Introducción a las redes neuronales: Red neuronal de alimentación progresiva

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Computación, bachillerato en Ingeniería en Computación,
PAttern Recongition and MAchine Learning Group (PARMA-Group)

27 de marzo de 2019

Resumen

Este material está basado en el capítulo de redes neuronales, del libro *Pattern recognition and Machine Learning* de Cristopher Bishop[1].

1 Introducción

Ya hemos discutido sobre el uso de modelos lineales compuestos por combinaciones lineales de funciones base fijas. Un enfoque alternativo es utilizar un número definido de funciones básicas con parámetros personalizables, permitiendo a tales funcionales cambiar tales parámetros durante el entrenamiento. El modelo más exitoso para este tipo de contexto es la red de alimentación hacia adelante o red *feed-forward* o más comúnmente conocida como el perceptrón multi-capa.

El término de red neuronal tiene sus orígenes en los intintos de encontrar representaciones matemáticas para el procesamiento de información en los sistemas biológicos (McCulloch and Pitts, 1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart et al., 1986)

1.1 Funciones de alimentación hacia adelante o alimentación progresiva

Para los modelos de regresión y clasificación vistos anteriormente, ya se analizó que están constituidos por combinaciones lineales de funciones base no lineales *fijas*, es decir, sin parámetros que alteren su forma, dadas por ϕ_j, con $\vec{x} \in \mathbb{R}^D$ el vector de características a clasificar, $\vec{w}$ el vector de pesos para la combinación lineal de las funciones base ϕ_j. Para plantear con una sola expresión

el problema de la regresión y clasificación podemos escribir:

$$y\left(\vec{x},\vec{w}\right)=f\left(\sum_{j=0}^{M}w_j\phi_j\left(\vec{x}\right)\right) \tag{1}$$

donde en el caso de la regresión $f\left(u\right)$ corresponde a la función identidad $f\left(u\right)=u$ y para el caso de la clasificación $f\left(u\right)$ es una función no lineal, por ejemplo la función escalón $f\left(u\right)=\begin{cases}1 & u>0\\ 0 & u\leq 0\end{cases}$. En general, a la función f se le denomina **función de activación**. Recordemos que tanto en el problema de la regresión como la clasificación, es necesario **estimar el arreglo de pesos** $\vec{w}\in\mathbb{R}^M$.

Una extensión que incrementa la flexibilidad del modelo es hacer que las funciones base ϕ_j presenten parámetros que les permitan ser no fijas o adaptables, por lo que las funciones bases vendrían dadas por $\phi_j\left(\vec{x},\vec{\theta}\right)$ con $\vec{\theta}\in\mathbb{R}^D$ **el arreglo de parámetros ajustables** en entrenamiento, lo que hace necesario **encontrar los arreglos** $\vec{\theta}$ **y** $\vec{w}$ **durante el entrenamiento**.

Existen muchas formas the construir esas funciones paramétricas $\phi_j\left(\vec{x},\vec{\theta}\right)$. Las redes neuronales usan como funciones base, funciones de forma similares a lo planteado en la función 1, por lo que una función base ϕ_j tiene la forma:

$$\phi_j\left(\vec{x},\vec{\theta}\right)=h\left(\sum_{i=0}^{D}\theta_i x_i\right),$$

por lo que:

$$y\left(\vec{x},\vec{w}\right)=f\left(\sum_{j=0}^{M}w_j\,h\left(\sum_{i=0}^{D}\theta_i x_i\right)\right) \tag{2}$$

donde $h\left(u\right)$ **es una función no lineal y su entrada** u **viene dada por una combinación lineal** de las entradas con los pesos $\vec{\theta}$. Veamos más detalladamente como están dadas tales funciones base, ilustrandolo con un grafo. Observe de izquierda a derecha la Figura 1.

La red es escencialmente un grafo, usualmente de tres capas:

- La capa de D nodos de entrada, constituidos por los valores $x_1,x_2,\ldots,x_D$ del arreglo de entrada $\vec{x}\in\mathbb{R}^D$,
- La capa de M nodos $y_0^o,y_1^o,y_2^o,\ldots,y_M^o$, en notación vectorial dada por $\overrightarrow{y^o}\in\mathbb{R}^M$, llamada capa oculta, la cual controla el nivel de generalización de la red (suavidad de la superficie de decisión).
- La capa de salida, constituida por K nodos $y_1^s,\ldots,y_K^s$, con $\overrightarrow{y^s}\in\mathbb{R}^K$, lo cual corresponde al **número** K **de clases por discriminar**, usando una notación de la salida de $1-K$ o ***one hot vector***.

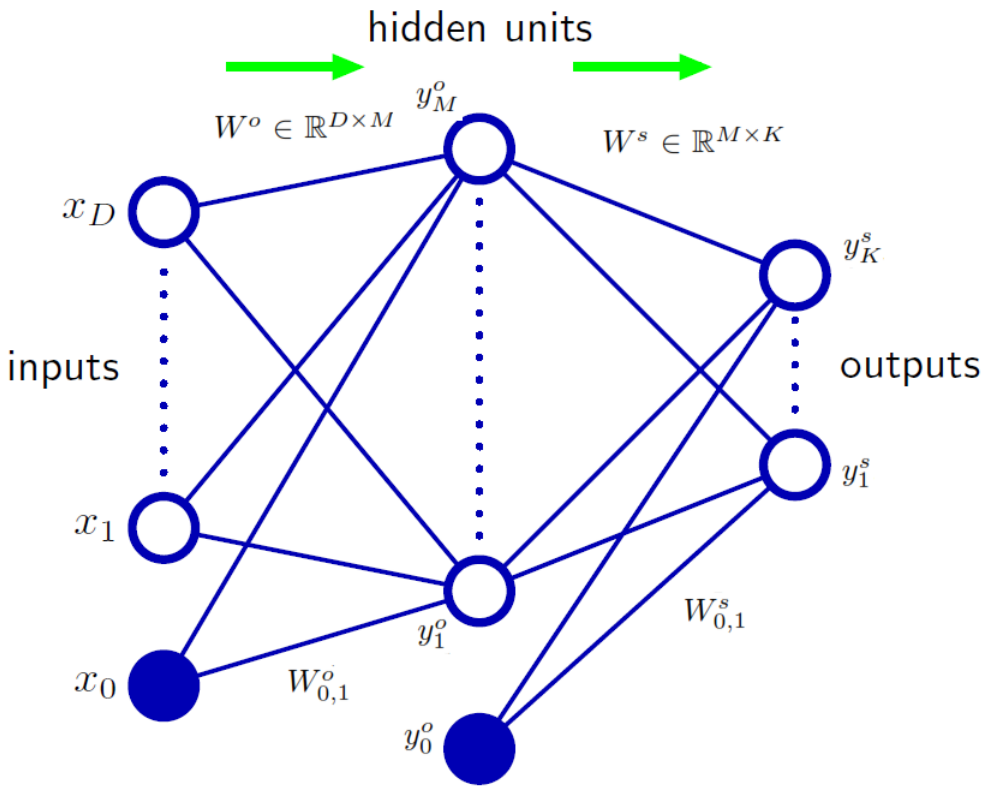

Figura 1: Diagrama de una red neuronal de dos capas (usualmente la capa de entrada no se incluye), tomado de [1].

1.1.1 La capa oculta

El grafo está definido por dos matrices de pesos $W^o \in \mathbb{R}^{D\times M}$ y $W^s \in \mathbb{R}^{M\times K}$ (se usará una notación en mayúscula para implicar el uso de una matriz, aunque también se puede suponer que W es un vector con dos subindices), La primer matriz de pesos conecta a la capa de entrada con la capa oculta, y la segunda matriz conecta la capa oculta con la capa de salida.

Para cada nodo j en la capa oculta, se define el **peso neto o coeficiente de activación** p_m^o el cual está dado por la combinación lineal de los valores en los nodos de entrada:

$$p_m^o\left(\vec{x}, W^o\right) = \sum_{d=1}^{D} W_{d,m}^o x_d + W_{0,m}^o, \tag{3}$$

donde el peso $W_{0,m}^o$ comúnmente se refiere como **sesgo**, por lo que para expresar al peso neto de la capa oculta p_m^o como únicamente una combinación lineal sin sesgo o desplazamiento, se reescribe:

$$p_m^o\left(\vec{x}, W^o\right) = \sum_{d=0}^{D} W_{d,m}^o x_d$$

se fija a $x_0 = 1$. El peso neto p_m de la unidad o neurona m es transformado usando una **función de activación** no lineal y diferenciable en la capa oculta $g^o(\cdot)$ para resultar en:

$$y_m^o\left(\vec{x}, W^o\right) = g^o\left(p_m^o\left(\vec{x}, W^o\right)\right), \tag{4}$$

con lo cual se puede observar que la función y_m^o corresponde a la función base $\phi_m\left(\vec{x}, \vec{\theta}\right)$,

$$\phi_m\left(\vec{x}, \vec{\theta}\right) = h\left(\sum_{j=0}^{D} \theta_j x_j\right),$$

con los parámetros de la función base dados por $\vec{\theta} = W^o$, y con $h = g^o$ y . El valor de los nodos $y_0^o, y_1^o, y_2^o, \ldots, y_M^o$ se les llama las **salidas de la capa oculta**.

