



UNIVERSITAT POLITECNICA DE CATALUNYA

ETSETB- DEPARTAMENT D'ENGINYERIA ELECTRONICA

Curso de Redes Neuronales Artificiales (2000-2001)

Aprendizaje Estadístico

Autor: Sergi Bermejo

Responsable del curso: Joan Cabestany

1. Aprendizaje Computacional.

El aprendizaje es una habilidad de la que disponen gran parte de los sistemas naturales para adaptarse al entorno en el que viven. Es por ello una propiedad interesante de emular de manera artificial, ya que muchos problemas de ingeniería requieren para su correcto funcionamiento algún tipo de adaptación al entorno en el que operan. Definir de manera única y precisa el término "*aprendizaje*" resulta complicado ya que se puede abordar desde diferentes puntos de vista. A continuación mostramos algunas definiciones posibles que ponen de manifiesto este hecho:

- A. "*Establecimiento de nuevas relaciones temporales entre un ser y su medio ambiental*" [Ambito: Psicología]
- B. "*Acción de adquirir conocimiento de una cosa por medio del estudio, ejercicio o experiencia*" [Ambito: Psicología]
- C. "*Tomar algo en la memoria*" [Ambito: Psicología]
- D. "*Un cambio relativamente permanente en el comportamiento que ocurre como resultado de una práctica de refuerzo*" [Ambito: Psicología]
- E. "*Un proceso por el cual los parámetros libres del sistema se adaptan a través de un proceso continuo de estimulación a partir del entorno en el que el sistema está inmerso*" [Ambito: Inteligencia artificial]
- F. "*Aprender significa poder inferir la relación entre X e Y del conjunto de entrenamiento D*" [Ambito: Inteligencia artificial]

En el contexto de los sistemas artificiales, el aprendizaje, también denominado **aprendizaje computacional**, se puede entender como:

Un **proceso** en el que un *aprendiz* produce una función de aplicación a través de la información de entrenamiento extraída de algún entorno. (fig.1)

Es por lo tanto un fenómeno que sucede a lo largo de un tiempo determinado que puede corresponder a una cierta etapa dentro de la vida del sistema artificial, o por el contrario se puede extender a lo largo de toda la vida de dicho sistema.

Durante este tiempo, el aprendiz (*learner*) busca en el espacio de todas las posibles soluciones que es capaz de construir¹, una solución óptima en relación con alguna medida de coste de la que dispone (fig.2) utilizando para ello *recursos computacionales limitados*. Es decir, el aprendiz tiene:

1. *Un tiempo de búsqueda limitado*. Para encontrar una solución el aprendiz debe utilizar un tiempo de CPU que no exceda un tiempo máximo asignado al aprendizaje.

¹ P.e. el espacio de las funciones polinómicas de grado p.

2. *Un espacio de búsqueda, o espacio de hipótesis, limitado.* Puesto que la solución se debe buscar en un tiempo finito, el espacio de hipótesis debe ser forzosamente restringido para que el aprendiz pueda encontrar una solución antes del tiempo máximo que se ha establecido.
3. *Una información limitada acerca del entorno.* En la práctica es habitual disponer de escasa información acerca del entorno sobre el que queremos definir una función, ya que p.e. no podemos modelarlo analíticamente de forma precisa. Por ello, se deberá buscar una solución acorde con la información disponible en el momento del aprendizaje pero compatible además con aquella información que se pueda extraer del entorno en el futuro.

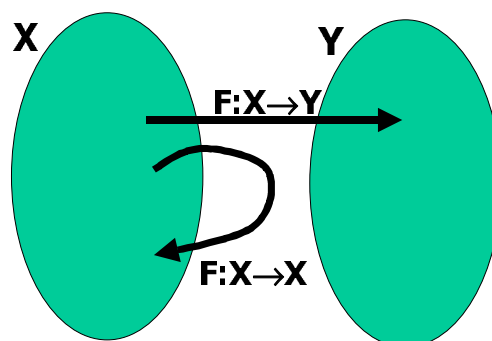


Figura 1. Aprendizaje computacional. El resultado del aprendizaje es simplemente una función definida sobre uno o más espacios (p.e. vectoriales)

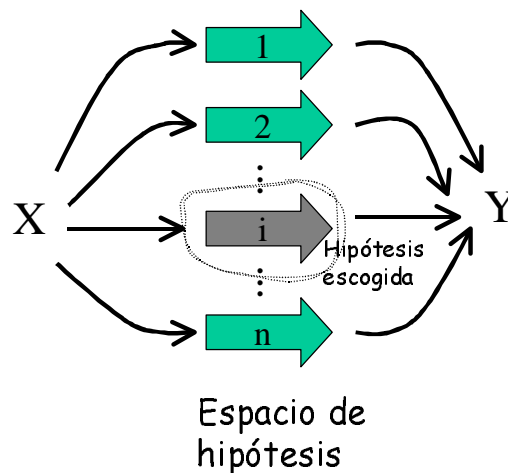


Figura 2. El aprendizaje como una búsqueda en un espacio de hipótesis. De todas las hipótesis posibles, el aprendiz debe escoger aquella que minimiza (o maximiza) una función de coste de la que dispone a priori. Así el aprendizaje pasa por un proceso de optimización de una función de coste definida sobre el espacio de hipótesis.

La búsqueda en este espacio de hipótesis puede ser ciega o guiada (fig.3). En el primer caso, no se dispone de ninguna información acerca de en qué sub-espacio puede residir la solución. En cambio, en la búsqueda guiada, el aprendiz si que dispone de información concreta de en qué región del espacio de hipótesis ha de buscar la solución. Esta guía permite así restringir la búsqueda a un subespacio, pudiendo resultar útil por dos motivos diferentes:

1. Puede eliminar tiempo de aprendizaje ya que se busca en un espacio menor

2. Puede hacer posible que la solución obtenida sea más fiable que la resultante de buscar en un subespacio mayor puesto que, dada una información fija acerca del entorno, el aprendiz puede (en general) dar una mejor solución a medida que el espacio de hipótesis es menor.

Existen diversas formas de guiar al aprendiz en un espacio de hipótesis. La primera consistiría en reducir la complejidad del problema a aprender ya que a medida que el problema sea más simple se deberá buscar en un espacio de hipótesis menor. Una segunda posibilidad sería utilizar conocimiento a priori del problema limitando así la búsqueda en aquellos lugares donde se presupone que la solución puede residir. Finalmente, una tercera opción consistiría en imponer limitaciones sobre el tipo de solución que busca a través de restringir de forma conveniente el proceso de optimización que se efectúa sobre la función de coste.

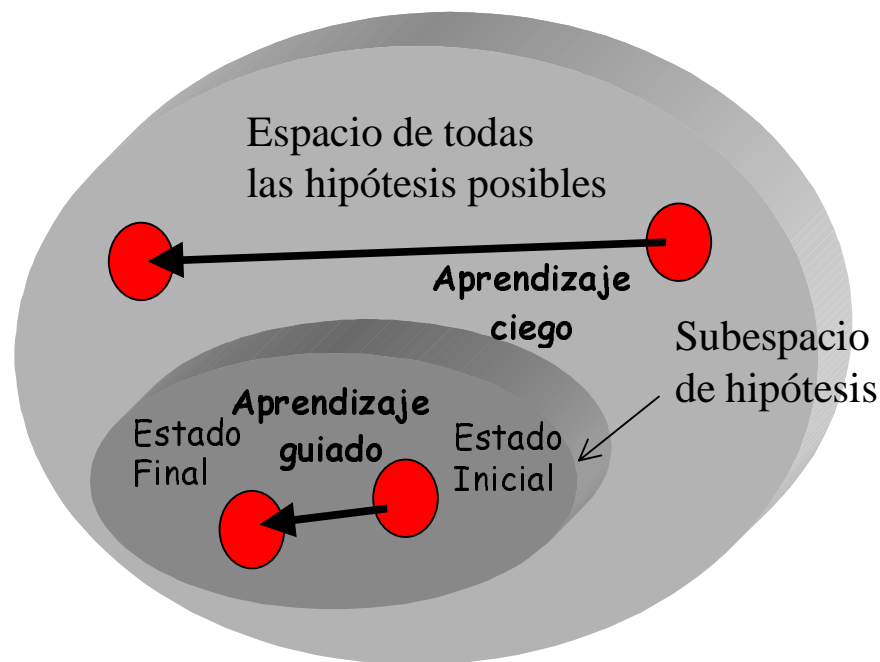


Figura 3. Aprendizaje guiado vs. aprendizaje ciego. En el aprendizaje ciego no se imponen restricciones en la búsqueda. En cambio, en el aprendizaje guiado sí. A efectos prácticos esta guía implica realizar una búsqueda restringida a un subespacio. Debido al uso de recursos computacionales limitados, puede resultar más práctico buscar en un espacio de hipótesis menor aunque limitemos por ello el número de posibles soluciones que nos puede dar el aprendiz.

El resultado del aprendizaje (la solución dada por el aprendiz) es un modelo² que *debería* permitir obtener al sistema leyes generales sobre el proceso del entorno del que se ha obtenido la información de entrenamiento. Estas leyes generales indican la estructura computacional³ de dicho proceso, permitiendo así predecir un número indefinido de nuevos fenómenos procedentes de él (**generalización**). Dicho con otras palabras, el aprendiz a partir de la información que recibe de un proceso observado en el entorno en el que opera, infiere un modelo de dicho proceso, es decir, un conjunto de relaciones entre diversas variables observables en él (fig. 4). **El aprendiz almacena este modelo inferido en forma de conocimiento en la memoria del sistema artificial.** Así, el sistema establece nuevas conductas, nuevas relaciones temporales, con su entorno, puesto que la información recibida le permite alterar la idea que hasta ahora tenía del proceso observado. Ello le permite al sistema mientras aprende, adaptarse al entorno en el que habita⁴.

² modelo = estructura matemática que da razón de un conjunto de fenómenos que guardan entre sí ciertas relaciones

³ las relaciones entre diversas variables, objetos, etc. existentes en el proceso del entorno que se observa

⁴ En un sistema natural esta adaptación permite en cierta manera predecir algunos eventos futuros y así sobrevivir.

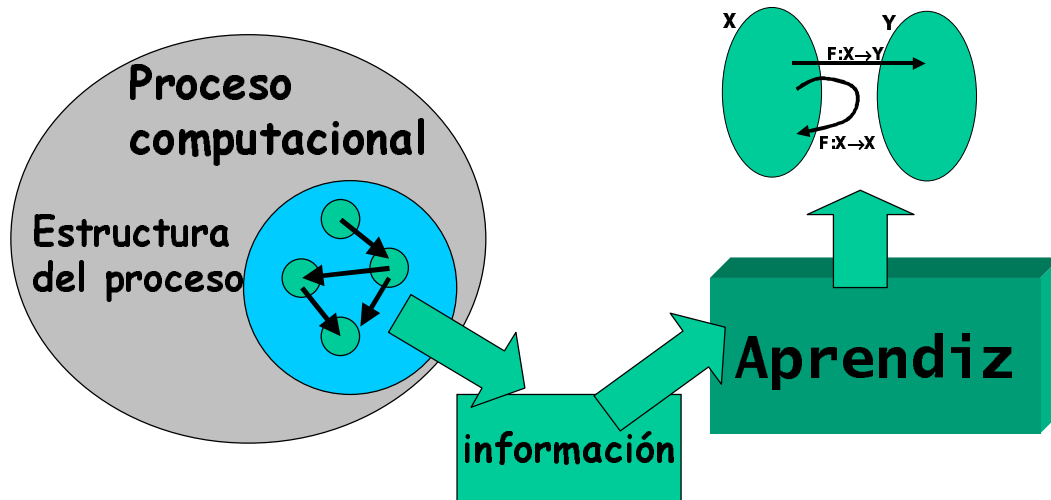


Figura 4. El aprendizaje como un modelado de un proceso computacional. A partir de algún tipo de información extraída de dicho proceso el aprendiz infiere un modelo que debe dar cuenta de él.

