Perceptrón multicapa y el algoritmo de backpropagation

Juan I. Perotti*
Instituto de Física Enrique Gaviola (IFEG-CONICET),
Facultad de Matemática, Astronomía, Física y Computación,
Universidad Nacional de Córdoba, Ciudad Universitaria, 5000 Córdoba, Argentina
(Dated: October 24, 2024)

I. PERCEPTRÓN MULTICAPA

Consideraremos un perceptrón multicapa, con capas enumeradas por $l = 0, 1, ..., L$. Denotemos por x_i^l el estado de la i-ésima neurona en la capa l. Diremos que la red posee n^l neuronas $i = 1, ..., n^l$ en la l-ésima capa. En particular, x^0 denota el vector de estados de la capa de entrada y x^L el vector de estados de la capa de salida. Se tiene que

$$x_i^l = g(h_i^l) \tag{1}$$

donde $g : \mathbb{R} \to \mathbb{R}$ es una función de activación, por ejemplo una ReLU, y

$$h_i^l = \sum_j w_{ij}^l x_j^{l-1} \tag{2}$$

es el campo local sufrido por la i-ésima neurona en la l-ésima capa . Además, w_{ij}^l denota la intensidad de la sinapsis que conecta la j-ésima neurona en la $(l-1)$-ésima capa con la i-ésima neurona en la l-ésima capa. Notar, la red depende de las matrices de pesos sinápticos $w^1, w^2, ..., w^L$.

A. Umbrales de activación

En cada una de las capas $l = 0, 1, ..., L-1$, se agrega una neurona extra $i = n^l + 1$ con un estado fijo $x_{n^l+1}^l = -1$. De esta manera, una nueva sinapsis $u_i^l := w_{i,n^{l-1}+1}^l$ hace las veces de umbral de activación de la i-ésima neurona en la l-ésima capa, ya que

$$h_i^{l+1} = w_{i,n^l+1}^{l+1} x_{n^l+1}^l + \sum_{j=1}^{n^l} w_{ij}^{l+1} x_j^l = -u_i^{l+1} + \sum_{j=1}^{n^l} w_{ij}^{l+1} x_j^l \tag{3}$$

II. CONJUNTO DE ENTRENAMIENTO

Los datos de entrenamiento consisten en un conjunto de pares $\{(e^m, s^m) : m = 1, ..., M\}$ donde $e^m \in \mathbb{R}^{n_0}$ y $s^m \in \mathbb{R}^{n_L}$ son vectores que representan el m-ésimo par de entrada-salida o *ejemplo* que debe aprender la red.

III. FUNCIÓN COSTO: EL ERROR CUADRÁTICO

Si pensamos que la salida de la red es una función de la entrada, i.e. que $x^L(x^0)$, podemos evaluar el error que comete la red sobre el conjunto de entramiento utilizando el *error cuadrático*

$$E = \sum_{m=1}^{M} F^m$$

como *función costo*, donde

$$F^m = \frac{1}{2} \sum_{i=1}^{n^L} (x_i^L(x^0 = e^m) - s_i^m)^2$$

es el error cuadrático que comete la red sobre el m-ésimo ejemplo.

IV. ENTRENAMIENTO: DESCENSO POR EL GRADIENTE

Entrenar la red consisten en encontrar valores de los pesos sinápticos w_{ij}^l que minimicen el error E. Para ello, expresamos el error en función de dichos pesos y calculamos las componentes de su gradiente

$$\frac{\partial E}{\partial w_{ij}^l} = \sum_m \frac{\partial F^m}{\partial w_{ij}^l}$$

De esta manera, podemos utilizar el algoritmo de descenso por el gradiente para actualizar los pesos hasta que el error alcance un mínimo global. Más precisamente, partiendo de valores aleatorios $(w_{ij}^l)^0$ para los pesos sinápticos, actualizamos iterativamente a los mismos con la siguiente regla

$$(w_{ij}^l)^{t+1} = (w_{ij}^l)^t - \eta \frac{\partial F^m}{\partial w_{ij}^l}((w_{ij}^l)^t) \tag{4}$$

para todo l, ij y m, donde el parámetro $0 < \eta \ll 1$ controla la tasa de aprendizaje. La iteración se detiene cuando ya no se advierten reducciones significativas del error E.

V. CÁLCULO DEL GRADIENTE DEL ERROR CUADRÁTICO

Con el fin de simplificar la notación, elegimos un valor arbitrario de m y obviamos la dependencia de las expresiones con éste índice.