La **función de activación** $h = g^o$ usualmente se escoge para que sea *suave* (**derivable**), y denote el estado de una neurona como activada o desactivada antes una entrada específica. Como veremos, la condición de que la función de activación sea derivable, es importante para poder calcular el gradiente y facilitar la minimización del error de clasificación. Las funciones de activación comunmente utilizadas son la tangente hiperbólica o la función sigmoidal, respectivamente dadas por:

$$\begin{array}{cc} \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} & \text{sigmoid}(x) = \frac{1}{1+e^{-x}} \\ \frac{\mathrm{d}}{\mathrm{d}x}\tanh(x) = 1 - \tanh^2(x) & \frac{\mathrm{d}}{\mathrm{d}x}\text{sigmoid}(x) = \frac{e^{-x}}{(1+e^{-x})^2} \end{array}$$

Sus gráficas se muestran en la Figura 2.

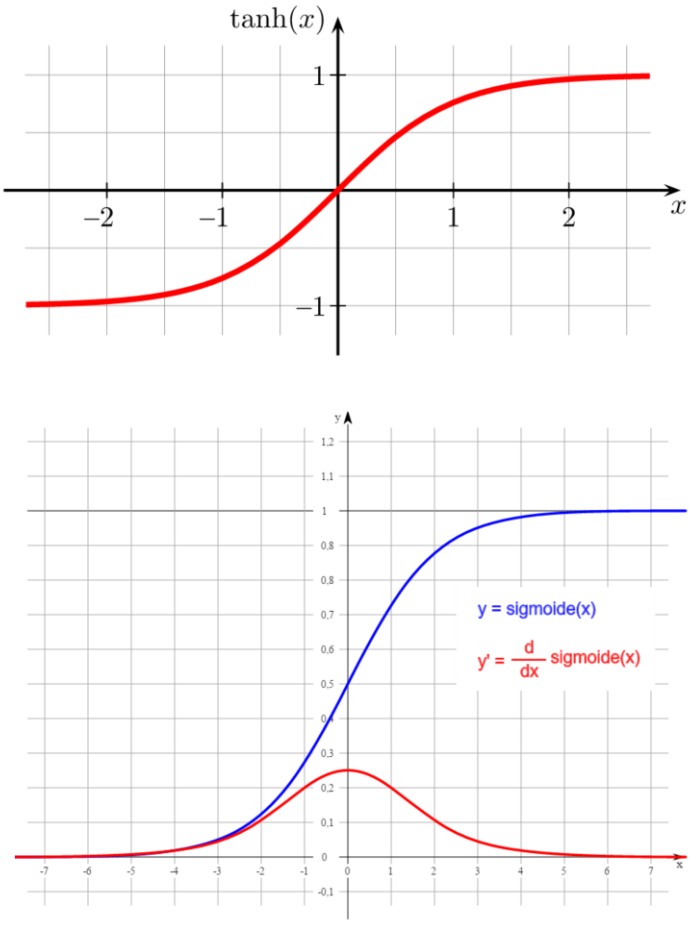

Figura 2: Funciones de activación, tangente hiperbólico y sigmoidal, de izquierda a derecha.

Dado el uso extensivo que haremos de la función *sigmoidal*, expresaremos su derivada de forma más compacta:

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sigmoid}\,(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})}\frac{e^{-x}}{(1+e^{-x})} \tag{5}$$

donde tomando el término derecho de tal multiplicación:

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

$$\Rightarrow \frac{1+e^{-x}}{(1+e^{-x})} - \frac{1}{(1+e^{-x})} = \left(1 - \frac{1}{(1+e^{-x})}\right),$$

por lo que entonces la ecuación 5 se puede reescribir como:

$$\frac{\mathrm{d}}{\mathrm{d}x}\text{sigmoid}\,(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})}\left(1 - \frac{1}{(1+e^{-x})}\right)$$

lo cual significa que:

$$\frac{\mathrm{d}}{\mathrm{d}x}\text{sigmoid}\,(x) = \text{sigmoid}\,(x)\,(1 - \text{sigmoid}\,(x)) \tag{6}$$

1.1.2 La capa de salida

Siguiendo el grafo de la red neuronal en la Figura 1, se define el peso neto para la unidad de salida k como:

$$p_k^s\,(\vec{x}, W^s) = \sum_{m=0}^{M} W_{m,k}^s y_m^o, \tag{7}$$

para cada unidad $k = 1, \ldots, K$, donde de manera similar para la capa anterior, $y_0^o = 1$. Observe que el peso neto de una unidad o neurona en la capa de salida está dado por la **combinación lineal de las salidas en las unidades ocultas**, usando los pesos definidos entre la capa oculta y de salida. La **salida de cada unidad de la capa de salida**, está dada por:

$$y_k^s\,(\vec{x}, W^s) = g^s\,(p_k^s\,(\vec{x}, W^s))$$

donde la función de activación $g^s\,(u)$ para las unidades de salida se elige usualmente según los siguientes casos:

1.1.3 Funciones de activación

- Para la **regresión** (clasificación en dominio continuo) se escoge la **función identidad**, de modo que $g^s\,(u) = u$ o **lineal**.
- Para la clasificación binaria o de dos clases donde entonces $K = 2$, se utiliza una función de activación de **tangente hiperbólico o sigmoidal.** Esto pues la red neuronal utiliza una codificación de la salida $1 - K$, de modo que si por ejemplo la red quiere dar a entender que una entrada $\vec{x}_a$ corresponde a la clase 1, entonces las unidades de salida deben representar lo anterior con $y_1 \approx 1$ y $y_2 \approx 0$, y en caso de corresponder a la clase 2, $y_1 \approx 0$ y $y_2 \approx 1$.
- Para la clasificación en múltiples clases $K > 2$ usualmente se utiliza la función *softmax o* la de tangente hiperbólico anteriormente vista. La función *softmax* es utilizada en el algoritmo de regresión logística *y* asocia sus

entradas a un valor de salida $0 \leq y_k^s \leq 1$, contribuyendo a la noción probabilística de que la entrada se de. Más formalmente, la función softmax denota la probabilidad de dado los valores de la muestra $\vec{x}$, la misma sea de la clase C_k, softmax $\left(\vec{p^s}, p_k^s\right) = p\left(C_k|\vec{p^s}\right)$ y está dada por:

$$\text{softmax}\left(\vec{p^s}, p_k^s\right) = \frac{e^{p_k^s}}{\sum_j e^{p_j^s}}.$$

La función *softmax* no es más que la normalización *respecto a las demás salidas de las unidades en la capa anterior, para* $K > 2$ clasesde la función *sigmoidal.* La función de activación sigmoidal sólo toma en cuenta el peso neto de su unidad, y no de las demás unidades en la misma capa:

$$\textit{sigmoid}\left(p_k^s\right) = \frac{1}{1 + e^{-p_k^s}}$$

A continuación el siguiente código:

```
a = [3 2 1];
ySoftmax(1) = exp(a(1))/ sum(exp(a));
ySoftmax(2) = exp(a(2))/ sum(exp(a));
ySoftmax(3) = exp(a(3))/ sum(exp(a));
%Salida con la funcion sigmoidal
ySigmoid(1) = 1 / (1 + exp(-a(1)))
ySigmoid(2) = 1 / (1 + exp(-a(2)))
ySigmoid(3) = 1 / (1 + exp(-a(3)))
```

muestra el ejemplo en el que el arreglo de coeficientes de activación está dado por $\vec{p^s} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$, donde en este caso el compontente p_1^s es el mayor y debe asignársele la mayor probabilidad, por lo que al hacer que $y_k\left(\vec{p^s}, p_k^s\right) = \text{softmax}\left(\vec{p^s}, p_k^s\right)$ para todos los valores p_k^s resulta en

$$\vec{y}_{\text{softmax}} = \begin{bmatrix} 0{,}6652 \\ 0{,}2447 \\ 00{,}09 \end{bmatrix} \qquad \vec{y}_{\text{sigmoid}} = \begin{bmatrix} 0{,}9526 \\ 0{,}8808 \\ 00{,}7311 \end{bmatrix}$$

, donde se observa como la salida de la función *softmax* da un valor mucho más bajo al tercer componente cuando se compara con la función *sigmoidal*.