1.1. Criterios para evaluar un sistema aprendizaje.

Los criterios que con mayor frecuencia se utilizan a la hora de evaluar un sistema de aprendizaje son dos:

- la precisión predictiva o **generalización**
- la **comprensibilidad** de sus modelos aprendidos

1.1.1. Generalización.

El objetivo de un sistema que aprende es extraer un modelo representativo a partir del conocimiento disponible de un proceso computacional. Un modelo representativo ha de poder predecir nuevos fenómenos del proceso, y por lo tanto ha de ser capaz de generalizar. Así, la capacidad de generalización de estos sistemas nos da cuenta de lo bien que el modelo obtenido por el aprendiz (en nuestro caso la red neuronal con sus parámetros ajustados) responde a estímulos que el sistema no ha visto a la hora de construir dicho modelo.

1.1.2. Comprensibilidad.

Si los sistemas de aprendizaje inducen un modelo de un proceso, es deseable que éste sea comprensible, es decir, que sea fácilmente inspeccionado y entendido. Existen diversas razones para ello. Podemos estar interesados en validar el modelo inducido. O bien podemos estar interesados en entender mejor los datos y descubrir las principales características y relaciones entre ellos. O bien podemos estar interesados en modificar ligeramente el modelo para mejorar así la capacidad de generalización. Para cualquiera de estos propósitos se hace necesario poder analizar el modelo construido.

2. Aprendizaje estadístico.

Como hemos visto un sistema con aprendizaje debe, a partir de un conjunto de información acerca de un proceso computacional, construir un modelo que permita predecir nuevos fenómenos asociados a él. Debe por lo tanto, generar un modelo con capacidad de generalizar. Se hace necesario entonces que el aprendizaje efectúe algún tipo de inducción⁵ a partir de la información disponible.

⁵ Inducir = Elevar el entendimiento, desde el conocimiento de los fenómenos hasta la ley que los rige

Para poder inducir se precisa de un conjunto de medidas o ejemplos asociado al proceso que se quiere modelar. Este tipo de aprendizaje, denominado **aprendizaje inductivo**, se convierte de hecho en un **aprendizaje con ejemplos** que es en el fondo un *problema (especialmente difícil) de aproximación de una función de la que se conoce únicamente un conjunto de puntos*. La complejidad de dicho problema es especialmente notoria si tenemos presente que:

1. El **número de variables** asociado al espacio de entrada de la función a aproximar es **elevado**. Puesto que los procesos a modelar por estos sistemas suelen ser altamente complejos ya que son difíciles de caracterizar, es habitual que dependan de muchas variables, máxime al ser estos extraídos de entornos reales.
2. Las **muestras** disponibles suelen ser **escasas**, estar **dispersas** y tener asociadas una cierta **incertidumbre**.

Extraer muestras de un proceso real puede ser costoso por lo que en general no se dispone de un número ilimitado de muestras. De esta manera, la información disponible para aproximar es limitada. Además, puesto que el número de variables de entrada suele ser elevado, las muestras tienden a estar muy alejadas entre sí en el espacio de entrada. Así la correcta reconstrucción de la función se hace más difícil. Por último, las muestras pueden llevar consigo de forma inherente una cierta **incertidumbre** (p.e. fluctuación, ruido). En consecuencia, no resulta útil aproximar de forma precisa los puntos disponibles (p.e. interpolar). Esta incertidumbre asociada a las muestras puede deberse a diversos motivos:

- ❑ Debido a la imprecisión de los aparatos de medida utilizados a la hora de extraer muestras del proceso, puede existir ruido en las medidas.
- ❑ La información de que dispone es incompleta (p.e. en algunas muestras faltan el valor de algunas las variables implicadas).
- ❑ El proceso que analizamos es no-determinístico.

Por lo tanto, el aprendiz debe reconstruir una función que suele operar en espacios de alta dimensión a partir de un conjunto limitado de ejemplos dispersos y con ruido. Para poder tratar con la incertidumbre existente en el proceso de reconstrucción se hace necesario que el aprendiz infiera o estime a partir del conjunto de muestras (o conjunto de entrenamiento) una estructura, modelo o función de tipo estadístico que defina al proceso computacional del que proceden los datos⁶. Además la definición del problema del aprendizaje en términos estadísticos nos va a permitir cuantificar claramente aspectos como la capacidad de generalización del sistema con aprendizaje. Este modelo estadístico inferido del proceso a partir de ejemplos puede servir tanto para realizar una tarea de interés (ingeniería) como para conseguir un mejor entendimiento de los datos disponibles y, por extensión, del proceso al que representan.

Existen dos maneras fundamentales de aprender con ejemplos. En la primera, conocida con el nombre de **aprendizaje no supervisado** o **auto-organizado**, el objetivo pasa por descubrir las propiedades estadísticas de un vector aleatorio X asociado al proceso computacional a modelar. Para ello únicamente se dispone de un conjunto de muestras procedentes de variables de entrada del proceso, llamado $D = \{\mathbf{x}_i, i=0 \dots N-1\}$ siendo \mathbf{x}_i una muestra aleatoria extraída de un vector aleatorio X de dimensión p que toma valores reales⁷ que opera sobre un espacio de entrada (o escrito de forma abreviada $\mathbf{x}_i \sim X \in \mathfrak{R}^p$).

⁶ Recordemos que la herramienta matemática por antonomasia que trata con la imprecisión es la teoría de la probabilidad y la estadística se encarga de aplicar de forma práctica sus preceptos al disponer de muestras aleatorias extraídas de cualquier proceso probabilístico.

⁷ A lo largo de todo este libro, trabajaremos con variables de entrada y salida numéricas que pertenecen a los reales. En general es posible trabajar en estos mismos sistemas con otro tipo de variables (p.e. simbólicas) puesto que pueden ser codificadas en forma de vector cuyas componentes son números reales.

En cambio, el segundo tipo de aprendizaje, conocido con el nombre de **aprendizaje supervisado**, tiene por finalidad revelar las relaciones existentes entre dos vectores aleatorios X e Y que, al igual que en el caso anterior, forman parte del proceso computacional a modelar. En este aprendizaje se dispone de un conjunto de pares de muestras procedentes de dichas variables de entrada y salida de dicho proceso, denominado $D = \{(\mathbf{x}_i, \mathbf{y}_i), i=0 \dots N-1\}$ con $\mathbf{x}_i \sim X \in \mathfrak{R}^p$ y $\mathbf{y}_i \sim Y \in \mathfrak{R}^m$.

De hecho, estos dos tipos de aprendizaje suelen ser complementarios puesto que es posible aprender de forma más fácil las relaciones entre X e Y si previamente tenemos información acerca de la estructura estadística de X . Por ello es habitual encontrar en la práctica sistemas con **aprendizaje híbrido** en los que el aprendizaje no supervisado facilita el aprendizaje de la parte supervisada, o si se quiere, el aprendizaje no supervisado guía al supervisado en la búsqueda de la solución. A continuación daremos un breve repaso a estos dos paradigmas de aprendizaje desde un punto de vista estadístico. Veremos también de qué elementos se compone un aprendiz estadístico y cuales son las limitaciones de tipo práctico existentes en este tipo de aprendizaje.

2.1. Aprendizaje no supervisado.

En el aprendizaje estadístico no supervisado el objetivo final sería estimar de forma precisa la función densidad de probabilidad del vector aleatorio X , denominada $f_X(\mathbf{x})$, ya que representa la estructura estadística de X , el cual está asociado al proceso computacional que queremos modelar. De esta manera obtenemos toda la información de acerca de X en términos estadísticos.

Por ejemplo, una información de la que disponemos a partir de esta información es la de los grupos naturales (*clusters*) existentes, ya que se corresponden con regiones en las que $f_X(\mathbf{x})$ tiene máximos locales. Así en el caso de disponer de un conjunto de imágenes sería posible a partir del aprendizaje no supervisado utilizar dicha información para poder almacenar estas imágenes de forma comprimida.

En la práctica la estimación de $p_X(\mathbf{x})$ es demasiado complicada debido a los escasos recursos disponibles. Así el aprendizaje no supervisado suele buscar una formulación del problema más directa. Esta se basa en construir una función que aproxime al vector aleatorio X . Buscamos por lo tanto construir una función de \mathbf{x} a partir de un conjunto de entrenamiento $D = \{\mathbf{x}_i, i=0 \dots N-1\}$, denominada $\mathbf{F}(\mathbf{x}; D)$ que sea un estimador de X tal que:

$$X \approx \hat{X} = \mathbf{F}(\mathbf{x}; D)$$

Para ello, el criterio más habitual a minimizar durante el aprendizaje se basa en el error cuadrático medio

$$I_{\text{emp}}^U[\mathbf{F}(\mathbf{x}; D)] = \frac{1}{2N} \sum_{i=0}^{N-1} \|\mathbf{x}_i - \mathbf{F}(\mathbf{x}_i)\|^2 = \frac{1}{2N} \sum_{i=0}^{N-1} \sum_{j=0}^{p-1} (x_{ij} - F_j(\mathbf{x}_i))^2$$

Dependiendo de la forma de aproximar de \mathbf{F} podemos encontrar diversas soluciones al mismo problema estadístico. \mathbf{F} puede aproximar de forma local, utilizando para ello un conjunto de vectores prototipos $\{\mathbf{w}_i, i=1 \dots K\}$ de la siguiente manera⁸

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^K g_i(\mathbf{x}) \mathbf{w}_i \quad \text{con} \quad g_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in R_i \\ 0 & \text{sino} \end{cases} \quad \text{y} \quad R_i = \left\{ \mathbf{x} \mid \|\mathbf{x} - \mathbf{w}_i\| \leq \min_{j=1 \dots K} \|\mathbf{x} - \mathbf{w}_j\| \right\}$$

⁸ A este tipo de sistemas se les conoce con el nombre de cuantificadores vectoriales (ver capítulo 3).

Entonces para minimizar I_{emp}^U cada \mathbf{w}_i resulta ser el centroide calculado a partir de las muestras de entrenamiento que caen en R_i , es decir, un estimador de la esperanza de pertenecer a la región de Voronoi R_i

$$\mathbf{w}_i = \frac{\sum_{j=0}^{N-1} g_i(\mathbf{x}_j) \mathbf{x}_j}{\sum_{j=0}^{N-1} g_i(\mathbf{x}_j)}$$

Así al minimizar I_{emp}^U , los K vectores prototipo se distribuyen sobre el espacio de entrada de manera que aproximan de forma discreta la densidad de probabilidad desconocida $f_X(\mathbf{x})$. Donde \mathbf{x} es denso o disperso, los codevectores tienden a ser densos o dispersos. La función densidad de probabilidad será bien aproximada por ese conjunto de vectores prototipo si hay un número de ellos que sea adecuado.

En cambio, \mathbf{F} puede construir su solución de forma global basándose en aproximar $\mathbf{x} \in \mathcal{R}^p$ mediante un vector $\hat{\mathbf{x}}$ resultante de proyectar \mathbf{x} sobre m ejes determinados (con $m < p$). Para realizar esta operación basta realizar una transformación lineal a un espacio de dimensión m

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} \quad \text{con } \mathbf{W} = [\mathbf{w}_1 \quad \dots \quad \mathbf{w}_m]$$

donde los vectores $\{\mathbf{w}_i, i=1\dots m\}$ tienen módulo 1 y T denota matriz traspuesta. Posteriormente se reconstruye de forma aproximada \mathbf{x} realizando la operación inversa

$$\mathbf{F}(\mathbf{x}) = \hat{\mathbf{x}} = \mathbf{W}\mathbf{z} = \mathbf{W}\mathbf{W}^T \mathbf{x}$$

Si esta \mathbf{F} minimiza I_{emp}^U entonces se puede demostrar que los ejes $\{\mathbf{w}_i, i=1\dots m\}$ son aquellas m direcciones sobre el espacio de entrada donde reside X en las que las muestras aleatorias $\{\mathbf{x}_i, i=0\dots N-1\}$ están más dispersas. Es decir, $\{\mathbf{w}_i, i=1\dots m\}$ estiman aquellas m direcciones de máxima varianza de X^9 .

