Entrenamiento de un perceptrón multicapa mediante el algoritmo de retropropagación

E.T.S.I. Sistemas Informaticos – Inteligencia Artificial
Alberto Díaz Álvarez

El entrenamiento de un perceptrón multicapa (mediante aprendizaje supervisado) se basa en estar constantemente enseñándole las entradas y salidas de nuestro conjunto de datos. Esto implica dos pasos, la propagación de los datos de entrada y la retropropagación del error cometido. Veamos cómo funciona.

1 Introducción

El algoritmo de retropropagación se va a encargar de *propagar* hacia atrás el error existente entre la salida real de un perceptrón multicapa y la salida esperada, actualizando los pesos de la red. De esta manera, se espera que tras este cambio de pesos, la red cometa un error más pequeño la próxima vez que vea el mismo ejemplo. Esto aplicado con muchos ejemplos muchas veces consigue (o mejor dicho, se *espera* conseguir) a la larga que la red aprenda a resolver el problema al que pertenecen los datos.

Hemos hablado del error existente. Esto nos obliga a realizar primero una inferencia, esto es, coger los datos de entrada de nuestro ejemplo y **propagarlo** hacia adelante para ver qué resultado nos da nuestra red. Así, sabiendo el resultado que da nuestra red para esos datos de entrada y el resultado que esperamos, podemos calcular el error.

Por tanto, para entrenar una red tenemos que repetir muchas muchas veces con cada ejemplo del dataset lo siguiente:

1. **Propagar** los datos de entrada de nuestro ejemplo hasta obtener el resultado.
2. Comparar los datos de salida esperados de nuestro ejemplo con los datos de salida obtenidos de la red para obtener el error.
3. **Retropropagar** el error cometido para actualizar los pesos.

2 Entrenamiento de una red neuronal

Supongamos que tenemos una red de L capas donde sus neuronas tienen una función de activación sigmoidal (en la ecuación 1 vemos tanto la función como su derivada).

$$f_a(x) = \frac{1}{1+e^x}; \quad f'_a(x) = x \cdot (1-x) \tag{1}$$

El ejemplo que vemos en la figura 1 tiene $L = 2$ capas, la capa oculta y la de salida. Ojo, aunque a la entrada se la denomina a veces *capa de entrada*, las capas son exclusivamente aquellas que están compuestas de neuronas artificiales.

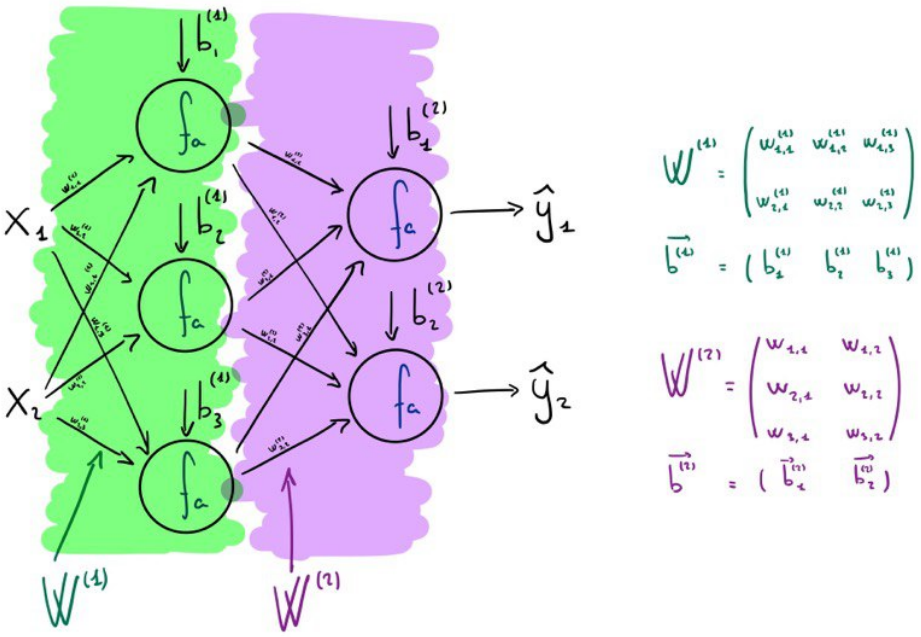


Figura 1: *Ejemplo de perceptrón multicapa de dos capas, una oculta (la de los pesos $W^{(1)}$, los bias $\vec{b}^{(1)}$ y sus neuronas) y una de salida (la otra). A la derecha, las matrices y vectores correspondientes, donde el número de filas se corresponde al número de neuronas de la capa precedente y el número de columnas al número de neuronas de la capa actual.*

También vamos a suponer que tenemos un ejemplo E con el que vamos a entrenar. Este ejemplo E estará compuesto por unos datos de entrada X y unos datos esperados de salida Y.