Retomando el funcional para las unidades de salida $y_k^s = g^s\left(p_k^s\right)$, reemplazando el funcional p_k^s por su definición en la Ecuación 7 $p_k^s\left(\vec{x}, W^s\right) = \sum_{m=0}^{M} W_{m,k}^s y_m^o$, se tiene que:

$$y_k^s\left(\vec{x}, W^s\right) = g^s\left(\sum_{m=0}^{M} W_{m,k}^s y_m^o\right) \tag{8}$$

donde a su vez, y_m^o corresponde al funcional de activación definido en la ecuación 4 como $y_m\left(\vec{x}, W^o\right) = g^o\left(p_m^o\left(\vec{x}, W^o\right)\right)$, por lo que reemplazando en la ecuación 8 junto con la definición de p_m^o en la ecuación ; $p_m^o\left(\vec{x}, W^o\right) = \sum_{d=0}^{D} W_{d,m}^o x_d$, se tiene que:

$$y_k\left(\vec{x}, W^o, W^s\right) = g^s\left(\sum_{m=0}^{M} W_{m,k}^s\, g^o\left(\sum_{d=0}^{D} W_{d,m}^o x_d\right)\right) = g^s\left(\sum_{m=1}^{M} W_{m,k}^s\, g^o\left(\sum_{d=1}^{D} W_{d,m}^o x_d + W_{0,m}^o\right) + W_{0,k}^s\right) \tag{9}$$

lo cual es análogo al modelo no lineal estudiado al inicio:

$$y\left(\vec{x}, \vec{w}\right) = f\left(\sum_{j=0}^{M} w_j\, h\left(\sum_{k=0}^{D} \theta_k x_k\right)\right) \tag{10}$$

donde entonces reemplazamos $\vec{w} = W^s$, $f = g^s$ y la función base está dada por $\phi_j\left(\vec{x}, \vec{\theta}\right) = g^o\left(\sum_{d=1}^{D} W_{d,m}^o x_d\right)$, con el vector de parámetros de la función base dado por $\vec{\theta} = W_{d,m}^o$. Observe que la no linealidad de la función base está dada por la función de activación g^o (usualmente definida como una tangente hiperbólica o una función sigmoidal), y los parámetros de la función base no lineal están dadas por la matriz de parámetros $\vec{\theta} = W_{d,m}^o$.

El proceso de evaluar la ecuación 9 se conoce como la **propagación hacia adelante.** Para realizar la propagación hacia adelante, la red no debe tener ciclos, aunque puede incluir conexiones entre capas no consecutivas. Una cantidad menor de neuronas en la capa oculta respecto a la cantidad de neuronas de entrada implica una reducción de dimensionalidad, disminuyendo el riesgo de sobre-ajuste. Como se mencionó, las redes neuronales son conocidas también como **perceptrones multicapa,** con la diferencia principal de que las redes neuronales usan funciones de activación sigmoidales no lineales, mientras que el perceptrón usa una función de escalón o *Heaviside,* lo que permite que al conectar múltiples perceptrones en se serie, la función de salida de la red sea **diferenciable**, facilitando el proceso de entrenamiento.

Las redes neuronales se dice que son ***aproximadores universales*** en cuanto son capaces de realizar la regresión a partir de un conjunto de puntos dados y aproximar un funcional, usando como función de activación en la capa de salida una función lineal. La Figura 3 muestra la aproximación de cuatro funciones $f\left(x\right) = x^2$, $f\left(x\right) = \sin\left(x\right)$, $f\left(x\right) = |x|$ y $f\left(x\right) = H\left(x\right)$ correspondiente a la función de Heaviside, con $N = 50$ puntos de entrenamiento. La Figura 3 muestra además las salida de 3 neuronas ocultas, ilustando como la combinación de estas resulta en la función de salida aproximada.

1.2 Entrenamiento de la red

Se seguirá con la misma terminología planteada para los problemas de regresión lineal y clasificación en dos categorías analizados anteriormente. En el

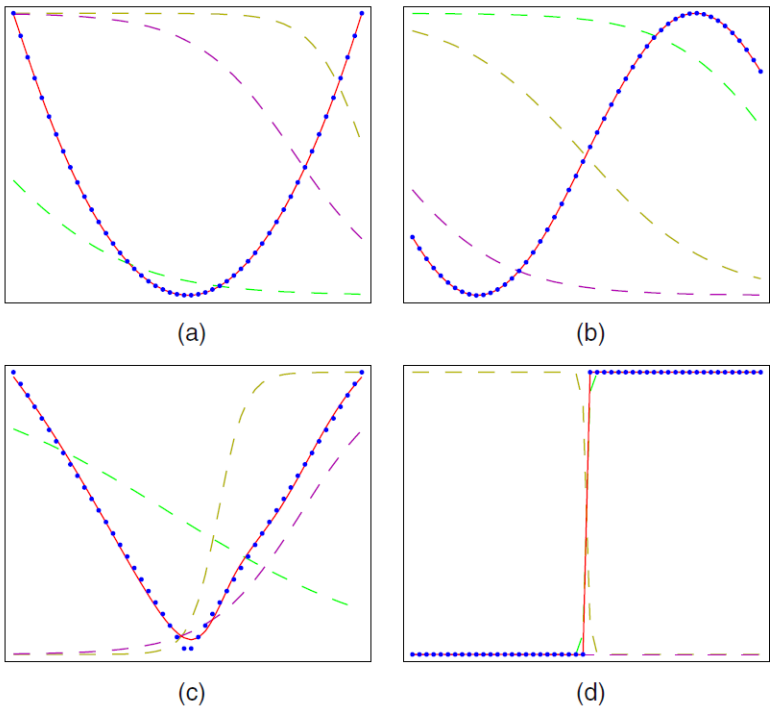

Figura 3: Aproximación de las funciones (a) $f(x) = x^2$ (b) $f(x) = \sin(x)$ (c) $f(x) = |x|$ y (d) la función escalón. En líneas punteadas las salidas de las neuronas ocultas, tomado de , tomado de [1].

problema de clasificación, se define entonces una matriz de muestras de entrada

$$X = \begin{bmatrix} | & | & | & | \\ \vec{x}_1 & \vec{x}_2 & \dots & \vec{x}_N \\ | & | & | & | \end{bmatrix}$$

(lo que corresponde a una manera de representar al conjunto de N vectores de entrenamiento $\{\vec{x}_1, \dots, \vec{x}_N\}$) y el conjunto correspondiente de vectores de etiquetas (conocido de antemano), representado en la matriz

$$T = \begin{bmatrix} | & | & | & | \\ \vec{t}_1 & \vec{t}_2 & \dots & \vec{t}_N \\ | & | & | & | \end{bmatrix}.$$

La notación general para un modelo de clasificación está dada por $y(\vec{x}, \vec{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\vec{x})\right)$, donde en el caso de la red neuronal de dos capas, los pesos a determinar están dados por dos matrices W^o y W^s y la función de activación $f = g^s$ es una función no lineal. Tal cual se procedió anteriormente para los clasificadore lineales, el objetivo para construir un clasificador es **encontrar el vector de pesos $\vec{w}$ que minimice la función de error** cuadrático (lo que corresponde a maximizar la función de verosimilitud como se concluyó anteriormente):

$$E(\vec{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\| y(\vec{x}_n, \vec{w}) - \vec{t}_n \right\|^2,$$

donde se observa que efectivamente la función $E(\vec{w})$ tiene su dominio en $\mathbb{R}^2$ y su codominio en $\mathbb{R}$, una función multivariable para la cual podemos calcular fácilmente sus derivadas parciales. Para ilustrar el problema de encontrar el $\vec{w}_{\text{opt}}$ que minimice a la función de error $E(\vec{w})$, imaginemos el caso en el que el vector de pesos $\vec{w} \in \mathbb{R}^2$, lo que nos permite dibujar una superficie correspondiente a la función multivariable $E(\vec{w}) = E(w_1, w_2)$ como la gráfica de la superficie en la Figura 4 muestra, con tres vectores específicos en la superficie $\vec{w}_A$, $\vec{w}_B$ y $\vec{w}_C$ y con el vector gradiente ∇E evaluado en el punto $\vec{w}_C$.

