Inf2B Coursework 2

(Ver. 1.2)

Submission due: 4pm, Friday 5th April 2019

Hiroshi Shimodaira and JinHong Lu

1 Outline

The coursework consists of two tasks, Task 1 – PCA and clustering, Task 2 – classification with K-NN and Bayes classification with Gaussian distributions, in which we use a data set of handwritten characters (digits).

You are required to submit (i) two 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 and 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 (excluding Task 1.5 when k=20) 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, the data provided to you, results of your experiments, or code to anyone else. This includes posting your coursework to the internet and making it accessible to other people not only during the coursework period, but also after that. Never copy-and-paste material of other people (including those available on the internet) 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(R2018a)/Octave or Python(version 2.7)+Numpy+Scipy+Matplolib. 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 programming in Matlab. For Python, put all the specified functions into a single file for each Task, so that `task1.py` for Task 1, and `task2.py` for Task 2. Output data should be stored in Matlab's MAT binary format.

2 Data

The coursework employs the MNIST handwritten digits data set `http://yann.lecun.com/exdb/mnist`. 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 MNIST data set is considered in this coursework.

You data set is stored in your coursework-data directory (denoted as *YourDataDir* hereafter) :

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

where *UUN* denotes your UUN (DICE login name). In the directory, you will find the following four files:

File name	Matlab variable (Class)	Description
'tron-images-idx3-ubyte'	Xtrn[Ntrn,784] (uint8)	training samples
'trn-labels-idx1-ubyte'	Ytrn[Ntrn,1] (uint8)	class labels of training samples
'tst-images-idx3-ubyte'	Xtst[Ntst,784] (uint8)	test samples
'tst-labels-idx1-ubyte'	Ytst[Ntst,1] (uint8)	class labels of test samples

where Ntrn and Ntst denote the total number of training samples and test samples, respectively, which are smaller than the original ones. The file formats are the same as the original ones.

A sample code to load the data files will be provided in the following directory.

```
/afs/inf.ed.ac.uk/group/teaching/inf2b/cwk2/matlab_codes/
```

There are ten classes, $C_1, \ldots, C_{10}$, (C_1 corresponds to digit '0', C_{10} to '9'), and each class has about 4200 training samples and 300 test samples on the average, respectively, but exact number of samples may be different among different classes.

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) numbers. Additionally, 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 8-bit unsigned integer (uint8) number between 0 and 9, where 0 denotes '0' and 9 '9', respectively. For Matlab programming, you should change the original labels so that class number starts at 1 rather than 0.

3 Task specifications

Task1 – PCA and Clustering [50 marks]

Task 1.1 [5 marks]

(a) Write a Matlab function that displays the images of the first ten samples for each class using montage() function, so that each figure shows the ten samples for class C_k, where $k = 1, \ldots, 10$. Save your code as 'task1_1.m'. Note that file names and function names are case sensitive. The syntax of the function should be as follows.

```
function task1_1(X, Y)
```

where

- `X` M-by-D data matrix (of floating-point numbers in double-precision format, which is the default in Matlab), where M is the number of 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.
- `Y` M-by-1 label vector (of uint8) for `X`. `Y(i)` is the class number of `X(i,:)`.

(b) Run the following:

```
task1_1(Xtrn, Ytrn);
```

and save the image (hardcopy) for each class as 'task1_1_imgs_class$K$.pdf', where $K = 1, \ldots, 10$. (e.g. task1_1_imgs_class3.pdf is a hardcopy for C_3 in PDF format.)
In your report, show the ten figures.

Task 1.2 [5 marks]

(a) Write a Matlab function that calculates a mean vector of data for each class ($k = 1, \ldots, K$, where $K = 10$) and for all, and displays the images of $K + 1$ mean vectors in a single graph using montage() function. Save your function as 'task1_2.m'. The syntax of the function should be as follows.

```
function M = task1_2(X, Y)
```

where X and Y are the same formats as in Task 1.1.

`M` (K+1)-by-D mean vector matrix (double), where K is the number of classes, and D is the same as in Task 1.1. `M(K+1,:)` is the mean vector of the whole data.

(b) Run the following:

```
M = task1_2(Xtrn,Ytrn);
```

and save `M` as '`task1_2_M.mat`', save the figure as '`task1_2_imgs.pdf`'.
In your report, show the figure.

