CS 470/670 – Intro to Artificial Intelligence – Spring 2021

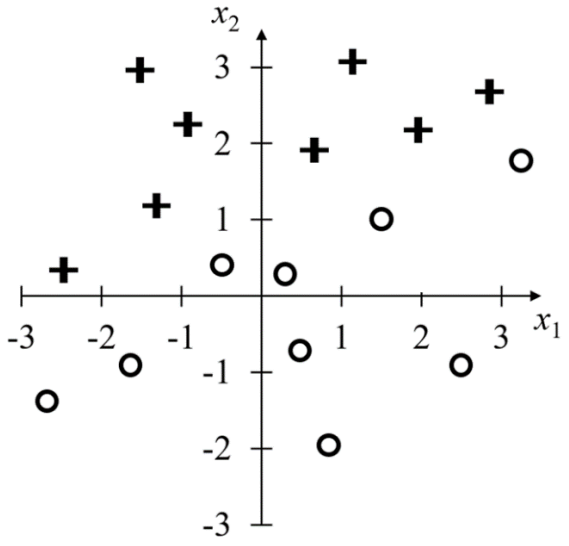
Instructor: Marc Pomplun

Assignment #5

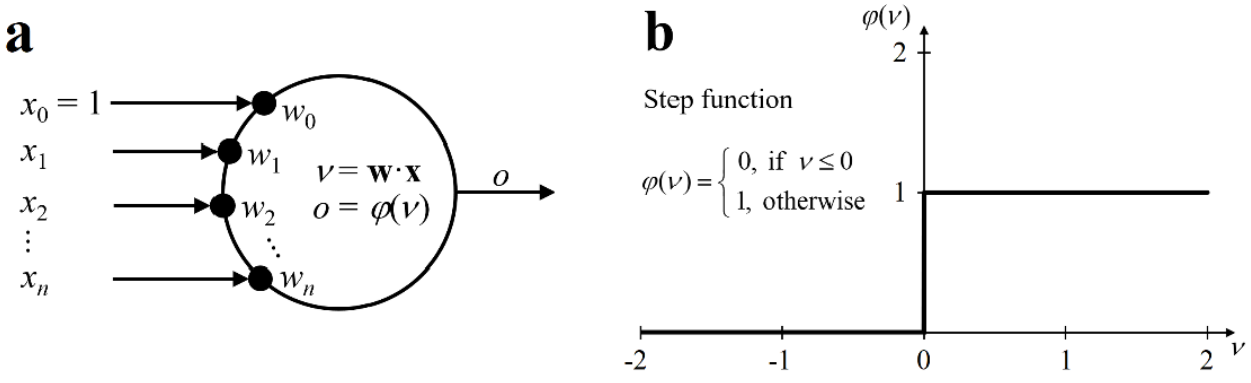
Posted on May 6, Due by May 11 at 4:00pm

Question 1: Classification with a Single Neuron

Let us imagine that you want to build a system that can determine whether someone is a computer scientist based on their ratings of caffeine dependence (variable x_1) and attraction to pizza (variable x_2). You obtain ratings for eight computer scientists (class 1 or "+") and nine non-computer scientists (class 0 or "o"). The results are shown below:



So, it seems that these two variables are good predictors of being a computer scientist. In fact, they separate the input space so nicely that you decide to build your system using only a single threshold (step function) neuron of the type shown in the book chapter:



The data to be classified are two-dimensional, which means that the neuron will receive inputs x_0, x_1, and x_2, where x_0 is the bias that is always set to 1, and x_1 and x_2 are the variables measured above. Consequently, the neuron has weights w_0, w_1, and w_2.

(a) Because this task is rather simple, you do not want to train the neuron but simply set the weights yourself so that the neuron outputs the value 1 if the input indicates a computer scientist and 0 otherwise. What values will you choose for weights w_0, w_1, and w_2? Explain your choices.

(b) While walking in downtown Boston, you meet someone and wonder if this person is a computer scientist. To find out, you ask them some questions to determine their values of x_1 and x_2. The result is $x_1 = 2$ and $x_2 = -1$. You enter these values into the mobile version of your system that uses the weights you found in (a). Write down the computation that your program will perform, and whether it determines the person to be a computer scientist or not.