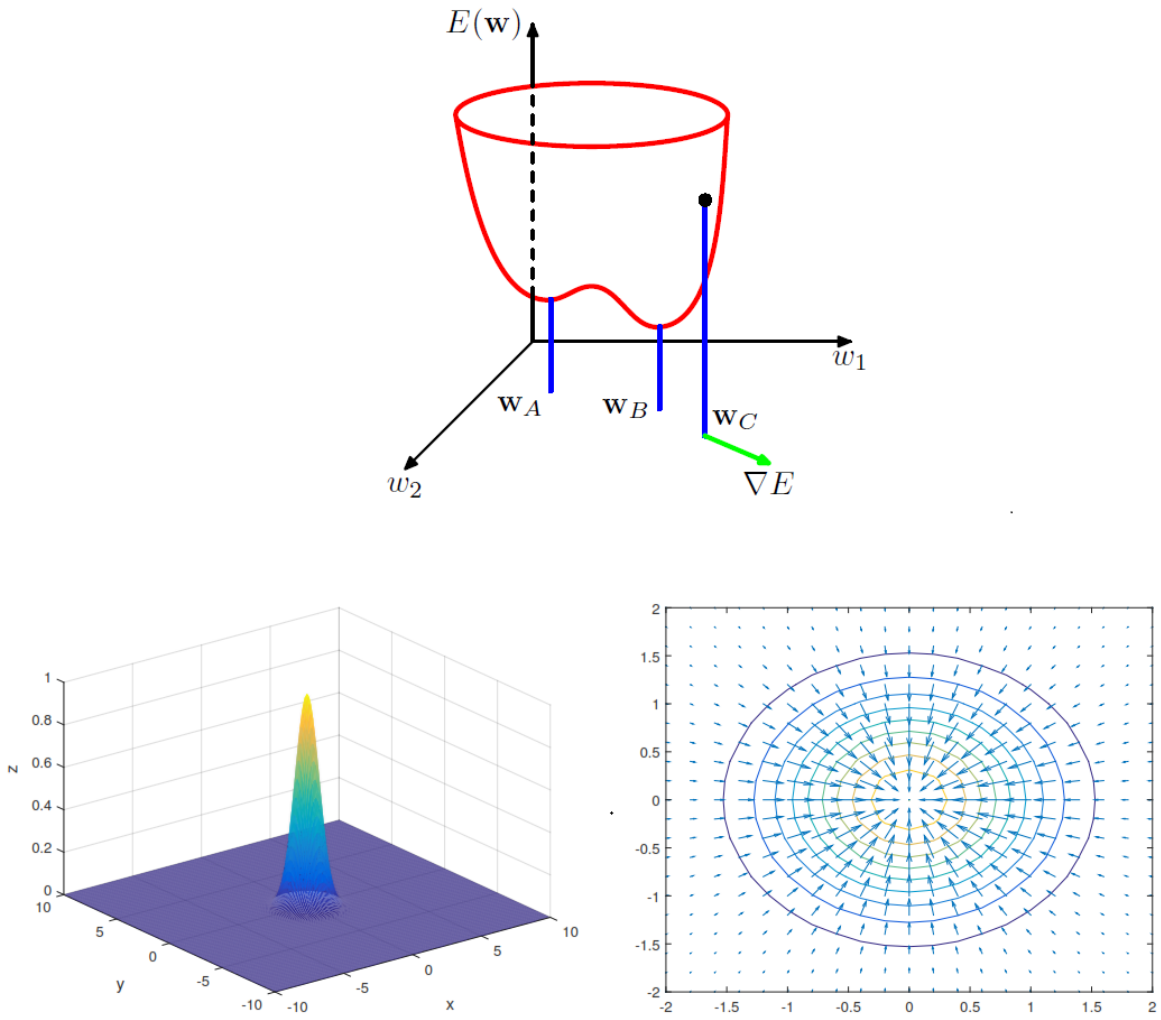

Figura 4: Superficie de la función de error $E(\vec{w}) = E(w_1, w_2)$, tomado de , tomado de [1] .

Siguiendo con la Figura 4, observe que si por ejemplo la búsqueda de un mínimo en la función $E(\vec{w})$ inicia por ejemplo en el vector $\vec{w}_C$, tal proceso de búsqueda debería, en una secuencia finita de iteraciones, encontrar un mínimo de la función de error, lo que podría coincidir con vector como $\vec{w}_B$. Observe

que un pequeño cambio en el vector $\vec{w}$ por otro *vector de cambio muy pequeño* $\vec{w}_\delta$ (observe que δ no es un escalar, sino una nomenclatura para el cambio) creando un vector nuevo $\vec{w}_{\text{nuevo}} = \vec{w} + \vec{w}_\delta$, el cual genera un cambio en la función de error E_δ que correspondiente a:

$$E_\delta \simeq (\vec{w}_\delta)^T \nabla E(\vec{w}) = (\vec{w}_\delta) \cdot \nabla E(\vec{w}) = \sum_{i=0} (\vec{w}_\delta)_i \nabla E(\vec{w})_i \tag{11}$$

equivalente al producto punto entre los vectores $\delta\vec{w}$ y $\nabla E(\vec{w})$, donde el último corresponde al vector gradiente, el cual apunta hacia la dirección de mayor crecimiento de la superficie, como también se ilustró anteriormente en la parte de abajo de la Figura 4, para el caso de una función Gaussiana.

Para ilustrar mejor lo anterior, tómese un ejemplo en el que el vector gradiente $\nabla E(\vec{w})$ sea perpendicular al vector de cambio de los pesos $\vec{w}_\delta$. En este caso el producto punto expresado en la ecuacion 11 resultaría en cero, por lo que entonces $E_\delta = 0$. Lo anterior resulta intuitivo, pues la función en esa dirección con $\vec{w}_\delta \to 0$ haría que el valor del funcional sea aproximadamente el mismo, $E(\vec{w}) \cong E(\vec{w}_\delta)$ (Imagine el caso en el que la función $E(\vec{w})$ correspondiera a un plano, en el caso comentado de perpendicularidad entre el vector gradiente y el desplazamiento de los pesos, el último correspondería a una **curva de nivel de tal funcional**).

La Figura 4 también ilustra el hecho de que el mínimo en una función multivariable como el caso de la función $E(\vec{w})$ ocurre cuando el gradiente tiende a cero:

$$\nabla E\left(\vec{w}_{\text{opt}}\right) = 0,$$

como se observa en los gradientes ilustrados para la superficie Gaussiana en las gráficas de la parte inferior en tal figura, o en el caso de la ilustración de la superficie $E(\vec{w})$ para el vector $\vec{w}_B$. Lo anterior significa que el objetivo de la búsqueda es encontrar un vector gradiente con magnitud lo suficientemente pequeña, determinada por ϵ:

$$\|\nabla E(\vec{w})\| < \epsilon.$$

Los puntos donde la magnitud del gradiente tiende a *desvanacerse* se conocen como **puntos estacionarios** los cuales pueden corresponder tanto a máximos como a mínimos locales de la función. Un máximo o mínimo global es un punto estacionario absolutamente máximo o mínimo en toda la función. Como se observa también en la Figura 4, el **seguir la dirección contraria al gradiente** $-\nabla E(\vec{w})$ **nos acerca más a un mínimo local**.

Dado que no es posible encontrar una solución analítica a la ecuación

$$\nabla E(\vec{w}) = 0,$$

es necesario utilizar métodos numéricos para ello. La optimización de funciones continuas no lineales es un problema muy estudiado y existen muchas técnicas para lograr tal objetivo. Muchas de las técnicas involucran la elección de

un valor inicial para el vector de pesos $\vec{w}_0 \in \mathbb{R}^M$, y con una serie de iteraciones cambiar tal vector *moviéndose* por el espacio $\mathbb{R}^M$, lo cual se expresa como:

$$\vec{w}(\tau+1) = \vec{w}(\tau) + \alpha \triangle \vec{w}(\tau), \tag{12}$$

donde τ etiqueta la iteración a la que corresponde el vector de pesos (como un funcional vectorial del tiempo), el vector $\triangle \vec{w}(\tau)$ es el vector de *actualización* y $\alpha \in \mathbb{R}$ es un coeficiente que determina la *velocidad* de la actualización. Existen diferentes enfoques para seleccionar el vector de actualización $\triangle \vec{w}(\tau)$, muchos se basan en la evaluación del gradiente $\nabla E(\vec{w}(\tau))$ para su definición. Para más información sobre el algoritmo de retropropagación y sus orígenes, puede ver `http://people.idsia.ch/~juergen/who-invented-backpropagation.html`.

1.3 Optimización de los pesos por descenso de gradiente y retropropagación

El enfoque más sencillo para escoger al vector de actualización $\triangle \vec{w}(\tau)$ es hacerlo igual al negativo del vector gradiente de modo que $\triangle \vec{w}(\tau) = -\nabla E(\vec{w}(\tau))$, por lo que entonces la ecuación 12 se reescribe como:

$$\vec{w}(\tau+1) = \vec{w}(\tau) - \alpha \nabla E(\vec{w}(\tau)).$$

Es común realizar varias corridas en las que el vector inicial $\vec{w}_0$ se fija con distintos valores aleatorios.