Task 1.3 [7 marks]

(a) Write a Matlab function that computes the principal components of a data set, and save it as '`comp_pca.m`'. The syntax of the function is as follows:

```
function [EVecs, EVals] = comp_pca(X)
```

`X` is the same format as in Task 1.1. `EVecs` are the eigenvectors (stored in a D-by-D matrix `EVecs`), and `EVals` are the corresponding eigen values $\{\lambda_i\}_{i=1}^{D}$ (stored in a D-by-1 vector). The eigenvalues should be sorted in descending order, so that λ_1 is the largest and λ_D is the smallest, and i'th column of `EVecs` should hold the eigenvector that corresponds to λ_i.
Eigenvectors are not unique by definition in terms of scale (length) and sign, but we make them unique in this coursework by putting the following additional constraints, which your `comp_pca()` should satisfy.

- The first element of each eigenvector is non-negative. If it is not the case, i.e. if the first element is negative, multiply -1 to the eigenvector (i.e. $\mathbf{v} \leftarrow -\mathbf{v}$) so that it gets the opposite direction.
- Each eigenvector is a unit vector, i.e. $\|\mathbf{v}\| = 1$, where $\mathbf{v}$ denotes an eigenvector. As far as you use Matlab's eig() or Python's numpy.linalg.eig(), you do not need to care about this, since either function ensures unit vectors.

(b) Write a Matlab function `task1_3()` that

- carries out PCA (i.e. calls `comp_pca`),
- computes cumulative variances, and plots them,
- finds the minimum number of PCA dimensions to cover 70%, 80%, 90%, 95% of the total variance,

and save it as '`task1_3.m`'. The syntax of the function should be as follows.

```
function [EVecs, EVals, CumVar, MinDims] = task1_3(X)
```

Where `EVecs` `EVals` the same as in part (a),

`Cumvar` D-by-1 vector (double) of cumulative variance.
`MinDims` 4-by-1 vector (int32) showing the minimum number of PCA dimensions shown above.

(c) Run the following:

```
[EVecs, EVals, CumVar, MinDims] = task1_3(Xtrn);
```

and save the graph as '`task1_3_graph.pdf`', and save `EVecs`, `Evals`, `CumVar`, `MinDims` as '`task1_3_evecs.mat`', '`task1_3_evals.mat`', '`task1_3_cumvar.mat`', and '`task1_3_mindims.mat`' respectively.
In your report, show the figure of commutative covariance, and show the values of `MinDims`.

Task 1.4 [5 marks]

(a) Write a Matlab function that displays the images of the first ten principal axes of PCA using montage() so that all the images are shown in a single graph. Save your code as '`task1_4.m`'. The syntax of the function should be as follows.

```
function task1_4(EVecs)
```

where `EVecs` is the same format as in Task 1.3.

(b) Run `task1_4(EVecs)` and save the figure as '`task1_4_imgs.pdf`', where `EVecs` is the eigenvector matrix obtained in Task 1.3.
In your report, show the images.

Task 1.5 [6 marks]

(a) Write a Matlab function that carries out the k-means clustering, and save it as `my_kMeansClustering.m`. The syntax of the function should be as follows.

```
function [C, idx, SSE] = my_kMeansClustering(X, k, initialCentres, maxIter)
```

where `X` is the same format as in Task 1.1.

k	The target number of clusters (int32)
initialCentres	Initial cluster centres in the k-by-D matrix (double), where each row represents a cluster centre.
maxIter	Maximum number of iterations (optional)
C	Cluster centres in the k-by-D matrix (double).
idx	Cluster indices in the N-by-1 vector (int32), where idx[i] indicates the cluster number (starting from 1) of i-th data point.
SSE	The sum squared error (the one shown in slide 21 for Lecture 3) for each iteration in the $(L+1)$-by-1 vector (double), where the first element is the SSE for the initial cluster centres, and L is the number of iterations done.

(b) Write a Matlab function that calls the k-mean clustering function for each k in a vector `Ks`, using the first k samples in the data set `X` as the initial cluster centres, and saves the returned `C`, `idx`, and `SSE` as '`task1_5_c_`$\underline{k}$`.mat`', '`task1_5_idx_`$\underline{k}$`.mat`', and '`task1_5_sse_`$\underline{k}$`.mat`', respectively. Save your code as '`task1_5.m`'. The syntax of the the function should be as follows.

```
function task1_5(X, Ks)
```

where `X` is the same as in Task 1.1, `Ks` is a 1-by-L vector (int32) of the numbers of nearest neighbours.

