Sprachverarbeitung: Übung

SoSe 24

Janis Pagel
Department for Digital Humanities, University of Cologne

2024-07-02

For this exercise, you need to both submit manual calculations as well as Python code. Please submit two files in Ilias, one a PDF with your calculations and one a file containing your Python code (either Jupyter Notebook or Python script). You can also combine both files into a zip-archive and submit only the archive. You can either solve the calculations by hand on a sheet of paper, scan it and submit as a PDF file or use the capabilities to write mathematical equations of tools like MS Word / LibreOffice / LaTeX, etc. to write down your calculations digitally.

Exercise 1.

Given is a Neural Network with one hidden layer and the following weights (see also Figure 1):

$$W_1 = \begin{bmatrix} 0.8 & -0.7 & 0.4 & -0.3 \\ 0.3 & 0.1 & -0.3 & 0.1 \\ -0.2 & -0.3 & 0.5 & 0.9 \end{bmatrix} W_2 = \begin{bmatrix} 0.3 & 0.6 & 0.2 \end{bmatrix} \tag{1}$$

Given are two input vectors x_1 and x_2:

$$x_1 = \begin{bmatrix} -0.5 & 0.8 & 1 & 0.2 \end{bmatrix} x_2 = \begin{bmatrix} 0.5 & 0 & 1 & -1 \end{bmatrix} \tag{2}$$

Calculate the output of the neural network y for both input vectors by calculating the dot product of each input vector with the weights of each neuron of the hidden layer (W_1) and afterwards the dot product of the hidden layer values with the weights of W_2. Use the sigmoid function for both the activation function of the hidden layer and the function of the output layer.

