UNIVERSIDADE
DE VIGO

Departamento de Estatística
e Investigación Operativa

Minería de Datos, 2008-2009
Redes Neuronales

Redes Neuronales: Estructura General

1. La mayor parte de las redes neuronales para problemas supervisados son modelos que:

   a) Implementan relaciones no lineales entre el input $\mathbf{x} = (x_1, ..., x_d)^T \in \mathcal{X} \subset \mathbb{R}^d$ y el output $\mathbf{y} = (y_1, ..., y_c)^T \subset \mathcal{Y} \subset \mathbb{R}^c$.

   b) Poseen típicamente dos niveles como resultado de componer dos funciones[1]:

   $$\begin{gathered}\mathbf{f}(\mathbf{x}) = \boldsymbol{\varphi}(\boldsymbol{\psi}(\mathbf{x})) = (\boldsymbol{\varphi} \circ \boldsymbol{\psi})(\mathbf{x}) \\ \boldsymbol{\psi} : \mathcal{X} \subset \mathbb{R}^d \to \mathcal{U} \subset \mathbb{R}^h \\ \boldsymbol{\varphi} : \mathcal{U} \subset \mathbb{R}^h \to \mathcal{Y} \subset \mathbb{R}^c\end{gathered}$$

   donde $\boldsymbol{\psi} = (\psi_1, ..., \psi_h)^T$ y $\boldsymbol{\varphi} = (\varphi_1, ..., \varphi_c)^T$, $\mathcal{X}$ es el espacio de entrada, $\mathcal{Y}$ el espacio de salida y $\mathcal{U}$ el espacio de variables ocultas denominado frecuentemente espacio de características (*features*).

   c) En general, el modelo anterior puede graficarse utilizando una representación conexionista propia de las redes neuronales (Figura 1).

2. En general, la consideración de un espacio de salida de varias dimensiones no supone pérdida de generalidad pues, puede implementarse mediante otras tantos modelos simultáneos con $c = 1$ (una por cada variable de salida).

   Por ello, para simplificar la exposición, supondremos $c = 1$ salvo que se indique lo contrario.

3. Así pues, suponiendo **una sola salida** y haciendo explícitos los parámetros, la expresión anterior toma la forma (nótese que h es también un parámetro de la arquitectura):

   $$\begin{gathered} f(\mathbf{x}) = \varphi\left(\sum_{j=1}^{h} c_j \psi_j(\mathbf{x}) + c_0\right) \\ \text{con: } \begin{cases} \mathbf{c} = (c_1, ..., c_h)^T \in \mathbb{R}^h \\ \psi_j = \psi(\cdot; \boldsymbol{\gamma}_j), \boldsymbol{\gamma}_j \in \mathbb{R}^{r_j}, j = 1 : h \end{cases}\end{gathered} \tag{1}$$

4. En la expresión anterior, $\psi_j(\mathbf{x}) = \psi(\mathbf{x}; \boldsymbol{\gamma}_j)$ se denomina **función de transferencia** y produce la salida de cada neurona del nivel oculto (Figura 2) con $\boldsymbol{\gamma}_j = (\mathbf{w}_j, w_{j0})$, donde ψ se denomina **función de activación**.

[1] Teóricamente, pueden construirse modelos con más niveles, pero no es frecuente en el caso de problemas supervisados. Existen también modelos con un sólo nivel (p. ej. algunos modelos clásicos de la estadísica).

Para problemas pseudo-supervisados (autoasociativos) las estructuras son similares pero pueden requerir más niveles (*capas ocultas*).

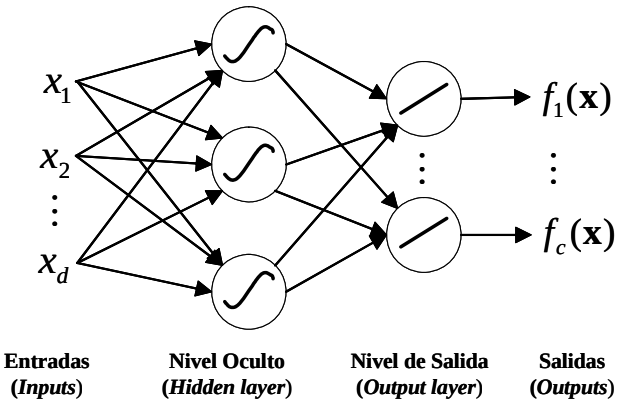

Figura 1: Estructura general de un modelo de red neuronal 'alimentada hacia adelante' (*feed-forward*).

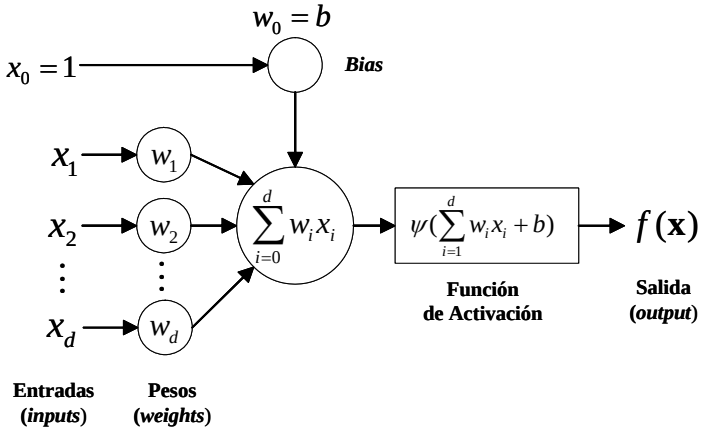

Figura 2: Estructura de una neurona en la que se muestran sus distintos elementos y se trata el *bias* o parámetro independiente como peso de una entrada constante.

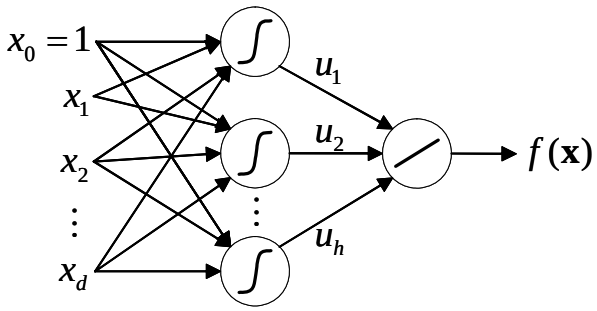

Figura 3: Arquitectura *multilayer perceptron* (MLP), (ecuación 2), para regresión. Las funciones de activación del nivel oculto suelen ser sigmoides y la del nivel de salida, lineal.

Casos Particulares

1. **Redes neuronales Multilayer Perceptron** (MLP):

   a) $\psi_j(\mathbf{x}) = \psi(\mathbf{w}_j^T\mathbf{x} + w_{j0})$ con $\mathbf{w}_j \in \mathbb{R}^d$, $j = 1 : h$, siendo ψ la **función de activación** de las unidades o **neuronas** (Figura 2) del **nivel oculto**, $\mathbf{w}_j$ el vector de parámetros de dicha unidad y $w_{j0} \in \mathbb{R}$ su valor umbral. Normalmente, la función ψ es de tipo **sigmoide**, (logística, tangente hiperbólica, etc.), pero pueden utilizarse otras que no sean polinomios.

   b) $\varphi_j(\mathbf{u}) = \varphi(\mathbf{c}_j^T\mathbf{u} + c_{j0})$, $j = 1 : c$, donde $\mathbf{u} = \boldsymbol{\psi}(\mathbf{x}) \in \mathcal{U}$, φ es la función de activación de la unidad j-ésima del **nivel de salida**, $\mathbf{c}_j \in \mathbb{R}^h$, $j = 1 : h$ el vector de pesos de dicha unidad y $c_{j0} \in \mathbb{R}$ su valor umbral. La función φ depende del problema en cuestión: para problemas de regresión suele utilizarse la función identidad , y para problemas de clasificación, la función *Heaviside* o cualquier función dicotómica.

   En particular, las redes MLP con una sóla salida ($c = 1$), responden al siguiente modelo específico (Figura 3):

$$f(\mathbf{x}) = \varphi\left(\sum_{j=1}^{h} c_j\psi(\mathbf{w}_j^T\mathbf{x} + w_{j0}) + c_0\right) \tag{2}$$

2. **Redes neuronales de funciones base radial**, (RBF):

   a) $\psi_j(\mathbf{x}) = \psi(\|\mathbf{x} - \mathbf{w}_j\| / \sigma_j)$ siendo ψ la función de activación de las unidades del nivel oculto, $\mathbf{w}_j$ el vector de pesos de dicha unidad y σ_j un parámetro umbral que determina el rango de influencia de la función de activación, (este parámetro puede generalizarse a una matriz de covarianza reflejando diferentes rangos de influencia en las distintas direcciones del espacio).

   b) $\varphi_j(\mathbf{u}) = \varphi(\mathbf{c}_j^T\mathbf{u} + c_{j0})$ con los mismos comentarios que para la red MLP, salvo que, normalmente, las redes RBF se utilizan para el problema de regresión por lo que la función φ suele ser siempre la función identidad.

