Tecnológico de Monterrey

Modelación Numérica de Sistemas Estocásticos

Daniel Otero Fadul

Departamento de Ciencia de Datos y Matemáticas
Escuela de Ingeniería y Ciencias

Redes Neuronales

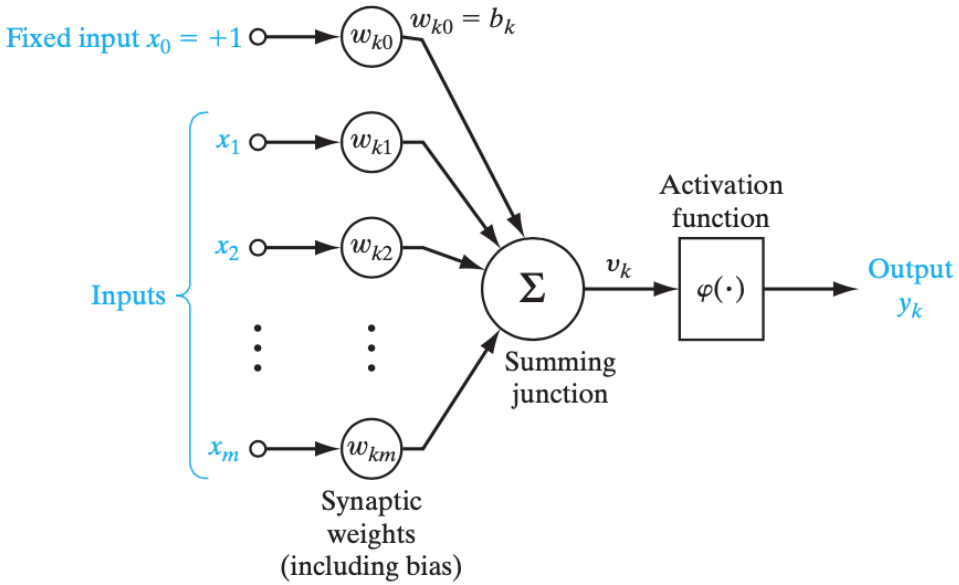

Figura: La neurona biológica se modela como una neurona artificial como la que se ve en la figura. Imagen tomada de [2].

Dado lo anterior, la expresión que relaciona las entradas y la salida de una neurona está dada por

$$\begin{aligned} y &= \phi\left(\sum_{i=0}^{n} w_i x_i\right) \\ &= \phi(w^T x), \end{aligned}$$

donde $w \in \mathbb{R}^{n+1}$ es un vector de pesos, $x_0 = 1$ y $\phi : \mathbb{R}^{n+1} \to \mathbb{R}$ es la **función de activación**.

Redes Neuronales

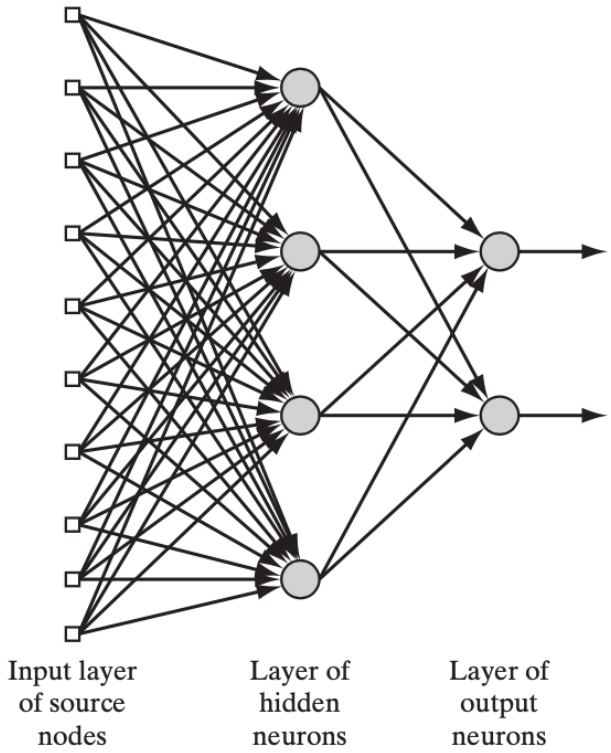

Figura: En la figura se puede ver un ejemplo de una red totalmente conectada que tiene una capa de entrada, una capa oculta y una capa de salida. Imagen tomada de [2].

Backpropagation

El entrenamiento de una red neuronal se realiza minimizando una función de costo. Si el problema que se quiere resolver es **clasificación múltiple**, la siguiente expresión es una función de costo apropiada, la cual se le conoce como **"categorical cross-entropy"** en inglés:

$$C(W) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}(y_k^{(i)}\log(h_W(x^{(i)}))_k + (1-y_k^{(i)})\log(1-h_W(x^{(i)}))_k),$$

donde K es el número de neuronas en la capa de salida.

