Tema 4: Redes Neuronales

Redes Neuronales 2

Prof. Wladimir Rodríguez

wladimir@ula.ve

Departamento de Computación

Entrenamiento de las Redes Neuronales: Propagación Hacia Atrás "Backpropagation"

A continuación se explicaran los conceptos de la función de costo logístico y el algoritmo de propagación hacia atrás "Backpropagation" que se implemantaron en la clase anterior para aprender los pesos de la Red Neuronal

Calcular la función de costo logístico

Debido a que se implementó el Perceptron Multicapa para realizar clasificación múltiple, la salida es un vector de t elementos usando la codificación "one-hot", por ejemplo la salida de la capa de salida para un ejemplo de la clase 2:

$$a^{(salida)} = \begin{bmatrix} 0.1 \\ 0.9 \\ \vdots \\ 0.3 \end{bmatrix}, \; y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Por lo que es necesario generalizar la función de costo logístico a todas las t unidades de activación de la red. Por lo que la función de costo es:

$$L(\boldsymbol{W}) = \frac{1}{n} \sum_{1}^{n} \sum_{j=1}^{t} (y_j^{[i]} - a_j^{(salida)[i]})^2$$

Recordar que nuestro objetivo es minimizar la función de costo $L(\boldsymbol{W})$; por lo tanto, se necesita calcular la derivada parcial de los parámetros $\boldsymbol{W}$ con respecto a cada peso para cada capa en la red:

$$\frac{\partial}{\partial w_{j,l}^{(l)}} L(\boldsymbol{W})$$

A continuación, se presenta el algoritmo de propagación hacia atrás "backpropagation", que permite calcular esas derivadas parciales para minimizar la función de costo.

Algoritmo de Propagación Hacia Atrás "Backpropagation"

En esencia, podemos pensar en la propagación hacia atrás "backpropagation" como un enfoque computacionalmente muy eficiente para calcular las derivadas parciales de una función de costo compleja en redes neuronales multicapa. Aquí, nuestro objetivo es usar esas derivadas para aprender los coeficientes de peso para parametrizar una red neuronal multicapa.

El desafío en la parametrización de redes neuronales es que se esta típicamente tratando con una gran cantidad de coeficientes de peso en un espacio de atributos de muchas dimensiones. En contraste con las funciones de costo de neuronales de una sola capa redes como Adaline o regresión logística, la superficie de error de una función de costo de una red neuronal no es convexa o lisa con respecto a los parámetros. Hay muchos baches en esta superficie de costo multi-dimensional (mínimos locales) que tenemos que superar para encontrar el mínimo de la función de costo.

La regla de la cadena es un enfoque para calcular la derivada de una función anidada compleja, como $f(g(x))$, de la siguiente manera:

$$\frac{d}{dx}[f(g(x))] = \frac{df}{dx} \cdot \frac{dg}{dx}$$

De manera similar, se puede usar la regla de cadena para una composición de función arbitrariamente larga. Por ejemplo, supongamos que tenemos cinco funciones diferentes, $f(x)$, $g(x)$, $h(x)$, $u(x)$ y $v(x)$, y que F sea la composición de la función: $F(x) = f(g(h(u(v(x)))))$. Aplicando la regla de cadena, podemos calcular la derivada de esta función de la siguiente manera:

$$\frac{dF}{dx} = \frac{d}{dx}F(x) = \frac{d}{dx}f(g(h(u(v(x))))) = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

Entrenando Redes Neuronales usando Propagación Hacia Atrás "Backpropagation"

En esta sección, explicaremos la matemática de propagación hacia atrás "backpropagation" para entender cómo se pueden aprender los pesos en una red neuronal de manera muy eficiente.

Anteriormente vimos como calcular el costo como la diferencia entre la activación de la capa de salida y y la etiqueta de clase objetivo. Ahora veremos como trabaja el algoritmo de propagación hacia atrás para actualizar los pesos en nuestro modelo de Perceptrón Multicapa.

