Imperial College London

Department of Mathematics
MATH60026 - MATH70026
Methods for Data Science
Years 3/4/5

Coursework - General instructions - please read carefully

The goal of this coursework is to analyse a dataset using tools and algorithms introduced in the lectures, which you have also studied in detail through the weekly Python notebooks containing the computational tasks. You will solve the tasks in this coursework using Python, hence competent Python code is expected.

You are allowed to use:

- Python code that you have developed in your coding tasks;
- all the standard libraries in Python;
- any other basic mathematical functions contained in `numpy`;
- `pandas` to build and clean up data tables;
- `matplotlib` and `seaborn` for visualisation;
- PyTorch to build, train and evaluate the performance of CNNs (Task 1 below);
- `scipy` only to perform matrix eigendecomposition;
- `tqdm` as a utility to visually track the progress of `for` loops.

You are *not* allowed to use:

- any other model-level, automatic differentiation or network analysis Python package (*e.g.*, scikit-learn, statsmodels, JAX, Tensorflow/Keras, NetworkX, etc);
- ready-made code found anywhere online;
- conversational AI tools such as ChatGPT, Microsoft Bing, GitHub Copilot etc. to generate code and written answers.

Needless to say, your submission must be your own individual work: *You may discuss the analysis with your colleagues but code, written answers, figures and analysis must be written independently on your own. The Department uses code profiling and tools such as Turnitin to* ***check for plagiarism of any sort****. Plagiarism is a major form of academic misconduct.*

Marks

Coursework 2 (CW2) has two parts:

- Part 1 (take-home, *this document*, to be submitted by 19 March at 1 pm)
- Part 2 (in-class, open book, to take place on 20 March)

CW2 is worth **60% of the total mark for the course** (with Part 1 = 20% of the total mark for the course, and Part 2 = 40% of the total mark for the course).

Mastery component: The mastery component amounts to 20% of this CW: one task **(1.3)** has one version for 3rd year BSc students only, and one version for MSc and 4th year MSci students only (Mastery component) - pay attention to this and choose the right option in this task! *You should only answer the version that applies to your case.*

Some general guidance about writing your 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, 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 using markdown cells that guide the reader through the notebook;
(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 a relevant 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 with a judicious use of summary tables of figures.

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, and this should be shown using markdown cells that explain briefly your reasoning, choices and explanation of the code to guide the reader. The weekly notebooks provide examples of this practice.

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.

When you comment on your results, ***precise pointers to your code and your plots must be provided****: generic, boiler-plate comments on a method that are not based on the specific problem and the analysis carried out by you will receive zero marks.*

Similarly, 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 to the mark if it reveals lack of understanding of the required task.

Submission

For the submission of your coursework, you need to save **two documents**:

- a **Jupyter notebook (file format: ipynb)** with all your tasks clearly labelled. You should use the template notebook called *CID_Coursework2_Part1.ipynb* provided on Blackboard (folder 'Coursework/Coursework 2'). The notebook should have clear headings to indicate the answers to each question, *e.g.* 'Task 1.1'.

  The notebook should contain the cells with your code and their output, plus some brief text explaining your calculations, choices, mathematical reasoning, and discussion of results. *(Important: Before submitting you must run the notebook, and the outputs of the cells must be printed. Having all cells executed sequentially is a way of showing to us that your notebook correctly produces the displayed output. The absence of this output will be penalised).* You can use Google Colab or develop your 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, you should save the notebook as an **html file** (with name *CID_Coursework2_Part1.html*). Your ipynb file must produce all the output that appears in your html file, *i.e.*, make sure you have run all cells in the notebook before exporting the html.

The submission is done **online via Blackboard,** using the drop boxes inside the folder 'Coursework' on Blackboard.

The deadline for CW2 Part 1 is Tuesday, 19 March 2024 at 1 pm.

The submission of your coursework must consist of **two items,** to upload **separately**:

1) *A **single zip folder** containing your Jupyter notebook as an **ipynb file** and your notebook exported as an **html file**. Name your zip folder 'CID_Coursework2_Part1.zip', where CID is your student CID, e.g. 123456_Coursework2_Part1.zip.*
2) *The html file, named CID_Coursework2_Part1.html, which will be used for the plagiarism check.*

The submission should consist *only* of these 2 items - **Do not submit multiple files.**

Important note: Make sure you submit to the right drop box on Blackboard

- If you are a 3rd year BSc student: use the 'Coursework Drop Boxes' inside the folder 'Coursework'. The html file must be uploaded to 'Coursework 2/Part 1/Coursework 2 - Part 1 Drop Box Spring 24 - HTML', the zip folder must be uploaded to 'Coursework 2/Part 1/Coursework 2 - Part 1 Drop Box Spring 24 - ZIP'.