   Esto no impide utilizar las redes RBF para clasificación estimando con ellas las probabilidades a posteriori de las clases y utilizando el clasificador *plug-in*.

   Con ello, la red neuronal RBF con una sóla salida ($c = 1$) responde al siguiente modelo (Figura 4):

$$f(\mathbf{x}) = \sum_{j=1}^{h} c_j\psi\left(\frac{\|\mathbf{x} - \mathbf{w}_j\|}{\sigma_j}\right) + c_0$$

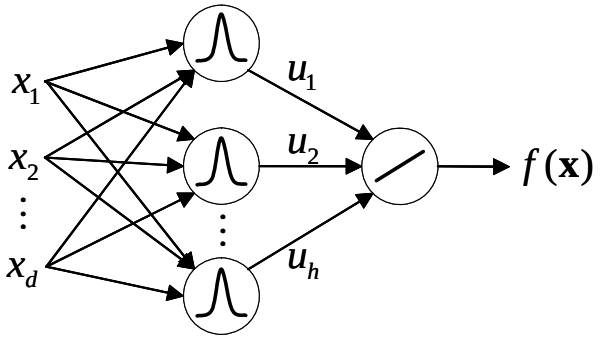

Figura 4: Arquitectura de función de base radial (RBF) (ecuación (4)) o de una support vector machine (SVM) para regresión. Las funciones de activación del nivel oculto son funciones radiales y la del nivel de salida, lineal.

Visión Conexionista de otras Técnicas Estadísticas y de Machine Learning (Problemas Supervisados)

1. **Máquinas de Vectores Soporte** (Support vector machines –SVM): poseen la misma estructura general que las redes neuronales, pudiendo ser de los dos tipos anteriores (de proyección como las MLP o radial como las RBF) según la función de transferencia $\psi_j(\mathbf{x})$ que utilicen en sus unidades ocultas. La función de activación ψ es el núcleo k.

2. **Árboles de Clasificación y Regresión** (Classification and Regression Trees –CART):

   a) $\psi_j(\mathbf{x}) = 1_{\{\mathbf{x}\in D_j\}}$ siendo $D_j = \prod_{k=1}^{d} I_k$ hiper-rectángulo en $\mathbb{R}^d$ con $I_{j,k} = [a_{j,k}, b_{j,k}]$ intervalos definidos en la dimensión de la variable x_k donde $a_{j,k}, b_{j,k} \in \mathbb{R} \cup \{-\infty, \infty\}$.

   b) $\varphi_j(\mathbf{u}) = \varphi(\mathbf{c}_j^T\mathbf{u})$ con φ la identidad.

   Con ello, los árboles de clasificación y regresión poseen como forma general:

   $$f(\mathbf{x}) = \sum_{j=1}^{h} c_j 1_{\{\mathbf{x}\in D_j\}} \text{ con } D_j = \prod_{k=1}^{d} I_k$$

   con lo que su estructura es similar a la de las redes RBF pero utilizando una función impulso d-dimensional en lugar de una función radial.

3. Además de los modelos paramétricos de regresión (lineal, polinómica, etc., p. ej. [Seber 1977], [Seber y Wild 1989]), muchos modelos de regresión y clasificación son también casos particulares del modelo general anterior. Por ejemplo:

   a) Los modelos multivariantes de regresión lineal generalizada (con transformación previa de variables), y los modelos aditivos (p. ej. [Hastie y Tibshirani 1990]), con $\varphi(u) = u$, que pueden verse como modelos de regresión lineal sobre un espacio de características (*features*) definidas por funciones de las variables de entrada.

   b) Los modelos lineales generalizados y aditivos generalizados (p. ej. [McCullagh y Nelder 1983], [Hastie y Tibshirani 1990]), para los que la función *link* se correspondería con la función inversa de φ.

c) La regresión tipo núcleo (regresión polinómico local mediante constantes), (p. ej. [Härdle 1990], [Wand y Jones 1995]), en la que $\psi_{\boldsymbol{\gamma}_j}$ es el núcleo, $\boldsymbol{\gamma}_j = (\mathbf{x}_j, \sigma_j)$ y $c_j = y_j$, $j = 1 : n$.

d) La regresión polinómico local, (p. ej. [Loader 1999], [Fan y Gijbels 1996]). Por ejemplo, si $d = 1$ para polinomios lineales se tiene $c_j = y_j$ y $\psi_{\boldsymbol{\gamma}_j}(x) = w_j(x)/\sum_{j=1}^{n} w_j(x)$ con:

$$\begin{aligned} w_j(x) &= \kappa_{\sigma_j}(x_j - x)[s_{n,2} - (x_j - x)s_{n,1}],\ j = 1 : n \\ \text{siendo:} \qquad & s_{n,j} = \sum_{j=1}^{n} \kappa_{\sigma_j}(x_j - x)(x_j - x)^j, j = 1, 2 \end{aligned}$$

donde $\kappa_{\sigma_j}(\cdot) = \kappa(\cdot/\sigma_j)/\sigma_j$ siendo κ alguna función de densidad y $\boldsymbol{\gamma}_j = (x_j, \sigma_j)$, $j = 1 : n$.

e) Los *splines*, (p. ej. [Green y Silverman 1994], [Wahba 1990], [Eubank 1988]), en los que por ejemplo, para $d = 1$: $\varphi_j(u) = u$ y las funciones $\psi_{\boldsymbol{\gamma}_j}$ son, o bien monomios x^r, $r = 0, ..., p$, y $(x - t_j)_+^p$ con t_j los *nudos* (*knots*), o bien las funciones básicas *B-splines*. Igualmente, los *smoothing splines* donde las funciones $\psi_{\boldsymbol{\gamma}_j}$ son expresiones en función del núcleo reproductor κ de un espacio de Hilbert con Núcleo Reproductor (RKHS), y términos polinómicos.

f) El modelo de regresión *projection pursuit*, ([Friedman 1987], [Friedman y Stuetzle 1981]), en el que $\varphi(u) = u$ y las funciones $\psi_{\boldsymbol{\gamma}_j}$ son suavizadores en cada proyección del espacio de entrada, donde $\boldsymbol{\gamma}_j = (\mathbf{w}_j, \boldsymbol{\alpha}_j)$, con $\mathbf{w}_j$ vector de proyección y $\boldsymbol{\alpha}_j$ vector de parámetros del suavizador correspondiente.

g) Expansiones en serie, (p. ej. [Efromovich 1999]), en las que $\varphi(u) = u$, las funciones $\psi_{\boldsymbol{\gamma}_j}$ son elementos de una base del espacio de funciones considerado y c_j son los coeficientes estimados de la expansión en términos de dicha base.

h) Caso particular de lo anterior, son las *wavelets* (p. ej. [Vidakovic 1999], [Mallat 1989], [Daubechies 1992]), (así como sus generalizaciones [Candes 1998]), donde $\varphi(u) = u$ y las funciones $\psi_{\boldsymbol{\gamma}_j}$ son los elementos de la base ortonormal de $L_2(\mathbb{R})$ constituida por las traslaciones de la función de escala (*father wavelet*), y escalamientos y traslaciones de la función *mother wavelet*, y c_j son los coeficientes estimados de la expansión.

i) Modelos basados en diccionarios constituidos por bases o estructuras (*frames*) redundantes en algún espacio de funciones, (Matching Pursuit [Mallat y Zhang 1992] y Basis Pursuit [Chen *et al.* 1999], en el que las funciones $\psi_{\boldsymbol{\gamma}_j}$ son funciones del diccionario.

j) Combinaciones de los modelos anteriores: *bagging*, (p. ej. [Breiman 1996]), *boosting*, (p. ej. [Freund y Schapire 1997]) y, en general, *stacking* ([Wolpert 1992], [Breiman 1993]).

4. Por tanto:

a) Todas estas técnicas admiten una representación conexionista como la mostrada en la Figura 1, lo que permite analizarlas desde una nueva perspectiva.

b) La consideración o no de dichas técnicas como técnicas de machine learning (redes neuronales, etc.) depende de la convención terminológica que más se adapte a nuestra finalidad.

c) El amplio abanico de modelos que pueden encuadrarse en la familia (1) sugiere que, al margen de las formas funcionales específicas de cada uno, la diferencia más profunda entre unas técnicas y otras, son los algoritmos empleados para la estimación o aprendizaje de sus parámetros.

Elementos y Tipología de las Redes Neuronales

Elementos de una Red Neuronal

1. Una red neuronal está compuesta por:

   a) Una arquitectura de unidades de proceso o neuronas.

   b) Un algoritmo de entrenamiento.

