el hiperplano $\omega_0 + \omega_1 x_1 + ... + \omega_n x_n = 0$. Si *N=2* existen 14 funciones booleanas representables por el perceptrón; si *N=3* se pueden representar 104 y 2000 si *N=4*.

Un ejemplo de función booleana no representable por el perceptrón real es la función XOR:

x_1	x_2	x_1 XOR x_2
1	1	1
1	-1	-1
-1	1	-1
-1	-1	1

Tabla 3.2. Función XOR bidimensional

puesto que no es posible encontrar una recta que separe los puntos en los que la función ha de tomar el valor 1 y en los que ha de valer -1:

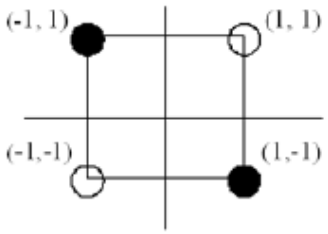


Figura 3.2. La función XOR

¿Cómo se solventa esta incapacidad del perceptrón de representar ciertas funciones booleanas? Existen tres soluciones:

a) Usar un perceptrón multicapa, con una capa oculta, como ya se explicó en el capítulo 1.

b) Dotar al espacio de entrada de una mayor dimensionalidad.

c) Usar un perceptrón formado por una única neurona compleja.

Veamos en que consiste la solución b) aplicada a la función XOR: consideremos un espacio de entrada $E_3^* = \left\{ (x_1, x_2, x_3) \in \{-1,1\}^3 \setminus (x_1, x_2) \in E_2,\ x_3 = x_1 x_2 \right\} \subset \{-1,1\}^3$. Si las entradas de la función XOR se expresan como puntos de este nuevo espacio, y se eligen

los pesos $\omega_0 = 0,\ \omega_1 = 1,\ \omega_2 = 1,\ \omega_3 = 2$, entonces obtenemos los siguientes valores de la función

x_1	x_2	$x_1 \cdot x_2$	$z = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$	*sgn(z)*	x_1 XOR x_2
1	1	1	4	1	1
1	-1	-1	-2	-1	-1
-1	1	-1	-2	-1	-1
-1	-1	1	0	1	1

Tabla 3.3. La función XOR en un espacio de entrada de mayor dimensión

Es decir, si el espacio de entrada se proyecta en un espacio de mayor dimensión, entonces el perceptrón real constituido por una única neurona puede representar la función XOR. En general, cuando el espacio de entrada se incluye en un espacio de mayor dimensión, aumenta la probabilidad de existencia de una clasificación lineal de los patrones de entrada (Teorema de Cover [78]). **Nota**: otro ejemplo de la solución b) lo vimos en el Apartado 2.1., donde se sustituyó una clasificación disjunta unidimensional, por una clasificación en un espacio bidimensional.

La solución c) muestra la potencia de las neuronas de variable compleja. Veamos cómo se puede representar la función XOR con un perceptrón formado por una sola neurona de variable compleja: se escoge la función de activación

$$\varphi(z) = \begin{cases} 1 & \text{si } 0 \le \arg(z) < \pi/2 \text{ o } \pi \le \arg(z) < 3\pi/2 \\ -1 & \text{si } \pi/2 \le \arg(z) < \pi \text{ o } 3\pi/2 \le \arg(z) < 2\pi \end{cases}$$

donde $z = |z| e^{\arg(z) i}$ con $\arg(z) \in [0, 2\pi)$; gráficamente

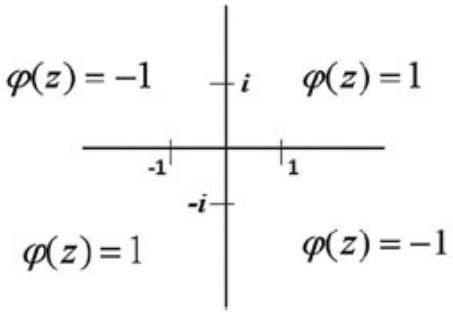


Figura 3.3. La función de activación $\varphi(z)$

Dando a los pesos los valores $\omega_0 = 0,\ \omega_1 = i,\ \omega_2 = 1$, se puede representar la función XOR:

x_1	x_2	$x_1 \cdot x_2$	$z = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$	$arg(z)$	$\varphi(z)$	x_1 XOR x_2
1	1	1	$1+i$	$\pi/4$	1	1
1	-1	-1	$1-i$	$3\pi/4$	-1	-1
-1	1	-1	$-1+i$	$5\pi/4$	-1	-1
-1	-1	1	$-1-i$	$7\pi/4$	1	1

Tabla 3.4. La función XOR representada por un perceptrón complejo

Lógicamente, la mayor capacidad funcional de las neuronas complejas da lugar a una mayor capacidad funcional de las CVNNs respecto a la de sus homólogas reales. Por esta razón, generalmente se requieren CVNNs de menor tamaño que las RVNNs para resolver los mismos problemas; además, su entrenamiento es más rápido.

[65] señala que las redes neuronales de variable compleja tienen mayor poder de generalización que las reales. La razón de ello es la propiedad de ortogonalidad inherente a las redes neuronales de variable compleja. Más adelante, en este mismo capítulo, se tratará de esta cuestión.

3.2. El algoritmo de retropropagación complejo.

Aunque conceptualmente las redes neuronales de variable compleja son similares a las de variable real, sin embargo el algoritmo de retropropagación complejo no es una simple extensión de su homólogo real, debido a la naturaleza peculiar de las funciones de variable compleja. Antes de continuar, es necesario recordar los siguientes resultados del análisis complejo [103]:

1. Sea $f : \Omega \to \mathbb{C}$ con $\Omega \subset \mathbb{C}$ abierto, siendo $f(z) = u(x,y) + iv(x,y)$ con $z = x + iy$. Se dice que f es derivable en $z \in \Omega$ si y sólo si

$$\exists \lim_{\Delta z \to 0} \frac{f(z+\Delta z) - f(z)}{\Delta z} = L$$

   para cualquier dirección de Δz. A L se le llama derivada de f en z, y se denota $f'(z)$.

2. Si $\exists f'(z)\ \ \forall z \in \Omega$, se dice que f es holomorfa en Ω. Si $\Omega = \mathbb{C}$ se dice que f es función entera. Si $\exists f'(z)$, entonces f continua en z.

3. $\exists f'$ en $D \subset \mathbb{C}$ disco abierto centrado en z, si y sólo si f analítica en z, es decir, si $f(z)$ se puede expresar mediante un desarrollo de Taylor en D.

4. f es diferenciable en z si y sólo si $\exists \omega \in \mathbb{C}$ tal que

$$\lim_{\Delta z \to 0} \frac{\left| f\left(z+\Delta z\right) - f\left(z\right) - \omega \Delta z \right|}{\left|\Delta z\right|} = 0$$

   Se cumple que $\omega = f'\left(z\right)$. Por lo tanto, en análisis complejo los conceptos de derivabilidad y diferenciabilidad son equivalentes.

5. Sea $z = x + iy$ tal que $\exists f'\left(z\right)$, entonces se cumplen las condiciones de Cauchy-Riemann:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} &= -\frac{\partial v}{\partial x} \end{aligned}\right\}$$

   La recíproca se cumple si *u,v* tienen las primeras derivadas parciales continuas.

   Las condiciones de Cauchy-Riemann se pueden expresar en una sola ecuación:

$$\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y} = 0$$

6. Teorema de Liouville:

   f es entera, acotada en $\mathbb{C}$ si y sólo si f es constante.

El algoritmo de retropropagación real requiere que sean diferenciables tanto la función de error como las funciones de activación; además, las funciones de activación deben ser acotadas y no lineales. Por lo tanto, en principio, resulta lógico exigir las mismas propiedades a las correspondientes funciones complejas. Sin embargo, se presentan dos problemas [11]:

1. El objetivo del algoritmo de retropropagación complejo es calcular el mínimo de una función error; normalmente:

$$E\left(\vec{\omega}\right) = \frac{1}{2}\sum_{n=1}^{N}\left(z\left(\vec{\omega},\vec{x}_n\right) - \vec{y}_n\right)\left(z\left(\vec{\omega},\vec{x}_n\right) - \vec{y}_n\right)^*$$

   con *N* número de patrones de entrenamiento, $\vec{x}_n$ *n*-ésimo patrón de entrenamiento de entrada, $\vec{y}_n$ *n*-ésimo patrón de entrenamiento de salida, $z\left(\vec{\omega},\vec{x}_n\right)$ salida de la red al suministrarle el patrón de entrada $\vec{x}_n$, $\vec{\omega}$ conjunto de parámetros de la red (pesos y bias). * denota el complejo conjugado.

$E(\vec{\omega})$ es una función real de variable vectorial compleja. Este tipo de funciones no cumplen las condiciones de Cauchy-Riemann, es decir, no son holomorfas [127]. Por lo tanto, el método para calcular el mínimo de $E(\vec{\omega})$ no se puede basar en algoritmos que requieran el uso de las derivadas de la función, por ejemplo, el algoritmo del descenso del gradiente (que es el fundamento del método de retropropagación de las RCNNs).

2. Las versiones complejas de las funciones de activación de las redes neuronales reales no son acotadas ni enteras, luego no se pueden usar como funciones de activación de las redes neuronales complejas. Por ejemplo, las gráficas de la parte real (a) y compleja (b) de la función $1/1+e^{-z}$ son:

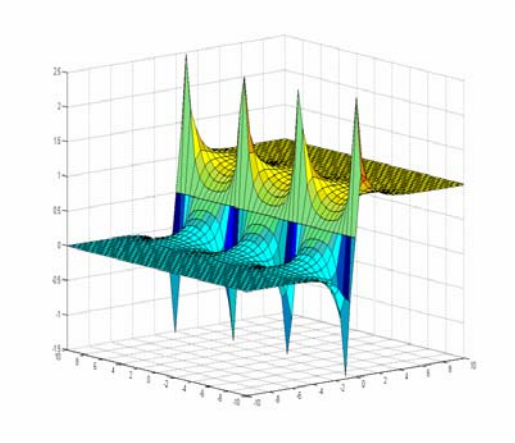


Figura 3.4. a) $\mathrm{Re}\left(1/1+e^{-z}\right)$

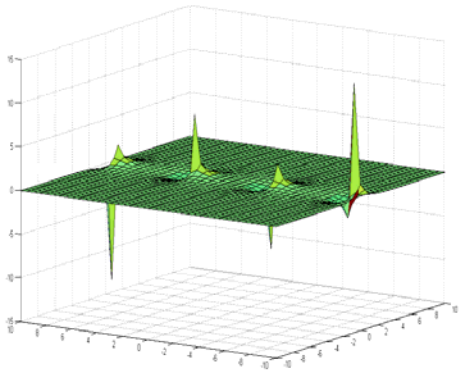


Figura 3.4. b) $\mathrm{Im}\left(1/1+e^{-z}\right)$

y de la función $tgh(z)$:

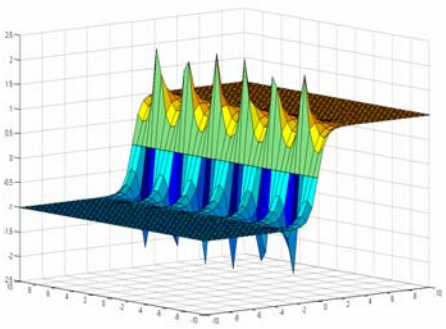


Figura 3.5. a) $\mathrm{Re}\left(tgh(z)\right)$

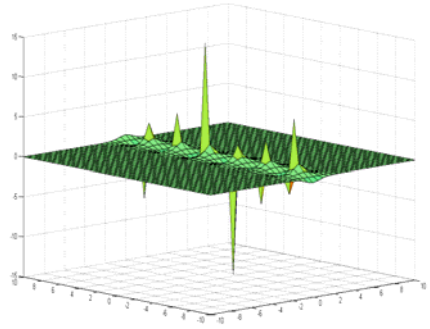


Figura 3.5. b) $\mathrm{Im}\left(tgh(z)\right)$

De hecho, según el teorema de Liouville, las únicas funciones enteras, acotadas en $\mathbb{C}$ son las funciones constantes y, obviamente este tipo de funciones no son adecuadas para entrenar una red neuronal.

Para resolver el primer problema existen dos soluciones posibles [11]:

1. Construir el algoritmo de entrenamiento en base a un método de optimización que soslaye la necesidad del uso de la derivación compleja para minimizar

$$E(\vec{\omega}) = \frac{1}{2}\sum_{n=1}^{N}\left(z(\vec{\omega},\vec{x}_n)-\vec{y}_n\right)\left(z(\vec{\omega},\vec{x}_n)-\vec{y}_n\right)^{*}.$$

2. Utilizar una ampliación del concepto de derivada compleja extensible a funciones no-holomorfas. El análisis complejo dotado de este nuevo concepto de derivada se denomina cálculo de Wirtinger [11] (ver Apéndice 5), y permite aplicar el algoritmo de retropropagación complejo de forma sencilla y rápida, sin necesidad de desarrollos engorrosos.

En cuanto al problema de las funciones de activación se han propuesto diversas soluciones:

1. Usar como función de activación una función compleja acotada en $\mathbb{C}$, pero no entera. Por ejemplo [50][64]:

$$f(z) = \frac{z}{c + \frac{|z|}{r}}$$

$$f(z) = tgh\left(\frac{s}{m}\right)e^{i\beta} \quad \text{con } z = se^{i\beta}$$

   siendo $c, r, m \in \mathbb{R}^{+}$.

   El problema es que estas funciones conservan la fase de z y, por tanto, no se pueden utilizar en todas las aplicaciones, es decir, no son “flexibles”.

2. Otra solución es usar como función de activación una función entera cuyo conjunto de no-acotación sea de medida nula (en la literatura, a este tipo de activación se le denomina fully neural activation). Es el caso de muchas funciones elementales del análisis complejo: tan(z), sen(z), tanh(z), etc [74]. La utilidad de este tipo de funciones depende de las aplicaciones prácticas. Por ejemplo, *tanh(z)* tiene sus singularidades en los puntos $\pm n\pi i/2$, lo cual hace que, en general, no se pueda utilizar en aplicaciones en el que el dominio de la activación de las neuronas sea grande. En cambio, si este dominio se encuentra en el interior del círculo unidad, esta función se puede emplear sin problemas.

3. La manera más usual de construir la función de activación consiste es usar funciones complejas con sus partes real y compleja desacopladas (en la literatura, a este tipo de activación se le denomina Split Neural Activation).

Existen dos modalidades de función de activación de tipo split [63]:

a) Función de activación de tipo real-imaginaria

$$f_C(z) = f_R(x) + if_R(y) \qquad \forall z = x + iy \in \mathbb{C}$$

siendo f_R función de activación real.

b) Función de activación de tipo amplitud-fase

$$f_C(z) = f_R(|z|)e^{i\arg(z)} \qquad \forall z = |z|e^{i\arg(z)} \in \mathbb{C}$$

Estas funciones no son enteras, pues no cumplen las condiciones de Cauchy-Riemann en todos los puntos de su dominio. Comprobémoslo con las funciones de tipo real-imaginario: llamando $u = f_R(x)$, $v = f_R(y)$ se tiene

$$u_x = f'_R(x)$$

$$v_y = f'_R(y)$$

y, en general, $u_x \neq v_y$ pues u_x, v_y únicamente dependen de *x* e *y*, respectivamente (en general, sólo coinciden en los puntos $z = x + iy$ tales que *x=y*. Por lo tanto, las funciones de activación a) no cumplen la primera condición de Cauchy-Riemann en todos los puntos de $\mathbb{C}$, luego no son enteras.

De forma similar se comprueba que las funciones de activación de tipo amplitud-fase tampoco son enteras: teniendo en cuenta que

$$f_C(z) = f_R(|z|)e^{i\arg(z)} = f_R(|z|)\frac{(x+iy)}{\sqrt{x^2+y^2}} = \frac{f_R(|z|)}{\sqrt{x^2+y^2}}x + i\frac{f_R(|z|)}{\sqrt{x^2+y^2}}y$$

y llamando

$$u = \frac{f_R(|z|)}{\sqrt{x^2+y^2}}x$$

$$v = \frac{f_R(|z|)}{\sqrt{x^2+y^2}}y$$

se ve que *u* y *v* son funcionalmente idénticas; por lo tanto u_x y v_y también lo son, pero, en general, toman valores distintos cuando $x \neq y$. Así pues, no se

cumple la primera condición de Cauchy-Riemann en todos los puntos de $\mathbb{C}$ y, por tanto, las funciones de activación b) no son enteras.

En lo que resta de capítulo se va a explicar el desarrollo de una red neuronal de variable compleja, en la que cada uno de los problemas mencionados anteriormente se resuelve de la siguiente forma: a) la activación de la red es del tipo split neural activation, b) el óptimo de la función de error se obtiene aplicando los resultados de la teoría de Complex Adaptive Pattern Classifier Model (Complex APCM), sin necesidad de utilizar derivación compleja.

La teoría de Complex APCM consiste en lo siguiente [101]: supongamos una variable aleatoria discreta compleja multidimensional $(\vec{x}, \vec{y})$ $\vec{x} \in \mathbb{C}^n$, $\vec{y} \in \mathbb{C}^m$, con una función de probabilidad $P(\vec{x}, \vec{y})$ desconocida. Se recoge una muestra formada por *N* elementos $(\vec{x}_i, \vec{y}_i)$, siendo $\vec{x}_i$ e $\vec{y}_i$ los patrones de entrada y salida, respectivamente, suministrados a una red neuronal de variable compleja $z(\vec{\omega}, \bullet): \mathbb{C}^n \to \mathbb{C}^m$. $z(\vec{\omega}, \vec{x}_i)$ constituye la estimación del patrón de salida $\vec{y}_i$ proporcionada por la red. La salida de la red neuronal depende del vector de parámetros $\vec{\omega} \in \mathbb{C}^p$ (pesos, bias), cuyo valor óptimo $\vec{\omega}^*$ corresponde al mínimo de una función de error promedio

$$R(\vec{\omega}) = \sum_i r(z(\vec{\omega}, \vec{x}_i), \vec{y}_i) \cdot P(\vec{x}_i, \vec{y}_i)$$

donde $r: \mathbb{C}^m \times \mathbb{C}^m \to \mathbb{R}^+$ es una función de error que mide la diferencia entre $z(\vec{\omega}, \vec{x}_i)$ e $\vec{y}_i$.

Si se cumplen estos supuestos, se demuestra que es posible calcular $\vec{\omega}^*$ mediante un proceso iterativo. Llamando $\vec{\omega}_{n-1}$ al valor del vector de parámetros calculado en la iteración (n-1)-ésima, se tiene

$$\vec{\omega}_n = \vec{\omega}_{n-1} + \Delta\vec{\omega}_n$$

con $\Delta\vec{\omega}_n$ calculado en el paso *n*. Si se descompone esta expresión en sus partes real e imaginaria:

$$\mathrm{Re}[\vec{\omega}_n] = \mathrm{Re}[\vec{\omega}_{n-1}] + \mathrm{Re}[\Delta\vec{\omega}_n]$$

$$\mathrm{Im}[\vec{\omega}_n] = \mathrm{Im}[\vec{\omega}_{n-1}] + \mathrm{Im}[\Delta\vec{\omega}_n]$$

Los valores $\mathrm{Re}\left[\Delta\vec{\omega}_n\right]$ y $\mathrm{Im}\left[\Delta\vec{\omega}_n\right]$ son calculados haciendo uso del siguiente teorema:

Teorema

Dados los patrones $\left\{\left(\vec{x}_i,\vec{y}_i\right),\ i\in\{1,...,P\}\right\}$ y la red neuronal $z\left(\vec{\omega},\bullet\right)$, $\vec{\omega}$ se puede aproximar cuanto se desee al óptimo $\vec{\omega}^*$ mediante entrenamiento iterativo si $\mathrm{Re}\left[\Delta\vec{\omega}_n\right]$ y $\mathrm{Im}\left[\Delta\vec{\omega}_n\right]$ cumplen

$$\mathrm{Re}\left[\Delta\vec{\omega}_n\right]=-\varepsilon A\nabla^{\mathrm{Re}}r\left(z\left(\vec{\omega}_n,\vec{x}_n\right),\vec{y}_n\right)\qquad\text{(I)}$$

$$\mathrm{Im}\left[\Delta\vec{\omega}_n\right]=-\varepsilon A\nabla^{\mathrm{Im}}r\left(z\left(\vec{\omega}_n,\vec{x}_n\right),\vec{y}_n\right)\qquad\text{(II)}$$

con $\varepsilon>0$ lo suficientemente pequeño y *A* una matriz definida positiva. ∇^{Re} y ∇^{Im} son los gradientes respecto a la parte real e imaginaria de $\vec{\omega}$, respectivamente.

La importancia del teorema radica en que, en virtud de las expresiones (I) y (II), $\vec{\omega}$ se puede optimizar sin necesidad de calcular las derivadas complejas de $R\left(\vec{\omega}\right)$ respecto a $\vec{\omega}$, que no existen pues $R:\mathbb{C}^p\to\mathbb{R}^+$ no es holomorfa.

El algoritmo iterativo de las redes neuronales de variable compleja puede ser on-line o batch, al igual que sucede con las reales. En el primer caso, $\vec{\omega}$ se actualiza cada vez que se suministra un patrón a la red, mientras que en el segundo la actualización se realiza una vez que se le han proporcionado todos los patrones.

Vamos a utilizar los resultados del teorema para actualizar los pesos de una red neuronal de variable compleja de dos capas. Se usará la siguiente nomenclatura:

f_C: función de activación. Se supone activación de tipo split: $f_C\left(z\right)=f_{\mathrm{Re}}\left(x\right)+if_{\mathrm{Im}}\left(y\right)$ con $z=x+iy$. f_{Re} y f_{Im} son dos funciones de activación reales, que normalmente coinciden.

I_l: salida de la neurona de entrada *l*

ω_{ml}: peso de la conexión entre la neurona oculta *m* y la neurona de entrada *l.*

θ_m: bias de la neurona oculta *m*.

U_m: activación de la neurona oculta *m*:

$$U_m=\sum_l\omega_{ml}I_l+\theta_m$$

H_m : salida de la neurona oculta *m*:

$$\mathrm{Re}\left[H_m\right]+i\,\mathrm{Im}\left[H_m\right]=f_C\left(U_m\right)=f_{\mathrm{Re}}\left(\mathrm{Re}\left[U_m\right]\right)+if_{\mathrm{Im}}\left(\mathrm{Im}\left[U_m\right]\right)$$

υ_{nm} : peso de la conexión entre la neurona de salida *n* y la neurona oculta *m*.

γ_n : bias de la neurona de salida *n*.

S_n : activación de la neurona de salida *n*. Se cumple:

$$S_n=\sum_m \upsilon_{nm}H_m+\gamma_n$$

O_n : salida de la neurona de salida *n*:

$$O_n=\mathrm{Re}\left[O_n\right]+i\,\mathrm{Im}\left[O_n\right]=f_C\left(S_n\right)=f_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)+if_{\mathrm{Im}}\left(\mathrm{Im}\left[S_n\right]\right)$$

T_n : componente n-ésima de un patrón de salida T correspondiente a un patrón de entrada I.

E_p : error correspondiente al patrón p-ésimo $\left(I^p,T^p\right)$:

$$E_p=\frac{1}{2}\sum_n\left|T_n^p-O_n^p\right|^2=\frac{1}{2}\sum_n\left|\delta_n^p\right|^2$$

con $\delta_n^p=T_n^p-O_n^p$

E : error total:

$$E=\sum_p E_p$$

Aplicando el teorema anterior, se obtiene la regla de actualización de los parámetros de la red en cada paso del algoritmo de entrenamiento [101]:

$$\Delta\upsilon_{nm}=-\varepsilon\frac{\partial E_{err}}{\partial\,\mathrm{Re}\left[\upsilon_{nm}\right]}-i\varepsilon\frac{\partial E_{err}}{\partial\,\mathrm{Im}\left[\upsilon_{nm}\right]}$$

$$\Delta\gamma_n=-\varepsilon\frac{\partial E_{err}}{\partial\,\mathrm{Re}\left[\gamma_n\right]}-i\varepsilon\frac{\partial E_{err}}{\partial\,\mathrm{Im}\left[\gamma_n\right]}$$

$$\Delta\omega_{ml} = -\varepsilon \frac{\partial E_{err}}{\partial \mathrm{Re}\left[\omega_{ml}\right]} - i\varepsilon \frac{\partial E_{err}}{\partial \mathrm{Im}\left[\omega_{ml}\right]}$$

$$\Delta\theta_{m} = -\varepsilon \frac{\partial E_{err}}{\partial \mathrm{Re}\left[\theta_{m}\right]} - i\varepsilon \frac{\partial E_{err}}{\partial \mathrm{Im}\left[\theta_{m}\right]}$$

donde $E_{err} = E_p$ si el entrenamiento es on-line, o $E_{err} = E$ si es batch.

A continuación se desarrollan estas expresiones. Por sencillez se va a suponer que $\varepsilon = 1$ y que el entrenamiento es on-line. Comencemos con $\Delta\gamma_n$:

$$\mathrm{Re}\left[\Delta\gamma_n\right] = -\frac{\partial E_p}{\partial \mathrm{Re}\left[\gamma_n\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left|\delta_n\right|^2\right)}{\partial \mathrm{Re}\left[\gamma_n\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{k\neq n}\left|\delta_n\right|^2 + \frac{1}{2}\left|\delta_n\right|^2\right)}{\partial \mathrm{Re}\left[\gamma_n\right]} = -\frac{\partial\left(\frac{1}{2}\left|\delta_n\right|^2\right)}{\partial \mathrm{Re}\left[\gamma_n\right]} =$$

$$= -\frac{\partial\left(\frac{1}{2}\left(\mathrm{Re}\left[\delta_n\right]^2 + \mathrm{Im}\left[\delta_n\right]^2\right)\right)}{\partial \mathrm{Re}\left[\gamma_n\right]}$$

Teniendo en cuenta que:

$$\mathrm{Re}\left[\delta_n\right] = \mathrm{Re}\left[T_n - O_n\right] = \mathrm{Re}\left[T_n\right] - \mathrm{Re}\left[O_n\right] = \mathrm{Re}\left[T_n\right] - f_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right) =$$

$$= \mathrm{Re}\left[T_n\right] - f_{\mathrm{Re}}\left(\mathrm{Re}\left[\sum_m \upsilon_{nm} H_m\right] + \mathrm{Re}\left[\gamma_n\right]\right)$$

$$\mathrm{Im}\left[\delta_n\right] = \mathrm{Im}\left[T_n\right] - f_{\mathrm{Im}}\left(\mathrm{Im}\left[\sum_m \upsilon_{nm} H_m\right] + \mathrm{Im}\left[\gamma_n\right]\right)$$

se advierte que $\mathrm{Re}\left[\delta_n\right]^2$ depende de $\mathrm{Re}\left[\gamma_n\right]$, pero $\mathrm{Im}\left[\delta_n\right]^2$ no.

Por lo tanto:

$$\mathrm{Re}\left[\Delta\gamma_n\right] = -\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\gamma_n\right]} = -\mathrm{Re}\left[\delta_n\right]\frac{\partial \mathrm{Re}\left[\delta_n\right]}{\partial \mathrm{Re}\left[\gamma_n\right]} = -\mathrm{Re}\left[\delta_n\right]\cdot -f'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right) =$$

$$= \mathrm{Re}\left[\delta_n\right] f'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)$$

Del mismo modo:

$$\mathrm{Im}\left[\Delta\gamma_n\right] = \mathrm{Im}\left[\delta_n\right] f\,'_{\mathrm{Im}}\left(\mathrm{Im}\left[S_n\right]\right)$$

Luego

$$\boxed{\Delta\gamma_n = \mathrm{Re}\left[\delta_n\right] f\,'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right) + i\,\mathrm{Im}\left[\delta_n\right] f\,'_{\mathrm{Im}}\left(\mathrm{Im}\left[S_n\right]\right)}$$

Desarrollemos $\Delta\upsilon_{nm}$. Primero se calcula $\mathrm{Re}\left[\Delta\upsilon_{nm}\right]$:

$$\mathrm{Re}\left[\Delta\upsilon_{nm}\right] = -\frac{\partial E_p}{\partial\,\mathrm{Re}\left[\upsilon_{nm}\right]} = -\frac{\partial\left((1/2)\left(\mathrm{Re}\left[\delta_n\right]^2 + \mathrm{Im}\left[\delta_n\right]^2\right)\right)}{\partial\,\mathrm{Re}\left[\upsilon_{nm}\right]} =$$

$$= -\frac{\partial\left((1/2)\,\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\delta_n\right]}\frac{\partial\,\mathrm{Re}\left[\delta_n\right]}{\partial\,\mathrm{Re}\left[S_n\right]}\frac{\partial\,\mathrm{Re}\left[S_n\right]}{\partial\,\mathrm{Re}\left[\upsilon_{nm}\right]} -$$

$$-\frac{\partial\left((1/2)\,\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Im}\left[\delta_n\right]}\frac{\partial\,\mathrm{Im}\left[\delta_n\right]}{\partial\,\mathrm{Im}\left[S_n\right]}\frac{\partial\,\mathrm{Im}\left[S_n\right]}{\partial\,\mathrm{Re}\left[\upsilon_{nm}\right]}$$

Recordando:

$$\mathrm{Re}\left[\delta_n\right] = \mathrm{Re}\left[T_n\right] - f_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)$$

$$\mathrm{Im}\left[\delta_n\right] = \mathrm{Im}\left[T_n\right] - f_{\mathrm{Im}}\left(\mathrm{Im}\left[S_n\right]\right)$$

entonces

$$\frac{\partial\,\mathrm{Re}\left[\delta_n\right]}{\partial\,\mathrm{Re}\left[S_n\right]} = -f\,'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)$$

$$\frac{\partial\,\mathrm{Im}\left[\delta_n\right]}{\partial\,\mathrm{Im}\left[S_n\right]} = -f\,'_{\mathrm{Im}}\left(\mathrm{Im}\left[S_n\right]\right)$$

Además:

$$\mathrm{Re}\left[S_n\right] = \mathrm{Re}\left[\sum_{k\neq m}\upsilon_{nk}H_k + \upsilon_{nm}H_m + \gamma_n\right] = \mathrm{Re}\left[\sum_{k\neq m}\upsilon_{nk}H_k\right] + \mathrm{Re}\left[\upsilon_{nm}H_m\right] + \mathrm{Re}\left[\gamma_n\right] =$$

$$= \mathrm{Re}\left[\sum_{k\neq m} \upsilon_{nk} H_k\right] + \left(\mathrm{Re}[\upsilon_{nm}]\mathrm{Re}[H_m] - \mathrm{Im}[\upsilon_{nm}]\mathrm{Im}[H_m]\right) + \mathrm{Re}[\gamma_n]$$

$$\mathrm{Im}[S_n] = \mathrm{Im}\left[\sum_{k\neq m} \upsilon_{nk} H_k + \upsilon_{nm} H_m + \gamma_n\right] = \mathrm{Im}\left[\sum_{k\neq m} \upsilon_{nk} H_k\right] + \mathrm{Im}[\upsilon_{nm} H_m] + \mathrm{Im}[\gamma_n] =$$

$$= \mathrm{Im}\left[\sum_{k\neq m} \upsilon_{nk} H_k\right] + \left(\mathrm{Im}[\upsilon_{nm}]\mathrm{Re}[H_m] + \mathrm{Re}[\upsilon_{nm}]\mathrm{Im}[H_m]\right) + \mathrm{Im}[\gamma_n]$$

Por lo tanto:

$$\mathrm{Re}[\Delta\upsilon_{nm}] = \mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[H_m]$$

Ahora el cálculo de $\mathrm{Im}[\Delta\upsilon_{nm}]$:

$$\mathrm{Im}[\Delta\upsilon_{nm}] = -\frac{\partial E_p}{\partial\,\mathrm{Im}[\upsilon_{nm}]} = -\frac{\partial\left((1/2)\left(\mathrm{Re}[\delta_n]^2 + \mathrm{Im}[\delta_n]^2\right)\right)}{\partial\,\mathrm{Im}[\upsilon_{nm}]} =$$

$$= -\frac{\partial\left((1/2)\,\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Re}[\delta_n]}\frac{\partial\,\mathrm{Re}[\delta_n]}{\partial\,\mathrm{Re}[S_n]}\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Im}[\upsilon_{nm}]} -$$

$$-\frac{\partial\left((1/2)\,\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Im}[\upsilon_{nm}]}$$

Usando las expresiones auxiliares de antes, resulta:

$$\mathrm{Im}[\Delta\upsilon_{nm}] = \mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\cdot -\mathrm{Im}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[H_m]$$

Por lo tanto:

$$\begin{aligned}\Delta\upsilon_{nm} = &\left(\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[H_m]\right) + \\ &+ i\left(\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\cdot -\mathrm{Im}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[H_m]\right)\end{aligned}$$

A continuación se desarrolla $\Delta\theta_m$:

$$\mathrm{Re}\left[\Delta\theta_m\right] = -\frac{\partial E_p}{\partial \mathrm{Re}\left[\theta_m\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left|\delta_n\right|^2\right)}{\partial \mathrm{Re}\left[\theta_m\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left(\mathrm{Re}\left[\delta_n\right]^2 + \mathrm{Im}\left[\delta_n\right]^2\right)\right)}{\partial \mathrm{Re}\left[\theta_m\right]} =$$

$$= -\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\theta_m\right]} - \sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\theta_m\right]} =$$

$$= -\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\delta_n\right]}\frac{\partial \mathrm{Re}\left[\delta_n\right]}{\partial \mathrm{Re}\left[S_n\right]}\frac{\partial \mathrm{Re}\left[S_n\right]}{\partial \mathrm{Re}\left[H_m\right]}\frac{\partial \mathrm{Re}\left[H_m\right]}{\partial \mathrm{Re}\left[U_m\right]}\frac{\partial \mathrm{Re}\left[U_m\right]}{\partial \mathrm{Re}\left[\theta_m\right]} -$$

$$-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial \mathrm{Im}\left[\delta_n\right]}\frac{\partial \mathrm{Im}\left[\delta_n\right]}{\partial \mathrm{Im}\left[S_n\right]}\frac{\partial \mathrm{Im}\left[S_n\right]}{\partial \mathrm{Re}\left[H_m\right]}\frac{\partial \mathrm{Re}\left[H_m\right]}{\partial \mathrm{Re}\left[U_m\right]}\frac{\partial \mathrm{Re}\left[U_m\right]}{\partial \mathrm{Re}\left[\theta_m\right]} =$$

$$= -\frac{\partial \mathrm{Re}\left[H_m\right]}{\partial \mathrm{Re}\left[U_m\right]}\frac{\partial \mathrm{Re}\left[U_m\right]}{\partial \mathrm{Re}\left[\theta_m\right]}\cdot\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\delta_n\right]}\frac{\partial \mathrm{Re}\left[\delta_n\right]}{\partial \mathrm{Re}\left[S_n\right]}\frac{\partial \mathrm{Re}\left[S_n\right]}{\partial \mathrm{Re}\left[H_m\right]} +$$

$$+\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial \mathrm{Im}\left[\delta_n\right]}\frac{\partial \mathrm{Im}\left[\delta_n\right]}{\partial \mathrm{Im}\left[S_n\right]}\frac{\partial \mathrm{Im}\left[S_n\right]}{\partial \mathrm{Re}\left[H_m\right]}$$

$$\mathrm{Im}\left[\Delta\theta_m\right] = -\frac{\partial E_p}{\partial \mathrm{Im}\left[\theta_m\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left|\delta_n\right|^2\right)}{\partial \mathrm{Im}\left[\theta_m\right]} = -\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left(\mathrm{Re}\left[\delta_n\right]^2 + \mathrm{Im}\left[\delta_n\right]^2\right)\right)}{\partial \mathrm{Im}\left[\theta_m\right]} =$$

$$= -\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Im}\left[\theta_m\right]} - \sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial \mathrm{Im}\left[\theta_m\right]} =$$

$$= -\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial \mathrm{Re}\left[\delta_n\right]}\frac{\partial \mathrm{Re}\left[\delta_n\right]}{\partial \mathrm{Re}\left[S_n\right]}\frac{\partial \mathrm{Re}\left[S_n\right]}{\partial \mathrm{Im}\left[H_m\right]}\frac{\partial \mathrm{Im}\left[H_m\right]}{\partial \mathrm{Im}\left[U_m\right]}\frac{\partial \mathrm{Im}\left[U_m\right]}{\partial \mathrm{Im}\left[\theta_m\right]} -$$

$$-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Im}[H_m]}\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\theta_m]}=$$

$$=-\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\theta_m]}\cdot\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Re}[\delta_n]}\frac{\partial\,\mathrm{Re}[\delta_n]}{\partial\,\mathrm{Re}[S_n]}\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Re}[H_m]}+$$

$$+\frac{\partial\left(\frac{1}{2}\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Re}[H_m]}$$