- If you are a 4th year MSci or MSc student: use the 'Mastery Coursework Drop Boxes' inside the folder 'Coursework'. The html file must be uploaded to 'Coursework 2/Part 1/Coursework 2 - Part 1 Drop Box Spring 24 - Mastery HTML', the zip file to 'Coursework 2/Part 1/Coursework 2 - Part 1 Drop Box Spring 24 - Mastery ZIP'.

Any mistake in the submission folder will cause a delay in the release of your mark. **Do not put your name on the files you submit** (only the CID), because the marking must be carried out preserving your anonymity.

Notes about online submissions:

1. *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.*
2. *You should also **check that your files are not empty or corrupted after submission**.*
3. ***To avoid last minute problems** with your online submission, we recommend that you **upload versions of your coursework early on**, before the deadline. You will be able to update your coursework until the deadline, but having this early version provides you with some safety backup. For the same reason, keep **backups of your work**, e.g. save regularly your notebook with its outputs as an .html file, which can be useful if something unpredicted happens just before the deadline.*
4. *If you have any issue with the submission, or you realise you have submitted your work to the wrong drop box, please contact directly the UGMathsOffice at maths-student-office@imperial.ac.uk or your MSc programme administrator, in such a way that they can help you solve the issue.*
5. *If you need an extension, or happen for any reason to submit your work late, please make a request for mitigating circumstances directly on ZINC.*

   *For points 4 and 5 above, **do not contact us - we, as lecturers, are not able to grant extensions nor make changes in the submission folders! We only get to see anonymised submissions.***

Coursework 2 - Part 1

Submission deadline: Tuesday, 19 March 2024 at 1pm

Dataset: In this coursework, you will work with a **dataset of images of stars measured with the Euclid telescope of the European Space Agency.** The training dataset comprises 648 images of stars (32 x 32 pixels) and the test set includes 32 images of stars (32 x 32 pixels). The stars in the dataset belong to four stellar classes: O5V, A0V, F5V and M5V. The first letter in the code denotes the *spectral type* (from hottest to coolest: O, B, A, F, G, K, M); the second digit represents the *spectral subtype* (from hottest to coolest: 1-9); the third letter V indicates that all the stars in our dataset are dwarf stars.

The training dataset is made available inside the folder 'Coursework/Coursework 2/Data/train' on Blackboard:

- The star images are made available on Blackboard as `star_images_train.npy`.
- The star classes O5V, A0V, F5V and M5V correspond, respectively, to the labels 0, 1, 2 and 3, and are made available on Blackboard as `star_classes_train.npy`.
- Additionally, we provide high quality embeddings (i.e., high-dimensional vectors) for each star image, computed using a deep neural network able to classify images into the 4 classes with high accuracy. These embeddings are made available on Blackboard as `star_embeddings_train.npy`.

In each of these three numpy arrays, the i^{th} element corresponds to the i^{th} star: i = 1,…,648.

The test set is provided in the folder 'Coursework/Coursework 2/Data/test' on Blackboard under the corresponding files `star_images_test.npy` and `star_classes_test.npy`.

Important:

- *Do **not** apply **standardisation** in any of the tasks apart from PCA (Task 2.1)*
- *Employ `np.random.choice` every time you are asked to split the training dataset into training and validation subsets, and set the seed to 0 using `np.random.seed(0)` before **each** call of the function `np.random.choice`.*

Task 1: Classification with a Convolutional Neural Network (25 marks)

1.1 (10 marks) - This task is about training a Convolutional Neural Network (CNN) to classify the star images into the 4 classes. You need to use PyTorch as in your coding task in the Notebook of Week 6.

First, split the training dataset in two parts: 75% for the actual training and 25% for the validation set.

Implement a CNN architecture that contains, in order from input to output:

- A 2-dimensional convolutional layer with 4 filters, kernel shape of (5, 5) and a ReLU activation function;
- A layer implementing max-pooling with a pooling window shape of (2, 2) and a stride of 2;
- Another 2-dimensional convolutional layer with 8 filters, kernel shape of (5, 5) and a ReLU activation function;
- A layer flattening the output of the previous layer;
- A linear layer followed by a softmax activation function.

