Imperial College London

**Department of Mathematics**
**MATH96007 - MATH97019 - MATH97097**
**Methods for Data Science**
**Years 3/4/5**

# Coursework 2 – Neural networks and unsupervised learning

In this second coursework, you will work with two very different datasets: a collection of images and a small social network dataset. Task 1 deals with **supervised learning**, and you will perform a classification task using neural networks on a data set of images from an online fashion store. Task 2 deals with **unsupervised learning**, and you will perform tasks related to clustering, dimensionality reduction and graph-based analysis on the same data set of images and on a small social network.

You will find a .ipynb skeleton for your notebook with all the empty tasks on Blackboard.

## Task 1: Neural networks (50 marks)

**Dataset 1:** In Task 1, you will explore how two different neural network architectures perform on a supervised classification task on the **Fashion-MNIST** dataset of images. This data set consists of two files: a *training set* with 60,000 images and a *test set* with 10,000 images belonging to 10 classes of items sold by a fashion online store, e.g., T-shirt/top, Trouser, Pullover, Dress, etc. The Fashion-MNIST dataset is well balanced, i.e., it has 6,000 images of each class in the training set and 1,000 images of each class in the test set.

You can load this dataset by running

```
def load_data():
 (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
 x_train = x_train.astype('float32') / 255
 x_test = x_test.astype('float32') / 255

 # convert labels to categorical samples
 y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
 y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
 return ((x_train, y_train), (x_test, y_test))

(x_train, y_train), (x_test, y_test) = load_data()
```

### 1.1 Multi-layer perceptron (20 marks)

**1.1.1** - **Using NumPy alone** (i.e., *without using TensorFlow*), implement a multi-layer perceptron (i.e., feed-forward neural network) according to the following architecture description:

*Architecture of the network:* Your network should have an input layer, 5 hidden layers (each with 400 neurons), followed by the output layer with 10 neurons (one for each class). As your activation function between all layers, you should use the LeakyReLU(x) with a slope of 0.01, and you should use the softmax function as the activation function on the output layer. Fix the optimisation method to be stochastic gradient descent (SGD), and define the loss function as (categorical) cross-entropy.

Train the MLP on batches of 256 data points with a learning rate of $10^{-3}$ for 40 epochs. Plot the loss function and the accuracy of this model as a function of the number of epochs for both the training and test sets.

**1.1.2** Train the MLP as in 1.1.1 but now changing the learning rate to: (i) $10^{-5}$ and (ii) $10^{-1}$. As in 1.1.1, plot the loss and accuracies of the MLP as a function of the number of epochs for both the training and test sets, for these two values of the learning rates. Describe and explain the differences you observe in terms of the convergence of the loss and the performance of the models trained in 1.1.1 and 1.1.2.

**1.1.3** Train the neural network defined in 1.1.1 for 80 epochs and compare your performance (loss and accuracy on both the training and test sets) to the one achieved by only training for 40 epochs. Explain your result in terms of how the number of epochs influences the loss and accuracy for the training and test sets.

## 1.2 Convolutional neural network (CNN) (30 marks)

**1.2.1 Using TensorFlow**, implement a convolutional neural network (CNN) according to the following architecture description:

*Architecture of the network:* Your network should have an input layer, 5 hidden layers in total (of which the first 4 are convolutional layers and the last is a fully-connected layer), followed by the output layer with 10 neurons (one for each class). Regarding the hidden layers: all convolutional layers apply $3 \times 3$ filters, but the first two use 8 feature maps and the last two use 16 feature maps (also called 'channels'). The last convolutional layer is followed by a $2 \times 2$ maximum pooling layer. The fully-connected layer has 64 neurons, and is followed by the output layer with 10 neurons. You should again use the LeakyReLU(x) with a slope of 0.01 for $x < 0$ as your activation function between all layers, and the softmax function as the activation function on the output layer. Fix the optimisation method to be stochastic gradient descent (SGD), and define the loss function as (categorical) cross-entropy.