Entonces:

$$\mathrm{Re}[\Delta\theta_m]=f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\cdot 1\cdot\sum_{n=1}^{N}\mathrm{Re}[\delta_n]f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]+\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]$$

$$\mathrm{Im}[\Delta\theta_m]=f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\cdot 1\cdot\sum_{n=1}^{N}-\mathrm{Re}[\delta_n]f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]$$

Es decir:

$$\begin{aligned}\Delta\theta_m=&\left(f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\sum_{n=1}^{N}\mathrm{Re}[\delta_n]f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]+\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]\right)+\\&+i\left(f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\sum_{n=1}^{N}-\mathrm{Re}[\delta_n]f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]\right)\end{aligned}$$

A continuación se desarrolla $\Delta\omega_{ml}$, pero antes de ello es necesario presentar unas expresiones auxiliares:

a)

$$\frac{\partial\,\mathrm{Re}[H_m]}{\partial\,\mathrm{Re}[U_m]}=f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)$$

$$\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}=f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)$$

b)

$$\omega_{ml}I_l=\left(\mathrm{Re}[\omega_{ml}]+i\,\mathrm{Im}[\omega_{ml}]\right)\left(\mathrm{Re}[I_l]+i\,\mathrm{Im}[I_l]\right)=$$

$$= \left(\mathrm{Re}\left[\omega_{ml}\right]\mathrm{Re}\left[I_l\right] - \mathrm{Im}\left[\omega_{ml}\right]\mathrm{Im}\left[I_l\right]\right) + i\left(\mathrm{Im}\left[\omega_{ml}\right]\mathrm{Re}\left[I_l\right] + \mathrm{Im}\left[I_l\right]\mathrm{Re}\left[\omega_{ml}\right]\right)$$

Por tanto:

$$\mathrm{Re}\left[\omega_{ml} I_l\right] = \mathrm{Re}\left[\omega_{ml}\right]\mathrm{Re}\left[I_l\right] - \mathrm{Im}\left[\omega_{ml}\right]\mathrm{Im}\left[I_l\right]$$

$$\mathrm{Im}\left[\omega_{ml} I_l\right] = \mathrm{Im}\left[\omega_{ml}\right]\mathrm{Re}\left[I_l\right] + \mathrm{Im}\left[I_l\right]\mathrm{Re}\left[\omega_{ml}\right]$$

c)

$$\mathrm{Re}\left[U_m\right] = \mathrm{Re}\left[\sum_l \omega_{ml} I_l + \theta_m\right] = \mathrm{Re}\left[\sum_{k \neq l} \omega_{mk} I_k\right] + \mathrm{Re}\left[\omega_{ml} I_l\right] + \mathrm{Re}\left[\theta_m\right]$$

$$\mathrm{Im}\left[U_m\right] = \mathrm{Im}\left[\sum_l \omega_{ml} I_l + \theta_m\right] = \mathrm{Im}\left[\sum_{k \neq l} \omega_{mk} I_k\right] + \mathrm{Im}\left[\omega_{ml} I_l\right] + \mathrm{Im}\left[\theta_m\right]$$

y aplicando b)

$$\mathrm{Re}\left[U_m\right] = \mathrm{Re}\left[\sum_{k \neq l} \omega_{mk} I_k\right] + \left(\mathrm{Re}\left[\omega_{ml}\right]\mathrm{Re}\left[I_l\right] - \mathrm{Im}\left[\omega_{ml}\right]\mathrm{Im}\left[I_l\right]\right) + \mathrm{Re}\left[\theta_m\right]$$

$$\mathrm{Im}\left[U_m\right] = \mathrm{Im}\left[\sum_{k \neq l} \omega_{mk} I_k\right] + \left(\mathrm{Re}\left[I_l\right]\mathrm{Im}\left[\omega_{ml}\right] + \mathrm{Im}\left[I_l\right]\mathrm{Re}\left[\omega_{ml}\right]\right) + \mathrm{Im}\left[\theta_m\right]$$

d)

$$\frac{\partial \mathrm{Re}\left[U_m\right]}{\partial \mathrm{Re}\left[\omega_{ml}\right]} = \mathrm{Re}\left[I_l\right]$$

$$\frac{\partial \mathrm{Im}\left[U_m\right]}{\partial \mathrm{Re}\left[\omega_{ml}\right]} = \mathrm{Im}\left[I_l\right]$$

$$\frac{\partial \mathrm{Re}\left[U_m\right]}{\partial \mathrm{Im}\left[\omega_{ml}\right]} = -\mathrm{Im}\left[I_l\right]$$

$$\frac{\partial \mathrm{Im}\left[U_m\right]}{\partial \mathrm{Im}\left[\omega_{ml}\right]} = \mathrm{Re}\left[I_l\right]$$

Desarrollamos $\Delta\omega_{ml}$:

$$\mathrm{Re}\left[\Delta\omega_{ml}\right]=-\frac{\partial E_p}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}=-\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left|\delta_n\right|^2\right)}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}=-\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left(\mathrm{Re}\left[\delta_n\right]^2+\mathrm{Im}\left[\delta_n\right]^2\right)\right)}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}=$$

$$=-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}=$$

$$=-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\delta_n\right]}\frac{\partial\,\mathrm{Re}\left[\delta_n\right]}{\partial\,\mathrm{Re}\left[S_n\right]}\left(\frac{\partial\,\mathrm{Re}\left[S_n\right]}{\partial\,\mathrm{Re}\left[H_m\right]}\frac{\partial\,\mathrm{Re}\left[H_m\right]}{\partial\,\mathrm{Re}\left[U_m\right]}\frac{\partial\,\mathrm{Re}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}+\frac{\partial\,\mathrm{Re}\left[S_n\right]}{\partial\,\mathrm{Im}\left[H_m\right]}\frac{\partial\,\mathrm{Im}\left[H_m\right]}{\partial\,\mathrm{Im}\left[U_m\right]}\frac{\partial\,\mathrm{Im}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\right)-$$

$$-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Im}\left[\delta_n\right]}\frac{\partial\,\mathrm{Im}\left[\delta_n\right]}{\partial\,\mathrm{Im}\left[S_n\right]}\left(\frac{\partial\,\mathrm{Im}\left[S_n\right]}{\partial\,\mathrm{Im}\left[H_m\right]}\frac{\partial\,\mathrm{Re}\left[H_m\right]}{\partial\,\mathrm{Re}\left[U_m\right]}\frac{\partial\,\mathrm{Re}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}+\frac{\partial\,\mathrm{Im}\left[S_n\right]}{\partial\,\mathrm{Im}\left[H_m\right]}\frac{\partial\,\mathrm{Im}\left[H_m\right]}{\partial\,\mathrm{Im}\left[U_m\right]}\frac{\partial\,\mathrm{Im}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\right)=$$

$$=-\frac{\partial\,\mathrm{Re}\left[H_m\right]}{\partial\,\mathrm{Re}\left[U_m\right]}\frac{\partial\,\mathrm{Re}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\delta_n\right]}\frac{\partial\,\mathrm{Re}\left[\delta_n\right]}{\partial\,\mathrm{Re}\left[S_n\right]}\frac{\partial\,\mathrm{Re}\left[S_n\right]}{\partial\,\mathrm{Re}\left[H_m\right]}-$$

$$-\frac{\partial\,\mathrm{Im}\left[H_m\right]}{\partial\,\mathrm{Im}\left[U_m\right]}\frac{\partial\,\mathrm{Im}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Re}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Re}\left[\delta_n\right]}\frac{\partial\,\mathrm{Re}\left[\delta_n\right]}{\partial\,\mathrm{Re}\left[S_n\right]}\frac{\partial\,\mathrm{Re}\left[S_n\right]}{\partial\,\mathrm{Im}\left[H_m\right]}-$$

$$-\frac{\partial\,\mathrm{Re}\left[H_m\right]}{\partial\,\mathrm{Re}\left[U_m\right]}\frac{\partial\,\mathrm{Re}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Im}\left[\delta_n\right]}\frac{\partial\,\mathrm{Im}\left[\delta_n\right]}{\partial\,\mathrm{Im}\left[S_n\right]}\frac{\partial\,\mathrm{Im}\left[S_n\right]}{\partial\,\mathrm{Re}\left[H_m\right]}-$$

$$-\frac{\partial\,\mathrm{Im}\left[H_m\right]}{\partial\,\mathrm{Im}\left[U_m\right]}\frac{\partial\,\mathrm{Im}\left[U_m\right]}{\partial\,\mathrm{Re}\left[\omega_{ml}\right]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Im}\left[\delta_n\right]^2\right)}{\partial\,\mathrm{Im}\left[\delta_n\right]}\frac{\partial\,\mathrm{Im}\left[\delta_n\right]}{\partial\,\mathrm{Im}\left[S_n\right]}\frac{\partial\,\mathrm{Im}\left[S_n\right]}{\partial\,\mathrm{Im}\left[H_m\right]}$$

Usando las expresiones a), b), c), d) queda:

$$\mathrm{Re}\left[\Delta\omega_{ml}\right]=f'_{\mathrm{Re}}\left(\mathrm{Re}\left[U_m\right]\right)\mathrm{Re}\left[I_l\right]\sum_{n=1}^{N}\mathrm{Re}\left[\delta_n\right]f'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)\mathrm{Re}\left[\upsilon_{nm}\right]-$$

$$-f'_{\mathrm{Im}}\left(\mathrm{Im}\left[U_m\right]\right)\mathrm{Im}\left[I_l\right]\sum_{n=1}^{N}\mathrm{Re}\left[\delta_n\right]f'_{\mathrm{Re}}\left(\mathrm{Re}\left[S_n\right]\right)\mathrm{Im}\left[\upsilon_{nm}\right]+$$

$$+f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+$$

$$+f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]$$

$$\mathrm{Im}[\Delta\omega_{ml}]=-\frac{\partial E_p}{\partial\,\mathrm{Im}[\omega_{ml}]}=-\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}|\delta_n|^2\right)}{\partial\,\mathrm{Im}[\omega_{ml}]}=-\frac{\partial\left(\frac{1}{2}\sum_{n=1}^{N}\left(\mathrm{Re}[\delta_n]^2+\mathrm{Im}[\delta_n]^2\right)\right)}{\partial\,\mathrm{Im}[\omega_{ml}]}=$$

$$=-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\omega_{ml}]}-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\omega_{ml}]}=$$

$$=-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Re}[\delta_n]}\frac{\partial\,\mathrm{Re}[\delta_n]}{\partial\,\mathrm{Re}[S_n]}\left(\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Re}[H_m]}\frac{\partial\,\mathrm{Re}[H_m]}{\partial\,\mathrm{Re}[U_m]}\frac{\partial\,\mathrm{Re}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}+\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Im}[H_m]}\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\right)-$$

$$-\sum_{n=1}^{N}\frac{\partial\left(\frac{1}{2}\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\left(\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Re}[H_m]}\frac{\partial\,\mathrm{Re}[H_m]}{\partial\,\mathrm{Re}[U_m]}\frac{\partial\,\mathrm{Re}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}+\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Im}[H_m]}\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\right)=$$

$$=-\frac{\partial\,\mathrm{Re}[H_m]}{\partial\,\mathrm{Re}[U_m]}\frac{\partial\,\mathrm{Re}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Re}[\delta_n]}\frac{\partial\,\mathrm{Re}[\delta_n]}{\partial\,\mathrm{Re}[S_n]}\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Re}[H_m]}-$$

$$-\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Re}[\delta_n]^2\right)}{\partial\,\mathrm{Re}[\delta_n]}\frac{\partial\,\mathrm{Re}[\delta_n]}{\partial\,\mathrm{Re}[S_n]}\frac{\partial\,\mathrm{Re}[S_n]}{\partial\,\mathrm{Im}[H_m]}-$$

$$-\frac{\partial\,\mathrm{Re}[H_m]}{\partial\,\mathrm{Re}[U_m]}\frac{\partial\,\mathrm{Re}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Re}[H_m]}-$$

$$-\frac{\partial\,\mathrm{Im}[H_m]}{\partial\,\mathrm{Im}[U_m]}\frac{\partial\,\mathrm{Im}[U_m]}{\partial\,\mathrm{Im}[\omega_{ml}]}\sum_{n=1}^{N}\frac{\partial\left((1/2)\,\mathrm{Im}[\delta_n]^2\right)}{\partial\,\mathrm{Im}[\delta_n]}\frac{\partial\,\mathrm{Im}[\delta_n]}{\partial\,\mathrm{Im}[S_n]}\frac{\partial\,\mathrm{Im}[S_n]}{\partial\,\mathrm{Im}[H_m]}$$

Usando las expresiones a), b), c), d) queda:

$$\mathrm{Im}[\Delta\omega_{ml}]=-f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]-$$

$$-f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]-$$

$$-f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+$$

$$+f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]$$

En suma:

$$\begin{aligned}\Delta\omega_{ml}=\Bigg(&f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]-\\&-f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+\\&+f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+\\&+f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]\Bigg)+\\+i\Bigg(&-f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]-\\&-f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\,f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]-\\&-f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]+\\&+f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Im}[\delta_n]\,f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]\Bigg)\end{aligned}$$

Existen unas expresiones que relacionan las fórmulas anteriores (* denota complejo conjugado):

$$H_m^* \Delta\gamma_n = \left(\mathrm{Re}[H_m] - i\,\mathrm{Im}[H_m]\right)\cdot\left(\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right) + i\,\mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\right) =$$

$$= \mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[H_m] +$$

$$+ i\,\mathrm{Re}[H_m]\mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right) - i\,\mathrm{Im}[H_m]\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right) =$$

$$= \left(\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[H_m] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[H_m]\right) +$$

$$+ i\left(-\mathrm{Im}[H_m]\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right) + \mathrm{Re}[H_m]\mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\right) = \Delta\upsilon_{nm}$$

$$I_l^* \Delta\theta_m = \left(\mathrm{Re}[I_l] - i\,\mathrm{Im}[I_l]\right)\cdot$$

$$\cdot\left(f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\sum_{n=1}^{N}\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] +\right.$$

$$\left. + i\left(f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\sum_{n=1}^{N} -\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]\right)\right) =$$

$$= f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] +$$

$$+ f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N} -\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] +$$

$$+ i\left(f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\mathrm{Re}[I_l]\sum_{n=1}^{N} -\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] -\right.$$

$$\left. - f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\mathrm{Im}[I_l]\sum_{n=1}^{N}\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]\right) = \Delta\omega_{ml}$$

En resumen, la regla de actualización de los parámetros de la red neuronal de variable compleja es (se retoma el valor original de ε):

$$\Delta\gamma_n = \varepsilon\left(\mathrm{Re}[\delta_n]\, f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right) + i\,\mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\right)$$

$$\Delta\upsilon_{nm} = H_m^*\Delta\gamma_n$$

$$\Delta\theta_m = \varepsilon\left(\left(f'_{\mathrm{Re}}\left(\mathrm{Re}[U_m]\right)\sum_{n=1}^{N}\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Re}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Im}[\upsilon_{nm}]\right) + \right.$$
$$\left. + i\left(f'_{\mathrm{Im}}\left(\mathrm{Im}[U_m]\right)\sum_{n=1}^{N} -\mathrm{Re}[\delta_n] f'_{\mathrm{Re}}\left(\mathrm{Re}[S_n]\right)\mathrm{Im}[\upsilon_{nm}] + \mathrm{Im}[\delta_n]\, f'_{\mathrm{Im}}\left(\mathrm{Im}[S_n]\right)\mathrm{Re}[\upsilon_{nm}]\right)\right)$$

$$\Delta\omega_{ml} = I_l^*\Delta\theta_m$$

Si

$$f_{\mathrm{Re}}(x) = f_{\mathrm{Im}}(x) = \frac{1}{1+\exp(-x)}$$

entonces

$$f'_{\mathrm{Re}}(x) = f_{\mathrm{Re}}(x)\left(1 - f_{\mathrm{Re}}(x)\right)$$

$$f'_{\mathrm{Im}}(x) = f_{\mathrm{Im}}(x)\left(1 - f_{\mathrm{Im}}(x)\right)$$

y sustituyendo en las expresiones de $\Delta\gamma_n$ y $\Delta\theta_m$:

$$\Delta\gamma_n = \varepsilon\left(\mathrm{Re}[\delta_n]\mathrm{Re}[O_n]\left(1-\mathrm{Re}[O_n]\right) + i\,\mathrm{Im}[\delta_n]\mathrm{Im}[O_n]\left(1-\mathrm{Im}[O_n]\right)\right)$$

$$\Delta\theta_m = \varepsilon\left(\mathrm{Re}[H_m]\left(1-\mathrm{Re}[H_m]\right)\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\mathrm{Re}[O_n]\left(1-\mathrm{Re}[O_n]\right)\mathrm{Re}[\upsilon_{nm}] + \right.$$
$$+ \mathrm{Im}[\delta_n]\mathrm{Im}[O_n]\left(1-\mathrm{Im}[O_n]\right)\mathrm{Im}[\upsilon_{nm}] +$$
$$-i\left(\mathrm{Im}[H_m]\left(1-\mathrm{Im}[H_m]\right)\sum_{n=1}^{N}\mathrm{Re}[\delta_n]\mathrm{Re}[O_n]\left(1-\mathrm{Re}[O_n]\right)\mathrm{Im}[\upsilon_{nm}] + \right.$$
$$\left.\left. - \mathrm{Im}[\delta_n]\mathrm{Im}[O_n]\left(1-\mathrm{Im}[O_n]\right)\mathrm{Re}[\upsilon_{nm}]\right)\right)$$

que son las expresiones deducidas por Nitta [101].

3.3. El algoritmo de retropropagación complejo con momentum.

Como ya se comentó en el Capítulo 2, uno de los mayores problemas de las redes neuronales es la lentitud de su entrenamiento. Una solución que aumenta la velocidad y estabilidad del proceso de aprendizaje consiste en añadir un término momento en la actualización de los parámetros de la red:

$$\omega_n = \omega_{n-1} + \eta\Delta\omega_n + \tau\Delta\omega_{n-1}$$

Zhang y Wei [146] presentaron una versión compleja del algoritmo de retropropagación con momentum para CVNNs con funciones de activación split del tipo real-imaginario. En la descripción de este algoritmo, usaremos la siguiente nomenclatura:

- ω_n: peso de una conexión entre dos neuronas en la etapa *n* del proceso de aprendizaje
- $\Delta\omega_n = \omega_n - \omega_{n-1}$
- $E(W_n)$: función de error de la red al comenzar la etapa *n*
- η: factor de aprendizaje
- $\tau \in (0,1)$: momentum
- $p_n^{\mathrm{Re}} = \dfrac{\partial E(W_n)}{\partial \omega^{\mathrm{Re}}}$
- $p_n^{\mathrm{Im}} = \dfrac{\partial E(W_n)}{\partial \omega^{\mathrm{Im}}}$
- $\tau_n^{\mathrm{Re}} = \begin{cases} \tau \dfrac{\left|p_n^{\mathrm{Re}}\right|}{\left|\Delta\omega_n^{\mathrm{Re}}\right|} & si \ \left|\Delta\omega_n^{\mathrm{Re}}\right| \neq 0 \\ 0 & si \ \left|\Delta\omega_n^{\mathrm{Re}}\right| = 0 \end{cases}$
- $\tau_n^{\mathrm{Im}} = \begin{cases} \tau \dfrac{\left|p_n^{\mathrm{Im}}\right|}{\left|\Delta\omega_n^{\mathrm{Im}}\right|} & si \ \left|\Delta\omega_n^{\mathrm{Im}}\right| \neq 0 \\ 0 & si \ \left|\Delta\omega_n^{\mathrm{Im}}\right| = 0 \end{cases}$

siendo $\left|\ \right|$ la norma euclidea.

La actualización de la etapa *n* viene dada por las expresiones:

$$\Delta\omega_{n+1}^{\mathrm{Re}} = -\eta\, p + \tau_n^{\mathrm{Re}} \Delta\omega_n^{\mathrm{Re}}$$

$$\Delta\omega_{n+1}^{\mathrm{Im}} = -\eta\, p + \tau_n^{\mathrm{Im}} \Delta\omega_n^{\mathrm{Im}}$$

Zhang y Wei demostraron en su artículo que, bajo ciertas hipótesis, el algoritmo backpropagation complejo split real-complejo con momentum usado con una red neuronal de dos capas, converge y el error se reduce monótonamente.

Veamos, por ejemplo, el efecto de aplicar el momento cuando la red se entrena con el fin de aprender los patrones de los Experimentos 1 y 2 de Nitta [101]. En el Experimento 1, la red debe aprender los siguientes patrones:

z	***t***
0	0
i	1
1	$1+i$
$1+i$	i

Tabla 3.5. Patrones del Experimento 1 de Nitta

donde *z* y *t* representan los valores de los patrones de entrada y salida, respectivamente.

Se elige una arquitectura 1-3-1. Los valores iniciales de los pesos y bias se escogen aleatoriamente en el intervalo $[-0.3, 0.3]$; la función de activación de la red tiene sus partes real e imaginaria desacopladas, siendo cada una de ellas la función logística con parámetro 1.0, y el algoritmo de retropropagación se detiene cuando se cumple

$$\sqrt{\sum_p \sum_n \left| T_n^p - O_n^p \right|^2} < 0.10$$

La red neuronal es entrenada con la versión batch del algoritmo de retropropagación; el factor de aprendizaje γ toma los valores 0.1, 0.2, 0.3, 0.4, 0.5 y 0.6. En el entrenamiento se emplea un momento *m*. Fijado *m=0.00*, se repite el experimento hasta lograr 100 entrenamientos exitosos (entrenamientos realizados en menos de 100000 épocas) para cada uno de los valores del factor de aprendizaje.

La figura siguiente muestra el número de épocas necesarias para la convergencia de los entrenamientos con éxito del Experimento 1 con momento *m=0.00*:

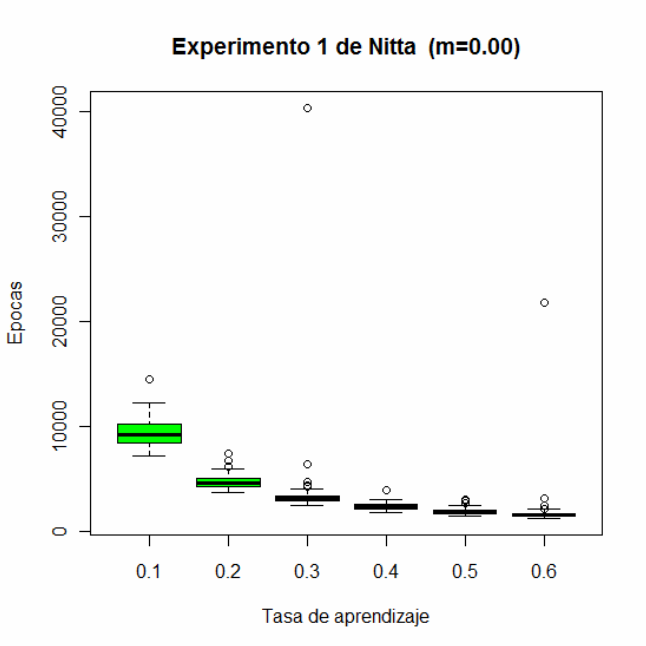


Figura 3.6. Resultados del Experimento 1 (con momento 0.00)

Definiendo el ratio de convergencia RC como:

$$RC = \frac{número\ entrenamientos\ exitosos}{número\ total\ entrenamientos} \cdot 100$$

en el Experimento 1 con momento 0.00 se han encontrado los siguientes valores:

γ	0.1	0.2	0.3	0.4	0.5	0.6
RC	100	100	100	100	100	100

Tabla 3.6. Ratio de convergencia del Experimento 1 (con momento 0.00)

La figura Figura 3.7. muestra los resultados correspondientes a los entrenamientos exitosos de Experimento 1 con *m=0.06*:

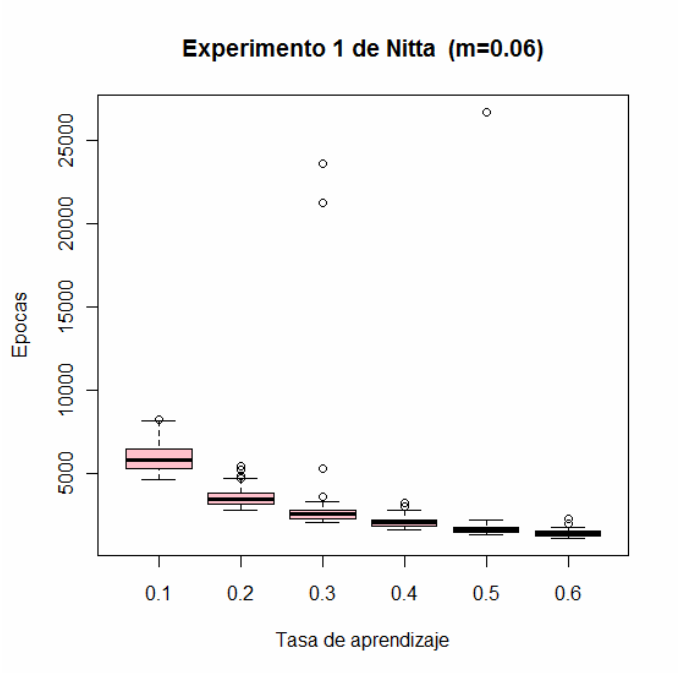


Figura 3.7. Resultados del Experimento 1 (con momento 0.06)

y los ratios de convergencia correspondientes son:

γ	0.1	0.2	0.3	0.4	0.5	0.6
RC	100	100	100	100	100	100

Tabla 3.7. Ratio de convergencia del Experimento 1 (con momento 0.06)

Los patrones a aprender en el Experimento 2 son los siguientes:

z_1	z_2	t
0	0	1
0	i	i
i	i	$1+i$
i	1	i
1	1	$1+i$
i	0	0
$1+i$	$1+i$	1
$1+i$	i	i

Tabla 3.8. Patrones del Experimento 2 de Nitta

donde z_1 y z_2 representan los valores de los patrones de entrada, y *t* el valor del patrón de salida.

En este caso, se elige una arquitectura de red 2-4-1; los valores iniciales de pesos y bias, las funciones de activación, el criterio de parada, los valores del factor de aprendizaje γ y la versión del algoritmo de retropropagación se escogen igual que en el Experimento 1. El entrenamiento tambíen se realiza con un momento *m*. Se fija el valor *m=0.00*, y se repite el experimento hasta lograr 100 entrenamientos exitosos para cada uno de los valores del factor de aprendizaje. En la Figura 3.8. se muestran las épocas necesarias para la convergencia de los entrenamientos con éxito:

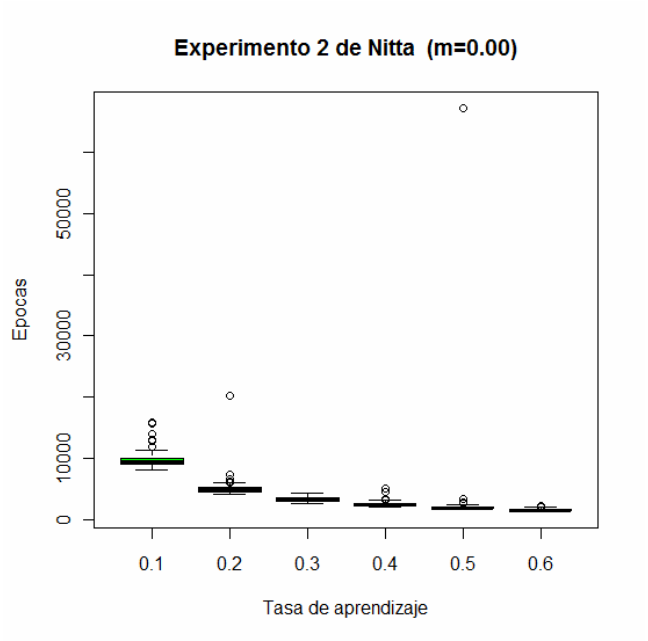


Figura 3.8. Resultados del Experimento 2 (con momento 0.00)

Sus ratios de convergencia son:

γ	0.1	0.2	0.3	0.4	0.5	0.6
RC	94.34	98.04	97.09	100	99.01	96.15

Tabla 3.9. Ratio de convergencia del Experimento 2 (con momento 0.00)

En la figura Figura 3.9. se muestran los resultados de los entrenamientos exitosos correspondientes al Experimento 2 con *m=0.06*:

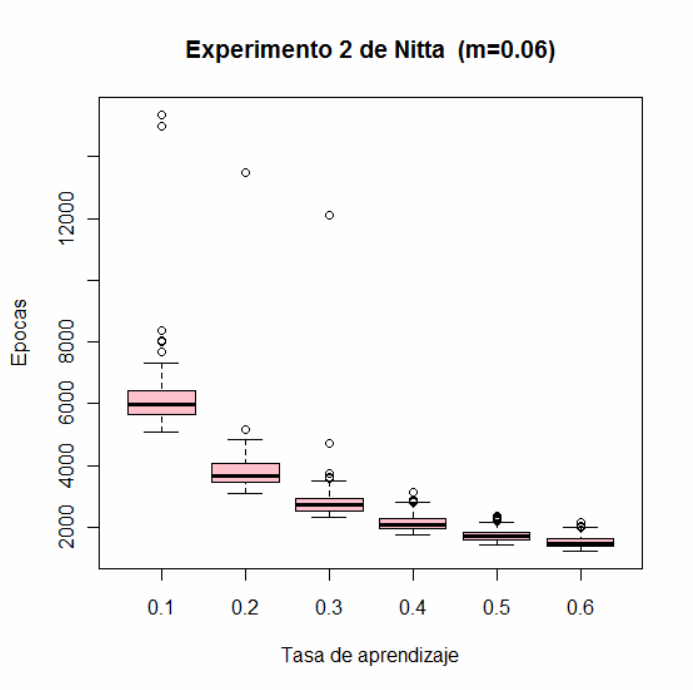


Figura 3.9. Resultados del Experimento 2 (con momento 0.06)

Sus ratios de convergencia son:

γ	0.1	0.2	0.3	0.4	0.5	0.6
RC	98.04	97.09	99.01	96.15	98.04	97.09

Tabla 3.10. Ratio de convergencia del Experimento 2 (con momento 0.06)

En ambos Experimentos se observa una disminución del número de épocas necesarias para entrenar a la red cuando aumenta el valor del momento.

3.4. La ortogonalidad de las fronteras de decisión de las neuronas complejas.

En el análisis del comportamiento de las redes neuronales resulta útil estudiar las regiones de decisión en las que las redes dividen el espacio *n*-dimensional de datos de entrada. A diferencia de lo que sucede en el caso real, las fronteras de decisión de las CVNNs son ortogonales. Esta propiedad confiere a las redes neuronales complejas una mayor capacidad de generalización de la que poseen sus homólogas reales.

