Coursework for *Machine Learning* (COMS30083)

James Cussens

1 Submission Instructions

Submission Deadline: 1pm Fri 29 Nov 2024.

1. Your submission should consist of exactly 5 files. One of these files will be a PDF called `report.pdf` which is your report. The other 4 files are all Python scripts which contain the code you have written. The scripts containing the code for Section 7–10 should be called `clustering.py`, `classification.py`, `regression.py` and `hmms.py`, respectively.

2. When you are ready to submit, put your 5 files into a zip file called `cw_⟨userid⟩.zip` (Replace `⟨userid⟩` with your university username, note that is **not** your examination number or student number.)

3. You should submit your coursework via Blackboard. Go to the same 'Course' where you got this coursework PDF from, go to the 'Assessment, submission and feedback' section and then scroll to the bottom of that page to find the submission point.

2 General Instructions and Advice

1. Note that, as explained in Section 6, you get a small number of marks merely for following the submission instructions.

2. With the exception of the marks given for following submission instructions, all marks are given either for fulfilling **Code Tasks** or **Report Tasks**. The marks for a **Code Task** are determined entirely by your submitted Python scripts and the marks for a **Report Task** are determined entirely by your report.

3. There is no point putting anything in your report that is not your answer to a **Report Task**.

4. Your PDF report should be be no more than **6 pages long** (in total, including references) and no less than 11 point font. If you submit more

than 6 pages you will not be penalised but only the first 6 pages will be considered. It is perfectly possible to submit an excellent report which is substantially shorter than 6 pages.

5. Your Python scripts should run without throwing errors.

6. Although in a real-word machine learning project producing quality code adhering to established software engineering practices is important, in this coursework assignment, quality of, or 'elegance' of, code does not contribute to your mark.

7. You can and should import appropriate libraries (e.g. scikit-learn, PyMC, PyTorch) to help you complete the tasks you are required to do.

8. Feel free to use any or all of the following as a starting point for your code: lab exercises, examples from the documentation for the libraries we used in the labs and also tutorial material from there. You do not need to acknowledge using code from such sources. Using code from elsewhere is allowed (as long as it does not come from another student on this unit), but please add an acknowledgement of its source as a comment in the code (not in your report).

9. Note that there are 62 marks available for **Code Tasks** and 38 marks available for **Report Tasks** so there is greater emphasis on successfully implementing training and testing than justifying choices and analysing results.

10. When developing code you may find it useful to write individual scripts for the various **Code Tasks**. But be sure that your submission has only 4 Python scripts.

3 Support available

1. There are *Coursework Support Sessions* which take place 1500-1700 in QB 1.80 on the following Thursdays: 14 Nov, 21 Nov and 28 Nov. Attendance at these is, of course, optional.

2. The key principle behind providing support to students for this coursework is that all students receive identical information. To this end all queries about the coursework (other than those raised at Coursework Support Sessions) should be posted on Teams on the 'Open' channel of the COMS30083 group.

4 Introduction

In Sections 7–9 of this coursework assignment you are given tasks from three central areas of machine learning, namely clustering, classification and regression, respectively. In each case, the machine learning problem is typical: you

are given some data, and sometimes additional information, and asked to apply various methods and analyse their performance. In Section 10 your goal is to learn the parameters of a Hidden Markov Model (HMM).

There are basically three activities which are being tested by this coursework assignment:

Implementing You will be asked to write Python code which, by exploiting functionality implemented in ML software libraries, performs machine learning on given datasets.

Choosing You will be asked to choose between alternative machine learning approaches. These choices can be informed either by theoretical knowledge or empirical results or a combination of the two.

Analysing You will be asked to analyse the performance of what is learned, explaining, as much as is possible, differences in performance of different approaches. It is often helpful to produce plots as part of this.

5 Marking criteria

The coursework has been broken down into 25 tasks partly to help you see how your coursework will be marked. Evidently, the mark you receive for a particular task will be based on how well you perform that task. In the case of **Code Tasks** you will get a non-zero mark for any working solution (e.g. for **Code Task 2** somehow applying k-means to do clustering on the given data) but to get full marks for a **Code Task** you need to implement a well-chosen approach (e.g. for **Code Task 2** applying k-means in a manner appropriate for the particular given machine learning problem).

