Imperial College London

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

Coursework 2 – Neural networks and unsupervised learning

Submission deadline: Friday, 26 March 2021, 5 pm

The goal of this coursework is to analyse two datasets using several of the tools and algorithms introduced in the lectures, which you have also studied in detail through the computational tasks in the course.

You will solve the tasks in this coursework using Python. You are allowed to use Python code that you will have developed in your coding tasks, but not ready-made solutions you can find on the Internet. You are also allowed to use any other basic mathematical functions contained in numpy. However, importantly, you are **not** allowed to use any model-level Python packages like sklearn, statsmodels, or similar, for your solutions unless we explicitly state it.

Submission

Your coursework will be presented as a **Jupyter notebook (file format: ipynb)** with all your tasks clearly labelled. The notebook should contain the cells with your code and their output, and some brief text explaining your calculations, choices, mathematical reasoning, and explanation of results. The notebook must be run and outputs of cells printed. You may produce your notebook with Google Colab, but we recommend to develop your local Jupyter notebook through the Anaconda environment (or any local Python environment) installed on your computer.

Once you have executed all cells in your notebook and their outputs are printed, also save the notebook as an **html file**, which you will also submit.

Submission instructions

The submission will be done **online via Turnitin on Blackboard**.

The deadline is **Friday, 26 March 2021 at 5 pm**.

*You will upload **two documents** to Blackboard, wrapped into a **single zip file**:*

1) *Your Jupyter notebook as an **ipynb file**.*
2) *Your notebook exported as an **html file**.*

You are also required to comply with these specific requirements:

- *Name your **zip file** as ‘SurnameCID.zip’, e.g. Smith1234567.zip. Do not submit multiple files.*
- *Your ipynb file must produce all plots that appear in your html file, i.e., make sure you have run all cells in the notebook before exporting the html.*
- *Use clear headings in your notebook to indicate the answers to each question, e.g. ‘Task 1.1’.*

Please pay attention to the following note about online submissions:

There are known issues with particular browsers (or settings with cookies or popup blockers) when submitting to Turnitin. If the submission 'hangs', please try another browser.

You should also check that your files are not empty or corrupted after submission.
To avoid last minute problems with your online submission, we recommend that you upload versions of your coursework early, before the deadline. You will be able to update your coursework until the deadline, but having these early versions provides you with some safety back up.

Needless to say, projects must be your own work: *You may discuss the analysis with your colleagues but the code, writing, figures and analysis must be your own and you should not use ready-made code you find on the Internet . The Department may use code profiling and tools such as Turnitin to check for plagiarism, as plagiarism cannot be tolerated.*

Marks

The coursework is worth **60% of your total mark for the course.**
This coursework contains a ***mastery component*** for MSc and 4th year MSci students.

Guidance about solutions and marking scheme:

Coursework tasks are different from exams: sometimes they can be more open-ended and may require going beyond what we have covered explicitly in lectures. In some parts of the tasks, initiative and creativity will be important, as is the ability to pull together the mathematical content of the course, drawing links between subjects and methods, and backing up your analysis with relevant computations that you will need to justify.

To gain the marks for each of the Tasks you are required to:

(1) complete the task as described;
(2) comment any code so that we can understand each step;
(3) provide a brief written introduction to the task explaining what you did and why you did it;
(4) provide appropriate, relevant, clearly labelled figures documenting and summarising your findings;
(5) provide an explanation of your findings in mathematical terms based on your own computations and analysis and linking the outcomes to concepts presented in class or in the literature;
(6) consider summarising your results of different methods and options with a judicious use of summary tables.

The quality of presentation and communication is very important, so use good combinations of tables and figures to present your results, as needed.

Explanation and understanding of the mathematical concepts are crucial.

Competent Python code is expected. As stated above, you are allowed to use your own code and the code developed in the coding tasks in the course, but you are not allowed to use Python packages like sklearn, statsmodels, etc unless explicitly stated or ready-made code found on the Internet.

Marks will be reserved and allocated for: presentation; quality of code; clarity of arguments; explanation of choices made and alternatives considered; mathematical interpretation of the results obtained; as well as additional relevant work that shows initiative and understanding beyond the task stated in the coursework.

Note that the mere addition of extra calculations (or ready-made 'pipelines') that are unrelated to the task without a clear explanation and justification of your rationale will not be beneficial in itself and, in fact, can also be detrimental if it reveals lack of understanding.

Overview

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 the dataset of images. Task 2 deals with **unsupervised learning**, and you will perform tasks related to clustering, dimensionality reduction and graph-based analysis on the social network data.

Task 1: Neural networks (40 marks)

In Task 1, you will explore how two different neural network architectures perform on a supervised classification task on the CIFAR-10 dataset of images. This dataset contains 50,000 training and 10,000 validation images belonging to 10 classes, e.g., airplanes, birds, cars, ships, etc The CIFAR-10 dataset is well balanced, i.e., it has 5,000 images of each class in the training set and 1,000 images of each class in the validation set.