En las redes neuronales de múltiples capas, recordemos que tenemos al menos dos matrices de pesos a optimizar (W^o y W^s) por lo que al proceso de aplicar el descenso de gradiente desde la capa de salida hacia la capa de entrada se le conoce como **retro-propagación del error**. El proceso de entrenamiento de retropropagación con descenso de gradiente se puede dividir en las siguientes etapas:

- Propagación del error para calcular las derivadas parciales desde la capa de salida hacia la entrada (hacia atrás).
- Utilizar el resultado del gradiente evaluado desde la entrada para computar los ajustes a realizar en los pesos, lo que corresponde a la aplicación de la técnica de **descenso del gradiente**.

Cabe destacar que esta técnica de retro-propagación del error puede ser combinada con cualquier otro método para fijar el valor nuevo del gradiente, por lo que es importante distinguir las dos etapas.

Retomando entonces la ecuación del error definida como:

$$E(\vec{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\| y(\vec{x}_n, \vec{w}) - \vec{t}_n \right\|^2,$$

para facilitar su análisis la podemos reescribir como:

$$E(\vec{w}) = \sum_{n=1}^{N} E_n(\vec{w})$$

con

$$E_n(\vec{w}) = \frac{1}{2} \|y(\vec{x}_n, \vec{w}) - t_n\|^2$$

definido como el error de clasificación para una muestra $\vec{x}_n$ o **función de pérdida** (*loss function*). Con el mismo afán de simplificar el análisis, examinaremos posteriormente el aprendizaje de cada capa, desde la capa de salida a la oculta.

1.3.1 Funciones de pérdida

La función de pérdida define la superficie de error, y qué aspectos de la estimación pesan en la contribución de tal error. La salida del modelo para categorizar K clases viene dada por un arreglo:

$$\vec{y} = y(\vec{x}_n, \vec{w})$$

con dimensionalidad $\vec{y} \in \mathbb{R}^K$, utilizando una representación de la salida *1-K*. Así por ejemplo, para un modelo construído con el objetivo de discriminar imágenes en 3 categorías (por ejemplo la clasificación de imágenes de carros en categorías de Toyota, Tesla y Peugot), una etiqueta del conjunto de etiquetas T como por ejemplo $\vec{t}_a = [1\ 0\ 0]^T$, por lo que si nuestro modelo produce una salida $\vec{y}_a = y(\vec{x}_a, \vec{w}) = [0{,}5\ 0{,}4\ 0]^T$, el modelo debe modificar sus parámetros $\vec{w}$ para hacer que $\vec{y}$ se acerque lo más posible a $\vec{t}$. Pero **cómo se puede medir la cercanía** de $\vec{y}$ con $\vec{t}$? Las distancias $\ell = p$ son posibles opciones, sin embargo, es deseable utilizar un criterio que tome en cuenta además la naturaleza excluyente de la clasificación, es decir, si se asigna una probabilidad alta de pertenencia a la muestra $\vec{x}_a$ a una categoría específica c, el resto de categorías deben asignarseles probabilidades bajas de pertenencia.

Entropía Tómese el siguiente ejemplo. Sea un conjunto de muestras de imágenes de autos recolectadas en el campo. Imagine por ejemplo que la transmisión binaria de esas muestras tiene un costo de 100 colones por bit. Asignar la misma cantidad de bits para transmitir cada una de las muestras, haría que en este caso se transmitieran siempre 2 bits por muestra. Sin embargo, si disponemos de la probabilidad de cada categoría en las muestras obtenidas:

$$p(k) = \left[0{,}5\ \frac{0{,}5}{128}\ 0{,}4961\right]$$

es posible estimar la proporción de bits por cada clase a emplear usando el logaritmo en base 2:

$$b_{\text{toyota}} = \log_2\left(\frac{1}{p(1)}\right) = \log_2\left(\frac{1}{0{,}5}\right) = 1$$
$$b_{\text{tesla}} = \log_2\left(\frac{1}{\frac{0{,}5}{128}}\right) = 8$$
$$b_{\text{peugot}} = \log_2\left(\frac{1}{0{,}4961}\right) = 1$$

con lo cual para las muestras de la categoría Tesla se deben emplear 8 bits más que para las muestras de Toyota y Peugot, por lo que de forma práctica se asigna 1 bit para transmitir las muestras de Toyota y Peugot y 2 bits para las muestras de la categoría Tesla. De esta forma, la **cantidad esperada de bits** en de la **codificación óptima definida por la función de densidad** p, está dada por la entropía:

$$H(p) = \sum_i p(i) \log\left(\frac{1}{p(i)}\right) = -\sum_i p(i) \log\left(p(i)\right)$$

La entropía cruzada Haciendo la analogía de una **función de densidad** p **como una herramienta para definir la codificación de la información**, la entropía se puede entonces conceptualizar como la cantidad de bits necesaria para codificar la información, usando la herramienta óptima p **definida como la función de densidad real de los datos**.

La entropía cruzada mide la cantidad de bits esperada necesaria para codificar los datos con distribución real p**, usando la distribución aproximada o incorrecta** y**:**

$$H(p, y) = \sum_i p_i \log\left(\frac{1}{y_i}\right)$$

usando entonces $\log\left(\frac{1}{y_i}\right)$ símbolos por categoría**.** Esto hace que siempre la **entropía cruzada sea mayor a la entropía, por lo que a mayor entropía, mayor el error.** Reemplazando la función de densidad real p por la etiqueta real t, se obtiene la expresión de la evaluación de la entropía cruzada para una muestra j:

$$H\left(\overrightarrow{t}_j, \overrightarrow{y}_j\right) = \sum_i \overrightarrow{t}_j(i) \log\left(\frac{1}{\overrightarrow{y}_j(i)}\right)$$

Para utilizar la función de pérdida de entropía cruzada, es necesario emplear una función de activación que garantice el cumplimiento de las propiedades de una función de densidad en la función de activación de las neuronas a la salida, como por ejemplo, la función de activación *softmax*.

1.3.2 Aprendizaje en la capa de salida

El cambio en el vector de pesos de la capa de salida W^s, analizado por cada componente o entrada de la matriz:

$$W^s_{m,k}(\tau+1) = W^s_{m,k}(\tau) - \alpha \triangle W^s_{m,k}(\tau), \tag{13}$$

donde el vector de actualización según el método del descenso de gradiente está dado por:

$$\triangle W^s_{m,k}(\tau) = \frac{\mathrm{d}}{\mathrm{d}W^s_{m,k}} \left(\sum_{n=1}^{N} E_n \left(W^s_{m,k} \right) \right) \tag{14}$$

A partir de ahora, para facilitar la notación se tendrá que $W^s_{m,k}(\tau) = W^s_{m,k'}$ con:

$$E_n(W^s) = \frac{1}{2} \left\| \vec{y^s}(\vec{x}_n, W^s) - \vec{t}_n \right\|^2 = \left\| \begin{bmatrix} y_1 = g^s\left(\sum_{m=0}^{M} W^s_{m,0} y^o_m\right) \\ \vdots \\ y_K = g^s\left(\sum_{m=0}^{M} W^s_{m,K} y^o_m\right) \end{bmatrix}_n - \vec{t}_n \right\|^2,$$

lo cual a su vez se puede simplificar como:

$$E_n(W^s) = \frac{1}{2} \left\| \overrightarrow{y^s}(\vec{x}_n, W^s) - \vec{t}_n \right\|^2 = \frac{1}{2} \sqrt{\sum_{k=0}^{K} \left(y^s_k(\vec{x}_n, W^s) - t_{k,n} \right)^2}^2 = \sum_{k=0}^{K} E_{k,n}(W^s),$$

con el **error por cada unidad de salida** k:

$$E_{k,n}(W^s) = \frac{1}{2} \left(y^s_k(\vec{x}_n, W^s) - t_{k,n} \right)^2 = \frac{1}{2} \left(g^s\left(p^s_{k,n}\right) - t_{k,n} \right)^2,$$

donde $t_{i,n}$ corresponde al componente i del vector $\vec{t}_n$, y con la función y^s_i definida en la ecuación 8, corresponde a la salida de la unidad $i = 1, \ldots, K$. Calculando la derivada parcial del error para una muestra n respecto a una entrada de la matriz de pesos $W_{m',k'}$ se tiene que:

$$\frac{\mathrm{d}E_n}{\mathrm{d}W^s_{m',k'}} = \frac{\mathrm{d}}{\mathrm{d}W^s_{m',k'}} \left(\sum_{k=0}^{K} E_{k,n}(W^s) \right)$$