Ambos modelos se pueden utilizar para fines diversos. No obstante, el modelo local encaja mejor para un análisis de clusters ya que distribuye vectores prototipo sobre X que justamente son centroides de su región de influencia. Así si existen diversas nubes de datos (grupos naturales) en X es posible detectarlos ya que los vectores prototipo tenderán a cada uno de los grupos. En cambio, el modelo global puede servir mejor como técnica de pre-procesado, es decir como técnica de reducción de la dimensión del espacio de entrada¹⁰. Ello se debe a que el modelo global construye un espacio de menor dimensión que el de entrada que permite comprimir de forma óptima a X ya que el modelo una vez entrenado minimiza el error de reconstrucción que se obtiene de transformar los vectores aleatorios pertenecientes a X y posteriormente anti-transformar volviendo de nuevo al espacio de entrada.

2.2. Aprendizaje supervisado.

Dados dos vectores aleatorios X e Y , es decir un conjunto de variables procedentes del espacio de entrada y salida del proceso a modelar, el objetivo del **aprendizaje supervisado** es descubrir la relación existente

⁹ A este tipo de sistemas se les conoce con el nombre de extractores de componentes principales (ver capítulo 3). Hemos supuesto que la esperanza de X es cero en la formulación de la solución de máxima varianza. Si esto no es así bastaría substituir \mathbf{x} por

$$\mathbf{x}' = \mathbf{x} - \sum_{i=0}^{N-1} \mathbf{x}_i$$

¹⁰ Como hemos visto antes una posible forma de guiar al aprendiz era reducir la complejidad del problema. Una posible forma de hacer esto es reducir la dimensión del espacio de entrada a partir del cual el sistema de aprendizaje debe construir su solución.

entre estos dos vectores. Por ejemplo, podemos necesitar relacionar la evolución de la bolsa a escala mundial (Y) en función de ciertas variables macroeconómicas (X).

Puesto que nos interesa caracterizar Y en relación con X y únicamente disponemos de un número finito de muestras $D = \{(\mathbf{x}_i, \mathbf{y}_i), i=0 \dots N-1\}$, deberemos utilizar D para construir una estimación de Y a partir de una función del vector aleatorio X , es decir,

$$Y \approx \hat{Y} = \mathbf{F}(\mathbf{x}; D)$$

En este tipo de aprendizaje distinguiremos un caso general (regresión) y otro particular (clasificación) cuyas peculiaridades hacen de él un caso de estudio aparte.

2.2.1. Regresión.

En el caso general se nos pide relacionar dos vectores aleatorios X e Y que pertenecen a espacios vectoriales reales. Por ejemplo, se nos pide cuantificar la influencia de ciertos indicadores financieros en la evolución de la bolsa. Para ello disponemos de un conjunto de muestras D que tiene la siguiente forma $\{(\mathbf{x}_i, \mathbf{y}_i), i=0 \dots N-1\}$ con $\mathbf{x}_i \sim X \in \mathfrak{R}^p$ y $\mathbf{y}_i \sim Y \in \mathfrak{R}^m$. Al igual que en el aprendizaje no supervisado, podemos construir un estimador de Y a partir de minimizar un criterio basado en el error cuadrático medio,

$$I_{\text{emp}}^R[\mathbf{F}(\mathbf{x}; D)] = \frac{1}{2N} \sum_{i=0}^{N-1} \|\mathbf{y}_i - \mathbf{F}(\mathbf{x}_i)\|^2 = \frac{1}{2N} \sum_{i=0}^{N-1} \sum_{j=0}^{m-1} (y_{ij} - F_j(\mathbf{x}_i))^2$$

Se puede demostrar que la función que minimiza dicho funcional ha de ser un estimador del centro de masa probabilístico en el intervalo $(\mathbf{x}, \mathbf{x} + d\mathbf{x})$ $E_{Y|X}(\mathbf{y}|\mathbf{x})$. Así, la relación existente entre los dos vectores aleatorios X e Y , queda caracterizada en la masa de probabilidad del hiperplano XY (p.170; Papoulis, 1991). En la figura 5 se muestra una interpretación gráfica del significado de $E_{Y|X}(\mathbf{y}|\mathbf{x})$.

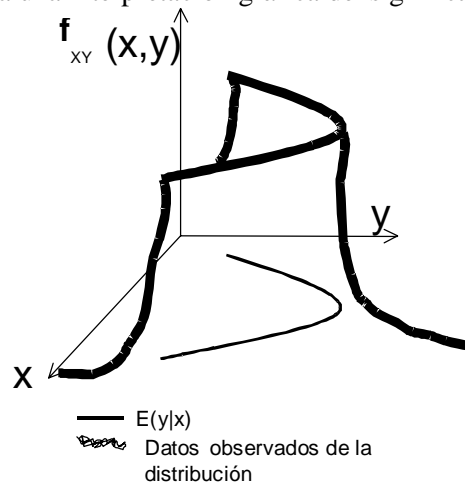


Figura 5. $E(y|x)$ curva de regresión.

\mathbf{F} puede ser construida de diversas formas. Así, si \mathbf{F} es un modelo basado en la aproximación lineal tendríamos que:

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^M \mathbf{w}_i \phi_i(\mathbf{x}) \quad \text{o} \quad F_j(\mathbf{x}) = \sum_{i=1}^M w_{ij} \phi_i(\mathbf{x}) \quad j = 1 \dots m$$

siendo $\{\phi_i(\mathbf{x}), i = 1 \dots M\}$ las denominadas funciones básicas. En el capítulo 4, veremos dos maneras bien diferenciadas de construir estas funciones básicas. En un perceptrón multicapa (MLP), ϕ_i se construye de forma recursiva a partir de aproximadores elementales no-lineales basados en hiperplanos:

$$\phi_i(\mathbf{x}) = \varphi \left(\sum_{j=1}^{M_1} w_{ij}^1 \varphi \left(\sum_{j=1}^{M_2} w_{ij}^2 \varphi \left(\dots \varphi \left(\sum_{j=1}^p w_{ij}^L x_j \right) \right) \right) \right)$$

con $\varphi : \mathcal{R} \rightarrow \mathcal{R}$ no lineal.

En cambio, una función básica radial (RBF) tiene, como su propio nombre indica, funciones básicas ϕ_i del tipo radial:

$$\phi_i(\mathbf{x}) = e^{-\frac{\|\mathbf{x} - \mathbf{m}_i\|^2}{\sigma_i}}$$

2.2.2. Clasificación.

En un problema de clasificación el objetivo es asociar cada objeto o patrón de entrada a una de las clases existentes. Supongamos que disponemos de un conjunto de imágenes que contienen caracteres manuscritos. Así un clasificador debe asignar una imagen que contenga la letra 'a' a la clase 'a', etc.

En este caso el vector (aleatorio) X corresponde a los patrones a clasificar e Y a un vector (aleatorio) que indica a cual de las m clases existentes pertenece X . El conjunto de muestras D tiene entonces la siguiente forma $\{(\mathbf{x}_i, \mathbf{y}_i), i=0 \dots N-1\}$ con $\mathbf{x}_i \sim X \in \mathcal{R}^p$ y $\mathbf{y}_i \sim Y \in \{0,1\}^m$. Aquí las componentes de cada vector aleatorio \mathbf{y}_i indican si \mathbf{x}_i pertenece o no cada una de las clases. En consecuencia,

$$y_{ij} = 1(\mathbf{x}_i \in \text{clase } j)$$

siendo $1(\text{condición})$ la función indicatriz que vale uno si la condición se cumple y cero si no es así. El clasificador a construir F es por lo tanto una función que únicamente toma valores 0 ó 1 en $\{0,1\}^m$.

El objetivo del aprendizaje en un problema de clasificación pasa por construir un clasificador F que estime Y . El mejor clasificador posible en términos probabilísticos es aquel que tiene una probabilidad de error de clasificación mínima para el problema en cuestión. Dicho clasificador se conoce con el nombre de clasificador de Bayes. Por lo tanto, el objetivo es entonces construir un clasificador F cercano al de Bayes.

Existen, al menos, dos maneras posibles de construir un estimador del clasificador de Bayes. La primera consistiría en minimizar directamente el número de muestras mal clasificadas (conocido también como error de clasificación),

$$I_{\text{emp}}^C[\mathbf{F}(\mathbf{x}; D)] = \frac{1}{N} \sum_{i=0}^{N-1} 1(\mathbf{y}_i \neq \mathbf{F}(\mathbf{x}_i)) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=1}^m 1(y_{ij} \neq F_j(\mathbf{x}_i))$$

No obstante, esta función presenta problemas a la hora de ser minimizada mediante los métodos de optimización que se emplean habitualmente. Por ello, se suele emplear en la práctica otro tipo de funcionales cuyos mínimos coinciden con los mínimos de I_{emp}^C o al menos se aproximan.

Otra posibilidad pasaría por construir la función F a partir del siguiente esquema de decisión:

$$F_j(\mathbf{x}) = \begin{cases} 1 & f_j(\mathbf{x}) = \max_{k=1\dots m} f_k(\mathbf{x}) \\ 0 & \text{sino} \end{cases} \quad j = 1\dots m$$

donde $\{f_j(\mathbf{x}), j=1\dots m\}$ son las denominadas funciones discriminantes. Estas funciones pueden, en principio, tomar cualquier valor real aunque resulta conveniente que estén sujetas a la siguiente restricción

$$\sum_{j=1}^m f_j(\mathbf{x}) = 1$$

ya que dichas funciones podrán ser interpretadas como estimadores de cierto tipo de probabilidades. De hecho, si sujetas a esta restricción minimizamos $I_{\text{emp}}^R[\mathbf{f}(\mathbf{x}; D)]$ se puede demostrar que las funciones discriminantes estiman la probabilidad a posteriori de pertenecer a cada una de las clases. Es decir,

$$\mathbf{f}^* = \min_{\mathbf{f}} I_{\text{emp}}^R[\mathbf{f}(\mathbf{x}; D)] \Leftrightarrow f_j^*(\mathbf{x}; D) = \hat{P}(X \in \text{clase } j | \mathbf{x}) \quad j = 1\dots m \quad \text{con} \quad \sum_{j=1}^m f_j(\mathbf{x}) = 1$$

Finalmente si utilizamos \mathbf{f}^* en el esquema de decisión propuesto, el clasificador resultante se acerca cada vez más al clasificador de bayes a medida que se estiman mejor las probabilidades a posteriori $\{P(X \in \text{clase } j | \mathbf{x}), j = 1\dots m\}$. Esto es debido a que el clasificador de Bayes utiliza el esquema de decisión propuesto utilizando las probabilidades a posteriori de pertenecer a cada clase como funciones discriminantes.

2.3. Componentes de un sistema de aprendizaje estadístico.

A partir de un conjunto de entrenamiento D de tamaño N y de una función parametrizable $F(\mathbf{x}; \mathbf{W})$ (p.e. una red neuronal) siendo \mathbf{W} el conjunto de parámetros asociados a la función (p.e. los pesos de la red neuronal), el problema del aprendizaje estadístico pasa por calcular \mathbf{W} de manera que se consiga un **objetivo estadístico**, p.e. minimizar **una función de coste estadística**. Para ello se utilizará algún **método de optimización**. El sistema de ecuaciones obtenido al aplicar el método de optimización sobre la función de coste estadístico es lo que se conoce como **algoritmo de entrenamiento**. Dicho algoritmo es en realidad un **sistema dinámico**, es decir un conjunto de ecuaciones que evolucionan en el tiempo. Este sistema dinámico deberá converger hacia el mínimo de la función de coste. No obstante, será habitual definir un **criterio de parada** del algoritmo que permita parar la ejecución del mismo antes de que converja.