You can load this dataset by running

```
def load_data():
 (x_train, y_train), (x_val, y_val) = tf.keras.datasets.cifar10.load_data()
 x_train = x_train.astype('float32') / 255
 x_val = x_val.astype('float32') / 255

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

(x_train, y_train), (x_val, y_val) = load_data()
```

1.1 Multi-layer perceptron (20 marks)

1.1.1 - **Using NumPy alone** (*without 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 five hidden layers, each with 400 neurons, followed by the output layer with 10 neurons. You should use the standardised *tanh(x)* function 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 cross-entropy.

Train the MLP on batches of 128 data points with a learning rate of 0.01 for 40 epochs. For both the training and validation sets, plot the loss function and the accuracy of this model as a function of the number of epochs. Discuss the convergence of the model.

1.1.2 - Train the MLP as in 1.1.1 but now changing the learning rate from 0.01 to: (i) 0.0001 and (ii) 0.1. As in 1.1.1, show the loss and accuracies of the MLP for these values of the learning rates. lDescribe and discuss the differences you observe in terms of the convergence of the loss and the performance of the models.

1.1.3 - Train the neural network defined in 1.1.1 for 80 epochs and compare your performance to the ones achieved by only training for 40 epochs in 1.1.1 and 1.1.2. Explain your results and the comparisons with 1.1.1 and 1.1.2 in terms of the hyperparameters of the training procedure and the mathematics of the algorithm.

1.2 Convolutional neural network (CNN) (20 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 four hidden layers in total, of which the first three are convolutional layers and the last is a fully-connected layer. All convolutional layers apply 3×3

feature maps, but the first uses 32 and the last two use 64 feature maps. Between the convolutional layers there are 2×2 maximum pooling layers. The fully-connected layer has 64 neurons and is followed by an output layer with 10 neurons. You should use ReLU 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 cross-entropy.

Train this model on batches of 128 data points with a learning rate of 0.1 for 40 epochs. Plot the loss function and the accuracy of this model as a function of the number of epochs. Discuss the convergence of the model and any signatures of underfitting or overfitting.

1.2.2 - Incorporate an L2 regularisation with a coefficient of $5 \cdot 10^{-3}$ in all of your convolutional layers and compare loss and accuracy to the model you defined in Task 1.2.1. Explain how the regularisation affects the training procedure.

1.2.3 - Starting again with your unregularised network of Task 1.2.1, implement Dropout with a rate of 0.5 and at least one other regularisation method of your choice and characterise the performance and the effect of the regularisations on the network or the training procedure. Compare the results of the additional regularisations to the L2 regularisation obtained above.

1.2.4 - Compare the results obtained with the MLP in 1.1.1 to the results obtained with the CNN in 1.2.3 with Dropout in terms of accuracy, computational time for training over the same number of epochs, and number of parameters in the models. Explain the observed differences.

Task 2: Unsupervised learning (60 marks)

Dataset: In Task 2, you will work with a dataset associated with Zachary's karate club. This is a classic small dataset in social science describing a karate club that eventually split acrimoniously into two separate groups. You can read the story behind it here. *Note that during this task we are **not** necessarily trying to find these two groups.*

Each sample in the dataset corresponds to one of the N=34 individuals who were members of the karate club. We have two sources of information:

1. **A set of features** characterising the personality profile of all the individuals. This information is given as a $N \times p$ feature matrix F, where N=34 samples with p=100 features capturing different traits.
2. **The social network of friendships** between the members of the club. The N=34 nodes of the graph are the members and the E=78 edges correspond to friendships between them. This information is given as a $N \times N$ adjacency matrix, A.

The feature matrix F, the adjacency matrix A, and the 'acrimonious split' partition observed in real life are all available on Blackboard.

2.1 Clustering of the feature matrix (20 marks): Using only NumPy (and based on the code developed in your coding tasks), you will employ the k-means algorithm to cluster the feature matrix F.

2.1.1 - Obtain optimised clusterings by running k-means from 100 random initialisations for all values of k in the interval [2, 10] and plot the average within-cluster distance as a function of k. Describe your results and how/if they could be used to choose an optimal clustering for this dataset.

2.1.2 - Code the Calinski-Harabasz (CH) score (you will need to research how to do this) and plot the CH as a function of increasing k. Find the optimal clustering according to the CH score and explain your answer.

2.1.3 - Use the different clusterings obtained from your random initialisations of the k-means algorithm in 2.1.1 to evaluate the robustness of the clusterings as a function of k. Explain your results using different measures that capture the consistency and variability of the clusterings obtained as a function of k, and explain how/if they could be used to choose an optimal clustering for this dataset.

2.2 Dimensionality reduction of the feature matrix (15 marks): Using only NumPy (and based on the code developed in your coding tasks), employ PCA to carry out dimensionality reduction on the feature matrix.