Tipología de las Redes Neuronales

1. Según su aplicación:

   a) Clasificación (implementando directamente la regla de clasificación).

   b) Regresión:

      1) Regresión.
      2) Regresión para clasificación (estimando las probabilidades a posteriori de las clases – para aplicar la regla plug-in posteriormente, u otra función discriminante sobre la que se construye la regla de clasificación).

   c) Reducción de dimensionalidad.

   d) Clustering y reducción de datos.

2. Según su arquitectura:

   a) De una o varias salidas (multi-output).

   b) De una o varias capas ocultas (multi-layer).

   c) Con alimentación hacia adelante (feed-forward) o retro-alimentada.

   d) De proyección o radial, según el método de similitud o disimilitud implementado.

3. Según su algoritmo de entrenamiento (condicionado por su arquitectura): interactivo (*on-line*) o por lotes (*batch*):

   a) Resolución de un problema de mínimos cuadrados (al modo de la regresión lineal). (RBF).

   b) Optimización no lineal de carácter local (gradiente, gradiente conjugado, hessiano): varias soluciones locales. (MLP y RBF).

   c) Algoritmos *greedy*. (RBF y MLP).

   d) Optimización global (algoritmos genéticos, optimización estocástica, etc.). (MLP y RBF).

   e) Programación cuadrática: solución única. (SVM).

Redes Neuronales de una Sóla Capa. Clasificación

El Perceptrón

1. **Modelo**: es una sóla neurona (Figura 2), [Rosemblatt 1958]:

$$\begin{array}{c} g_{\bar{\mathbf{w}}}(\mathbf{x}) = \psi(\mathbf{w}^T\mathbf{x} + w_0) = \psi(\bar{\mathbf{w}}^T\bar{\mathbf{x}}) \\ \text{con: } \left\{ \begin{array}{l} \bar{\mathbf{w}} = (w_0, \mathbf{w}^T)^T;\ \bar{\mathbf{x}} = (1, \mathbf{x}^T)^T \\ \psi(r) = \text{signo}(r) = \left\{ \begin{array}{ll} 1 & r \geq 0 \\ -1 & r < 0 \end{array} \right. \end{array} \right. \end{array} \tag{3}$$

   a) De esta forma, para $\mathcal{Y} = \{-1, 1\}$, el perceptrón implementa una regla de clasificación basada en una **función discriminante lineal**: $f_{\bar{\mathbf{w}}}(\mathbf{x}) = \bar{\mathbf{w}}^T\bar{\mathbf{x}} = \mathbf{w}^T\mathbf{x} + w_0$.

   b) Al conjunto $\{\mathbf{w}^T\mathbf{x} + w_0 = 0\}$ se le denomina **hiperplano separador**.

2. **Algoritmo de Entrenamiento**:

   a) Dada una muestra aleatoria $\mathbf{z}^n = (\mathbf{z}_1, ..., \mathbf{z}_n)$, con $\mathbf{z}_i = (\mathbf{x}_i, y_i)$, $i = 1 : n$, el aprendizaje del perceptrón utiliza como **principio inductivo**, la minimización de la siguiente función de error:

$$\hat{R}_n(\bar{\mathbf{w}}) = -\sum_{y_i f_{\bar{\mathbf{w}}}(\mathbf{x}_i) < 0} y_i f_{\bar{\mathbf{w}}}(\mathbf{x}_i) = -\sum_{y_i f_{\bar{\mathbf{w}}}(\mathbf{x}_i) < 0} y_i(\mathbf{w}^T\mathbf{x}_i + w_0)$$

   b) El algoritmo de aprendizaje del perceptrón resulta de la aplicación del algoritmo del gradiente al problema de minimización de $\hat{R}_n(\bar{\mathbf{w}})$.

      1) En cada iteración $(s+1)$-ésima, sólo los ejemplos mal clasificados modifican el hiperplano actual, mediante:

$$\bar{\mathbf{w}}(s+1) = \bar{\mathbf{w}}(s) + \rho y_i \bar{\mathbf{x}}_i \tag{4}$$

         donde $\rho > 0$ es un parámetro que controla la velocidad de convergencia.

      2) Ello provoca que, en cada iteración, se reduzca el error de los ejemplos mal clasificados:

$$-y_i\bar{\mathbf{w}}(s+1)^T\bar{\mathbf{x}}_i = -[y_i\bar{\mathbf{w}}(s)^T\bar{\mathbf{x}}_i + \rho y_i^2\bar{\mathbf{x}}_i^T\bar{\mathbf{x}}_i] < -y_i\bar{\mathbf{w}}(s)^T\bar{\mathbf{x}}_i$$

   c) Si la muestra es linealmente separable, el algoritmo converge (th. Novikoff [Vapnik 1998]) a una solución en un número finito de iteraciones que depende del margen conseguido.

   d) Si la muestra no es **linealmente separable** (no puede separarse perfectamente mediante un hiperplano), el algoritmo no finalizaría pues, la posible clasificación correcta de algún punto mal clasificado, provocaría la mala clasificación de alguno de los puntos correctos, provocando un comportamiento oscilatorio.

3. La versión interactiva (*on-line*) del aprendizaje es exactamente análoga debido a la aditividad del error.

Adaline

1. **Modelo**. Con la misma codificación $\mathcal{Y} = \{-1, 1\}$ anterior, la arquitectura original de adaline es la misma que la del perceptrón (3) (Figura 2)

2. **Algoritmo de Entrenamiento**. Adaline se diferencia del Perceptrón en que la pérdida es ahora la de **mínimos cuadrados** sobre el argumento $f_{\bar{\mathbf{w}}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ de la función de activación ψ:

$$\hat{R}_n(\bar{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f_{\bar{\mathbf{w}}}(\mathbf{x}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - \bar{\mathbf{x}}_i^T\bar{\mathbf{w}})^2 = \frac{1}{2}\left\|\mathbf{y} - \bar{\mathbf{X}}\bar{\mathbf{w}}\right\|^2$$

donde $\bar{\mathbf{X}}$ es la matriz que tiene por filas los vectores $\bar{\mathbf{x}}_i^T$, $i = 1 : n$, e $\mathbf{y} = (y_1, ..., y_n)^T$.

El algoritmo puede ser de tipo *on-line* o *batch*:

a) *Batch*. En este último caso, se obtiene analíticamente la solución de mínimos cuadrados:

$$\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y} \Rightarrow f_{\bar{\mathbf{w}}}(\mathbf{x}) = \bar{\mathbf{x}}^T(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y}$$

donde se supone que $\bar{\mathbf{X}}^T\bar{\mathbf{X}}$ es de rango máximo. La matriz $\mathbf{P}_{\bar{\mathbf{X}}} = \bar{\mathbf{X}}(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^-\bar{\mathbf{X}}^T$(simétrica e idempotente) proyecta $\mathbf{y}$ sobre $\langle\{\mathbf{x}_1, ..., \mathbf{x}_n\}\rangle$, el espacio lineal generado por $\{\mathbf{x}_i\}_{i=1}^n$.

b) *On line*. En el caso interactivo, se utiliza el algoritmo del gradiente, que da lugar a la denominada **regla delta** o de **Widrow-Hoff**, [Widrow y Hoff 1988].

1) Si denotamos $\ell_i(\bar{\mathbf{w}}) = \ell(y_i, f_{\bar{\mathbf{w}}}(\mathbf{x}_i)) = (y_i - \bar{\mathbf{x}}_i^T\bar{\mathbf{w}})^2$, entonces, para cada nuevo ejemplo $(\mathbf{x}_i, y_i)$ se tiene:

$$\begin{aligned}\bar{\mathbf{w}}(s+1) &= \bar{\mathbf{w}}(s) - \rho(s)\nabla_{\bar{\mathbf{w}}}\ell_i(\bar{\mathbf{w}}(s)) \\ &= \bar{\mathbf{w}}(s) - \rho(s)(y_i - \bar{\mathbf{x}}_i^T\bar{\mathbf{w}})\bar{\mathbf{x}}_i\end{aligned} \tag{5}$$

2) La cantidad $\delta_i = (y_i - \bar{\mathbf{x}}_i^T\bar{\mathbf{w}})$ da nombre al algoritmo.

3. **Relación con el Discriminante Lineal de Fisher**. Si la codificación de las clases es $\mathcal{Y} = \{n/n_1, -n/n_2\}$ en lugar de $\{-1, 1\}$, Adaline produce la misma regla de clasificación que el Discriminante lineal de Fisher (p. ej. [Duda *et al.* 2001, p. 242], [Bishop 1995, p. 109]).