Train this model on batches of 256 data points with a learning rate of $10^{-3}$ for 40 epochs. Plot the loss function and the accuracy of this model as a function of the number of epochs, both on the training and test set. Discuss the convergence and the accuracy of the model, and how they compare to the MLP model from 1.1.1.

**1.2.2** Incorporate Dropout *in the fully connected layer*. Use now only 80% of the training set for the actual training and leave the other 20% as a validation set. Scan over a suitable range of the dropout probability (range [0.1, 0.9] in steps of 0.1) to find an optimal value of this dropout probability, using accuracy on the validation set as the measure of performance for this search. Fix the optimal dropout, and retrain the model on the full training set. Evaluate the loss and accuracy over epochs for both the training and test sets, and compare them to the model you defined in Task 1.2.1. Explain how the dropout regularisation affects the training procedure, the model performance (accuracy), and the features learnt by the fully connected layer (for the latter, plot the activations of the hidden units of the fully connected layer when the model is evaluated on the test set).

**1.2.3** Compare the results obtained in 1.1.1 (MLP) to the results obtained in 1.2.2 (CNN with Dropout) in terms of accuracy relative to the number of parameters in the models. Explain the observed differences.

**1.2.4** Choose one image from the test set. Using imshow() or otherwise, visualise the image and one feature map of your choice from each convolutional layer of the architecture in 1.2.2 and comment on the features extracted by each of the layers.

## Task 2: Unsupervised learning (50 marks)

### 2.1 Dimensionality reduction and clustering of a subset of the Fashion-MNIST image data (20 marks)

Consider the first N=1000 images found in the Fashion-MNIST data set used in Task 1.

**2.1.1 Using only NumPy/SciPy** (and based on the code developed in your coding tasks), normalize and centre the data, and perform PCA to carry out dimensionality reduction on this subset of *N=1000* images of dimensionality *p=784*

**2.1.2** Compute the top *m=25* principal components, and plot the corresponding eigenvalues and the fraction of variance explained as *m* is increased. Based on these plots, comment on what could be an optimal value for the reduced dimension *m* according to PCA.

**2.1.3** Consider the top *m=2* principal components and produce a plot of the *N=1000* data in the PCA space spanned by these *m=2* components, coloring each point according to their class image. Discuss your results.

**2.1.4** As in 2.1.3, consider the data points as described by the top *m=2* principal components of PCA. Apply *k*-means to cluster the points, varying *k* in the *k*-means algorithm between 2 and 10. Is the optimal *k* equal to the true number of classes (10)? Discuss your results and explain how you chose the optimal *k*.

**Dataset 2:** In this Task 2, you will also work with another data set from a social network of bottlenose dolphins. This data set was originally collected in a study of the social behaviour and interactions of a group of 62 bottlenose dolphins observed by marine biologists in New Zealand over several years.

You can read more about this study on the *paper by Lusseau et al,* which we have uploaded to Blackboard.

While there are no clearly identified cliques in this social network, three groups tended to spend more time together than the others. However, note *that in this task we are **not** necessarily trying to find these three groups but rather study properties of the data in an unsupervised way.*

Each row in the dataset corresponds to one of the *N*=62 bottlenose dolphins in the studied network. We have three sources of information:

1. **A set of features** characterising each individual. This information is given as a *N x p* feature matrix *F*, where *N*=62 samples with *p*=32 features capturing different traits of the dolphins.
2. **The social network of associations** between the individual dolphins. The *N*=62 nodes of the graph are the individuals and the *E*=159 edges correspond to the frequent associations between them. This information is given as a *N x N* adjacency matrix, *A*.
3. **A table listing the dolphin names** ordered according to the rows of the feature and adjacency matrices. You may use these names in analysing or presenting some of your findings.

The feature matrix *F*, the adjacency matrix *A* of the graph, and the table with the names of the dolphins are all available on Blackboard.

### 2.2 Clustering of the feature matrix (15 marks)

In this subtask, we will analyse the feature matrix *F* containing the characteristics of the dolphins.