Primero necesitamos aplicar la propagación hacia adelante para obtener la activación de la capa de salida, que formulamos de la siguiente manera:

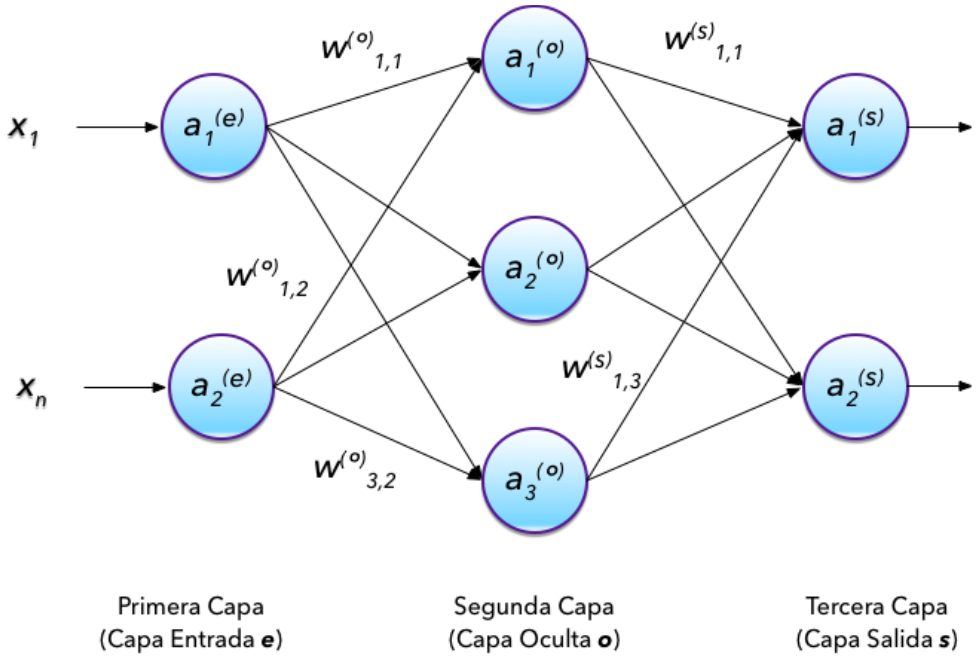
$$\boldsymbol{Z}^{(oculta)} = \boldsymbol{A}^{(entrada)}\boldsymbol{W}^{(oculta)T} \ \ (entrada\ a\ la\ capa\ oculta)$$

$$\boldsymbol{A}^{(oculta)} = \phi(\boldsymbol{Z}^{(oculta)}) \ \ (activación\ de\ la\ capa\ oculta)$$

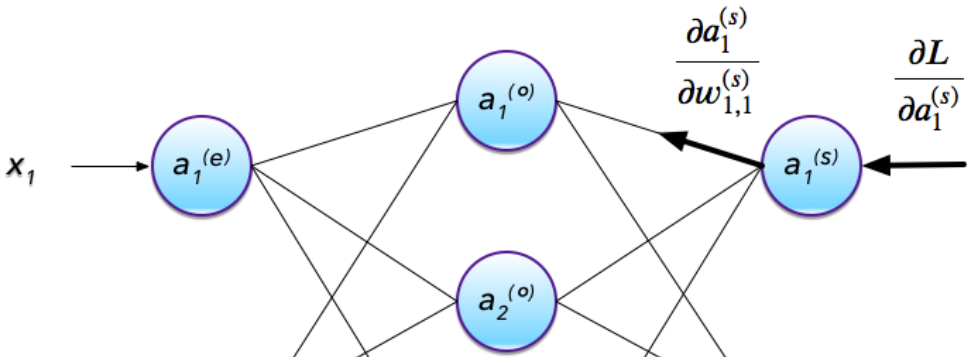
$$\boldsymbol{Z}^{(salida)} = \boldsymbol{A}^{(oculta)}\boldsymbol{W}^{(salida)T} \ \ (entrada\ a\ la\ capa\ de\ salida)$$