Como acabamos de ver diversos elementos dar lugar a la formación de un algoritmo de aprendizaje estadístico. A continuación hablaremos de cada uno de ellos con un mayor detalle.

2.3.1. Función de coste estadística.

Como hemos visto existen dos tipos fundamentales de aprendizaje con ejemplos: supervisado y no supervisado. En ellos el objetivo pasa por modelar la estructura estadística que determina a un proceso computacional a partir de una serie de ejemplos (conjunto de entrenamiento D) que se ha extraído de dicho proceso. Para construir dicho modelo se dispone a priori de una función parametrizable $F(\mathbf{x}; \mathbf{W})$ siendo \mathbf{W} el conjunto de parámetros asociados a la función. En consecuencia, el objetivo del aprendizaje es el cálculo del conjunto de parámetros \mathbf{W} de manera que $F(\mathbf{x}; \mathbf{W})$ sea un buen

modelo de la estructura estadística que define el proceso computacional. Para poder determinar \mathbf{W} de la manera más simple, basta definir una función de coste que mida la distancia entre el modelo y la estructura estadística que conocemos a través de un conjunto de ejemplos. De esta manera $\mathbf{F}(\mathbf{x}; \mathbf{W})$ será en principio mejor modelo a medida que su distancia con la estructura estadística del proceso sea menor. Así el $\mathbf{F}(\mathbf{x}; \mathbf{W})$ óptimo será aquel que minimice este tipo de función de coste basado en la distancia. En el caso de utilizar la distancia euclidiana, las dos funciones de coste resultantes para el aprendizaje no supervisado y supervisado son las ya vistas en el punto anterior $I_{\text{emp}}^U[\mathbf{F}(\mathbf{x}; \mathbf{W}, D)]$ y $I_{\text{emp}}^R[\mathbf{F}(\mathbf{x}; \mathbf{W}, D)]$. Por lo tanto en cada caso aquel \mathbf{W} óptimo hará que $\mathbf{F}(\mathbf{x}; \mathbf{W})$ minimice el funcional asociado. Por lo tanto buscamos un conjunto de parámetros óptimos tal que,

$$\mathbf{W}_*^U = \arg \min_{\mathbf{w}} I_{\text{emp}}^U[\mathbf{F}(\mathbf{x}; \mathbf{W}, D)] = \arg \min_{\mathbf{w}} \frac{1}{2N} \sum_{i=0}^{N-1} \|\mathbf{x}_i - \mathbf{F}(\mathbf{x}_i; \mathbf{W})\|^2$$

$$\mathbf{W}_*^R = \arg \min_{\mathbf{w}} I_{\text{emp}}^R[\mathbf{F}(\mathbf{x}; \mathbf{W}, D)] = \arg \min_{\mathbf{w}} \frac{1}{2N} \sum_{i=0}^{N-1} \|\mathbf{y}_i - \mathbf{F}(\mathbf{x}_i; \mathbf{W})\|^2$$

2.3.2. Método de optimización.

Una vez se ha propuesto una función de coste a minimizar cuya solución permita obtener un modelo *cercano* a aquel visto únicamente a través de un número de ejemplos, basta proponer un método de optimización que nos permita a través del conjunto de entrenamiento minimizar de forma efectiva dicha función de coste para obtener así la solución deseada. Al aplicar un método de optimización determinado sobre una función de coste se obtiene el denominado algoritmo de optimización (o de aprendizaje).

¿Qué es un algoritmo de optimización?

Los algoritmos de optimización suelen ser *ecuaciones iterativas de punto fijo* del tipo

$$\mathbf{W}[k+1] = \mathbf{f}(J(\mathbf{W}[k]))$$

donde \mathbf{W} es el conjunto de parámetros a calcular, $J(\mathbf{W})$ es la función de coste a minimizar y \mathbf{f} es una función definida sobre J que depende del método de optimización empleado. Estos algoritmos convergen en el punto (fijo) $\mathbf{W}^* = \mathbf{f}(J(\mathbf{W}^*))$. Si todo ha ido bien, dicha solución minimiza J . En general el algoritmo converge de forma asintótica en $k \rightarrow \infty$. En la práctica, el algoritmo converge en tiempo finito debido a que la precisión de cálculo es limitada por lo que a partir de un número determinado de decimales el algoritmo se para.

Aspectos que influyen en el rendimiento de un método de optimización.

Estaríamos interesados en utilizar aquel algoritmo que converja lo más rápidamente posible hacia un mínimo global de J utilizando el menor número de operaciones posibles. En la práctica este algoritmo ideal será difícil, sino imposible, de conseguir debido a una serie de aspectos que detallaremos a continuación:

- **La velocidad de convergencia del método de optimización.** Cada método de optimización converge de forma asintótica hacia la solución \mathbf{W}^* con una velocidad determinada (velocidad de convergencia). Esta velocidad se cuantifica midiendo, cuando el algoritmo está muy cerca de la solución, cuanto se avanza hacia la solución entre iteraciones. Para ello se divide la distancia que hay a la solución \mathbf{W}^* en dos puntos seguidos $\mathbf{W}[k]$ y $\mathbf{W}[k+1]$. De forma precisa, se dice que el método es de orden m si:

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{W}[k+1] - \mathbf{W}^*\|}{\|\mathbf{W}[k] - \mathbf{W}^*\|^m} = c \neq 0$$

En general a medida que se consiguen métodos que convergen con mayor velocidad hacia la solución, es decir métodos de orden superior, la complejidad de los algoritmos resultantes aumenta puesto que estos algoritmos necesitan mayor información.

- **La complejidad de cálculo del método de optimización.** Los algoritmos de optimización tienen mayor complejidad a medida que el número de operaciones necesarias por iteración aumenta. Como ya se ha comentado, los métodos más rápidos tienden a necesitar más información a la hora de avanzar hacia la solución. Por ello, el número de operaciones requeridas por iteración suele aumentar a medida que la velocidad de convergencia es mejor.
- **La complejidad del problema de optimización a resolver.** A medida que el problema a optimizar (p.e. la función de coste) es más complejo, el algoritmo de optimización deberá efectuar mayor número de operaciones. Así finalmente el número de operaciones que deberá efectuar el algoritmo de optimización dependerá de: i) el método de optimización y ii) la complejidad del problema. La complejidad del problema a optimizar en nuestro caso depende de:
 1. Las dimensiones de los espacios de entrada \mathbf{X} ($\in \mathfrak{R}^p$) y salida \mathbf{Y} ($\in \mathfrak{R}^m$), p y m .
 2. El número de muestras de entrenamiento, N .
 3. El número de parámetros libres de la función parametrizable $\mathbf{F}(\mathbf{x}; \mathbf{W})$, $|\mathbf{W}|$.
- **Los errores cometidos por el método de optimización.** Buscamos minimizar globalmente J . En realidad la solución obtenida por el algoritmo no siempre cumplirá este requisito. Es habitual utilizar técnicas que únicamente son capaces de buscar un mínimo local de J a partir de del punto de partida $\mathbf{W}[0]$. Además nos podemos encontrar incluso que el algoritmo no converja exactamente en ese mínimo local. Todo este tipo de errores asociados al algoritmo de optimización que hacen converger hacia una solución diferente de un mínimo global de J es lo que se conoce como **errores de optimización**.
- **Las condiciones iniciales del algoritmo de optimización.** El valor inicial de la secuencia $\{\mathbf{W}[k]\}$ suele ser de vital importancia en muchos algoritmos de optimización. Pensemos p.e. en que si $\mathbf{W}[0]$ esta cerca de \mathbf{W}^* el algoritmo tendrá mayores posibilidades de alcanzar \mathbf{W}^* y además utilizará para ello un número de iteraciones menor. En general será difícil obtener un valor de $\mathbf{W}[0]$ adecuado puesto que esto implicaría tener un cierto conocimiento de la solución deseada \mathbf{W}^* . En algunos casos esta información está disponible ya que se tiene un conocimiento a priori sobre el tipo de solución que se busca por lo que esta se puede utilizar para crear una buena condición inicial $\mathbf{W}[0]$.

¿De qué depende el tiempo de ejecución de un algoritmo de entrenamiento?

Como hemos visto los métodos de orden superior convergen hacia la solución con una mayor velocidad. Sin embargo suelen emplear un mayor número de operaciones por iteración que suele aumentar de forma exponencial a medida que el problema de optimización es más complejo. Por lo tanto dado un problema de optimizar, el uso de un método de orden superior no garantiza que el **tiempo de entrenamiento** sea menor que el de un método de orden inferior ya que el número de operaciones del primero puede ser especialmente prohibitivo para el problema en cuestión.

En general podremos decir que el **tiempo de entrenamiento** (t_e), entendido como el tiempo que transcurre hasta que el algoritmo cumple una condición de parada, puede responder a la siguiente relación:

$$t_e = I \text{ it} * F \frac{\text{flops}}{\text{it}} * C \frac{\text{seg}}{\text{flops}} = I * F * C \text{ (seg.)}$$

El número de iteraciones efectuada por el algoritmo I depende del criterio de parada, de las condiciones iniciales y de la velocidad de convergencia del algoritmo de optimización. El número de operaciones en coma flotante por iteración F depende de la complejidad del problema y del método de optimización. Finalmente el tiempo C que se tarda en efectuar una operación en coma flotante (1 flop) depende del ordenador empleado.

Métodos de optimización seguidores de trayectorias.

Suponed el siguiente problema de optimización:

$$\text{Minimizar la función escalar } J(\mathbf{W}) \text{ con } \mathbf{W} \in \mathfrak{R}^H$$

Los métodos iterativos que se basan en seguir una trayectoria determinada para minimizar $J(\mathbf{W})$, utilizan la siguiente ecuación iterativa para generar una secuencia de puntos de búsqueda $\mathbf{W}[k]$:

$$\mathbf{W}[k+1] = \mathbf{W}[k] + \alpha[k] \mathbf{d}[k]$$

siendo $\alpha[k] > 0$ la función escalar que determina la longitud del salto a dar en la dirección del vector $\mathbf{d}[k]$.

Existen diferentes formas de obtener $\alpha[k]$ y $\mathbf{d}[k]$. Para el cálculo de la dirección $\mathbf{d}[k]$, comentaremos aquí únicamente los cuatro métodos que se emplean con mayor frecuencia:

□ **Descenso de gradiente:**

$$\mathbf{d}[k] = -\nabla J(\mathbf{W}[k]) = -\nabla J(\mathbf{W}) \Big|_{\mathbf{W} = \mathbf{W}[k]} = \begin{bmatrix} \frac{\partial J(\mathbf{W}[k])}{\partial w_1} \\ \frac{\partial J(\mathbf{W}[k])}{\partial w_H} \end{bmatrix}$$

Este método es el más simple, es decir el que necesita un menor número de operaciones. El algoritmo desciende a través de la dirección de máxima varianza de J que es la apunta el gradiente $\nabla J(\mathbf{W})$. Su velocidad de convergencia es habitualmente lenta y en algunos casos extremadamente lenta (p.e. el método es de orden inferior a 1 para la mayoría de problemas). En la sección 2.3.2.1. hablaremos con mayor detalle de este método puesto que es la base de muchos de los algoritmos de aprendizaje que veremos a lo largo de este libro.

