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 different datasets: a collection of images and a gene expression matrix. Task 1 deals with the image data: there, you will perform a classification task using neural networks and image denoising/image clustering using dimensionality reduction techniques. Task 2 deals with the gene expression data, and you will perform tasks related to clustering and graph-based analysis of the gene network.

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

You are allowed to use only NumPy/SciPy, you are NOT allowed to use sklearn, statsmodels, networkX and any other advanced library we haven't mentioned.

Task 1: Neural Networks, Dimensionality Reduction and Mixture Models (65 marks)

Dataset 1: In Task 1, you will explore how a neural network architecture and techniques for dimensionality reduction perform on a set of images from the MNIST dataset. This data set consists of two files: a *training set* with 6000 images (file `MNIST_train.csv`) and a *test set* with 1000 images (file `MNIST_test.csv`). Each image is 28 x 28 pixels, encoded as an array of size 784. The images represent handwritten digits from 0 to 9, so they belong to 10 different classes (the column 'Label' indicates the class of each image). The dataset is well balanced, i.e., it has 600 images of each class in the training set and 100 images of each class in the test set. First, you will train a feed-forward neural network to classify images; then, you will perform image denoising through Principal Component Analysis (PCA) and Non-negative Matrix Factorization (NMF), and model the data as a Gaussian Mixture.

1.1 Multi-Layer Perceptron (MLP) (25 marks)

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

Architecture of the network: Your network should have an input layer, **3** hidden layers (with **200** neurons each), followed by the output layer with **10** neurons (one for each class). As your activation function between all layers, you should use the **softplus** activation function:

$$\mathrm{Softplus}(x) = \frac{\log(1+\exp(\beta x))}{\beta}$$

with $\beta = 1$. You should use the **softmax** function as the activation function on the output layer. Fix the optimisation method as stochastic gradient descent (SGD), and define the loss function as the **KL divergence** [Note: you may want to consider transformations of the KL divergence to avoid underflow issues if necessary].

Using the training set**,** train the MLP on batches of 128 data points for 40 epochs. Train the network several times, gradually varying the learning rate after each training between 10^{-5} and 10^{-1} (take no more than 7 values in this interval). Record the final loss achieved for each learning rate value and plot it as a function of the learning rate. Comment on the trend you observe and use this plot to set the optimal learning rate.

1.1.2 - Retrain the MLP with the optimal learning rate, using batches of 128 data points and 40 epochs again. Plot the loss and the accuracy of the MLP as a function of the number of epochs for both the training and test sets.

1.1.3 - Retrain the MLP of task **1.1.2,** reducing the width of the hidden layers width to 50 neurons. Evaluate the accuracy on both the training and test sets and compare it to the accuracies of the MLP of task **1.1.2**. Discuss this comparison.

1.1.4 - Introduce in the MLP from task **1.1.2** a dropout with rate=0.2 at each hidden layer and scale the output of each neuron in the hidden layers to compensate for the dropout during training. Plot the loss and the accuracy of the MLP as a function of the number of epochs for both the training and test sets to demonstrate convergence [Hint: you can use larger learning rates and number of epochs in the presence of dropout]. Produce histograms of the activations of the units in the first hidden layer when the network is evaluated on a batch of your choice of the test data, both for the network with dropout and without dropout (task **1.1.2**). Discuss the comparison between these histograms and, more generally, the effect of the dropout on the model performance compared with the model from task **1.1.2**.

1.2 Dimensionality reduction (20 marks)

In this task, you will use PCA and NMF to perform image denoising on a version of the MNIST dataset that is corrupted by noise (files `MNIST_train_noisy.txt` and `MNIST_test_noisy.txt`). Each row in `MNIST_train_noisy.txt` and `MNIST_test_noisy.txt` contains the same image as the corresponding row in, respectively, `MNIST_train.csv` and `MNIST_test.csv`, but with added zero mean Gaussian noise. Consider first the image data in `MNIST_train_noisy.txt`.

1.2.1 - Perform PCA to carry out dimensionality reduction - we shall use *m* to denote the number of principal components. Plot the fraction of variance explained by PCA as *m* is increased and provide the 3 values of *m* at which, respectively, 70%, 80% and 90% of the variance is explained. Using *imshow* (like in your notebooks), visualise the first *m=10* principal components from PCA.

1.2.2 - Perform dimensionality reduction implementing NMF, with *m=10* dimensions onto which to project the data. Plot the loss as a function of the number of iterations, showing that your chosen number of iterations is sufficient for convergence. NMF expresses the data as a linear combination of non-negative components. Visualise the *m=10* NMF components using *imshow*, discuss the comparison between NMF and PCA components (obtained in Task **1.2.1**), explaining why they differ.