**2.2.1 Using only NumPy/SciPy** (and based on the code developed in your coding tasks), employ hierarchical clustering to cluster the feature matrix *F.* Use average linkage and take Euclidean distance as your distance. Report your sequence of clusters from finest to coarsest.

**2.2.2 Using only NumPy/SciPy,** code a function to calculate the Silhouette Score of a given clustering (**you are NOT allowed to use sklearn**). The Silhouette Score is a measure of the quality of clustering

and is defined as the mean Silhouette Coefficient over all samples, where the Silhouette Coefficient of a given sample is given by:

$$\text{Silhouette Coefficient} = \frac{(d_2 - d_1)}{max(d_1, d_2)}$$

where $d_1$ is the average distance between the sample and the points in its cluster, and $d_2$ is the average distance between the sample and the points in the nearest cluster.

Use your function to compute the Silhouette Score for all the clusterings obtained at all the levels of hierarchical clustering, from finest to coarsest. Determine what level is optimal (i.e., which one maximizes the Silhouette Score).

**2.3 Graph-based analysis (15 marks)**:

In this subtask, we will analyse the dolphin social network, i.e. the graph of frequent associations encoded by the adjacency matrix *A*.

**2.3.1 Spectral clustering: Using only NumPy/SciPy**, compute the normalized Laplacian of the dolphin network, and its two smallest eigenvalues (and corresponding eigenvectors). Comment on these values and use them to obtain a spectral partition of the graph into two sets of nodes. Visualize your results on the network using the draw() function from **NetworkX**.

**2.3.2 Centralities: Using only NumPy/SciPy,** produce code to obtain three measures of centrality: PageRank, degree centrality, and eigenvector centrality. Apply them to the dolphin graph and report the values of the three centralities for all the nodes in the graph. Study which nodes (if any) are highly central according to the three centralities. Using appropriate correlation plots (or otherwise), discuss the similarity between the node rankings according to the different centrality measures and explain why the centrality rankings might differ. Use the file provided on Blackboard to identify the name of some of the most central dolphins, and discuss your centrality results with respect to the groups found by the marine biologists in the experiment reported in the original paper by Lusseau *et al.*

## Task 3: Mastery component *(for MSc and MSci students only)* - (25 marks)

**3.1 Non-Negative Matrix Factorization (NMF) for dimensionality reduction (15 marks):**

Consider the *same N=1000* images from the Fashion-MNIST dataset that you used in Task 2.1.

**3.1.1 Using only NumPy/SciPy** (and based on the code developed in your coding tasks), perform dimensionality reduction using Non-Negative Matrix Factorization (**you are NOT allowed to use sklearn**). Find the optimal NMF using 500 iterations and *m=10* dimensions onto which to project the data. Plot the convergence of your algorithm as a function of the iterations.

**3.1.2 Visualisation of components:** Write the formula to express the image data as a linear combination of non-negative components and visualize the *m=10* components using imshow (as in your notebooks). Since these are the non-negative equivalent of the principal components from PCA, visualise using imshow the first *m=10* principal components from PCA (obtained in Task 2.1). Discuss the comparison between NMF and PCA components and explain the reasons why they differ. Comment also on any differences with the feature maps visualised in Task 1.2.4.

**3.2 Community detection (10 marks):**

For this subtask **you are allowed to use NetworkX**. Ensure that you have NetworkX version v2.7 (or later).

**3.2.1. Modularity maximisation with Louvain algorithm:** Using the Louvain algorithm to maximise modularity in the NetworkX package, find the optimal community structure for the dolphin graph. Use NetworkX to plot the obtained clusters on the graph by assigning different colours to the nodes in each community. Discuss any connections of the optimal graph clusters with the most central nodes obtained in Task 2.3.2.

**3.2.2. Compare clusterings:** Use the Adjusted Rand Index (ARI) function **from sklearn** to quantify how similar the graph-based node clustering obtained in 3.2.1 is to the feature-based node clusterings obtained in 2.2. Discuss your results in terms of the *homophily principle in social networks.*