□ **Newton:**

$$\mathbf{d}[k] = -\mathbf{H}[k]^{-1} \nabla J(\mathbf{W}[k])$$

siendo \mathbf{H}^{-1} el inverso de la matriz Hessiana \mathbf{H} que se calcula de la siguiente manera:

$$\mathbf{H} = \nabla^2 J(\mathbf{W}[k]) = \begin{bmatrix} \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_1^2} & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_1 \partial w_H} \\ \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_2 \partial w_1} & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_2^2} & \dots & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_2 \partial w_H} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_H \partial w_1} & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_H \partial w_2} & \dots & \frac{\partial^2 J(\mathbf{W}[k])}{\partial w_H^2} \end{bmatrix}$$

La velocidad de convergencia de Newton es superlineal (orden mayor que uno) y muchas veces cuadrática (orden dos) (p.e. cerca de un mínimo). Aunque en la mayoría de casos el número de operaciones a efectuar es especialmente prohibitivo. Además para el uso correcto de este método se han de tener presente algunas deficiencias que presenta. Puede divergir violentamente si el algoritmo no está cerca de un mínimo. Por otro lado, \mathbf{H} debe estar definida positivamente, es decir las componentes de la matriz Hessiana deben ser positivas ($H_{ij} > 0$). Si no es así la solución obtenida por el algoritmo tiende a aproximar un máximo o un punto de inflexión de J . Finalmente, la inversión de \mathbf{H} puede presentar en algunos casos problemas de estabilidad numérica.

□ **Quasi-Newton:**

Puesto que el cálculo del inverso de \mathbf{H} puede implicar un número excesivo de operaciones, existen métodos denominados quasi-Newton que aproximan \mathbf{H}^{-1} con una matriz simétrica $H \times H$ \mathbf{B} de la siguiente manera:

$$\mathbf{d}[k] = -\mathbf{B}[k] \nabla J(\mathbf{W}[k])$$

Aunque estos algoritmos pueden ser extremadamente eficientes, deben almacenar y calcular la matriz \mathbf{B} que es totalmente densa.

□ **Gradiente conjugado:**

Los algoritmos basados en la técnica del gradiente conjugado pretenden conseguir una buena velocidad de convergencia con relación al número de operaciones que efectúan utilizando para ello un número de operaciones y memoria reducidos. Evitan el cálculo de \mathbf{H} , utilizando una combinación lineal del gradiente actual y las direcciones de búsqueda previas para el cálculo de $\mathbf{d}[k]$. En la forma más simple $\mathbf{d}[k]$ se calcula como

$$\mathbf{d}[k] = -\nabla J(\mathbf{W}[k]) + \beta[k] \mathbf{d}[k-1] \quad \text{con } \mathbf{d}[0] = -\nabla J(\mathbf{W}[0])$$

donde $\beta[k]$ es una función escalar que debe asegurar que la secuencia de $\mathbf{d}[k]$ cumple una determinada condición.

En cuanto al cálculo de $\alpha[k]$ se refiere basta decir que depende del método empleado para obtener $\mathbf{d}[k]$. En el caso de Newton $\alpha[k]$ se suele mantener a 1. En el resto de casos puede llegar a ser una función que se calcula en cada iteración o bien una función prefijada (p.e. $\alpha[k]$ constante o linealmente decreciente).

Métodos de optimización en lotes y en línea.

Los métodos de optimización que hemos presentado minimizan una función de coste $J(\mathbf{W})$. Esta función puede construirse fundamentalmente de dos formas. Utilizando todo el conjunto de entrenamiento o únicamente una muestra (p.e. escogida aleatoriamente del conjunto de entrenamiento).

La primera solución da lugar a los **algoritmos de entrenamiento por lotes**. En este tipo de aprendizaje se dispone de una única función de coste construida a partir de todo el conjunto de entrenamiento que el algoritmo de optimización minimiza. $J(\mathbf{W}[k])$ es entonces $I_{\text{emp}}^U [\mathbf{F}(\mathbf{x}; \mathbf{W}[k], D)]$ o $I_{\text{emp}}^R [\mathbf{F}(\mathbf{x}; \mathbf{W}[k], D)]$.

En cambio en el caso de los algoritmos de entrenamiento en línea los funcionales a utilizar únicamente emplean una única muestra del conjunto de entrenamiento. Así para el caso no supervisado tendríamos que

$$J(\mathbf{W}[k]) = I_{\text{emp}}^U [\mathbf{F}(\mathbf{x}[k]; \mathbf{W}[k])] = \frac{1}{2} \|\mathbf{x}[k] - \mathbf{F}(\mathbf{x}[k]; \mathbf{W}[k])\|^2$$

donde $\mathbf{x}[k]$ es una muestra del conjunto de entrenamiento escogida p.e. de forma aleatoria. Para el caso supervisado

$$J(\mathbf{W}[k]) = I_{\text{emp}}^R [\mathbf{F}(\mathbf{x}[k]; \mathbf{W}[k])] = \frac{1}{2} \|\mathbf{y}[k] - \mathbf{F}(\mathbf{x}[k]; \mathbf{W}[k])\|^2$$

En el caso de los algoritmos en línea el algoritmo de optimización ve en cada momento una función de coste diferente que está formada únicamente por una muestra. Así en cada iteración el algoritmo tiende hacia un mínimo del $J(\mathbf{W}[k])$ actual. A medida que el número de iteraciones es elevado, el algoritmo ha visto un número suficiente de veces todas las funciones de coste posible $J(\mathbf{W}[k])$. Por lo tanto, el algoritmo de optimización debería tender hacia una solución de compromiso que aunque no minimizara ninguna $J(\mathbf{W}[k])$ concreta si resultara especialmente satisfactoria en promedio para las $\{J(\mathbf{W}[k])\}$. Para que esta solución óptima en promedio fuera realmente útil debería ser un mínimo de $J(\mathbf{W})$, la función de coste que utiliza todo el conjunto de muestras. Desde el punto de vista teórico la convergencia de los algoritmos en línea hacia un mínimo de $J(\mathbf{W})$ sólo puede ser garantizada cuando $N \rightarrow \infty$. No obstante, su uso práctico (con N finito) está bien establecido gracias al buen comportamiento que presentan experimentalmente ya que:

1. Suelen encontrar en muchos casos soluciones parecidas a las obtenidas por los algoritmos en lotes
2. Utilizan un número de operaciones mucho más reducido por iteración (de orden de N veces menos).
3. Su dinámica es mucho más ruidosa que la de sus homólogos y por lo tanto puede ayudar al algoritmo a evitar quedar atrapado en puntos indeseables (p.e. mínimos locales de J).

2.3.2.1. La técnica del gradiente (steepest-descent gradient).

Sea una función $J(\mathbf{W})$ de la que se quiere obtener una \mathbf{W}^* tal que minimice (o maximice) dicha función:

$$\mathbf{W}_* = \arg \min J(\mathbf{W}) \quad (\text{o} \quad \mathbf{W}_* = \arg \max J(\mathbf{W}))$$

El método del gradiente persigue encontrar dicha \mathbf{W}^* con el siguiente algoritmo iterativo:

$$\mathbf{W}[k+1] = \mathbf{W}[k] + c \nabla J(\mathbf{W})_{\mathbf{W}=\mathbf{W}[k]}; \quad \text{con} \quad \mathbf{W}[0] \text{ dado} \quad \text{y} \quad \nabla J(\mathbf{W}) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_H} \end{bmatrix}$$

donde $c > 0$ para encontrar el máximo y $c < 0$ para encontrar el mínimo y $\nabla J(\mathbf{W})$ es el gradiente de la función respecto a \mathbf{W} .

Se puede demostrar que la dirección hacia la que apunta el gradiente es la de máxima variación de la función por lo que subiendo o bajando por ella podemos llegar a un **máximo o mínimo local** de la función. Si c es pequeño el algoritmo seguirá de manera estrecha el camino del gradiente, aunque la convergencia será pequeña. En cambio si c es grande la convergencia inicialmente será mayor aunque el algoritmo puede llegar a oscilar entorno a la solución.

Por supuesto si queremos con esta simple técnica encontrar un mínimo o máximo global de dicha función, deberemos asegurar que el valor inicial esté cerca de él cosa en la práctica difícil sino imposible de precisar (Fig. 6). Por supuesto existen heurísticas que pueden añadirse para que esta técnica llegue con bastante más probabilidad a un mínimo o máximo global independientemente del punto de partida (por ejemplo, el método de Branin, p.104; Cichocki, 1993).

Otra limitación de esta técnica es que su velocidad de convergencia, es decir el tiempo que transcurre (el número de iteraciones) desde que se empieza en $\mathbf{W}[0]$ hasta que se llega a \mathbf{W}^* , es lenta. Es decir el orden de convergencia p del método es bajo frente a otros. Por ejemplo para una $f(\mathbf{W})=1/2 \mathbf{W}^T \mathbf{Q} \mathbf{W}$ y un valor de c óptimo, p está entre cero y uno (p.148; Wismer, 1978). La velocidad de convergencia para este caso es lineal y el método del gradiente resulta considerablemente más lento que otros métodos.

Una visión general de los algoritmos de aprendizaje basados en descenso de gradiente

Los algoritmos de aprendizaje que minimicen una función de coste J basados en la técnica del gradiente utilizan tendrán típicamente los siguiente pasos:

```

T={ $(\mathbf{x}_i, \mathbf{y}_i)$ },  $i=1..N$  /* conjunto de entrenamiento */
N=|T|; /* N= tamaño del conjunto de entrenamiento */
k=0;
c=c0;
W=W0; /*valor inicial de los pesos del perceptrón*/
while (condición parada!=TRUE){
    if (version_en_línea) {
        i= k % N; /* k módulo N */
        W+=-c*gradiente( J(W,  $(\mathbf{x}_i, \mathbf{y}_i)$  );
    }else /* versión en lote */ W+=-c*gradiente( J(W, T) );
    ++k;
}
/* J(W, .)= función de coste a minimizar */

```

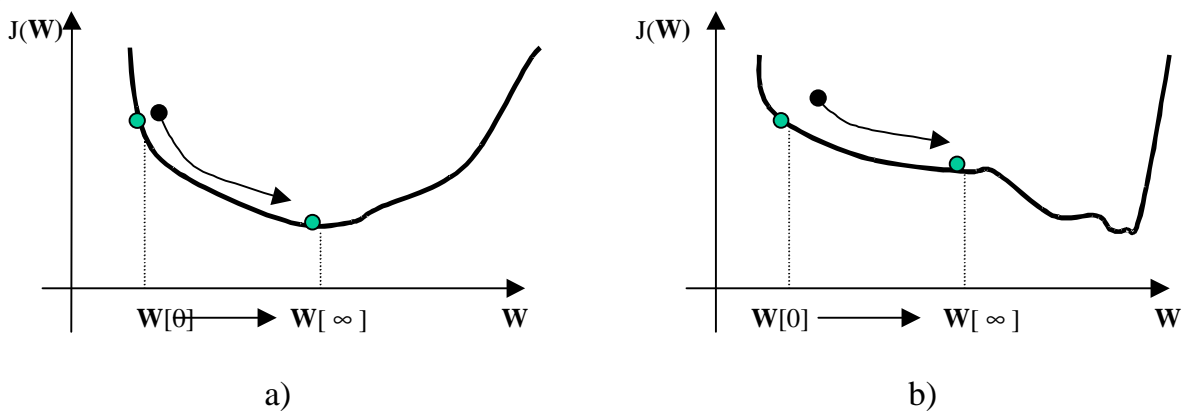


Figura 6. La técnica del gradiente. a) solución para una función con un único mínimo b) solución para una función con mínimos locales.