Por último, estableceremos un parámetro α denominado *factor de aprendizaje* o *tasa de aprendizaje* que servirá como factor de ajuste en la velocidad de entrenamiento.

2.1 Propagación

La propagación o inferencia consiste en obtener un resultado de la red a partir de unos datos de entrada. Durante el entrenamiento nosotros le iremos dando los datos de entrada del conjunto de entrenamiento, en nuestro caso X.

Como un perceptrón multicapa está compuesto de varias capas, el proceso será ir aplicando la ecuación 2 a cada una de las capas:

$$\vec{S}^{(k)} = f_a(\vec{S}^{(k-1)} W^{(k)} + \vec{b}^{(k)}) \tag{2}$$

Que desarrollado quedaría más o menos como se ve en la ecuación 3

$$\vec{S}^{(k)} = f_a((s_1, s_2, \dots, s_n)^{(k-1)} \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{pmatrix}^{(k)} + (b_1, b_2, \dots, b_m)^{(k)}) \quad (3)$$

peso de la conexión entre la neurona de entrada 1 y la de salida 2

m neuronas en la capa k

n conexiones de entrada

Como se puede ver, el orden de aplicación es de la primera a la última capa, ya que para calcular la salida de una capa se usa como entrada la salida de la capa anterior, así hasta llegar a $S^{(0)}$ que sería equivalente a nuestro vector de entrada X. Asímismo, al valor de salida de la última capa $S^{(L)}$ le denominaremos $\hat{Y}$.

2.2 Retropropagación

Consiste en propagar desde la última capa hasta la primera el error existente entre el valor $\hat{Y}$ que devuelve la red para la entrada X y el valor de salida esperado Y.

Lo primero que hay que hacer es calcular el error cometido tanto en cada una de las capas para actualizar sus correspondientes pesos. La clave aquí es que hay que calcular primero el error de la última capa, y luego los anteriores, usando el error de su correspondiente capa siguiente. Se denota como $\vec{\delta}^{(k)}$ y se define como indica la ecuación 4:

$$\vec{\delta}^{(k)} = \begin{cases} (Y - \hat{Y}) \odot f_a'(\vec{S}^{(k)}) & si \quad k = L \\ \vec{\delta}^{(k+1)} W^{(k+1)T} \odot f_a'(\vec{S}^{(k)}) & si \quad k < L \end{cases} \quad (4)$$

la salida esperada — la salida de nuestra red ($S^{(k)}$)

en la última capa — el número de capas de nuestra red

el error de la siguiente capa — el error cometido — ¡transpuesta! — derivada de la función de activación

¡producto punto a punto!

Después, **una vez calculados los errores de todas las capas**, pasamos a calcular las matrices de cambio, esto es, las matrices que contendrán los valores con los que se van a modificar las matrices de pesos de cada capa. Estas se calcularán como indican la ecuación 5:

$$\begin{aligned} \Delta W^{(k)} &= \alpha \cdot \vec{S}^{(k-1)T} \vec{\delta}^{(k)} \\ \Delta \vec{b}^{(k)} &= \alpha \cdot \vec{\delta}^{(k)} \end{aligned} \quad (5)$$

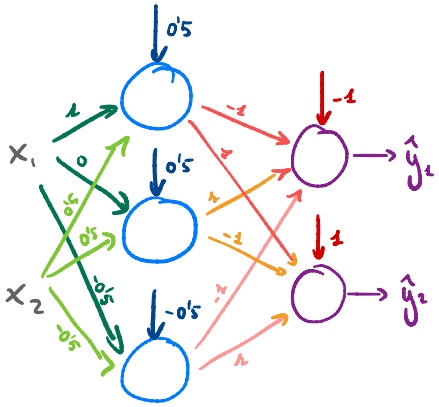
como la regla delta, pero con este error

¡transpuesta!

