TrejoArriaga_red_multicapa_ejercicio

April 25, 2024

1 Red neuronal multicapa

Una red neuronal multicapa tiene capas ocultas entre la entrada y la salida. A este tipo de red también se le conoce como perceptrón multicapa (MLP por sus siglas en inglés de multilayer perceptron)

En este ejercicio implementaremos una red neuronal multicapa usando únicamente numpy.

INSTRUCCIONES: Completa el código faltante.

1.0.1 Alumno: Rodrigo Gerardo Trejo Arriaga

[ ]:
```
# Importamos paquetes
import numpy as np

np.random.seed(42)
```

[ ]:
```
# Funciones necesarias
def sigmoid(x):
    """
    Función sigmoide
    """
    return 1/(1+np.exp(-x))
```

1.1 Definamos la arquitectura de la red

[ ]:
```
# TODO: Especifica el tamaño de las capas de acuerdo al diagrama.
N_input = 4
N_hidden = 3
N_output = 2
```

1.2 Definir los pesos

Recordemos que los pesos ahora los representamos con matrices y utilizaremos el producton de la siguiente forma:

$$h = XW$$

tal que las entradas son

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

y

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix}$$

recuerda que los renglones se relacionan con las entradas y las columnas con los nodos intermedios.

Como herramienta usaremos la función normal. Verifica la documentación oficial de la función.

> numpy.random.normal(loc=0.0, scale=1.0, size=None)

TODO: construye a contrinuación matrices con valores aleatorios para la matriz de pesos entre la entrada y la oculta y para la oculta y la salida.

[ ]:
```
mean = 0
stdev = 0.5

W_1 = np.random.normal(loc=mean, scale=stdev, size=(N_input, N_hidden))
W_2 = np.random.normal(loc=mean, scale=stdev, size=(N_hidden, N_output))
X = np.random.rand(1, N_input)
```

1.3 Calcular la salida

Recordemos de la lección que

$$h = XW$$

$$y = f(H)$$

[ ]:
```
H_1 = np.dot(X, W_1)
A_1 = sigmoid(H_1)

print("salida de la capa oculta: ", A_1)

H_2 = np.dot(A_1, W_2)
Y_gorrito = sigmoid(H_2)

print("salida de la red: ", Y_gorrito)
```

```
salida de la capa oculta:  [[0.71599158 0.48250233 0.44073576]]
salida de la red:  [[0.36523534 0.32052748]]
```

Como resultado debes de ver la salida de la red como un vector de dos elementos.

Agrega tus conclusiones:

1.4 Conclusión

Esta práctica nos ha proporcionado una comprensión detallada y práctica de cómo funcionan las redes neuronales multicapa. Al construir la red desde cero, inicializando los pesos con distribuciones normales y calculando las salidas mediante la propagación hacia adelante, hemos adquirido una apreciación más profunda de los mecanismos subyacentes que hacen que las redes neuronales sean herramientas poderosas para el aprendizaje automático.

Uno de los aprendizajes clave de esta práctica es la importancia de la vectorización en el cómputo de redes neuronales, permitiendo cálculos eficientes y escalables que son esenciales para trabajar con grandes volúmenes de datos. Además nos ha permitido construir una arquitectura de red neuronal desde cero, identificando con ello el numero de neruronas de entrada, ocultas y de salida, lo cual es crucial a la hora de construir un modelo de aprendizaje profundo, pues de no especificarlo de manera correcta existirán errores en la ejecución del código, o bien, caeremos en problemas de overfiting u underfiting