Redes Neuronales de una Capa para la Estimación de las Probabilidades a Posteriori

1. En hipótesis de normalidad e igualdad de covarianzas para las distribuciones de las clases $\mathsf{p}_{\mathsf{X}|\mathcal{C}_j}$, $j = 1, 2$, y suponiendo desconocimiento a priori sobre la probabilidad de cada clase, la regla discriminante correspondiente al discriminante lineal de Fisher minimiza la probabilidad de error de clasificación $\mathsf{P}_{\mathsf{Z}}\{\mathsf{Y} \neq g(\mathsf{X})\}$ (p. ej. [Johnson y Wichern 1992]). Sin embargo, cuando estas hipótesis no se verifican, son recomendables otros métodos, (un análisis detallado de robustez puede verse p. ej. en [Seber 1984, p. 297 y 317]).

a) Una alternativa, consiste en transformar las variables de entrada para que se cumplan las hipótesis anteriores y se mejore la discriminación. Es lo que se conoce en la literatura como discriminante lineal generalizado, (p. ej. [Duda *et al.* 2001, p. 219], [Ripley 1996, p. 121]) y, como se verá más adelante, se consigue con la inclusión de una capa adicional de neuronas: el nivel oculto.

b) Otro camino, consiste en la utilización de la arquitectura adaline con una función de activación ψ no lineal. Pudiera parecer, que el objetivo de esta modificación, fuese la obtención de una frontera discriminante no lineal con mayor potencial para separar datos no linealmente separables. Sin embargo, las funciones de activación utilizadas suelen ser monótonas, con lo que la frontera resultante sigue siendo lineal.

En realidad, la mejora que se persigue es otra: la estimación de las probabilidades a posteriori $\mathbb{P}(\mathsf{Y} = y|\mathbf{x})$ para aproximar, mediante el método *plug-in*, la regla de Bayes, en contextos más generales que la normalidad y la igualdad de varianzas. Además, este enfoque, aporta una medida de la seguridad con la que se clasifica un nuevo ejemplo.

2. **Estimación de probabilidades a posteriori mediante regresión por mínimos cuadrados**. La estimación de las probabilidades a posteriori puede realizarse mediante regresión por mínimos cuadrados con codificación $\mathcal{Y} = \{0, 1\}$ pues, en ese caso:

$$\mathbb{E}_{\mathsf{Y}|\mathbf{x}}(\mathsf{Y}) = 1 \cdot \mathbb{P}(\mathsf{Y} = 1|\mathbf{x}) + 0 \cdot \mathbb{P}(\mathsf{Y} = 0|\mathbf{x}) = \mathbb{P}(\mathsf{Y} = 1|\mathbf{x})$$

   a) Así, mediante una estructura tipo adaline con activación **Heaviside** $\psi(r) = 1$ si $r > 0$ y $\psi(r) = 0$ si $r \leq 0$, podría implementarse la denominada regla *plug-in*:

$$g_{\hat{f}_n}(\mathbf{x}) = \begin{cases} 0 & \text{si } \hat{f}_n(\mathbf{x}) \leq \frac{1}{2} \\ 1 & \text{si } \hat{f}_n(\mathbf{x}) > \frac{1}{2} \end{cases}$$

   donde $\hat{f}_n(\mathbf{x}) = \hat{\mathbb{P}}(\mathsf{Y} = 1|\mathbf{x})$ es la estimación de las probabilidades a posteriori mediante regresión por mínimos cuadrados.

   b) Este enfoque está justificado por el siguiente teorema:

   Teorema 1 *(p. ej. [Devroye* et al. *1996, th. 2.2]) Sea* $\mathsf{f}(\mathbf{x}) = \mathbb{P}(\mathsf{Y} = 1|\mathbf{x})$ *y* g_{f} *la regla de Bayes que satisface* $R(g_{\mathsf{f}}) = \inf_g R(g)$ *con* $R(g) = \mathbb{P}_{\mathsf{Z}}[g(\mathsf{X}) \neq \mathsf{Y}]$. *La regla plug-in* g_f *construida con* f, *verifica:*

$$R(g_f) - R(g_{\mathsf{f}}) \leq 2\mathbb{E}_{\mathsf{X}}\left[|f(\mathbf{x}) - \mathsf{f}(\mathbf{x})|\right] \leq 2\left(\mathbb{E}_{\mathsf{X}}\left[|\mathsf{f}(\mathbf{x}) - f(\mathbf{x})|\right]^2\right)^{1/2}$$

   Es decir, si $\hat{f}_n$ es buen estimador de las verdaderas probabilidades a posteriori en el sentido $L_1(\mathsf{P}_{\mathsf{X}})$, (y, por la desigualdad de Cauchy-Schwartz, en el sentido $L_2(\mathsf{P}_{\mathsf{X}})$), entonces la regla *plug-in* $g_{\hat{f}_n}$ es un buen estimador de la regla de Bayes.

3. Sin embargo, el enfoque de mínimos cuadrados presenta otros problemas bien conocidos:

   a) La distribución de $\mathsf{Y}|\mathbf{x}$ es de Bernouilli, con lo que el criterio de mínimos cuadrados no es equivalente a la máxima verosimilitud, lo que no garantiza la eficiencia de los estimadores ni que $\hat{f}_n(\mathbf{x}) = \hat{\mathbb{P}}(\mathsf{Y} = 1|\mathbf{x}) \in [0, 1]$.

   b) Las perturbaciones son heterocedásticas, lo que exigiría mínimos cuadrados ponderados.

   Sólamente, en el caso de observaciones repetidas $\{(t_{ij}, \mathbf{x}_i),\ j = 1, ...n_i, i = 1 : n\}$ con $t_{ij} \in \{0, 1\}$, si n_i es suficientemente grande, se toma como codificación $y_i = \frac{1}{n_i}\sum_{j=1}^{n_i} t_{ij}$ y las clases son igualmente probables, podría utilizarse Adaline de forma aproximada ya que, entonces, $\mathsf{Y}|\mathbf{x}$ es asintóticamente normal.

Adaline con Activación Sigmoide. Regresión Logística

1. **Modelo**.

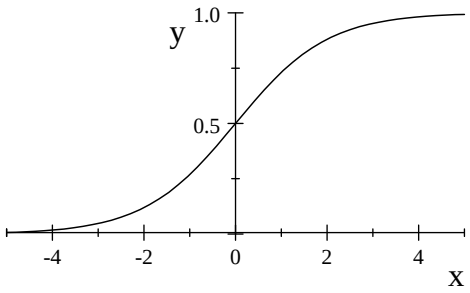

Figura 5: Función Logística: $\psi(r) = (1 + \exp(-r))^{-1}$

a) Con el fin de construir funciones discriminantes que representasen probabilidades a posteriori y, al mismo tiempo, dotar de mejores propiedades analíticas al modelo perceptrón-adaline, se sustituyó su función de activación discreta por una función de activación de tipo **sigmoide**, es decir, monótona con derivada continua y con:

$$-\infty < a = \lim_{a \to -\infty} \psi(r) < \lim_{a \to \infty} \psi(r) = a' < \infty$$

entre las que, la más frecuentemente utilizada es la **función logística**[2]:

$$\psi(r) = \frac{1}{1 + e^{-r}} = \frac{e^r}{1 + e^r}$$

a) Ahora, la neurona no representa ya una **regla de decisión** sino un **modelo discriminante** que, una vez entrenado contra una codificación $\mathcal{Y} = \{0, 1\}$, implementa una **regresión logística** (p. ej. [McCullagh y Nelder 1983], [Seber 1984, p. 308]). Ello es debido a que:

$$\ln \frac{\mathsf{p}(\mathcal{C}_1|\mathbf{x})}{1 - \mathsf{p}(\mathcal{C}_1|\mathbf{x})} = \bar{\mathbf{w}}^T \bar{\mathbf{x}} \Rightarrow \mathsf{p}(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + e^{-\bar{\mathbf{w}}^T \bar{\mathbf{x}}}}$$

donde $\mathsf{p}(\mathcal{C}_1|\mathbf{x}) \equiv \mathsf{p}_{\mathcal{C}_1|\mathbf{x}}(\mathbf{x})$ es la probabilidad a posteriori de la clase $\mathcal{C}_1$ y $\ln \frac{a}{1-a}$ es la función *link*.

b) La metodología anterior puede generalizarse a otros **modelos lineales generalizados**, utilizando como función de activación la inversa de la función *link*.
Por ejemplo, si $\psi(r) = \Phi(r) = \frac{2}{\sqrt{\pi}} \int_0^r e^{-t^2} dt$, función de distribución gausiana, (*error function*) se tiene el modelo **probit**:

$$f_{\bar{\mathbf{w}}}(\mathbf{x}) = \Phi(\bar{\mathbf{w}}^T \bar{\mathbf{x}})$$

orientado a la estimación de frecuencias binomiales que dependen del nivel $\mathbf{x}$ de un conjunto de factores, (p. ej. [McCullagh y Nelder 1983]).

2. **Algoritmo de Entrenamiento**.

a) En principio, con el apoyo del teorema 1, puede utilizarse la pérdida cuadrática:

[2]Nótese que la consideración posterior de una activación tipo Heaviside produce realmente una red neuronal con dos neuronas superpuestas (dos capas): una Adaline con activación logística para estimar las probabilidades a posteriori y una lineal con activación Heaviside para implementar la regla de clasificación. Sin embargo, esta última neurona no suele incluirse en la estructura.

