I302 - Aprendizaje Automático y Aprendizaje Profundo

Roberto A. Bunge

Universidad de San Andrés

1 Backpropagation

1.1 Derivación de las ecuaciones de Backpropagation

Sea un conjunto de datos de entrenamiento $\mathcal{D} = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_N, y_N)\}$

Sea un modelo predictivo: $\hat{y} = \hat{y}(x, w)$, donde w son los parámetros del modelo que debemos ajustar.

Sea una función de costo (también llamada función de pérdida):

$$L(w) = \sum_{i \in \mathcal{D}} L_i(w)$$

Queremos encontrar los parámetros del modelo que minimizen la función de costo:

$$w^* = \min_w \ L(w) \tag{1}$$

Sea el modelo predictivo una red neuronal multi-capa densamente conectada, con funciones de actvacion no-lineales, como se muestra en la Figura 1.

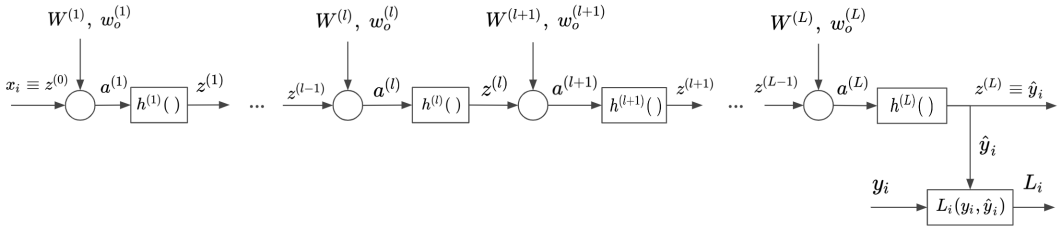

Figure 1: Representacion vectorial de un MLP.

Donde $a^{(l)}$ y $z^{(l)}$ son las señales de "pre-activacion" y de "salida" de la capa (l), respectivamente, y $W^{(l)}$ y $w_0^{(l)}$ son los parámetros (pesos) de la capa (l). $W^{(l)}$ es la "matriz de proyección" y $w_0^{(l)}$ es el vector de "bias" de la capa, definidos como:

$$W^{(l)} = \begin{bmatrix} w_{11}^{(l)} & \cdots & w_{1M^{(l-1)}}^{(l)} \\ \vdots & \ddots & \vdots \\ w_{M^{(l)}1}^{(l)} & \cdots & w_{M^{(l)}M^{(l-1)}}^{(l)} \end{bmatrix} \in \mathbb{R}^{M^{(l)} \times M^{(l-1)}}$$

$$w_0^{(l)} = \begin{bmatrix} w_{10}^{(l)} \\ \vdots \\ w_{M^{(l)}0}^{(l)} \end{bmatrix}, \quad a^{(l)} = \begin{bmatrix} a_1^{(l)} \\ \vdots \\ a_{M^{(l)}}^{(l)} \end{bmatrix}, \quad z^{(l)} = \begin{bmatrix} z_1^{(l)} \\ \vdots \\ z_{M^{(l)}}^{(l)} \end{bmatrix} \in \mathbb{R}^{M^{(l)} \times 1}$$

La relación entre las señales y pesos de una capa (l) y la siguiente capa $(l+1)$ está dada por:

$$a^{(l)} = W^{(l)} z^{(l-1)} + w_0^{(l)} \tag{2}$$

$$z^{(l)} = h^{(l)}(a^{(l)}) \tag{3}$$

$$a^{(l+1)} = W^{(l+1)} z^{(l)} + w_0^{(l+1)} \tag{4}$$

Esta arquitectura de modelo resulta en una funcion de costo que es función no-lineal y no-convexa con respecto a los parámetros del modelo (pesos de la red). Las nolinealidades son tales que no es posible encontrar un minimo de manera analítica, por lo cual solo queda encontrar un mínimo mediante algun algoritmo de optimización numérica. Por otra parte, dada la no-convexitud de la función objetivo, pueden existir mínimos locales que sean distintos del mínimo global.

Cualquier algoritmo de optimizacíon numérica de primer o segundo orden, necesitará el gradiente de la función objetivo (la funcion de costo) con respecto a las variables de optimización (los pesos de la red).

$$\nabla_w L(w) = \sum_{i \in D} \nabla_w L_i(w)$$

Nos concentraremos en computar $\nabla_w L_i(w)$. Para simplificar el desarrollo de las ecauaciones, usaremos la notacion $\frac{\partial f(x)}{\partial x} \equiv \nabla_x f(x)$, para referirnos al gradiente de una funcion vectorial $f(x)$ con respecto a un vector o matriz x.

