ERHVERVSAKADEMI AARHUS

Machine Learning – Week 5
Ita. ML.

Exercise 1

See Playground TensorFlow exercises.

Exercise 2 - output layer.

Consider a neural network for classifying email spam.
How many neurons would you need in the output layer?
Consider a another network for classifying written digits (numbers 0 to 9 - the MNIST training set). How many neurons would you need in the output layer?
(and what would be reasonable to expect the values in the output layer to actually represent?)

Exercise 3 - activation functions.

Look in chapter 10 in the book. Find all the different activation functions used and draw a graph of them to get a feel for how they look. Try to think about disadvantages that could be an issue

Exercise 4 - weights and neural networks.

The weights on the connections between neurons in a neural network is usually initialized to (small) random numbers before starting to train.
Why do you think that is?

Exercise 5 - more hidden layers

It may seem counterintuitive that training a network with 2 hidden layers can actually in some cases be faster than with 1 hidden layer (the number of neurons in each layer though, is not the same). How can this be? If you don't know the answer, then (re)read page 349-50 (323-24 in v2, and 270-71 in v.1 of the book) in chapter 10. It is crucial you get this understanding.

Exercise 6.

Use a multilayer perceptron to classify petal length and width for the Iris data set.

Get inspiration in the following code, i.e. go though it, step by step. And adjust values etc. in order to understand the code. Try the code without feature scaling.

```
#This is a short example of using a MLP to predict data.


from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.ensemble import RandomForestClassifier

iris = load_iris()
#print(iris)

# setal length and width, so 2D information, only use the first two data rows
X = iris.data[:,(2,3)] # the :, indicates we want all row values for the two columns
(column 2 and 3)
y = iris.target  # these are the correct classifications
# check how many samples we have
#print(X)
print("Number of samples: " +str(len(y)))
#visulize the dataset
plt.figure()
#define colors - red, green, blue
colormap = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
# pplot labxel
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
# Plot the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=colormap,edgecolor='black', s=20)

plt.show()

# Split in train and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

# Feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp.fit(X_train, y_train)

#predictions
predictions = mlp.predict(X_test)

print(predictions)

import numpy as np

h = .02  # step size in the mesh

# create a mesh to plot in
x_min, x_max = X_test[:, 0].min() - 1, X_test[:, 0].max() + 1
```

```
y_min, y_max = X_test[:, 1].min() - 1, X_test[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

plt.subplot(1, 1, 1)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

Z = mlp.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

# Plot also the training points
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.title("MLP")

plt.show()
```

I.e. –
Questions:

A) What happens if you change the testsize ? Make it smaller, and larger. 5%, 90 %
B) Remove the scaler (StandardScaler). Can the classifier work without the scaler?
C) Change the number of iterations, epochs, max_iter, to 100. Was that a good idea?
D) Reset max_iter to 1000. Experiment with the number of hidden layers. Start with 1 hidden layer. How many hidden layers with how many nodes gives the best results?
Will mlp = MLPClassifier(hidden_layer_sizes=(2), max_iter=1000) work? Why not?

Exercise 7.

Use a multilayer perceptron to classify the “blobs-dataset”.
See the file Week6_Exercise7.py

Questions:

A) Adjust number of hidden layers, and number of iterations in order to have a proper classification.
B) Again, try without the scaler (?).
C) Experiment with the settings for blob-centers, and number of samples.