En este algoritmo se pueden distinguir dos versiones diferentes, una denominada **en línea** (*on-line*) y otra denominada **en lotes** (*batch*). En la primera cada una de las muestras de entrenamiento $(\mathbf{x}_i, \mathbf{y}_i) \in T$ es pasada de forma cíclica y el algoritmo realiza en cada iteración descenso de gradiente sobre una función diferente

$J(\mathbf{W}, (\mathbf{x}_i, \mathbf{y}_i))$ que depende del valor actual de vector de pesos y de la muestra de entrenamiento actual $(\mathbf{x}_i, \mathbf{y}_i)$. La idea es que minimizar cada $J(\mathbf{W}, (\mathbf{x}_i, \mathbf{y}_i))$ ayuda a minimizar al resto puesto que todas ellas están definidas con el mismo propósito. Así después de varias pasadas por todo el conjunto de entrenamiento el algoritmo puede converger hacia una solución global tal que minimice la suma de todas las $J(\mathbf{W}, (\mathbf{x}_i, \mathbf{y}_i))$ que han sido de facto minimizadas localmente. En cambio la segunda versión minimiza una única función global definida sobre \mathbf{W} y todo el conjunto de entrenamiento. Así en cada iteración se calcula de nuevo el valor actualizado de J que utiliza *todo* el conjunto de entrenamiento. Como vemos es más costoso desde un punto de vista computacional, aunque en algunos casos puede resultar más estable que la versión en línea.

En cuando al criterio de parada se refiere, podemos utilizar al menos las siguientes condiciones:

1. $k \geq$ número de iteraciones prefijado
2. $J(\mathbf{W}, T) <$ cantidad prefijada
3. $\|\nabla J(\mathbf{W}, T)\| <$ cantidad prefijada
4. combinaciones de las anteriores

No obstante como veremos más adelante, debido a ciertas limitaciones inherentes al aprendizaje con ejemplos se suelen emplear otros criterios de parada que aseguren una buena generalización de la solución obtenida.

2.3.3. Sistema dinámico.

El algoritmo de aprendizaje resultante de aplicar un método de optimización sobre una función de coste es un sistema dinámico puesto que es un sistema de ecuaciones que evolucionan en el tiempo. El algoritmo de aprendizaje converge cuando el sistema dinámico alcanza un atractor. Este atractor debería coincidir con un mínimo, local al menos, de la función de coste a minimizar. Durante el aprendizaje el sistema dinámico puede huir de repulsores (p.e. máximos de la función de coste) antes de ser atraído por un atractor, o bien quedar atrapado en ciertas órbitas que giran en torno a algún punto determinado. En fin, la dinámica del sistema dinámico determina el tiempo que dura el aprendizaje y el tipo de solución obtenida por el sistema de aprendizaje.

2.3.4. Criterio de parada.

Finalmente, el criterio de parada es de una importancia vital en general en todo algoritmo de optimización y en este caso concreto en todo algoritmo de entrenamiento de un sistema de aprendizaje basado en ejemplos. Ha de servir fundamentalmente para:

1. *Reducir el tiempo de aprendizaje.* Es frecuente al utilizar ciertos métodos de optimización que la llegada al mínimo sea muy lenta por lo que es aconseja parar antes.
2. *Asegurar que el sistema de aprendizaje tenga una buena generalización.* Como veremos en el siguiente apartado, debido a ciertas limitaciones del aprendizaje estadístico aconsejan parar el entrenamiento antes de alcanzar un mínimo de la función de coste.

2.4. Limitaciones del aprendizaje estadístico.

El aprendizaje estadístico utiliza recursos finitos, es decir, un número limitado de ejemplos, un aproximador con una capacidad de aproximación limitada y un tiempo de aprendizaje limitado. Además como hemos visto los algoritmos de optimización que se suelen emplear no consiguen converger hacia un mínimo global de la función. Por ello se hace necesario precisar todas las fuentes de error inherentes al aprendizaje estadístico y que por consiguiente determinan el error de

generalización, esto es, una medida de la capacidad de generalización del modelo inferido (p.e. una red neuronal).

En los siguientes puntos veremos dos formas complementarias de medir el error de generalización. La primera se basa en los **errores de aproximación y estimación**. En cambio la segunda utiliza el **sesgo** y la **varianza**. Posteriormente introduciremos dos de los fenómenos que se producen de manera más habitual en el entrenamiento y que provocan un error de generalización notable en el modelo inferido. Nos referimos al **sobreajuste** y el **sobreentrenamiento**. Finalmente comentaremos una simple estrategia, denominada **parada temprana**, que palia en cierto grado las limitaciones inherentes al aprendizaje estadístico basado en la minimización de una función de coste.

2.4.1. Errores de aproximación, estimación y optimización.

Como hemos comentado la solución obtenida por el sistema de aprendizaje es aquella que minimiza (p.e. localmente o de forma aproximada) un funcional empírico I_{emp} , construido con un conjunto de entrenamiento D_N de tamaño N . Además la función obtenida pertenece a un espacio de hipótesis reducido que denominaremos H_l donde l es un escalar que indica la complejidad del espacio de hipótesis (p.e. l en una RBF podría ser igual al número de funciones básicas). Así la función que obtenemos es la siguiente:

$$\mathbf{F}_{l,N}^{\text{local}}(\mathbf{x}) = \arg \min_{\mathbf{F} \in H_l} \text{local } I_{\text{emp}}[\mathbf{F}; D_N]$$

o

$$\mathbf{F}_{l,N}^{\text{aprox}}(\mathbf{x}) = \arg \min_{\mathbf{F} \in H_l} \text{aprox } I_{\text{emp}}[\mathbf{F}; D_N]$$

Si el algoritmo de optimización hubiera minimizado correctamente la función que deberíamos haber obtenido es

$$\mathbf{F}_{l,N}(\mathbf{x}) = \arg \min_{\mathbf{F} \in H_l} I_{\text{emp}}[\mathbf{F}; D_N]$$

De esta manera aparece un primer error que se debe al uso de un algoritmo de optimización que no minimiza correctamente I_{emp} . A este error lo llamaremos **error de optimización**.

En realidad la solución que desearíamos encontrar en el espacio de hipótesis H_l es aquella que se calculara con un conjunto de entrenamiento infinito. Así desde el punto de vista estadístico el cálculo sería exacto. Por consiguiente la mejor solución posible en H_l sería

$$\mathbf{F}_l(\mathbf{x}) = \arg \min_{\mathbf{F} \in H_l} \lim_{N \rightarrow \infty} I_{\text{emp}}[\mathbf{F}; D_N]$$

Aquí se introduce por lo tanto una segunda fuente de error debido a utilizar en el aprendizaje un conjunto de entrenamiento de tamaño finito. Este error se conoce con el nombre de **error de estimación**.

En general la solución óptima no tiene porque residir en H_l . En consecuencia si utilizamos un espacio de hipótesis infinito (p.e. H_l con $l \rightarrow \infty$) o espacio de hipótesis de todas las funciones posibles que denominaremos G , la mejor solución existente sería

$$\mathbf{F}_o(\mathbf{x}) = \arg \min_{\mathbf{F} \in G} \lim_{N \rightarrow \infty} I_{\text{emp}}[\mathbf{F}; D_N]$$

Vemos finalmente la aparición de una tercera fuente de error debido a que en el aprendizaje la búsqueda se realiza en un espacio de hipótesis mucho menor al espacio G . Este tercer error se denomina **error de aproximación**. En la figura 7 se muestran los tres tipos de error.

Suponiendo un error de optimización nulo, el **error de generalización** del modelo inferido a partir del aprendizaje estadístico es igual a:

Error de generalización = error de aproximación + error de estimación + factor de acoplamiento

En general los errores de aproximación y de estimación están estrechamente relacionados. Dado un conjunto de entrenamiento finito de tamaño fijo, si aumentamos el espacio de hipótesis H_1 (p.e. aumentamos el número de funciones básicas en una RBF o el número de nodos en un MLP) disminuiríamos el error de aproximación puesto que podremos encontrar una solución más compleja. Por el contrario al aumentar el poder computacional del aproximador, los valores de los parámetros estimados en el entrenamiento tienden a ser menos fiables desde el punto de vista estadístico. Así el error de estimación aumenta. En la práctica para asegurar una buena generalización (esto es, un error de generalización bajo) se deberá conseguir una nivelación entre ambos errores.

2.4.2 Sesgo y varianza.

Otra forma posible de medir el error de generalización, proveniente de la inferencia estadística, se basa en el uso del sesgo y la varianza.

El aprendiz obtiene a partir de un único conjunto de entrenamiento D_N (de tamaño N) una función $F_{1,N}$ que modela la estructura computacional que se observa en D_N . En general cualquier medida que pretenda cuantificar el error de generalización deberá tener presente la totalidad de conjuntos de tamaño N posibles $\{D_N\}$ ya que existe una variabilidad estadística entre cada conjunto de muestras. Así podemos obtener la media $\bar{F}_{1,N}$ y varianza $\sigma^2_{F_{1,N}}$ de $F_{1,N}$ en función de los conjuntos de tamaño N D_N :

$$\bar{F}_{1,N}(\mathbf{x}) = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M F_{1,N}(\mathbf{x}; D_N^i)$$

$$\sigma^2_{F_{1,N}}(\mathbf{x}) = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M (F_{1,N}(\mathbf{x}; D_N^i) - \bar{F}_{1,N}(\mathbf{x}))^2$$

donde D_N^i es una de las realizaciones posibles de tamaño N .

La **media** nos indica aquella solución a la que el aprendiz tiende en promedio al utilizar $\{D_N\}$. Dicho de otra manera, sería aquella solución que el aprendiz obtendría si utilizara un conjunto D_N típico.

Por otro lado, la **varianza** nos señala la dispersión de las soluciones obtenidas para diferentes conjuntos D_N respecto a la solución a la que en media se tiende (o solución típica) $\bar{F}_{1,N}$. Esta medida nos puede servir para cuantificar la dependencia de la solución respecto al conjunto de entrenamiento. Por ejemplo, $\sigma^2_{F_{1,N}} \rightarrow 0$ nos indicaría que la diferencia entre las soluciones que se obtendrían al utilizar conjuntos de entrenamientos diferentes (pero del mismo tamaño) sería prácticamente nula.

Finalmente, utilizaremos una tercera estadística que denominaremos **sesgo** que mide la distancia euclidiana entre $\bar{F}_{1,N}$ y F_0 , esto es, cuanto se aleja el modelo obtenido del óptimo. Se puede demostrar que si el error de optimización es nulo, el error de generalización queda entonces determinado por la siguiente relación:

$$\text{Error de generalización} = \text{sesgo}^2 (F_O, F_{1,N}) + \text{varianza}$$

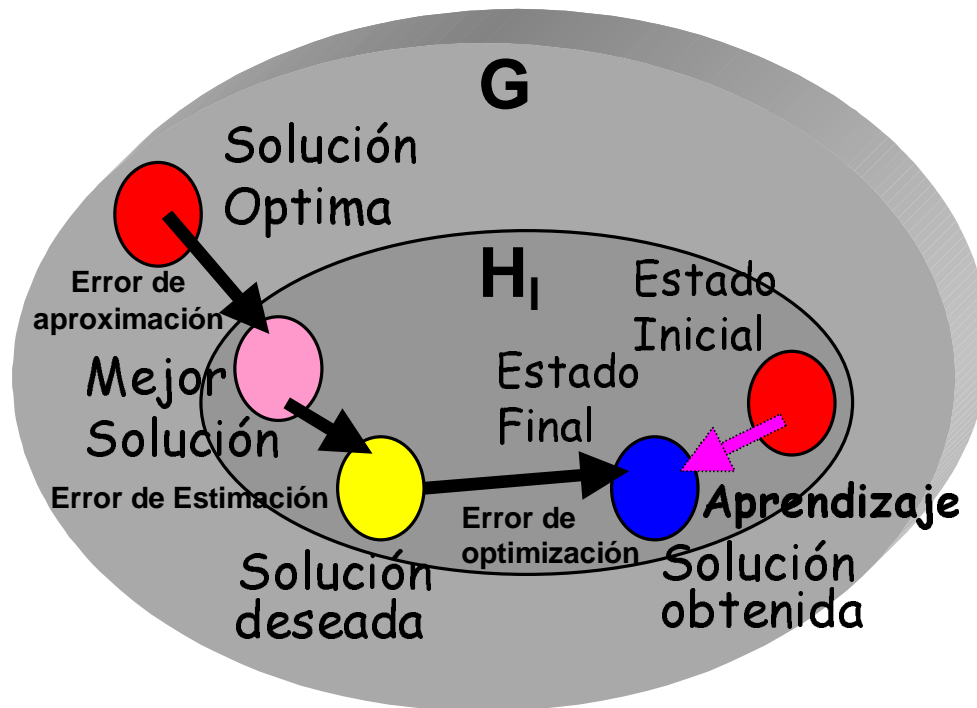


Figura 7. Errores cometidos durante el aprendizaje: 1) debido a buscar en un espacio de hipótesis más reducido se comete un error de aproximación ya que la función F_o no reside en H_b , 2) debido a que $\mathbb{I}[F]$ se estima con $\mathbb{I}_{emp}[F]$, en lugar de F_1 se obtiene $F_{1,N}$ cometiéndose un error debido a dicha estimación 3) finalmente, el algoritmo de optimización minimiza localmente el funcional empírico con $\mathbb{I}_{emp}[F]$ por lo que existe un error de optimización

