Inf2B assignment 2
Natural images classification

(Ver. 1.1a)

Submission due: 4pm, Wednesday 30 March 2016

Hiroshi Shimodaira and Pol Moreno

This assignment is out of 100 marks and forms 12.5% of your final Inf2b grade.

Assessed work is subject to University regulations on academic conduct:
`http://www.ed.ac.uk/academic-services/students/undergraduate/discipline/academic-misconduct`
and
`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

You are not allowed to show any written material or code to anyone else. Never copy-and-paste material into your assignment and edit it. High-level discussion without exchange of written materials is fine. You can use the code shown in the lecture notes, slides, and labs of this course.

Programming: Write code in Matlab/Octave or python+numpy+scipy+matplotlib. Your code should run with the software installed on DICE. There are some functions that you should write the code by yourself rather than using those of standard libraries available. See section 5 for details.

If you use Python, replace the filename extension `'.m'` with `'.py'` in the file names shown in the following sections.

Extensions: The School does not normally allow late submissions. See: `http://www.inf.ed.ac.uk/student-services/teaching-organisation/for-taught-students/coursework-and-projects/late-coursework-submission` for exceptions to this rule, e.g. in case of serious medical illness or serious personal problems. Any extension request must be made to the ITO, not the lecturer.

Introduction

This assignment consists of three tasks, Task1 – visualisation of data with PCA, Task2 – classification with k-NN and Gaussian classifiers, and Task3 – visualisation of decision boundaries, using an image data set of house numbers. The data set employed for this assignment is the street view house numbers (SVHN) data set `http://ufldl.stanford.edu/housenumbers/`, whose example is shown in Fig. 1.

Fig. 1: Street View House Number dataset. The digit in the centre indicates the class of each image.

1 Data

Each image in the original SVHN dataset is represented as 32-by-32 pixels in RGB, which has been converted to a grey-scale image and processed to finally give a feature vector of 100 dimension for this assignment. We will use the feature vectors throughout the tasks rather than the pixel data. The pixel data are provided in case you would like to see the original images.

You will work on a data set in which the training set has a size of $10,000$ and the test set has a data size of $1,000$ data points. Each data point is represented as a 100 dimensional feature vector, and given a class label, where labels 1-9 correspond to digits 1-9, and label 10 to digit 0. There are 10 classes in total.

The data is stored in a Matlab file named `'svhn.mat'` located in the directory:

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

where *UUN* denotes your UUN (DICE login name). Note that you are not allowed to give or show your data to anyone else.

You can load the data in Matlab using:

`load('`*DirName*`/svhn.mat')`

where *DirName* is the directory shown above.

The data file contains 6 matrices: `train_features` and `test_features` – the matrices with a feature vector per row (data point), `train_classes` and `test_classes` – the train and test class labels for each data point, and `train_images` and `test_images` – the original pixel image data.

You can also load the data in Python using:

```
import scipy.io data = scipy.io.loadmat('svhn.mat')
```

in which case `data` will be a dictionary containing the 6 matrices.

2 Task1 – Exploratory Analysis [30 marks]

1. Write the code to compute the principal components of a data set. For this you should create a function

   ```
   [EVecs, EVals] = compute_pca(X)
   ```

   which receives as input a N-by-D data matrix X (where N and D denote the number of data points and the dimension of data point, respectively) and returns the eigenvectors (stored in a D-by-D matrix `EVecs`) and eigenvalues $\{\lambda_i\}_{i=1}^{D}$ (stored in a D-by-1 vector `EVals`) of the covariance matrix of X. 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 assignment by putting the following additional constraints, which your `compute_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.

   Save you code as `'compute_pca.m'`. See section 6 for details about the directory structure for submission.

   Remember to subtract the mean of the data before computing the covariance matrix.

   Hint: you will probably find the Matlab function eig helpful. In Python, you can use numpy.linalg.eig

   [6 marks]

2. Let E_2 be a D-by-2 matrix, which holds the two eigenvectors for the largest two eigenvalues. Letting `X = train_features` and using `compute_pca()`, obtain E_2, and project the feature vectors of the training set to the two principal component axes by

   $$X^{PCA} = XE_2.$$

   Show, in your report, the largest two eigenvalues (i.e. λ_1, λ_2), the first five rows of E_2, and the first five rows of X^{PCA}. (NB: the number of significant figures should be no less than six, which should be followed hereafter.)

   Save your code for this as `'apply_pca.m'`.

   [6 marks]

3. Show a scatter plot of the two dimensions of each data point in X^{PCA}. You should colour each point according to the class it belongs to, where each class should have a unique colour.

   Save your code for this as '`show_scatter_plot.m`'.

   [6 marks]

4. Show a plot of the cumulative variance:

$$Y_k = \sum_{i=1}^{k} \lambda_i \quad \text{for } k = 1, \ldots, D$$

   where you put k on x-axis and Y_k on y-axis. The cumulative variance tells us the amount of variance that is expressed with the first k principal components. Note that Y_D gives the total variance of the data, i.e. $\sum_{i=1}^{D} \sigma_{ii}$, where σ_{ii} denotes the variance of x_i (i.e. i'th feature).

   Save your code for this as '`show_cumulative_variance.m`'.

   By looking at the plot determine how many components are necessary to explain at least 90% (rough estimate) of the total variance of the data.

   What can we say about the data from the cumulative variance explained?

   Hint: you will probably find the Matlab function cumsum helpful. In Python, you can use numpy.cumsum.