Comencemos demostrando la propiedad de ortogonalidad de las fronteras de decisión en el caso de que el perceptrón complejo conste de una única neurona [65]: llamando

ω: vector de pesos (o matriz columna) de pesos de la neurona, siendo $\omega_m \in \mathbb{C}$ $m \in \{1,...,M\}$ la componente m-ésima de ω, con M número de entradas del perceptrón. Si denominamos ω_m^r y ω_m^i a las partes real y compleja de ω_m, respectivamente, se puede escribir:

$$\omega = \begin{bmatrix} \omega_1 \\ \cdot \\ \cdot \\ \cdot \\ \omega_M \end{bmatrix} = \left[\omega_1,...,\omega_M\right]^t = \left[\omega_1^r + i\omega_1^i,...,\omega_M^r + i\omega_M^i\right]^t = \left[\omega_1^r,...,\omega_M^r\right]^t + i\left[\omega_1^i,...,\omega_M^i\right]^t =$$

$$= \omega^r + i\omega^i$$

donde $\omega^r = \left[\omega_1^r,...,\omega_M^r\right]^t$ y $\omega^i = \left[\omega_1^i,...,\omega_M^i\right]^t$.

z: vector (o matriz columna) de entradas a la neurona, con $z_m \in \mathbb{C}$ $m \in \{1,...,M\}$ la componente m-ésima de *z*. Se puede escribir:

$$z = \begin{bmatrix} z_1 \\ \cdot \\ \cdot \\ \cdot \\ z_M \end{bmatrix} = \left[z_1,...,z_M\right]^t = \left[z_1^r + iz_1^i,...,z_M^r + iz_M^i\right]^t = \left[z_1^r,...,z_M^r\right]^t + i\left[z_1^i,...,z_M^i\right]^t =$$

$$= x + iy$$

donde $x = \left[z_1^r,...,z_M^r\right]^t$ e $y = \left[z_1^i,...,z_M^i\right]^t$

θ: bias de la neurona. Llamando θ^r, θ^i a la parte real y compleja de θ, respectivamente: $\theta = \theta^r + i\theta^i$

f_C: función de activación de la neurona. Se va a considerar que tiene la forma:

$$f_C(z) = f_R(x) + if_R(y) \quad \forall z = x + iy \in \mathbb{C}$$

siendo

$$f_R(u) = \frac{1}{1+e^{-u}} \quad \forall u \in \mathbb{R}$$

Utilizando esta notación, la salida del perceptrón, que en este caso está formado por una única neurona, tiene la siguiente expresión:

$$X + iY = f_C(\omega \circ z + \theta)$$

donde $\circ$ representa el producto escalar de dos vectores. Desarrollando esta expresión:

$$X + iY = f_C\left(\sum_{m=1}^{M} \omega_m z_m + \theta\right) = f_C\left(\sum_{m=1}^{M} \left(\omega_m^r + i\omega_m^i\right)\left(z_m^r + iz_m^i\right) + \theta\right) =$$

$$= f_C\left(\sum_{m=1}^{M} \left(\omega_m^r z_m^r - \omega_m^i z_m^i\right) + i\left(\omega_m^i z_m^r + \omega_m^r z_m^i\right) + \theta\right) =$$

$$= f_C\left(\left(\sum_{m=1}^{M} \omega_m^r z_m^r - \omega_m^i z_m^i\right) + i\left(\sum_{m=1}^{M} \omega_m^i z_m^r + \omega_m^r z_m^i\right) + \theta\right) =$$

$$= f_C\left(\left(\sum_{m=1}^{M} \omega_m^r z_m^r - \omega_m^i z_m^i + \theta^r\right) + i\left(\sum_{m=1}^{M} \omega_m^i z_m^r + \omega_m^r z_m^i + \theta^i\right)\right) =$$

$$= f_C\left(\left(\begin{bmatrix} {}^t\omega^r & -{}^t\omega^i \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) + i\left(\begin{bmatrix} {}^t\omega^i & {}^t\omega^r \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right)\right) =$$

$$= f_R\left(\begin{bmatrix} {}^t\omega^r & -{}^t\omega^i \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) + if_R\left(\begin{bmatrix} {}^t\omega^i & {}^t\omega^r \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right)$$

Tomando dos valores constantes $C^R, C^I \in (0,1)$, se obtienen las fronteras de decisión real y compleja:

$$X(x,y) = f_R\left(\begin{bmatrix} {}^t\omega^r & -{}^t\omega^i \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^r\right) = C^R \qquad \text{Frontera real}$$

$$Y(x,y) = f_R\left(\begin{bmatrix} {}^t\omega^i & {}^t\omega^r \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) = C^I \qquad \text{Frontera compleja}$$

que son curvas en $\mathbb{C}^M$. Podemos analizar estas curvas considerando que su dominio es $\mathbb{R}^{2M}$.

Los vectores normales a las fronteras de decisión son:

$$Q^R(x,y)=\left(\frac{\partial X}{\partial x_1},...,\frac{\partial X}{\partial x_M},\frac{\partial X}{\partial y_1},...,\frac{\partial X}{\partial y_M}\right)=$$

$$=f'_R\left(\begin{bmatrix}{}^t\omega^r & -{}^t\omega^i\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\theta^r\right)\begin{bmatrix}{}^t\omega^r & -{}^t\omega^i\end{bmatrix}$$

$$Q^I(x,y)=\left(\frac{\partial Y}{\partial x_1},...,\frac{\partial Y}{\partial x_M},\frac{\partial Y}{\partial y_1},...,\frac{\partial Y}{\partial y_M}\right)=$$

$$=f'_R\left(\begin{bmatrix}{}^t\omega^i & {}^t\omega^r\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\theta^i\right)\begin{bmatrix}{}^t\omega^i & {}^t\omega^r\end{bmatrix}$$

(para su cálculo se ha empleado la expresión de la derivada de la función sigmoidal. Ver Apéndice 1).

Notemos que los vectores normales de cada frontera tienen dirección constante; sólo cambia su magnitud. Esto quiere decir que las fronteras de decisión de un perceptrón complejo que conste de una única neurona y con función de activación $f_C(z)=\left(1/1+e^{-x}\right)+i\left(1/1+e^{-y}\right)$, son rectas. Haciendo el producto escalar de los vectores normales a las fronteras de decisión se comprueba su ortogonalidad:

$$Q^R\circ Q^I=f'_R\left(\begin{bmatrix}{}^t\omega^r & -{}^t\omega^i\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\theta^r\right)f'_R\left(\begin{bmatrix}{}^t\omega^i & {}^t\omega^r\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\theta^i\right)\begin{bmatrix}{}^t\omega^r & -{}^t\omega^i\end{bmatrix}\begin{bmatrix}\omega^i\\\omega^r\end{bmatrix}=$$

$$={}^t\omega^r\omega^i-{}^t\omega^i\omega^r=0$$

Ahora analicemos las fronteras de decisión de un perceptrón complejo que tiene una capa oculta (esta discusión se puede encontrar en [65]), con un número *L* de neuronas de entrada, *M* neuronas en la capa oculta, y *N* neuronas de salida; además, la función de activación de las neuronas es f_C, tal como se ha definido en el caso del perceptrón con una sola neurona. Llamando:

- $\omega_{ji}=\omega_{ji}^r+i\omega_{ji}^i$, peso de la conexión entre la neurona de entrada i y la neurona oculta j
- $\omega_j=\begin{bmatrix}\omega_{j1},...,\omega_{jL}\end{bmatrix}^t$
- $\upsilon_{kj}=\upsilon_{kj}^r+i\upsilon_{kj}^i$, peso de la conexión entre la neurona oculta j y la neurona de salida k

- $\upsilon_k = [\upsilon_{k1},...,\upsilon_{kM}]^t$
- $\theta_j = \theta_j^r + i\theta_j^i$, bias de la neurona oculta j
- $\gamma_k = \gamma_k^r + i\gamma_k^i$, bias de la neurona de salida k
- *z* el vector (o matriz columna) de entrada:

$$z = \begin{bmatrix} z_1 \\ \cdot \\ \cdot \\ \cdot \\ z_L \end{bmatrix} = [z_1,...,z_L]^t = [z_1^r + iz_1^i,...,z_L^r + iz_L^i]^t = [z_1^r,...,z_L^r]^t + i[z_1^i,...,z_L^i]^t = x + iy$$

donde $x = [z_1^r,...,z_L^r]^t$ e $y = [z_1^i,...,z_L^i]^t$

- U_j: activación de la neurona j de la capa oculta, cuyo valor es:

$$U_j = U_j^r + iU_j^i = \omega_j \circ z + \theta_j$$

Desarrollando:

$$U_j = \begin{bmatrix} {}^t\omega_j^r & -{}^t\omega_j^i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + i \begin{bmatrix} {}^t\omega_j^i & {}^t\omega_j^r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \theta_j$$

(el desarrollo es similar al realizado cuando el perceptrón consta de una única neurona)

- H_j: salida de la neurona j de la capa oculta:

$$H_j = H_j^r + iH_j^i = f_R(U_j^r) + if_R(U_j^i) =$$

$$= f_R\left(\begin{bmatrix} {}^t\omega_j^r & -{}^t\omega_j^i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \theta_j^r \right) + if_R\left(\begin{bmatrix} {}^t\omega_j^i & {}^t\omega_j^r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \theta_j^i \right)$$

- $H = [H_1,...,H_M]^t = [H_1^r,...,H_L^r]^t + i[H_1^i,...,H_L^i]^t = H^r + iH^i$
- S_k: activación de la neurona k de la capa de salida, cuyo valor es:

$$S_k = S_k^r + iS_k^i = \upsilon_k \circ H + \gamma_k$$

Desarrollando esta expresión:

$$S_k = S_k^r + iS_k^i = \begin{bmatrix} {}^t\upsilon_k^r & -{}^t\upsilon_k^i \end{bmatrix} \begin{bmatrix} H^r \\ H^i \end{bmatrix} + i \begin{bmatrix} {}^t\upsilon_k^i & {}^t\upsilon_k^r \end{bmatrix} \begin{bmatrix} H^r \\ H^i \end{bmatrix} + \gamma_k$$

- O_k : salida de la neurona k de la capa de salida:

$$O_k = O_k^r + iO_k^i = f_R\left(S_k^r\right) + if_R\left(S_k^i\right) =$$

$$= f_R\left(\begin{bmatrix} {}^t\upsilon_k^r & -{}^t\upsilon_k^i \end{bmatrix} \begin{bmatrix} H^r \\ H^i \end{bmatrix} + \gamma_k^r \right) + if_R\left(\begin{bmatrix} {}^t\upsilon_k^i & {}^t\upsilon_k^r \end{bmatrix} \begin{bmatrix} H^r \\ H^i \end{bmatrix} + \gamma_k^i \right)$$

A partir de los valores constantes $C^R, C^I \in (0,1)$ se crean las fronteras de decisión de la neurona k de la capa de salida:

$$O_k^r(x,y) = C^R$$

$$O_k^i(x,y) = C^I$$

Estudiemos la ortogonalidad de los vectores normales a estas curvas:

$$Q_k^R(x,y) = \left(\frac{\partial O_k^r}{\partial x_1}, ..., \frac{\partial O_k^r}{\partial x_L}, \frac{\partial O_k^r}{\partial y_1}, ..., \frac{\partial O_k^r}{\partial y_L} \right)$$

$$Q_k^I(x,y) = \left(\frac{\partial O_k^i}{\partial x_1}, ..., \frac{\partial O_k^i}{\partial x_L}, \frac{\partial O_k^i}{\partial y_1}, ..., \frac{\partial O_k^i}{\partial y_L} \right)$$

$$Q_k^R(x,y) \circ Q_k^I(x,y) = \frac{\partial O_k^r}{\partial x_1}\frac{\partial O_k^i}{\partial x_1} + ... + \frac{\partial O_k^r}{\partial x_L}\frac{\partial O_k^i}{\partial x_L} + \frac{\partial O_k^r}{\partial y_1}\frac{\partial O_k^i}{\partial y_1} + ... + \frac{\partial O_k^r}{\partial y_L}\frac{\partial O_k^i}{\partial y_L} =$$

$$= \left(\frac{\partial O_k^r}{\partial x_1}\frac{\partial O_k^i}{\partial x_1} + \frac{\partial O_k^r}{\partial y_1}\frac{\partial O_k^i}{\partial y_1} \right) + ... + \left(\frac{\partial O_k^r}{\partial x_L}\frac{\partial O_k^i}{\partial x_L} + \frac{\partial O_k^r}{\partial y_L}\frac{\partial O_k^i}{\partial y_L} \right) = \sum_{i=1}^{L} T_i^k$$

donde

$$T_i^k = \frac{\partial O_k^r}{\partial x_i}\frac{\partial O_k^i}{\partial x_i} + \frac{\partial O_k^r}{\partial y_i}\frac{\partial O_k^i}{\partial y_i}$$

Para desarrollar $T_i^k \quad i \in \{1,...,L\}$, previamente se deben calcular:

$$\frac{\partial O_k^r}{\partial x_i} = \frac{\partial O_k^r}{\partial S_k^r}\frac{\partial S_k^r}{\partial x_i} = \frac{\partial O_k^r}{\partial S_k^r}\sum_{j=1}^{M}\frac{\partial S_k^r}{\partial H_j^r}\frac{\partial H_j^r}{\partial U_j^r}\frac{\partial U_j^r}{\partial x_i} + \frac{\partial S_k^r}{\partial H_j^i}\frac{\partial H_j^i}{\partial U_j^i}\frac{\partial U_j^i}{\partial x_i} =$$

$$= f'_R\left(S_k^r\right)\sum_{j=1}^{M}\upsilon_{kj}^r\omega_{ji}^r f'_R\left(U_j^r\right) - \upsilon_{kj}^i\omega_{ji}^i f'_R\left(U_j^i\right)$$

$$\frac{\partial O_k^i}{\partial x_i} = \frac{\partial O_k^i}{\partial S_k^i}\frac{\partial S_k^i}{\partial x_i} = \frac{\partial O_k^i}{\partial S_k^i}\sum_{j=1}^{M}\frac{\partial S_k^i}{\partial H_j^r}\frac{\partial H_j^r}{\partial U_j^r}\frac{\partial U_j^r}{\partial x_i} + \frac{\partial S_k^i}{\partial H_j^i}\frac{\partial H_j^i}{\partial U_j^i}\frac{\partial U_j^i}{\partial x_i} =$$

$$= f'_R\left(S_k^i\right)\sum_{j=1}^{M}\upsilon_{kj}^i\omega_{ji}^r f'_R\left(U_j^r\right) + \upsilon_{kj}^r\omega_{ji}^i f'_R\left(U_j^i\right)$$

$$\frac{\partial O_k^r}{\partial y_i} = \frac{\partial O_k^r}{\partial S_k^r}\frac{\partial S_k^r}{\partial y_i} = \frac{\partial O_k^r}{\partial S_k^r}\sum_{j=1}^{M}\frac{\partial S_k^r}{\partial H_j^r}\frac{\partial H_j^r}{\partial U_j^r}\frac{\partial U_j^r}{\partial y_i} + \frac{\partial S_k^r}{\partial H_j^i}\frac{\partial H_j^i}{\partial U_j^i}\frac{\partial U_j^i}{\partial y_i} =$$

$$= f'_R\left(S_k^r\right)\sum_{j=1}^{M} -\upsilon_{kj}^r\omega_{ji}^i f'_R\left(U_j^r\right) - \upsilon_{kj}^i\omega_{ji}^r f'_R\left(U_j^i\right)$$

$$\frac{\partial O_k^i}{\partial y_i} = \frac{\partial O_k^i}{\partial S_k^i}\frac{\partial S_k^i}{\partial y_i} = \frac{\partial O_k^i}{\partial S_k^i}\sum_{j=1}^{M}\frac{\partial S_k^i}{\partial H_j^r}\frac{\partial H_j^r}{\partial U_j^r}\frac{\partial U_j^r}{\partial y_i} + \frac{\partial S_k^i}{\partial H_j^i}\frac{\partial H_j^i}{\partial U_j^i}\frac{\partial U_j^i}{\partial y_i} =$$

$$= f'_R\left(S_k^i\right)\sum_{j=1}^{M} -\upsilon_{kj}^i\omega_{ji}^i f'_R\left(U_j^r\right) + \upsilon_{kj}^r\omega_{ji}^r f'_R\left(U_j^i\right)$$

Entonces:

$$T_i^k = \frac{\partial O_k^r}{\partial x_i}\frac{\partial O_k^i}{\partial x_i} + \frac{\partial O_k^r}{\partial y_i}\frac{\partial O_k^i}{\partial y_i} =$$

$$= f'_R\left(S_k^r\right) f'_R\left(S_k^i\right)\cdot\left(\sum_{j=1}^{M}\upsilon_{kj}^r\omega_{ji}^r f'_R\left(U_j^r\right) - \upsilon_{kj}^i\omega_{ji}^i f'_R\left(U_j^i\right)\right)\cdot\left(\sum_{j=1}^{M}\upsilon_{kj}^i\omega_{ji}^r f'_R\left(U_j^r\right) + \upsilon_{kj}^r\omega_{ji}^i f'_R\left(U_j^i\right)\right) -$$

$$- f'_R\left(S_k^r\right) f'_R\left(S_k^i\right)\cdot\left(\sum_{j=1}^{M} -\upsilon_{kj}^i\omega_{ji}^i f'_R\left(U_j^r\right) - \upsilon_{kj}^r\omega_{ji}^r f'_R\left(U_j^i\right)\right)\cdot\left(\sum_{j=1}^{M} -\upsilon_{kj}^i\omega_{ji}^i f'_R\left(U_j^r\right) + \upsilon_{kj}^r\omega_{ji}^r f'_R\left(U_j^i\right)\right)$$

De esta expresión se deduce que, en general:

$$T^k = Q_k^R(x,y) \circ Q_k^I(x,y) = \sum_{i=1}^{L} T_i^k \neq 0$$

en concreto, esto se cumple en el punto de intersección.

Por lo tanto, en este caso, las fronteras de decisión real y compleja de las neuronas de salida no son ortogonales. Sin embargo, se puede asegurar la ortogonalidad si se aplica la siguiente restricción: $\exists k_1, k_2 \in \mathbb{R}$ suficientemente grandes que cumplen

$$\left|U_j^r\right| > k_1$$

$$\left|U_j^i\right| > k_2$$

en este caso, por la forma de la función f_R, se cumple:

$$f'_R\left(U_j^r\right) \simeq f'_R\left(U_j^i\right)$$

y, por tanto $T_i^k \simeq 0 \quad \forall i \in \{1,...,L\}$.

Es decir, en una red neuronal de variable compleja de dos capas, si los valores absolutos de las partes real y compleja de la activación de las neuronas ocultas es suficientemente grande, las fronteras de decisión de las neuronas de salida se intersecan ortogonalmente. El problema es que, en general, el algoritmo de retropropagación no se puede controlar con el fin de asegurar el cumplimiento de esta restricción. De todas formas, la ortogonalidad se suele cumplir en la mayor parte de los casos prácticos.

Por ejemplo [65], la red neuronal de variable compleja de dos capas con topología $1-12-1$, entrenada mediante el algoritmo complex-BP a partir de los patrones

Patrones entrada	Patrones salida
-0.03-0.03i	1+i
0.03-0.03i	i
0.03+0.03i	0
-0.03+0.03i	1

Tabla 3.11. Patrones de entrenamiento de la red neuronal 1-12-1 empleada en el análisis de las fronteras de decisión

y con $RSE \leq 0.05$, tiene las siguientes fronteras de decisión correspondientes a $C^R = C^I = 0.5$:

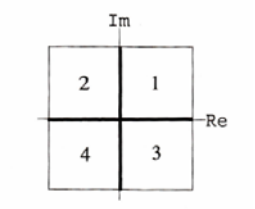


Figura 3.10. Fronteras de decisión de la red neuronal compleja de dos capas $1-12-1$

Como vemos, las fronteras de decisión dividen $\mathbb{C}$, el espacio de entrada, en cuatro regiones. Si $y = \mathrm{Red}(\mathrm{z}) = y^r + iy^i$ representa la salida de la red (ya entrenada) al suministrarle el número complejo *z*, a cada región le corresponde la siguiente salida:

Región 1. $z \in \mathrm{Re}\, g\ 1 \Leftrightarrow y^r < 0.5 \wedge y^i < 0.5$

Región 2. $z \in \mathrm{Re}\, g\ 2 \Leftrightarrow y^r > 0.5 \wedge y^i < 0.5$

Región 3. $z \in \mathrm{Re}\, g\ 3 \Leftrightarrow y^r < 0.5 \wedge y^i > 0.5$

Región 4. $z \in \mathrm{Re}\, g\ 4 \Leftrightarrow y^r > 0.5 \wedge y^i > 0.5$

La frontera vertical (que separa $1 \cup 3$ de $2 \cup 4$) es la frontera de decisión real, y la horizontal (que separa $2 \cup 1$ de $4 \cup 3$) es la frontera de decisión compleja. Ambas fronteras son ortogonales.

Sin embargo, si se entrena una red neuronal real 2-14-2 mediante el algoritmo real-BP a partir de los mismos patrones obtenemos las siguientes fronteras de decisión:

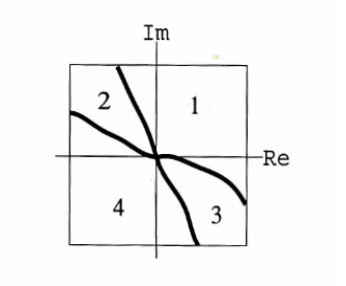


Figura 3.11. Fronteras de decisión de la red neuronal real de doscapas 2-14-2, que claramente no son ortogonales

La ortogonalidad de las fronteras de decisión es una característica clave de las redes neuronales de variable compleja, pues se considera que es una de las principales razones del poder de generalización y rapidez de entrenamiento que, en la práctica, se encuentra en estas redes.

4. La Geometría Proyectiva y sus invariantes

4.1. La Geometría Proyectiva.

La Percepción Visual se caracteriza por captar la forma de los objetos de manera distinta dependiendo del punto de vista del observador. El ejemplo clásico son las vías del tren: vistas desde el aire constituyen dos rectas paralelas, pero observadas a ras del suelo parecen converger:

Figura 4.1 El efecto de perspectiva en las vías de tren.

Los pintores del Renacimiento fueron los primeros en tener en cuenta este efecto en sus obras. Rafael, Piero Della Francesca, Bellini,…, entre otros, dotaron con gran maestría de profundidad escénica a sus obras.

Figura 4.2 La Escuela de Atenas, de Rafael. Una muestra de la perfección en el arte de la perspectiva alcanzada en el Renacimiento.

La Geometría Proyectiva tiene sus raíces en los intentos de estos autores de proporcionar una mayor verosimilitud al efecto de profundidad en sus cuadros, para lo cual realizaron investigaciones acerca de las leyes matemáticas que rigen las relaciones existentes entre la disposición geométrica de los cuerpos y sus "vistas". Con el tiempo estos estudios adquirieron una mayor generalidad y entraron a formar parte del ámbito de trabajo de los matemáticos. El siglo XIX es decisivo para la Geometría Proyectiva, pues en él alcanza su madurez y un grado de elaboración tal que su teoría incluye dentro de un mismo sistema las geometrías de Euclides, Lovatcheski y Riemann [40].

La Geometría Proyectiva es en la actualidad una teoría matemática muy formalizada, sin embargo, el concepto clave que subyace a todo su aparato matemático, el concepto de proyección, es sencillo. En la figura se muestra el resultado de proyectar una figura plana *F* sobre el plano β desde un punto O[10]

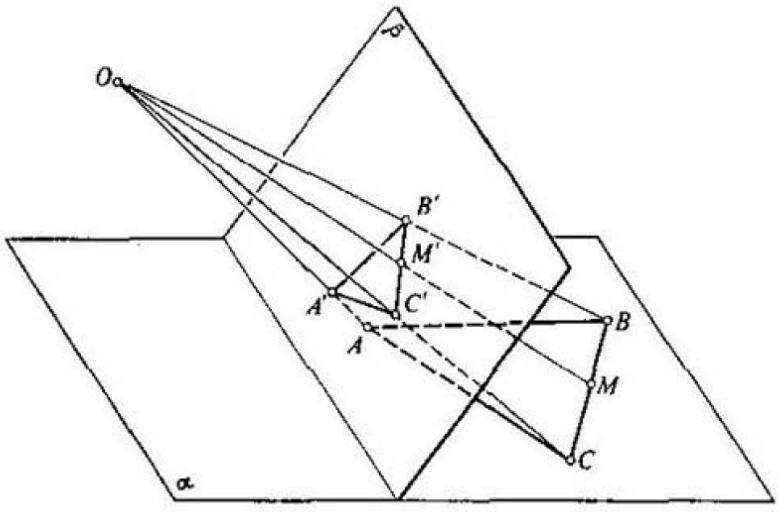


Figura 4.3. Proyección de una figura plana sobre un plano

Las transformaciones entre figuras planas obtenidas mediante proyecciones se denominan homografías. Las figuras homográficas pueden diferir mucho entre si [40]; por ejemplo, una circunferencia puede convertirse en una elipse, parábola o hipérbola, en función de la orientación del plano de proyección, un triángulo regular puede transformarse en otro no regular,… Esto se debe a que las homografías no conservan las propiedades métricas de las figuras (la longitud, el área, etc); sin embargo, existen otras propiedades que si se mantienen, son las denominadas propiedades proyectivas, y constituyen el objeto de estudio de la Geometría Proyectiva [40]. La colinealidad y la pertenencia o no a una cónica, son invariantes proyectivos; esto significa que las rectas y las cónicas forman parte de la Geometría Proyectiva.

[10] En lo que sigue nos vamos a restringir a proyecciones entre figuras planas. El espacio proyectivo que nos interesa es el plano proyectivo P^2.

En una proyección, los rayos paralelos al plano de proyección no intersecan con éste, como se advierte en la figura

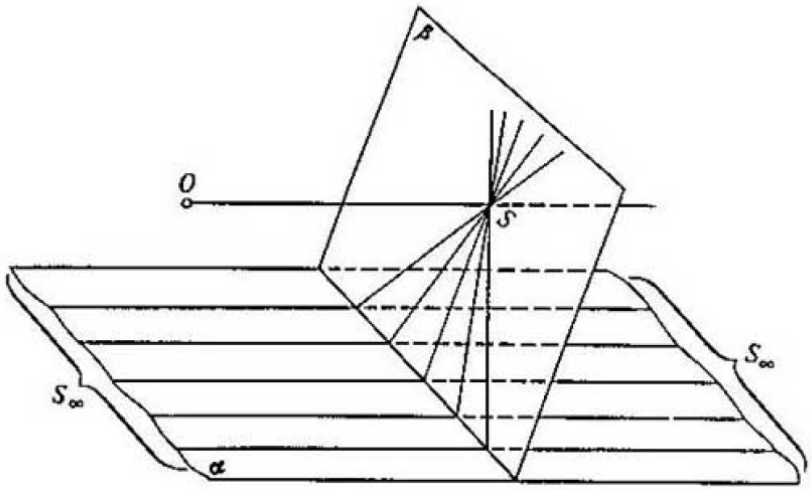


Figura 4.4. Rayo paralelo al plano proyectivo

Este hecho se formaliza en la Geometría Proyectiva añadiendo a los puntos ordinarios del plano los llamados puntos del infinito. A cada rayo paralelo al plano de proyección le corresponde biunívocamente un punto del infinito. La unión de puntos del infinito forma la recta impropia, y el plano proyectivo es la unión de esta recta y el plano usual.

Es necesario señalar que en Geometría Elemental también aparecen los puntos impropios, pero básicamente como una forma diferente de expresar conceptos geométricos; en cambio, en la Geometría Proyectiva los puntos del infinito tienen entidad geométrica propia. La causa de esta diferencia radica en que la Geometría Elemental maneja propiedades métricas, mientras que la Proyectiva no.

La Geometría Proyectiva (en general, la Geometría) se divide en Sintética y Analítica. Ambos tipos de geometrías se complementan, siendo su principal diferencia el utillaje matemático que usan en las demostraciones: lógico-deductivo en el primer caso, el algebra y el análisis matemático en el segundo. En este Proyecto se emplean resultados pertenecientes al enfoque analítico de la Geometría Proyectiva.

Un concepto esencial de la Geometría Proyectiva Analítica es el de las coordenadas homogéneas. A continuación, se definen formalmente el plano proyectivo P^2 y las coordenadas homogéneas [118]:

P^2 es el espacio cociente asociado a la relación de equivalencia ~ de $\mathbb{R}^3 \setminus \{(0,0,0)\}$:

$$\forall u, u' \in \mathbb{R}^3 \setminus \{(0,0,0)\}: \; u \sim u' \text{ si y sólo si } \exists k \neq 0 \in \mathbb{R} \setminus \; u' = ku$$

Las ternas $(u,v,w) \in \mathbb{R}^3 \setminus \{(0,0,0)\}$ se denominan coordenadas homogéneas del plano proyectivo. Notemos que $\mathbb{R}^2 \subset P^2$. Al punto (x,y) de $\mathbb{R}^2$ le corresponden las ternas (kx,ky,k) $k \neq 0$ de P^2, en concreto $(x,y,1)$. Los puntos $(kx,ky,0)$ son los puntos del infinito.

Una homografía de P^2 es una transformación lineal:

$$\begin{array}{ccc} P^2 & \rightarrow & P^2 \\ \begin{bmatrix} kx \\ ky \\ k \end{bmatrix} & \rightarrow & \begin{bmatrix} k'x' \\ k'y' \\ k' \end{bmatrix} = H \cdot \begin{bmatrix} kx \\ ky \\ k \end{bmatrix} \end{array}$$

siendo $[kx,ky,k]^t$ $k \neq 0$ las coordenadas homogéneas de un punto y H una matriz 3x3 invertible

$$H = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

H es matriz de la homografía. Se cumple:

1. Al ser H invertible, la última fila debe tener algún elemento distinto de cero; podemos suponer $a_{33} \neq 0$.

2. Las matrices $k \cdot H$ $\forall k \neq 0$ también son matrices de la homografía.

De 2) se deduce que las homografías tienen, en general, ocho grados de libertad, pues multiplicando $k = 1/a_{33}$ por H se obtiene la matriz

$$H' = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} \\ a'_{21} & a'_{22} & a'_{23} \\ a'_{31} & a'_{32} & 1 \end{bmatrix}$$

la cual, necesita para concretarse (o, lo que es lo mismo, para precisar la homografía) que se de valor a ocho coeficientes. H' es la matriz normal de la homografía.

Las transformaciones proyectivas también se pueden expresar en coordenadas ordinarias: partiendo de la expresión de la transformación en coordenadas homogéneas se deduce

$$\frac{k'x'}{k'} = \frac{a_{11}kx + a_{12}ky + a_{13}k}{a_{31}kx + a_{32}ky + a_{33}k} \qquad \frac{k'y'}{k'} = \frac{a_{21}kx + a_{22}ky + a_{23}k}{a_{31}kx + a_{32}ky + a_{33}k}$$

y, por tanto

$$x' = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}} \qquad y' = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}}$$

Las homografías son transformaciones muy generales que contienen en si dos tipos particulares de transformaciones: de similaridad y afines. Las primeras consisten en combinaciones de traslaciones, giros y cambios de escala; tienen la característica esencial de no deformar las figuras, pues sus ángulos no cambian ni tampoco el cociente entre magnitudes homólogas. Por el contrario, las afinidades deforman las figuras. Algunas propiedades destacables de las afinidades son [54]:

- Preservan las relaciones de colinealidad entre puntos.
- Las líneas paralelas se transforman en líneas paralelas.
- Sean *P, Q, R* puntos de una línea, y *P', Q', R'* sus puntos correspondientes por la afinidad, entonces se cumple: $|PQ|/|PR| = |P'Q'|/|P'R'|$
- Las afinidades transforman el centroide de una figura en el centroide de la figura transformada.
- En general, las afinidades no respetan ángulos, distancias, ni áreas. Las áreas cumplen la siguiente relación: $S' = \det(A) \cdot S$, siendo *A* la matriz de la afinidad.

En la figura se puede observar el resultado de aplicar a un cuadrado distintos tipos de transformación [131]:

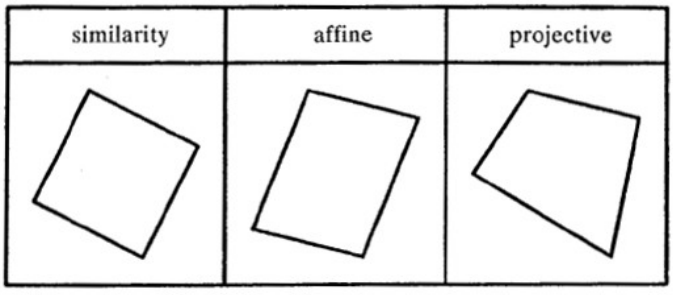


Figura 4.5. Transformaciones de similaridad, afín y proyectiva de un cuadrado.

Cuando la transformación es de similaridad, la matriz *H* normalizada toma la forma:

$$H = \begin{bmatrix} k\cos(\alpha) & -ksen(\alpha) & t_x \\ ksen(\alpha) & k\cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

con α ángulo de rotación, *k* factor de escala y $t = \left[t_x, t_y\right]^t$ el vector de traslación, y si es una afinidad:

$$H = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

o expresada en forma de bloques

$$H = \begin{bmatrix} A & t \\ 0^t & 1 \end{bmatrix}$$

Como se advierte en la matriz *H*, las transformaciones por similaridad y las afinidades tienen cuatro y seis grados de libertad, respectivamente.

Si se trabaja en $\mathbb{R}^2$ la expresión matricial de las afinidades adopta la forma:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \cdot \begin{bmatrix} x \\ y \end{bmatrix} + T$$

es decir

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Se demuestra que las transformaciones afines se pueden expresar como una composición de un escalado, un cizallado, una rotación y una traslación (las afinidades contienen a las transformaciones por similaridad). Por tanto:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A_r A_{sh} A_s \cdot \begin{bmatrix} x \\ y \end{bmatrix} + T$$

donde

$A_r = \begin{bmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{bmatrix}$ matriz de rotación (rotate) (θ positiva en sentido antihorario)

$A_{sh} = \begin{bmatrix} 1 & h_x \\ 0 & 1 \end{bmatrix}$ matriz de cizallado (shear) paralelo a eje x

$A_s = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$ matriz de escalado (scale) (si el escalado es homogéneo $S_x = S_y$)

$T = \begin{bmatrix} a & b \end{bmatrix}^t$ matriz columna de traslación (traslate)

En vez de una matriz de cizallado paralela al eje x, se puede utilizar una matriz de cizallado paralela al eje y:

$$A_{sh} = \begin{bmatrix} 1 & 0 \\ h_y & 1 \end{bmatrix}$$

Notar que la suma del número de coeficientes de las matrices de rotación, cizallado, escalado y traslación coincide con el número de grados de libertad de una afinidad.

Si los puntos del plano se representan como puntos del plano complejo $\mathbb{C}$, las afinidades tienen la siguiente expresión [77]:

$$z' = az + bz^* + c \quad a,b,c \in \mathbb{C}$$

donde $z = x + yi$, z^* es su complejo conjugado y z' es su transformado por la afinidad.

Se puede pasar fácilmente de la expresión compleja de la transformación a la expresión matricial, y viceversa: llamando $a = (a_x, a_y)$, $b = (b_x, b_y)$, $c = (c_x, c_y)$

$$z' = x' + y'i = (a_x + a_y i)(x + yi) + (b_x + b_y i)(x - yi) + (c_x + c_y i) =$$

$$= (a_x x + a_y xi + a_x yi - a_y y) + (b_x x + b_y xi - b_x yi + b_y y) + (c_x + c_y i) =$$

$$= (a_x x - a_y y + b_x x + b_y y + c_x) + (a_y x + a_x y + b_y x - b_x y + c_y) i =$$

$$= \left[(a_x + b_x) x + (-a_y + b_y) y + c_x \right] + \left[(a_y + b_y) x + (a_x - b_x) y + c_y \right] i$$

de donde se obtiene el sistema:

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ b_x \\ b_y \end{bmatrix}$$

El determinante de la matriz 4x4 es -2, luego el sistema es compatible determinado; así pues, se puede pasar de la expresión compleja a la matricial, y viceversa.

4.2 Invariantes de las transformaciones afines

El concepto de invariante de un grupo de transformación (geométrica) es fundamental en visión por computador, pues permite caracterizar las imágenes con independencia de las condiciones (perspectiva, iluminación, etc) bajo las cuales se perciben. En este Proyecto nos ocupamos únicamente de transformaciones debidas a cambios en el punto de vista, es decir, a efectos de perspectiva. Nos interesan, por tanto, los invariantes proyectivos. Las afinidades, que, como se vio anteriormente, son un tipo particular de proyección, tienen su propio conjunto de invariantes, y en este Apartado nos vamos a ocupar de su estudio, dejando para el siguiente el análisis de los invariantes proyectivos.