The **Report Tasks** (which total only 38 marks) are deliberately more open-ended than **Code Tasks**. It is up to you to decide what counts as a good justification/explanation/analysis. Note that if you make a poor choice of machine learning approach you will lose marks for the relevant **Code Task** (as explained above) and also with any associated **Report Task**, since it is not possible to give a good justification for a bad choice.

6 Submission (5 marks)

You will receive 5 marks for following the submission instructions given in Section 1. If they are not followed exactly then you will lose all of these 5 marks.

7 Clustering (25 marks)

One way to compare different clustering approaches is

1. to use data where each datapoint has a class label,

2. to view the classes as the 'true' clusters,

3. to run each clustering algorithm on the data *without the class labels*,

4. and to compare the learned clusters with the 'true' clusters

In this part of the coursework you will be doing this using the *Covertype* dataset from the UCI Machine Learning Repository. You can download this data using the scikit-learn function `sklearn.datasets.fetch_covtype`.

Use K-means and also a Gaussian Mixture model to cluster a subset of the data with the target variable omitted. This subset should be a randomly chosen subset of 10,000 datapoints (the complete dataset has 581,012 datapoints.) In addition, create a baseline clustering where each datapoint is randomly assigned a cluster label. In all cases set the number of clusters to $K = 7$ (so that, for example, in the baseline clustering each datapoint is assigned one of the cluster labels $\{1, 2, 3, 4, 5, 6, 7\}$ with probability $1/7$). The necessary code to do all this should be in your script `clustering.py`.

Code Task 1 Create a 10,000 datapoint random subset of the data. (3 marks)

Code Task 2 Do clustering using k-means. (3 marks)

Code Task 3 Do clustering using a Gaussian mixture model (GMM). (3 marks)

Code Task 4 Do clustering using the random baseline. (3 marks)

Report Task 1 There are various ways to run k-means and GMM. Provide a justification for how you chose to run them. Do not use the class labels to inform your choice (i.e. pretend you do not have access to the class labels when making your choices). (4 marks)

To assess the qualities of the 3 clustering methods, you should use the class label (which the clustering methods did not see) as follows. For each pair of datapoints with the same class label, record whether (i) K-means, (ii) GMM and (iii) the random baseline assigns the two datapoints to different clusters; doing this counts as an error made by the clustering algorithm.

Code Task 5 Write Python code to count the errors made by the various clustering algorithms. (3 marks)

Report Task 2 In your report, for each of the 3 methods report how many errors they made. Also report the total number of pairs of datapoints with the same class label. (2 marks)

Report Task 3 Analyse these error counts, attempting to explain why they vary. (4 marks)

8 Classification (28 marks)

In this part of the coursework you compare different classification algorithms using the typical approach of comparing performance on a test set. Your job here is simply stated. Using the *Covertype* dataset again apply (1) logistic regression, (2) a single decision tree method and (3) some ensemble method to build a classifier from a randomly sampled subset of 80% of the data. You are also asked to attempt to build an SVM classifier. You should test performance on the remaining 20% and explain your results. The goal when training the classifiers is to achieve good test set results, although, of course, you are not permitted to use the test set to achieve this goal.

Code Task 6 Split the data into 80% train and 20% test. (2 marks)

Code Task 7 Train logistic regression and compute its test set accuracy. (3 marks)

Code Task 8 Train a decision tree and compute its test set accuracy. (3 marks)

Code Task 9 Train an ensemble method and compute its test set accuracy. (6 marks)

Report Task 4 Describe and explain the problems that occur when you attempt to train an SVM classifier. (3 marks)

Report Task 5 For each of the 3 classifiers explain any choices you made. It is, of course, fine to make a choice based on empirical testing (not on the test set!). If you do this be sure to report on the approaches you tried out and eventually rejected. (6 marks)

