Lab 4 – Neural Networks

Question 1 – Exploring Neural Networks
Play around with Tensorflow Playground, https://playground.tensorflow.org

Question 2 – Multi-Layer Perceptron
Let us use neural networks on the breast cancer dataset, just like we did previously. Again, use stratified sampling.

1. Import `MLPClassifier` from `sklearn.neural_network`. Create an object with `random_state=0` and train a model. Compute the training and test scores.
2. MLPs work better with normalized data. Use `StandardScaler` as a preprocessing step on the data.
3. Train a new classifier and compare the new scores.
4. Train a new classifier with a larger value for the regularization parameter α in order to increase the penalties on the coefficients. For example, `alpha=1`.
5. One disadvantage of neural networks is that they are difficult to interpret. Still, on smaller networks, or on subsets of interest within a network, it is possible to observe the weights and try to get some insight. Use the code below.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,5))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='plasma')
plt.yticks(range(30), cancer.feature_names)
plt.xlabel("Neurons")
plt.ylabel("Features")
plt.colorbar()
```

6. There are many different parameters for MLPs. Use `GridSearchCV` or `RandomizeSearchCV` to explore different parameters and network structures.
7. Use pipelines and parameter tuning to perform model selection between a Random Forest model and an MLP.

Question 3 – Image Data
Neural networks are interesting to manipulate image date. Let us use the well-known MNIST handwritten digit dataset.

1. Scikit-Learn allows us to load data from openml.org. Here we will load a version of MNIST with smaller images (28x28 pixels, or 784 pixels).

```
from sklearn.datasets import fetch_openml
X, y = fetch_openml("mnist_784", return_X_y=True, as_frame=False)
```

2. We can view some of the images with the following code.

```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(1, 10, figsize=(20,2))
for image, ax in zip(X, axes.ravel()):
    ax.imshow(image.reshape(28, 28), cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())
plt.show()
```

3. Split the data into training and test sets. There are 70000 images. Usually 60000 are used for training and the rest for testing. However, training with that many images will take a lot of time, so set `train_size=6000`.
4. Train an MLP with 2 hidden layers, the first with 256 neurons, the second with 64 neurons. Evaluate its performance.
5. Since we are using image data, we can try to reconstruct what each neuron detects using information from the neural network (coefficients and intercepts). The code below allows us to observe the structures detected by the output neurons.

```
import numpy as np
images_n0 = mlp.coefs_[0]+mlp.intercepts_[0]
images_n1 = np.dot(images_n0, mlp.coefs_[1])+mlp.intercepts_[1]
images_n2 = np.dot(images_n1, mlp.coefs_[2])+mlp.intercepts_[2]

images = images_n2
fig, axes = plt.subplots(len(images.T)//10, 10, figsize=(20,20))
for image, ax in zip(images.T, axes.ravel()):
    ax.imshow(image.reshape(28, 28), cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())
plt.show()
```

6. Do we even need 2 hidden layers in the neural network? What would happen with no hidden layers?
7. Do we even need a neural network? What would happen with logistic regression?