Los invariantes a afinidades son fundamentalmente de dos tipos, diferenciales y algebraicos, o basados en momentos[11] [120] [119]:

1. Los invariantes afines diferenciales se aplican a figuras, y se calculan a partir de las derivadas de sus contornos. Dos ejemplos son la longitud de arco afín invariante y la curvatura afín (ver Apéndice 5). El problema es que estos invariantes dependen de derivadas de orden elevado (segundo y cuarto orden la longitud de arco y la curvatura afines, respectivamente), lo que hace que sean muy sensibles al ruido, aparte de que el cálculo de las derivadas a partir de los píxeles de las imágenes (derivación digital) no es sencillo [38]; por esta razón, en la práctica no se usan. Algunos investigadores han empleado como invariantes ciertas magnitudes deducidas de los invariantes afines diferenciales, pero siguen mostrando sensibilidad al ruido [8]. Para evitar este problema, Arbter [15] introdujo una nuevo parámetro, el área encerrada, que tiene dos ventajas: 1) sólo depende de las primeras derivadas 2) se transforma linealmente al aplicar una afinidad a la figura. Arbter se basó en este parámetro para construir los descriptores de Fourier independientes a transformaciones afines [15].

2. Los invariantes afín algebraicos, denominados momentos invariantes a afinidades (Affine Moment Invariants, AMIs) se fundamentan en el concepto de momento de una imagen (de ahí que, para poder aplicar estos invariantes a figuras definidas por su contorno, sea necesario previamente asociar una imagen

[11] En realidad, esta división es cierta para todos los invariantes de la visión por computador. Existe otra clasificación: locales y globales. Los locales se calculan a partir de la vecindad de un punto, y los globales requieren para su cálculo todos los puntos de la figura.

a cada figura). En la práctica dan muy buenos resultados, y son empleados en este Proyecto; el único inconveniente que presentan es que, al calcularse a partir de la totalidad de puntos de la imagen, no dan buenos resultados cuando hay oclusiones. El Apartado 3.3. se dedica a su descripción.

En la literatura también se encuentran algunas técnicas de reconocimiento de patrones afín invariantes que no es posible clasificar dentro de los dos tipos anteriores; por ejemplo, en [8] es descrito un método que combina componentes principales y una transformación wavelet diádica (los componentes principales sirven para eliminar las distorsiones afines de las figuras, tras lo cual se clasifican con la transformación wavelet), en [95], [1], [2] se presenta un método curvature scale-space (CSS) de detección de afinidades.

4.3. Momentos invariantes a afinidades (AMIs).

El método de los Momentos Invariantes a Afinidades (Affine Moment Invariants, AMIs) constituye uno de los mejores procedimientos para detectar la existencia de afinidades. Los AMIs son un conjunto de valores que se calculan a partir de las imágenes. Tienen la característica esencial de ser independientes a las transformaciones afines; esto les confiere el poder de hacer clasificaciones por afinidad.

La construcción de los AMIs se basa en la Teoría de los Invariantes Algebraicos. Existen varios procedimientos para obtenerlos: método de grafos, de normalización, basado en la ecuación de Cayley-Aronhold, etc. En este Apartado se explica brevemente el cálculo de los invariantes por el método de grafos (los resultados de este Apartado se encuentran en [44]).

Los momentos invariantes de una imagen se calculan a partir de los momentos de la imagen, los cuales se definen de la siguiente forma:

1. Momento general $M_{pq}^{(f)}$ de una imagen $f(x,y)$ con $p,q \geq 0$, $p,q \in \mathbb{N}$

$$M_{pq}^{(f)} = \iint_D p_{pq}(x,y)\, f(x,y)\, dxdy$$

con $p_{pq}(x,y)$ función polinomial. $r = p+q$ es el orden del momento.

2. Momento geométrico $m_{pq}^{(f)}$ de una imagen $f(x,y)$ con $p,q \geq 0$, $p,q \in \mathbb{N}$. Son los momentos generales con $p_{pq}(x,y) = x^p y^q$ y $D = \mathbb{R}^2$; es decir

$$m_{pq}^{(f)} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y)\, dxdy$$

Como casos particulares:

m_{00}: "área" de la imagen

$\frac{m_{10}}{m_{00}}$: componente *x* del centroide de la imagen

$\frac{m_{01}}{m_{00}}$: componente *y* del centroide de la imagen

3. Momento central $\mu_{pq}^{(f)}$ de una imagen $f(x,y)$ con $p,q \geq 0$, $p,q \in \mathbb{N}$

$$\eta_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - x_c)^p (y - y_c)^q \, f(x,y) \, dxdy$$

con

$$x_c = \frac{m_{10}}{m_{00}}$$

$$y_c = \frac{m_{01}}{m_{00}}$$

Los momentos centrales son invariantes a traslaciones.

A partir de ahora, por sencillez, se prescindirá del superíndice (f).

Los momentos invariantes a afinidades se obtienen a partir del siguiente funcional:

$$I(f) = \int_{-\infty}^{+\infty} \overset{2r}{\ldots\ldots} \int_{-\infty}^{+\infty} \prod_{k,j=1}^{r} C_{kj}^{n_{kj}} \prod_{i=1}^{r} f(x_i, y_i) \, dx_i dy_i$$

donde

$f(x,y)$: imagen

$C_{kj} = \bar{x}_k \bar{y}_j - \bar{x}_j \bar{y}_k$

siendo

$\bar{x} = x - x_c$

$\bar{y} = y - y_c$

Se cumple: $C_{jk} = -C_{kj}$ y $C_{kk} = 0$. $I(f)$ depende de r y $n_{kj} \in \mathbb{N} \cup \{0\}$ $k, j \in \{1,\ldots,r\}$, que son parámetros a elegir.

Si se aplica una transformación afín de matriz *A* a la imagen (se está suponiendo que no hay traslación; si la hubiera, bastaría con tomar el centroide de la imagen como origen del sistema coordenado), y teniendo en cuenta que

$$\begin{bmatrix} \bar{x}' \\ \bar{y}' \end{bmatrix} = \begin{bmatrix} x' - x_c' \\ y' - y_c' \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x_c' \\ y_c' \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} - A\begin{bmatrix} x_c \\ y_c \end{bmatrix} = A\begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix} = A\begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix}$$

entonces:

$$C_{kj}' = \left\| \begin{bmatrix} \bar{x}_k' & \bar{y}_k' \\ \bar{x}_j' & \bar{y}_j' \end{bmatrix} \right\| = \left\| \begin{bmatrix} \bar{x}_k' & \bar{x}_j' \\ \bar{y}_k' & \bar{y}_j' \end{bmatrix}^t \right\| = \left\| \left(A \begin{bmatrix} \bar{x}_k & \bar{x}_j \\ \bar{y}_k & \bar{y}_j \end{bmatrix} \right)^t \right\| = \left\| \begin{bmatrix} \bar{x}_k & \bar{x}_j \\ \bar{y}_k & \bar{y}_j \end{bmatrix}^t A^t \right\| =$$

$$= \left\| \begin{bmatrix} \bar{x}_k & \bar{x}_j \\ \bar{y}_k & \bar{y}_j \end{bmatrix}^t \right\| \cdot \left\| A^t \right\| = \left\| A^t \right\| \cdot \left\| \begin{bmatrix} \bar{x}_k & \bar{x}_j \\ \bar{y}_k & \bar{y}_j \end{bmatrix}^t \right\| = \left\| A \right\| \cdot \left\| \begin{bmatrix} \bar{x}_k & \bar{y}_k \\ \bar{x}_j & \bar{y}_j \end{bmatrix} \right\| = \left\| A \right\| C_{kj}$$

donde $\|\cdot\|$ significa determinante.

Por lo tanto

$$I(f)' = \int_{-\infty}^{+\infty} \overset{2r}{\ldots\ldots} \int_{-\infty}^{+\infty} \prod_{k,j=1}^{r} C'^{n_{kj}}_{kj} \prod_{i=1}^{r} f'(x'_i, y'_i)\, dx'_i\, dy'_i =$$

$$\int_{-\infty}^{+\infty} \overset{2r}{\ldots\ldots} \int_{-\infty}^{+\infty} \left\| A \right\|^{\sum_{k,j=1}^{r} n_{kj}} \prod_{k,j=1}^{r} C_{kj}^{n_{kj}} \prod_{i=1}^{r} f(x_i, y_i) \left\| I \right\| dx_i dy_i =$$

con

$$I = \begin{bmatrix} \dfrac{\partial x'_1}{\partial x_1} & \dfrac{\partial x'_1}{\partial y_1} & \dfrac{\partial x'_1}{\partial x_2} & \dfrac{\partial x'_1}{\partial y_2} & \ldots\ldots \\ \dfrac{\partial y'_1}{\partial x_1} & \dfrac{\partial y'_1}{\partial y_1} & \dfrac{\partial y'_1}{\partial x_2} & \dfrac{\partial y'_1}{\partial y_2} & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \end{bmatrix}$$

La matriz jacobiana *I* es una matriz diagonal por bloques, siendo cada uno de sus bloques de tamaño *2x2*, pues

$$\frac{\partial x'_i}{\partial x_j} = \frac{\partial y'_i}{\partial x_j} = \frac{\partial x'_i}{\partial y_j} = \frac{\partial y'_i}{\partial y_j} = 0 \quad i \neq j$$

entonces, teniendo en cuenta las propiedades de los determinantes, se cumple:

$$\|I\| = \|A\|^r$$

Por lo tanto, llamando $\omega = \sum_{k,j=1}^{r} n_{kj}$ se tiene:

$$I(f)' = \|A\|^{\omega} \int_{-\infty}^{+\infty} \overset{2r}{\ldots\ldots} \int_{-\infty}^{+\infty} \prod_{k,j=1}^{r} C_{kj}^{n_{kj}} \prod_{i=1}^{r} f(x_i, y_i) \left|\|A\|^r\right| dx_i dy_i =$$

$$= \|A\|^{\omega} \left|\|A\|^r\right| \int_{-\infty}^{+\infty} \overset{2r}{\ldots\ldots} \int_{-\infty}^{+\infty} \prod_{k,j=1}^{r} C_{kj}^{n_{kj}} \prod_{i=1}^{r} f(x_i, y_i)\, dx_i dy_i = \|A\|^{\omega} \left|\|A\|^r\right| I(f)$$

Además, para la misma transformación afín:

$$\mu'_{00} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f'(x', y') dx' dy' = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) \|\|A\|\| dx dy = \|\|A\|\| \mu_{00}$$

Suponiendo $\|A\| > 0$, entonces tenemos:

$$\left(\frac{I(f)}{\mu_{00}^{\omega+r}}\right)' = \frac{I(f)}{\mu_{00}^{\omega+r}}$$

que no depende de la afinidad. Por lo tanto

$$\frac{I(f)}{\mu_{00}^{\omega+r}}$$

es un invariante a afinidades. *r* y ω son el *grado* y el *peso* del invariante, respectivamente. Si ω es impar y $\|A\| < 0$, en la relación anterior aparece un factor -1. Por simplicidad, en el resto de la discusión se supondrá $\|A\| > 0$.

Se demuestra que $I(f)$ consiste en un sumatorio de términos, cada uno de los cuales es un producto de momentos (multiplicados por un coeficiente). El orden máximo de los momentos que componen $I(f)$ se llama *orden o* de $I(f)$ y, por tanto, de $I(f)/\mu_{00}^{\omega+r}$. Se cumple $o \le \omega$. Se denomina estructura de un invariante a un vector de enteros $s = (k_2, \ldots, k_o)$, siendo k_i el número de momentos de orden *i* que aparecen en cada término (cada término tiene la misma estructura). Se ha de tener en cuenta que: a) *s* comienza en k_2, pues se demuestra que $k_0 = k_1 = 0$; b) si en un término aparece un momento de orden *i* elevado a la potencia *n*-ésima, se ha de contar *n* veces.

Eligiendo r y $n_{jk}\ \ j,k \in \{1,...,r\}$ se obtienen distintos invariantes. Por ejemplo, si $r=2$, $n_{12}=2$, $n_{11}=n_{22}=n_{21}=0$ (es decir, $\omega = n_{12}=2$), entonces

$$I(f)=\int_{-\infty}^{+\infty}{}^{4}.....\int_{-\infty}^{+\infty}\left(\bar{x}_1\bar{y}_2-\bar{x}_2\bar{y}_1\right)^2 f(x_1,y_1)\,f(x_2,y_2)\,dx_1dy_1dx_2dy_2=2\left(\mu_{20}\mu_{02}-\mu_{11}^2\right)$$

Luego, prescindiendo del factor 2, se obtiene

$$I_1=\frac{\mu_{20}\mu_{02}-\mu_{11}^2}{\mu_{00}^4}$$

que es un invariante con estructura (2) y grado 2.

Otro ejemplo: si $r=3$, $n_{12}=n_{13}=2$ (los demás exponentes cero) ($\omega=4$), entonces

$$I(f)=\int_{-\infty}^{+\infty}{}^{6}.....\int_{-\infty}^{+\infty}\left(\bar{x}_1\bar{y}_2-\bar{x}_2\bar{y}_1\right)^2\left(\bar{x}_1\bar{y}_3-\bar{x}_3\bar{y}_1\right)^2 f(x_1,y_1)\,f(x_2,y_2)\,f(x_3,y_3)\,dx_1dy_1dx_2dy_2dx_3dy_3=$$

$$=\mu_{20}^2\mu_{04}-4\mu_{20}\mu_{11}\mu_{13}+2\mu_{20}\mu_{02}\mu_{22}+4\mu_{11}^2\mu_{22}-4\mu_{11}\mu_{02}\mu_{31}+\mu_{02}^2\mu_{40}$$

A los invariantes $I(f)/\mu_{00}^{\omega+r}$ se les puede representar mediante un grafo construido de la siguiente forma: el grafo consta de *r* nodos, y entre los nodos *j* y *k* hay n_{jk} arcos, $j,k \in \{1,...,r\}$. En consecuencia, el grafo consta en total de ω arcos; es decir, el número de arcos del grafo coincide con el peso del invariante. Además, se cumple:

1. El número de arcos que sale de cada nodo es igual al orden de los momentos en cuyo cálculo participa ese nodo

2. r, es decir, el grado del invariante, coincide con el número de momentos que se multiplican en cada término.

Los dos invariantes anteriores puestos como ejemplo tienen asociados los siguientes grafos:

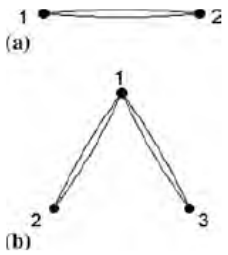


Figura 4.6. a) grafo de $r=2$, $n_{12}=2$, $n_{11}=n_{22}=n_{21}=0$
b) grafo de $r=3$, $n_{12}=n_{13}=2$ (los demás exponentes cero)

El cálculo de los momentos invariantes por el método de grafos consiste en fijar un peso ω y obtener todos los grafos conexos del tipo antes descrito, que tengan al menos dos nodos y cuyo número total de arcos sea ω. Se denota G_ω al conjunto de grafos que cumple lo anterior; cada uno de sus elementos representa un invariante.

Aunque el método de los grafos es fácil de implementar, sin embargo tiene un problema: entre los grafos de G_ω (o lo que es lo mismo, entre los invariantes que representan) existen numerosas relaciones de dependencia, lo cual, en la práctica, supone una disminución de la eficacia del uso de los invariantes.

Se dice que un invariante es reducible si es nulo, es idéntico a otro invariante, o se puede obtener a partir de otros mediante productos o combinaciones lineales. Existen algunas técnicas para detectar y eliminar los invariantes reducibles. Los invariantes que quedan en G_ω tras eliminar los reducibles se denominan irreducibles. Por último, entre éstos es necesario buscar aquellos que tengan independencia polinomial.

Por ejemplo, los siguientes diez AMIs forman un conjunto de invariantes irreducibles y con independencia polinomial:

$$I_1 = (\mu_{20}\mu_{02} - \mu_{11}^2)/\mu_{00}^4$$

$$I_2 = (-\mu_{30}^2\mu_{03}^2 + 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} - 4\mu_{30}\mu_{12}^3 - 4\mu_{21}^3\mu_{03} + 3\mu_{21}^2\mu_{12}^2)/\mu_{00}^{10}$$

$$I_3 = (\mu_{20}\mu_{21}\mu_{03} - \mu_{20}\mu_{12}^2 - \mu_{11}\mu_{30}\mu_{03} + \mu_{11}\mu_{21}\mu_{12} + \mu_{02}\mu_{30}\mu_{12} - \mu_{02}\mu_{21}^2)/\mu_{00}^7$$

$$I_4 = (-\mu_{20}^3\mu_{03}^2 + 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 3\mu_{20}^2\mu_{02}\mu_{12}^2 - 6\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} - 6\mu_{20}\mu_{11}^2\mu_{12}^2 + 12\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 3\mu_{20}\mu_{02}^2\mu_{21}^2 + 2\mu_{11}^3\mu_{30}\mu_{03} + 6\mu_{11}^3\mu_{21}\mu_{12} - 6\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}^2\mu_{02}\mu_{21}^2 + 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} - \mu_{02}^3\mu_{30}^2)/\mu_{00}^{11}$$

$$
\begin{aligned}
I_5 = (&\mu_{20}^3\mu_{30}\mu_{03}^3 - 3\mu_{20}^3\mu_{21}\mu_{12}\mu_{03}^2 + 2\mu_{20}^3\mu_{12}^3\mu_{03} - 6\mu_{20}^2\mu_{11}\mu_{30}\mu_{12}\mu_{03}^2 \\
&+ 6\mu_{20}^2\mu_{11}\mu_{21}^2\mu_{03}^2 + 6\mu_{20}^2\mu_{11}\mu_{21}\mu_{12}^2\mu_{03} - 6\mu_{20}^2\mu_{11}\mu_{12}^4 \\
&+ 3\mu_{20}^2\mu_{02}\mu_{30}\mu_{12}^2\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}^2\mu_{12}\mu_{03} + 3\mu_{20}^2\mu_{02}\mu_{21}\mu_{12}^3 \\
&+ 12\mu_{20}\mu_{11}^2\mu_{30}\mu_{12}^2\mu_{03} - 24\mu_{20}\mu_{11}^2\mu_{21}^2\mu_{12}\mu_{03} + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{12}^3 \\
&- 12\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{12}^3 + 12\mu_{20}\mu_{11}\mu_{02}\mu_{21}^3\mu_{03} - 3\mu_{20}\mu_{02}^2\mu_{30}\mu_{21}^2\mu_{03} \\
&+ 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{21}\mu_{12}^2 - 3\mu_{20}\mu_{02}^2\mu_{21}^3\mu_{12} - 8\mu_{11}^3\mu_{30}\mu_{12}^3 + 8\mu_{11}^3\mu_{21}^3\mu_{03} \\
&- 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{21}^2\mu_{03} + 24\mu_{11}^2\mu_{02}\mu_{30}\mu_{21}\mu_{12}^2 - 12\mu_{11}^2\mu_{02}\mu_{21}^3\mu_{12} \\
&+ 6\mu_{11}\mu_{02}^2\mu_{30}^2\mu_{21}\mu_{03} - 6\mu_{11}\mu_{02}^2\mu_{30}^2\mu_{12}^2 - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21}^2\mu_{12} \\
&+ 6\mu_{11}\mu_{02}^2\mu_{21}^4 - \mu_{02}^3\mu_{30}^3\mu_{03} + 3\mu_{02}^3\mu_{30}^2\mu_{21}\mu_{12} - 2\mu_{02}^3\mu_{30}\mu_{21}^3)/\mu_{00}^{16}
\end{aligned}
$$

$$
I_6 = (\mu_{40}\mu_{04} - 4\mu_{31}\mu_{13} + 3\mu_{22}^2)/\mu_{00}^6
$$

$$
I_7 = (\mu_{40}\mu_{22}\mu_{04} - \mu_{40}\mu_{13}^2 - \mu_{31}^2\mu_{04} + 2\mu_{31}\mu_{22}\mu_{13} - \mu_{22}^3)/\mu_{00}^9
$$

$$
\begin{aligned}
I_8 = (&\mu_{20}^2\mu_{04} - 4\mu_{20}\mu_{11}\mu_{13} + 2\mu_{20}\mu_{02}\mu_{22} + 4\mu_{11}^2\mu_{22} \\
&- 4\mu_{11}\mu_{02}\mu_{31} + \mu_{02}^2\mu_{40})/\mu_{00}^7
\end{aligned}
$$

$$
\begin{aligned}
I_9 = (&\mu_{20}^2\mu_{22}\mu_{04} - \mu_{20}^2\mu_{13}^2 - 2\mu_{20}\mu_{11}\mu_{31}\mu_{04} + 2\mu_{20}\mu_{11}\mu_{22}\mu_{13} \\
&+ \mu_{20}\mu_{02}\mu_{40}\mu_{04} - 2\mu_{20}\mu_{02}\mu_{31}\mu_{13} + \mu_{20}\mu_{02}\mu_{22}^2 + 4\mu_{11}^2\mu_{31}\mu_{13} - 4\mu_{11}^2\mu_{22}^2 \\
&- 2\mu_{11}\mu_{02}\mu_{40}\mu_{13} + 2\mu_{11}\mu_{02}\mu_{31}\mu_{22} + \mu_{02}^2\mu_{40}\mu_{22} - \mu_{02}^2\mu_{31}^2)/\mu_{00}^{10}
\end{aligned}
$$

$$
\begin{aligned}
I_{19} = (&\mu_{20}\mu_{30}\mu_{12}\mu_{04} - \mu_{20}\mu_{30}\mu_{03}\mu_{13} - \mu_{20}\mu_{21}^2\mu_{04} + \mu_{20}\mu_{21}\mu_{12}\mu_{13} \\
&+ \mu_{20}\mu_{21}\mu_{03}\mu_{22} - \mu_{20}\mu_{12}^2\mu_{22} - 2\mu_{11}\mu_{30}\mu_{12}\mu_{13} + 2\mu_{11}\mu_{30}\mu_{03}\mu_{22} \\
&+ 2\mu_{11}\mu_{21}^2\mu_{13} - 2\mu_{11}\mu_{21}\mu_{12}\mu_{22} - 2\mu_{11}\mu_{21}\mu_{03}\mu_{31} + 2\mu_{11}\mu_{12}^2\mu_{31} \\
&+ \mu_{02}\mu_{30}\mu_{12}\mu_{22} - \mu_{02}\mu_{30}\mu_{03}\mu_{31} - \mu_{02}\mu_{21}^2\mu_{22} + \mu_{02}\mu_{21}\mu_{12}\mu_{31} \\
&+ \mu_{02}\mu_{21}\mu_{03}\mu_{40} - \mu_{02}\mu_{12}^2\mu_{40})/\mu_{00}^{10}
\end{aligned}
$$

En la dirección web http://zoi_zmije.utia.cas.cz/files/afinvs12irred_graphs.pdf se puede descargar un archivo pdf con los 1589 invariantes irreducibles con peso $\omega \leq 12$.

Los momentos invariantes a afinidad de orden elevado son sensibles al ruido, y costosos de implementar [134].

Para calcular los AMIs de una figura definida por su contorno *C* es necesario asociar una imagen a la figura; en la práctica, resulta conveniente escoger

$$f(x,y)=\begin{cases}1 & (x,y)\in \overline{C} \\ 0 & (x,y)\notin \overline{C}\end{cases}$$

donde $\overline{C}$ es el conjunto de puntos de la figura (se cumple $C\subset\overline{C}$), pues así se simplifica la expresión de η_{pq}

$$\eta_{pq}=\iint_C (x-x_c)^p (y-y_c)^q \, dxdy$$

Además, suponemos que el origen de coordenadas de la imagen coincide con el baricentro de la figura; entonces

$$\eta_{pq}=\iint_C x^p y^q dxdy$$

Por otro lado, en este Proyecto los contornos de las figuras se expresan como poligonales con vértices $p_i=(x_i,y_i)\ i\in\{0,...,n\}$ y $p_0=p_n$ (recordemos que el contorno es cerrado). Entonces, aplicando el teorema de Green, se puede demostrar que η_{pq} se calcula por la siguiente expresión [89]:

$$\eta_{pq}=\frac{1}{(p+q+2)(p+q+1)\binom{p+q}{p}}\sum_{i=1}^{n}(x_{i-1}y_i-x_iy_{i-1})\sum_{k=0}^{p}\sum_{l=0}^{q}\binom{k+l}{l}\binom{p+q-k-l}{q-l}x_i^k x_{i-1}^{p-k} y_i^l y_{i-1}^{q-l}$$

Debe notarse que ahora la expresión de η_{pq} sólo depende de los vértices del contorno y no de todos los puntos de la imagen.

4. 4. Invariantes a transformaciones proyectivas.

En el caso de las transformaciones proyectivas se pueden utilizar tres tipos de invariantes:

Invariantes proyectivos diferenciales. El más importante es la curvatura proyectiva, pero depende de derivadas de séptimo grado y, por lo tanto, es extremadamente sensible al ruido; lógicamente, no se utiliza. En [56] se puede encontrar una discusión sobre este tema.

Invariantes algebraicos proyectivos. En 2004, Suk y Flusser [128] presentaron dos invariantes proyectivos independientes basados en el concepto de momento de una imagen. A diferencia de sus homólogos afines, su construcción no se basa en la teoría de invariantes algebraicos, pues las proyectividades no son transformaciones lineales ni preservan los baricentros; otra diferencia sustancial que presentan estos invariantes es que se calculan a partir de momentos con índices enteros arbitrarios (no positivos, como en el caso afín):

$$m_{pq}^{(f)} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f\left(x,y\right) dxdy \qquad p,q \in \mathbb{Z}$$

Los dos invariantes algebraicos proyectivos independientes son los siguientes:

$$\begin{aligned} I_1 = 2^3 \sum_{i,j,k=0}^{\infty} & \sum_{\substack{i_1+i_2+i_3+i_4+i_5=i \\ i_1,i_2,i_3,i_4,i_5\geq 0}} \frac{i!(-1)^{i_2+i_4}}{i_1!i_2!i_3!i_4!i_5!} \\ & \sum_{\substack{j_1+j_2+j_3+j_4+j_5=j \\ j_1,j_2,j_3,j_4,j_5\geq 0}} \frac{j!(-1)^{j_2+j_4}}{j_1!j_2!j_3!j_4!j_5!} \\ & \sum_{\substack{k_1+k_2+k_3+k_4+k_5=k \\ k_1,k_2,k_3,k_4,k_5\geq 0}} \frac{k!(-1)^{k_2+k_4}}{k_1!k_2!k_3!k_4!k_5!} \\ & m_{-1-i+i_5+k_1+k_2+j_3+j_4,-1-k+k_3+i_1+i_4+j_2+j_5} \\ & m_{-1-j+j_5+i_1+i_2+k_3+k_4,-1-i+i_3+j_1+j_4+k_2+k_5} \\ & m_{-1-k+k_5+j_1+j_2+i_3+i_4,-1-j+j_3+i_2+i_5+k_1+k_4}. \end{aligned}$$

$$\begin{aligned} I_2 = 2^4 \sum_{i,j,k,\ell=0}^{\infty} & \sum_{\substack{i_1+i_2+i_3+i_4+i_5=i \\ i_1,i_2,i_3,i_4,i_5\geq 0}} \frac{i!(-1)^{i_2+i_4}}{i_1!i_2!i_3!i_4!i_5!} \\ & \sum_{\substack{j_1+j_2+j_3+j_4+j_5=j \\ j_1,j_2,j_3,j_4,j_5\geq 0}} \frac{j!(-1)^{j_2+j_4}}{j_1!j_2!j_3!j_4!j_5!} \\ & \sum_{\substack{k_1+k_2+k_3+k_4+k_5=k \\ k_1,k_2,k_3,k_4,k_5\geq 0}} \frac{k!(-1)^{k_2+k_4}}{k_1!k_2!k_3!k_4!k_5!} \\ & \sum_{\substack{\ell_1+\ell_2+\ell_3+\ell_4+\ell_5=\ell \\ \ell_1,\ell_2,\ell_3,\ell_4,\ell_5\geq 0}} \frac{\ell!(-1)^{\ell_2+\ell_4}}{\ell_1!\ell_2!\ell_3!\ell_4!\ell_5!} \\ & m_{-1-i+i_5+j_1+j_2+k_3+k_4,-1-j+j_3+i_1+i_4+k_2+k_5} \\ & m_{-1-j+j_5+i_1+i_2+\ell_3+\ell_4,-1-i+i_3+j_1+j_4+\ell_2+\ell_5} \\ & m_{-1-k+k_5+i_3+i_4+\ell_1+\ell_2,-1-\ell+\ell_3+i_2+i_5+k_1+k_4} \\ & m_{-1-\ell+\ell_5+j_3+j_4+k_1+k_2,-1-k+k_3+j_2+j_5+\ell_1+\ell_4}. \end{aligned}$$

El descubrimiento de estos invariantes supuso todo un logro (hasta entonces se había pensado que no existían); sin embargo, en la práctica no son útiles, ya que se expresan en forma de series infinitas y, aunque éstas se trunquen, el número de iteraciones necesario para su cálculo resulta excesivo. Un problema añadido es que no admiten oclusiones.

El invariante clásico de la geometría proyectiva plana es la razón doble de cinco puntos (Five Points Cross-Ratio Invariant) [129]. Dados cinco puntos, se pueden calcular dos razones dobles [16]:

$$CR(0;1,2,3,4) = \frac{V(0,1,2)V(0,3,4)}{V(0,1,4)V(0,2,3)} \qquad (1)$$

$$CR(1;0,2,3,4) = \frac{V(0,1,2)V(1,3,4)}{V(0,1,4)V(1,2,3)}$$

donde $V(i,j,k)$ es el área del triángulo definido por los puntos z_i, z_j y z_k. El valor $CR(0;1,2,3,4)$ representa el cociente entre el producto de las áreas no sombreadas y el producto de las áreas sombreadas:

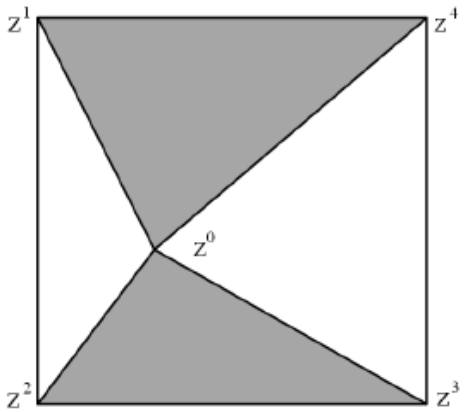


Figura 4.7. Áreas que intervienen en $CR(0;1,2,3,4)$

Los cross-ratios son invariantes proyectivos locales y carecen de unicidad (dentro de una misma imagen (o figura) existen infinidad de combinaciones de cinco puntos cuyos cross-ratios tienen el mismo valor; sin embargo, el conjunto de cross-ratios de todas las combinaciones de cinco puntos de una imagen (o figura) caracterizan ésta de forma unívoca [16].

El uso de razones dobles para reconocer proyectividades tiene una ventaja y una desventaja: la ventaja es que, gracias al carácter local de estos invariantes, tienen buen

comportamiento frente a oclusiones; la desventaja es que sólo se pueden aplicar si en las imágenes es posible reconocer grupos de puntos homólogos.

No vamos a entrar en detalles acerca de los métodos de detección de puntos homólogos de imágenes que existen (SIFT, detector de Harris de esquinas, etc); en este Proyecto sólo nos interesan aquellos puntos de los contornos de las figuras (en general, de curvas) que resultan homólogos en las transformaciones proyectivas; los vamos a denominar puntos *especiales*. En [55] se demuestra que, bajo homografías, las curvas tienen tres tipos de puntos especiales:

- Los puntos de bitangencia. Una bitangente de una curva es una recta que sólo tiene dos puntos de contacto con la curva; cada uno de estos puntos es un punto de bitangencia.

- Los puntos de inflexión, o puntos con la derivada segunda igual a cero.

- Los puntos en los que se anula la derivada de la función de curvatura afín unimodular (ver Apéndice 5).

Aunque en teoría todos estos puntos se pueden obtener derivando las curvas, en la práctica no resulta sencillo [75]. Existen formas alternativas de calcular puntos de bitangencia [28] [104]; una de ellas es a partir del cierre convexo[12] (convex hull), como se puede apreciar en la figura:

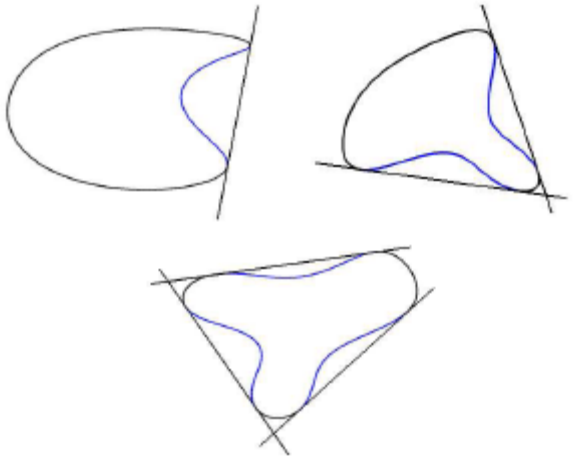


Figura 4.8. Unas curvas y sus bitangentes

El método del cierre convexo sólo se puede aplicar a transformaciones proyectivas cuasi-afines (las que transforman el cierre convexo de una curva en el cierre convexo de la transformada de la curva [57]) y que preservan la orientación (el determinante de su

[12] El cierre convexo de una curva es el menor conjunto convexo que contiene a dicha curva. Se ha de tener en cuenta que, por definición, en el espacio proyectivo los conjuntos convexos no tienen puntos del infinito; por lo tanto, el cierre convexo está formado únicamente por puntos propios.

matriz jacobiana es positivo), pues en este caso los puntos de bitangencia de una curva pertenecientes al convex hull se transforman en puntos de bitangencia de la transformada de la curva pertenecientes también al convex hull. Obviamente, el método del cierre convexo para detectar puntos de bitangencia no es exhaustivo (puede haber puntos de bitangencia dentro de las concavidades que no son detectados por el método)

4.5. Transformaciones proyectivas generales.

Todo lo que se ha visto en los apartados precedentes se refiere a proyecciones que transforman una figura plana *F* contenida en un plano β en otra figura (plana) *F'* perteneciente al plano de proyección β' (ver la Figura 4.1.); sin embargo, si β no es un plano, sino cualquier otro tipo de superficie, en general dejan de tener validez los invariantes proyectivos descritos hasta ahora. El problema consiste, entonces, en cómo identificar figuras homólogas, ya que no se pueden utilizar los invariantes proyectivos tradicionales.

En los últimos años ha habido un interés creciente en este tema y han aparecido diversas soluciones [113] [109] [69] [26]. Todos estos métodos tienen una mecánica de trabajo común:

1. Identifican ciertos puntos homólogos de las imágenes utilizando alguna característica distintiva (color, corners, etc)

2. Eligen la superficie (warp) que mejor se ajusta a la relación entre puntos homólogos, utilizando algún criterio de optimización. La superficie se suele representar en forma de producto tensorial de B-splines [113], elementos finitos [109], diferencias finitas [69], NURBS [26], etc; normalmente, se supone que es suave o suave a trozos.