puesto que un cambio en el peso $W^s_{m',k'}$ **solo afecta a la unidad de salida** k', la derivada del error para una muestra n respecto a un componente $W^s_{m,k}$ de los pesos viene dada por :

$$\Rightarrow \frac{\mathrm{d}E_{k,n}}{\mathrm{d}W^s_{m',k'}} = \frac{\mathrm{d}}{\mathrm{d}W^s_{m',k'}} \left(\frac{1}{2} \left(y^s_{k',n} - t_{k',n} \right)^2 \right) = \frac{\mathrm{d}}{\mathrm{d}W^s_{m',k'}} \left(\frac{1}{2} \left(g^s\left(p^s_{k',n}\right) - t_{k',n} \right)^2 \right), \tag{15}$$

con el funcional $p^s_{k'}\left(\vec{x}, W^s\right) = \sum_{m=0}^{M} W^s_{m,k'}\, z_m$. Tal derivada parcial planteada en la ecuación 15, se puede descomponer por regla de la cadena como:

$$\frac{\mathrm{d}E_{k,n}}{\mathrm{d}W^s_{m',k'}} = \frac{\mathrm{d}E_{k,n}}{\mathrm{d}y^s_{k',n}} \frac{\mathrm{d}y^s_{k',n}}{\mathrm{d}p^s_{k',n}} \frac{\mathrm{d}p^s_{k',n}}{\mathrm{d}W^s_{m',k'}}. \tag{16}$$

Analizaremos el caso en el que la función de activación sea sigmoidal, $y^s_{k',n} = g^s\left(p^s_{k',n}\right) = \text{sigmoid}\left(p^s_{k',n}\right)$, para la cual su derivada había sido simplificada en la ecuación 6 y dada como $\frac{\mathrm{d}}{\mathrm{d}p^s_{k',n}}\text{sigmoid}\left(p^s_{k',n}\right) = \text{sigmoid}\left(p^s_{k',n}\right)\left(1 - \text{sigmoid}\left(p^s_{k',n}\right)\right)$. Cada una de las derivadas se desarrolla como sigue:

$$\frac{\mathrm{d}E_{k',n}}{\mathrm{d}y^s_{k',n}} = \frac{\mathrm{d}E_{k',n}}{\mathrm{d}y^s_{k',n}}\left(\frac{1}{2}\left(y^s_{k',n} - t_{k',n}\right)^2\right) = y^s_{k',n} - t_{k',n} = g^s\left(p^s_{k',n}\right) - t_{k',n} \tag{17}$$

$$\frac{\mathrm{d}y^s_{k',n}}{\mathrm{d}p^s_{k',n}} = \text{sigmoid}\left(p^s_{k',n}\right)\left(1 - \text{sigmoid}\left(p^s_{k',n}\right)\right) = y^s_{k',n}\left(1 - y^s_{k',n}\right) \tag{18}$$

$$\frac{\mathrm{d}p^s_{k'}}{\mathrm{d}W^s_{m',k'}} = \frac{\mathrm{d}}{\mathrm{d}W^s_{m',k'}}\left(\sum_{m=0}^{M} W^s_{m,k'} y^o_{m,n}\right) = y^o_{m',n}, \tag{19}$$

y es por tales ecuaciones que la derivada expresada en la ecuación 16 vendría dada por:

$$\frac{\mathrm{d}E_{k',n}}{\mathrm{d}W^s_{m',k'}} = \left(y^s_{k',n} - t_{k',n}\right)\,\left(y^s_{k',n}\left(1 - y^s_{k',n}\right)\right)\,y^o_{m',n}. \tag{20}$$

lo que implica que, dado que el peso $W^s_{m',k'}$ solo afecta a la neurona $y^s_{k'}$, la derivada parcial del error para todas las otras $K-1$ neuronas es cero:

$$\frac{\mathrm{d}E_n}{\mathrm{d}W^s_{m',k'}} = \frac{\mathrm{d}E_{k',n}}{\mathrm{d}W^s_{m',k'}} = \left(y^s_{k',n} - t_{k',n}\right)\,\left(y^s_{k',n}\left(1 - y^s_{k',n}\right)\right)\,y^o_{m',n}$$

y **para todas las muestras**:

$$\frac{\mathrm{d}E}{\mathrm{d}W^s_{m',k'}} = \sum_{n=1}^{N}\frac{\mathrm{d}E_n}{\mathrm{d}W^s_{m',k'}} = \sum_{n=1}^{N}\left(y^s_{k',n} - t_{k,n}\right)\,\left(y^s_{k',n}\left(1 - y^s_{k',n}\right)\right)\,y^o_{m',n} \tag{21}$$

la última ecuación establece la derivada parcial $W^s_{m,k}$ del error para todas las muestras y la anterior la derivada parcial para una muestra. Una cualidad importante que se puede concluir de la ecuación 21, es que el valor del gradiente no depende de la matriz de pesos $W^s_{m',k'}$.

Para simplificar la notación, para una muestra n, se define el *delta o cambio de aprendizaje* δ^s_k como:

$$\delta^s_{k',n} = \left(y^s_{k',n} - t_{k',n}\right)\,\left(y^s_{k',n}\left(1 - y^s_{k',n}\right)\right),$$

de esta manera, la actualización del peso $W^s_{m',k'}$ vendría dada, según la ecuación 13 por cada muestra como:

$$W^s_{m',k'}(\tau+1) = W^s_{m',k'}(\tau) - \alpha \triangle W^s_{m',k'}(\tau), \tag{22}$$

con

$$\triangle W^s_{m',k'}(\tau) = \delta^s_{k',n}\, y^o_{m',n}.$$

Tal notación facilitará la propagación del error como se verá posteriormente.

1.3.3 Aprendizaje en la capa oculta

De manera similar, se desarrollará la ecuación del vector de actualización de los pesos, pero para los pesos que conectan la capa de entrada con la oculta:

$$W^o_{d',m'}(\tau+1) = W^o_{d',m'}(\tau) - \alpha \triangle W^o_{d',m'}(\tau),$$

donde el vector de actualización, de forma similar a lo desarrollado anteriormente, está dado por el negativo del vector gradiente:

$$\triangle W^o_{d',m'}(\tau) = \frac{\mathrm{d}}{\mathrm{d}W^o_{d',m'}} \left(\sum_{n=1}^{N} E_n \left(W^o_{d',m'} \right) \right)$$

Observe que el gradiente se calcula respecto a los pesos de la primer capa, para la función de **error a la salida.** De manera similar a lo realizado anteriormente, definimos el error por cada unidad k de la capa de salida:

$$E_{k,n}(W^o) = \frac{1}{2}\left(y^s_k(\vec{x}_n, W^o) - t_{k,n}\right)^2 = \frac{1}{2}\left(y^s_k\left(p^s_k(\vec{x}_n, W^o)\right) - t_{k,n}\right)^2$$

$$\Rightarrow E_{k,n}(W^o) = \frac{1}{2}\left(y^s_k\left(\sum_{m=0}^{M} W^s_{m,k}\, y^o_m\left(p^o_m(\vec{x}_n, W^o)\right)\right) - t_{k,n}\right)^2 = \frac{1}{2}\left(g^s\left(\sum_{m=0}^{M} W^s_{m,k}\, g^o\left(\sum_{d=1}^{D} W^o_{d,m} x_d\right)\right) - t_{k,n}\right)^2$$

recordando que: $y^s_{k,n} = g^s\left(p^s_{k,n}\right)$, $p^s_{k,n}(\vec{x}_n, W^s) = \sum_{m=0}^{M} W^s_{m,k}\, y^o_m$, $y^o_m(\vec{x}_n, W^o) = g^o(p^o_m)$, $p^o_m(\vec{x}, W^o) = \sum_{d=0}^{D} W^o_{d,m} x_d$, tenemos que la derivada del error se puede descomponer por regla de la cadena como:

$$\frac{\mathrm{d}E_{k,n}}{\mathrm{d}W^o_{d',m'}} = \frac{\mathrm{d}E_{k,n}}{\mathrm{d}y^s_{k,n}} \frac{\mathrm{d}y^s_{k,n}}{\mathrm{d}p^s_{k,n}} \frac{\mathrm{d}p^s_{k,n}}{\mathrm{d}y^o_{m',n}} \frac{\mathrm{d}y^o_{m',n}}{\mathrm{d}p^o_{m',n}} \frac{\mathrm{d}p^o_{m',n}}{W^o_{d',m'}}.$$