2.2.1 - Plot the $N=34$ samples of the dataset as points in the d-dimensional PCA space for dimensions: (i) $d=1$, (ii) $d=2$, and (iii) $d=3$. Do you find any relation of these projecttions with the clustering results obtained in 2.1? Explain and reason your answer.

2.2.2 - Plot the proportion of explained variance of the PCA approximations of reduced dimensionality d for all values of d in the interval [1, 10]. Is there any indication that the dataset is well described in a reduced dimension? Explain your answer in terms of the spectral decomposition of $F^T F$.

2.3 Graph-based analysis (25 marks): In this subtask, we will analyse the friendship graph encoded by the adjacency matrix A.

Note: Some of the subtasks in 2.3 are NumPy-only, whereas in others you will be able to use the NetworkX package. These will be clearly indicated. The NetworkX package contains pre-implemented graph-theoretical algorithms and good plotting capabilities for networks that you should use to report your *results. The package is self-explanatory but we will give a quick recorded introduction to some of its features on Blackboard.*

2.3.1 - **Centralities:** Using only NumPy/SciPy**,** produce code to obtain three measures of centrality: PageRank, degree centrality, and eigenvector centrality. Apply them to the karate club 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.

2.3.2 - **Community detection:** For this subtask **you are allowed to use NetworkX**. Using the Clauset-Newman-Moore greedy modularity maximisation algorithm in NetworkX, compute the optimal number of communities k^* and the corresponding partition of the karate club graph. Use NetworkX to plot the obtained clusters on the graph assigning different colours to the nodes in each community. Show the distribution across the k^* communities of the top 8 most central nodes according to: (i) degree centrality and (ii) Pagerank, as computed in 2.3.1. Explain your findings.

2.3.3 - **Comparing clusterings:** Code the Adjusted Rand Index (ARI) and use it to quantify how similar the optimal clusterings obtained in 2.1.2 and 2.3.2 are to each other, as well as how similar those clusterings are to the real-life acrimonious split given on Blackboard. Discuss your results.

Task 3: Mastery component *(for MSc and MSci students only)*

Modularity maximisation and spectral partitioning - (25 marks)

3.1 Louvain algorithm for community detection (10 Marks): For this subtask **you are allowed to use NetworkX**. Install the **community package** (https://python-louvain.readthedocs.io/en/latest/) by executing in your terminal `pip install python-louvain`.

Use the Louvain algorithm with modularity in this package to find the optimal community structure for the karate club graph. *(You do not need to implement the Louvain algorithm by yourself. You can use the inbuilt functions in the community package.)*

Compare the results of this method to those obtained in 2.3.2.

Based on your own personal reading of the Louvain method and modularity maximisation, explain the differences between the Louvain algorithm used here in 3.1 and the algorithm used in 2.3.2.

3.2 Modularity maximisation with spectral partitioning (15 Marks):

3.2.1 - Bipartitioning algorithm: For this subtask 3.2.1 **you are only allowed to use NumPy/SciPy**.

On Blackboard you will find a Python notebook (CW2_Task3_Mastery.ipynb) that contains a skeleton of code in NumPy/SciPy for an algorithm that maximises modularity using spectral partitioning. The notebook contains gaps that you must fill (similarly to what you have done during the coding tasks).

Starting from that Python notebook on BB, and **using NumPy/SciPy alone**, develop an algorithm that uses spectral partitioning to find the partition that maximises modularity.

The algorithm follows the following main steps which we illustrate here through an example:

1. Find the spectral bipartition given by the normalised Laplacian of your network, say, [0,0,0,0,1,1,1,1] where 0 and 1 are the community id's.
2. Calculate the Modularity of this partition for the given graph.
3. Take one of the two communities found above and define the subgraph where the nodes of the subgraph are those within the community, e.g. take community 0 which contains nodes 1,2,3,4.
4. Compute the spectral bipartition of this subgraph, e.g. [0,1,1,1] is now the subpartition of the subgraph with nodes 1,2,3,4.
5. Replace the original community with this new sub-partition, e.g., we now have [0,2,2,2,1,1,1,1] after splitting community 0 into two separate communities.
6. Calculate the Modularity of this new partition. Does it increase or decrease? If it increases, then keep this partition. Otherwise we reject it.
7. Continue for each subgraph and further subpartitions until the modularity no longer increases.

Finalise the code of the algorithm provided in *CW2_Task3_Mastery.ipynb*; apply it to the karate club graph; and report the final partition obtained by the algorithm.

3.2.2 - Compare the graph clustering: You are allowed to use NetworkX in 3.2.2 to report your results and produce plots.

Use NetworkX to plot the obtained clusters on the graph by assigning different colours to the nodes in each community and compare it to those obtained in 2.3.2. Explain your findings.

Based on your own personal reading, give a brief explanation of why spectral partitioning can be used for community detection as in this algorithm.