Notas TFG

Juan

11 de marzo de 2017

Backpropagation

Entre las ANN's existentes, una de las más empleadas por su alta eficiencia es el Perceptrón Multicapa (MLP) mediante el uso del algoritmo de *Back-Propagation*. A diferencia del Perceptrón de una única capa, el MLP puede implementar gran variedad de funciones complejas, incluída la función XOR, que no puede ser realizada por el de una única capa como demostraron Minsky y Papertˆ{ojo} (Perceptrons: an introduction to computational geometry is a book written by Marvin Minsky and Seymour Papert and published in 1969).

En un MLP una unidad solo puede conectarse a la capa adyacente siguiente, no permitiéndose conexiones recurrentes ni en la misma capa. Sea K el número de capas, M el número de inputs y N el número de outputs. El input en una unidad (siempre que no sea en la capa de entrada) es la suma de los outputs de unidades conectadas en la capa anterior. Sea x_i^j el input de la unidad i en la capa j, w_{ij}^k el peso de la conexión entre la unidad i en la capa k y la unidad j de la capa $k+1$, y_i^j el output de la unidad i en la capa j y, por último, θ_i^j el umbral (o sesgo) de la unidad i en la capa j. Entonces tenemos que los input de la capa $k+1$ son:

$$x_j^{k+1} = \sum_i w_{ij}^k y_i^k$$

donde

$$y_i^j = f(x_i^j) \quad (1)$$

siendo f la función de activación. Nosotros utilizaremos como f la función sigmoide

$$f(x_i^j) = \frac{1}{1 + e^{-\frac{x_i^j - \theta_i^j}{T}}}.$$

Dependiendo de las aplicaciones un output que tome valores negativos es necesario, y podemos utilizar entonces la función

$$f(x_i^j) = \frac{1 - e^{-\frac{x_i^j - \theta_i^j}{T}}}{1 + e^{-\frac{x_i^j - \theta_i^j}{T}}}.$$

Hay dos fases en el algoritmo de *Back-Propagation*, la primera es computar el cálculo a través de las capas utilizando (1). La segunda fase es actualizar los pesos, operación que se realiza computando el error entre el valor esperado y el valor real calculado en la primera fase. Este proceso clasifica el algoritmo de *Back-Propagation* dentro de la categoría de los algoritmos de aprendizaje supervisado. Básicamente el algoritmo de *Back-Propagation* es un algoritmo de gradiente descendente.

A continuación explicaremos cómo se realiza la actualización de los pesos, una vez obtenido el output a través de (1). Primero definiremos una función de error $\mathcal{E}$ que nos dará cuenta de la discrepancia entre el valor calculado y el real. Sea N el número de outputs, d_i el output deseado y y_i el obtenido, con $i \in \{1, \ldots, N\}$. Entonces

$$\mathcal{E} = \frac{1}{2} \sum_j (d_j - y_j)^2, \quad (2)$$

El error total será simplemente $E_{total} = \sum_{k=1}^{N} E_k$. Nuestro objetivo será minimizar este error total, para ello utilizamos un algoritmo de gradiente descendente. Este tipo de algoritmo busca un mínimo en la función (la función error en nuestro caso) dando pasos proporcionales al negativo del gradiente. Es por ello que es tan importante que la función que usamos como función de activación sea derivable.

Así, el ajuste de los pesos es proporcional a la derivada

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \quad (2)$$

donde η es el tamaño del paso, importante para asegurar la convergencia. También se puede añadir un término de inercia, mediante una dependencia temporal, que nos ayudará a mejorar la convergencia evitando rápidos cambios en Δw_{ij}^k.

$$\Delta w_{ij}^k(t) = -\eta \frac{\partial E}{\partial w_{ij}^k(t)} + \alpha \Delta w_{ij}^k(t-1)$$

donde α es un positivo real pequeño. En el siguiente capítulo veremos cómo usar esta inercia en redes neuronales que usan modelos de Poisson en su generación de inputs.

Estrictamente hablando, la E que se utiliza en (2) debería ser E_{total}. Sin embargo es mucho más práctico actualizar pesos con cada input de una muestra de entrenamiento. En este caso puede utilizarse el E definido en (1).

Así, para cada capa de un perceptrón simple, tendríamos el output dado por

$$y_j = f(\sum_{i=1}^{n-1} w_i I_i + \theta_j) = f(s) \quad (3)$$

El sesgo θ_j puede añadirse como input siempre activo ($I_n = 1$) con peso $w_n = \theta_j$

$$y_j = f(\sum_{i=1}^{n} w_i I_i) = f(s) \quad (3)$$

Si el ouput deseado es d_j, entonces de (1) tenemos

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial w_i}$$

$$= -(d_j - y_j) f'(s) I_i$$

Sea

$$\delta_j = \eta(d_j - y_j)$$

Entonces el ajuste de pesos viene dado por

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \delta_j f'(s) I_i \quad (4)$$