Redes Neuronales Multilineales - Richard Duda - Capítulo 6

José Benjumeda Rubio

Febrero, 2020

1 Introducción

En éste capítulo veremos las redes neuronales multilineales, que llegan a dónde las redes neuronales sin capa oculta se quedan cortas.

Una red neuronal consiste en capas con neuronas, donde cada neurona es una función no lineal, que recibe una suma ponderada de las salidas de las neuronas de la capa anterior o del vector de entrada, y cuya salida es a su vez parte de la suma ponderada que será la entrada de las neuronas de la siguiente capa. La clave de estos modelos es que admiten unos algoritmos relativamente simples, en los que la función no lineal se puede aprender a partir del conjunto de datos.

Uno de los métodos más conocidos para entrenar una red neuronal multilineal es el backpropagation, que veremos en cierta profundidad, ya que es un método potente y útil a la vez que relativamente simple, y dará pie a entender otros métodos más complicados como modificaciones de éste.

Veremos también que cada problema tiene una arquitectura o topología de red neuronal multilineal propia, y aquí es donde entra en juego el conocimiento que tengamos sobre el dominio del problema: incluso las ideas más informales o heurísticas pueden incorporarse fácilmente a nuestro modelo, modificando el número de capas ocultas y el número de neuronas de cada una de ellas. La simplicidad del método de backpropagation permiten probar distintas alternativas, sin que esto suponga un gran trabajo.

Una de las mayores dificultades a la hora de utilizar redes neuronales es ajustar la complejidad de la red, es decir, la selección del modelo. Si la red tiene demasiadas capas ocultas con muchas neuronas, el modelo tendrá overfitting, mientras que si no tiene suficientes, generalizará demasiado (underfitting).

Por último, es importante recordar que el uso de redes neuronales no exime al diseñador de adquirir un conocimiento profundo del problema.

2 Feedforward operation and classification

Las primeras explicaciones se harán sobre un modelo sencillo para solucionar el problema de la XOR, que ya un modelo lineal no es capaz de conseguir. Utilizamos una red neuronal con dos neuronas en la capa de entrada, una en la capa de salida y dos en una capa oculta, que se conectan por pesos. El sesgo se incluye como una neurona más que se conecta a todas las demás excepto a las de la capa de entrada, que emite un valor constante y no tiene entradas.

El vector que representa el dato de entrada se introduce en la primera capa. Las neuronas de esta capa transmiten el valor de su entrada a su salida sin modificarlo, es decir, su función es la función identidad.

Las siguientes neuronas tienen todas la misma función:

$$f(x) = \begin{cases} 1 & \text{si } x > 0 \\ \\ -1 & \text{si } x < 0 \end{cases} \tag{1}$$

que recibe una combinación lineal de las salidas de las neuronas de la capa anterior y la neurona del sesgo. Llamaremos x_i a la salida de las neuronas de la primera capa, y_j a las de la capa oculta, y z_k a las de la capa de salida. Decimos además

$$x = (x_1, \ldots, x_{n1})$$
$$y = (y_1, \ldots, y_{n2})$$
$$z = (z_1, \ldots, z_{n3})$$

$$z_k(y) = f\Big(\sum_0^n w_{kj} y_j\Big) \tag{2}$$

Donde w_{kj} son los pesos que unen las neuronas de la capa oculta con la neurona k de la capa de salida. Además, sabemos que la salida de cada neurona de la capa oculta es

$$y_j(x) = f\Big(\sum_0^n w_{ji} x_i\Big) \tag{3}$$

Con w_{ji} los pesos que unen las neuronas de la capa de entrada con la neurona j de la capa oculta, análogamente al caso anterior. Sustituyendo obtenemos la salida de la red neuronal en función de la entrada:

$$z_k(x) = f\Big(\sum_0^n w_{kj} f\Big(\sum_0^n w_{ji} x_i\Big)\Big) \tag{4}$$

2.1 Expressive power of multilayer networks

El teorema de **Kolmogorov** demuestra que cualquier función continua g(x) cuyo dominio esté restringido a un hipercubo $I^n : I = [0,1], n \geq 2$ puede expresarse en la forma

$$g(x) = \sum_{j=1}^{2n+1} \Xi_j \Big(\sum_{i=1}^{d} \Psi_{ij}(x_i)\Big) \tag{5}$$

para funciones Ξ_j y Ψ_{ij} adecuadas.

Este teorema nos permite asegurar que **cualquier función continua puede expresarse en una red neuronal de 3 capas**, si ponemos $2n + 1$ neuronas ocultas, cada una con una función Ξ_j, y teniendo en cada una de las neuronas de la capa de entrada, que deberán ser $d(2n+1)$, una función Ψ_{ij}. En la última capa habría una única neurona que sumaría las salidas de todas las neuronas ocultas.

Otra prueba del poder de las redes neuronales es el **teorema de Fourier**, que dice que cualquier función continua puede aproximarse arbitrariamente cerca por una suma de funciones armónicas, posiblemente infinita.

Esto sería una red neuronal con la función identidad en las neuronas de la capa de entrada, una cantidad posiblemente muy grande de neuronas en la capa oculta, con dichas funciones armónicas, y una sola neurona en la capa de salida, con una función que sume las salidas de las funciones de la capa oculta.

Sabemos que un conjunto completo de funciones, por ejemplo los polinomios, **pueden representar cualquier función**, sin embargo, esto también puede conseguirse **utilizando una sola función**, siempre que utilicemos los parámetros adecuados. Esto es lo que queremos conseguir con las redes neuronales multilineales, cuyas neuronas siempre tendrán la misma función de transferencia.