En este caso, a diferencia del análisis del impacto de un cambio en un peso de la capa de salida, un cambio en la capa oculta ocasiona un cambio en el error de todas las unidades en la capa de salida, el cual dice que se **propaga hacia la capa de salida**, por lo que entonces:

$$\frac{dE_n}{dW^o_{d',m'}} = \sum_{k=0}^{K} \frac{\mathrm{d}E_{k,n}}{\mathrm{d}W^o_{d',m'}} = \sum_{k=0}^{K} \left(\frac{\mathrm{d}E_{k,n}}{\mathrm{d}y^s_{k,n}} \frac{\mathrm{d}y^s_{k,n}}{\mathrm{d}p^s_{k,n}} \frac{\mathrm{d}p^s_{k,n}}{\mathrm{d}y^o_{m',n}} \right) \frac{\mathrm{d}y^o_{m',n}}{\mathrm{d}p^o_{m',n}} \frac{\mathrm{d}p^o_{m',n}}{W^o_{d',m'}}$$

Con las derivadas parciales dadas por:

$$\frac{\mathrm{d}E_{k,n}}{\mathrm{d}y_k^s} = g^s\left(p_k^s\right) - t_{k,n} = y_{k,n}^s - t_{k,n} \tag{23}$$

$$\frac{\mathrm{d}y_{k,n}^s}{\mathrm{d}p_{k,n}^s} = \text{sigmoid}\left(p_{k,n}^s\right)\left(1 - \text{sigmoid}\left(p_{k,n}^s\right)\right) = y_{k,n}^s\left(1 - y_{k,n}^s\right) \tag{24}$$

$$\frac{\mathrm{d}p_{k,n}^s}{\mathrm{d}y_{m'}^o} = W_{m',k}^s$$

$$\frac{\mathrm{d}y_{m',n}^o}{\mathrm{d}p_{m',n}^o} = \text{sigmoid}\left(p_{m',n}^o\right)\left(1 - \text{sigmoid}\left(p_{m',n}^o\right)\right) = y_{m',n}^o\left(1 - y_{m',n}^o\right)$$

$$\frac{\mathrm{d}p_{m',n}^o}{W_{d',m'}^o} = x_{d'},$$

por lo que entonces la derivada parcial del error respecto a un peso específico, para una muestra n, está dada por:

$$\frac{dE_n}{dW_{d',m'}^o} = \sum_{k=0}^{K}\left(\left(y_{k,n}^s - t_{k,n}\right)\left(y_{k,n}^s\left(1 - y_{k,n}^s\right)\right)\left(W_{m',k}^s\right)\right)\left(y_{m',n}^o\left(1 - y_{m',n}^o\right)\right)x_{d'}. \tag{25}$$

Para simplificar la notación, recuerde que se había definido en el aprendizaje de la capa de salida el delta $\delta_{k,n}^s = \left(y_{k,n}^s - t_{k,n}\right)\left(y_{k,n}^s\left(1 - y_{k,n}^s\right)\right)$, y tal término lo podemos encontrar en la ecuación anterior 25. Basado en lo anterior, podemos definir el delta en la capa oculta como:

$$\delta_{m',n}^o = \left(\sum_{k=0}^{K}\delta_{k,n}^s W_{m',k}^s\right)\left(y_{m',n}^o\left(1 - y_{m',n}^o\right)\right)$$

y de manera análoga para la notación en la capa anterior, respecto a la ecuación de aprendizaje $W_{d',m'}^o(\tau+1) = W_{d',m'}^o(\tau) - \alpha \triangle W_{d',m'}^o(\tau)$, definimos:

$$\triangle W_{d',m'}^o(\tau) = \delta_{m',n}^o x_{d'}$$

1.3.4 Criterio de parada

El criterio de parada puede fijarse como un número de iteraciones P a cumplir, o un máximo porcentaje de las muestras a clasificar incorrectamente.

1.3.5 Ejemplo de entrenamiento

La Figura 5 muestra una red neuronal con una entrada $\vec{x} \in \mathbb{R}^2$ (con $D = 2$, sin incluir la neurona con valor unitario), con $\vec{y^o} \in \mathbb{R}^2$ (con $D = 2$, sin incluir la neurona con valor unitario) y $\vec{y^s} \in \mathbb{R}^1$ ($K = 1$ denotando la pertenencia a una clase $C = 1$ y la no pertenencia a la misma). A continuación se definen

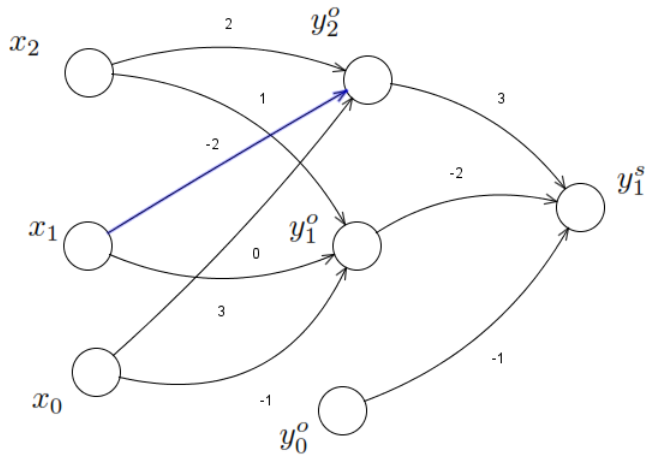

Figura 5: Ejemplo de una red neuronal de dos capas.

los valores de la entrada y los pesos en la capa oculta y de salida (los cuales se pueden suponer fueron inicializados aleatoriamente):

$$\begin{array}{llll} x_0 = 1 & t_1 = 0 & W^o_{0,1} = -1 & W^s_{0,1} = -1 \\ x_1 = 0 & & W^o_{0,2} = 0 & W^s_{1,1} = -2 \\ x_2 = 1 & & W^o_{1,1} = 3 & W^s_{2,1} = 3 \\ & & W^o_{1,2} = -2 & \\ & & W^o_{2,1} = 1 & \\ & & W^o_{2,2} = 2 & \end{array}$$

Suponga que $\alpha = 1$.

Para actualizar los pesos de la red para la iteración $\tau = 1$, realizamos las dos etapas: pasada hacia adelante y retropropagación del error

Pasada hacia adelante A partir de los datos anteriores, calculamos primero los pesos netos para la capa oculta:

$$\begin{aligned} p^o_1 = &\quad W^o_{0,1}\, x_0 + W^o_{1,1}\, x_1 + W^o_{2,1}\, x_2 = -1 * 1 + 3 * 0 + 1 * 1 = 0 \\ p^o_2 = &\quad W^o_{0,2}\, x_0 + W^o_{1,2}\, x_1 + W^o_{2,2}\, x_2 = 0 * 1 + -2 * 0 + 2 * 1 = 2 \end{aligned}$$

para posteriormente calcular la salida de cada unidad oculta:

$$\begin{aligned} \Rightarrow y^o_1 = g^o\left(p^o_1\right) = \tfrac{1}{(1+e^0)} = 0{,}5 \\ \Rightarrow y^o_2 = g^o\left(p^o_2\right) = \tfrac{1}{(1+e^{-2})} = 0{,}8808 \end{aligned}$$

Respecto a la capa de salida se tiene que:

$$p_1^s = \quad W_{0,1}^s\, y_0^o + W_{1,1}^s\, y_1^o + W_{2,1}^s\, y_2^o = -1 * 1 + -2 * 0{,}5 + 3 * 0{,}8808 = 0{,}6424$$

$$\Rightarrow y_1^s = g^s\left(p_1^s\right) = \tfrac{1}{(1+e^{-0{,}64})} = 0{,}6553$$

Pasada hacia atrás y actualización de los pesos Para la capa de salida, calculamos el delta de la única unidad como:

$$\delta_1^s = \left(y_1^s - t_{1,1}\right)\, \left(y_1^s\left(1 - y_1^s\right)\right) = \left(0{,}6553 - 0\right)\left(0{,}6553\ \left(1 - 0{,}6553\right)\right) = 0{,}148$$

Con base al cálculo del delta para la capa de salida, se actualizan los pesos en la capa de salida según la ecuación $W_{m,k}^s\left(\tau + 1\right) = W_{m,k}^s\left(\tau\right) - \alpha\, \delta_k^s\, y_m^o$:

$$\begin{gathered} W_{0,1}^s\left(\tau + 1\right) = W_{0,1}^s\left(\tau\right) - 1\, \delta_1^s\, y_0^o = -1 - 0{,}148 * 1 = -1{,}148 \\ W_{1,1}^s\left(\tau + 1\right) = W_{1,1}^s\left(\tau\right) - 1\, \delta_1^s\, y_1^o = -2 - 0{,}148 * 0{,}5 = -2{,}074 \\ W_{2,1}^s\left(\tau + 1\right) = W_{2,1}^s\left(\tau\right) - 1\, \delta_1^s\, y_2^o = 3 - 0{,}148 * 0{,}8808 = 2{,}8696 \end{gathered}$$