Y por fin, ya con todas las matrices de cambio hechas, lo único que nos quedaría sería actualizar los valores de las matrices de peso de la red, cada una con su matriz de cambio correspondiente, tal y como se expresa enla ecuación 6

$$\begin{aligned} W_{t+1}^{(k)} &= W_t^{(k)} + \Delta W^{(k)} \\ \vec{b}_{t+1}^{(k)} &= \vec{b}_t^{(k)} + \Delta \vec{b}^{(k)} \end{aligned} \quad (6)$$

3 Ejemplo



Ejemplo: $\overbrace{(x_1, x_2}^{\text{entradas}}, \underbrace{y_1, y_2}_{\text{salidas}}) = (1, 0, 0, 1)$

Función de activación: $\begin{cases} f_a(x) = \dfrac{1}{1+e^{-x}} \\ f_a'(x) = x \cdot (1-x) \end{cases}$

Factor de aprendizaje: $\alpha = 0'5$

3.1. Propagación

$$\vec{S}^{(k)} = f_a(\vec{S}^{(k-1)} W^{(k)} + \vec{b}^{(k)}) \quad \begin{cases} \vec{S}^{(1)} = f_a((1\ 0)\cdot\begin{pmatrix} 1 & 0 & -0'5 \\ 0'5 & 0'5 & -0'5 \end{pmatrix} + (0'5\ 0'5\ -0'5)) = f_a(1'5\ 0'5\ -1) = (0'818\ 0'622\ 0'269) \\ \vec{S}^{(2)} = f_a((0'818\ 0'622\ 0'269)\cdot\begin{pmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} + (-1\ 1)) = f_a(-1'465\ 1'465) = (0'188,\ 0'812) \end{cases}$$

3.2. Retropropagación

$$S^{(k)} = \begin{cases} (Y - \hat{Y}) \odot f_a'(S^{(k)}) & \text{si } k = L \\ S^{(k+1)} W^{(k+1)T} \odot f_a'(S^{(k)}) & \text{si } k < L \end{cases} \Rightarrow \begin{cases} S^{(2)} = ((0\ 1) - (0'188\ 0'812)) \odot (0'188\ 0'812) \odot (1 - (0'188\ 0'812)) = (-0'029\ 0'029) \\ S^{(1)} = (-0'029\ 0'029)\cdot\begin{pmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix} \odot (0'818\ 0'622\ 0'269) \odot (1 - (0'818\ 0'622\ 0'269)) = \\ \quad = (0'009\ \ -0'014\ \ 0'011) \end{cases}$$

$$\begin{matrix} \Delta W^{(k)} = \alpha \vec{S}^{(k-1)T} S^{(k)} \\ \Delta \vec{b}^{(k)} = \alpha S^{(k)} \end{matrix} \Rightarrow \begin{cases} \Delta W^{(1)} = 0'5 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot (0'009\ \ -0'014\ \ 0'011) = \begin{pmatrix} 0'005 & -0'007 & 0'006 \\ 0 & 0 & 0 \end{pmatrix} \\ \Delta \vec{b}^{(1)} = 0'5 \cdot (0'009\ \ -0'014\ \ 0'011) = (0'005\ \ -0'007\ \ 0'006) \\ \Delta W^{(2)} = 0'5 \cdot \begin{pmatrix} 0'818 \\ 0'622 \\ 0'269 \end{pmatrix} \cdot (-0'029\ \ 0'029) = \begin{pmatrix} -0'012 & 0'012 \\ -0'009 & 0'009 \\ -0'004 & 0'004 \end{pmatrix} \\ \Delta \vec{b}^{(2)} = 0'5 \cdot (-0'029\ \ 0'029) = (-0'145\ \ 0'145) \end{cases}$$

$$\begin{matrix} W_{t+1}^{(k)} = W_t^{(k)} + \Delta W^{(k)} \\ \vec{b}_{t+1}^{(k)} = \vec{b}_t^{(k)} + \Delta \vec{b}^{(k)} \end{matrix} \Rightarrow \begin{cases} W^{(1)} = \begin{pmatrix} 1 & 0 & -0'5 \\ 0'5 & 0'5 & -0'5 \end{pmatrix} + \begin{pmatrix} 0'005 & -0'007 & 0'006 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1'005 & -0'007 & -0'494 \\ 0'5 & 0'5 & -0'5 \end{pmatrix} \\ \vec{b}^{(1)} = (0'5\ 0'5\ -0'5) + (0'005\ \ -0'007\ \ 0'006) = (0'505\ \ 0'493\ \ 0'494) \\ W^{(2)} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} + \begin{pmatrix} -0'012 & 0'012 \\ -0'009 & 0'009 \\ -0'004 & 0'004 \end{pmatrix} = \begin{pmatrix} -1'012 & 1'012 \\ 0'991 & -0'991 \\ -1'004 & 1'004 \end{pmatrix} \\ \vec{b}^{(2)} = (-1\ \ 1) + (-0'145\ \ 0'145) = (-1'145\ \ 1'145) \end{cases}$$