Inf2B Coursework 2

(Ver. 1.0.1)

Submission due: 4pm, Friday 6th April 2018

Hiroshi Shimodaira and Heru Praptono

1 Outline

The coursework consists of three tasks, Task 1 – K-NN classification, Task 2 – Bernoulli naive Bayes classification, and Task 3 – Bayes classification with Gaussian distributions, in which we use a data set of handwritten characters.

You are required to submit (i) three reports, one for each task, (ii) code, and (iii) results of experiments if specified, using the electronic submission command. Details are given in the corresponding task sections below. Some of the code and results of experiments submitted will be checked with an automated marking system in the DICE computing environment, so that it is essential that you follow the syntax of function or file format specified. No marks will be given if it does not meet the specifications. Efficiency of code and programming style (e.g. comments, indentation, and variable names) count. Those pieces of code that do not run or that do not finish in approximately five minutes on a standard DICE machine will not be marked. This coursework is out of 100 marks and forms 12.5% of your final Inf2b grade.

This coursework is individual coursework - group work is forbidden. You should work alone to complete the coursework. You are not allowed to show any written materials, data provided to you, results of your experiments, or code to anyone else. Never copy-and-paste material into your coursework and edit it. You can, however, use the code provided in the lecture notes, slides, and labs of this course, excluding some functions described later. High-level discussion that is not directly related to this coursework is fine.

Please note that assessed work is subject to University regulations on academic misconduct:
`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`
For late coursework and extension requests, see the page: `http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests`
Note that any extension request must be made to the ITO, and not to the lecturer.

Programming: Write code in Matlab(R2015a)/Octave or Python(version 2.7)+Numpy+Scipy+Matplotlib. Your code should run on standard DICE machines without the need of any additional software. There are some functions that you should write the code by yourself rather than using those of standard libraries available. See section 4 for details.

This document assumes Matlab programming. For Python, replace the Matlab filename extension (`.m`) with the one for Python (`.py`) for function/script files, but this does not apply to other files (e.g. data sets and experimental results).

2 Data

The coursework employs the EMNIST handwritten character data set `https://www.nist.gov/itl/iad/image-group/emnist-dataset`. Each character image is represented as 28-by-28 pixels in gray scale, being stored as a row vector of 784 elements ($28 \times 28 = 784$). A subset of the original EMNIST data set is considered in the coursework, restricting characters to English alphabet of 26 letters in either upper case or lower case.

You data set is stored in a Matlab file named `'data.mat'` and located in your coursework-data directory (denoted as *YourDataDir* hereafter) :

`/afs/inf.ed.ac.uk/group/teaching/inf2b/cwk2/d/`*UUN*`/data.mat`

where *UUN* denotes your UUN (DICE login name).

There are 1800 training samples and 300 test samples for each class. You can load the data set file in Matlab with the 'load' function, e.g. loading the training set by:

```
load('YourDataDir/data.mat');
```

which contains the following arrays/vectors:

Name	Size (Class)	Description
dataset.train.images	46800x784 (uint8)	training samples
dataset.train.labels	46800x1 (double)	class labels of training samples
dataset.test.images	7800x784 (uint8)	test samples
dataset.test.labels	7800x1 (double)	class labels of test samples

Each pixel value is represented as an unsigned byte integer (uint8) with the range in $[0, 255]$. Note that, after you load the data in your program, you should at first convert the image data to floating point (double precision) numbers. Additionally, for Task 1 and Task 3, you should divide the numbers by 255.0 so that the maximum value is less than or equal to 1.0.

A class label is represented as an integer number between 1 and 26, where 1 denotes 'A' and 26 'Z', respectively. The data set is supposed to contain letters in either upper case or lower case only, but you should expect that the actual data set allocated to you may contain both.

