Machine Learning & Neural Computation 424 2012/2013
Module leader & Lecturer: Dr Aldo Faisal – Imperial College London

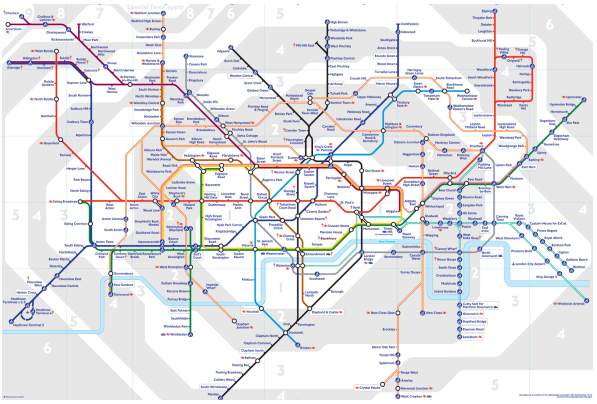
Coursework issue date: 19/10/2012
Coursework submission: 16/11/2012 (online via CATE)
Coursework return & feedback: 14 days after submission

Coursework has to be submitted individually and will be marked as such. You are welcome to discuss during the lab sessions your work with other students and your TAs. All text-based files must contain a comment line identifying the Name and CID of the submitter.

Download and open the coursework zip-file from CATE. The focus of this coursework is supervised learning, and has two questions.

Question A

1-bedroom/studio rents in London



You have been provided with a dataset that contains thousands of rental prices (pcm) for single bedroom properties in London along with corresponding Lat/Long coordinates. Also provided is a set of coordinates for all stations on the London Underground network.

This is a real dataset collected using automated scripts from current sources (October 2012). There may be a number of properties that are not located within Greater London. The aim of this project is to use supervised learning techniques from the course to learn from the supplied sample data the mapping from geographical location, specified as two-dimensional vector x = [latitude longitude], to rental costs per calendar month y = [GBP]. Thus, for an arbitrary location in Greater London we want to know the predicted rental price from your machine learning system. You can choose any of the methods discussed in the course, provided that you implement them yourself from first principles.

Your task is to write two functions, so as to be able to answer the subsequently described subquestions:

1. `trainRegressor.m` – a Matlab function that accepts training inputs **`trainIn`** (a two-column matrix of Longitude and Latitude pairs, where the rows correspond to different training data points) and training outputs **`trainOut`** (a column vector of rent prices, where each row rental price corresponds to the rental location in trainIn). The function is to return a single variable (potentially a structure or matrix) **`params`**.

```
>>params = trainRegressor ( trainIn , trainOut )
```

2. `testRegressor.m` – a Matlab function that accepts testing inputs **`testIn`** (a two-column matrix of Longitude and Latitude pairs, where the rows correspond to different test data points) and **`params`** (the parameter variable created by `trainRegressor`). The function should return a column vector of rental prices for the corresponding locations called `results`.

```
>>results = testRegressor ( testIn , params )
```

In your coursework zip directory you will find the following files:

- `QuestionA.mat` contains the location and price data that you will be using for the exercise, stored as a structure named **`prices`**, with two fields **`rent`** and **`location`** (in latitude and longitude). In addition it also contains a structure named **`tube`**, which contains two fields **`station`** and **`location`** with all London Underground stations.

- `trainRegressor.m` – an empty stub file you have to write in

- `testRegressor.m` – an empty stub file you have to write in

- sanityCheck.m is a matlab function that will check whether your two functions match the specifications we use. Use the function with the following command in matlab:

```
>>sanityCheck ( @trainRegressor  ,@testRegressor )
```

  Note: If your functions do not pass the automatic tests performed by this function we will be unable to automatically evaluate their performance and may deduct marks.

In the following sub-questions you will be required to generate plots from data or your results. Note: Plots that do not have appropriate labels (i.e. the name of dimension and the units in square brackets, e.g. "Rent [£]") will not be counted. Therefore, figures should be exported so that lines and data points are clearly visible when exported and any text in the figures is clearly readable. Use "export figure" to export the figure from Matlab and store it as PNG. Untidy or grainy figures will not be considered.

a) Visualise in a 3D plot (`plot3`) the rental price raw data for Greater London (rotate the view appropriately to be as informative as possible).

b) Write the two functions and train your system with the data set. Your code should be documented with sufficient comments so a stranger is able to follow why and what and you did. Any free parameters that you do not want or cannot learn from the provided date set, have to be set as constants within the `trainRegressor` function. Explain and discuss your solution approach and how you chose any parameters that you set as constants.

c) Test your two functions by calculating the Root Mean Square (RMS) error of your rental price predictions, by performing cross-validation (you will have to choose suitable sizes for the training and testing data chunks). How does the number of training points affect the accuracy of your regression? Hint: acceptable solution will have a RMS lower than £950 per

calendar month.

d) Calculate the predicted rental price at all London tube stations, what is the mean and standard deviation of these rental prices?

e) Use a bar chart (`bar`) plot to visualise the rent profile for the Central Line from Ealing to Stratford.

f) What is the price to rent at Imperial College (Latitude 51.499019°, Longitude -0.176256°), and discuss if you believe the predicted value based on the data and the machine learning strategy used by you.