(c) Run the following:

```
Ks = [1,2,3,4,5,7,10,15,20];
task1_5(Xtrn, Ks);
```

In your report, show a graph of SSE vs k, and show the time taken for each k.

Task 1.6 [5 marks]

(a) Write a Matlab function that displays the image of each cluster centre, where you should use montage() function to put all the images into a single figure. Save your code as '`task1_6.m`'. The syntax of the function should be as follows.

```
function task1_6(MAT_ClusterCentres)
```

where `MAT_ClusterCentres` is a MAT binary file that contains `C`, the cluster centre matrix shown in Task 1.5.

(b) Run `task1_6('task1_5_c_`$\underline{k}$`.mat')` for k in `Ks` defined in Task 1.5, and save each figure as '`task1_6_imgs_`$\underline{k}$`.pdf`'.
In your report, show the figures.

Task 1.7 [7 marks]

(a) Write a Matlab function that visualises cluster regions on the 2D-PCA plane, where the position of the plane is specified by a point vector. The plotting range should be $m \pm 5\sigma$, where m is the mean and σ is the standard deviation of the data on the corresponding PCA axis. Save your code as '`task1_7.m`'. The syntax of the function should be as follows.

```
function Dmap = task1_7(MAT_ClusterCentres, MAT_M, MAT_evecs, MAT_evals,
posVec, nbins)
```

where the first three arguments are MAT files of cluster centres, eigenvectors, and eigenvalues. `posVec` is a 1-by-D vector (double) to specify the position of the 2D-PCA plane. We assume that `nbins` by `nbins` square grid points are used to plot the cluster regions. The output, `Dmap` is a nbin-by-nbin matrix (uint8), each element represents the cluster number that the point belongs to.

(b) Run the following for $k = 1, 2, 3, 5, 10$,

```
Dmap = task1_7('task1_5_c_k.mat', 'task1_2_M.mat', 'task1_3_evecs.mat',
'task1_3_evals.mat', mean(Xtrn), 200);
```

and save `Dmap` as '`task1_7_dmap_k.mat`', and save the figure of cluster regions as '`task1_7_imgs_k.pdf`'.

(c) In your report, show the images, and describe the method implemented for visualising the regions.

Task 1.8 [10 marks]

This is a mini research project, in which you are asked to investigate the k-means clustering in terms of initial cluster centres, i.e. how different initial cluster centres result in different cluster centres, for which employ SSE to measure the clustering performance. Report your experiments and findings in your report, and save your code as '`task1_8.m`'.

Task 2 – Classification [50 marks]

In the following classification experiments, assume a uniform prior distribution over class, and *use the maximum likelihood estimation (MLE) to estimate model parameters.* In programming, you can assume that the number of classes is equal to the maximum value of labels for training data, given that label values start at 1 rather than 0.

Task 2.1 [7 marks]

(a) Write a Matlab function for the classification with k-NN, save it as '`run_knn_classifier.m`'. The syntax of the function should be as follows.

```
function [Ypreds] = run_knn_classifier(Xtrain, Ytrain, Xtest, Ks)
```

where

`Ks`	1-by-L vector of the numbers of nearest neighbours.
`Ypreds`	N-by-L matrix (uint8) of predicted class labels for `Xtest`. `Ypreds(i,j)` is the predicted class for `Xtest(i,:)` with the number of nearest neighbours being `Ks(j)`.

(b) Write a Matlab function that creates a confusion matrix, and save the code as '`comp_confmat.m`'. The syntax of the function should be as follows.

```
function [CM, acc] = comp_confmat(Ytrues, Ypreds, K)
```

where

`Ytrues`	N-by-1 vector (uint8) of ground truth (target) class labels
`Ypreds`	N-by-1 vector (uint8) of predicted class labels
`K`	The number of classes (which is 10 for our data set)
`CM`	K-by-K confusion matrix (int32), where `CM(i,j)` is the number of samples classified as j'th class among those whose true class label is i.
`acc`	A scalar variable (double) representing the accuracy (i.e. correct classification rate) with the range in $[0, 1]$.