3 Task specifications

Task1 – K-NN classification [35 marks]

Task1.1 Write a Matlab function for k-NN classification with the (squared) Euclidean distance measure, and save it as 'Task1/my_knn_classify.m'. (NB: file names and function names are case sensitive) The syntax of the function should be as follows.

```
[Cpreds] = my_knn_classify(Xtrn, Ctrn, Xtst, Ks)
```

where

`Xtrn`	M-by-D training data matrix (of floating-point numbers in double-precision format, which is the default in Matlab) of training data, where M is the number of training samples, and D is the the number of elements in a sample. Note that each sample is represented as a row vector rather than a column vector.
`Ctrn`	M-by-1 label vector for `Xtrn`. `Ctrn(i)` is the class number of `Xtrn(i,:)`.
`Xtst`	N-by-D test data matrix, where N is the number of test samples.
`Ks`	L-by-1 vector of numbers of nearest neighbours.
`Cpreds`	N-by-L matrix of predicted class labels for `Xtst`. `Cpreds(i,j)` is the predicted class for `Xtst(i,:)` with the number of nearest neighbours being `Ks(j)`.

In case of ties (where there is more than one majority group), choose the smallest index (class label).

[15 marks]

Task1.2 Write a Matlab function for creating a confusion matrix, and save it as 'Task1/my_confusion.m'. The syntax of the function should be as follows.

```
[CM, acc] = my_confusion(Ctrues, Cpreds)
```

where

`Ctrues`	N-by-1 vector of ground truth (target) class labels
`Cpreds`	N-by-1 vector of predicted class labels
`CM`	K-by-K confusion matrix, where `CM(i,j)` is the number of samples whose target is the i'th class that was classified as j. K is the number of classes, which is 26 for the data set.
`acc`	A scalar variable representing the accuracy (i.e. correct classification rate) with the range in $[0, 1]$.

[5 marks]

Task1.3

(a) Write a Matlab script that carries out k-NN classification for the the given data set, and save the script as '`Task1/my_knn_system.m`'. The specifications of the script are as follows.

- Loads the data set.
- Runs a classification experiment on the data set, calling the classification function (`my_knn_classify`) with `kb = [1,3,5,10,20]`.
- Measures the user time taken for the classification experiment, and display the time (in seconds) to the standard output (i.e. display).
- Saves the confusion matrix for each k to a matrix variable `cm`$\underline{k}$, and save it with the file name '`Task1/cm`$\underline{k}$`.mat`', where $\underline{k}$ denotes the number of nearest neighbours (i.e. k) specified above. For example, assuming that your current directory is Task1 and k = 3,

  ```
  save("cm3.mat", "cm3");.
  ```

- Displays the following information (to the standard output).

k	The number of nearest neighbours
N	The number of test samples
Nerrs	The number of wrongly classified test samples
acc	Accuracy (i.e. correct classification rate)

(b) Run the script '`Task1/my_knn_system.m`', and report the user time taken and result shown on the display. The experimental result should be shown in a table.

[10 marks]

Task1.4 In your report, explain your implementation of k-NN classification in terms of speeding up, using mathematical expressions if possible. For example, nested loops are required for the algorithm to calculate distance for possible pairs of training samples and test samples, but the loop operation can be avoided or the number of loops can be reduced with vectorisation techniques.

[5 marks]

Task 2 – Bernoulli naive Bayes classification [30 marks]

This task considers naive Bayes classification with multivariate Bernoulli distributions. To that end, we convert an original pixel image vector $\mathbf{x} = (x_1, \ldots, x_D)^T$ to a binary image vector $\mathbf{b} = (b_1, \ldots, b_D)^T$, where b_i is a binary value of either 0 or 1, and $D = 784$ for the EMNIST data set. This conversion is called *binarisation*.

The likelihood for class C_k is given as follows.

$$P(\mathbf{b}|C_k) = \Pi_{i=1}^{D} P(b_i|C_k) = \Pi_{i=1}^{D} P(b_i{=}0|C_k)^{1-b_i} P(b_i{=}1|C_k)^{b_i}$$

A uniform prior distribution over class is assumed for the data set.