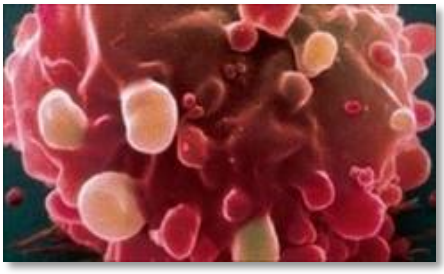
g) What is the rent price at Upminster station, and discuss if you believe the predicted value based on the data and the machine learning strategy used by you.

h) Generate a finely spaced grid of sample points (e.g. you could use the Matlab function `meshgrid`) and visualise the "landscape" of London rental prices using the function `surf` and add the tube stations (but not necessarily all tube stations names) in the plot (using `plot3` and `text`). Rotate the view so that you see London from above, you will thus get a "heatmap". Use `colorbar` to further annotate the plot. This allows you to answer the following questions:
    1. What is your predictor's most expensive location in London and where is it located. Provide price, latitude and longitude and use Google maps to visualise your location.
    2. Where is the price gradient largest - i.e. where is the transition from an expensive area to a cheaper area steepest?

Optional: How good does your machine learner predict your own rent and what is the relative erro

Question B

Bowl cancer classification



Bowel cancer, commonly known as **colon cancer or colorectal cancer**, results from the uncontrolled cell growth in the large intestine or appendix. Colorectal cancer is the third most commonly diagnosed cancer in the world, but it is more common in industrialised countries. Around 60% of cases were diagnosed in the developed world. About 1.23 million new cases of colorectal cancer are clinically diagnosed each year, and killed 608,000 people per annum.

Thus, your mission is to develop an automated way of diagnosing bowel cancer from clinical data, so as to free up time of clinicians for primary care, reduce cost and increase throughput of patient tests. To this end you have been

provided with a Bowel Cancer data set, from a recent clinical screen. Expert clinicians manually annotated this data.

The data is organised inside the structure **`cancer`** with two fields. 1. A n**`input`** matrix of feature row vectors extracted from screening images of cells. There are 30 real valued feature per patient which include a) radius (mean of distances from center to points on the perimeter) b) texture (standard deviation of gray-scale values) c) perimeter d) area e) smoothness (local variation in radius lengths) f) compactness (perimeter^2 / area - 1.0) g) concavity (severity of concave portions of the contour) h) concave points (number of concave portions of the contour) i) symmetry j) fractal dimension ("how jagged is the contour"). 2. A binary column vector **`output`**, with the expert annotations of the images {0=malign, 1=benign} diagnosis.

Your task is to build two classifiers for the Bowel Cancer Data set using

1. the k-nearest neighbours alogrihtm with respective functions `trainClassifierKNN.m` and `testClassifierKNN.m`

2. using the Bayesian generative classification approach (a Gaussian per class) `trainClassifierGen.m and testClassifierGen.m`

For each of the two classifiers do the following:

Build a classifier (MatLab) for the Cancer Data Set using one of the methods described in the lectures

1. Describe your classifier + supply the Matlab code.
   Write a function *Classifier* that takes as arguments an *input* vector and a matrix/vector of *parameters* and returns the predicted *class*. Call the m-file Classifier.m, it's first line should look like this:

   function class = Classifier(input,parameters)

   Write a function TrainClassifier(inputs,output) that takes as arguments an input matrix of data points (each row one data point, the columns correspond to the dimensions of the data point) and the training class labels (desired class labels) as a column vector of numbers. The function should return the parameters required by your Classifier function. Call the m-file TrainClassifier.m, it's first line should look like this:

   function class = TrainClassifier(inputs, output)
2. Test it's performance by
   a. Dividing data into equally sized training and test data set. Use only the training data to train your classifier. Report your classifier's performance (% correct) on the test data set.
   b. Do "leave-one-data-point-out" cross validation. Train your classifier with the whole data set, leaving out one test data point. Measure the performance. Repeat this for each data point, report the averaged performance across all runs.
3. Explain what properties of the data are responsible for your classifiers performance.

Make sure that you conform to the standards set out above, if our automatic testing fails due to not keeping to the specifications, marks will be automatically deducted.