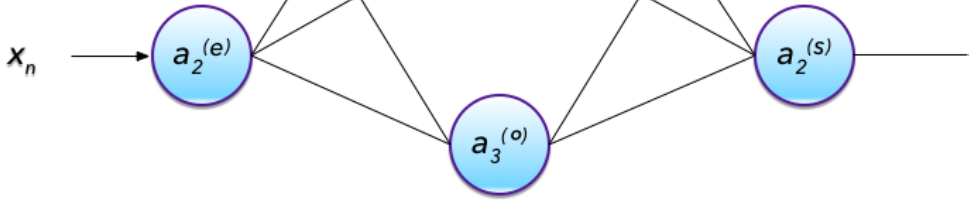
$$\boldsymbol{A}^{(salida)} = \phi(\boldsymbol{Z}^{(salida)}) \ \ (activación\ de\ la\ capa\ \ de\ salida)$$

La siguiente figura muestra este proceso:



En propagación hacia atrás, propagamos el error de derecha a izquierda. Podemos pensar en esto como una aplicación de la regla de la cadena al cálculo de la propagación hacia adelante para calcular el gradiente de la pérdida con respecto a los pesos del modelo (y las unidades de sesgo). Para simplificar, ilustraremos este proceso para la derivada parcial utilizada para actualizar el primer peso en la matriz de peso de la capa de salida. Las rutas del cálculo que propagamos hacia atrás se resaltan mediante las flechas en negrita a continuación:

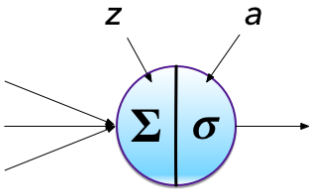




El gradiente para el peso $w_{1,1}^{(s)}$ de la capa de salida:

$$\frac{\partial L}{\partial w_{1,1}^{(s)}} = \frac{\partial L}{\partial a_1^{(s)}} \cdot \frac{\partial a_1^{(s)}}{\partial w_{1,1}^{(s)}}$$

Si incluimos la entrada z a la neurona:



$$\frac{\partial L}{\partial w_{1,1}^{(s)}} = \frac{\partial L}{\partial a_1^{(s)}} \cdot \frac{\partial a_1^{(s)}}{\partial z_1^{(s)}} \cdot \frac{\partial z_1^{(s)}}{\partial w_{1,1}^{(s)}}$$

Para calcular esta derivada parcial, comenzaremos con $\frac{\partial L}{\partial a_1^{(s)}}$, que es la derivada parcial de la función de perdida MSE.

$$\frac{\partial L}{\partial a_1^{(s)}} = \frac{\partial L}{\partial a_1^{(s)}}(y_1 - a_1^{(s)})^2 = 2(a_1^{(s)} - y)$$

El próximo término es la derivada de la función de activación logística que se usa en la capa de salida:

$$\frac{\partial a_1^{(s)}}{\partial z_1^{(s)}} = \frac{\partial}{\partial z_1^{(s)}} \frac{1}{1 + e^{z_1^{(s)}}} = \cdots = \left(\frac{1}{1 + e^{z_1^{(s)}}}\right)\left(1 - \frac{1}{1 + e^{z_1^{(s)}}}\right)$$

$$= a_1^{(s)}(1 - a_1^{(s)})$$

Por último, calculamos la derivada de la entrada con respecto al peso:

$$\frac{\partial z_1^{(s)}}{\partial w_{1,1}^{(s)}} = \frac{\partial}{\partial w_{1,1}^{(s)}} a_1^{(o)} w_{1,1}^{(s)} + b_1^{(s)} = a_1^{(o)}$$

Poniendo todo junto, obtenemos lo siguiente:

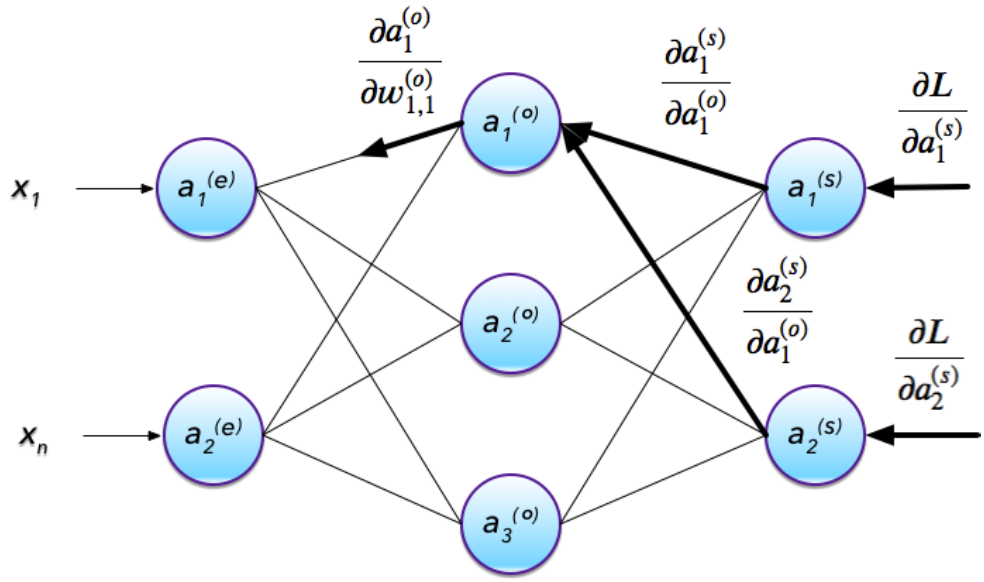
$$\frac{\partial L}{\partial w_{1,1}^{(s)}} = \frac{\partial L}{\partial a_1^{(s)}} \cdot \frac{\partial a_1^{(s)}}{\partial z_1^{(s)}} \cdot \frac{\partial z_1^{(s)}}{\partial w_{1,1}^{(s)}} = 2(a_1^{(s)} - y) \cdot a_1^{(s)}(1 - a_1^{(s)}) \cdot a_1^{(o)}$$

Luego usamos este valor para actualizar el peso a través de la conocida actualización de descenso de gradiente estocástico con una tasa de aprendizaje de η:

$$w_{1,1}^{(s)} = w_{1,1}^{(s)} - \eta \frac{\partial L}{\partial}$$

$$\frac{\partial L}{\partial w_{1,1}^{(o)}}$$

A continuacion se ilustra cómo calcular la derivada parcial de la pérdida con respecto al primer peso de la capa oculta:



Es importante resaltar que dado que el peso $w_{1,1}^{(o)}$ está conectado a ambos nodos de salida, tenemos que usar la regla de la cadena multivariable para sumar las dos rutas resaltadas con flechas en negrita.

$$\begin{aligned}\frac{\partial L}{\partial w_{1,1}^{(o)}} &= \frac{\partial L}{\partial a_1^{(s)}} \cdot \frac{\partial a_1^{(s)}}{\partial z_1^{(s)}} \cdot \frac{\partial z_1^{(s)}}{\partial a_1^{(o)}} \cdot \frac{\partial a_1^{(o)}}{\partial z_1^{(o)}} \cdot \frac{\partial z_1^{(o)}}{\partial w_{1,1}^{(o)}} \\ &+ \frac{\partial L}{\partial a_2^{(s)}} \cdot \frac{\partial a_2^{(s)}}{\partial z_2^{(s)}} \cdot \frac{\partial z_2^{(s)}}{\partial a_1^{(o)}} \cdot \frac{\partial a_1^{(o)}}{\partial z_1^{(o)}} \cdot \frac{\partial z_1^{(o)}}{\partial w_{1,1}^{(o)}}\end{aligned}$$