3. Una vez construida la superficie, se puede utilizar para registrar los puntos homólogos que no fueron empleados en el cálculo de la superficie.

En general, estos métodos sólo funcionan bien cuando las imágenes presentan perspectiva afín; únicamente, el método Schwarps [72] considera proyectividades generales, pero la superficie se obtiene resolviendo un sistema de ecuaciones diferenciales parciales 2D, lo que requiere disponer de bastante información local bidimensional de las imágenes.

5. Clasificación de patrones proyectivo-invariantes mediante un perceptrón de variable compleja.

Como ya fue explicado en el Capítulo 0, el objetivo del Proyecto es resolver el siguiente problema: supongamos que se dispone de un conjunto de curvas, que representan los contornos de unas figuras, dibujadas en unas superficies planas orientadas aleatoriamente; a continuación, estas superficies se deforman sin rasgaduras, también de forma aleatoria; lo único que se exige es que al final las figuras no presenten oclusiones al proyectarse en el plano de proyección. La clasificación de estas curvas es un problema de clasificación proyectivo-invariante,

Este problema involucra transformadas generales, no afinidades ni homografías; por tanto, no se pueden aplicar para resolverlo los invariantes de los Apartados 4.2, 4.3 y 4.4; por otro lado, los métodos descritos en el Apartado 4.5 tampoco se pueden usar porque, o bien sólo son aplicables a afinidades o bien requieren mucha información local bidimensional de la cual no se dispone en este caso.

La solución descrita en este Proyecto consiste en combinar un detector de puntos dominantes de los contornos de las figuras con una red neuronal de variable compleja que tome como entradas dichos puntos y actúe como clasificador de las figuras. La razón de la elección de una red neuronal de variable compleja y no una real, que es lo usual, se debe a la mayor capacidad funcional de las primeras (ver Capítulo 3).

En el Apartado siguiente se definen los puntos dominantes de una curva y se describe un método para calcularlos. En el Apartado 5.2 se desarrolla el clasificador proyectivo-invariante propuesto para resolver el problema enunciado más arriba, y en el Apartado 5.3. se aplica a unos casos prácticos.

5.1. Los puntos dominantes de una curva.

Los puntos dominantes de una curva son aquellos que tienen un valor de curvatura elevado [4]. A estos puntos se les suele denominar vértices de la curva, por extensión del concepto de vértice de un polígono. La importancia de los puntos dominantes del contorno de una figura en el reconocimiento de ésta ya era señalado en 1954 en el trabajo pionero de Attneave [17]. Se distinguen dos tipos de métodos de reconocimiento de los puntos dominantes de una curva [4]:

1. Obtener los puntos dominantes directamente de la figura, utilizando, por ejemplo, métodos de detección de esquinas (corners).

2. Sustituir la curva por un polígono, calculado de manera que se cumplan ciertas restricciones. Entonces, se toman como puntos dominantes de la curva los vértices del polígono.

Di Ruberto y Morgera han desarrollado un método simple que no requiere derivación digital y resulta, sin embargo, poderoso y versátil. El método es del tipo 1 señalado más arriba e iterativo; en su etapa inicial requiere disponer de cuatro vértices de la curva, que siempre existen en virtud del Teorema de los cuatro vértices [33].

Antes de explicar cómo se localizan en la curva los cuatro vértices iniciales es necesario definir el concepto de firma de una curva (signature of curve). Una firma de una curva 2D es una función 1D que describe alguna característica de ella, y se calcula a partir de las coordenadas de sus puntos [47] [144]. Por lo tanto, una firma de una curva retiene cierta información perteneciente a ésta, aunque, en general, sea insuficiente para su reconstrucción. Ejemplos de firmas son la distancia de los puntos al centroide, el ángulo de la tangente en cada punto, el ángulo acumulado de la tangente en cada punto, la curvatura, el área del radio vector, etc. En nuestro caso, para el cálculo de los cuatro vértices iniciales, se usa la firma *distancia de los puntos de la curva a su centroide*.

El centroide se calcula por la siguiente expresión:

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad y_c = \frac{1}{n}\sum_{i=1}^{n} y_i$$

donde x_i, y_i son las coordenadas del *i*-ésimo punto del contorno, que se ha supuesto que consta de *n* puntos. La firma *distancia de los puntos de la curva a su centroide* es la función $s(i) = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \quad i \in \{1,...,n\}$.

Los cuatro vértices iniciales del método de Di Ruberto y Morgera son los puntos de la curva que corresponden a los dos puntos máximos relativos con mayor ordenada y a los puntos mínimos relativos con menor ordenada de la firma *distancia de los puntos de la curva a su centroide*. Por ejemplo, en la figura 4.6. se muestra un contorno, su firma *distancia de los puntos de la curva a su centroide* y los cuatro vértices iniciales:

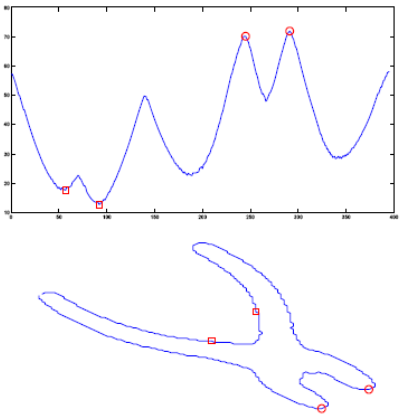


Figura 5.1. Un contorno, su firma *distancia de los puntos de la curva a su centroide* y los cuatro vértices iniciales

Los puntos dominantes de la curva se calculan a partir de estos cuatro puntos mediante el siguiente proceso iterativo: en cada etapa se dispone de una lista ordenada de puntos dominantes $D = \{d_i \ i = 1,...,m\}$, con *m=4* en la etapa inicial. Para cada par de puntos $d_i d_{i+1}$, con $i+1 \equiv i+1 \pmod{m}$, y para todo punto *p* de la curva situado entre ambos, se calcula la distancia entre este punto y la recta que une d_i y d_{i+1}, mediante la expresión:

$$T_{d_i d_{i+1}}(p) = \frac{\left|(p_x - x_i)(y_{i+1} - y_i) - (p_y - y_i)(x_{i+1} - x_i)\right|}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}$$

donde $T_{d_i d_{i+1}}(p)$ denota la distancia entre p y la recta que une d_i y d_{i+1}, con $d_i = (x_i, y_i)$, $d_{i+1} = (x_{i+1}, y_{i+1})$ y $p = (p_x, p_y)$.

Si sumamos todas estas distancias se obtiene una distancia global $T_{d_i d_{i+1}}$ asociada al segmento $d_i d_{i+1}$:

$$T_{d_i d_{i+1}} = \sum_p T_{d_i d_{i+1}}(p)$$

$T_{d_i d_{i+1}}$ se compara con el valor $s_i = l \cdot n_i$, donde l es un parámetro del algoritmo y n_i es el número de puntos de la curva situados entre los puntos d_i y d_{i+1}. Si $T_{d_i d_{i+1}} > s_i$, el punto p con mayor valor $T_{d_i d_{i+1}}(p)$ se considera nuevo punto dominante y se añade a la lista D entre d_i y d_{i+1}; si $T_{d_i d_{i+1}} \leq s_i$, ningún punto de la curva situado entre d_i y d_{i+1} se toma como punto dominante.

El algoritmo se detiene cuando en dos iteraciones consecutivas no se añaden nuevos puntos dominantes.

Una vez finalizado el algoritmo es necesario refinar la lista D obtenida para evitar que existan puntos muy próximos entre si. El procedimiento, que es iterativo, consiste en sustituir las parejas de puntos (d_i, d_{i+1}) tales que $dist(d_i, d_{i+1}) < \tau$ por su punto medio, donde *dist* significa distancia entre dos puntos y τ es un coeficiente propio del método de refinado.

El número de puntos dominantes que se obtiene por este método depende del parámetro l: cuanto mayor es este valor, menor es el número de vértices que se obtienen.

El procedimiento de cálculo de puntos dominantes usado en este Proyecto es básicamente el mismo que el de Di Ruberto y Morera, aunque se ha hecho una simplificación en la forma de añadir nuevos vértices: en cada iteración, para añadir un vértice que esté situado entre los vértices d_i y d_{i+1}, se escogen aquellos puntos de la curva cuya distancia a la recta que une d_i y d_{i+1} sea mayor que l, y se selecciona el que está a mayor distancia de la recta; éste es el vértice buscado.

5.2. El clasificador proyectivo-invariante basado en redes neuronales de variable compleja.

Existen dos formas de utilizar las redes neuronales para reconocer formas 2D descritas por su contorno, y cuya apariencia varía debido a cambios en el punto de vista:

1. Asociar a cada contorno unos coeficientes proyectivo-invariantes. La red neuronal toma como entradas estos coeficientes y hace la clasificación.

2. Los puntos de los contornos de las figuras son usados directamente como entradas de la red neuronal. El problema es que, en este caso, la red debería tener un gran número de entradas y sería, por lo tanto, muy compleja. La solución es sustituir el contorno de cada figura por una aproximación lo suficientemente significativa que logre mantener la bondad de la clasificación.

La mayor parte de las soluciones que se encuentran en la literatura son del tipo descrito en 1, pero se limitan a resolver problemas afín-invariantes [7], [142], [143], [86], [147], [20], [34]; además, utilizan redes neuronales de variable real, cuyos valores de entrada son el resultado de aplicar a las figuras alguno de los invariantes afines descritos en los Apartados 4.2 y 4.3.

Las soluciones del tipo 2. son mucho más escasas; en el caso de deformaciones afines se pueden citar [19], [41] (el objetivo de ambos no es clasificar las figuras, sino estimar los coeficientes de las afinidades) y [84], que utiliza una red Hopfield; en el caso de transformaciones afines se han encontrado tres trabajos [105], [83] y [85], los cuales emplean redes Hopfield, y se limitan a homografías. En todos estos artículos las redes neuronales empleadas son de variable real.

En este Proyecto se propone un método de clasificación proyectivo-invariante de figuras planas que consta de dos partes:

1. Primeramente, las figuras, expresadas como figuras del plano complejo, son pasadas por un preproceso que genera una versión simplificada de las mismas. El sistema de simplificación consiste en un procedimiento de detección de los puntos dominantes de los contornos de las figuras.

2. Una red neuronal de variable compleja clasifica las figuras "resumidas". Los patrones de entrenamiento están formados por parejas *figura "resumida"-etiqueta de clase*. La red neuronal empleada es el perceptrón de variable compleja, de dos capas, con función de activación split real-compleja y entrenamiento por retropropagación compleja, desarrollado por Nitta [59] y descrito en el Capítulo 3.

En el buen funcionamiento del método propuesto es clave la capacidad de las redes neuronales de variable compleja de modelar transformaciones geométricas. En el trabajo original de Nitta [101] se muestra como este tipo de redes son capaces de aprender a reconocer traslaciones y rotaciones; en trabajos más recientes [102], este autor ha extendido sus investigaciones a afinidades. El método desarrollado en este Proyecto puede verse como una aplicación práctica de estos estudios.

Es importante señalar que las salidas de una red neuronal de variable compleja no se pueden, en general, interpretar como una probabilidad (una probabilidad no es un número complejo). Por lo tanto, en el entrenamiento de la red que clasifica las figuras se va a prescindir de cuestiones estadísticas (ver referencia a pié 6 de Capítulo 2).

Una dificultad que surge a la hora de poner en práctica el método de simplificación es que, en principio, el número de puntos dominantes que se extraen de cada figura no tiene porque ser siempre el mismo. Esto es un problema, ya que la topología de la red neuronal empleada en la clasificación no puede variar con cada contorno. La solución adoptada es:

1. Se fija el número de entradas de la red neuronal *Nent*.

2. El parámetro *l* del procedimiento de extracción de puntos dominantes se ajusta a cada contorno, de manera que la cantidad de puntos dominantes extraída siempre supere a *Nent*. Recordemos que cuanto menor sea *l*, mayor cantidad de puntos dominantes se obtiene. El ajuste es iterativo: se parte de un valor $l^0 = \text{longitud contorno}/\eta^0$, siendo η^0 un parámetro a elegir, y se define un parámetro Δl; en cada iteración, se compara el número de puntos dominantes *nv* con *Nent*: si $nv < Nent$ se calcula $l = l - \Delta l$, si $nv \geq Nent$ el ajuste se detiene.

Por último, antes de ver la aplicación del método desarrollado en este Proyecto en casos concretos, se presenta un listado de sus ventajas:

- Su concepto es sencillo.

- No requiere derivación digital.

- Se puede aplicar a cualquier tipo de figura, incluidos polígonos y figuras no convexas.

Las desventajas de este procedimiento derivan del uso de la red neuronal como clasificador, y son las comunes a cualquier método que utilice redes neuronales; esto es: lentitud y dificultad de entrenamiento de la red. La lentitud de entrenamiento aconseja el uso off-line del procedimiento. Respecto a la dificultad de entrenamiento, cuanto mayor es el número de neuronas mayor es el número de parámetros de la red y más compleja es la función de error; por lo tanto, encontrar el factor de aprendizaje óptimo en las aplicaciones prácticas requiere destreza, es más un arte que un procedimiento sistemático.

5.3. Evaluación del desempeño del clasificador descrito en el Apartado 5.2.

Con el fin de evaluar la bondad del método descrito en el Apartado anterior, se ha empleado en resolver tres problemas de clasificación: una clasificación afin-invariante, una clasificación proyectivo-invariante, restringida a homografías, y una clasificación proyectiva-invariante, que trabaja con transformaciones proyectivas más generales. En todos los casos el problema consiste en clasificar un conjunto de curvas, obtenidas transformando el contorno de dos patrones[13]: el patrón Perro

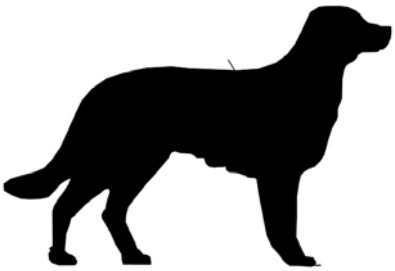


Figura 5.2. Patrón Perro

y el patrón Gato:

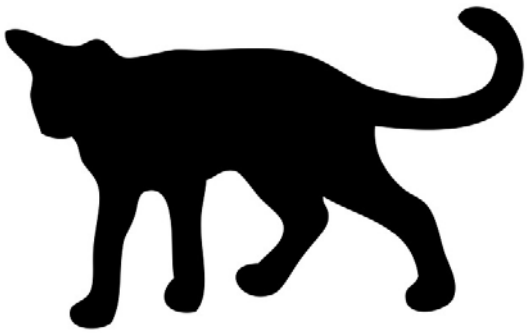


Figura 5.3. Patrón Gato

La mitad de las curvas corresponden a Perro, y la otra mitad a Gato. El contorno de los patrones[14] se ha extrae utilizando la librería gráfica OpenCV, se normalizan de manera

[13] En esencia, el problema es equivalente a clasificar figuras definidas por sus contornos.
[14] Estas siluetas se pueden obtener en las direcciones web:
Patrón Perro: http://pixabay.com/en/dog-lab-drawing-vector-sketch-163659/

que sus puntos pertenezcan al cuadrado $[0,1]^2$ y se suavizan con un filtro de la media con ventana de tamaño 7. Los puntos de todas las curvas se representan como números complejos.

5.3.1. Evaluación del desempeño del clasificador con transformaciones afines

En este apartado se estudia el comportamiento del clasificador cuando la deformación de las figuras se debe a afinidades. Se siguen los siguientes pasos:

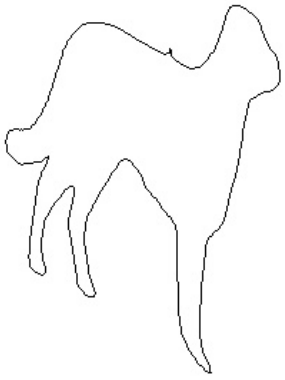
1. Se obtienen 1130 curvas aplicando transformaciones afines a los dos patrones originales; los coeficientes matriciales de estas transformaciones se eligen dentro de los intervalos:

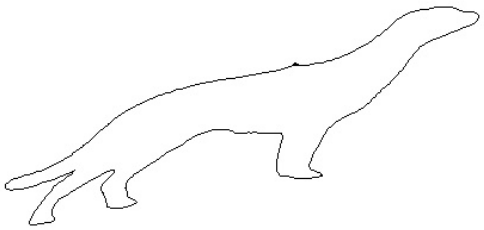
$$a_{11} \in (0.6,1.4) \qquad a_{12} \in (-0.4,0.4) \qquad a_{13} \in (-0.4,0.4)$$

$$a_{21} \in (-0.4,0.4) \qquad a_{22} \in (0.6,1.4) \qquad a_{23} \in (-0.4,0.4)$$

$$a_{31} = 0 \qquad a_{32} = 0 \qquad a_{33} = 1$$

   Una muestra de estas curvas; por ejemplo, de Perro





Patrón Gato: http://pixabay.com/en/cat-silhouette-black-animal-pet-163559/

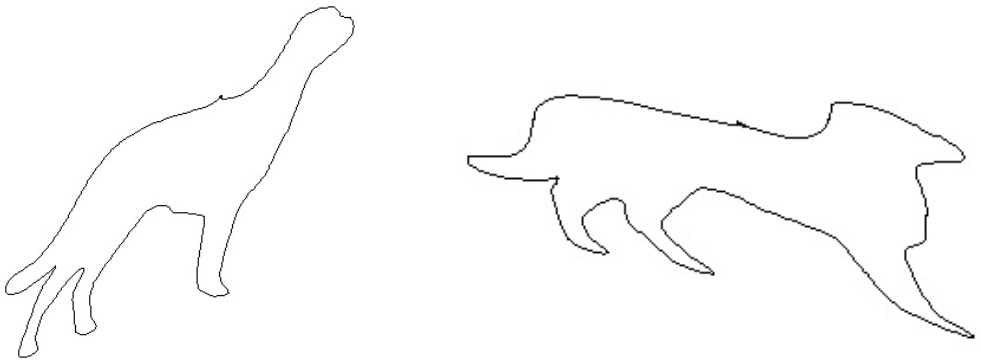


Figura 5.4. Una muestra de las curvas de la clase Perro obtenidas aplicando afinidades.

y de Gato:

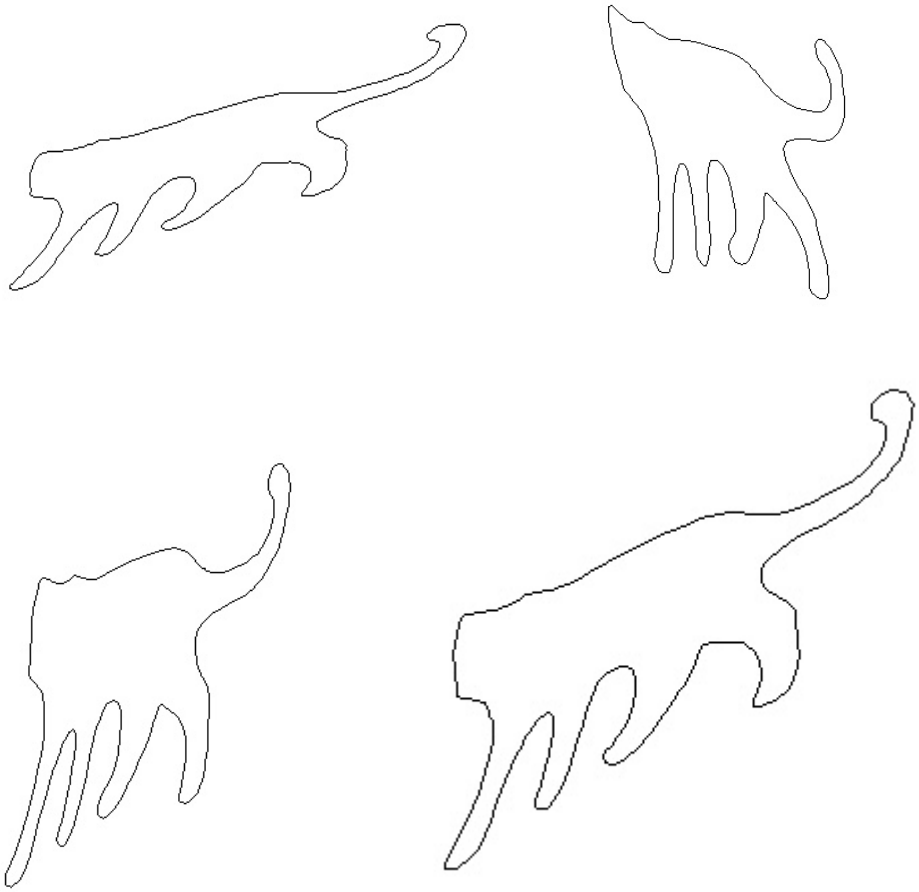


Figura 5.5. Una muestra de las curvas de la clase Gato obtenidas aplicando afinidades.

2. Del conjunto de curvas se escogen 130 para entrenar la red y el resto para validar.

3. Se detectan los puntos dominantes de las curvas por el método de Di Ruberto y Morgera. Por ejemplo, los puntos dominantes de la primera curva de la muestra de Perro son:

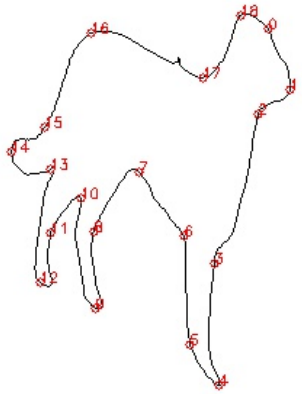


Figura 5.6. Puntos dominantes de una curva de la clase Perro obtenida aplicando afinidades.

y los de la primera curva de la muestra de la clase Gato:

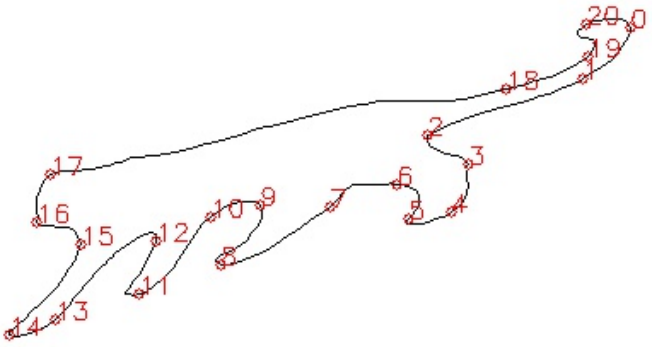


Figura 5.7. Puntos dominantes de una curva de la clase Gato obtenida aplicando afinidades.

4. Se escogen un conjunto de puntos dominantes de cada curva y se suministran como patrones de entrada del perceptrón de variable compleja de dos capas para que realice la clasificación. La capa de salida se compone de una única neurona, cuya salida representa la clase que la red asigna a los patrones de entrada: (1,0) si el patrón de entrada corresponde a Perro, y (0,1) si corresponde a Gato. La red tiene función de activación split y se entrena mediante el algoritmo de retropropagación complejo descrito en el Capítulo 3.

5. La red se entrena a partir de los puntos dominantes de las 130 curvas escogidas para entrenar.

6. La red se valida con las 1000 curvas restantes

El entrenamiento de una red neuronal es un proceso complejo en cuyo comportamiento influyen numerosos factores: el protocolo de entrenamiento (on-line o batch), el número de patrones de entrenamiento, la topología de la red (el número de capas y su tamaño, el tipo de conexiones entre las neuronas, etc), los valores iniciales de los pesos y bias, los valores del coeficiente de aprendizaje y del momento, el tipo de funciones de activación y el valor de sus coeficientes, la función de control de error, etc. Obviamente, en la práctica es imposible hallar la combinación óptima de factores; en las pruebas que se han hecho se ha encontrado que el perceptrón de variable compleja de dos capas da un buen resultado con:

- Protocolo de entrenamiento: on-line
- Número de patrones de entrenamiento: 130
- Número de neuronas de entrada: 15
- Número de neuronas de la capa oculta: 8
- Número de neuronas de salida: 1
- Dominio en el que se escogen los pesos y bias iniciales: $[-0.9, 0.9]^2 \subset \mathbb{C}$
- Factor de aprendizaje: 0.2
- Momento: 0.0
- Funciones de activación: logística con coeficiente *c=1.0*.
- Control de error de entrenamiento:

$$\frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{n}\left(O_i(p)-Y_i(p)\right)^2 < 0.8$$

  donde *P* y *n* son el número de patrones y de neuronas de salida, respectivamente; en nuestro caso: *P=130* y *n=1*. $O_i(p)$ representa la salida *i*-ésima de la red calculada a partir del patrón *p*-ésimo; $Y_i(p)$ es la componente *i*-ésima del patrón de salida *p*-ésimo.

- Test de reconocimiento: se considera que la red neuronal clasifica exitosamente una figura p si la salida de la red $O(p) = O_x(p) + O_y(p)i$ cumple:

a) si p es Perro, $O_x(p) > O_y(p)$

a) si p es Gato, $O_x(p) < O_y(p)$

La red necesitó 16517 ciclos para entrenarse. El procedimiento de clasificación consiguió un 97.3 % de aciertos con los 1000 patrones de validación.

El desempeño del clasificador basado en la red neuronal se ha comparado con un clasificador basado en los AMIs descritos en el Apartado 4.3., que consiste en calcular los valores I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9 y I_{19} para cada figura y clasificarlos con K-means[15]. En este caso, el porcentaje de aciertos es del 100 %.

5.3.2. Evaluación del desempeño del clasificador con homografías.

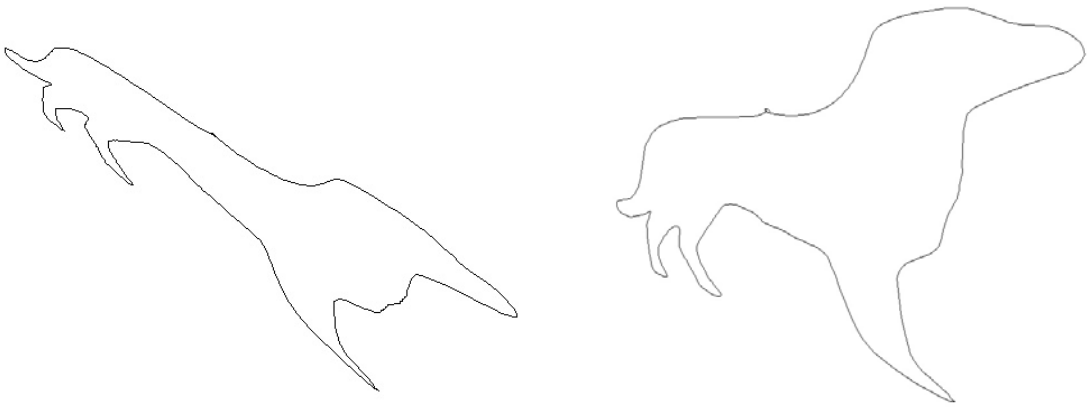
Se procede igual que en el Apartado anterior; la diferencia está en los valores de los coeficientes de las matrices de deformación:

$$a_{11} \in (0.6, 1.4) \qquad a_{12} \in (-0.4, 0.4) \qquad a_{13} \in (-0.4, 0.4)$$

$$a_{21} \in (-0.4, 0.4) \qquad a_{22} \in (0.6, 1.4) \qquad a_{23} \in (-0.4, 0.4)$$

$$a_{31} \in (-0.8, 0.8) \qquad a_{32} \in (0.0, 0.8) \qquad a_{33} = 1$$

Veamos una muestra de las curvas; de la clase Perro



[15] Para hacer la clasificación por K-means se ha empleado el paquete amap de R.

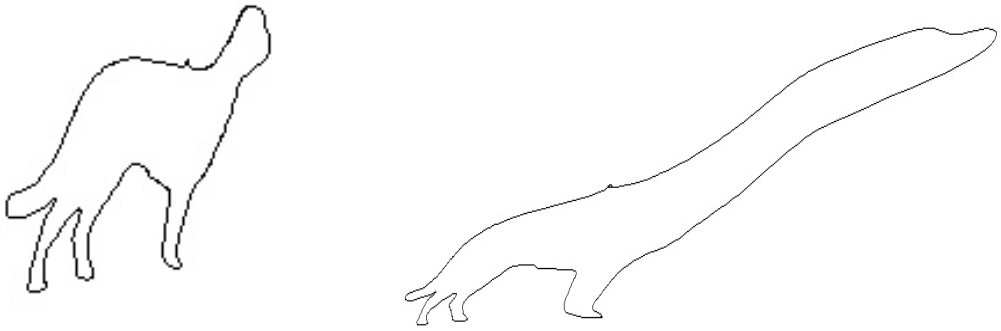


Figura 5.8. Una muestra de las curvas de la clase Perro obtenidas aplicando homografías.

y de la clase Gato

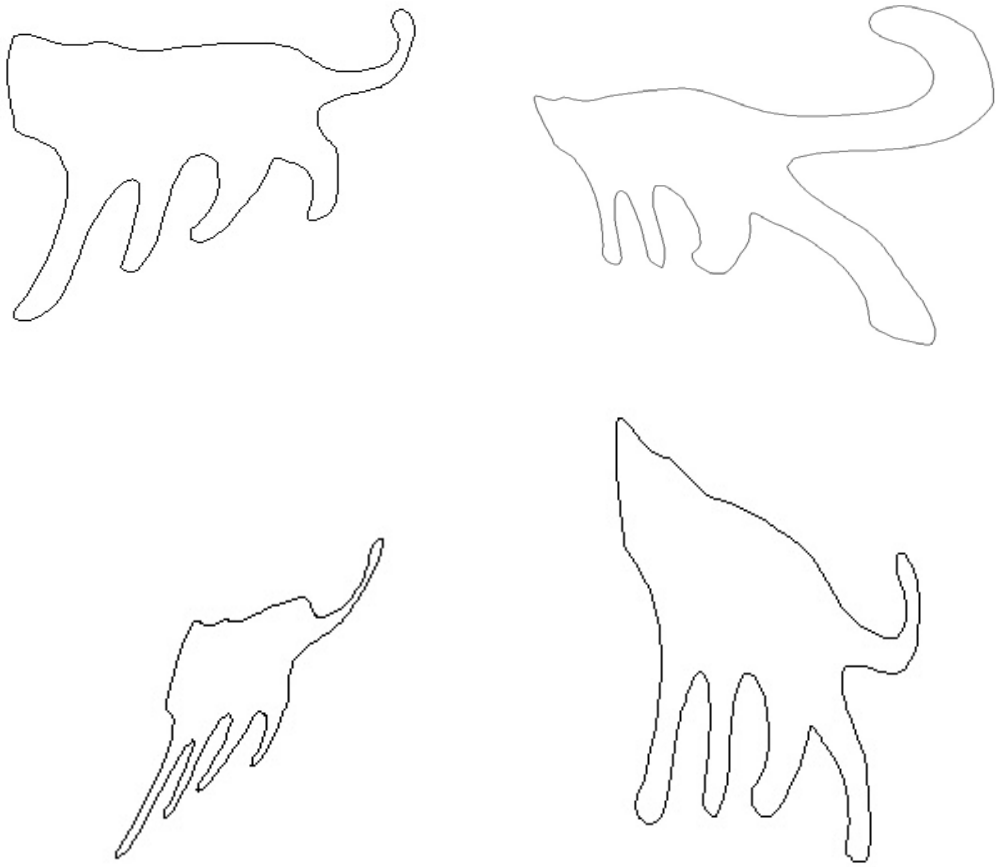


Figura 5.9. Una muestra de las curvas de la clase Gato obtenidas aplicando homografías.

Se calculan los puntos dominantes y se pasan como entradas al perceptrón de variable compleja, que se entrena considerando los siguientes parámetros:

- Protocolo de entrenamiento: on-line
- Número de patrones de entrenamiento: 180
- Número de neuronas de entrada: 25
- Número de neuronas de la capa oculta: 8
- Número de neuronas de salida: 1
- Dominio en el que se escogen los pesos y bias iniciales: $\left[-0.9, 0.9\right]^2 \subset \mathbb{C}$
- Factor de aprendizaje: 0.3
- Momento: 0.0
- Función de activación: logística con coeficiente *c=1.0*.
- Control de error de entrenamiento:

$$\frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{n}\left(O_i(p)-Y_i(p)\right)^2 < 1.6$$

  donde *P* y *n* son el número de patrones y de neuronas de salida, respectivamente; en nuestro caso: *P=180* y *n=1*.

El test de reconocimiento de los patrones es el mismo usado en el Apartado anterior.

El perceptrón complejo consiguió 97.3 % de aciertos, necesitando 59117 ciclos para entrenarse.

Si en vez de aplicar el perceptrón complejo se usa un perceptrón real con la misma topología y el mismo factor de aprendizaje, los resultados son mucho peores: en el mejor de los casos se obtiene un error del 14.8 %. Esto corrobora lo explicado en el Capítulo 3 acerca de la mayor funcionalidad de las redes neuronales complejas respecto a las reales.

Comparemos el desempeño del clasificador basado en la red neuronal compleja con un clasificador basado en la razón doble de cinco puntos (Apartado 4.4). Se necesita, por tanto, detectar en las curvas una serie de puntos que sean homólogos en las homografías, por ejemplo los puntos *especiales* (Apartado 4.4). En este caso, se han detectado los puntos bitangentes a partir del cierre convexo; por ejemplo, los puntos bitangentes de las primeras dos curvas de la muestra de la clase Perro

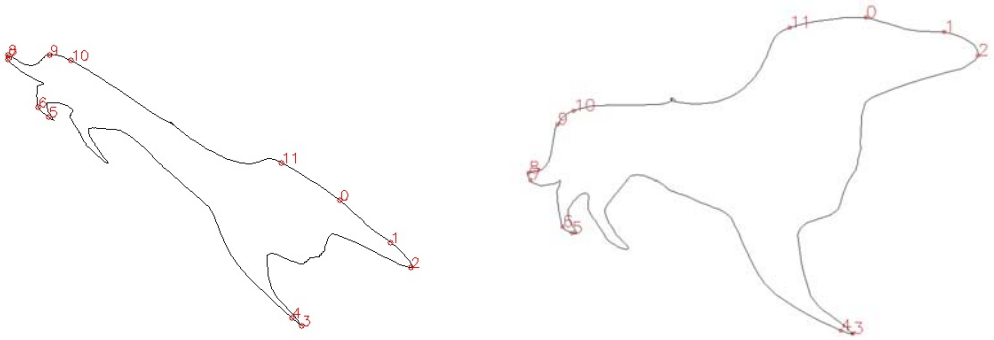


Figura 5.10. Puntos bitangentes de unas curvas de la clase Perro obtenidas aplicando homografías.

y de las primeras dos curvas de la muestra de la clase Gato

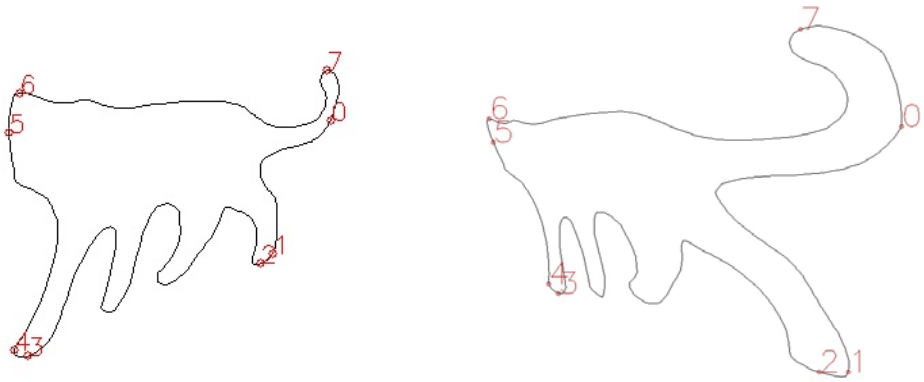


Figura 5.11. Puntos bitangentes de unas curvas de la clase Gato obtenidas aplicando homografías.