Podemos computar el gradiente de $L_i(w)$ con respecto a los pesos $W^{(l)}$, $w_0^{(l)}$ de una capa l, aplicando la regla de la cadena:

$$\frac{\partial L_i}{\partial W^{(l)}} = \frac{\partial L_i}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial W^{(l)}} = \frac{\partial L_i}{\partial a^{(l)}} z^{(l-1)^T} \tag{5}$$

$$\frac{\partial L_i}{\partial w_0^{(l)}} = \frac{\partial L_i}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial w_0^{(l)}} = \frac{\partial L_i}{\partial a^{(l)}} I = \frac{\partial L_i}{\partial a^{(l)}} \tag{6}$$

$$\frac{\partial L_i}{\partial a^{(l)}} = \frac{\partial z^{(l)}}{\partial a^{(l)}} \frac{\partial L_i}{\partial z^{(l)}} = \text{diag}(h'^{(l)}(a^{(l)})) \frac{\partial L_i}{\partial z^{(l)}} \tag{7}$$

$$\frac{\partial L_i}{\partial z^{(l)}} = \frac{\partial a^{(l+1)}}{\partial z^{(l)}} \frac{\partial L_i}{\partial a^{(l+1)}} = W^{(l+1)^T} \frac{\partial L_i}{\partial a^{(l+1)}} \tag{8}$$

Definiendo $\delta^{(l)} \triangleq \frac{\partial L_i}{\partial a^{(l)}}$, que llamamos la señal de "error" de la capa l, y juntando las ecuaciones anteriores tenemos:

$$\delta^{(l)} = \text{diag}(h'^{(l)}(a^l)) W^{(l+1)^T} \delta^{l+1} \tag{9}$$

$$\frac{\partial L_i}{\partial W^{(l)}} = \delta^{(l)} z^{(l-1)^T} \tag{10}$$

$$\frac{\partial L_i}{\partial w_0^{(l)}} = \delta^{(l)} \tag{11}$$

Como se ve, surge una estructura recursiva, donde si tengo $\delta^{(l+1)}$, puedo computar $\delta^{(l)}$, y con ello los gradientes $\frac{\partial L_i}{\partial W^{(l)}}$ y $\frac{\partial L_i}{\partial w_0^{(l)}}$.

Para la capa de salida podemos computar $\delta^{(L)}$ volviendo a la definicion de $\delta^{(l)}$ y aplicando la regla de la cadena sobre L_i con respecto a $\hat{y}_i$ y $a^{(L)}$ (donde $\hat{y}_i = \hat{y}(x_i, w)$):

$$\delta^{(L)} \triangleq \frac{\partial L_i}{\partial a^{(L)}} = \frac{\partial \hat{y}_i}{\partial a^{(L)}} \frac{\partial L_i}{\partial \hat{y}_i} = \text{diag}(h'^{(L)}(a^L)) \frac{\partial L_i}{\partial \hat{y}_i} \tag{12}$$

Teniendo $\delta^{(L)}$, podemos computar $\frac{\partial L_i}{\partial W^{(L)}}$ y $\frac{\partial L_i}{\partial w_0^{(L)}}$ aplicando la formula de back-propagation:

$$\frac{\partial L_i}{\partial W^{(L)}} = \delta^{(L)} z^{(L-1)^T} \tag{13}$$

$$\frac{\partial L_i}{\partial w_0^{(L)}} = \delta^{(L)} \tag{14}$$

En la práctica, se puede hacer una mejora notando que la mutlplicacion matricial entre una matriz diagonal $diag(a)$ y vector columna b se puede computar mas eficientemente mediante el producto elemento a elemento de la diagonal y el vector columna:

$$diag(a)b = a \odot b$$

1.2 Algoritmo Backpropagation

Para computar la gradiente de L_i con respecto a los pesos de las distintas capas ocultas $W^{(l)}$ y $w_0^{(l)}$, debemos:

1. Para cada capa, computar la señal de pre-activacion $a^{(l)}$ y señal de salida $z^{(l)}$, yendo de la capa de entrada hasta la capa de salida. Esto se conoce como la fase de "forward-propagation" o "forward-pass".

2. Para cada capa, computar la señal de error $\delta^{(l)}$ y el gradiente de L_i con respecto a $W^{(l)}$ y $w_0^{(l)}$, yendo de la capa de salida, en dirección inversa, hasta la capa de entrada. Esto se conoce como la fase de "backward-propagation" o "backward-pass".

Algorithm 1 Algoritmo backpropagation

INPUTS: $x_i, W^{(:)}, w_o^{(:)}$
 # Forward-pass
$z^{(0)} = x_i$
for $l = 1$ **to** L **do**
  $a^{(l)} = W^{(l)} z^{(l-1)} + w_0^{(l)}$
  $z^{(l)} = h^{(l)}(a^{(l)})$
end for
$\hat{y}_i = z^{(L)}$
$L_i = L_i(y_i, \hat{y}_i)$

 # Backward-pass
$\delta^{(L)} = h'^{(L)}(a^l) \odot \frac{\partial L_i}{\partial \hat{y}_i}$
$\frac{\partial L_i}{\partial W^{(L)}} = \delta^{(L)} {z^{(L-1)}}^T$
for $l = L - 1$ **to** 1 **do**
  $\delta^{(l)} = h'^{(l)}(a^{(l)}) \odot {W^{(l+1)}}^T \delta^{(l+1)}$
  $\frac{\partial L_i}{\partial W^{(l)}} = \delta^{(l)} {z^{(l-1)}}^T$
  $\frac{\partial L_i}{\partial w_0^{(l)}} = \delta^{(l)}$
end for
OUTPUTS: $\hat{y}_i, L_i, \frac{\partial L_i}{\partial W^{(:)}}, \frac{\partial L_i}{\partial w_0^{(:)}}$