Para asegurar un error de generalización bajo se deberá nivelar el valor del sesgo y la varianza ya, que al igual que en el caso anterior, son estadísticas que dependen mutuamente entre sí. A medida que el sesgo se reduce, la capacidad de aproximación del aprendiz aumenta y por consiguiente el modelo a inferir tiende a tener un mayor número de parámetros. Esto provocará que la estimación de esos parámetros (dado un tamaño fijo de muestras) sea estadísticamente menos fiable y por ello dependa más de una realización concreta del conjunto de entrenamiento D_N . Así habrá una mayor variabilidad de los valores estimados con relación a $\{D_N\}$ por lo que la varianza aumentará. En cambio, si el sesgo aumenta habrá menos parámetros que estimar y por lo tanto la varianza disminuirá ya que la estimación será más fiable y por ello la variabilidad de la solución obtenida para diferentes D_N será menor.

Como se habrá podido apreciar ya existe una estrecha relación entre los errores de aproximación y estimación, el sesgo y la varianza (ver fig. 8). Se puede demostrar (ver anexo) que el sesgo de hecho incluye al error de aproximación y al factor de acoplamiento entre los errores de estimación y aproximación. En cambio, el error de estimación incluye a la varianza. Es decir:

$$\begin{aligned} \text{sesgo}^2 (F_O, F_{1,N}) &= \text{error de aproximación} + \text{sesgo}^2 (F_1, F_{1,N}) + \text{factor de acoplamiento} \\ \text{error de estimación} &= \text{sesgo}^2 (F_1, F_{1,N}) + \text{varianza} \end{aligned}$$

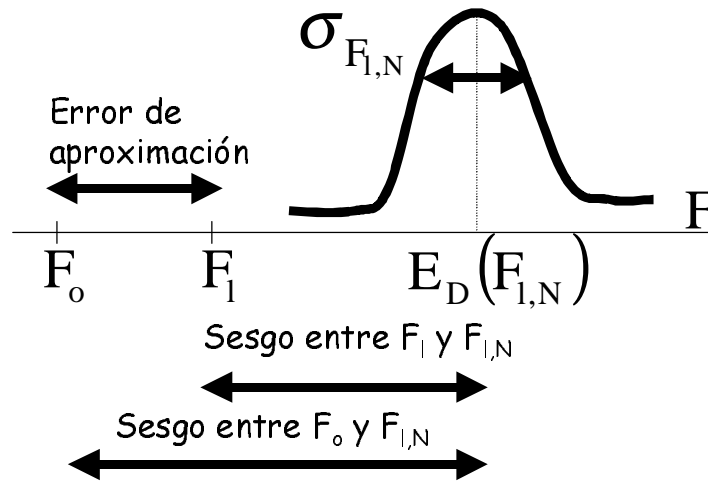


Figura 8. Relación entre sesgo, varianza y error de aproximación.

2.4.3. Sobreajuste (Overfitting).

Dado un conjunto de entrenamiento D_N (de tamaño N), a medida que el número de parámetros del modelo aumenta el aprendizaje tiende a obtener una solución basada más en la interpolación que en la aproximación. Es decir la función obtenida tiende a ajustarse más a los puntos de entrenamiento que ve. Por ello, la varianza aumenta ya que la solución obtenida depende más del conjunto de entrenamiento. No obstante, el sesgo se reduce ya que existen más grados de libertad para obtener una solución similar a la óptima. En el caso límite, es decir, cuando la función es un simple interpolador, el sesgo sería mínimo ya que siempre se podría ajustar la solución a cualquier D_N aunque la varianza sería máxima ya que esta dependería completamente del conjunto de muestras. Estaríamos entonces en el límite superior de una región del espacio de soluciones con **sobreajuste** a D_N (ver fig. 9). En cambio si el modelo tiene muy pocos parámetros que calcular con relación a N , la solución obtenida tenderá a depender menos del conjunto de entrenamiento aunque entonces la solución puede tener un sesgo elevado ya que la capacidad aproximativa es reducida. En este caso estaríamos en una región de espacio de soluciones con **escaso ajuste** a D_N . En la tabla 1 y la figura 10 quedan resumidos el sesgo y la varianza de soluciones con sobreajuste y escaso ajuste.

	<i>Sobreajuste (Overfitting)</i>	<i>Ajuste escaso (Underfitting)</i>
Sesgo	Bajo	Alto
Error de aproximación	Reducido	Elevado
Varianza	Alta	Baja
Error de estimación	Elevado	Reducido

Tabla 1. Error de generalización con soluciones con sobreajuste o escaso ajuste.

2.4.4. Sobreentrenamiento (Overtraining).

A medida que avanza el entrenamiento de un modelo (que supondremos con un número de parámetros fijo) a partir de un conjunto de muestras D_N (de tamaño N), la función de coste asociada tiende a ser minimizada. El mínimo de la función de coste lleva asociado un valor de los parámetros del modelo que se corresponde a la estimación deseada. Sin embargo, como ya se ha visto, existe un error entre la estimación de dichos parámetros y los óptimos debido al uso de un tamaño finito de datos. En

consecuencia, finalizar el entrenamiento cuando se alcanza el mínimo de la función de coste puede ser espacialmente nocivo cuando el error de estimación sea especialmente elevado. Este fenómeno se conoce con el nombre de **sobreentrenamiento**, ya que el aprendiz se acerca demasiado a un mínimo de la función de coste que lleva asociado un error de estimación elevado.

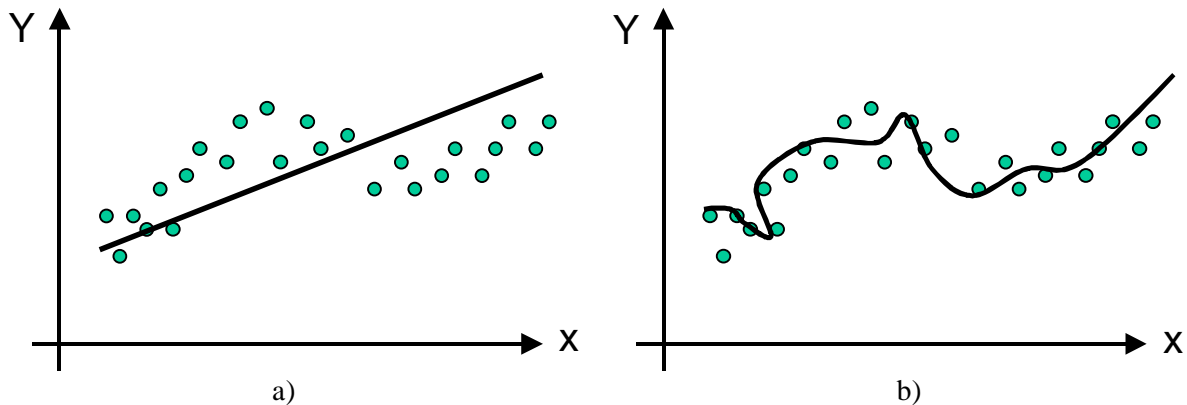


Fig. 9. a) Escaso ajuste del modelo (lineal en este caso) a los datos observados que parecen corresponder una función cúbica. b) Sobreajuste del modelo a los datos ya que el orden del modelo es superior a 3.

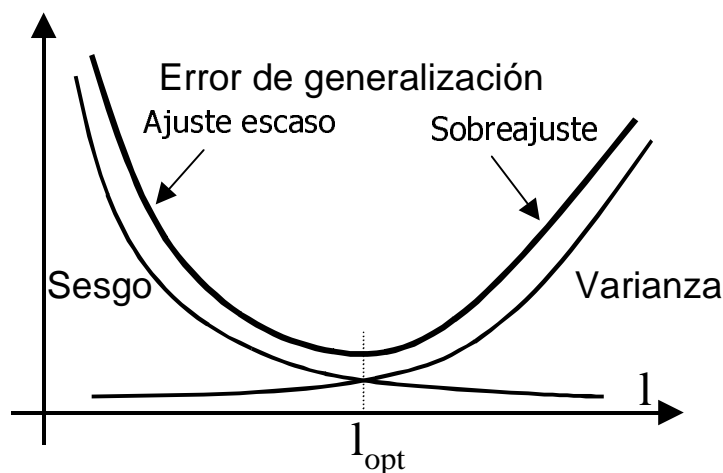


Fig. 10. Evolución del error de generalización en función de la complejidad del modelo l del espacio de hipótesis H_l . Cuando la complejidad es baja el modelo se ajusta pobremente al problema. Así el error de aproximación es elevado. A medida que l crece el error de generalización decrece hasta alcanzar un mínimo. Si $l > l_{opt}$ el error de generalización comienza a crecer de nuevo debido a que la solución está demasiado ajustada a los datos de entrenamiento y por consiguiente el error de estimación crece.

2.4.5. Parada temprana (*Early stopping*).

Parar el entrenamiento cuando se está cerca de un mínimo de la función de coste es únicamente conveniente si el error de estimación es muy bajo. Si el error de estimación es elevado o no se tiene información acerca del valor de dicho error resulta conveniente utilizar la técnica que se conoce como **parada temprana** (fig. 11). Con esta técnica se pretende evitar el **sobreentrenamiento**, esto es finalizar el entrenamiento cerca de un mínimo de la función de coste. Para ello el entrenamiento se acaba mucho antes. Esto se consigue utilizando otra función de coste construida con un conjunto independiente al de entrenamiento. Este nuevo conjunto de datos se denomina **conjunto de validación** V_S (de tamaño S). Así el sistema se entrena minimizando (localmente) una función de coste que utiliza D_N y la ejecución del algoritmo de entrenamiento se para cuando este alcanza un mínimo local de la función de coste que emplea V_S . Es decir buscamos calcular $\mathbf{F}_{1,N}^{local}(\mathbf{x}; D_N)$ de la siguiente manera:

$$\mathbf{F}_{1,N}^{\text{local}}(\mathbf{x}; D_N) = \arg \min_{\mathbf{F} \in H_1, k > 0} \text{local } I_{\text{emp}}[\mathbf{F}_{1,N}[k]; V_S] \text{ con}$$

$$\mathbf{F}_{1,N}[k+1] = \mathbf{Q}(I_{\text{emp}}[\mathbf{F}_{1,N}[k]; D_N])$$

$$\lim_{k \rightarrow \infty} \mathbf{F}_{1,N}[k] = \arg \min_{\mathbf{F} \in H_1} \text{local } I_{\text{emp}}[\mathbf{F}_{1,N}; D_N]$$

donde \mathbf{Q} es una función que calcula $\mathbf{F}_{1,N}$ en el instante $k+1$ a partir de una función de coste I_{emp} que utiliza $\mathbf{F}_{1,N}$ en el instante anterior k y el conjunto de entrenamiento D_N . Dicha función \mathbf{Q} se calcula a partir de un método de optimización concreto para que la secuencia de $\{\mathbf{F}_{1,N}[k], k > 0\}$ converja a un mínimo local de I_{emp} .