Report Task 6 Explain the differences in performance of the 3 classifiers on the test set. (5 marks)

9 Regression (30 marks)

Download (from Blackboard) the file `regression_train.txt`. This file contains 100 datapoints, where each datapoint is a pair of real numbers (x, y). Your task is to use this data to construct a regression model for predicting y values from x values with the goal of getting good predictions on a test set. This data is synthetic data generated as follows: the x values were generated with the following line of code: `x = np.linspace(-10,10,num=100)`. Each y value was sampled from a distribution $f(x)+\epsilon$ where f is some unknown (to you!) function and ϵ is a random variable whose distribution is uniform in the interval $[0, 200]$. Note that ϵ does **not** have a Gaussian distribution.

Code Task 10 Train a linear regression model where the parameters are fit using maximum likelihood estimation (equivalently by minimising square error). (4 marks)

Code Task 11 Train a neural network where the network architecture is up to you. You should use squared error as the cost function. You must use `pytorch` to do this. (8 marks)

Code Task 12 Take a Bayesian approach for a parametric regression model of your choice. You should use `PyMC` to generate (estimates of) the posterior distributions. As always with a Bayesian approach it is up to you to choose the prior distributions for the model parameters. (6 marks)

Report Task 7 For each of the 3 methods you have used justify any choices you made. Using empirical testing (not on the test set) to inform your choice is fine. (4 marks)

Download (from Blackboard) the file `regression_test.txt`. This file is similar to `regression_train.txt` except the x values run from 10 up to 20.

Code Task 13 Produce predictions for the linear regression model and neural network model for each test data point and compute the mean squared error (on the test set) for each method. Note that you are **not** required to produce test set predictions from the Bayesian model, since this is just too tricky! (2 marks).

Report Task 8 Include the test set accuracies computed in Code Task 13 in your report. In addition, include 2 figures each for the linear regression model and the neural network (so 4 figures in total). In the first figure, the plot should have the training data from `regression_train.txt` represented by a scatter plot and the predictions for y made by the model on the training set x values. The second plot should be similar except that the test data from `regression_test.txt` should be used (3 marks).

Report Task 9 Analyse any differences in performance of the linear regression mode and neural network on the test data (3 marks).

10 Hidden Markov Models (12 marks)

This question concerns an imaginary robot that moves around a 3×3 grid world. So at any time point the robot is at one of these 9 co-ordinates: $(0, 0)$, $(0, 1)$, $(0, 2)$, $(1, 0)$, $(1, 1)$, $(1, 2)$, $(2, 0)$, $(2, 1)$ or $(2, 2)$. Associated with each of these locations is a probability distribution over rewards the robot can receive. The possible reward values are 0, 1 or 2.

The robot moves probabilistically between locations but can only move to a neighbouring location. The locations which neighbour location (x, y) are the subset of locations $(x, y+1)$, $(x, y-1)$ $(x+1, y)$ and $(x-1, y)$ which are in the grid world. If a there are n neighbouring locations, the transition probability for each of them is $1/n$. Note that the robot never stays still; it always moves to a neighbouring location.

There is also a probability distribution over starting location for the robot. Clearly this scenario can be modelled as a (discrete) hidden Markov model. Download (from Blackboard) the file `rewards.txt`. This file contains a sequence of rewards received by the robot as it moves through the grid world with particular values for the start, transition and emission probability distributions.

Code Task 14 Use the EM algorithm provided by the `hmmlearn` package to attempt to learn these probability distributions. You have been given the true transition probabilities above, but do not provide these to the EM algorithm (5 marks).

Code Task 15 Apply EM once again to the rewards data. But this time let the algorithm know the true transition probabilities (given above). In `hmmlearn` you can do this by setting the `transmat_` attribute appropriately and setting both `params` and `init_params` to `'se'` when constructing the relevant HMM object (3 marks).

Report Task 10 Present and analyse the results of learning both with and without true transition probabilities. How confident are you that the estimated parameters are reasonably close to the true parameters? Note that for **Code Task 14** you can compare the learned estimates of the transition probabilities to their true values (4 marks).