Se escogen, por ejemplo, los puntos bitangentes 0, 1, 3, 4, y 5 de cada curva y los cross-ratios J1 y J2 respectivos; usando K-means para clasificar los cross-ratios de todas las curvas se obtiene una clasificación con un 100 % de aciertos, como se advierte en la gráfica siguiente:

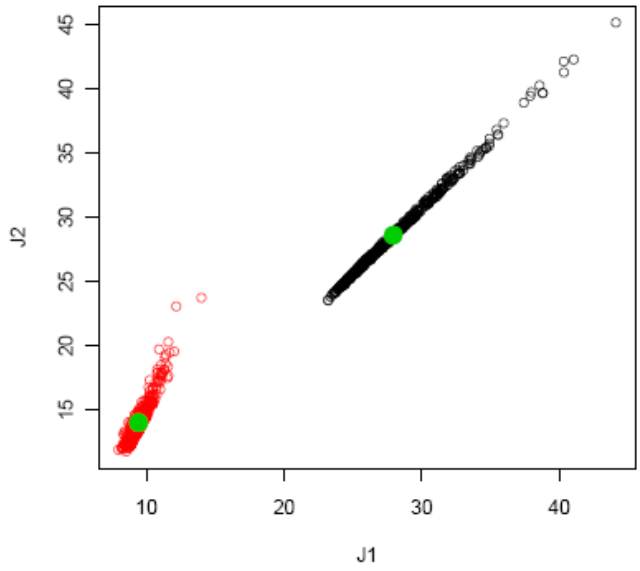


Figura 5.12. Clasificación por K-means de los cross-ratios de las curvas obtenidas aplicando homografías.

Si se calculan los AMIs I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9 y I_{19} de cada curva y se clasifican por K-means se obtiene un acierto en la clasificación de aproximadamente el 20 %, lo cual es un resultado acorde con el hecho de que los AMIs son invariantes afines y no proyectivos en general.

5.3.3. Evaluación del desempeño del clasificador con proyectividades generales.

En este caso se va a aplicar el clasificador basado en el perceptrón complejo para resolver un problema de clasificación de figuras obtenidas aplicando una transformación proyectiva general a los patrones Perro y Gato. Se va a simular que las figuras están incluidas en superficies regladas cuya generatriz son líneas quebradas (con un único quiebro, como se ve en la Figura 5.13, que se controla por el ángulo A y la longitud L).; además, las rectas que generan estas superficies al deslizarse por las generatrices, son rectas paralelas al plano de proyección; esto es, visto en alzado, la relación entre la superficie bidimensional y el plano de proyección es, por ejemplo, de la siguiente forma:

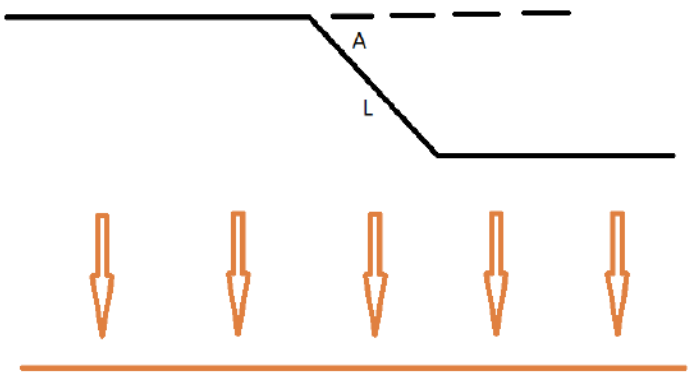


Figura 5.13. Vista en alzado de la relación entre la superficie bidimensional que contiene a una figura y el plano de proyección. Las flechas representan los rayos proyectantes.

Se supone que el foco de proyección está lo suficientemente lejos para que los rayos de proyección se pueden considerar paralelos; además, simplifiquemos el problema considerando que inciden perpendicularmente sobre el plano de proyección. Veamos el aspecto que muestran las curvas; por ejemplo, del patrón Perro

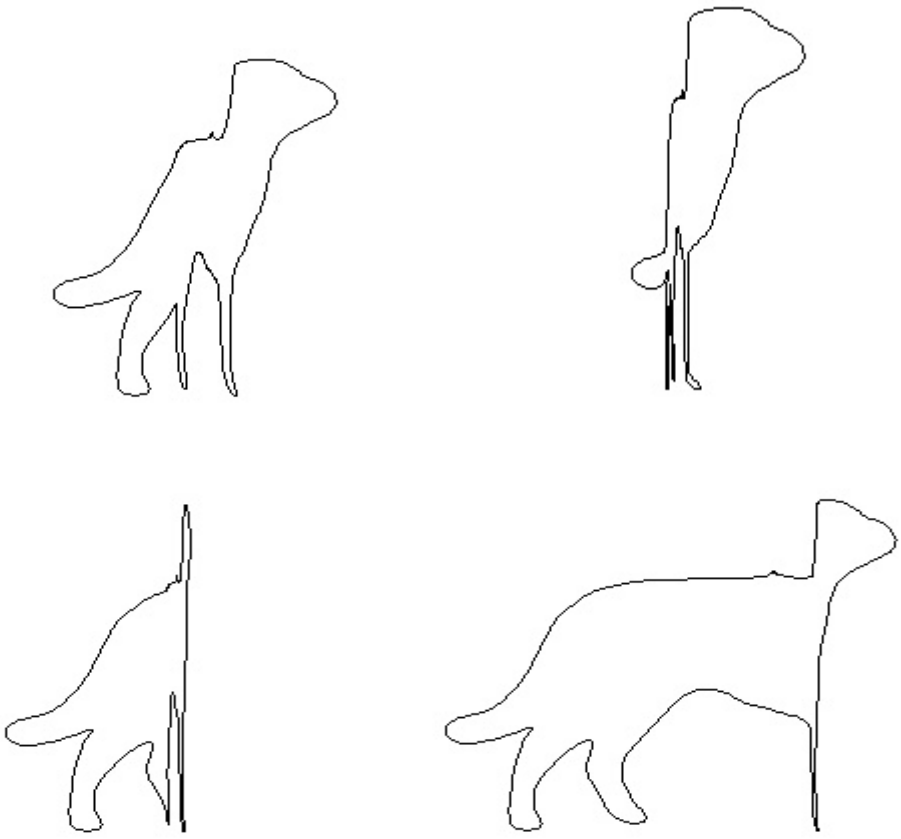


Figura 5.14. Muestra de las curvas de la clase Perro a las que se ha aplicado una transformación proyectiva general.

y del patrón Gato:

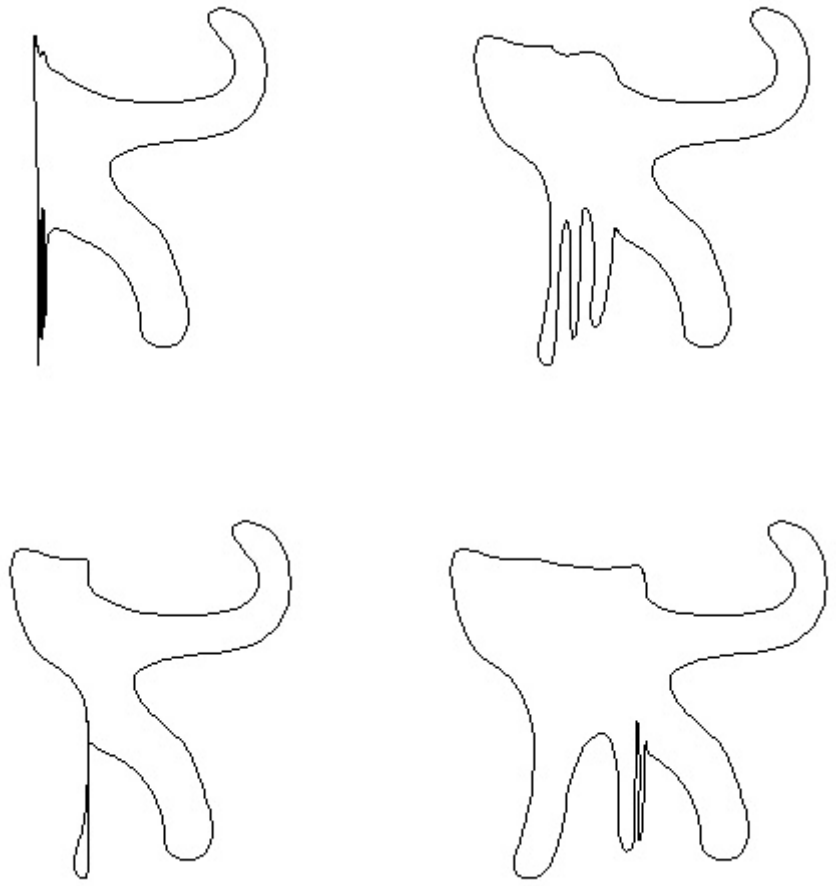


Figura 5.15. Muestra de las curvas de la clase Gato a las que se ha aplicado una transformación proyectiva general.

Se calculan los puntos dominantes y se clasifican con la red neuronal compleja; la red se entrena con los siguientes parámetros:

- Protocolo de entrenamiento: on-line
- Número de patrones de entrenamiento: 20
- Número de neuronas de entrada: 10
- Número de neuronas de la capa oculta: 8
- Número de neuronas de salida: 1
- Dominio en el que se escogen los pesos y bias iniciales: $[-0.9, 0.9]^2 \subset \mathbb{C}$

- Factor de aprendizaje: 0.9
- Momento: 0.0
- Función de activación: logística con coeficiente *c=1.0*.
- Control de error de entrenamiento:

$$\frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{n}\left(O_i(p)-Y_i(p)\right)^2 < 0.1$$

donde *P* y *n* son el número de patrones y de neuronas de salida, respectivamente; en nuestro caso: *P=20* y *n=1*.

El test de reconocimiento de los patrones es el mismo usado en el Apartado anterior.

El perceptrón complejo ha realizado la clasificación con un 99.9 % de aciertos, y ha requerido 1804 ciclos para entrenarse. Se ha intentado hacer la misma clasificación con un perceptrón de variable real y no ha sido posible hacer que la red convergiese.

Recordemos (Apéndice 3) que las razones doble de cinco puntos sólo tienen sentido con transformaciones homográficas; por lo tanto, no se pueden aplicar en este caso.

5.3.4. Comentarios a los resultados de los experimentos.

Como se ha visto en los experimentos anteriores, cuando las transformaciones son afinidades u homografías, el clasificador basado en la combinación puntos dominantes-perceptrón de variable compleja es efectivo, pero no puede competir con los métodos tradicionales (invariantes), ya que éstos son mucho más rápidos y tienen un ratio de aciertos del 100 %. Sin embargo, la situación cambia completamente cuando las transformaciones son proyectivas generales, como se ha puesto de manifiesto en un caso concreto.

Otro aspecto que se pone de manifiesto en los experimentos es la importancia de usar perceptrones de variable compleja en vez de sus homólogos reales, los cuales muestran claramente peores resultados, e incluso pueden no ser capaces de hacer la clasificación.

Apéndice 1 Expresiones útiles para el trabajo con las funciones de activación

1. $y = \frac{1}{1+e^{-cx}} = \frac{1}{2}\left(\tanh\left(\frac{cx}{2}\right)+1\right)$

   Demostración

$$\tanh\left(\frac{cx}{2}\right)+1 = \frac{e^{\frac{cx}{2}} - e^{-\frac{cx}{2}}}{e^{\frac{cx}{2}} + e^{-\frac{cx}{2}}} + 1 = \frac{2e^{\frac{cx}{2}}}{e^{\frac{cx}{2}} + e^{-\frac{cx}{2}}} = \frac{e^{\frac{cx}{2}} \cdot 2e^{\frac{cx}{2}}}{e^{\frac{cx}{2}} \cdot \left(e^{\frac{cx}{2}} + e^{-\frac{cx}{2}}\right)} = \frac{2e^{cx}}{e^{cx}+1} = \frac{2}{1+e^{-cx}}$$

2. $\tanh(cx) = \frac{e^{cx} - e^{-cx}}{e^{cx} + e^{-cx}} = \frac{2}{1+e^{-2cx}} - 1 = \tan sig(cx)$

   Demostración

$$\tanh(cx) = \frac{e^{cx} - e^{-cx}}{e^{cx} + e^{-cx}} = \frac{e^{cx}\left(1-e^{-2cx}\right)}{e^{cx}\left(1+e^{-2cx}\right)} = \frac{1-e^{-2cx}}{1+e^{-2cx}} = \frac{1}{1+e^{-2cx}} - \frac{e^{-2cx}}{1+e^{-2cx}} =$$

$$= \frac{1}{1+e^{-2cx}} + \left(\frac{1}{1+e^{-2cx}} - \frac{1}{1+e^{-2cx}}\right) - \frac{e^{-2cx}}{1+e^{-2cx}} = \frac{2}{1+e^{-2cx}} - \frac{1+e^{-2cx}}{1+e^{-2cx}} =$$

$$= \frac{2}{1+e^{-2cx}} - 1 = \tan sig(cx)$$

3. Sea $f_c(x) = \frac{1}{1+e^{-cx}} = \left(1+e^{-cx}\right)^{-1}$, entonces $f_c'(x) = cf_c(x)\left(1-f_c(x)\right)$

   Demostración

$$f_c'(x) = -1\cdot\left(1+e^{-cx}\right)^{-2}\cdot -ce^{-cx} = ce^{-cx}\left(1+e^{-cx}\right)^{-2} = \frac{ce^{-cx}}{\left(1+e^{-cx}\right)^2} = c\frac{1}{1+e^{-cx}}\cdot\frac{e^{-cx}}{1+e^{-cx}} =$$

$$= c\frac{1}{1+e^{-cx}}\cdot\frac{\left(1-1+e^{-cx}\right)}{1+e^{-cx}} = c\frac{1}{1+e^{-cx}}\cdot\left(\frac{1+e^{-cx}}{1+e^{-cx}} - \frac{1}{1+e^{-cx}}\right) = c\frac{1}{1+e^{-cx}}\cdot\left(1-\frac{1}{1+e^{-cx}}\right) =$$

$$= cf_c(x)\left(1-f_c(x)\right)$$

4. Sea $f_c(x) = \tanh(cx) = \dfrac{e^{cx} - e^{-cx}}{e^{cx} + e^{-cx}}$, entonces $f_c^{'}(x) = c\left(1 - f_c^{2}(x)\right)$

Demostración

$$f_c\,'(x) = \frac{c\left(e^{cx} + e^{-cx}\right)\left(e^{cx} + e^{-cx}\right) - c\left(e^{cx} - e^{-cx}\right)\left(e^{cx} - e^{-cx}\right)}{\left(e^{cx} + e^{-cx}\right)^2} = \frac{c\left(e^{2cx} + 2 + e^{-2cx} - e^{2cx} + 2 - e^{-2cx}\right)}{\left(e^{cx} + e^{-cx}\right)^2} =$$

$$= \frac{4c}{\left(e^{cx} + e^{-cx}\right)^2} = c - c + \frac{4c}{\left(e^{cx} + e^{-cx}\right)^2} = c - c\left(1 - \frac{4}{\left(e^{cx} + e^{-cx}\right)^2}\right) = c - c\left(\frac{e^{2cx} + e^{-2cx} + 2 - 4}{\left(e^{cx} + e^{-cx}\right)^2}\right) =$$

$$= c - c\left(\frac{e^{2cx} + e^{-2cx} - 2}{\left(e^{cx} + e^{-cx}\right)^2}\right) = c - c\left(\frac{e^{cx} - e^{-cx}}{e^{cx} + e^{-cx}}\right)^2 = c - c\tanh^2(cx) = c\left(1 - \tanh^2(cx)\right)$$

Apéndice 2 Conceptos de la Teoría de probabilidades

En este Apéndice se presentan aquellos conceptos de la Teoría de Probabilidades que se han considerado necesarios para seguir los desarrollos del Proyecto. En su elaboración se ha trabajado fundamentalmente con el texto [48], y si se ha acudido a otra fuente se ha indicado expresamente.

Definición 1 σ – álgebra

Sea un conjunto Ω y una clase $\mathrm{A} \subset P(\Omega)$, con $P(\Omega)$ el conjunto de partes de Ω. Se dice que A tiene estructura de σ – álgebra si y sólo si se cumple:

1. $\Omega \in \mathrm{A}$ (Ω se denomina suceso total)
2. $\forall A \in \mathrm{A}$ se tiene $\overline{A} \in \mathrm{A}$ (en consecuencia, $\varnothing = \overline{\Omega} \in \mathrm{A}$)
3. $\forall \{A_n\} \subset \mathrm{A}$ se tiene $\bigcup_{n=1}^{\infty} A_n \in \mathrm{A}$

Definición 2 Espacio probabilizable

Sea Ω un conjunto y $\mathrm{A} \in P(\Omega)$, tal que A σ – álgebra. Se dice que el par $\{\Omega, \mathrm{A}\}$ es un espacio probabilizable.

Definición 3 Espacio probabilístico

Sea $\{\Omega, \mathrm{A}\}$ un espacio probabilizable y $p : \mathrm{A} \longrightarrow \mathbb{R}$ una función de conjunto que cumple:

1. $\forall A \in \mathrm{A} \quad p(A) \geq 0$
2. $p(\Omega) = 1$
3. $\forall \{A_n\} \subset \mathrm{A}$ tal que $A_i \cap A_j = \varnothing \ \ \forall i \neq j$, se cumple $p\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} p(A_n)$

A la función p se le denomina probabilidad y a la terna $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico, o espacio de probabilidad. Ω y A son el espacio muestral y el espacio de sucesos, respectivamente, de dicho espacio probabilístico. A los elementos de A se les denomina sucesos.

Definición 4 Probabilidad condicionada

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico y $A \in \mathrm{A}$ tal que $p(A) > 0$. Se denomina probabilidad condicionada del suceso $B \in \mathrm{A}$ respecto al suceso $A \in \mathrm{A}$, y se escribe $p(B \mid A)$ a

$$p(B \mid A) = \frac{p(A \cap B)}{p(A)}$$

Definición 5 Sucesos independientes

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico y $A, B \in \mathrm{A}$ dos sucesos. Se dice que ambos sucesos son independientes si y sólo si $p(A \cap B) = p(A) \cdot p(B)$, de lo contrario, se dice que son dependientes. En virtud de la definición 4, si $p(A), p(B) > 0$, , la condición de independencia se puede expresar $p(A \mid B) = p(A)$, o igualmente $p(B \mid A) = p(B)$. **Nota**: no se debe confundir el concepto de independencia con el de exclusión. Dos sucesos son excluyentes si su intersección es el suceso nulo. Obviamente, si dos sucesos A y B son excluyentes y cumplen $p(A) \neq 0$, $p(B) \neq 0$, entonces son dependientes, pues $p(A) \cdot p(B) \neq 0$ y $p(A \cap B) = 0$

Definición 6 Sucesos mutuamente independientes

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico y $\{A_i\}_{i \in I} \subset \mathrm{A}$ una familia de sucesos. Se dice que $\{A_i\}_{i \in I}$ forma una familia de sucesos mutuamente independientes (es común referirse a ellos simplemente como independientes) si para todo $\{i_1, ..., i_n\} \subset I \quad n \geq 2$ se satisface

$$p\left(\bigcap_{j=1}^{n} A_{i_j}\right) = \prod_{j=1}^{n} p\left(A_{i_j}\right)$$

La independencia mutua implica la independencia dos a dos, pero no viceversa (ambos conceptos coinciden si $i \leq 2$). Se puede ver con un ejemplo [97]:

Se tiene un tetraedro con una cara roja, una cara negra, una cara blanca y la cuarta cara pintada con los tres colores. La probabilidad de que al lanzarlo sobre una mesa se obtenga cualquiera de las caras es $1/4$. El experimento aleatorio consiste en lanzar el tetraedro y ver qué color tiene la cara en la que ha caído. El espacio muestral es $\Omega = \{C_R, C_N, C_B, C_T\}$, donde C_R, C_N, C_B y C_T representan cara roja, negra, blanca y tricolor, respectivamente, y se definen los sucesos

$$R = \{\text{el tetrahedro se apoya en una cara con color rojo}\} = \{C_R, C_T\}$$

$N = \{\text{el tetrahedro se apoya en una cara con color negro}\} = \{C_N, C_T\}$

$B = \{\text{el tetrahedro se apoya en una cara con color blanco}\} = \{C_B, C_T\}$

Entonces, $p(R \cap N) = p(\{C_T\}) = 1/4 = p(R) \cdot p(N)$. Lo mismo para $p(R \cap B)$ y $p(N \cap B)$. Por lo tanto, se cumple la independencia dos a dos. Sin embargo, $p(R \cap N \cap B) = p(\{C_T\}) = 1/4 \neq 1/8 = p(R) \cdot p(N) \cdot p(B)$, luego no hay independencia mutua.

Definición 7 Clases de sucesos mutuamente independientes

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico y $\{C_i\}_{i \in I}$ una familia de clases de sucesos, es decir, $C_i \subset \mathrm{A} \quad \forall i \in I$. Se dice que las clases $C_i \quad \forall i \in I$ son mutuamente independientes (nos referiremos a ellos simplemente como independientes) si y sólo si para todo subconjunto finito de I $\{i_1, ..., i_n\} \subset I \quad n \geq 1$ y para toda posible elección de sucesos $A_{i_j} \in C_{i_j} \quad j \in \{1, ..., n\}$ se cumple:

$$p\left(\bigcap_{j=1}^{n} A_{i_j}\right) = \prod_{j=1}^{n} p\left(A_{i_j}\right)$$

Proposición 1 Teorema del producto

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico. Si $A, B \in \mathrm{A} \setminus p(A) > 0$, $p(B) > 0$ se tiene:

$$p(A \cap B) = p(A) \cdot p(B \mid A) = p(B) \cdot p(A \mid B)$$

Demostración: es obvia a partir de la Definición 4.

Proposición 2 Teorema de la probabilidad total

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico, $S = \{A_n\} \subset \mathrm{A}$ un sistema completo de sucesos (es decir, $A_i \cap A_j = \varnothing \quad \forall i \neq j$, y $\bigcup_{n=1}^{\infty} A_n = \Omega$), $p(A_n)$ conocida $\forall A_n \in S$ y $B \in \mathrm{A}$ tal que $p(B \mid A_n)$ conocida $\forall A_n \in S$. Entonces se cumple:

$$p(B) = \sum_{n=1}^{\infty} p(B \mid A_n) p(A_n)$$

Proposición 3 Teorema de Bayes

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico, $\{A_n\} \subset \mathrm{A}$ un sistema completo de sucesos tal que $p(A_n) > 0 \ \ \forall n \in \mathbb{N}$, $B \in \mathrm{A} \setminus \ p(B) > 0$ y las probabilidades $p(B \mid A_n) \ \ \forall n \in \mathbb{N}$ son conocidas. Entonces:

$$p(A_n \mid B) = \frac{p(B \mid A_n)\, p(A_n)}{\sum_{k=1}^{\infty} p(B \mid A_k) p(A_k)} \qquad \forall n \in \mathbb{N}$$

A las probabilidades $p(A_n)$ y $p(A_n \mid B)$ se les denomina probabilidades a priori y posteriori, respectivamente. A las probabilidades $p(B \mid A_n)$ se les llama verosimilitudes.

Proposición 4

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico, $A \in \mathrm{A} \setminus \ p(A) > 0$. Entonces, $\{\Omega, \mathrm{A}, p(\cdot \mid A)\}$ es un espacio probabilístico. Se demuestra que $\mathrm{A}_A = \mathrm{A} \cap A \triangleq \{A \cap B, \ \forall B \in A\}$ es un $\sigma-$álgebra de A y $p_A = p(\cdot \mid A)$ es una probabilidad sobre A_A. Por lo tanto, si $p(A) > 0$, entonces $\{A, \mathrm{A}_A, p_A\}$ es un espacio probabilístico

Las probabilidades condicionadas tienen utilidad en experimentos aleatorios "dinámicos", es decir, realizados en varias fases, en cada una de las cuales, mediante el teorema de Bayes, se incorpora nueva información relativa al fenómeno en estudio.

Un ejemplo simple puede servir para clarificar la forma en que el teorema de Bayes permite actualizar la información de un experimento aleatorio: supónganse tres urnas, denominadas I, II y III, que contienen dos tipos de bolas, blancas y rojas. La urna I tiene tres bolas blancas y dos rojas, la urna II tiene cuatro blancas y dos rojas y la III tiene tres bolas rojas. El experimento aleatorio consiste en extraer a ciegas una bola de alguna de las urnas y decir a cual pertenece. A este experimento le asociamos un espacio muestral $\Omega = \textit{conjunto de bolas}$ y un conjunto de sucesos $U = \{U_I, U_{II}, U_{III}\}$, siendo U_i el conjunto de bolas de la urna i. Obviamente, *U* es un conjunto de sucesos completo, pues está formado por conjuntos disjuntos dos a dos y su unión es el suceso total.

Comencemos el experimento extrayendo una bola de alguna de las urnas sin tener en cuenta su color. Al carecer de la suficiente información para hacer una asignación de probabilidades a los sucesos U_i, se emplea el Principio de la Razón Insuficiente, es decir:

$$p(U_i) = \frac{1}{3} \qquad i \in \{I, II, III\}$$

A continuación, la bola se devuelve a la urna y se repite el experimento, pero en este caso se observa el color de la bola extraída: es blanca. Se consideran dos nuevos sucesos: *B*=conjunto de bolas blancas y *R*=conjunto de bolas rojas. La bola que se extrae es blanca, por lo tanto, se ha cumplido el suceso *B*. Esta información se utiliza para actualizar el valor de las probabilidades de U_i mediante el teorema de Bayes; las probabilidades de la fase anterior del experimento son las probabilidades a priori, y las nuevas probabilidades son las probabilidades a posteriori:

$$p(U_I \mid B) = \frac{p(B \mid U_I)\, p(U_I)}{\sum_{i \in \{I,II,III\}} p(B \mid U_i)\, p(U_i)} = \frac{3/5 \cdot 1/3}{3/5 \cdot 1/3 + 4/6 \cdot 1/3 + 0 \cdot 1/3} = \frac{1/5}{1/5 + 2/9} = \frac{1/5}{19/45} = \frac{9}{19}$$

$$p(U_{II} \mid B) = \frac{p(B \mid U_{II})\, p(U_{II})}{\sum_{i \in \{I,II,III\}} p(B \mid U_i)\, p(U_i)} = \frac{4/6 \cdot 1/3}{3/5 \cdot 1/3 + 4/6 \cdot 1/3 + 0 \cdot 1/3} = \frac{2/9}{1/5 + 2/9} = \frac{2/9}{19/45} = \frac{10}{19}$$

$$p(U_{III} \mid B) = \frac{p(B \mid U_{III})\, p(U_{III})}{\sum_{i \in \{I,II,III\}} p(B \mid U_i)\, p(U_i)} = \frac{0}{10/19} = 0$$

Las probabilidades condicionadas constituyen una parte fundamental de la llamada estadística bayesiana, que se basa en:

1. Las probabilidades de los modelos probabilísticos son probabilidades condicionadas Como se ve en la Proposición 4, el espacio probabilístico condicionado es un verdadero espacio probabilístico y, por lo tanto, es consistente con los axiomas de Kolmogoroff.

2. Las probabilidades reflejan información que puede ser objetiva (frecuencias) o no (grado de creencia). Mediante un proceso iterativo, basado en el teorema de Bayes, y partiendo de modelos probabilísticos poco informativos, se llega a modelos probabilísticos que reflejan de forma más fiel el comportamiento del fenómeno.

La principal diferencia entre la estadística clásica y la bayesiana es su interpretación de la probabilidad: frecuentista en la estadística clásica, y subjetiva en la bayesiana. Para la estadística clásica la probabilidad es un concepto objetivo, resultado de la observación de la naturaleza, de aquí la importancia que da al muestreo, del que extrae la información con que calcula las probabilidades. Por contra, en la estadística bayesiana la probabilidad es subjetiva, depende del observador, es una medida personal de la creencia en la ocurrencia de una afirmación dada cierta evidencia acerca de ésta. Las probabilidades bayesianas son siempre probabilidades condicionadas, pues sólo tienen sentido en el marco de unos supuestos previos [9].

Las diferencias en el concepto de probabilidad que tienen ambas escuelas estadísticas muestra su reflejo en la forma en la que conciben la inferencia. En la inferencia clásica

el modelo probabilístico (es decir, la familia de funciones de distribución que modelan el fenómeno aleatorio) es conocido y la incógnita es el parámetro(s) determinista del que depende dicho modelo, que se obtiene, por ejemplo, por el Método de Máxima Verosimilitud. En la inferencia bayesiana se parte de un modelo probabilístico a priori con unos parámetros no deterministas (es decir, a los que se asocia una función de distribución) y, a medida que se incorpora nueva información a través de experimentos aleatorios sucesivos, se refina el modelo del fenómeno. Es necesario señalar que la inferencia bayesiana sólo resulta verdaderamente útil cuando sus resultados son insensibles a la elección de las probabilidades a priori.

Ahora, veamos brevemente el concepto de variable aleatoria real (continua y discreta).

Definición 8 Variable aleatoria real

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico, $(\mathbb{R}, \mathrm{B})$ el espacio probabilizable formado por la recta real y el $\sigma-$álgebra de los conjuntos de Borel $\mathrm{B} = \mathrm{B}(\mathbb{R})$ (B está formado por todos los conjuntos de la forma (a,b), $[a,b]$, $[a,b)$, $(a,b]$, $(-\infty,b)$, $(-\infty,b]$, $(a,+\infty)$, $[a,+\infty)$, $\{a\}$ y sus uniones, siendo $a,b \in \mathbb{R}$) y $X : \Omega \longrightarrow \mathbb{R}$ una aplicación entre Ω y $\mathbb{R}$. Entonces, *X* es variable aleatoria si y sólo si $X^{-1}(B) \in \mathrm{A}$ $\forall B \in \mathrm{B}$.

La variable aleatoria *X* induce en $(\mathbb{R}, \mathrm{B})$ una probabilidad p_X de la siguiente forma:

$$p_X(B) = p\left(X^{-1}(B)\right) = p(A) \quad \forall B \in \mathrm{B} \text{ siendo } X(A) = B$$

Por lo tanto, $(\mathbb{R}, \mathrm{B}, p_X)$ es un espacio probabilístico. Las variables aleatorias reales transforman el espacio probablístico $\{\Omega, \mathrm{A}, p\}$ en el espacio probabilístico $(\mathbb{R}, \mathrm{B}, p_X)$

Definición 9 Realización de una variable aleatoria

Se llama realización de una variable aleatoria a cualquier valor $x \in \mathbb{R} \setminus \exists \omega \in \Omega : \; x = X(\omega)$

Definición 10 Variables aleatorias reales mutuamente independientes

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico y $\{X_i\}_{i \in I}$ una familia de variables aleatorias reales. Se puede definir la familia $\{\sigma(X_i)\}_{i \in I}$, con $\sigma(X_i) = \{A \in \mathrm{A} \setminus \exists B \in \mathrm{B}(\mathbb{R}) : \; X_i(A) \in B\}$; luego, $\{\sigma(X_i)\}_{i \in I}$ es una familia de clases de sucesos. Entonces: $\{X_i\}_{i \in I}$ son variables aleatorias reales mutuamente independientes (usualmente se denominan independientes) si y sólo si $\{\sigma(X_i)\}_{i \in I}$ es una familia de clases de sucesos independientes (ver Definición 7).

Definición 11 Función de distribución de $\mathbb{R}$

Se denomina función de distribución de $\mathbb{R}$ a toda aplicación $F:\mathbb{R}\longrightarrow\mathbb{R}$ que cumpla:

a) F monótona no decreciente.

b) Continua por la derecha.

c) $\lim_{x\to-\infty} F(x)=0$

d) $\lim_{x\to\infty} F(x)=1$

Dado el espacio probabilístico $(\mathbb{R},\mathrm{B},p_X)$, se puede demostrar que

$$F_X(x)=p_X\left((-\infty,x]\right)\quad\forall x\in\mathbb{R}$$

es una función de distribución. Inversamente, dada una función de distribución $F_X(x)$ $\forall x\in\mathbb{R}$, se induce de forma única una probabilidad p_X tal que $(\mathbb{R},\mathrm{B},p_X)$ es espacio probabilístico.

Definición 12 Variables aleatorias igualmente distribuidas

Sean las variables aleatorias reales X e Y. Son idénticamente distribuidas si y sólo si $\forall x\in X:\quad \exists 1 y_x\in Y \;\backslash\; Y\left(X^{-1}(-\infty,x]\right)=(-\infty,y_x]$ y $F_X(x)=F_Y(y_x)$. Que ambas variables aleatorias sean igualmente distribuidas no significa que sean iguales (es decir, $X=Y$), del mismo modo que pueden ser iguales y tener distribuciones distintas.

Definición 13 Función de probabilidad en $\mathbb{R}$

Sea X una variable aleatoria definida sobre el espacio probabilístico $\{\Omega,\mathrm{A},p\}$, y p_X la probabilidad inducida por X en $(\mathbb{R},\mathrm{B})$. Se llama función de probabilidad P_X a la aplicación $P_X:\mathbb{R}\longrightarrow[0,1]$ definida por

$$P_X(x)=p_X(\{x\})=p\left(X^{-1}(\{x\})\right)$$

Definición 14 Función de densidad en $\mathbb{R}$

Se denomina función de densidad sobre $\mathbb{R}$ a toda función $f:\mathbb{R}\longrightarrow\mathbb{R}$ que cumple:

1. $f(x)\geq 0\quad\forall x\in\mathbb{R}$

2. f es integrable Riemann.