   [6 marks]

5. Investigation and discussions

   Based on the analysis above, describe your findings and discuss how they might relate to the digit classification in Task2. For higher marks, you would need to run further analysis and give decent discussions, e.g. find two other PCA eigenvectors for which the projected features are also well separated with respect to their corresponding classes, and produce a similar scatter plot this time using these features.

   [6 marks]

3 Task2 – Digit classification [40 marks]

For this task you will use the full feature (i.e. 100-dimensional) space data matrix `train_features` to train the classifiers. `test_features` will be the data with which you will evaluate the classification methods.

3.1 K-nearest neighbours [10 marks]

- Write the code that (i) runs a test experiment on `test_features`, using k-nearest neighbours using the Euclidean distance with `train_features` and (ii) displays the confusion matrix.

  In case of ties, *i.e.* if several classes have the greatest number of points in the k nearest neighbours, choose the nearest point among the tied classes.

  Save the code for this as '`knn.m`'.

- Report the accuracy and confusion matrix for $k = 1$ neighbour.

3.2 Gaussian models [20 marks]

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

- Fit a full covariance Gaussian model to each class and use them to classify the test data. Report the determinant of the covariance matrix of each class, and the confusion matrix and classification accuracy for the test set.

  Save the code for this as '`gaussian_full.m`'.

- Fit a Gaussian model to each class, in which all the classes share the same full covariance matrix. Obtain a linear discriminant function from these models and classify the test dataset using the linear discriminant function (denoted as LDA hereafter). Note that your discriminant function should not have quadratic terms of $\mathbf{x}$. Report the determinant of the shared covariance matrix, and the confusion matrix and classification accuracy for the test set.

  Save the code for this as '`gaussian_lda.m`'.

3.3 Investigation and discussions [10 marks]

Based on the experiments you ran above, describe your findings, including a brief comparison of the differences between the classification methods. Do the classification results reflect their relative advantages and disadvantages? For higher marks, investigate classification errors and discuss what modifications you would try to reduced the errors.

4 Task3 – Visualisation of decision boundaries [30 marks]

In this section we will visualise the decision boundaries defined by the previous classification methods. Since it is not feasible to visualise decision boundaries in a high dimensional vector space, we will use two-dimensional feature vectors by picking out the first two elements of each feature vector of 100 dimension.

For all your training and test data, select the first two features and use the same methods as the previous sections to classify the test data in this 2D-space.

1. Show a table with the classification results for the classification methods used in Task2. How does it compare to the classification accuracy using the full feature space?

   Save your code for running the training and classification tests as '`classify_with2d.m`'.

   [8 marks]

2. Create a function to visualise the decision boundaries by colouring the regions of the features according to the class given by the 1-Nearest Neighbours classifies. You can do so by using a dense 2D grid of points and evaluating your Nearest Neighbour method for each of these point. Produce a plot visualising the decision boundaries for the ten classes, and show a scatter plot of the test set on top of the decision boundary plot, and colour them according to their true classes.

   Save your code as '`show_knn_decision_boundaries.m`'.

   [8 marks]

3. Similarly, write code to visualise decision boundaries of the full covariance model and the LDA model from last section. Again, show the plots with both the coloured decision boundaries and the scatter plot of the test set.

   Save your code for the full covariance model as '`show_full_cov_decision_boundaries.m`', and the one for LDA model as '`show_lda_decision_boundaries.m`'.

   [8 marks]

4. Investigation and discussions.
   How do the shapes of the decision boundaries look for each method? Compare these methods based on the visualisation of these decision boundaries.

   For higher marks, do further analysis and give decent discussions, e.g. show two decision boundary plots in which you only show the decision boundaries and data points corresponding to two classes of your choosing which are clearly separable, and one plot in which they are not. Explain why some classes are harder to separate.

   Hint: One might want to think of the data in terms of its original image representation with respect to their classes.

   [6 marks]

5 Functions that are not allowed to use

Since one of the objectives of this coursework is to implement basic algorithms for machine learning effectively, 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
compute the mean	mean()	MyMean()
compute the covariance matrix	cov()	MyCov()
compute Gaussian probability densities	mvnpdf()	
k-nearest neighbour classification	fitcknn()	
compute confusion matrix	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()

(NB: the list is not exhaustive)

6 Submission

You should submit your work by the deadline given at the head of this document. Your submission is complete if (i) you submit the printed version of your reports to ITO and (ii) you submit your reports and code via the DICE electronic submission system.

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. Make sure that colour figures are really shown in colours in the PDF version of reports.

For the printed version of reports, use separate sheets for the three reports, i.e. do not put more than one report into a single sheet, print them in monochrome, and staple all the sheets of three reports into one. A report for each task should be concise and brief – 1 or 2 pages long.

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. The directory should look like this (the code files may be named differently, and you can put additional code files):

```
|-- LearnCW
   |-- report_task1.pdf
   |-- report_task2.pdf
   |-- report_task3.pdf
   |-- Task1
      |-- compute_pca.m
      |-- apply_pca.m
      |-- show_scatter_plot.m
      |-- show_cumulative_variance.m
   |-- Task2
      |-- knn.m
      |-- gaussian_full.m
      |-- gaussian_lda.m
   |-- Task3
      |-- classify_with2d.m
      |-- show_knn_decision_boundaries.m
      |-- show_full_cov_decision_boundaries.m
      |-- show_lda_decision_boundaries.m
```

Submit it from a DICE machine using: `submit inf2b 2 LearnCW`