Use `Adam` as the optimisation algorithm and define the loss function as categorical cross-entropy. Run the training for a maximum of 2000 epochs, with batches of size 128. Include early stopping during the training, monitoring the loss as your metric of performance and setting the `max_patience` parameter to 150 epochs. During training, also monitor the performance by computing the classification accuracy.

Produce a plot of the loss evaluated on the training and validation sets as a function of the epochs, and a plot of classification accuracy on the training and validation sets as a function of the epochs. Discuss the training convergence. Evaluate the classification accuracy on the test set and comment on the performance of the CNN.

1.2 (5 marks) - Keeping the same hyperparameters as in **1.1**, add L_2 regularisation to both 2-dimensional convolutional layers. Use the classification accuracy on the validation set to find the optimal penalty coefficient for the L_2 regularisation by scanning over values 0.001, 0.01, 0.1, 1, and 10. Using the optimal L_2 regularisation coefficient obtained, re-train the CNN on the training set and evaluate the classification accuracy on the test set. Discuss the performance of this CNN with regularisation in comparison to Task **1.1.**

1.3 (10 marks, 3rd-year students only) - In this task, you will gauge the sensitivity of the CNN to class imbalance, and you will test strategies to mitigate class imbalance. You will compare a strategy based on a reweighted loss function and a strategy based on *data augmentation,* i.e., producing additional training data.

First, create an imbalanced training set by randomly dropping half of the data points in class 3 (M5V) from the original training set, and split the resulting dataset into 75% for actual training and 25% for validation. Re-train the CNN from Task **1.1** on this imbalanced training set, plotting training and validation accuracies during training. Report the accuracy on the test set of this re-trained CNN model and discuss your result.

You will now follow two alternative strategies to deal with class imbalance:

Reweighted loss function: Modify the CNN loss function so that class 3 (now the minority class) has more weight, choosing an appropriate reweighting strategy. Train the CNN from Task **1.1** with this reweighted loss, plotting the training and validation accuracies during training. Report and discuss the test accuracy.

Data augmentation: To re-balance the training data, produce new training data of the minority class (Class 3) by using each image in the depleted class 3 to obtain an additional training image by adding Gaussian noise on each pixel drawn from the Gaussian distribution $\mathcal{N}$:

$$\text{Noise on pixel } p \sim \mathcal{N}(0, c\sigma_p^2)$$

where c is a hyper-parameter to optimise and σ_p^2 is the variance of the value of pixel p calculated across the images in the depleted class 3.

Once you have generated the augmented dataset, repeat the steps above: split the training set into 75% for actual training and 25% for validation; use the validation set to find an optimal value for c by scanning over the values 0.01, 0.1 and 1; re-train the CNN architecture from Task **1.1** on this augmented dataset, plotting training and validation accuracies during training. Report and discuss the test accuracy.

Finally, conclude **1.3** by discussing the effect of the two strategies (reweighted loss function and data augmentation), comparing their performance.

1.3 (10 marks, MSc/4th-year students only) - In this task, you will gauge the sensitivity of the CNN to class imbalance, and you will test strategies to mitigate class imbalance. You will compare a strategy based on a reweighted loss function and a strategy based on *data augmentation,* i.e., producing additional training data.

First, create an imbalanced training set by randomly dropping half of the data points in Class 3 (M5V) from the original training set, and split the resulting dataset into 75% for actual training and 25% for validation. Re-train the CNN from Task **1.1** on this imbalanced training set, plotting training and validation accuracies during training. Report the accuracy on the test set of this re-trained CNN model and discuss your result.

You will now follow two alternative strategies to deal with class imbalance:

Reweighted loss function: Modify the CNN loss function so that class 3 (now the minority class) has more weight, choosing an appropriate reweighting strategy. Train the CNN from Task **1.1** with this reweighted loss, plotting the training and validation accuracies during training. Report and discuss the test accuracy.

Data augmentation: To re-balance the training data, produce new training data of the minority class (Class 3) by using each image in the depleted Class 3 to obtain one additional training image by rotating it, choosing uniformly at random a rotation of either 90, 180, or 270 degrees [Hint: you can use different functions from `numpy` for this].

Once you have generated the augmented dataset, repeat the steps above: split the training set into 75% for actual training and 25% for validation, and re-train the CNN architecture from Task **1.1** on this augmented dataset, plotting training and validation accuracies during training. Report and discuss the test accuracy.