1) Ahora, la función de activación logística garantiza que $\hat{f}_n(\mathbf{x}) \in [0,1]$, aunque se pierde la ventaja de un problema de optimización cuadrático.
2) Por otra parte, la pérdida cuadrática se elige al margen de la distribución de los datos, con lo que queda en cuestión la eficiencia de los estimadores.

b) Una alternativa más fundamentada consiste en utilizar la función de verosimilitud basada en la distribución condicionada:

$$\mathsf{p}_{\mathbf{Y}^n|\mathbf{x}^n}(\mathbf{y}^n;\mathbf{x}^n) = \prod_{i=1}^{n} \mathsf{f}(\mathbf{x}_i)^{y_i}[1-\mathsf{f}(\mathbf{x}_i)]^{1-y_i}$$

que da lugar a la función de pérdida de entropía cruzada:

$$\ell(y_i, f(\mathbf{x}_i)) = -[y_i \ln f_{\bar{\mathbf{w}}}(\mathbf{x}_i) + (1-y_i)\ln(1-f_{\bar{\mathbf{w}}}(\mathbf{x}_i))]$$

y al riesgo empírico a minimizar, (corregido para que su mínimo sea cero):

$$\hat{R}_n(\bar{\mathbf{w}}) = \frac{1}{n}\sum_{i=1}^{n}[y_i \ln \frac{y_i}{f_{\bar{\mathbf{w}}}(\mathbf{x}_i)} + (1-y_i)\ln\frac{1-y_i}{1-f_{\bar{\mathbf{w}}}(\mathbf{x}_i)}] \tag{6}$$

que es la versión empírica de la distancia de Kullback-Leibler ya vista con anterioridad.

c) El algoritmo de optimización más utilizado es la **regla delta** obtenida de la aplicación del algoritmo del gradiente, como en el caso de Adaline con activación dicotómica.

Redes Neuronales de Una Capa para el Problema de Regresión

1. La estimación de las probabilidades a posteriori es un enfoque al problema de clasificación basado en el método de regresión.

2. Salvo la red adaline con activación sigmoide, históricamente no se han desarrollado redes de una sóla capa para regresión con otras activaciones no lineales, probablemente, porque las redes que podían resultar de interés producían modelos paramétricos de sobra conocidos en estadística.

3. Por ejemplo, una neurona adaline con una función de activación identidad ψ produce el modelo lineal de regresión:

$$f(\mathbf{x}) = \psi(\bar{\mathbf{w}}^T\bar{\mathbf{x}}) = \bar{\mathbf{w}}^T\bar{\mathbf{x}}$$

y con una función de activación ψ polinómica produce un modelo polinómico:

$$f(\mathbf{x}) = \psi(\bar{\mathbf{w}}^T\bar{\mathbf{x}}) = (\bar{\mathbf{w}}^T\bar{\mathbf{x}})^2 = w_0^2 + 2w_0w_1x_1 + 2w_0w_2x_2 + 2w_1w_2x_1x_2 + w_1^2x_1^2 + w_2^2x_2^2$$

4. Una mayor flexibilidad se obtiene con una **transformación no lineal previa** de las variables de entrada, lo que requiere un nivel oculto adicional.

   Esta técnica era también conocida de la estadística, pero se utilizaba más bien para el pre-proceso de datos con objetivos muy específicos, (normalidad, homocedasticidad, incorrelación, etc.), pero no como una técnica para resolver problemas generales.

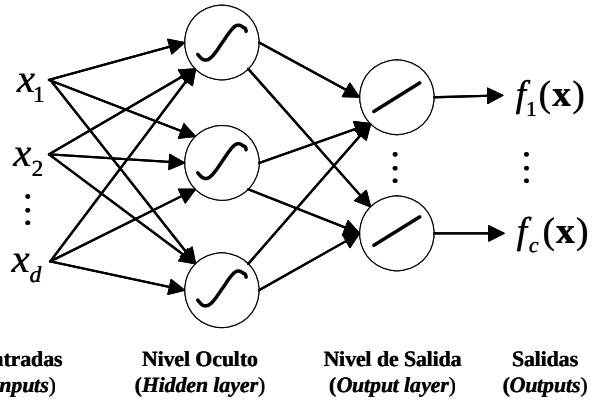

Figura 6: Estructura general de un modelo de red neuronal 'alimentada hacia adelante' (*feed-forward*).

Redes Neuronales Multi-layer Feed-forward (MLF)

1. Los modelos supervisados más utilizados son las **redes multilayer feed-forward** (MLF) de **una sóla capa oculta**, descritos al principio, que responden a la forma general:

$$\mathbf{f}(\mathbf{x};\boldsymbol{\theta}) = \begin{bmatrix} f(\mathbf{x};\boldsymbol{\theta}_1) \\ \vdots \\ f(\mathbf{x};\boldsymbol{\theta}_c) \end{bmatrix} = \begin{bmatrix} \varphi(\sum_{j=1}^{h} c_{1,j}\psi_{\mathbf{w}_j}(\mathbf{x}) + c_{1,0}) \\ \vdots \\ \varphi(\sum_{j=1}^{h} c_{c,j}\psi_{\mathbf{w}_j}(\mathbf{x}) + c_{c,0}) \end{bmatrix}$$

   donde:

   a) $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_c\}$, con $\boldsymbol{\theta}_k = \{\mathbf{W}, \mathbf{c}_k\}$, donde: $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_h)$ es el conjunto de **pesos del nivel oculto** y $\mathbf{c}_k = (c_{k,1}, ..., c_{k,h})^T$ es el vector de **pesos de la salida** k-ésima.

   b) $\{\psi_{\mathbf{w}_j}(\cdot) = \psi(\cdot;\mathbf{w}_j)\}$ es un conjunto de funciones de una familia $\mathcal{F}_\psi = \{\psi_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^r\}$ parametrizada mediante el vector de parámetros $\mathbf{w} \in \mathbb{R}^r$.

   c) φ es una función que es la misma en todas las componentes de $\mathbf{f}(\mathbf{x};\boldsymbol{\theta})$.

2. Como caso particular, una red MLF con una sóla salida (o cada componente de una red de varias salidas) puede escribirse como:

$$f(\mathbf{x};\boldsymbol{\theta}) = \varphi(\sum_{j=1}^{h} c_j\psi_j(\mathbf{x};\mathbf{w}_j) + c_0) = \varphi(\mathbf{c}^T\boldsymbol{\psi} + c_0) = \varphi(\bar{\mathbf{c}}^T\bar{\boldsymbol{\psi}})$$

   donde $\boldsymbol{\theta} = \{\mathbf{W}, \bar{\mathbf{c}}\}$ con $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_h\}$ y $\bar{\mathbf{c}} = (c_0, c_1, ..., c_h)^T$.

Funcionamiento de las Redes MLF

1. Por tanto, las redes MLF:

   a) Realizan en primer lugar, una **transformación de las variables de entrada** produciendo el nuevo vector de variables $\bar{\boldsymbol{\psi}}$:

$$\bar{\mathbf{x}} = (1, \mathbf{x})^T \rightarrow \bar{\boldsymbol{\psi}} = (1, \boldsymbol{\psi})^T$$

b) En segundo lugar, aplican a dichas variables transformadas la función $y = \varphi(\bar{\mathbf{c}}^T\bar{\boldsymbol{\psi}})$ (suponemos una sóla salida).

2. De esta forma, a dos puntos "similares"según las variables transformadas u ocultas, la red neuronal les asignará valores similares. Por tanto:

   a) La capa oculta tiene por objeto crear un espacio intermedio con un nuevo sistema de distancias entre puntos diferente a la del espacio de entrada, y lo más adecuado posible para permitir que

   b) La capa de salida consiga reproducir cómodamente (a veces con simples funciones lineales) la salida deseada.