3. $\int_{-\infty}^{+\infty} f(x)\,dx = 1$

Definición 15 Variable aleatoria discreta en $\mathbb{R}$

Sea X una variable aleatoria definida sobre el espacio probabilístico $\{\Omega, \mathrm{A}, p\}$, que tiene una función de probabilidad P_X. Se define el conjunto $D_X = \{x \in \mathbb{R} \setminus P_X(x) > 0\}$, denominado soporte de la variable aleatoria. Entonces, *X* es variable aleatoria discreta si y sólo si $D_X \neq \varnothing$ y

$$\sum_{x \in D_X} P_X(x) = 1$$

En este caso, se cumple: $F_X(x) = \sum_{x' \in D_X \ \wedge\ x' \leq x} P_X(x')$

Definición 16 Variable aleatoria continua en $\mathbb{R}$

Sea $X : (\Omega, \mathrm{A}) \longrightarrow (\mathbb{R}, \mathrm{B})$ una variable aleatoria. Se dice que *X* es continua si y sólo si $\exists f_X$ función de densidad en $\mathbb{R}$ tal que su función de distribución F_X se puede expresar:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,dt$$

f_X es la función de densidad de *X*.

Cuando una variable aleatoria *X* es continua se cumple $p(\langle a,b \rangle) = \int_a^b f_X(t)\,dt$, con $\langle\ \rangle = (\ ), [\ ], (\ ], [\ )$. En consecuencia

$$p(\{x\}) = p(X = x) = p(\langle x,x \rangle) = \int_x^x f_X(t)\,dt = 0 \ \forall x \in X$$

Los conceptos de variable aleatoria unidimensional se extienden fácilmente al caso n-dimensional.

Definición 17 Vector aleatorio n-dimensional ó variable aleatoria n-dimensional

Sea $\{\Omega, \mathrm{A}, p\}$ espacio probabilístico. La aplicación vectorial $X = (X_1, ..., X_n)$ definida por

$$\begin{aligned} X : \Omega &\longrightarrow \mathbb{R}^n \\ \omega &\longrightarrow (X_1(\omega), ..., X_n(\omega)) \end{aligned}$$

es variable aleatoria n-dimensional si y sólo si $X^{-1}(B) \in \mathrm{A} \quad \forall B \in \mathrm{B}_n$, donde B_n es el σ – álgebra de Borel de $\mathbb{R}^n$, que se define de forma similar a B en $\mathbb{R}$.

Se demuestra que $X = (X_1, ..., X_n)$ es variable aleatoria n-dimensional si y sólo si las $X_i \quad i \in \{1, ..., n\}$ son variables aleatorias unidimensionales.

Los conceptos de función de distribución y de densidad de vectores aleatorios son simples extensiones de los correspondientes conceptos del caso unidimensional. A las funciones de distribución y densidad de un vector aleatorio $X = (X_1, ..., X_n)$ se les denomina funciones de distribución y densidad conjuntas de las variables aleatorias unidimensionales $X_i \quad i \in \{1, ..., n\}$.

Proposición 5

Sean $X_1, ..., X_n$ variables aleatorias unidimensionales sobre $\{\Omega, \mathrm{A}, p\}$. Entonces, la función de distribución sobre $(X_1, ..., X_n)$ es:

$$F(x_1, ..., x_n) = p(X_1 \le x_1, ..., X_n \le x_n) = \prod_{i=1}^{n} F_{X_i}(x_i) \qquad \forall (x_1, ..., x_n) \in \mathbb{R}^n$$

Definición 18 Variable aleatoria continua

Sea el vector aleatorio $X = (X_1, ..., X_n)$. Se dice que X es continuo si y sólo si $\exists f_{(X_1, ..., X_n)}$ función de densidad en $\mathbb{R}$ tal que su función de distribución $F_{(X_1, ..., X_n)}$ puede expresarse:

$$F_{(X_1, ..., X_n)}(x_1, ..., x_n) = \int_{-\infty}^{x_1} ... \int_{-\infty}^{x_n} f_{(X_1, ..., X_n)}(u_1, ..., u_n)\, du \cdot ... \cdot du_n$$

$f_{(X_1, ..., X_n)}$ es la función de densidad de X.

Definición 19 Funciones de densidad marginales

Sea $f_{(X,Y)}$ función de densidad de la variable aleatoria bidimensional continua (X, Y). Se llaman funciones de densidad marginales de las variables aleatorias X e Y continuas a las funciones:

$$f_X : \mathbb{R} \longrightarrow \mathbb{R}$$
$$x \longrightarrow \int_{-\infty}^{+\infty} f(x,v)\,dv$$

$$f_Y : \mathbb{R} \longrightarrow \mathbb{R}$$
$$y \longrightarrow \int_{-\infty}^{+\infty} f(u,y)\,du$$

Proposición 6

Sean $X_1,...,X_n$ variables aleatorias continuas sobre $\{\Omega, A, p\}$, con funciones de densidades marginales $f_{X_1},\ldots, f_{X_n}$. Entonces, $X_1,...,X_n$ son independientes si y sólo si

$$f(x_1,...,x_n) = \prod_{i=1}^{n} f_{X_i}(x_i) \qquad \forall (x_1,...,x_n) \in \mathbb{R}^n$$

Ahora veamos el concepto de función de distribución de la variable aleatoria *X* condicionada por *Y*=y, siendo *X,Y* variables aleatorias continuas. Con este fin, se define en el vector aleatorio bidimensional (X,Y) la expresión:

$$p\{X \le x \mid y-\varepsilon < Y \le y+\varepsilon\} = \frac{p_{(X,Y)}\{X \le x, y-\varepsilon < Y \le y+\varepsilon\}}{p_Y\{Y \in (y-\varepsilon, y+\varepsilon)\}}$$

con $\varepsilon \in \mathbb{R}^+$, que está bien definida porque el denominador es una probabilidad evaluada en un intervalo y no en un punto. A partir de esta expresión se define:

Definición 20 Función de distribución de la variable aleatoria X condicionada por Y=y (ambas variables aleatorias continuas)

Se denomina función de distribución de la variable aleatoria *X* condicionada por *Y=y* a

$$\lim_{\varepsilon \to 0^+} p\{X \le x \mid y-\varepsilon < Y \le y+\varepsilon\}$$

cuando tal límite exista. En ese caso constituye una función de distribución en $\mathbb{R}$ y se representa por $F_{X|Y}(x \mid y)$.

Definición 21 Función de densidad de la variable aleatoria X condicionada por Y=y (ambas variables aleatorias continuas)

Se denomina función de densidad de la variable aleatoria X condicionada por Y=y, y se representa por $f_{X|Y}(x|y)$, a una función no negativa tal que:

$$F(x|y) = \int_{-\infty}^{x} f_{X|Y}(t|y)\,dt \qquad \forall x \in \mathbb{R}$$

Proposición 7

Sea la variable aleatoria bidimensional continua (X,Y) con función de densidad $f_{(X,Y)}$. Entonces, $\forall (x,y) \in \mathbb{R}^2$ en el que $f_{(X,Y)}$ y f_Y son continuas, y $f_Y > 0$, la función de densidad condicional de la variable aleatoria *X* condicionada por *Y=y* existe y se expresa así:

$$f_{X|Y}(x|y) = \frac{f_{(X,Y)}(x,y)}{f_Y(y)}$$

es decir, $f_{(X,Y)}(x,y) = f_{X|Y}(x|y) \cdot f_Y(y)$.

Lo mismo para $f_{Y|X}$.

Si se cumplen las condiciones de la Proposición 7, y teniendo en cuenta la Proposición 6, es inmediato demostrar que dos variables aleatorias unidimensionales continuas X e Y son independientes si y sólo si

$$f_{X|Y}(x|y) = f_X(x) \quad \forall y \in Y$$

o igualmente si y sólo si

$$f_{Y|X}(y|x) = f_Y(y) \quad \forall x \in X$$

Por último se resumen los conceptos asociados al Método de máxima verosimilitud, usado por la escuela estadística clásica para estimar el valor del parámetro(s) de un modelo probabilístico asociado a un fenómeno. La esencia del método consiste en dar como estimación del parámetro(s) del modelo, de entre todos los posibles, aquel que maximice la probabilidad de la muestra observada. El enunciado del método requiere unas definiciones previas [49], [29].

Definición 22 Muestra aleatoria

Sea una variable aleatoria *X*. Se denomina muestra aleatoria de *X* a una variable aleatoria n-dimensional $(X_1,...,X_n)$ tal que $X_i = X \quad \forall i \in \{1,...,n\}$. La función de distribución de $(X_1,...,X_n)$ depende de la técnica concreta de muestreo.

Definición 23 Muestra aleatoria simple

Sea una variable aleatoria $X \sim F_X$. Se denomina muestra aleatoria simple de *X* a un vector aleatorio n-dimensional $(X_1,...,X_n)$ tal que:

a) $X_i = X$ y $F_{X_i} = F_X \quad \forall i \in \{1,...,n\}$.

b) Las variables X_i son mutuamente independientes, luego

$$F_{(X_1,...,X_n)}(x_1,...,x_n) = F_{X_1}(x_1)\cdot...\cdot F_{X_n}(x_n)$$

***Definición 24 Estadístico muestral* [49]**

Un estadístico muestral de la muestra $(X_1,...,X_n)$ de una variable aleatoria X es una función

$$T:\left((X_1,...,X_n),\mathrm{B}^n\right)\longrightarrow\left(\mathbb{R}^k,\mathrm{B}^k\right)$$

medible Lebesgue en las σ – álgebras B^n y B^k. *k* es la dimensión del estadístico. El estadístico muestral constituye un vector aleatorio *k*-dimensional. También se le suele denominar estadístico puntual.

***Definición 25 Verosimilitud de una muestra de una variable aleatoria* [49]**

Es un caso particular de estimador muestral, en el que *T* es la función de probabilidad (si *X* es discreta) o de densidad (si *X* es continua) de la muestra $(X_1,...,X_n)$. Se representa por $L(X_1,...,X_n)$. Si la muestra es aleatoria simple, se cumple:

$$L(X_1,...,X_n) = \prod_{i=1}^{n} p(X_i)$$

***Definición 26 Estimación de un estimador* [49]**

Se denomina estimación de un estimador muestral *T* a una realización de $T(X_1,...,X_n)$, es decir, a un valor $T(x_1,...,x_n)$, siendo $(x_1,...,x_n)$ una realización del vector aleatorio $(X_1,...,X_n)$.

***Definición 27 Función de verosimilitud de una muestra aleatoria simple de una variable aleatoria* [49]**

Sea $(X_1,...,X_n)$ una muestra de una variable aleatoria *X*, que vamos a suponer que es continua y cuya función de densidad depende del parámetro poblacional $\vec{\theta}$ (si es discreta se hace igual). Además, supongamos que la muestra es aleatoria simple. Si $(x_1,...,x_n)$ es una realización muestral, se obtiene una estimacion de L:

$$L\left(x_1,...,x_n;\vec{\theta}\right)=\prod_{i=1}^{n} f\left(x_i;\vec{\theta}\right)$$

$L\left(x_1,...,x_n;\vec{\theta}\right)$ es una función de $\vec{\theta}$ que se denomina función de verosimilitud de la muestra. Notemos que es posible obtener tantas funciones de verosimilitud como realizaciones muestrales $(x_1,...,x_n)$.

***Definición 28 Estimación máximo verosímil* [29]**

Sea $L\left(x_1,...,x_n;\vec{\theta}\right)$ una función de verosimilitud de la muestra aleatoria simple $(X_1,...,X_n)$ de una variable aleatoria *X* que tiene un parámetro poblacional $\vec{\theta}$. La estimación máximo verosímil del parámetro $\vec{\theta}$ es el valor $\hat{\vec{\theta}}$ que cumple

$$\prod_{i=1}^{n} f\left(x_i;\hat{\vec{\theta}}\right)=\max_{\vec{\theta}}\left\{\prod_{i=1}^{n} f\left(x_i;\vec{\theta}\right)\right\}$$

Por sencillez, $\prod_{i=1}^{n} f\left(x_i;\vec{\theta}\right)$ se sustituye por la función $\log\left(\prod_{i=1}^{n} f\left(x_i;\vec{\theta}\right)\right)=\sum_{i=1}^{n}\log\left(f\left(x_i;\vec{\theta}\right)\right)$ y $\hat{\vec{\theta}}$ debe ser una solución del sistema de ecuaciones

$$\frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{n}\log\left(f\left(x_i;\vec{\theta}\right)\right)\right)=0 \quad j\in\{1,...,m\}$$

siendo *m* la dimensión del parámetro $\vec{\theta}$. El valor $\hat{\vec{\theta}}$ calculado es efectivamente un máximo si cumple

$$\frac{\partial^2}{\partial\theta_j^2}L\left(x_1,...,x_n;\hat{\vec{\theta}}\right)<0$$

Apéndice 3. Conceptos de la teoría de la información

La bibliografía consultada para elaborar este Apéndice ha sido fundamentalmente [14]; en aquellos casos en que no haya sido así, se ha indicado expresamente la fuente correspondiente.

La Teoría de la Información es una disciplina matemática surgida a mitad del siglo XX en el ámbito de la teoría de las comunicaciones para dar respuesta a problemas del tipo: ¿cómo enviar un mensaje por un canal ruidoso, es decir, un canal que distorsiona el mensaje, de manera que la distorsión haya sido mínima al concluir la transmisión? En la actualidad, existen dos enfoques de esta teoría: clásico o probabilístico, cuyo concepto clave es la entropía de Shannon, y algorítmico, basado en la complejidad de Kolmogorov. En este trabajo sólo usaremos conceptos pertenecientes al primer enfoque.

Para la teoría de la información, un mensaje es un conjunto de símbolos pertenecientes a un alfabeto. El mensaje es emitido por una fuente, viaja por un canal, y es recogido por un receptor. Al emisor se le asocia un espacio probabilístico (el espacio muestral es el alfabeto) cuya ley de probabilidades no varía con el tiempo (es decir, el emisor constituye una cadena de Markov ergódica [136]). La información de los mensajes caracteriza la incertidumbre asociada a su formación: un mensaje es más informativo cuanto menor es la probabilidad de su creación, y viceversa. Como se puede ver, los conceptos esenciales de la teoría de la información clásica, si se desvinculan de su ámbito de nacimiento, quedan como simples conceptos pertenecientes a la teoría de las probabilidades.

Definición 1 Información

La información asociada a un espacio probabilístico (Ω, A, p) es una función:

$$\begin{aligned} I : A &\longrightarrow \mathbb{R} \\ a &\longrightarrow I(a) = -K \log_b \left(p(a) \right) \end{aligned} \qquad \text{(I)}$$

donde *K, b* son constantes positivas. *b* es la base del logaritmo.

Una función información cumple tres propiedades: sean dos sucesos $a, b \in A$

1. $I(a) \geq 0$

2. Si $p(a) \leq p(b)$, entonces $I(a) \geq I(b)$. De aquí se deduce que $I(a \cap b) \geq I(a)$ y $I(a \cap b) \geq I(b)$.

3. Si *a,b* independientes, entonces $I(a \cap b) = I(a) + I(b)$

Se demuestra que las únicas funciones que cumplen las propiedades 1., 2. y 3. son las que tienen la expresión (I). Ha de notarse que (I) establece una relación inversa entre probabilidad e información de los sucesos: a mayor probabilidad, menor información, y viceversa.

Según (I), la probabilidad *p* correspondiente a un valor 1 de información depende de los valores *K* y *b*. En lo que sigue se va a considerar siempre *K=1*:

$$I(a) = -\log_b\left(p(a)\right)$$

Si se escoge $b = 2$, un valor 1 de información corresponde a $p = 1/2$; en este caso, a la unidad de información se le denomina bit; si $b = 3$, la unidad de información corresponde a $p = 1/3$, y se denomina trit, etc. El valor de *b* se escoge en función del espacio probabilístico asociado al fenómeno; por ejemplo, en teoría de las comunicaciones digitales, la unidad de información más usual es el bit, ya que el emisor de los mensajes se modela por una variable aleatoria con dos valores posibles equiprobables. Por simplicidad, en el resto de este Apéndice no se va a hacer mención explícita de *b*, dado que su valor no influye en el desarrollo de la teoría.

Definición 2 Entropía de Shannon

Sea $X \sim p_X$ la variable aleatoria discreta $X = \{x_1, x_2, ..., x_n\}$ con función de probabilidad p_X. Se define la entropía de Shannon como el promedio del contenido de información de *X*:

$$H_{p_X}(X) = \mathrm{E}\left(I(X)\right) = -\sum_{j=1}^{n} p_j \log\left(p_j\right)$$

Se toma el convenio $0 \cdot \log(0) = 0$.

Notemos que la entropía de Shannon depende tanto de *X* como de p_X, lo cual se pone de manifiesto en la notación. Normalmente, cuando está claro cual es p_X, al denotar la entropía se prescinde del subíndice, quedando $H(X)$. Así haremos a partir de ahora.

Por ejemplo, la entropía de la variable aleatoria discreta uniforme es:

$$H(X) = -\sum_{j=1}^{n} \frac{1}{n} \cdot \log\left(\frac{1}{n}\right) = \sum_{j=1}^{n} \frac{1}{n} \cdot \log(n) = \frac{1}{n} \cdot \log(n) \cdot \sum_{j=1}^{n} 1 = \log(n)$$

La entropía también se puede definir para variables aleatorias continuas. En este caso se denomina entropía continua o diferencial.

Definición 3 Entropía diferencial

Sea $X \sim f_X$ la variable aleatoria continua *X* con función de densidad f_X. Se define la entropía continua o diferencial como el promedio del contenido de información de *X*:

$$H(X) = -\int_X f_X(x) \cdot \log\left(f_X(x)\right) dx = E\left(\log\left(\frac{1}{f_X(X)}\right)\right)$$

La entropía diferencial no es una simple extensión de la entropía de Shannon, pues no cumple algunas de sus propiedades. Así, aunque la entropía de Shannon siempre es mayor o igual que cero, la entropía diferencial puede ser menor que cero; por ejemplo: si $X \sim U\left[0,1/2\right]$, entonces $f_X\left(x\right)=2$ y su entropía es

$$H\left(X\right)=-\int_0^{1/2} 2\cdot\log\left(2\right)dx=-\log\left(2\right)<0$$

La entropía de Shannon presenta la siguiente propiedad importante:

Proposición 1

Sea $X=\left\{x_1,...,x_n\right\}$ una variable aleatoria discreta. Se cumple:

$$0\le H\left(X\right)\le\log\left(n\right)$$

Además, $H\left(X\right)=0$ si y sólo si $\exists j\in\left\{1,...,n\right\}\setminus\ \ p\left(x_j\right)=1$

Este resultado afirma que la entropía de Shannon es nula en modelos deterministas y máxima cuando la distribución es uniforme. En otras palabras, una variable aleatoria discreta tiene más entropía cuanto más repartida esté la probabilidad entre sus elementos. Por lo tanto, la entropía de Shannon es una medida del grado de equidistribución de la probabilidad entre los puntos de una variable aleatoria.

En el caso de la entropía diferencial, la población normal desempeña el mismo papel que la población uniforme en la entropía de Shannon:

Proposición 2

Si *X* es una variable aleatoria continua se cumple

$$H\left(X\right)\le H_N$$

siendo $H_N=\log\left(\sigma\sqrt{2\pi e}\right)$ la entropía diferencial de la distribución normal $N\left(\mu,\sigma\right)$. La igualdad se cumple únicamente si $X\sim N\left(\mu,\sigma\right)$.

Es importante notar que la entropía de $N\left(\mu,\sigma\right)$ no depende de la media μ. La razón es simple: la entropía mide la "concentración" de probabilidad, que sólo depende de la desviación típica σ; la media μ sólo sirve para trasladar la función de densidad a lo largo del eje de abscisas.

La entropía de Shannon permite asignar probabilidades a variables aleatorias discretas cuando no se dispone de toda la información estadística necesaria. Para ello se usa el siguiente principio:

Principio de máxima entropía

> A una variable aleatoria discreta X con una función de probabilidad desconocida, siempre se le asignarán aquellas probabilidades que maximicen la entropía $H(X)$, dadas unas ciertas restricciones que se obtienen del conocimiento que se posee de la variable aleatoria.

Por ejemplo, sea X una variable aleatoria con rango $\{x_1,...,x_n\}$ y $p_X = \{p_1,...,p_n\}$ su función de probabilidad, de la cual se conoce la media $\mathrm{E}(X) = E$ pero no sus valores p_i. El principio de máxima entropía afirma que los valores p_i deberían ser tales que se cumpla el siguiente problema de extremos condicionados:

$$\max\ H(X)$$

sujeto a las restricciones

$$\left.\begin{aligned} \sum_{i=1}^{n} p_i &= 1 \\ \sum_{i=1}^{n} x_i p_i &= E \end{aligned}\right\}$$

Se demuestra que la solución es una función de probabilidad de la forma:

$$p_i = \frac{e^{\mu x_i}}{Z(\mu)} \qquad i \in \{1,...,n\}$$

donde

$$Z(\mu) = \sum_{i=1}^{n} e^{\mu x_i} \quad \text{(función de partición)}$$

μ se obtiene resolviendo numéricamente la ecuación

$$\frac{\sum_{i=1}^{n} x_i e^{\mu x_i}}{\sum_{i=1}^{n} e^{\mu x_i}} = E$$

A la función de probabilidad solución se le denomina función de distribución de Gibbs.

Si la única restricción es $\sum_{i=1}^{n} p_i = 1$, la solución es la población uniforme

$$p_i = \frac{1}{n} \quad i \in \{1, ..., n\}$$

que es la misma función de probabilidad que postula el Principio de Simetría o de la Razón Insuficiente. Por lo tanto, el Principio de Máxima Entropía puede considerarse una generalización del principio de simetría.

Definición 4 Funciones de probabilidad informativas

Sea una variable aleatoria X discreta y dos funciones de probabilidad P_1 y P_2 de X. Se dice que P_1 es más informativa que P_2 si

$$H_{P_1}(X) \leq H_{P_2}(X)$$

Por lo tanto, cuanto más informativa es la función de probabilidad que modela un fenómeno, menos incertidumbre hay respecto a sus resultados; es decir, mejor se conoce el fenómeno.

A continuación, se definen una serie de extensiones del concepto de entropía que involucran dos variables aleatorias:

Definición 5 Entropía conjunta

Sean X e Y dos variables aleatorias discretas y (X,Y) la variable aleatoria bidimensional discreta producto cartesiano de ambas. Se define la entropía conjunta de X e Y como

$$H(X,Y) = -\sum_{j=1}^{n}\sum_{k=1}^{m} p_{jk} \log\left(p_{jk}\right)$$

Se cumple $H(X,Y) = H(Y,X)$. Además, se puede demostrar $H(X,Y) \leq H(X) + H(Y)$, con igualdad si y sólo si *X* e *Y* independientes.

Definición 6 Entropía de Y condicionada a $X = x_i$

Denominemos $p_i(k) = p(Y = y_k \mid X = x_i)$. La entropía de *Y* condicionada a *X* se define:

$$H_i(Y) = -\sum_{k=1}^{m} p_i(k) \log\left(p_i(k)\right)$$

A partir de los valores $H_i(Y)$ se puede construir la variable aleatoria discreta $H.(Y)$ con rango $\{H_1(Y),...,H_n(Y)\}$ y función de probabilidad $\{p_1,...,p_n\}$.

Definición 7 Entropía de Y condicionada por X

Se denomina entropía de Y condicionada por X al valor, denotado $H_X(Y)$, definido por la expresión

$$H_X(Y) = E(H.(Y)) = \sum_{i=1}^{n} p_i H_i(Y)$$

$H_X(Y)$ también recibe el nombre de equivocación.

$H_i(Y)$ mide la incertidumbre de la variable aleatoria *Y*, cuando se conoce el valor $X = x_i$; $H_X(Y)$ es una medida de la incertidumbre de *Y* si no se conoce el valor de X.

Proposición 3

Se demuestra que:

1. $H_X(Y) = -\sum_{j=1}^{n}\sum_{k=1}^{m} p_{jk} \log(p_j(k))$ (comparar con la definición $H(X,Y)$ para ver la diferencia)
2. $H(X,Y) = H(X) + H_X(Y)$ $\quad H(X,Y) = H(Y) + H_Y(X)$
3. *X* e *Y* independientes si y sólo si $H(X,Y) = H(X) + H(Y)$
4. $H_Y(X) \le H(X)$ $\quad H_X(Y) \le H(Y)$ (la igualdad si y sólo si *X* e *Y* independientes)

El punto 2. establece que $H_X(Y)$ mide el contenido de información de *Y* que no está contenido en *X*. Igualmente, $H_Y(X)$ mide el contenido de información de *X* que no está contenido en *Y*.

Definición 8 Información mutua de X e Y

Se denomina información mutua de las variables aleatorias discretas *X* e *Y*, y se denota $I(X,Y)$, a la cantidad

$$I(X,Y) = H(Y) - H_X(Y)$$

Proposición 4

La información mutua de dos variables aleatorias discretas *X* e *Y* cumple:

1. $I(X,Y)=\sum_{j=1}^{n}\sum_{k=1}^{m} p_{jk}\log\left(\frac{p_{jk}}{p_j p_k}\right)$

   (Nota: se toma el convenio $0\cdot\log(0/q)=0$, y $p\cdot\log(p/0)=\infty$)

2. $I(X,Y)=I(Y,X)$

3. Si X e Y independientes, $I(X,Y)=0$

Por último, se va a definir una "medida" de la diferencia entre dos funciones de probabilidad asociadas a la misma variable aleatoria, relacionada con el concepto de entropía.

Definición 9 Distancia de Kullback-Leibler [31]

Dadas dos funciones de probabilidad P_X y Q_X asociadas a la misma variable aleatoria discreta *X*, se define la distancia de Kullback-Leibler, denotada por $D_{k-L}(P,Q)$, por la expresión:

$$D_{k-L}(P,Q)=\sum_{i=1}^{n} p(x_i)\log\left(\frac{p(x_i)}{q(x_i)}\right)$$

$D_{k-L}(P,Q)$ verifica:

1. $D_{k-L}(P,Q)\geq 0$

2. $D_{k-L}(P,Q)=0$ si y sólo si $P=Q$

D_{k-L} determina la diferencia entre las funciones de probabilidad P y Q, pero no es una verdadera medida, puesto que no cumple la propiedad de simetría: en general, $D_{k-L}(P,Q)\neq D_{k-L}(Q,P)$.

Proposición 5 [31]

Sea (X,Y) variable aleatoria bidimensional discreta, con una función de probabilidad $T=\{t_{ij};\ i\in\{1,...,n\},\ j\in\{1,...,m\}\}$. A partir de ella se define la función

de probabilidad $P = \left\{ p_i = \sum_{j=1}^{m} t_{ij};\ i \in \{1,...,n\} \right\}$ de X y la función de probabilidad $Q = \left\{ q_j = \sum_{i=1}^{n} t_{ij};\ \mathrm{j} \in \{1,...,m\} \right\}$ de *Y*. PQ es la función de probabilidad bidimensional $PQ = \left\{ p_i q_j;\ i \in \{1,...,n\},\ \mathrm{j} \in \{1,...,m\} \right\}$. La expresión

$$D_{k-L}(T, PQ) = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \log\left(\frac{t_{ij}}{p_i q_j} \right)$$

cumple

$$I(X, Y) = D_{k-L}(T, PQ)$$

La demostración es inmediata a partir de la proposición 4.1.

Si *X* es variable aleatoria continua, también se puede definir la distancia de Kullback-Leibler haciendo uso de las funciones de densidad:

$$D_{k-L}(f, g) = \int_X f(x) \log\left(\frac{f(x)}{g(x)} \right) dx$$

donde *f,g* funciones de densidad de X.

Definición 10 Entropía cruzada

Dadas dos funciones de probabilidad P y Q de la misma variable aleatoria discreta *X*, se define la entropía cruzada como

$$H_C(P, Q) = H(P) + D_{K-L}(P, Q)$$

donde $H(P) \equiv H_P(X)$ denota la entropía de la variable aleatoria *X* usando la función de probabilidad *P*.

Se demuestra:

$$H_C(P, Q) = -\sum_i P_i \log Q_i \qquad \text{(II)}$$

El concepto también se puede extender a variables aleatorias continuas:

$$H_C(f, g) = -\int_X f(x) \log g(x) dx$$

Apéndice 4. El cálculo de Wirtinger [11]

Sea

$$\begin{array}{rccc} g: & \mathbb{C} & \longrightarrow & \mathbb{C} \\ & \mathrm{z} & \longrightarrow & u(\mathrm{z})+iv(\mathrm{z}) \end{array}$$

una función compleja de variable compleja, y supongamos que esta función se puede expresar, a su vez, en la forma

$$g(\mathrm{z}) = f\left(\mathrm{z}, z^*\right)$$

con z^* el complejo conjugado de *z*.

Se definen

$$\boxed{\begin{array}{l} \dfrac{\partial f}{\partial z} \triangleq \dfrac{1}{2}\left(\dfrac{\partial f}{\partial x} - i\dfrac{\partial f}{\partial y}\right) \quad (1) \\ \\ \dfrac{\partial f}{\partial z^*} \triangleq \dfrac{1}{2}\left(\dfrac{\partial f}{\partial x} + i\dfrac{\partial f}{\partial y}\right) \quad (2) \end{array}}$$

donde las derivadas $\partial f/\partial x$, $\partial f/\partial y$ significan:

$$\frac{\partial f}{\partial x} = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial y} = \frac{\partial u}{\partial y} + i\frac{\partial v}{\partial y}$$

Las derivadas $\partial f/\partial z$ y $\partial f/\partial z^*$ se pueden calcular siempre, con independencia de la holomorfía o no de f; de ahí su interés.

Si *f* es holomorfa, cumple las condiciones de Cauchy-Riemann:

$$\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y} = 0$$

luego, teniendo en cuenta (2)

$$\frac{\partial f}{\partial z^*} = 0$$

Esto significa las funciones holomorfas no dependen de z^*.

El concepto de derivada de Wirtinger se puede considerar una ampliación del concepto de derivada compleja tradicional. Las derivadas de Wirtinger satisfacen todas las reglas del análisis complejo (regla de la conjugación, de la cadena, del diferencial, etc), dando lugar al cálculo de Wirtinger.

Apéndice 5. Longitud de arco de curva invariante a afinidades

En este Apéndice[16] se deduce la expresión de la longitud de arco de curva (plana) afín invariante a partir del concepto de frame móvil. Un frame es simplemente un conjunto de vectores $\{a_1,...,a_n\}$ expresados en forma de matrices columna. El concepto de frame es una herramienta muy útil para calcular los invariantes de curvas en distintas geometrías, entendiendo geometría en el sentido de Klein (conjunto de invariantes a grupos de transformación).

En general, una afinidad tiene la expresión matricial $x' = Mx + T$, con $\|M\| \neq 0$ ($\|\cdot\|$ denota determinante). En la discusión que sigue se va a suponer que la afinidad que se aplica a la curva cumple:

- Conserva áreas, es decir, $\|M\| = 1$ o lo que es lo mismo $M \in SL_2(\mathbb{R})$. Más adelante se verá la razón de esta suposición.

- No genera traslaciones, es decir, $T = 0$. Esta restricción evita la necesidad de tener en cuenta en los cálculos el efecto de la traslación, toda vez que este efecto siempre se puede soslayar tomando como origen del sistema coordenado el centroide de la curva.

Sea una curva plana $x(t) = x_1(t)e_1 + x_2(t)e_2$, con t una parametrización y e_1, e_2 dos vectores de $\mathbb{R}^2$. Se definen dos tipos de frames asociados a la curva, fijos y móviles, según dependan del parámetro de la curva o no, respectivamente. $\{e_1, e_2\}$ es un frame fijo, y $\{a_1(t), a_2(t)\}$ uno móvil; los vectores de ambos se relacionan por el sistema:

$$\left.\begin{aligned} a_1(t) &= a_{11}(t)e_1 + a_{12}(t)e_2 \\ a_2(t) &= a_{21}(t)e_1 + a_{22}(t)e_2 \end{aligned}\right\}$$

o lo que es lo mismo

$$\{a_1(t), a_2(t)\} = A(t)\{e_1, e_2\}$$

siendo

$$\{a_1(t), a_2(t)\} \equiv \begin{bmatrix} a_1(t) \\ a_2(t) \end{bmatrix}$$

[16] La Bibliografía consultada para elaborar este Apéndice es [54]

$$\{e_1, e_2\} \equiv \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

$$A(t) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$ Matriz asociada al frame móvil

Se supone que $\|A\| \neq 0$ y $A(t)$ es diferenciable (es decir, cada una de sus componentes es diferenciable). Notar que la expresión de $A(t)$ depende tanto del parámetro *t* como indirectamente del frame fijo.

En la figura se representan ambos frames:

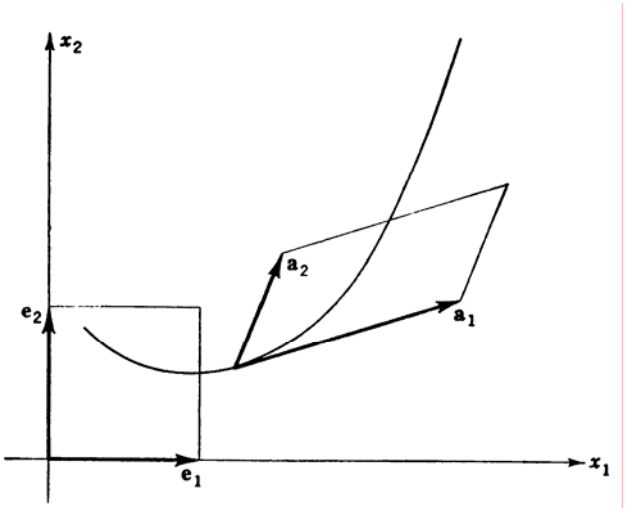


Figura A.1. El frame móvil y el frame fijo de una curva.

En general, si se aplica una transformación afín caracterizada por la matriz *M*, $\{e_1, e_2\}$ y su transformado $\{e'_1, e'_2\}$ se relacionan mediante:

$$\{e_1, e_2\} = M^{-1}\{e'_1, e'_2\}$$

El frame móvil expresado en el nuevo frame fijo es:

$$\{a_1(t), a_2(t)\} = A(t)\{e_1, e_2\} = A(t)M^{-1}\{e'_1, e'_2\}$$

luego, la nueva matriz asociada al frame móvil es

$$A'(t) = A(t)M^{-1}$$

de la cual diremos que es la transformada por *M* de la matriz $A(t)$.

A la matriz del frame móvil $A(t)$ se le asocia la matriz de Cartan $C(A)(t)$. En general, las matrices de Cartan se definen de la siguiente forma:

Sea G un grupo de matrices cuadradas diferenciables no singulares. Se denomina matriz de Cartan de la matriz $A(t) \in G$ *a la matriz*

$$C(A)(t) = \frac{dA(t)}{dt} A(t)^{-1}$$

donde $dA(t)/dt = \left[da_{ij}(t)/dt \right]$.