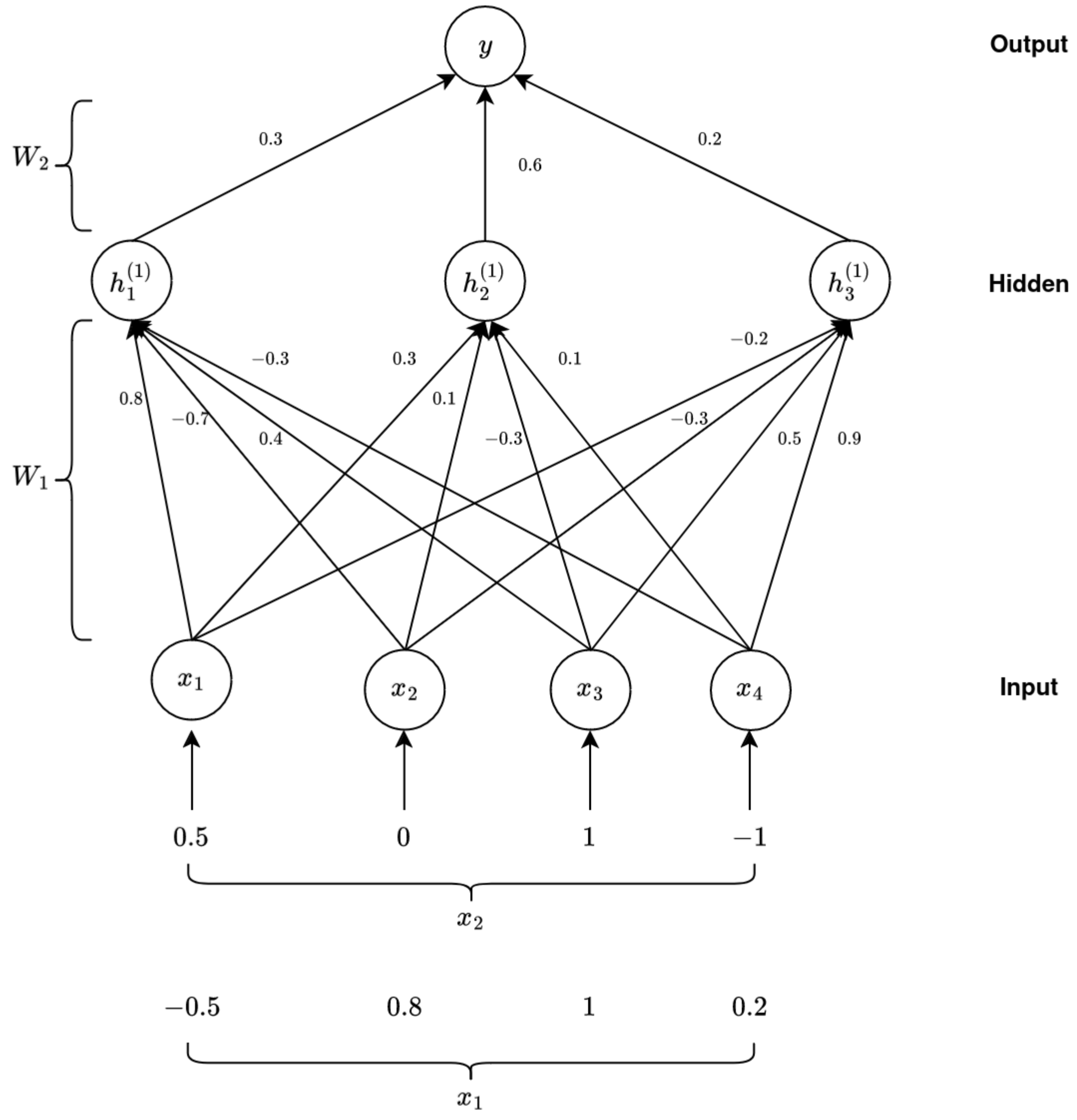

Figure 1: Neural Network

Solution 1.

$$\begin{aligned}
x_1 \cdot W_1^1 = [-0.5, 0.8, 1, 0.2] \cdot [0.8, -0.7, 0.4, -0.3] = \\
(-0.5 \times 0.8) + (0.8 \times -0.7) + (1 \times 0.4) + (0.2 \times -0.3) \approx -0.62 \\
h_1^1 = \frac{1}{1 + e^{0.62}} \approx 0.35 \\
x_1 \cdot W_1^2 = [-0.5, 0.8, 1, 0.2] \cdot [0.3, 0.1, -0.3, 0.1] = -0.35 \\
h_1^2 = \frac{1}{1 + e^{0.35}} \approx 0.41 \\
x_1 \cdot W_1^3 = [-0.5, 0.8, 1, 0.2] \cdot [-0.2, -0.3, 0.5, 0.9] = 0.54 \\
h_1^3 = \frac{1}{1 + e^{-0.54}} \approx 0.63 \\
h_1 \cdot W_2 = [0.35, 0.41, 0.63] \cdot [0.3, 0.6, 0.2] = 0.477 \\
y = \frac{1}{1 + e^{-0.477}} \approx 0.62
\end{aligned}$$

$$\begin{aligned}
x_2 \cdot W_1^1 = [0.5, 0, 1, -1] \cdot [0.8, -0.7, 0.4, -0.3] = \\
(0.5 \times 0.8) + (0 \times -0.7) + (1 \times 0.4) + (-1 \times -0.3) \approx 1.1 \\
h_1^1 = \frac{1}{1 + e^{-1.1}} \approx 0.75 \\
x_1 \cdot W_1^2 = [0.5, 0, 1, -1] \cdot [0.3, 0.1, -0.3, 0.1] = -0.25 \\
h_1^2 = \frac{1}{1 + e^{0.25}} \approx 0.44 \\
x_1 \cdot W_1^3 = [0.5, 0, 1, -1] \cdot [-0.2, -0.3, 0.5, 0.9] = -0.5 \\
h_1^3 = \frac{1}{1 + e^{0.5}} \approx 0.38 \\
h_1 \cdot W_2 = [0.75, 0.44, 0.38] \cdot [0.3, 0.6, 0.2] = 0.565 \\
y = \frac{1}{1 + e^{-0.565}} \approx 0.64
\end{aligned}$$

The probability the network outputs for x_2 (0.64) is slightly than for x_1 (0.62), hence the network predicts x_2 to be more close to the target class.

Exercise 2.

Implement a neural network with Keras and Tensorflow in Python using the `Sequential ()` object. You can decide on the number of hidden layers, the number of neurons in each layer, the activation functions and the number of epochs yourself. Use the data of apartment sales given on `https://lehre.idh.uni-koeln.de/site/assets/files/5151/apartments_sales.csv` and split it into 75% training and 25% test sets. Use the features “distance_to_city_center”, “rooms” and “size” to predict if the apartment has already been sold or not (column “sold”) Train your neural network on the train set and print the accuracy score for the test set.

Solution 2.

```
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

df = pd.read_csv("apartments_sales.csv")

X_train,X_test,y_train,y_test=train_test_split(df[["distance_to_city_center",
                                       "rooms", "size"]], df[["sold"]],
                                      random_state=42, test_size=0.25)

nn = Sequential()
nn.add(Dense(100, activation='sigmoid'))
nn.add(Dense(1, activation='sigmoid'))
nn.compile(loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'])
nn.fit(X_train, y_train, epochs=200, initial_epoch=0, verbose=0)
loss, accuracy = nn.evaluate(X_test, y_test, verbose=1)

print(f'Test Accuracy {round(accuracy*100,2)}')
```