Sustentación Tópicos y Aprendizajes de Máquinas

Taller II

April 2, 2024

1 Red Neuronal de 3 Capas

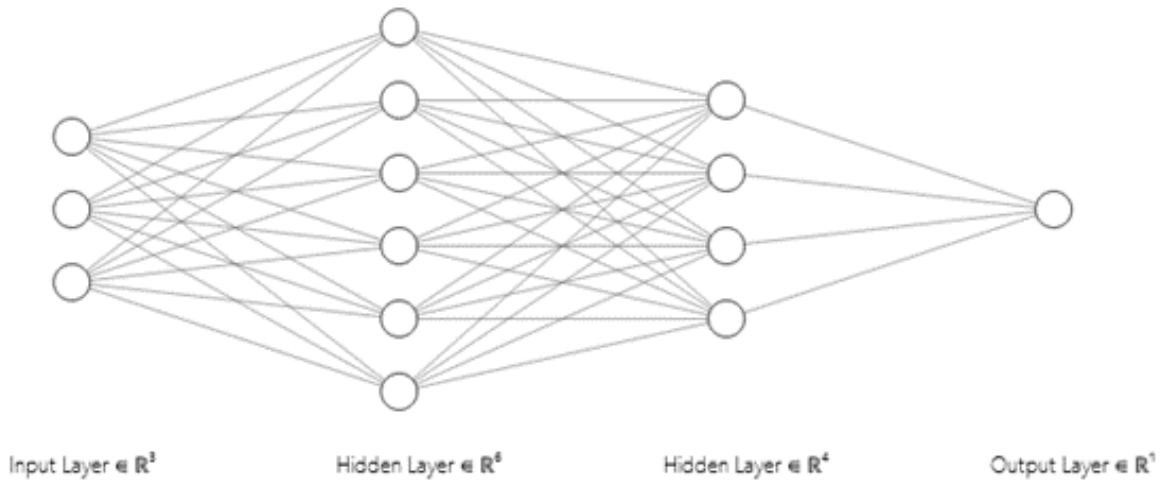

Figure 1: Red Neuronal de 3 Capas

1.1 Ecuaciones de Forward Propagation

A continuación usaremos la notación del curso de Deeplearnign de Andrew Ng, para las ecuaciones vectorizadas de Forward y Backward Propagation y asumimos que la ecuación de activación es la función sigmoid. Primero, denotemos el número de nodos de cada capa

- $n^{[0]} = 3$
- $n^{[1]} = 6$
- $n^{[2]} = 4$
- $n^{[3]} = 1$

Ahora bien, escribamos las ecuaciones para cada una de las tres capas:

$$\underset{(6,1)}{Z^{[1]}} = \underset{(6,3)}{w^{[1]}}\underset{(3,1)}{X} + \underset{(6,1)}{b^{[1]}}$$

$$\underset{(6,1)}{A^{[1]}} = \sigma(\underset{(6,1)}{Z^{[1]}})$$

$$\underset{(4,1)}{Z^{[2]}} = \underset{(4,6)}{w^{[2]}}\underset{(6,1)}{A^{[1]}} + \underset{(4,1)}{b^{[2]}}$$

$$\underset{(4,1)}{A^{[2]}} = \sigma(\underset{(4,1)}{Z^{[2]}})$$

$$\underset{(1,1)}{Z^{[3]}} = \underset{(1,4)}{w^{[3]}}\underset{(4,1)}{A^{[2]}} + \underset{(1,1)}{b^{[3]}}$$

$$\underset{(1,1)}{A^{[3]}} = \sigma(\underset{(1,1)}{Z^{[3]}}) = \hat{Y}$$

1.2 Ecuaciones de Backward Propagation

1.2.1 Tercera Capa

$$dZ^{[3]} = A^{[3]} - Y$$

$$dW^{[3]} = \frac{1}{m} dZ^{[3]} A^{[2]T}$$

$$db^{[3]} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[3](i)}$$

$$dA^{[2]} = W^{[3]T} dZ^{[3]}$$

1.2.2 Segunda Capa

$$dZ^{[2]} = dA^{[2]} * g'(Z^{[2]})$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[2](i)}$$