(c) Write the following Matlab function, and save it as '`task2_1.m`'. The syntax of the function is as follows

```
function task2_1(Xtrain, Ytrain, Xtest, Ytest, Ks)
```

The specifications of the script are as follows.

- Runs a classification experiment on the data set using `run_knn_classifier`.

- 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`, and save it with the file name '`task2_1_cm`$\underline{k}$`.mat`', where $\underline{k}$ denotes the number of nearest neighbours (i.e. k) specified in Ks.
- 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)

(d) Run the following:

```
Ks = [1,3,5,10,20];
task2_1(Xtrn, Ytrn, Xtst, Ytst, Ks);
```

In your report, report the information shown on the display.

Task 2.2 [6 marks]

(a) Write a Matlab function that visualises decision regions of k-NN on a 2D PCA plane, where the position of the plane is specified by a point vector. Use the same specifications for plotting as in Task 1.7. The syntax of the function is as follows.

```
function Dmap = task2_2(Xtrain, Ytrain, k, MAT_evecs, MAT_evals, posVec,
nbins)
```

where `k` is the number of nearest neighbours in k-NN.

(b) Run the following for `k=1,3`.

```
N = 2000;
Xtrn2 = Xtrn(1:N,:); Ytrn2 = Ytrn(1:N,:);
Dmap = task2_2(Xtrn2, Ytrn2, k, 'task1_3_evecs.mat', 'task1_3_evals.mat',
mean(Xtrn), 200);
```

and save each `Dmap` as '`task2_2_dmap_`$\underline{k}$`.map`', and save the figure of decision regions as '`task2_2_imgs_`$\underline{k}$`.pdf`'.
In your report, show the figures.

Task 2.3 [7 marks]

(a) Write a Matlab function that transforms data to the ones in 2D PCA space and plots a contour of Gaussian distribution for each class. Save your code as '`task2_3.m`'. The syntax of the function should be as follows.

```
function task2_3(Xtrain, Ytrain)
```

The specifications of the function are as follows.

- Transform `X` to the data in the 2D space spanned by the first two principal components.
- Estimate the parameters (mean vector and covariance matrix) of Gaussian distribution for each class in the 2D space.
- On a single graph, plot a contour of the distribution for each class using plot() function. *Do not use functions for plotting contours such as contour().* The lengths of longest / shortest axis of an ellipse should be proportional to the standard deviation for the axis. (Please note that that contours of different distributions plotted by this method do not necessary give the set of points of the same pdf value.)

(b) Run `task2_3(Xtrn, Ytrn)` and save the figure in '`task2_3_img.pdf`'.
In your report, show the figure.

Task 2.4 [5 marks]

(a) Write a Matlab function that calculates correlation r_{12} on 2D-PCA for each class and for all the classes (i.e. whole data). Note that the whole data (Xtrain shown below) should be used for PCA. Save your code as '`task2_4.m`'. The syntax of the function is as follows.

```
function [Corrs] = task2_4(Xtrain, Ytrain)
```

where `Corrs` is a 11-by-1 vector (double) - Corrs(k) holds the correlation for class k ($k = 1, \ldots, 10$), and Output(11) holds the correlation for the whole data.

(b) Run `task2_4(Xtrn, Ytrn)` and save `Corrs` as '`task2_4_corrs.mat`'.
In your report, show the correlations.

Task 2.5 [6 marks]

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

```
[Ypreds, Ms, Covs] = run_gaussian_classifiers(Xtrain, Ytran, Xtest, epsilon)
```

where `Xtrn`, `Ytrn`, `Xtst`, and `Ypreds` are the same as those in Task1. `epsilon` is a scalar (double) for the regularisation of covariance matrix described in Lecture 8, in which we add a small positive number (ϵ) to the diagonal elements of covariance matrix, i.e. $\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \epsilon I$, where I is the identity matrix.

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

Note that, as opposed to Tasks 2.3 and 2.4 that use PCA, we do not apply PCA to the data in this task.

(b) Write a Matlab function for a classification experiment, and save the code as '`task2_5.m`'. The syntax of the function is as follows.

```
function task2_5(Xtrain, Ytrain, Xtest, Ytest, epsilon)
```

The specifications of the function are as follows.