Las matrices de Cartan tienen cuatro propiedades importantes: sean $A(t)$, $B(t)$, $M \in G$ (*M* es matriz constante), con *G* un grupo de matrices cuadradas diferenciables no singulares, entonces:

1. $C(AB)(t) = C(A)(t) + A(t)B(t)A(t)^{-1}$
2. Si $G = SL_n$, $C(A)(t)$ tiene nulos los elementos de la diagonal.
3. Si $G = O_n$, $C(A)(t)^t = -C(A)(t)$, es decir, $C(A)(t)$ es antisimétrica.
4. $C(M) = 0$
5. $C(AM)(t) = C(A)(t)$

Empleando el concepto de matriz de Cartan de una curva se puede demostrar la siguiente proposición:

Proposición

Sea $M \in G$, con un G un grupo de matrices invertibles, y sean $A(\sigma)$ y $A'(\sigma)$ las matrices asociadas a un frame móvil y su transformada por *M*, respectivamente; se cumple:

si $A(\sigma) \in G$, entonces σ es invariante de *G*

Demostración: si $A(\sigma) \in G$, entonces $A'(\sigma) = A(\sigma)M^{-1} \in G$ pues, por ser G un grupo, es cerrado al producto. En consecuencia, aplicando la propiedad 5. de las matrices de Cartan, se cumple $C(A')(\sigma) = C(A)(\sigma)$ y el parámetro σ resulta ser invariante del grupo de transformaciónes *G*.

Apliquemos lo anterior para obtener la expresión de la longitud de arco de curva afín-invariante de una curva paramétrica $x(\sigma)$. El grupo de transformación asociado a las afinidades *M* que conservan áreas es $G = SL_2(\mathbb{R})$; busquemos, entonces, una matriz $A(\sigma) \in SL_2(\mathbb{R})$.

Elegir la matriz $A(\sigma)$ es lo mismo que elegir el frame móvil $\{a_1(\sigma), a_2(\sigma)\}$, lo cual a su vez es lo mismo que elegir el parámetro σ; entonces, se escoge σ tal que:

$$a_1(\sigma) = \frac{\dot{x}(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}}$$

$$a_2(\sigma) = \frac{\ddot{x}(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}}$$

con

$$\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\| = \left\| \begin{bmatrix} \dot{x}_1(\sigma) & \ddot{x}_1(\sigma) \\ \dot{x}_2(\sigma) & \ddot{x}_2(\sigma) \end{bmatrix} \right\| = \dot{x}_1(\sigma)\ddot{x}_2(\sigma) - \dot{x}_2(\sigma)\ddot{x}_1(\sigma)$$

que suponemos es distinto de cero.

Por lo tanto

$$A(\sigma) = \begin{bmatrix} \dfrac{\dot{x}_1(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}} & \dfrac{\ddot{x}_1(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}} \\ \dfrac{\dot{x}_2(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}} & \dfrac{\ddot{x}_2(\sigma)}{\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}} \end{bmatrix}$$

que cumple

$$\left\|A(\sigma)\right\| = \frac{1}{\left(\left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\|^{1/2}\right)^2} \left\|\dot{x}(\sigma), \ddot{x}(\sigma)\right\| = 1$$

luego, $A(\sigma) \in SL_2(\mathbb{R})$, y es la matriz buscada. Así pues, por la Proposición, σ es invariante al grupo de transformación $G = SL_2(\mathbb{R})$, es decir, es independiente de cualquier transformación afín (que conserve áreas) aplicada a la curva. En definitiva, σ es la longitud de arco afín invariante.

Para integrar σ es necesario trabajar con la matriz de Cartan de $A(\sigma)$:

$$C(A)(\sigma)=\begin{bmatrix} -\frac{1}{2}\frac{\|\dot{x},\ddot{x}\|}{\|\dot{x},\ddot{x}\|} & 1 \\ -\frac{\|\ddot{x},\dddot{x}\|}{\|\dot{x},\ddot{x}\|} & \frac{1}{2}\frac{\|\dot{x},\dddot{x}\|}{\|\dot{x},\ddot{x}\|} \end{bmatrix}$$

Los valores de la diagonal de $C(A)(\sigma)$ son nulos por la propiedad 2. de las matrices de Cartan; entonces:

$$\|\dot{x},\ddot{x}\|=0$$

Desarrollando $\|\dot{x},\ddot{x}\|$:

$$\|\dot{x},\ddot{x}\|=\left\|\begin{bmatrix} \dot{x}_1 & \ddot{x}_1 \\ \dot{x}_2 & \ddot{x}_2 \end{bmatrix}\right\|=\dot{x}_1\ddot{x}_2-\dot{x}_2\ddot{x}_1=\left(\dot{x}_1\ddot{x}_2+\ddot{x}_1\ddot{x}_2\right)-\left(\ddot{x}_1\ddot{x}_2+\dot{x}_2\ddot{x}_1\right)=\left(\dot{x}_1\ddot{x}_2\right)'-\left(\dot{x}_2\ddot{x}_1\right)'=$$

$$=\left(\dot{x}_1\ddot{x}_2-\dot{x}_2\ddot{x}_1\right)'=\|\dot{x},\ddot{x}\|'$$

Por tanto:

$$\|\dot{x},\ddot{x}\|'=0$$

Integrando:

$$\|\dot{x}(\sigma),\ddot{x}(\sigma)\|=k \quad k\in\mathbb{R}$$

Se elige $k=1$. σ se puede expresar en función de cualquier otro parámetro *t*:

$$\left\|\frac{dx}{dt},\frac{d^2x}{dt^2}\right\|=\left\|\frac{dx}{d\sigma}\cdot\frac{d\sigma}{dt},\frac{d^2x}{d\sigma^2}\cdot\left(\frac{d\sigma}{dt}\right)^2\right\|=\left\|\frac{dx}{d\sigma},\frac{d^2x}{d\sigma^2}\right\|\cdot\left(\frac{d\sigma}{dt}\right)^3=1\cdot\left(\frac{d\sigma}{dt}\right)^3=\left(\frac{d\sigma}{dt}\right)^3$$

luego

$$\sigma=\int_{t_0}^{t}\|\dot{x}(\upsilon),\ddot{x}(\upsilon)\|^{1/3}\,d\upsilon$$

Como σ depende de la derivada segunda respecto al parámetro general, es un invariante afín de segundo orden.

Es importante recordar que en toda la discusión anterior se ha supuesto $M \in SL_2(\mathbb{R})$, es decir, que la afinidad conserva áreas; por lo tanto, la diagonal de $C(A)(\sigma)$ es nula y es fácil obtener el invariante σ. Si la afinidad es general, $M \in GL_2(\mathbb{R})$ (es decir, $\|M\| \neq 0$) y resulta más complicado obtener la expresión de σ.
Por lo tanto, si σ es la longitud de arco afín invariante se cumple:

$$\|\dot{x}(\sigma), \ddot{x}(\sigma)\| = 1$$

luego

$$A(\sigma) = \begin{bmatrix} \dot{x}_1(\sigma) & \ddot{x}_1(\sigma) \\ \dot{x}_2(\sigma) & \ddot{x}_2(\sigma) \end{bmatrix}$$

y

$$C(A)(\sigma) = \begin{bmatrix} 0 & 1 \\ -\|\ddot{x}, \dddot{x}\| & 0 \end{bmatrix}$$

$\chi(\sigma) = \|\ddot{x}(\sigma), \dddot{x}(\sigma)\|$ se denomina curvatura afín, y teniendo en cuenta la propiedad 5. de las matrices de Cartan, es un invariante afín. Si χ se expresa en función de un parámetro cualquiera *t*, se puede demostrar que es un invariante de cuarto grado.

Bibliografía

[1] Abbasi, S., and Mokhtarian, F.: Robustness of shape similarity retrieval under affine transformation, *Proceedings of Challenge of Image Retrieval*, Newcastle upon Tyne, UK, 1999.

[2] Abbasi, S., and Mokhtarian, F.: Matching shapes with self-intersections: application to leaf classification, *IEEE Transactions on Image Processing*, Vol. 13, Issue 5, pp. 653-661, IEEE, 2004.

[3] Abbasi, S.; Mokhtarian, F, and Kittler, J.: Curvature scale-space image in shape similarity retrieval, *Multimedia Systems*, Vol. 7, pp. 467-476, Springer-Verlag, 1999.

[4] Acharya, T., and Ray, A. K.: *Image processing: principles and applications*, Wiley-Interscience, 2005.

[5] Aizenberg, I.: Complex-valued neural networks with multi-valued neurons, *Studies in Computational Intelligence*, Vol. 353, Springer, 2011.

[6] Aizenberg, N.N.; Ivaskiv, Y.L. and Pospelov, D.A.: About one generalization of the threshold function, *Doklady Akademii Nauk SSSR (The Reports of the Academy of Sciences of the USSR)* 196(6), pp. 1287-1290, 1971 (en ruso)

[7] Al-Rawi, M. S.: Learning affine invariant pattern recognition using high-order neural networks, *AIML 06 International Conference*, Sharm El Sheikh, Egypt, 2006.

[8] Ali, A.; Gilani, S., and Memon, N.: Affine invariant contour descriptors using independent component analysis and dyadic wavelet transform, *Journal of Computing and Information Technology (CIT)*, Vol. 16, No 3, pp. 169-181, 2008.

[9] Alonso, D., y Tubau, E.: Inferencias bayesianas: una revisión teórica, *Anuario de Psicología*, Vol. 33, No. 1, pp. 25-47, Facultat de Psicología, Universitat de Barcelona, 2002.

[10] Amari, S.-I.: Forty years of perceptrons, *ICONIP proceedings*, 1, pp. 3-7, International Conference on neural information processing, 2001.

[11] Amin, Md. F.: Complex-valued neural networks: learning algorithms and applications, *Dissertation for the degree of Doctor of Engineering*, Department of System Design Engineering, University of Fukui, 2012.

[12] Amin, Md. F.; Islam, Md. M. and Islam, K.: Ensemble of single-layered complex-valued neural networks for classification tasks, *Neurocomputing*, Vol. 72, Issues 10-12, pp. 2227-2234, Elsevier, 2009

[13] Amin, Md. F. and Murase, K.: Single-layered complex-valued neural network for real-valued classification problems, *Neurocomputing*, Vol. 72, Issues 4-6, pp. 945-955, Elsevier, 2009

[14] Applebaum, D.: *Probability and information: an integrated approach*, second edition, Cambridge University Press, 2008.

[15] Arbter, K.; Snyder, W. E.; Burkhardt, H., and Hirzinger, G.: Application of affine-invariant Fourier descriptors to recognition of 3-D objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 12, Issue: 7, pp. 640-647, 1990.

[16] Arora, R; Hu, Y. H. and Dyer, C.: Estimating correspondence between multiple cameras using joint invariants, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, pp.805-808, IEEE, 2009

[17] Attneave, F.: Some informational aspects of visual perceptions, *Psychological Review*, Vol. 61, Issue 3, pp. 183-193, 1954.

[18] Barto, A.G.; Sutton R.S. and Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, pp. 834-846, IEEE, 1983

[19] Bebis, G.; Georgiopoulos, M.; Da Vitoria Lobo, N., and Shah, M.: Learning affine transformation of the plane for model-based object recognition, *Proceedings of the 13th International Conference on Pattern Recognition*, Vol. 4, pp. 60-64, Vienna, IEEE, 1996.

[20] Bebis, G.; Papadourakis, G., and Orphanoudakis, S.: Recognition using curvature scale space and artificial neural networks, *Proceedings of the IASTED International Conference of Signal and Image Processing*, Las Vegas, NV, USA, 1998.

[21] Ben-David, S.; Cesa-Bianchi, N.; Haussler, D., and Long, P. M.: Characterizations of learnability for classes of {0,…,n}-valued functions, *Journal of Computer and System Sciences*, Vol. 50, no. 1, pp. 74-86, 1995.

[22] Bhowmick, P., and Bhattacharya, B. B.: CODE: an adaptive algorithm for detecting corners and directions of incident edges, *Proceedings of the Fourth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP 2004)*, pp. 509-515, Kolkata, India, 2004.

[23] Bishop, Ch. M.: *Neural networks for pattern recognition*, Oxford University Press, Oxford, UK, 1995.

[24] Boser, B.E.; Guyon, I.M. and Vapnik, V.N.: A training algorithm for optimal margin classifiers, *Fifth Annual Workshop on Computational Learning Theory*, pp. 144-152, Morgan Kaufmann, San Mateo, CA, USA, 1992

[25] Broomhead, D.S. and Lowe, D.: Multivariate functional interpolation and adaptive networks, *Complex Systems*, Vol. 2, Issue 3, pp. 321-355, Complex Systems Publications Inc., USA, 1988

[26] Brunet, F.; Bartoli, A.; Navab, N.; Malgouyres, R.: NURBS warps, *British Machine Vision Conference*, 2009

[27] Buchholz, S. and Sommer, G.: On Clifford neurons and Clifford multi-layer perceptrons, *Neural Networks*, Vol. 21, Issue 7, pp. 925-935, Elsevier, 2008

[28] Buesching, D.: Efficiently finding bitangents, *Proceedings of the 13th International Conference on Pattern Recognition*, Vol. 1, pp. 428-432, IEEE, 1996

[29] Canavos, G. C.: *Probabilidad y estadística: aplicaciones y métodos*, McGraw-Hill/Interamericana de Mexico, 1988.

[30] Castillo, E.; Cobo, A.; Gutiérrez, J. M., y Pruneda, R. E.: *Introducción a las redes funcionales con aplicaciones: un nuevo paradigma neuronal*, Editorial Paraninfo, Madrid, España, 1999.

[31] Cortés de la Fuente, J.: La información mutua como medida de asociación y su utilidad en análisis genéticos, *Trabajo final del Máster Interuniversitario en Estadística e Investigación Operativa, Facultat de Matemàtiques i Estadística*, Universitat Politècnica de Catalunya, 2011.

[32] Cover, T.M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Transactions on Electronic Computers*, EC-14, pp. 326-334, 1965

[33] DeTurck, D.; Gluck. H.; Pomerleano, D., and Vick, D. S.: The four vertex theorem and its converse, *Notice of the AMS*, Vol. 54, No. 2, pp. 192-207, 2007.

[34] De Vries, J.: Object recognition: a shape-based approach using artificial neural networks, *Thesis of Master*, Department of Computer Science, University of Utrech, Holland, 2006.
[Available in http://www.ai.rug.nl/~mwiering/ObjectRecognition.pdf]

[35] Di Ruberto, C., and Morgera, A.: A fast iterative method for dominant points detection of digital curves, *8th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'09)*, pp. 271-278, Cambridge, UK, 2009.

[36] Di Ruberto, C., and Morgera, A.: A new iterative approach for dominant points extraction in planar curves, *WSEAS Transactions on Computer*s, Vol. 8, No. 3, pp. 482-493, 2009.

[37] Di Ruberto, C.; Morgera, A., and Gaviano, M.: Shape matching by curve modelling and alignment, *WSEAS Transactions on Information Science and Applications*, Vol. 6, No. 4, pp. 567-578, 2009.

[38] Donahue, M. J., and Rokhlin, S.I.: On the use of level curves in image analysis, *CVGIP: Image Understanding Journal*, Vol. 57, Issue 2, pp. 185-203, Elsevier, 1993.

[39] Dutta, D.; Roy, A. and Choudhury, K.: Training artificial neural network using particle swarm optimization algorithm, *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, Issue 3, 2013

[40] Efimov, N.V.: *Geometría superior* , Editorial Mir, 1978

[41] Elhachloufi, M.; El Oirrak, A.; Aboutajdine, D., and Kaddioui, M. N.: Affine invariant descriptor and recognition of 3D objects using neural networks and principal component analysis, *Journal of Theoretical and Applied Information Technology*, Vol. 17, Issue 1/2, pp. 136, 2010.

[42] Fiori, S.: Extended hebbian learning for blind separation of complex-valued sources, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 50, Issue 4, pp. 195-202, IEEE, 2003

[43] Fiori, S.: Non-linear complex-valued extensions of hebbian learning: an essay, *Neural Computation*, Vol. 17, Issue 4, pp. 779-838, MIT Press, Cambridge, MA, USA, 2005

[44] Flusser, J.; Suk, T., and Zitov, B.: *Moments and moment invariants in pattern recognition*, John Wiley & Sons, 2009.

[45] Flusser, J.; Rahtu, E.; Salo, M., and Heikkila, J.: Generalized affine moment invariants for object recognition, *Proceedings of International Conference on Pattern Recognition*, Vol. 2, pp. 634-637, IEEE , Hong Kong, 2006.

[46] Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics*, Vol. 36, pp. 193-202, Springer-Verlag, 1980

[47] Gal, R.; Shamir, A., and Cohen-Or, D.: Pose-oblivious shape signature, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, Issue 2, pp. 261-271, IEEE, 2007.

[48] García, A., y Quesada, V.: *Lecciones de cálculo de probabilidades*, Diaz de Santos, 1988.

[49] García, A., y Vélez, R.: *Principios de inferencia estadística*, cuarta reimpresión, UNED, España, 2002.

[50] Georgiou, G. M., and Koutsougeras, C.: Complex domain backpropagation, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 39, Issue: 5, pp. 330-334, 1992.

[51] Goh, S. L. and Mandic, D. P.: An augmented extended Kalman filter algorithm for complex-valued recurrent neural networks, *Neural Computation*, Vol. 19, Issue 4, pp. 1039-1055, MIT Press, 2007

[52] Greenwood, M., and Oxspring, R.: The applicability of Occam's razor to neural network architecture, *Undergraduate coursework*, Department of Computer Science, The University of Sheffield, UK, 2001.

[53] Grossberg, S.: How does a brain build a cognitive code?, *Psychological Review*,Vol. 87, Number 1, APA, 1980

[54] Guggenheimer, H. W.: *Differential geometry*, Dover Publications, New York, USA, 1977.

[55] Hann, C. E.: Recognising two planar objects under a projective transformation, *Ph. D. University of Canterbury*, Christchurch, New Zealand, 2001

[56] Hann, C.E. and Hickman, M.S.: Projective curvature and integral invariants, *Acta Applicandae Mathematicae*, Vol. 74, Issue 2, pp. 177-193, University of Canterbury. Mechanical Engineering, 2002

[57] Hartley, R. and Zisserman, A.: *Multiple view geometry in computer vision* 2ª edition, Cambridge University Press, 2004

[58] Hassoun., M. H.: *Fundamentals of artificial neural networks*. MIT Press, online version:, 1995. [Disponible en http://neuron.eng.wayne.edu/]

[59] Haykin S.: *Neural networks: A comprehensive foundation*, second internacional edition, Prentice Hall International, London, UK, 1999.

[60] He, X. C., and Yung, N. H. C.: Corner detector based on global and local curvature properties, *Optical Engineering*, Vol. 47, No. 5, pp. 1-12, 2008.

[61] Hebb, D.: *The organization of behaviour*, Wiley & Sons Ltd., New York, USA, 1949

[62] Hecht-Nielsen, R.: Kolmogorov's mapping neural network existence theorem, *IEEE First Annual International Conference on Neural Networks*, Vol. 3, pp. 11-13, 1987

[63] Hirose, A.: *Complex-valued neural networks*, Springer, 2006.

[64] Hirose, A.: Continuous complex-valued back-propagation learning, *Electronics Letters*, Vol. 28, Issue: 20, pp. 1854-1855, 1992.

[65] Hirose, A. (editor): *Complex-valued neural networks: theories and applications*, Series on Innovative Intelligence, Vol. 5, World Scientific Publishing, 2003.

[66] Hirose, A.: Dynamics of fully complex-valued neural networks, *Electronic Letters*, Vol. 28, Issue 16, pp. 1492-1494, IET, 1992

[67] Holland, J.H.: *Adaptation in natural and artificial systems: an introductory analysis with applications in biology, control and artificial intelligence*, University of Michigan Press, 1975

[68] Hopfield, J.J.: Neural network and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, Vol. 79, pp. 2554-2558, USA, 1982

[69] Horn, B. and Schunck, B.: Determining optimal flow, *Artificial Intelligence*, Vol. 17, pp. 185-203, 1981

[70] Jähne, B.: *Digital image processing*, 6th revised and extended edition, Springer-Verlag, 2005.

[71] Karpinski, M. and Macintyre, A.: Polynomial bounds for VC-dimension of sigmoidal neural networks, *Research Report 85116-CS*, Univ. of Bonn, 1994

[72] Khan, R.; Pizarro, D. and Bartoldi, A.: Schwarps: Locally projective image warps based on 2D schwarzian derivatives, Computer Vision – *ECCV 2014, Lecture Notes in Computer Science*, Vol. 8692, pp. 1-15, Springer International Publishing, 2014

[73] Kim, T., and Adali, T.: "Approximation by fully complex multilayer perceptrons", *Neural Computation Journal*., Vol. 15, pp. 1641-1666, MIT Press, 2003.

[74] Kim, T., and Adali, T.: Complex backpropation neural networks using elementary trascendental activation functions, *Proceedings of IEEE International Conference on Acustics, Speech, Signal Processing (ICASSP)*, Vol. 2, Salt Lake City, UT, pp. 1281-1284, 2001.

[75] Koehn, P.: Combining genetic algorithms and neural networks: the encoding problem, *Thesis for the Master of Science Degree of The University of Tennessee*, Knoxville, USA, 1994

[76] Kohonen, T.: Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, Vol. 43, pp. 59-69, Springer-Verlag, 1982

[77] Kovic, L., and Matejic, M. M.: Contractive affine transformation of complex plane and applications, *Facta Universitatis Series Mathematics Informatics*, Vol. 21, pp. 65-75, University of Nis, Serbia, 2006.
[Disponible en http://facta.junis.ni.ac.rs/mai/mai21/f21-65-75.pdf]

[78] Kröse, B., and Van der Smagt, P.: *An introduction to neural networks*, eighth edition, University of Amsterdam, Amsterdam, Holland, 1996.

[79] Lago, L. F.; Sánchez-Montañés, M., y González, A.: Support vector machines, *Transparencias del Curso Métodos avanzados en inteligencia artificial*, Universidad Autónoma de Madrid, España, 2010.

[80] Lakehal, A.; El Beqqali, O., and Ait Zemzami, O.: Retrieval of similar shapes under affine transform using affine length parameterization, *Journal of Computer Science*, Vol. 6, Issue: 10, pp. 1226-1232, Science Publications, 2010.

[81] Lee, S.C. and Lee, E.T.: Fuzzy sets and neural networks, *Journal of Cybernetics*, Vol. 4, Nº 2, pp. 83-103, 1974

[82] Leung, H. and Haykin, S.: The complex backpropagation algorithm, *IEEE Transactions on Signal Processing*, Vol. 39, Issue 9, pp. 2101-2104, IEEE, 1991

[83] Li, W; Lee, T. and Tsui, H.: Automatic feature matching using coplanar projective invariants for object recognition, *Proceedings of the 5th Asian Conference on Computer Vision (ACCV 2002)*, pp. 247-252, Asian Federation of Computer Vision Societies, 2002

[84] Li, W. and Lee, T.: Hopfield network for affine invariant object recognition, *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE Congress on Computational Intelligence (IJCNN' 01)*, Vol. 1, pp. 588-593, IEEE, 2001

[85] Li, W. and Lee, T.: Projective invariant object recognition by a Hopfield network, *Neurocomputing*, Vol. 62, pp. 1-18, Elsevier B.V., 2004

[86] Londhe, R., and Pawar, V.: Facial expression recognition based on affine moment invariants, *IJCSI International Journal of Computer Science Issues*, Vol. 9, Issue 6, No 2, 2012.

[87] Maass, W.: Vapnik-Chervonenkis dimension of neural nets, *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib ed, pp. 1000-1003, MIT Press, 1995

[88] Macintyre, M. and Sontag, E.D:: Finiteness results for sigmoidal "neural" networks, *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, ACM Press, Nueva York, USA, 1993

[89] MacKay, D. J. C.: *Information theory, inference, and learning algorithms*, Cambridge University Press, Cambridge, UK, 2003.

[90] Martín del Brío, B., y Sáenz Molina A.: *Redes neuronales y sistemas borrosos*, tercera edición, Ra-Ma Editorial, Madrid, 2006.

[91] Martín, Q., y Del Rosario de Paz, Y.: *Aplicaciones de las redes neuronales artificiales a la regresión*, Cuadernos de Estadística, Editorial La Muralla, Madrid, 2007.

[92] Martínez Estudillo, A. C.: Modelos de regresión basados en redes neuronales de unidades producto diseñadas y entrenadas mediante algoritmos de optimización híbrida. Aplicaciones, *Tesis doctoral*, E. T. S. de Ingeniería Informática, Universidad de Granada, Granada, España, 2005.

[93] McCulloch, W.S. and Pitts, W.: A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943

[94] Minsky, M.L. and Papert, S.A.: *Perceptrons*, MIT Press, Cambridge, MA, USA, 1969

[95] Mokhtarian, F., and Mackworth, A. K.: A theory of multiscale, curvature-based shape representation for planar curves; *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 8, IEEE, 1992.

[96] Montana, D.J. and Davis, L.: Training feedforward neural networks using genetic algorithms, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 762-767, Morgan Kaufmann, USA, 1989

[97] Montes, F.: *Introducción a la probabilidad*, Departament d'Estadistica I Investigació Operativa, Universitat de València, 2007.
[Disponible en www.uv.es/montes/probabilitat/manual.pdf]

[98] Morgera, A.: Dominant points detection for shape analysis, *Thesis of Universty of Cagliary*, Italy, 2011.

[99] Neal, R.: Bayesian learning for neural networks, Ph.D. *Thesis*, Department of Computer Science, University of Toronto, 1994.

[100] Nitta, A.: A solution of the 4-bit parity problem with a single quaternary neuron, *Neural Information Processing – Letters and reviews*, Vol. 5, Nº 2, pp. 33-39, 2004

[101] Nitta, T.: "An extension of backpropagation algorithm to complex numbers", *Neural Networks Journal*, Vol. 10, no. 8, pp. 1392-1415, Elsevier Science, 1997.

[102] Nitta, T.: Learning transformations with complex-valued neurocomputing, *International Journal of Organizational and Collective Intelligence*, Vol. 3, Issue 2, pp. 81-116, IGI Publishing, Hershey, PA, USA, 2012.

[103] O'Neil, P. V.: *Matemáticas avanzadas para ingeniería: Análisis de Fourier, ecuaciones diferenciales parciales y análisis complejo*, quinta edición, Internacional Thomson Editores, 2004.

[104] Ozgur, E. and Unel, M.: Image based visual servoing using bitangent points applied to planar shape alignment, *Proceedings of the 13th IASTED International Conference on Robotics and Applications (RA' 07)*, pp. 275-280, ACTA Press Anaheim, CA. USA, 2007

[105] Pajares, G.; Cruz; J.M. and Aranda, J.: Relaxation by Hopfield network in stereo image matching, *Pattern Recognition*, Vol. 31, Issue 5, pp. 561-574, Elsevier, 1998

[106] Palit, A. K., y Popovic, D.: *Computational intelligence in time series forecasting: theory and engineering applications*, Springer-Verlag London Limited, London, 2005.

[107] Pazos A., y Dans, C. M.: Integración de las redes de neuronas artificiales con lógica difusa, *Redes de neuronas artificiales y algoritmos genéticos*, pp. 297-324, Departamento de Computación Universidade da Coruña, A Coruña, España, 1996.

[108] Pérez Ortiz, J. A.: *Clasificación con discriminantes: un enfoque neuronal*, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, España, 1999.

[109] Pilet, J.; Lepetit, V.; Fua, P.: Fast non-rigid surface detection, registration and realistic augmentation, *International Journal on Computer Vision*, Vol. 76, Issue 2, pp. 109-122, 2007

[110] Prieto, R.; Herrera, A.; Pérez, J. L., y Padrón, A.: *El modelo neuronal de McCulloch y Pitts: interpretación comparativa del modelo*, Laboratorio de computación adaptativa, UNAM, Mexico D. F., Mexico.

[111] Radcliffe, N.J.: Genetic set recombination and its application to neural network topology optimisation, *Neural Computing & Applications*, Vol. 1, Issue 1, pp. 67-90, Springer-Verlag, 1993

[112] Rojas R.: *Neural networks: A systematic introduction*, Springer-Verlag, Berlin, New-York, 1996.

[113] Ruecker, D.; Sonoda, L.I.; Hayes, C.; Hill, D.L.G.; Leach, M.O. and Hawkes, D.J.: Nonrigid registration using free-form deformations: applications to breast MR images, *IEEE Transactions on Medical Imaging*, Vol. 18, pp. 712-721, IEEE, 1999

[114] Rumelhart, D.E.; Hinton, G.E. and Williams, R.J.: Learning representations of back-propagation errors, *Nature (London)*, Vol. 323, pp. 533-536, Nature Publishing Group, 1986

[115] Rumelhart, D. E.; Hinton, G. E., and Williams, R. J.: "Learning internal representations by error propagation", *Parallel distributed processing: explorations in the microstructure of cognition*, Vol. 1, pp. 318-362, MIT Press, Cambrige, MA, USA, 1986.

[116] Rosenblatt, F.: The Perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, Vol. 65, Nº 6, pp. 832-837, APA, 1958

[117] Sánchez Camperos, E. N., y Alanís García, A. Y.: *Redes neuronales: conceptos fundamentales y aplicaciones a control automático*, Pearson Educación S. A., Madrid, España, 2006.

[118] Sánchez Pérez, J.; *Visión tridimensional*, Facultad de Informática, Universidad de Las Palmas de Gran Canaria.
[Disponible en http://ami.dis.ulpgc.es/biblio/bibliography/documentos/curso_doctorado_Javier.pdf]

[119] Sato, J. and Cipolla, R.: Affine integral invariants and matching of curves, *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, Vol. 1, pp. 915-919, IEEE, 1996

[120] Sato, J. and Cipolla, R.: Affine integral invariants for extracting symmetry axes, *Image and Video Computing*, Vol. 15, Issue 8, pp. 627-635, Elsevier B.V., 1997

[121] Schaffer, J.D., Whitley, D. and Eshelman, L.J.: Combinations of genetic algorithms and neural networks: a survey of the state of the art, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, pp. 1-37, IEEE Press, USA, 1992

[122] Schraudolph, N. N.; "A fast, compact approximation of the exponential function", *Neural Computation*, 11(4), pp. 853-862, 1999.

[123] Shinohara, A.: Complexity of computing Vapnik-Chervonenkis dimension and some generalized dimensions, *Theoretical Computer Science*, Vol. 137, pp. 129-144, 1995.

[124] Silva, L. M.; Marques de Sá, J., and Alexandre, L. A.: "Data classification with multilayer perceptrons using a generalized error function", *Neural Networks Journal*, Vol. 21, pp. 1302-1310, 2008.

[125] Stathakis, D.: How many hidden layers and nodes?, *International Journal of Remote Sensing*, Vol. 30, Nº 8, pp. 2133-2147, Taylor & Francis, 2009

[126] Steger, C.: On the calculation of arbitrary moments of polygons, *Technical Report FGBV-96-05, Forschungsgruppe Bildverstehen (FG BV)*, Informatik IX, Technische Universität München, 1996

[127] Sorber, L.; Van Barel, M., and De Lathauwer, L.: Unconstrained optimization of real functions in complex variables, *SIAM Journal on Optimization (SIOPT)*, Vol 22(3), pp. 879-898, 2012.

[128] Suk, T. and Flusser, J.: Projective moment invariants, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol. 26, Issue 10, pp. 1364-1367, IEEE, 2004

[129] Suk, T. and Flusser, J.: The projective invariants for polygons, *Computer Analysis of Images and Patterns*, Lecture Notes in Computer Science, Vol. 970, pp. 729-734, Springer Berlin Heidelberg, 1995

[130] Uttley, A.M.: A theory of the mechanism of learning based on the computation of conditional probabilities, *Proceedings of the First International Conference on Cybernetics*, Gauthier-Villars, Paris, France, 1956

[131] Van Gool, L.J.; Moons, T.; Pauwels, E. and Wagemans, J.: Invariance from the Euclidean geometer's perspective, *Perception*, Vol. 23, Issue 5, pp. 547-561, 1994

[132] Von der Malsburg, C.: Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik*, Vol. 14, Issue 2, pp. 85-100, Springer-Verlag, 1973

[133] Wang, L. and Quek, H.: Optimal size of a forward neural network: How much does it matter?, *IEEE Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005)*, 2005.

[134] Wang, Y. B.; Wang, X. W.; Zhang, B., and Wang, Y.: A novel form of affine moment invariants of grayscale images, *Electronics and Electrical Engineering Journal*, Vol. 19, No 1, pp. 77-82, 2013.

[135] Werbos, P.: Backpropagation, past and future, *Proceedings of the Second International Conference on Neural Networks*, IEEE, New York, USA, 1988

[136] Welsh, D.: *Codes and cryptography*, Oxford University Press, reprinted (with corrections), 1989.

[137] Whitley, D.: "Genetic algorithms and neural networks", *Genetic algorithms in engineering and computer science*, pp. 191-201, John Wiley & Sons, USA, 1995.

[138] Widrow, B.: Generalization and information storage in networks of Adeline neurons, *Self-Organizing Systems Yovits et al. eds*, pp. 435-461, Spartan Books, Washington, DC, USA, 1962

[139] Widrow, B. and Hoff, M.E.: Adaptive switching circuits, *IRE WESCON Convention Record*, pp. 96-104, 1960

[140] Willshaw, D.J. and von der Malsburg, C.: How patterned neural connections can be set up by self-organization, *Proceedings of the Royal Society of London Series B*, Vol. 194, pp. 431-445, 1976

[141] Wilson, D. R., and Martinez, T. R.: "The inefficiency of batch training for large training sets", *Proceedings of the International Joint Conference on Neural Networks (IJCNN2000)*, Vol. 2, pp. 113-117, 2000.

[142] Xirouhakis, Y.; Avrithis, Y., and Kollias, S.: Image retrieval and classification using affine invariant B-spline representation and neural networks, *Proceedings of the IEEE Colloquium on Neural Nets and Multimedia*, pp. 4/1-4/4, IET, London, UK, 1998.

[143] Xirouhakis, Y.; Avrithis, Y., and Kollias, S.: Affine invariant representation and classification of object contours for image and video retrieval, *Computational Intelligence and Applications*, World Scientific and Engineering Society Press, pp. 342-347, 1999.

[144] Zhang, D., and Lu, G.: Review of shape representation and description techniques, *Pattern Recognition*, Vol. 37, pp. 1-19, 2004.

[145] Zhang, B. and Mühlenbein, H.: Genetic programming of minimal neural nets using Occam's razor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 342-351, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1993

[146] Zhang, H., and Wu, W.: Convergence of split-complex backpropagation algorithm with momentum, *Neural Netw World*, Vol. 21(1), pp. 75-90, Institute of Information and Computer Technology ASCR, Faculty of Transport, Czech Polytechnic University, Prague, 2011.

[147] Zhang, Y.; Wen Ch.; Zhang, Y., and Soh, Y. Ch.: Neural network based classification using blur degradation and affine deformation invariant features, *International Journal on Artificial Intelligence Tools*, Vol. 10, No. 01n02, pp. 243-256, 2001.

[148] Zurada, J. M.: *Introduction to artificial neural systems*, West Publishing Company, St. Paul, MN, USA,1992