Ninguno de los teoremas anteriores nos dan **ninguna pista sobre la cantidad de neuronas ocultas ni sobre los pesos correctos**. Además, aunque existiese, **una prueba constructiva nos sería de poca ayuda**, ya que normalmente no sabremos cómo es la función buscada. Sin embargo, estos resultados nos ayudan a pensar que vamos en la dirección correcta.

2.2 Backward propagation algorithm

Las redes neuronales tienen **dos modos de funcionamiento: feedforward y aprendizaje**. Feedforward consiste en introducir un vector en la capa de entrada para obtener un resultado por la capa de salida, mientras que aprendizaje (supervisado) consiste en introducir un vector en la capa de entrada pero conociendo el resultado correcto, y según sea el producido por la red, ajustar los pesos para que se parezcan lo máximo posible.

Sin embargo, **no sabemos cómo modificar los pesos de la capa de entrada a la oculta**, ya que no sabemos que salidas deberíamos estar obteniendo por las neuronas de esta capa. Éste es el **problema de asignación de crédito** (credit assignment problem). El algoritmo de **backpropagation** es una de las soluciones que hay.

2.3 Aprendizaje de la red

Queremos medir la **distancia entre cada predicción y el resultado real**, así que utilizamos el error cuadrático. Si fijamos el valor de entrada, el **error cuadrático** se puede ver como una **función de todos los pesos de la red**. La llamamos $J(\mathbf{w})$ (multiplicamos por $\frac{1}{2}$ para derivar más cómodamente):

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}(\mathbf{t} - \mathbf{z})^2 \tag{6}$$

siendo $\mathbf{t}$ y $\mathbf{z}$ son los vectores esperado (target) y obtenido, y $\mathbf{w}$ el vector de todos los pesos de la red (weight). Buscamos su mínimo.

El algoritmo de backpropagation consiste en intentar llegar a un **mínimo local** de $J(\mathbf{w})$ haciendo pequeñas modificaciones de $\mathbf{w}$ en la **dirección contraria a la del vector gradiente** en ese punto, que es aquella en la que más rápido decrece la imagen.

Las **modificaciones serán proporcionales** al error cuadrático, pues si este es cercano a cero, ya estamos muy cerca del mínimo (que como mucho, será 0). Representamos el incremento de $\mathbf{w}$ así:

$$\Delta\mathbf{w} = -\mu\frac{\partial J}{\partial \mathbf{w}} \tag{7}$$

o, componente a componente:

$$\Delta w_{mn} = -\mu\frac{\partial J}{\partial w_{mn}} \tag{8}$$

donde μ indica el tamaño del incremento.

Primero vamos a calcular el incremento para los pesos de entre la capa oculta y la de salida, y luego para los pesos de entre la capa de entrada y la oculta. Llamamos $\boldsymbol{net_k}$ a la entrada de la neurona k de la capa de salida, y $\boldsymbol{net_j}$ a la entrada de la neurona j de la capa oculta.

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} \tag{9}$$

y definimos la sensibilidad de la neurona k de la capa oculta como

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k)f'(net_k) \tag{10}$$

Luego, derivando respecto de w_{kj} en la ecuación

$$net_k = \sum_{j=1}^{n_h} y_j w_{kj} \tag{11}$$

obtenemos

$$\frac{\partial net_k}{\partial w_{kj}} = y_j \tag{12}$$

Finalmente, sustituimos en la ecuación del incremento de w_{kj}:

$$\Delta w_{kj} = -\mu \frac{\partial J}{\partial w_{kj}} = -\mu \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}}$$

$$\boxed{\Delta w_{kj} = \mu \delta_k y_j = \mu (t_k - z_k) f'(net_k) y_j} \tag{13}$$

Ahora hacemos lo mismo, pero un poco más complicado, para los pesos de la capa de entrada a la capa oculta:

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

Desarrollamos el primer término:

$$\frac{\partial J}{\partial y_j} = \left(\frac{\partial}{\partial y_j} \right) \left(\frac{1}{2} \sum_{k=1}^{c} (t_k - z_k)^2 \right) = -\sum_{k=1}^{c} (t_k - z_k) \frac{\partial z_k}{\partial y_j} =$$

$$= -\sum_{k=1}^{c} (t_k - z_k) \frac{\partial z_k}{\partial net_k} \frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c} (t_k - z_k) f'(net_k) \frac{\partial net_k}{\partial y_j}$$

y, por último, derivando respecto de y_j en la ecuación (15), obtenemos

$$\frac{\partial J}{\partial y_j} = -\sum_{k=1}^{c} (t_k - z_k) f'(net_k) w_{kj} \tag{14}$$

De manera análoga a como se hizo antes, definimos la sensibilidad de una neurona oculta como

$$\delta_j = f'(net_j) \sum_{k=1}^{c} w_{kj} \delta_k = f'(net_j) \sum_{k=1}^{c} w_{kj} (t_k - z_k) f'(net_k)$$

Además, derivando respecto de w_{ji} en la siguiente ecuación

$$net_j = \sum_{j=1}^{n_h} x_i w_{ji} \tag{15}$$

obtenemos

$$\frac{\partial net_j}{\partial w_{ji}} = x_i \tag{16}$$

Finalmente, haciendo todas estas sustituciones en la ecuación incremento de w_{ji}:

$$\Delta w_{ji} = -\mu \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$\boxed{\Delta w_{ji} = \mu \left(\sum_{k=1}^{c} (t_k - z_k) f'(net_k) w_{jk} \right) f'(net_j) x_i} \tag{17}$$

Y también se puede expresar respecto de δ_j previamente definido haciendo:

$$\Delta w_{ji} = \mu \left(\sum_{k=1}^{c} \delta_k w_{jk} \right) f'(net_j) x_i$$

$$\Delta w_{ji} = \mu \delta_j x_i$$