Y para la capa oculta tenemos de forma similar, el delta para las dos unidades:

$$\delta_1^o = \left(\sum_{k=1}^{K=1} \delta_k^s\, W_{1,k}^s\right)\left(y_1^o\left(1 - y_1^o\right)\right) = \left(0{,}148 * -2\right) * \left(0{,}5 * \left(1 - 0{,}5\right)\right) = -0{,}0740$$

$$\delta_2^o = \left(\sum_{k=1}^{K=1} \delta_k^s\, W_{2,k}^s\right)\left(y_2^o\left(1 - y_2^o\right)\right) = \left(0{,}148 * 3\right) * \left(0{,}8808 * \left(1 - 0{,}8808\right)\right) = 0{,}0466$$

Por lo que los pesos nuevos para la capa oculta según la ecuación $W_{d,m}^o\left(\tau + 1\right) = W_{d,m}^o\left(\tau\right) - \alpha\, \delta_m^o\, x_d$, vienen dados por:

$$\begin{gathered} W_{0,1}^o\left(\tau + 1\right) = W_{0,1}^o\left(\tau\right) - 1\, \delta_1^o\, x_0 = -1 + 0{,}074 * 1 = -0{,}9260 \\ W_{1,1}^o\left(\tau + 1\right) = W_{1,1}^o\left(\tau\right) - 1\, \delta_1^o\, x_1 = 3 + 0{,}074 * 0 = 3 \\ W_{2,1}^o\left(\tau + 1\right) = W_{2,1}^o\left(\tau\right) - 1\, \delta_1^o\, x_2 = 1 + 0{,}074 * 1 = 1{,}074 \\ W_{0,2}^o\left(\tau + 1\right) = W_{0,2}^o\left(\tau\right) - 1\, \delta_2^o\, x_0 = 0 - 1 * 0{,}0466 * 1 = -0{,}0466 \\ W_{1,2}^o\left(\tau + 1\right) = W_{1,2}^o\left(\tau\right) - 1\, \delta_2^o\, x_1 = -2 - 1 * 0{,}0466 * 0 = -2 \\ W_{2,2}^o\left(\tau + 1\right) = W_{2,2}^o\left(\tau\right) - 1\, \delta_2^o\, x_2 = 2 - 1 * 0{,}0466 * 1 = 1{,}9534 \end{gathered}$$

1.4 Descenso de gradiente estocástico

En la ecuación general de actualización de los pesos:

$$\vec{w}\left(\tau + 1\right) = \vec{w}\left(\tau\right) - \alpha\, \nabla E\left(\vec{w}\left(\tau\right)\right) \tag{26}$$

se estableció que el gradiente de error se calcula sobre las N muestras que conforman el conjunto de muestras de entrenamiento:

$$E\left(\vec{w}\right) = \frac{1}{2} \sum_{n=1}^{N} \left\| y\left(\vec{x}_n, \vec{w}\right) - \vec{t}_n \right\|^2,$$

para facilitar su análisis la podemos reescribir como:

$$E\left(\vec{w}\right) = \sum_{n=1}^{N} E_n\left(\vec{w}\right)$$

donde $E_n = \frac{1}{2}\left\|y\left(\vec{x}_n, \vec{w}\right) - t_n\right\|^2$, correspondiente al error en una sola muestra. Con esto la Ecuación 26 se re-escribe como:

$$\vec{w}\left(\tau+1\right) = \vec{w}\left(\tau\right) - \alpha\,\frac{1}{Q}\left(\sum_{n=1}^{Q} \nabla E_n\left(\vec{w}\right)\right) \tag{27}$$

Se recomienda normalizar el aporte de cada muestra al cambio del gradiente (multiplicando por $\frac{1}{Q}$), para evitar que los *saltos* en la superficie de error dependan de la cantidad de muestras en el conjunto de datos.

Respecto a la cantidad y las muestras del conjunto de entrenamiento Q, se definen tres enfoques de entrenamiento distintos:

1. Descenso de gradiente en el lote: $Q = N$, en este caso, se toman en cuenta todas las muestras del conjunto de muestras de entrenamiento N, sin ningún criterio estocástico.

2. Descenso de gradiente estocástico en línea: $Q = 1$. Se escoge una sola muestra aleatoria para posteriormente actualizar los pesos.

3. Descenso de gradiente estocástico por mini-lotes: $1 < Q < N$. En este caso se construye un sub-conjunto de las N muestras de entrenamiento para realizar el entrenamiento. Cuando $Q \to 1$, la señal de error de la función tenderá a ser más ruidosa, e inestable, mientras que si $Q \to N$, la señal será más estable, pero tardará más en converger. Q corresponde a la **cantidad de muestras del mini-lote**.

De esta forma, el algoritmo de entrenamiento generalizado, para R pasadas o epochs, está dado por:

1. $\tau \leftarrow 0$

2. Mientras $\tau \leftarrow R$:

   (a) $\vec{w}\left(\tau+1\right) \leftarrow \vec{w}\left(\tau\right) - \alpha\,\frac{1}{Q}\left(\sum_{n=1}^{Q} \nabla E_n\left(\vec{w}\right)\right)$

   (b) $\tau \leftarrow \tau + 1$

1.5 Ajuste de parámetros y regularización en redes neuronales

El número de neuronas en la capa de entrada D y las K neuronas en la capa de salida son generalmente determinadas por la dimensionalidad de las entradas y la cantidad de clases, respectivamente. La cantidad de neuronas en la capa oculta M es un parámetro libre, el cual controla el número de parámetros

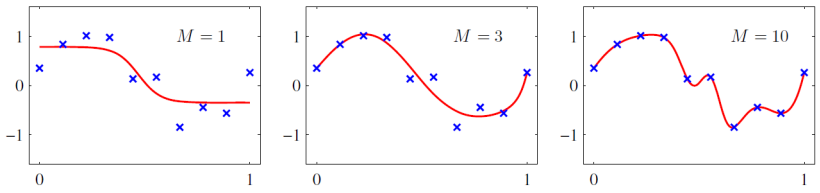

Figura 6: Redes neuronales con distintos valores de M para el problema de la regresión, tomado de , tomado de [1].

del modelo de la red, por lo que es usual buscar una cantidad M de neuronas que generen un modelo con un bajo sobre-ajuste, maximizando su generalización. La Figura 6 muestra como la red neuronal se comporta al ajustarse a 10 muestras para el problema de regresión, con distintos valores de M.

Sin embargo, el error de generalización no es una simple función de M, por la existencia de mínimos locales, por lo que se recomienda realizar una gráfica del error en un **conjunto de datos de validación** para evaluar el mejor modelo en función de M. Otra alternativa consisten en implementar una regularización como la analizada para el caso de la regresión polinomial, con una ecuación del error modificado como y fijando un M relativamente grande:

$$\widetilde{E}\left(\vec{w}\right) = E\left(\vec{w}\right) + \frac{\lambda}{2}\vec{w}^T\vec{w}$$

Una heurística común es fijar el número de neuronas en la capa oculta M en el siguiente rango:

$$\frac{D}{2} \leq M \leq 2D$$

donde usualmente se consiguen los menores errores en el conjunto de datos de validación con valores menores.

1.6 Invariantes

Como se discutió en el capítulo de preprocesamiento, la predicción de una clase para una nueva muestra $\vec{x}$ debe ser *invariante* bajo una o más transformaciones de tal muestra de entrada $\vec{x}$. Por ejemplo, para la clasificación de imágenes digitales de firmas, para una entrada $\vec{x}_i$, la clase asignada por el clasificador debe ser la misma sin importar cambios en la traslación o la escala de tal muestra.

Si existe una suficiente cantidad de muestras, un modelo adaptativo como una red perceptrón multi-capa puede llegar a *aprender* o generalizar tal invarianza, incluyendo entonces una cantidad grande de los ejemplos con las distintas transformaciones, por ejemplo, incluyendo muchas muestras rotadas para el caso de las firmas. Sin embargo, esto puede ser impráctico si el número de posibles transformaciones es muy alto, o existen pocas muestras en el conjunto original de datos. Existen entonces 3 alternativas básicas para lidiar con las variaciones en las muestras de entrada:

1. Aumentar el conjunto de datos usando réplicas del conjunto de datos originales, de acuerdo a las invariantes deseadas.

2. Implementar una etapa de preprocesamiento y extracción de características, donde estas últimas sean invariantes a las características deseadas, lo que requiere también la aplicación de ambas etapas para las nuevas muestras a clasificar.

3. Construir las propiedades invariantes dentro de la misma red, usando lo que que se conoce como espacios receptivos y pesos compartidos, conceptos implementados en las redes convolucionales.

4. Combinar los enfoques 1 y 3.

Referencias

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.