(c) **Bonus Question:** Of course, this example is not realistic (or maybe it is?), but let us assume that you actually took the measurements as shown the first diagram for a real classification problem and that you randomly picked the exemplars from classes 0 and 1 from all the relevant objects, people, animals, etc., that you want to classify. Do you think that your neuron would perform perfectly on new inputs, i.e., always classify them correctly? Would there be some error? Or would it just perform at chance level, i.e., only give 50% correct answers? Give a reason for your answer.

Question 2: Predicting Soccer Data

The German soccer team FC Lederhosen currently has 22 players, all of whom are world class players and also proficient beer drinkers. Most remarkably, their team policy dictates that every player must be able to play as a goalkeeper and that they do not substitute players during a match, because it is a sign of weakness. This means that the coach selects any 11 out of the 22 players to participate in a given match.

Many players have strong personalities and get along very well with some players and not at all with others. This means that if certain groups of two or three players are on the field together, they always play together well and likely win the match, while other groups do not interact well and may lose the match. Similarly, the selection of players affects the total number of drinks that the players consume afterwards and also, through TV ratings and ticket sales, the money that the team earns (or loses) through the match.

The attached dataset soccer_data.pickle contains the relevant statistics from the past 5,000 matches (the club plays quite frequently). Please read the file like this:

```
import pickle
import numpy as np

with open('soccer_data.pickle', 'rb') as f:
    (x_train, y_train), (x_test, y_test) = pickle.load(f)
```

Conveniently, the data are now in the same form as the MNIST data after flattening their exemplars into pairs of row vectors (see MNIST_dense.py). The inputs are 22-element binary vectors. Each element represents one of the 22 players, with a 1 indicating that the player participated in a given match and a 0 meaning that he did not. The outputs are 4-element vectors indicating:

- The number of goals scored,
- The number of goals conceded,
- The total number of drinks that the team had after the match, and
- The money (in Euros) that the team gained (positive number) or lost (negative number) through the match.

The training and test sets include 4,000 and 1,000 exemplars, respectively.

(a) To help the coach with picking players, your task is to use Keras/Tensorflow to train an artificial neural network on the training data and assess its prediction accuracy for future matches using the test data. You can use any of the code that I uploaded as a starting point and modify it as necessary. This first approach should **only use an input layer and an output layer**, but no hidden layers.
Important: Note that this is not a classification but a value prediction task. This means that you should not use Softmax as your output function, because the outputs do not add up to one. Consequently, you should not use cross-entropy as your loss (error) function; a better choice would be the mean squared error (MSE) between the desired and actual outputs ('mse' in Keras). Finally, note that the y-values are not scaled identically and have very different ranges. Like in the weather prediction example, you need to scale these values before using them for training and then rescale them when you want to see the network's actual predictions for a given choice of players.

Run the network for a number of epochs that you choose. After training, determine the standard deviation (square root of MSE) of the network's predictions across all test exemplars for each of the four variables to get an idea of how well it works. Note that you need these predictions in the actual units and not your scaled ranges. For instance, we want to know the extent to which we can rely on the prediction of beer consumption and therefore need the liter values. If the network predicts a consumption of, for example, 50 liters and we know the standard deviation of the predictions from the actual values to be 2 liters, then we can (roughly) estimate that there is a 68% chance for the actual amount of beer to be between 48 and 52 liters, and a 95% chance that it will be between 46 and 54 liters. This will be very helpful for planning!

Note that the number of goals has to be a non-negative integer. Consequently, you could round the network's predictions to the closest integer before you compute the standard deviation.

Please upload this version of your code as "soccer_a.py". The number of epochs you used, the activation function, the final MSE for training and test set, and the standard deviation of the prediction for each variable can either be added to the code or reported in a separate file.

(b) The previous network used a very minimalistic approach. Please try to improve its predictive power. You can use any of the Keras and TensorFlow objects and functionalities that we discussed, and you can use others you know or find in the literature. Regardless, please describe the network structure that you choose for this task and explain why you chose it. Then report the results in the same way as you did in (a), and submit your code as "soccer_b.py". Please also add a comment on why you think your new code works better or does not work better.