Task2.1 Write a Matlab function for the classification and save the code as '`Task2/my_bnb_classify.m`'. The syntax of the function should be as follows.

```
[Cpreds] = my_bnb_classify(Xtrn, Ctrn, Xtst, threshold)
```

where `Xtrn`, `Ctrn`, and `Xtst` are the same as those in Task1.

`threshold`	A scalar value for binarasation, where $b_i = 0$ if $x_i <$ threshold, 1 otherwise.
`Cpreds`	N-by-1 vector of predicted class labels for `Xtst`. `Cpreds(i)` is the predicted class for `Xtst(i,:)`.

Note that the binarisation should be carried out in this function.

[15 marks]

Task2.2

(a) Write a Matlab script that carries out the classification for the the given data set, and save the script as '`Task2/my_bnb_system.m`'. The specifications of the script are as follows.

- Loads the data set.
- Run a classification experiment on the data set, calling the classification function (`my_bnb_classify` with threshold=1.
- Measures the user time taken for the classification experiment, and display the time (in seconds) to the standard output.
- Obtains the confusion matrix using `my_counfusion()`, stores the confusion matrix to a matrix variable `cm`, and saves it with the file name '`Task2/cm.mat`'.
- Displays the following information (to the standard output).

N	The number of test samples
Nerrs	The number of wrongly classified test samples
acc	Accuracy (i.e. correct classification rate)

(b) Run the script '`Task2/my_bnb_system.m`', and report the result in your report using a table that shows the information of user time taken, N, Nerrs, and acc.

[10 marks]

Task2.3 Investigate the effect of the threshold on classification accuracy, and report your findings in the report.

[5 marks]

Task 3 – Bayes classification with Gaussian distributions [35 marks]

In this task, we consider Bayes classification with Gaussian distributions, where each class is modelled with a multivariate Gaussian distribution. Due to the nature of the data we use, it is likely that the determinant of a covariance matrix is zero or almost zero, and the matrix is not invertible. To avoid the problem, we employ the simple regularisation technique shown in Lecture 8 (multivariate Gaussians and classification), in which we add a small positive number (ϵ) to the diagonal elements of covariance matirix, i.e. $\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \epsilon I$, where I is the identity matrix. In addition to the regularisation, you should calculate determinants and likelihoods in the log domain to avoid numerical underflow.

In the following classification experiments, assume a uniform prior distribution over class, and *use the maximum likelihood estimation (MLE) to estimate model parameters.*

Task3.1 Write a Matlab function for the classification with a single Gaussian distribution per class, and save the code as '`Task3/my_gaussian_classify.m`'. The syntax of the function should be as follows.

```
[Cpreds, Ms, Covs] = my_gaussian_classify(Xtrn, Ctrn, Xtst, epsilon)
```

where `Xtrn`, `Ctrn`, `Xtst`, and `Cpreds` are the same as those in Task2, and `epsilon` is a scalar for the regularisation described above.

`Ms`	D-by-K matrix of mean vectors, where `Ms(:,k)` is the sample mean vector for class k.
`Covs`	D-by-D-by-K 3D array of covariance matrices, where `Cov(:,:,k)` is the sample covariance matrix (after regularisation) for class k.

[15 marks]

Task3.2

(a) Write a Matlab script that carries out the classification for the given data set, and save the script as 'Task3/my_gaussian_system.m'. The specifications of the script are as follows.

- Loads the data set.
- Calls the classification function with epsilon=0.01.
- Measures the user time taken for the classification experiment, and display the time (in seconds) to the standard output.
- Obtains the confusion matrix, stores it to a matrix variable `cm`, and saves it with the file name 'Task3/cm.mat'.
- Saves the mean vector and covariance matrix for Class 26, i.e, Ms(:,26) and Covs(:,:,26), to files with the file names 'Task3/m26.mat' and 'Task3/cov26.mat', respectively.
- Displays the following information (to the standard output).