Desde el punto de vista teórico, el uso de un conjunto de validación para parar el entrenamiento es la única manera de garantizar la integridad de un proceso de aprendizaje basado en la minimización de una función de coste empírica. A pesar de que la fiabilidad de esta técnica puede ser únicamente asegurada cuando dicho conjunto tiene un tamaño suficientemente elevado, esta se suele emplear en la práctica con conjuntos de validación de tamaño moderado obteniéndose unos resultados suficientemente satisfactorios.

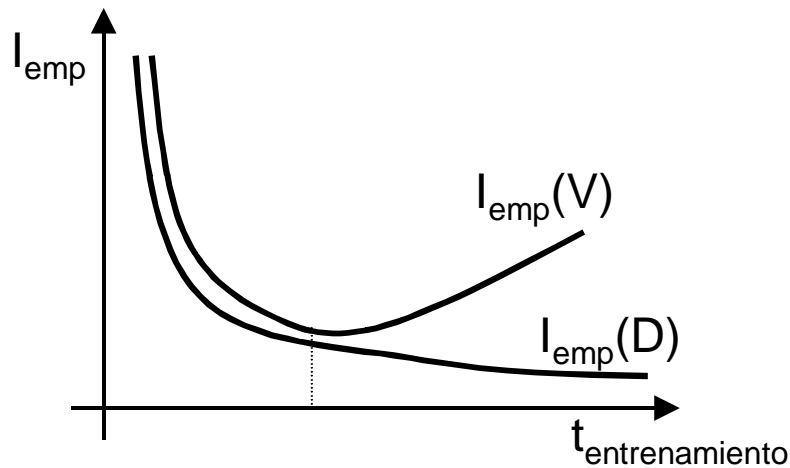


Fig. 11. Evolución típica de las funciones de coste aplicadas sobre el conjunto de entrenamiento D ($I_{\text{emp}}(D)$) y sobre el conjunto de validación V ($I_{\text{emp}}(V)$). A medida que el tiempo de entrenamiento avanza $I_{\text{emp}}(D)$ va decreciendo hasta alcanzar un mínimo. En cambio $I_{\text{emp}}(V)$ decrece y posteriormente vuelve a crecer. Esto es debido a que la estimación de los parámetros va empeorando a medida que se está más cerca de un mínimo de $I_{\text{emp}}(D)$. Para solventar este problema se emplea la técnica de la parada temprana. Con esta simple técnica, el aprendizaje se para al alcanzar un mínimo de $I_{\text{emp}}(V)$.

Referencias

(Papoulis, 1991) Papoulis, A. "Probability, Random Variables, and Stochastic Processes", McGraw-Hill, 1991

(Cichocki, 1993) Cichocki, A., Unbehauen, R. "Neural Networks for Optimization and Signal Processing", John Wiley & Sons, 1993

(Wismer, 1978) Wismer, David A., Chattergy, R. "Introduction to nonlinear optimization", North-Holland, 1978

Anexo: Error de Generalización (caso supervisado)

El error de generalización para el caso supervisado se mide con el siguiente funcional:

$$I[\mathbf{F}] = E_{XY} E_D \left[\| \mathbf{y} - \mathbf{F}(\mathbf{x}; D) \|^2 \right]$$

donde $\mathbf{F}(\mathbf{x}; D)$ es la solución obtenida por el sistema de aprendizaje al utilizar un conjunto de entrenamiento D de tamaño N , E_{XY} es el operador esperanza del vector aleatorio conjunto XY esto es

$$E_{XY}(\bullet) = \int \bullet f_{XY}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}$$

siendo f_{XY} la función densidad de probabilidad conjunta y E_D es el operador esperanza de todos los conjuntos de entrenamiento posibles de tamaño N . Este operador se define como

$$E_D(\mathbf{F}) = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M \mathbf{F}(\mathbf{x}; D_N^i)$$

siendo $\mathbf{F}(\mathbf{x}; D_N^i)$ la función obtenida por el aprendiz al utilizar el conjunto de entrenamiento D_N^i . Estos dos operadores¹¹ son intercambiables, es decir $E_{XY} E_D(\bullet) = E_D E_{XY}(\bullet)$.

$I[\mathbf{F}]$ se puede descomponer como suma del error de generalización para cada componente de los vectores aleatorios X e Y ,

$$I[\mathbf{F}] = \sum_{i=1}^m I[F_i] = \sum_{i=1}^m E_{XY_i} E_D \left[(y_i - F_i(\mathbf{x}; D))^2 \right]$$

De esta manera podemos estudiar el error de generalización de cada componente $\{I[F_i], i=1\dots m\}$ que denominaremos en lo que sigue para simplificar la notación únicamente como $I[F]$.

Debido a que $E_{XY} \left[(y - E_{y|x}(y|\mathbf{x})) (E_{y|x}(y|\mathbf{x}) - F(\mathbf{x}; D))^2 \right] = 0$, $I[F]$ admite la siguiente descomposición:

$$I[F] = E_D E_{XY} \left[(y - F(\mathbf{x}; D))^2 \right] = E_D E_{XY} \left[(E_{y|x}(y|\mathbf{x}) - F(\mathbf{x}; D))^2 \right] + E_D E_{XY} \left[(y - E_{y|x}(y|\mathbf{x}))^2 \right]$$

siendo $F_o(\mathbf{x}) = E_{y|x}(y|\mathbf{x})$ la esperanza de Y condicionada a un valor fijo de $X=\mathbf{x}$ o, también conocida como curva de regresión. Puesto que $F_o(\mathbf{x})$ depende únicamente de X y además (como Y) no depende de ningún conjunto de entrenamiento, $I[F]$ queda finalmente como:

$$I[F] = E_D E_X \left[(E_{y|x}(y|\mathbf{x}) - F(\mathbf{x}; D))^2 \right] + E_{XY} \left[(y - E_{y|x}(y|\mathbf{x}))^2 \right]$$

¹¹ En la práctica el error de generalización se puede medir utilizando los siguientes estimadores de los operadores E_X y E_D : $E_X(g(\mathbf{x})) \approx \frac{1}{N} \sum_{i=0}^{N-1} g(\mathbf{x}_i)$ y $E_D(\mathbf{F}) \approx \frac{1}{M} \sum_{i=1}^M \mathbf{F}(\mathbf{x}; D_N^i)$ donde $g(\mathbf{x})$ es cualquier función definida sobre el espacio X , $\{\mathbf{x}_i, i=0..N-1\}$ son los puntos pertenecientes al conjunto de entrenamiento, $\mathbf{F}(\mathbf{x}; D_N^i)$ es la función resultante de ser entrenada con el conjunto D_N^i y $\{D_N^i, i=1..M\}$ son los M conjuntos de entrenamiento disponibles.

En el mejor de los casos posibles $F(\mathbf{x}; D) = E_{y|\mathbf{x}}(y|\mathbf{x})$ para todo D . Entonces el error de generalización $I[F]$ sería igual a $E_{XY} \left[(y - E_{y|\mathbf{x}}(y|\mathbf{x}))^2 \right]$ siendo este el error mínimo posible. Por ejemplo en el caso de construir un clasificador este término correspondería al error cometido por el clasificador de Bayes para la clase asociada a F .

Utilizando de nuevo que $E_D(F_0(\mathbf{x})) = F_0(\mathbf{x})$ puesto que la curva de regresión no depende de D y que $E_{XY} E_D(\bullet) = E_D E_{XY}(\bullet)$, podemos llegar a obtener la siguiente descomposición del primer término de $I[F]$:

$$\begin{aligned} E_D E_X \left[(F_0(\mathbf{x}) - F(\mathbf{x}; D))^2 \right] &= \text{sesgo}^2(F_0, F) + \text{var}(F) = \\ &= E_X \left[(F_0(\mathbf{x}) - E_D(F(\mathbf{x}; D)))^2 \right] + E_D \left[(F(\mathbf{x}; D) - E_D(F(\mathbf{x}; D)))^2 \right] \end{aligned}$$

El primer término de esta descomposición se conoce con el nombre de **sesgo** al cuadrado entre F_0 y F . Mide la distancia euclidiana media (calculada en todo X) entre la curva de regresión y la solución promedia $\bar{F}(\mathbf{x}) = E_D(F(\mathbf{x}; D))$ obtenida de entrenar con un conjunto de entrenamiento de tamaño N . En cambio, el segundo término se denomina varianza de F y cuantifica la dispersión que se produce respecto de la solución promedio $\bar{F}(\mathbf{x})$ en función de D .

El sesgo y la varianza están estrechamente relacionados con los errores de aproximación y estimación. Para determinar esta relación, introduciremos las siguientes definiciones:

$$\begin{aligned} \text{sesgo}^2(F_1, F) &= E_X \left[(F_1(\mathbf{x}) - E_D(F(\mathbf{x}; D)))^2 \right] \\ \text{error de aproximación} &= E_X \left[(F_0(\mathbf{x}) - F_1(\mathbf{x}))^2 \right] \\ \text{error de estimación} &= E_X E_D \left[(F_1(\mathbf{x}) - F(\mathbf{x}; D))^2 \right] \end{aligned}$$

donde $F_1(\mathbf{x})$ es aquella función que obtendría el sistema de aprendizaje con $N \rightarrow \infty$ (y sin error de optimización) es decir $F_1(\mathbf{x}) = \arg \min_{F \in H_1} \lim_{N \rightarrow \infty} I_{\text{emp}}[F; D_N]$. Entonces, se puede llegar a demostrar las siguientes relaciones:

$$\begin{aligned} \text{sesgo}^2(F_0, F) &= \text{error de aprox.} + \text{sesgo}^2(F_1, F) + 2E_X \left[(F_0(\mathbf{x}) - F_1(\mathbf{x})) (F_1(\mathbf{x}) - E_D(F(\mathbf{x}; D))) \right] \\ \text{error de estimación} &= \text{sesgo}^2(F_1, F) + \text{var}(F) \end{aligned}$$

La distancia entre F_0 y F es la suma del error de aproximación (la distancia entre F_0 y F_1), la distancia entre F_1 y F y un término acoplado. El error de aproximación será, en general, difícil de controlar ya que dependerá de cada problema en cuestión. En cambio la distancia entre F_1 y F si se podrá controlar. Dicha distancia se puede cuantificar al estudiar la relación entre $\lim_{N \rightarrow \infty} I_{\text{emp}}(F; D_N)$ y $I_{\text{emp}}(F; D_N)$. Esta queda determinada por la siguiente relación:

$$\lim_{N \rightarrow \infty} I_{\text{emp}}(F; D_N) \leq I_{\text{emp}}(F; D_N) + \Omega\left(\frac{N}{h}\right)$$

donde Ω es una función concreta y h es la **capacidad** del sistema de aprendizaje. En general se puede decir que si N/h es grande (p.e. >20), $\lim_{N \rightarrow \infty} I_{\text{emp}}(F; D_N)$ estará cerca de $I_{\text{emp}}(F; D_N)$ y por lo tanto un

valor de pequeño de $\lim_{N \rightarrow \infty} I_{\text{emp}}(F; D_N)$ garantizará un valor pequeño de $I_{\text{emp}}(F; D_N)$. En este caso, el sesgo o distancia entre F_1 y F será prácticamente nulo. Finalmente vemos que el error de estimación es la suma de este último sesgo y de la varianza que de hecho también depende de la relación N/h .