Finally, conclude **1.3** by discussing the effect of the two strategies (reweighted loss function and data augmentation), comparing their performance.

Task 2: Dimensionality Reduction: PCA vs ISOMAP (25 marks)

In this task, you will work with the embeddings of the star images provided in `star_embeddings_train.npy`. These embeddings are 180-dimensional vectors X_i, $i = 1, ..., 648$ (i.e., one for each star), which have been obtained from a deep learning model that captures complex pixel relationships informative about the star class. Here, you will explore dimensionality reduction of these embeddings through two techniques: PCA and an algorithm inspired by Isomap.

2.1 (7 marks) - Perform Principal Component Analysis (PCA) of the embedding vectors provided in `star_embeddings_train.npy`, following the standard PCA algorithm with data standardisation. Visualise each embedding in the 2D space of the first two principal components of PCA, colouring each embedding according to their star class. Discuss your results.

Next, evaluate how well this 2D representation matches the division into classes. Consider the 2D representations of the embeddings, and each class as a 'cluster' to compute the Davies-Bouldin Index DB measure of clustering quality:

$$DB = \frac{1}{C} \sum_{\alpha=1}^{C} \max_{\beta, \beta \neq \alpha} F_{\alpha\beta}$$

$$F_{\alpha\beta} = \frac{S_\alpha + S_\beta}{M_{\alpha\beta}}$$

where $C = 4$ is the number of clusters; S_α is the average Euclidean distance between each point of cluster α and the centroid of that cluster; and $M_{\alpha\beta}$ is the Euclidean distance between the centroids of clusters α and β (the centroids are computed like in k-means). Discuss your results in terms of DB.

Next you will implement dimensionality reduction using an algorithm inspired by Isomap (see notes and in-class session).

2.2 (8 marks) - First, we construct a graph where each embedding vector is a node of the graph. Compute the matrix D of pairwise cosine distances, where D_{ij} corresponds to the pairwise cosine distance between embedding vectors X_i and X_j (without standardisation):

$$D_{ij} = 1 - \frac{X_i \cdot X_j}{\|X_i\|_2 \, \|X_j\|_2}$$

From the cosine distance matrix D, construct the adjacency matrix A of the k-Nearest Neighbour (kNN) graph for $k = 9$, where two nodes i and j are connected with an undirected edge with weight D_{ij} if X_i is within the k-nearest neighbours of X_j or if X_j is within the k-nearest neighbours of X_i. Visualise the graph corresponding to A using the spectral layout introduced in the Notebook from Week 9, where the x- and y-coordinates of the nodes are given by the entries of the normalised second and third eigenvectors of the symmetric normalised graph Laplacian. Colour the nodes of the graph according to star types, and discuss the layout of the kNN graph in relation to the star classes. Explain your results.

2.3 (10 marks) - The second step in the Isomap-like algorithm is to compute distances between nodes in the graph constructed in **2.2**. In particular, compute the resistance distance matrix R, where R_{ij} is given by:

$$R_{ij} := \Gamma_{ii} + \Gamma_{jj} - 2\Gamma_{ij}$$

$$\Gamma = (L - \frac{1}{N}\mathbf{1}\mathbf{1}^T)^+$$

where $^+$ denotes the Moore–Penrose inverse; L is the combinatorial graph Laplacian; N is the number of nodes; and $\mathbf{1}$ is the N-dimensional vector of ones.

Use a scatter plot to represent, for each i, the mean cosine distance from X_i to all other data points (computed in Task **2.2)** against the mean resistance distance from node i to all other nodes. Comment on and explain the differences that you observe between the two distances.

Finally, obtain a 2D projection of the graph distances, by computing the centred distance matrix τ from the resistance distance matrix R as given by the formula:

$$\tau := -HSH/2$$

$$H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$$

where S is given by $S_{ij} = R_{ij}^2$ and I is the $N \times N$ identity matrix.

Find the eigenvectors of τ and visualise the embeddings in the 2D space of the top two eigenvectors of τ (i.e., corresponding to the two eigenvalues with the largest value), colouring each embedding according to their star class. Evaluate how well this 2D representation matches the division into classes by calculating the DB index, as in Task **2.1**, and compare it to the results obtained from the 2D representation from PCA in Task **2.1**. Discuss and explain the observed differences.