Backpropagation

Si la aplicación es **regresión**, que es lo que aplica en nuestro caso, la función de costo que se debe minimizar es, normalmente, el **error cuadrático medio**:

$$C(W) = -\frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K}((h_W(x^{(i)})_k - y^{(i)})^2.$$

Backpropagation

Dicho lo anterior, el entrenamiento de una red neuronal consiste en una minimización de la función de costo por medio del método de gradiente descendente, junto con un algoritmo conocido como **"backpropagation"**, el cual es el algoritmo que se utiliza para realizar el cálculo del gradiente de la función de costo en cada iteración de gradiente descendente.

Backpropagation

En otras palabras, entrenar una red neuronal consiste en obtener un conjunto de parámetros óptimos que minimizan la función de costo de la diapositiva anterior, lo cual se realiza numéricamente utilizando el método de gradiente descendente. Como sabemos, es necesario calcular este gradiente para utilizar este método, lo cual es posible gracias al algoritmo de "backpropagation". En otras palabras, "backpropagation" nos permite calcular estas derivadas parciales:

$$\frac{\partial C(W)}{\partial w_{ij}^{(l)}},$$

es decir, la derivada parcial de la función de costo respecto al parámetro $w_{ij}^{(l)}$: el peso asociado a la entrada **j** de la neurona **i** de la capa **l+1**. Los pesos $w_{ij}^{(l)}$ de la capa l se almacenan en una matriz que denotamos como $W^{(l)}$.

Backpropagation

Sea $a^{(l)}$, $l = 2, \ldots, L$, un vector que contienen las **activaciones** de las neuronas de la capa l, con $a^{(1)} = x$. Sea $\delta^{(L)} = a^{(L)} - y$ el error entre la salida esperada y y lo entregado por la última capa de la red neuronal $a^{(L)}$. Definimos $\delta^{(l)}$, $l = 2, \ldots, L-1$, como

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) \circ a^{(l)} \circ (1 - a^{(l)}).$$

Nótese que $\circ$ es el producto matricial de Hadamard.

Backpropagation

Dado un conjunto de entrenamiento $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, el algoritmo de "backpropagation" es el siguiente:

inicializar $\Delta^{(l)} = 0,\ l = 2, 3, \ldots, L-1,\ i = 1$
repetir n veces
$a^{(1)} = x^{(i)}$
calcular $a^{(l)},\ l = 2, 3, \ldots, L$
$\delta^{(L)} = a^{(L)} - y^{(i)}$
calcular $\delta^{(l)},\ l = L-1, L-2, \ldots, 2$
calcular $\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T,\ l = 1, 2, \ldots, L-1$
$i = i + 1$
retornar $D^{(l)} = \frac{1}{n}\Delta^{(l)},\ l = 1, 2, \ldots, L-1$

En este caso,

$$\frac{\partial C(W)}{\partial w_{ij}^{(l)}} = D_{ij}^{(l)}.$$

Backpropagation

Teniendo en cuenta lo anterior, el entrenamiento de una red neuronal totalmente conectada se puede llevar a cabo ejecutando el siguiente algoritmo:

inicializar $W = W_0$, $\alpha \in (0, 1)$
repetir
    calcular $\nabla C(W)$ utilizando **backpropagation**
    $W = W - \alpha \nabla C(W)$
hasta cumplir con criterio de parada
retornar W

Backpropagation

La elección de la taza de aprendizaje α es un factor importante del entrenamiento: un α muy pequeño volvería el entrenamiento muy lento, mientras que un α muy grande podría hacer que el método de gradiente descendente no converja. En cualquier caso, el conjunto de parámetros obtenido durante la fase de aprendizaje no sería útil. Dicho esto, es conveniente escoger una taza de aprendizaje adecuada.

Backpropagation

Una alternativa para el ajuste de la taza de aprendizaje es implementar un algoritmo que permita que esta taza se adapte de manera adecuada en cada iteración del método de gradiente descendente:

inicializar $W = W_0$, $\alpha = \alpha_0 \in (0, 1)$
repetir
    calcular $\nabla C(W)$ utilizando **backpropagation**
    $W = W - \alpha \nabla C(W)$
    calcular α
hasta cumplir con criterio de parada
retornar W

BIBLIOGRAFÍA

1 Dangeti, P. *"Statistics for machine learning,"* Packt Publishing Ltd., 2017.

2 Haykin, S., *"Neural networks and learning machines"*, Pearson Education India, 2010.

3 Stuart R., Norvig P, *"Artificial intelligence: a modern approach,"* 2002.