1.2.3 - Image denoising: PCA and NMF can be applied to the task of image denoising, i.e. to reconstruct the original images from images corrupted by noise. Train PCA and NMF on `MNIST_train_noisy.txt` to find the set of principal components in PCA and non-negative components in NMF, using *m* = 100 for both PCA and NMF. Consider now the test set of noisy images (`MNIST_test_noisy.txt`) and write the formula to find, for each image in this set, the corresponding reconstructed image as a linear combination of the principal components in PCA and of the non-negative components in NMF. Choose an image from the noisy test set and visualise it along with its original version (from `MNIST_test.csv`) and its reconstructed version by both PCA and NMF with *m* = 100. Discuss the result of image denoising of both these methods.

1.2.4 - Consider the PCA implementation of image denoising from **1.2.3**. Compute the Mean-Squared Error (MSE) between the reconstructed images and (i) the test set of uncorrupted images (`MNIST_test.csv`) and (ii) the test set of noise-corrupted images (`MNIST_test_noisy.txt`). Plot, on the same figure, these two MSEs as a function of the number of PCA components *m*, varying *m* in the interval 5-400 with a step size 5. Explain the results plotted. Like in **1.2.3**, visualise an example of a reconstructed image for *m* = 10, 40, 100, 200 to illustrate your discussion of the MSEs.

1.3 Gaussian Mixture Models (20 marks).

1.3.1 - Use the first five principal components of the PCA on the first *N=1000* images of the dataset `MNIST_train.csv` to perform clustering using a Gaussian Mixture Model with 10 hidden components. Visualise the space spanned by the top two principal components, colouring each point according to their cluster.

1.3.2 - Use the class labels to visualise the space spanned by the top two principal components and colour each point according to their class label. Map the class labels to the best-fitting cluster index, print and discuss the result. To this end, use the log-likelihood of each mixture component evaluated at the images in each class to measure the fit quality.

1.3.3 - Use the label-cluster index map to compute the cluster probabilities for each digit. For each class, visualise the space spanned by the top two components by colouring each point according to the cluster probability of the best-fitting cluster. Interpret the result with respect to the uncertainty of the clustering.

1.3.4 - Retrain the Gaussian Mixture Model with a reduced number of five and eight components and compute the map of the class labels to the best-fitting cluster index (as in **1.3.2**). Compare your results to the 10 component results (tasks **1.3.2** and **1.3.3**).

Task 2: Clustering and graph-based analysis (35 marks)

Dataset 2: In Task 2, you will work on a dataset reporting measurements of gene expression in different tumour samples (file `gene_expression_data.csv`). This dataset consists of a matrix 800 x 96: each of the 800 rows corresponds to one sample of gene expression levels; the 95 columns containing 'Gene' in the name give the expression levels of different genes (the descriptors); the column 'Type' contains acronyms denoting the type of tumour from which each sample comes from (BRCA, KIRC, COAD, LUAD and PRAD). First, you will perform clustering through *k*-means to identify groups of samples with shared gene expression patterns; next, you will resort to graph-based analysis to study the structure of the network of co-regulation among genes that can be constructed from these expression data.

2.1 Clustering (15 marks):

2.1.1 - Consider the 800 x 95 matrix of gene expression levels across the 800 samples**.** Implement *k*-means on these data to find groups of samples with similar gene expression patterns. Code a function to compute the Calinski-Harabasz index of a given clustering, which measures the clustering quality based on a ratio of the between-cluster scatter matrix, and the within-cluster scatter matrix. The definition of the Calinski-Harabasz index is given by expressions (2)-(3)-(4) in this article.

Use this function to determine the optimal *k*, running *k*-means, for each *k,* for 5 different initialisations and averaging the corresponding Calinski-Harabasz indices. Provide the size of the clusters found by *k*-means with the optimal *k*. Plot the average Calinski-Harabasz index as a function of *k*.

2.1.2 - The tumour type of each sample (indicated by the column 'Type') can be considered as the class to which the sample belongs. In this subtask, you will assess the consistency between the clustering of the samples you obtained in **2.1.1** and the assignment to these classes (tumour types). To this end, code a function to compute the homogeneity score, a score measuring the extent to which the data points in one cluster belong to the same class (hence the homogeneity of the clusters). Based on this value, discuss whether the clusters are consistent with the sample tumour types.

Given a set of classes C and a set of clusters K of size, respectively $|C|$ and $|K|$, the homogeneity score is given by:

$$\text{Homogeneity score} = 1 - \frac{H(C|K)}{H(C)}$$

where the symbols H denote entropies defined as follows:

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \qquad H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{N} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{N}$$

N is the total number of data points and a_{ck} is the number of data points that belong to class c and are assigned to cluster k.