Notar que los vectores x^l y h^l sólo dependen de las matrices $w^1, ..., w^l$. De esta manera, observamos que

$$\frac{\partial x_i^l}{\partial w_{pq}^r} = g'(h_i^l)\frac{\partial h_i^l}{\partial w_{pq}^r}$$

si $r \leq l$, y

$$\frac{\partial x_i^l}{\partial w_{pq}^r} = 0$$

en caso contrario. Por otro lado,

$$\begin{aligned}\frac{\partial h_i^l}{\partial w_{pq}^r} &= \frac{\partial}{\partial w_{pq}^r}\left(\sum_j w_{ij}^l x_j^{l-1}\right) \\ &= \sum_j w_{ij}^l \frac{\partial x_j^{l-1}}{\partial w_{pq}^r}\end{aligned}$$

si $r < l$, y

$$\frac{\partial h_i^l}{\partial w_{pq}^l} = \sum_j \delta_{ip}\delta_{jq}x_j^{l-1} = \delta_{ip}x_q^{l-1}$$

Con estas ecuaciones se pueden establecer una relación de recurrencia que nos permite calcular las componentes del

gradiente de F. A saber

$$\begin{aligned}
\frac{\partial F}{\partial w_{pq}^r} &= \sum_i (x_i^L - s_i) \frac{\partial x^L}{\partial w_{pq}^r} \\
&= \sum_i (x_i^L - s_i) g'(h_i^L) \frac{\partial h_i^L}{\partial w_{pq}^r} \\
&= \sum_i D_i^L \frac{\partial h_i^L}{\partial w_{pq}^r} \\
&= \sum_i D_i^L \sum_j w_{ij}^L \frac{\partial x_i^{L-1}}{\partial w_{pq}^r} \\
&= \sum_i D_i^L \sum_j w_{ij}^L g'(h_j^{L-1}) \frac{\partial h_i^{L-1}}{\partial w_{pq}^r} \\
&= \sum_j \left(g'(h_j^{L-1}) \sum_i w_{ij}^L D_i^L \right) \frac{\partial h_i^{L-1}}{\partial w_{pq}^r} \\
&= \sum_j D_j^{L-1} \frac{\partial h_i^{L-1}}{\partial w_{pq}^r}
\end{aligned}$$

donde

$$D_i^L := (x_i^L - s_i) g'(h_i^L) \tag{5}$$

y

$$D_j^{L-1} := g'(h_j^{L-1}) \sum_i w_{ij}^L D_i^L$$

representan los *errores locales* de la i-ésima neurona en la L-ésima capa y la j-ésima neurona en la $(L-1)$-ésima capa, respectivamente.

El anterior procedimiento puede continuarse capa por capa, con cada capa l tal que $r < l$ o, equivalentemente, $l > r$, de manera que

$$\frac{\partial F}{\partial w_{pq}^r} = \sum_j D_j^l \frac{\partial h_j^l}{\partial w_{pq}^r}$$

donde

$$D_j^l := g'(h_j^l) \sum_i w_{ij}^{l+1} D_i^{l+1} \tag{6}$$

hasta que eventualmente se alcanza la capa $l = r$, y se obtiene

$$\begin{aligned}
\frac{\partial F}{\partial w_{pq}^r} &= \sum_j D_j^r \frac{\partial h_j^r}{\partial w_{pq}^r} \\
&= \sum_j D_j^r \delta_{jp} x_q^{r-1} \\
&= D_p^r x_q^{r-1}
\end{aligned} \tag{7}$$

En particular, este último resultado se verifica para el caso $r = L$ de pq arbitrario, y también se verifica para el caso en que $q = n^{r-1} + 1$, y r y p son arbitrarios, en donde $x_q^{r-1} = -1$ corresponde al estado fijo de la neurona en la capa $(r-1)$-ésima que permite simular la acción de umbrales en la capa r-ésima, tal como se describe en la Ec. 3.

VI. EL ALGORITMO DE BACKPROPAGATION

Los resultados anteriores pueden condensarse en el llamado *algoritmo de backpropagation*, el cuál permite el cálculo del gradiente y la actualización de los pesos sinápticos, y consiste en la siguiente lista de pasos. Para cada ejemplo $m = 1, ..., M$, ejecutar:

1. *Forward pass:* calcular la salida x^L de la red ante la entrada $x^1 = e^m$ utilizando las Ecs. 1 y 2. En el proceso, guardar los valores de activación x^l y de los correspondientes campos locales h^l obtenidos en las distintas capas $l = 2, ..., L$, ya que serán útiles más adelante.

2. Calcular el vector de errores D^L de la capa de salida utilizando la Ec. 5.

3. Propagar los errores hacia atrás, i.e. calcular los errores D^l para $l = L-1, L-2, ..., 1$ utilizando la Ec. 6.

4. Para cada l, i y j, calcular el gradiente $\frac{\partial F^m}{\partial w_{ij}^l}$ utilizando la Ec. 7 y actualizar el correspondiente peso sináptico w_{ij}^l utilizando la Ec. 4.

---

* juan.perotti@unc.edu.ar