N	The number of test samples
Nerrs	The number of wrongly classified test samples
acc	Accuracy (i.e. correct classification rate)

(b) Run the script 'Task3/my_gaussian_system.m', and report the result in your report using a table that shows the information of N, Nerrs, and acc.

[5 marks]

Task3.3 This is a mini project in which you try to improve classification accuracy by modifying the classifier developed in Task3.1, and write a short report about your investigation. The new classifier needs to be still in the the framework of Bayes classification with Gaussian distributions, but you can adopt techniques covered in the Inf2b course. For example, using the k-means clustering algorithm to obtain multiple Gaussian distributions per class, and dimensionality reduction with PCA may be worth exploring.

(a) Write a Matlab function for the improved classifier, and save the code as 'Task3/my_improved_gaussian_classify.m'. The syntax of the function should be as follows.

```
[Cpreds] = my_improved_gaussian_classify(Xtrn, Ctrn, Xtst)
```

where `Xtrn`, `Ctrn`, `Xtst`, and `Cpreds` are the same as described before. You may add arguments to the function if necessary.

(b) Write a Matlab script that carries out the classification for the given data set, and save the script as 'Task3/my_improved_gaussian_system.m'. The specifications of the script are basically the same as in Task3.2, but the confusion matrix should be saved as 'Task3/cm_improved.mat'.

(c) In your report, describe your investigation, clarifying the methods you employed, and report the results of experiment. Give discussions as to remaining problems and further improvement.

[15 marks]

4 Functions that are not allowed to use

Since one of the objectives of this coursework is to understand and implement basic algorithms for machine learning, you are not allowed to use those functions in standard libraries listed below. You should write the code by yourself using the basic operations of arithmetic for scalars, vectors, and matrices. If it is the case, use a different function name from the original one in standard libraries (e.g. MyCov() for cov() as shown in the table below). You may, however, use them for comparison purposes, i.e. to check your code.

Description of function	Typical names	Suggested name to implement
Pairwise (squared) Euclidean distance	pdist2()	MySqDist()
Compute the mean	mean()	MyMean()
Compute the covariance matrix	cov()	MyCov()
Compute Gaussian probability densities	mvnpdf()	
K-NN classification	fitcknn()	my_knn_classify()
K-means clustering	kmeans()	MyKmeans()
Compute confusion matrix	confusion()	my_confusion()
Other utilities for classification		

You may use those functions or operations:

Description	Typical names
Sum function	sum()
Cumulative sum	cumsum()
Square root function	sqrt()
Exponential function	e, exp()
Logarithmic function	log(), ln()
Matrix transpose	transpose(), '
Matrix inverse	inv()
Determinant	det()
Log determinant	logdet() ··· available in Inf2b cwk2 directory
Eigen values/vectors	eig()
Sort	sort()
Sample mode	mode()
Vectorisation helpers	bsxfun(), arrayfun()

(NB: the list is not exhaustive)

5 Submission

You should submit your work electronically via the DICE submit command by the deadline. No submission of printed document is required.

Since marking for each task will be done separately, you should prepare *separate reports* for the three tasks, and save your report files in PDF format and name them '`report_task1.pdf`', '`report_task2.pdf`', and '`report_task3.pdf`'. Remember to place your student number and the task name prominently at the top of each report. Do not indicate your name anywhere. Your report should be concise and brief, 1 or 2 pages long, for each task.

Create a directory named `LearnCW`, copy the PDF files of your reports in it. Create sub directories, `Task1`, `Task2`, and `Task3`, under `LearnCW`, and copy all of your code for each task to the corresponding sub directory.

Make sure that each task directory contains necessary pieces of code (with the correct names – file/directory names are case sensitive) so that coursework markers can run your code properly in DICE. A checklist will be available from the coursework web page. Submit your coursework from a DICE machine using:

```
submit inf2b 2 LearnCW
```