2.2 Graph-based analysis (20 marks):

2.2.1 - Compute the matrix of connected correlations between genes (columns) and use it as a similarity matrix to construct a graph that links genes. Set to zero the diagonal terms and all the terms whose absolute value is smaller than 0.75 (i.e., perform thresholding with threshold = 0.75 on the absolute value). Take the resulting matrix as the adjacency matrix *A* of a weighted graph describing gene co-expression. Visualise the adjacency matrix *A* using *imshow*.

2.2.2 - Compute the degree centrality of each node (gene) in the network, sort the genes in descending order by these values and print the index of the 5 top ranking genes.

2.2.3 - Compute the symmetric normalised Laplacian of the graph and derive its eigenvalues and eigenvectors. Plot the spectrum of eigenvalues, marking on the plot the gap between eigenvalues that, upon numerical rounding, have respectively zero and non-zero values. Provide the number *r* of zero eigenvalues and explain what they tell about the network structure.

2.2.4 - Let *U* be the 95 x *r* matrix containing the *r* eigenvectors of the symmetric normalised Laplacian corresponding to zero eigenvalues as columns. From *U*, form a new matrix *T* of size 95 x *r* by normalising the rows of *U* to norm 1, that is, set:

$$T_{ij} = U_{ij} / \sqrt{\sum_k U_{ik}^2}$$

(this row normalisation step is usually required for spectral partitioning with the symmetric normalised Laplacian according to the famous algorithm by Ng, Jordan, and Weiss 2001). Apply *k*-means to the matrix *T* varying *k.* Use the normalised within-cluster distance (expression 9.6 in the lecture notes) and the elbow method to set the appropriate value of *k*. Obtain the clustering corresponding to this *k* and provide an interpretation of such clusters.

2.2.5 - Consider the biggest cluster identified in **2.2.4** and use the normalised Laplacian to perform spectral partitioning and derive a binary partition of the subgraph corresponding to this cluster. Design a plot (or a set of plots) to visualise the magnitude of the network links within each partition and across the two partitions, commenting on the pattern you observe.

2.2.6 - Consider the subgraph from **2.2.5,** compute the degree centrality of the nodes in this subgraph and print the index of 5 top-ranking ones (like in **2.2.2**). Compare the degree centrality values with the ones obtained for the full graph in **2.2.2**, and explain the results of this comparison.

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

In the Mastery component, you will focus on another measure of centrality for the gene network of task **2.2** and on a time-series analysis of stock prices using Hidden Markov Models (HMM).

3.1 Betweenness Centrality (10 marks):

3.1.1 - Consider here the unweighted version of the graph from task **2.2,** i.e., set to 1 all the elements of the adjacency matrix *A* corresponding to an edge. Repeat tasks **2.2.2** and **2.2.6** using the betweenness centrality instead of the degree centrality. You need to code a function to compute the betweenness centrality of each node: remember that you are allowed to use only NumPy/SciPy while **you are NOT allowed to use sklearn, networkX and any other graph analysis library**.

To this end, use the skeleton function contained in the file skeleton_betweenness.ipynb (see the coursework folder), which guides you through implementing Brandes' algorithm. Discuss your results, commenting on the comparison between the ranking of nodes by betweenness centrality and the one by degree centrality (obtained in **2.2.2** and **2.2.6**).

3.2 Hidden Markov Models (15 marks):

3.2.1 - Consider the monthly time series of the historical stock price of the S&P 500 Stock Index from 2000-2015 (file `stock.csv`). Extract the closing price, return (percentage of the monthly change of closing price) and range (the relative difference between the monthly low and high prices) and visualise the time series.

3.2.2 - Use the Expectation-Maximisation (EM) algorithm with a suitable termination criterion to train an HMM with three components on the time series obtained in **3.2.1**. Provide an interpretation of the hidden states by visualising the means and variances of the mixture components.

3.2.3 - Cluster the time series and visualise the inferred sequence of hidden states by distinguishing them in the original time series (e.g. using different plot markers).

3.2.4 - Perform a grid search optimising over the number of mixture components k using the Akaike Information Criterion (AIC) in the range $k = 1, 2, ..., 10$, and visualise the log-likelihood and the AIC. You can initialise the EM algorithm using the optimal values of the HMM with one less component. Visualise the inferred sequence of hidden states for the optimum number of mixture components.

3.2.5 - Repeat the clustering by considering only data before August 2007. Next, by sampling 10 times from the underlying Markov chain, provide a forecast of the time series for the following two years. Comment on the predictive capacity of your model in general and with respect to the dip of the Global Financial Crisis (2007-2008). Briefly discuss your findings in connection with the Markov assumption underlying HMMs.