$$dA^{[1]} = W^{[2]T} dZ^{[2]}$$

1.2.3 Primera Capa

$$dZ^{[1]} = dA^{[1]} * g'(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^{T}$$

$$db^{[1]} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[1](i)}$$

1.3 Activaciones Lineales

Supongamos que tenemos una red neuronal con una capa de entrada X, una capa oculta y una capa de salida. Si eliminamos las funciones de activación no lineales y usamos solo funciones de activación lineales, la salida de cada capa sería simplemente una combinación lineal de las entradas. Esto se puede expresar de la siguiente manera, dado una capa oculta y una capa de salida:

1. **Capa Oculta:**
$$A^{[1]} = Z^{[1]} = W^{[1]}X + b^{[1]}$$
2. **Capa de Salida:**
$$A^{[2]} = Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

Donde $W^{[1]}$ y $W^{[2]}$ corresponden a las matrices de pesos, $b^{[1]}$ y $b^{[2]}$ son los vectores de sesgo, y $A^{[1]}$ y $A^{[2]}$ son las matrices de activación de la capa oculta y de salida respectivamente.

En este caso, la salida de la red neuronal $A^{[2]}$ sería simplemente una combinación lineal de las entradas X a través de las matrices de pesos $W^{[1]}$ y $W^{[2]}$, sin ninguna transformación no lineal entre capas.

Por lo tanto, incluso si agregamos múltiples capas ocultas a la red neuronal, la salida seguiría siendo una función lineal de las entradas, ya que la composición de funciones lineales es lineal, lo que limitaría la capacidad de la red para aprender y modelar relaciones complejas y no lineales en los datos.

1.4 Clasificación Múltiple

Para resolver un problema de clasificación múltiple, es necesario hacer las siguientes modificaciones a la red neuronal:

1. **Dimensiones de la capa de salidad:** Dadas k-clasificaciones la dimensión de la capa de salida debe ser un vector de dimensión $(k, 1)$. De igual forma, la codificación de este vector debe usar el estándar de one-hot para representar las etiquetas de clase, donde cada clase se representa de forma binaria con un valor de 1 en la posición correspondiente a la clase y 0 en las demás posiciones.

2. **Función de Activación:** La función sigmoid es usada principalmente para problemas de clasificación binaria. Sin embargo, no es la mejor para un problema de clasificación múltiple. Por ejemplo, la función softmax
$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$
normaliza los valores del vector de entrada para producir una distribución de probabilidad, donde la suma de todas las salidas es igual a 1.

3. **Función de Perdida:** Asimismo, al existir distintas entradas es necesaria modificar la función de perdida de entropia cruzada para que se ajuste a las k-clasificaciones
$$J(y, \hat{y}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)})$$
Esta función de pérdida penaliza las predicciones incorrectas más severamente, especialmente cuando la probabilidad asignada a la clase verdadera es baja. La pérdida total se calcula promediando las pérdidas de todos los ejemplos de entrenamiento.

De resto, se mantienen las dimensiones de las capas y otras convenciones de la red neuronal de clasificación binaria.

1.5 Problema de Regresión

Para resolver un problema de regresión, es necesario hacer varias modificaciones a la red neuronal:

1. **Función de Activación (Capa de Salida):** Asumiendo que estamos usando la función estándar sigmoid (que se encuentra acotada entre 0 y 1 ideal para la clasificación binaria) es necesario cambiar la función de activación a la función de identidad ($g(z) = z$).

2. **Función de Perdida:** En lugar de la función de entropia cruzada, que está diseñada para los problemas de clasificación binaria, ya que penaliza los errores acotados entre 0 y 1. Es mejor en este caso usar una función de perdida como el error cuadrático medio (MSE) o el error absoluto medio (MAE).

De resto, se mantienen las dimensiones de las capas y otras convenciones de la red neuronal de clasificación binaria.