REDES NEURONALES ARTIFICIALES

Lección 3

EL PERCEPTRÓN MULTICAPA

EducRNA

Entrenamiento – Algoritmo de retropropagación

El entrenamiento del perceptrón multicapa es, al igual que el perceptrón simple, de tipo supervisado, esto significa que necesitamos un conjunto de entradas de prueba para las cuales se conozca su salida, y así poder realizar las comparaciones con los resultados obtenidos de la red.

En este caso el algoritmo utilizado se le conoce como el algoritmo de retropropagación (Backpropagation en inglés), y este consiste en una serie de cálculos para el ajuste de los pesos y los umbrales de la red, con el fin de minimizar el error y acercarse lo más posible a la salida deseada.

Entrenamiento – Algoritmo de retropropagación

A continuación verán las formulas que se necesitan para el entrenamiento de un perceptrón multicapa.

Comenzamos con las funciones utilizadas para calcular tanto el error como el error cuadrático:

- Función error:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n) \qquad (3.8)$$

- Función error cuadrático:

$$e(n) = \frac{1}{n_C}\sum_{i=1}^{n_C} (s_i(n) - y_i(n))^2 \qquad (3.9)$$

EducRNA

Entrenamiento – Algoritmo de retropropagación

Aunque el aprendizaje busca minimizar el error total, el procedimiento mas común es uno basado en métodos de gradiente estocástico, los cuales minimizan los errores para cada entrada, e**(n)**, en vez de minimizar el error total **E**. Aplicando este método, cada peso **w** de la red se modifica para cada entrada **n** de acuerdo a la siguiente ley de aprendizaje:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} \qquad (3.10)$$

Donde e**(n)** es el error para la entrada **n**, y **α** es la tasa de aprendizaje.

El término retropropagación es utilizado debido a la forma de implementar este método de la gradiente, pues el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red.

EducRNA

Entrenamiento – Algoritmo de retropropagación

Para el desarrollo de esta regla es necesario distinguir dos casos: uno para los pesos de la capa oculta **C−1** a la capa de salida**(C)** y para los umbrales de las neuronas de salida, y otro para el resto de los pesos y umbrales de la red, pues las reglas de modificación de estos parámetros son diferentes.

Adaptación de pesos de la capa oculta **C-1** a la capa de salida y umbrales de la capa de salida:

Sea w_{ji}^{C-1} el peso de la conexión de la neurona j de la capa $C-1$ a la neurona i de la capa de salida. Utilizando el método de descenso del gradiente

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) - \alpha \frac{\partial e(n)}{\partial w_{ji}^{C-1}} \qquad (3.11)$$

EducRNA

Entrenamiento – Algoritmo de retropropagación

Después de realizar las derivadas parciales y todas las substituciones se obtiene la formula final para la adaptación del peso y umbrales:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha\delta_i^C(n)a_j^{C-1} \quad para\ j = 1, 2, ..., n_{C-1}\ i = 1, 2, ..., n_C \tag{3.16}$$

$$u_i^C(n) = u_i^C(n-1) + \alpha\delta_i^C(n)\ para\ i = 1, 2, ..., n_C \tag{3.17}$$

- Si la función de activación es la sigmoidal, la ecuación de **δ** seria:

$$\delta_i^C(n) = (s_i(n) - y_i(n))y_i(n)(1 - y_i(n))\ para\ i = 1, 2, ..., n_C \tag{3.31}$$

(**S_i (n)** es la salida deseada mientras que **Y_i (n)** es la salida obtenida de la entrada **n**)

- Para la función de tangente hiperbólica, la formula es la misma multiplicada por 2

EducRNA

Entrenamiento – Algoritmo de retropropagación

Adaptación de pesos de la capa oculta c a la capa oculta **C+1** y umbrales de la capa **C+1**:

Con el objetivo de que el desarrollo de la regla de aprendizaje para el resto de los pesos y umbrales de la red sea lo más claro posible, se elige un peso de la capa $C-2$ a la capa $C-1$. Sea w_{kj}^{C-1} el peso de la conexión de la neurona k de la capa $C-2$ a la neurona j de la capa $C-1$. Siguiendo el método de descenso del gradiente, la ley para actualizar dicho peso viene dada por:

$$w_{kj}^{C-2}(n) = w_{kj}^{C-2}(n-1) + \alpha \frac{\partial e(n)}{\partial w_{kj}^{C-2}} \qquad (3.18)$$

EducRNA

Entrenamiento – Algoritmo de retropropagación

Después de realizar las derivadas parciales y todas las substituciones se obtiene la formula final para la adaptación del peso y umbrales:

$$w_{kj}^{c}(n) = w_{kj}^{c}(n-1) + \alpha\delta_{j}^{c+1}(n)a_{k}^{c}(n)$$
$$para\ k = 1, 2, ..., n_c,\ j = 1, 2, ..., n_{c+1}\ y\ c = 1, 2, ..., C-2 \quad (3.26)$$

$$u_{j}^{c+1}(n) = u_{j}^{c+1}(n-1) + \alpha\delta_{j}^{c+1}(n)$$
$$para\ j = 1, 2, ..., n_{c+1}\ y\ c = 1, 2, ..., C-2 \quad (3.28)$$

- Si la función de activación es la sigmoidal, la ecuación de **δ** seria:

$$\delta_{j}^{c+1}(n) = a_{j}^{c}(n)(1 - a_{j}^{c}(n)) \sum_{i=1}^{n_{c+1}} \delta^{c+2}(n)w_{ji}^{c}$$
$$para\ j = 1, 2, ..., n_{c+1}\ y\ c = 1, 2, ..., C-2 \quad (3.32)$$

- Para la función de tangente hiperbólica, la formula es la misma multiplicada por 2

EducRNA

Entrenamiento – Algoritmo de retropropagación

Los pasos para aplicar las formulas son los siguientes:

1. Se inicializan aleatoriamente, y con valores cercanos a cero, los pesos y umbrales.

2. Se toma una entrada del conjunto de entradas de entrenamiento, y se propaga hacia la salida utilizando las ecuaciones correspondientes.

3. Se evalúa el error cuadrático con la ecuación *(3.9)*.

Entrenamiento – Algoritmo de retropropagación

4. Se evalúan los nuevos pesos, para lo cual se requiere:

   - ✓ Calcular los δ de todas las neuronas de la capa de salida con la ecuación *(3.14)*.
   - ✓ Calcular los δ de las neuronas de las demás capas empezando por la ultima oculta, con la ecuación *(3.27)*.
   - ✓ Modificar los pesos y umbrales con las ecuaciones *(3.16)* y *(3.17)* para la capa de salida y *(3.26)* y *(3.28)* para el resto de las capas.

Entrenamiento – Algoritmo de retropropagación

5. Se repiten los pasos 2, 3 y 4 para todos los valores del conjunto de entrenamiento, completando un ciclo de aprendizaje, también conocido como epoch.

6. Se evalúa el error total con la ecuación *(3.8)* cometido por la red.

7. Se repiten los pasos 2, 3, 4, 5 y 6 hasta tener un mínimo del error total.

El fin del entrenamiento es encontrar un conjunto de pesos y umbrales que determine un mínimo global en la función de error.

Early Stopping

Una manera de evitar el problema del sobre-entrenamiento es el early stopping.

- Se extrae un subconjunto del conjunto de entrenamiento, denominado conjunto de validación.
- Se entrena solamente en el conjunto de entrenamiento, mientras que el conjunto de validación es solo para calcular su error (error de validación).

Early Stopping

- La función de este conjunto es evaluar el error de validación de la red tras cada epoch (o cierto número de epochs), y determinar cuando este empieza a aumentar.
- Cuando el error de validación aumenta con respecto a la ultima vez que fue evaluado se detiene el entrenamiento.
- Se toman los valores del epoch anterior al aumento del error de validación.