- 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 '`task2_5_cm.mat`'.
- Copy the mean vector and covariance matrix for Class 10, i.e., `Ms(10,:)` and `Covs(10,:,:)`, to new variables, `M10` and `Cov10`, respectively, in the following manner:

  ```
  M10 = Ms(10,:); Cov10 = Covs(10,:,:);
  ```

  and save them as '`task2_5_m10.mat`' and '`task2_5_cov10.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)

(c) Run `task2_5(Xtrn, Ytrn, Xtst, Ytst, 0.01)`.
In your report, report the information you obtained.

Task 2.6 [6 marks]

(a) Write a Matlab function that visualises decision regions of the Gaussian classifiers on a 2D-PCA plane. Save your code as '`task2_6.m`'. The syntax of the function is as follows.

```
function Dmap = task2_6(X, Y, epsilon, MAT_evecs, MAT_evals, posVec, nbins)
```

where `MAT_evecs`, `MAT_evals`, `posVec`, and `nbins` are the same as in Task 1.7.

(b) Run the following.

```
Dmap = task2_6(Xtrn, Ytrn, 0.01, 'task1_3_evecs.mat', 'task1_3_evals.mat',
mean(Xtrn), 200);
```

and save `Dmap` as '`task2_6_dmap.mat`', and save the the figure of decision regions as '`task2_6_img.pdf`'.
In your report, show the figure.

Task 2.7 [5 marks]

This task aims to investigate the effect of amount of training data on classification performance for Gaussian classifiers.

(a) Write a Matlab function that run an experiment using a subset of training data, and save your code as '`task2_7.m`'. The syntax of the function is as follows.

```
function [CM, acc] = task2_7(Xtrain, Ytrain, Xtest, Ytest, epsilon, ratio)
```

where `ratio` specifies the ratio of training data to use. If it is 0.9, use the first 90% of samples in Xtrain.

(b) Run the following for `ratio = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3`.

```
task2_7(Xtrn, Ytrn, Xtst, Ytst, 0.01, ratio);
```

and save each `CM` as '`task2_7_cm_`$\underline{R}$`.mat`', where $\underline{R}$ = `ratio` * 100, so that '`task2_7_cm_90.mat`' for `ratio=0.9`.
In report, report the result.

Task 2.8 [8 marks]

In this task, you

(a) Write a Matlab function that applies k-means clustering to each class to obtain multiple Gaussian classifiers per class, and carries out classification. Save the function as '`run_mgcs.m`'

The syntax of the function is as follows.

```
function [Ypreds, MMs, MCovs] = run_mgcs(Xtrain, Ytrain, Xtest, epsilon,
L)
```

where L denotes the number of clusters per class.

`MMs`	L*K-by-D matrix (double) of mean vectors.
`MCovs`	(L*K)-by-D-by-D 3D array (double) of covariance matrices.

(b) Write a Matlab function that runs an experiment, and save the code as '`task2_8.m`'. The syntax of the function is as follows.

```
function task2_8(Xtrain, Ytrain, Xtest, Ytest, epsilon, L)
```

The specifications of the function are as follows.

- Calls the classification function.
- 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 '`task2_8_cm_`$\underline{L}$`.mat`'.
- Copy the mean vectors and covariance matrices for Class 1, i.e., `MMs(1:L,:)` and `MCovs(1:L,:,:)`, to new variables, `Ms1` and `Covs1`, respectively, in the following manner:

  ```
  Ms1 = MMs(1:L,:); Covs1 = MCovs(1:L,:,:);
  ```

  Save `Ms1` and `Covs1` as '`task2_8_g`$\underline{L}$`_m1.mat`' and '`task2_8_g`$\underline{L}$`_cov1.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)

(c) Run `run_mgcs(Xtrn, Ytrn, Xtst, Ytst, 0.01, L);` for L=2,5,10.
In your report, report the information you obtained.

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()	run_knn_classifier()
K-means clustering	kmeans()	my_kMeansClustering()
Compute confusion matrix	confusion()	comp_confmat()
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 two tasks, and save your report files in PDF format and name them `'report_task1.pdf'` and `'report_task2.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 for each task.

Create a directory named `LearnCW`, copy all of the requested files to the the directory, but *do NOT put the data set files in it.*

A checklist will be available from the coursework web page. Submit your coursework from a DICE machine using:

```
submit inf2b cw2 LearnCW
```