Capa Oculta

1. Existen dos grandes familias de modelos según el **tipo de transformación de la capa oculta**, o lo que es lo mismo según la **medida de similitud o diferencia**, utilizada:

   a) **Modelos de Proyección**: proyectan los puntos sobre el hiperplano $\mathbf{w}_j^T\mathbf{x} + w_0$ definido por cada vector de pesos $\mathbf{w}_j$ antes de aplicar la función de activación, produciendo:

   $$\psi_{\bar{\mathbf{w}}_j}(\mathbf{x}) = \psi(\mathbf{w}_j^T\mathbf{x} + w_0)$$

      1) La variable $\psi_{\bar{\mathbf{w}}_j}(\mathbf{x})$ producirá valores iguales sobre puntos ubicados en esa dirección.
      2) Las funciones de transferencia resultantes $\psi_{\bar{\mathbf{w}}_j}(\mathbf{x})$ se denominan funciones *ridge* (protuberancia).
      3) Ejemplos: redes neuronales **multilayer perceptron** (MLP), modelos **projection pursuit** (PP) y redes **support vector machines** (SVM) con núcleo basado en proyecciones.

   b) **Modelos Radiales**: determinan las diferencias $\mathbf{x} - \mathbf{w}_j$ y, muy frecuentemente, las distancias $\|\mathbf{x} - \mathbf{w}_j\|$ con respecto a cada vector referencia $\mathbf{w}_j$, antes de aplicar la función de activación, produciendo:

   $$\psi_{\bar{\mathbf{w}}_j}(\mathbf{x}) = \psi_j\left(\frac{\mathbf{x} - \mathbf{w}_j}{w_0}\right) \text{ ó bien } \psi_{\bar{\mathbf{w}}_j}(\mathbf{x}) = \psi_j\left(\left\|\frac{\mathbf{x} - \mathbf{w}_j}{w_0}\right\|\right)$$

      1) La variable $\psi_{\bar{\mathbf{w}}_j}(\mathbf{x})$ producirá valores similares sobre puntos ubicados de manera similar con respecto a la referencia $\mathbf{w}_j$ (que actúa a modo de patrón o modelo).
      2) Las funciones de transferencia resultantes $\psi_{\bar{\mathbf{w}}_j}(\mathbf{x})$ se denominan **radiales** (en el segundo caso).
      3) Ejemplos: redes neuronales de **base radial** (RBF) y **support vector machines** (SVM) con núcleo radial.

2. Nótese que:

   $$\|\mathbf{x} - \mathbf{w}\|^2 = \langle(\mathbf{x} - \mathbf{w}), (\mathbf{x} - \mathbf{w})\rangle = \langle\mathbf{x}, \mathbf{x}\rangle + \langle\mathbf{w}, \mathbf{w}\rangle - 2\langle\mathbf{x}, \mathbf{w}\rangle$$

   a) Con lo que, si los vectores están normalizados, se obtiene: $\|\mathbf{x} - \mathbf{w}\|^2 = 2 - 2\langle\mathbf{x}, \mathbf{w}\rangle$.

   b) De esta forma, a mayor distancia, menor producto interno, y, así, puede pensarse en $\langle\cdot,\cdot\rangle$ como una **medida de similitud** y en $\|\cdot - \cdot\|^2$ como una **medida de disimilitud**.

3. Realmente, puede decirse que la diferencia más importante entre los diferentes tipos de funciones de transferencia es el carácter **global** de las de tipo proyección, y el carácter **local** de las de tipo radial.

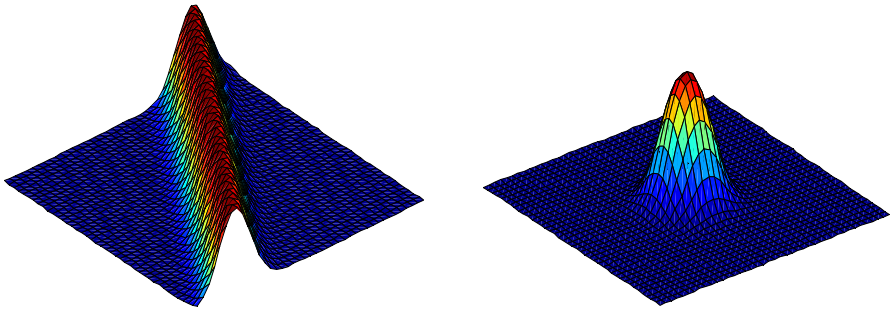

Figura 7: Izquierda: activación gausiana con proyección: $f(\mathbf{x}) = \exp(-0{,}1(2x_1 + 3x_2)^2)$. Derecha: activación logística aplicada a una transformación distancia: $f(\mathbf{x}) = -(1 + e^{-(x_1^2+x_2^2)})^{-1}$

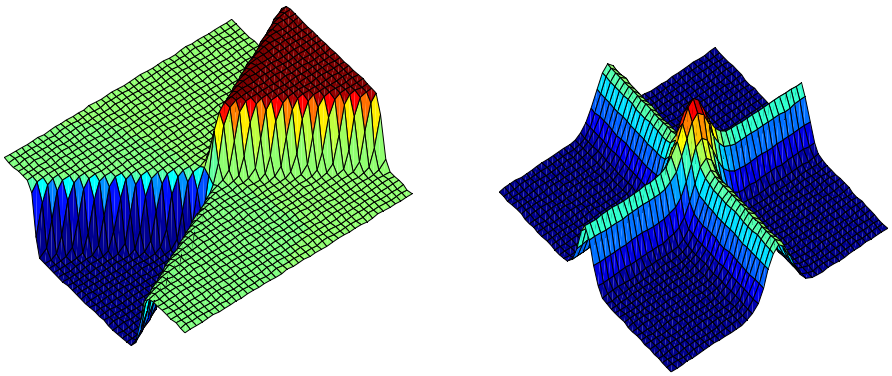

Figura 8: Izquierda: dos neuronas con activación logística con transformación proyección: $f(\mathbf{x}) = (1 + e^{-(2x_1+3x_2)})^{-1} + (1 + e^{-(x_1-2x_2)})^{-1}$. Derecha: dos neuronas con activación gausiana y transformación distancia: $f(\mathbf{x}) = e^{-x^2} + e^{-2y^2}$.

a) En este sentido, la transformación de proyección provoca que puntos muy alejados (globalidad) pero situados en un mismo hiperplano $\{\mathbf{w}^T\mathbf{x} = b\}$ posean respuestas muy similares (figura 7, izquierda).

b) Sin embargo, la suma de dos funciones de transferencia puede adquirir ya carácter global, como era de esperar, para poder aproximar cualquier función regular (figura 8).

4. Un criterio general perseguido para la elección de la función de transferencia es que aporte la suficiente complejidad a la red neuronal, para poder aproximar cualquier función del espacio de interés y así poder reducir a cero el error de aproximación tratado en el capítulo anterior.

5. Con respecto al primer grupo de arquitecturas, un requisito adicional es una regularidad suficiente para permitir la utilización de algoritmos de optimización no lineal en caso necesario (normalmente, que posean segunda derivada continua).

Nivel de Salida

1. El nivel de salida constituye realmente una red neuronal de una sóla capa (o una neurona si se trata de una sóla salida) cuya entrada es ahora la salida del nivel oculto.

2. Las funciones de activación de este nivel están determinadas fundamentalmente por el problema de aplicación:

   a) **Clasificación**. Si se trata de un problema de clasificación en el que se pretende un estimador *plug-in* de la regla de Bayes, del análisis de la sección anterior se deduce que para estimar las probabilidades a posteriori pueden utilizarse funciones de activación sigmoides o lineales, o softmax para el caso de varias clases, con función de pérdida cuadrática o de entropía cruzada. La segunda es la más natural pero la primera produce buenos resultados y es muy utilizada.

   b) **Regresión**. Si se trata de un problema de regresión, es necesario utilizar funciones de activación compatibles con el rango de los datos. En este sentido, las funciones sigmoides restringen la salida al intervalo $[0, 1]$ y, a pesar de que los datos podrían ser transformados previamente para ser compatibles con ello, suelen utilizarse activaciones lineales (la identidad), debido a su mayor facilidad de entrenamiento con pérdida cuadrática. Ello, como veremos, no penaliza la capacidad de la red pues su naturaleza no lineal se realiza previamente en el nivel oculto.

3. Por tanto, dado que el método *plug-in* en el problema de clasificación utiliza un enfoque de regresión, en lo que sigue, salvo que se especifique lo contrario, trataremos el problema de regresión como problema marco. Únicamente, haremos referencia explícita al problema de clasificación cuando una técnica específica lo requiera.

Red Neuronal Multi-layer Perceptron (MLP)

1. La red neuronal perceptrón multicapa (MLP), se caracteriza por utilizar la misma función de activación para todas las neuronas del nivel oculto.

2. Para el problema de regresión, la red MLP posee como forma general:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{j=1}^{h} c_j \psi(\mathbf{w}_j^T \mathbf{x} + w_{j0}) + c_0 \tag{7}$$

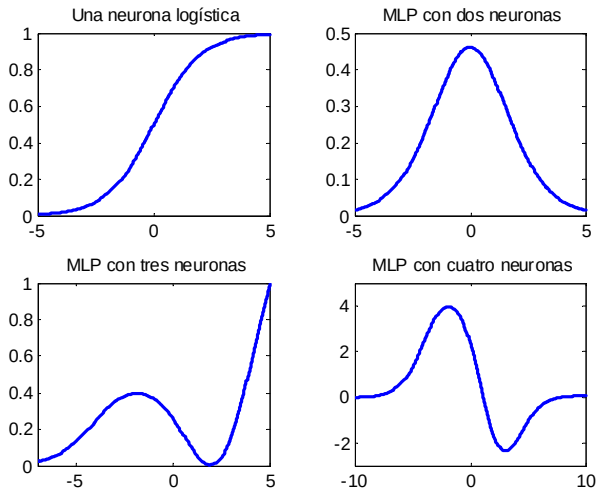

Figura 9: Redes MLP con diferente complejidad. De izquierda a derecha y de arriba a abajo: una neurona logística $(1+e^{-x})^{-1}$, y MLP con dos, tres y cuatro neuronas, respectivamente.

3. En el problema de clasificación, como ya se ha mencionado, si se desea que la red estime las probabilidades a posteriori para utilizar después la regla *plug-in*, las activaciones de salida pueden ser igualmente lineales o también sigmoides, en ambos casos bajo pérdida cuadrática o de entropía cruzada.

4. En el nivel oculto, suelen utilizarse funciones sigmoides, usualmente la **logística**, pero también transformaciones de ella como, por ejemplo, la función **tangente hiperbólica** $\psi^1(r)=\tanh(r)\in[-1,1]$, cuya relación con la logística $\psi^0(r)$ es:

$$\psi^1(r)=\frac{e^r-e^{-r}}{e^r+e^{-r}}=\frac{1-e^{-2r}}{1+e^{-2r}}=2\frac{1}{1+e^{-2r}}-1=2\psi^0(2r)-1$$

5. **Consistencia**. En general, un red MLP con suficientes neuronas en el nivel oculto posee la **propiedad de aproximación universal**, que le permite aproximar cualquier función continua, siempre que las funciones de activación del nivel oculto no sean polinomios [Leshno *et al.* 1993] (Figura 9).

   De esta forma, el error de aproximación puede hacerse tan pequeño como se quiera, con lo que sólo queda hacer tender a cero el error de estimación mediante una adecuada selección del modelo.

Algoritmos de Entrenamiento para la Red MLP

1. Los más utilizados son de dos tipos:

   a) Sin regularización (p. ej. [Bishop 1995], [Haykin 1999]):

      1) Minimizan el riesgo empírico:

$$\hat{R}_n(\boldsymbol{\theta})=\frac{1}{n}\sum_{i=1}^{n}\ell(y_i,f(\mathbf{x}_i;\boldsymbol{\theta}))$$

utilizando algoritmos de optimización no lineal (algoritmos basados en el gradiente o el Hessiano) adaptados (*Back-propagation*) para aprovechar la estructura de red *feed-forward* del modelo.

2) Posteriormente, es necesario realizar el control de la complejidad y la selección del modelo (por ejemplo, mediante validación cruzada).

b) Con regularización (Métodos Bayesianos), [Foresee y Hagan 1997], [Bishop 1995].

1) Minimizan el riesgo empírico regularizado (p. ej. con pérdida cuadrática):

$$\hat{\mathcal{R}}_n(\boldsymbol{\theta}) = \beta \hat{R}_n(\mathbf{y}^n|\boldsymbol{\theta}) + \lambda \mathcal{E}(\boldsymbol{\theta}) = \beta \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\boldsymbol{\theta}))^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

obteniéndose los óptimos para $\boldsymbol{\theta}$, β y λ.

2) De esta forma, realizan al mismo tiempo el control de la complejidad y la selección del modelo.

2. Los algoritmos de optimización basados en el gradiente o hessiano usualmente utilizados con las redes MLP poseen los siguientes inconvenientes:

a) Los problemas de optimización poseen múltiples óptimos locales en los que puede quedar atrapado el algoritmo de entrenamiento.

b) Si se repite el aprendizaje con los mismos datos, no tiene por qué obtenerse la misma red neuronal pues los algoritmos comienzan con valores iniciales aleatorios para el parámetro $\boldsymbol{\theta}$.

La solución reside en utilizar algoritmos que busquen óptimos globales (p. ej. algoritmos genéticos) pero suponen una mayor carga computacional.

Red Neuronal de Base Radial (RBF)

1. La red neuronal de base radial (RBF) se caracteriza, al igual que la MLP, por utilizar la misma función de activación para todas las neuronas del nivel oculto.

2. Para el problema de regresión, la red RBF más utilizada posee como forma general:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{j=1}^{h} c_j \psi \left(\|\mathbf{x} - \mathbf{w}_j\|_{\boldsymbol{\Sigma}_j} \right) + c_0 \tag{8}$$

donde $\|\mathbf{x} - \mathbf{w}_j\|_{\boldsymbol{\Sigma}_j} = (\mathbf{x} - \mathbf{w}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \mathbf{w}_j)$ para alguna matriz $\boldsymbol{\Sigma}_j$.

3. Sin embargo, ese modelo posee muchos parámetros en las matrices $\boldsymbol{\Sigma}_j$ y es muy difícil de entrenar. Por ello, es frecuente utilizar como alternativas:

a) $\boldsymbol{\Sigma}_j = \sigma_j^2 \mathbf{I}_d$, $j \in \{1 : h\}$, lo que significa que la función de transferencia es esférica con amplitud determinada por σ_j^2.

b) $\boldsymbol{\Sigma}_j = \Sigma = \sigma^2 \mathbf{I}_d$, es decir todos los parámetros σ_j son el mismo con valor σ.

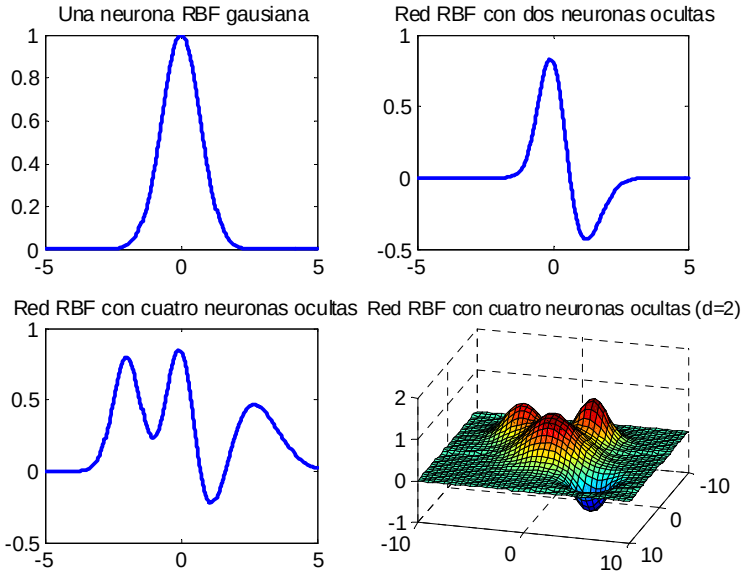

Figura 10: Redes RBF con diferente complejidad. De izquierda a derecha y de arriba a abajo: una neurona gausiana $\exp(-x^2)$, redes RBF con dos y cuatro neuronas, respectivamente, y red RBF con cuatro neuronas en $\mathbb{R}^2$.

4. Es frecuente utilizar como función de activación la gausiana:

$$\psi\left(\|\mathbf{x}-\mathbf{w}_j\|_{\mathbf{\Sigma}_j}\right) \propto \exp\left\{-\frac{1}{2}\|\mathbf{x}-\mathbf{w}_j\|^2_{\mathbf{\Sigma}_j}\right\} = \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{w}_j)^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}-\mathbf{w}_j)\right\}$$

   que, en el último caso anterior, resulta:

$$\psi\left(\|\mathbf{x}-\mathbf{w}_j\|_{\mathbf{\Sigma}_j}\right) \propto \exp\left\{-\frac{1}{2}\left\|\frac{\mathbf{x}-\mathbf{w}_j}{\sigma}\right\|^2\right\} = \psi\left(\left\|\frac{\mathbf{x}-\mathbf{w}_j}{\sigma}\right\|\right)$$

5. Los parámetros σ_j (o σ si son todos iguales) se denominan parámetros **amplitud, ventana** o **radio** de cada función de transferencia.

   Los parámetros $\mathbf{w}_j$ suelen denominarse **centros** de la red RBF y actúan a modo de modelos o puntos de referencia de los datos, en relación con el problema de que se trate (clasificación o regresión).

6. **Consistencia**. En general, un red RBF con suficientes neuronas en el nivel oculto posee la **propiedad de aproximación universal**, que le permite aproximar cualquier función continua, siempre que las funciones de transferencia del nivel oculto sean de la forma $\psi(\mathbf{x}-\mathbf{w}_i)$ y ψ sea integrable, acotada y continua (p. ej. [Park y Sandberg 1991]) (Figura 10).

   De esta forma, el error de aproximación puede hacerse tan pequeño como se quiera, con lo que sólo queda hacer tender a cero el error de estimación mediante una adecuada selección del modelo.

Algoritmos de Entrenamiento para la Red RBF

1. Los algoritmos de entrenamiento que pueden utilizarse con las redes RBF son:

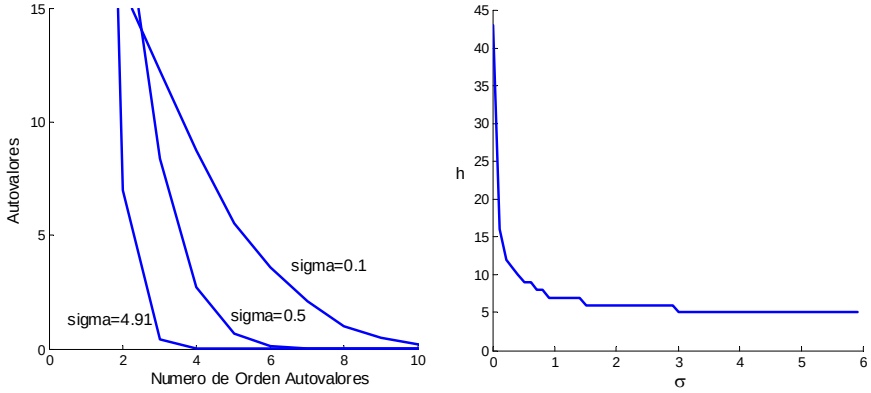

Figura 11: Izquierda: influencia de σ en el número de autovalores significativos de la matriz $\mathbf{\Psi}$, indicando el grado de redundancia de las variables ocultas, en cada caso. Derecha: número de funciones básicas no redundantes determinado por el valor de σ. Para un σ dado, la adición de nuevas variables a partir de un número h no aporta capacidad de representación (p. ej. si $\sigma = 5$, la utilización de más de 4 neuronas no aporta beneficios y produce un mal condicionamiento de la matriz $\mathbf{\Psi}$.

a) Los mismos algoritmos mencionados para las redes MLP, con los mismos inconvenientes.

b) Algoritmos lineales que, aunque resultan subóptimos, poseen mucha menor carga computacional.

2. **Algoritmo Lineal**. Suponemos un mismo parámetro ventana σ:

a) Se elige un conjunto Λ de posibles valores para σ.

b) Para cada valor $\sigma \in \Lambda$:

1) Realizar la siguiente transformación (capa oculta) de los datos $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$:

$$\begin{aligned} \boldsymbol{\psi}_i &= (\psi_1(\mathbf{x}_i), ..., \psi_n(\mathbf{x}_i))^T, i = 1 : n \\ \text{con} &: \psi_j(\mathbf{x}_i) = \psi\left(\left\|\frac{\mathbf{x}_i - \mathbf{x}_j}{\sigma}\right\|\right) \end{aligned}$$

(Nótese que todos los datos se utilizan inicialmente como centros).

2) Determinar el número de unidades ocultas necesarias con el radio σ, realizando un análisis espectral de la matriz $\mathbf{\Psi}$ tal que $\mathbf{\Psi}_{ij} = \psi_j(\mathbf{x}_i)$: se calculan los autovalores de la matriz y se determina el número h_σ de autovalores mayores que un umbral.
Ese número h_σ indica el número de variables ψ_j no redundantes (Figura 11).

3) Resolver el siguiente problema de regresión lineal (pérdida cuadrática):

$$\min_{\bar{\mathbf{c}}} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\psi}_i^T \bar{\mathbf{c}})\right)^2$$

mediante un algoritmo de mínimos cuadrados con selección de variables [Chen *et al.* 1991] que eligirá las h_σ variables ocultas más importantes para el problema. La solución es de la forma:

$$\begin{aligned} \hat{\bar{\mathbf{c}}}_\sigma &= (\mathbf{\Psi}^T\mathbf{\Psi})^{-1}\mathbf{\Psi}^T\mathbf{y} \\ &\Rightarrow \hat{f}_\sigma(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^T\hat{\bar{\mathbf{c}}}_\sigma = \boldsymbol{\psi}(\mathbf{x})^T(\mathbf{\Psi}^T\mathbf{\Psi})^{-1}\mathbf{\Psi}^T\mathbf{y} \end{aligned}$$

donde $\mathbf{\Psi}$ es la matriz cuyas columnas son las variables $\psi_0 = 1, \psi_1, ..., \psi_h$ valoradas en los n puntos $\mathbf{x}_i$, $i = 1 : n$.

c) Como resultado, se obtienen tantas redes RBF $\hat{f}_\sigma$ entrenadas, como valores de σ hay en Λ.
Ahora, basta realizar la selección del modelo mediante algún método de selección válido para modelos lineales (capítulo anterior) determinando la mejor red:

$$\hat{f}_{\sigma_*}(\mathbf{x}) = \sum_{j=1}^{h} c_j \psi\left(\frac{\mathbf{x} - \mathbf{w}_j}{\sigma_*}\right)$$

Bibliografía

Bishop C M [1995]. *Neural Networks and Pattern Recognition*, Oxford University Press.

Breiman L [1993]. Stacked Regression, Technical report, University of California, Berkeley.

Breiman L [1996]. Bagging Predictors, *Machine Learning* **24**, 123–140.

Candes E J [1998], Ridgelets: Theory and Applications, PhD thesis, Department of Statistics, Stanford University.

Chen S, Cowan C FÑ y Grant P M [1991]. Orthogonal Least Squares Learning for Radial Basis Function Networks, *IEEE Transactions on Neural Networks* **2**, 302–309.

Chen S S, Donoho D L y Saunders M A [1999]. Atomic Decomposition by Basis Pursuit, *SIAM Journal of Scientific Computing* **20**, 33–61.

Daubechies I [1992]. *Ten Lectures on Wavelets*, SIAM.

Devroye L, Györfi L y Lugosi G [1996]. *A Probabilistic Theory of Pattern Recognition*, Springer.

Duda R O, Hart P E y Stork D G [2001]. *Pattern Classification*, John Wiley.

Efromovich S [1999]. *Nonparametric Curve Estimation*, Springer.

Eubank R L [1988]. *Spline Smoothing and Nonparametric Regression*, Marcel Dekker.

Fan J y Gijbels I [1996]. *Local Polynomial Modelling and Its Applications*, Chapman and Hall.

Foresee F D y Hagan T [1997]. Gauss-Newton Approximation to Bayesian Regularization, *en Proceedings of the 1997 International Joint Conference on Neural Networks*, pp. 1930–1935.

Freund Y y Schapire R E [1997]. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting, *Journal of Computer and Systems Sciences* **55**, 119–139.

Friedman J H [1987]. Exploratory Projection Pursuit, *Journal of the American Statistical Association* **82**, 249–266.

Friedman J H y Stuetzle W [1981]. Projection Pursuit Regression, *Journal of the American Statistical Association* **76**, 817–823.

Green P J y Silverman B W [1994]. *Nonparametric Regression and Generalized Lineal Models*, Chapman and Hall.

Härdle W [1990]. *Applied Nonparametric Regression*, Cambridge University Press.

Hastie T J y Tibshirani R J [1990]. *Generalized Additive Models*, Chapman and Hall.

Haykin S [1999]. *Neural Networks. A Comprehensive Foundation*, Prentice Hall.

Johnson R A y Wichern D W [1992]. *Applied Multivariate Statistical Analysis*, Prentice-Hall.

Leshno M, Lin V Y, Pinkus A y Schocken S [1993]. Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function, *Neural Networks* **6**, 861–867.

Loader C [1999]. *Local Regression and Likelihood*, Springer.

Mallat S [1989]. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**, 674–693.

Mallat S y Zhang Z [1992]. Matching Pursuit in a Time-Frequency Dictionary, *IEEE Transactions on Information Theory* **38**, 617–643.

McCullagh P y Nelder J A [1983]. *Generalized Linear Models*, 1989 edn, Chapman and Hall.

Park J y Sandberg I W [1991]. Universal Approximation Using Radial-Basis-Function Networks, *Neural Computation* **3**, 246–257.

Ripley B D [1996]. *Pattern Recognition and Neural Networks*, Cambridge University Press.

Rosemblatt F [1958]. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review* **65**, 386–408.

Seber G A F [1977]. *Linear Regression Analysis*, John Wiley.

Seber G A F [1984]. *Multivariate Observations*, John Wiley.

Seber G A F y Wild C J [1989]. *Nonlinear Regression*, John Wiley.

Vapnik V [1998]. *Statistical Learning Theory*, John Wiley.

Vidakovic B [1999]. *Statistical Modeling by Wavelets*, John Wiley.

Wahba G [1990]. *Spline Models for Observational Data*, SIAM.

Wand M P y Jones M C [1995]. *Kernel Smoothing*, Chapman and Hall.

Widrow B y Hoff M E [1988]. Adaptive Switching Circuits, *en* J A Anderson y E Rosenfeld, eds, *Neurocomputing: Foundations of Research*, MIT Press.

Wolpert D H [1992]. Stacked Generalization, *Neural Networks* **5**, 241–259.