# Coursework : Practical Data Analysis using Python

## Overview

The coursework for the Intelligent Data Analysis and Probabilistic Inference Course has two objectives. Firstly it is intended to help you fully understand some of the algorithms covered in the course by doing some practical data analysis work, and secondly it serves as an introduction to Python which is an excellent environment for doing numerical analysis. Python has a powerful syntax that allows you to write code that looks very much like the equations found in the notes. It provides an interface to a highly robust and fast numerical programming library. The syntax has some similarities to Matlab but Python is available for free. There are many good online tutorials which you may want to work through if you find the coursework notes too terse.

## Dates

The coursework is divided into four parts with hand in dates spaced fairly evenly throughout the term. The suggested start dates and the hand in dates are as follows:

| Part | Start | Hand In |
|---|---|---|
| 1 | after lecture 2: (12th Oct) | 26th Oct |
| 2 | after lecture 6: (2nd Nov) | 16th Nov |
| 3 | after lecture 8: (9th Nov) | 30th Nov |
| 4 | after lecture 14: (30th Nov) | 14th Dec |

Although the fourth exercise can be started after lecture 14, it depends a lot on the lecture 15 material, so you may need to read this in advance of the lecture to progress far with it.

## Group Working

You are encouraged to work in pairs or in groups of three. Group working can be enjoyable and is an effective way of learning providing everybody in the group participates. If you work in a group then you submit just one solution to CATE for each exercise. You don't nave to be in the same group for all four courseworks. For example you could do number one by yourself and the other three in a group with two of your friends.

### Submitting work

All four courseworks will be submitted electronically only. For each you will submit two files. One will be your python script in text format. The other will be a results file. For courseworks 1 and 3 this is a simple text file, but for courseworks 2 and 4 you will need to submit some images in a .pdf file.

## Assessment

There is a total of 100 marks available for the coursework. The handouts for each exercise give you a breakdown of the marks available for each part. The intention is that you can score up to 60% fairly easily, and with a bit more effort get an A grade, however after that the going gets harder. Some of the parts are marked as hard, so you may choose to leave them out if you are under pressure to meet other deadlines. As a compensation to those who want to work alone each exercise will have two bonus marks for individual solutions and one bonus mark for groups of 2. This means that if you work in a group of 2 you can score a maximum of 96% and in a group of 3 you can score a maximum of 92%. On the other hand you may well be able to complete more of the exercises in a group.

## Coursework Files

The files that you need for the coursework are available from the course web page:
http://www.doc.ic.ac.uk/ dfg/ProbabilisticInference.
You should download copies of these to your working directory. The files are as follows:

- There are two python scripts with extension .py. One is a library of functions for doing input and output of different kinds of data. The other is a skeleton solution for the four courseworks, to which you will add your own code. You should look through these to work out what they are doing.
- There are six face image files which are the data needed for part four of the coursework. These are named a.pgm to f.pgm. They are taken from a standard open source database used as a benchmark for face recognition.
- There is a file called "PrincipalComponents.txt" which is face data but in a different format. You will need this for coursework 4.
- There are two files of discrete data. One is a small and highly dependent data set for recognition of four developmental stages of neurones grown in culture from five measured variables. The other is a large data set of measurements on patients suffering from Hepatitis C.

The format of the last two files is as follows:
*Number of Variables*
*Number of Root nodes*
for each variable *The number of states of each variable*
*Number of Data Points*
for each data point
    for each variable *the state*
All data items (shown *in italics*) are integers and are separated either by a space or a carriage return. The states are numbered consecutively from 0 upwards. Example:

| | | | | |
|---|---|---|---|---|
| 4 | | | | the data file has four variables |
| 2 | | | | there are two root nodes (the root nodes will always be placed first in the variables) |
| 5 | 2 | 4 | 2 | variable 1 has 5 states, variable 2 has 2 states etc |
| 7 | | | | there are seven data points |
| 2 | 0 | 3 | 1 | data point 1 variable 1 (a root) is in state 2, variable 2 (a root) is in state 0 etc. |
| 4 | 1 | 3 | 1 | data point 2 |
| 4 | 0 | 0 | 1 | etc. |
| 0 | 0 | 3 | 0 | |
| 3 | 0 | 2 | 1 | |
| 1 | 1 | 2 | 1 | |
| 2 | 0 | 2 | 1 | |

## Working Environment

The suggested environment for doing this coursework is Ubuntu Linux, but you can also do it in the windows and mac operating systems. On Ubuntu it is convenient to use 'Kate' which you can start from the Applications/Other menu, or by right clicking on a .py file and choosing 'Kate' from the 'open with' option. When kate is running, open the terminal at the bottom of the window by clicking on it. You can then edit the file in the top window, save it and then execute it by typing 'python myprog.py' in the terminal.

## Coursework Part 1: The Naive Bayesian Network

In the first coursework we will look at calculating joint and conditional probability tables (link matrices) and making inferences with a naive Bayesian network.

Take a look at the first four functions in IDAPICourseworkLibrary.py to make sure you understand what is going on. You may find the python syntax self explanatory, but if not you can find several good online tutorials through google. The first function reads a data file in the above format, extracting the data so that it can be used. It is extensively commented to help you understand what is going on. The second writes a data array to a file, appending it to what is there already. The data is written out in a fixed floating point format which is suitable for printing probabilities. The third appends a list (one dimensional array or vector) to a text file and the fourth appends a string to a text file. These allow you to save your results in a text file and add comments to them. Thus you can put together your results in a suitable form for submitting your solution to CATE.

Now take a look at the file IDAPICourseworkSkeleton.py. The first five functions are to be completed for coursework 1 as detailed below. They are all quite short, the longest part in my solution is 11 lines.

### Task 1.1: 4 marks

Complete the definition of function Prior. It should calculate the prior distribution over the states of the variable passed as the parameter 'root' in the data array. The first line simply sets up an array for the result with zero entries.

### Task 1.2: 5 marks

Complete the definition of CPT which calculates a conditional probability table (or link matrix) between two variables indicated by parameters varP (the parent) and varC (the child). In the skeleton the conditional probability table is simply initialised to zeros.

### Task 1.3: 4 marks

Complete the definition of function JPT which calculates the joint probability table of any two variables. The joint probability is simply the frequency of a state pair occurring in the data. So for state $a_0$ of variable $A$ and state $b_3$ of variable $B$, if $N(a_0\&b_3)$ is the number of data points containing both $a_0$ and $b_3$ then $P(a_0\&b_3) = N(a_0\&b_3)/noDataPoints$. The table should be arranged so that for $P(A\&B)$ the states of $A$ are rows and the states of $B$ are columns.

### Task 1.4: 4 marks

Complete the function JPT2CPT which calculates a conditional probability table from a joint probability table. This is purely a matter of normalising each column to sum to 1. In effect we use the equation $P(A|B) = P(A\&B)/P(A)$.

### Task 1.5: 5 marks (harder)

Finally complete the function Query which calculates the probability distribution over the root node of a naive Bayesian network. To represent it in Python we will use a list containing an entry for each node (in numeric order) giving the associated probability table: [prior, cpt1, cpt2, cpt3, cpt4, cpt5]. You can calculate the prior of the root and the conditional probability tables between each child variable and the root using your solutions to Tasks 1 and 2.

### Results File

You should finish by writing a main program part at the end of the skeleton file. Some example code is there to help you do this. You should use the Neurones.txt data set, and create a results file containing:

1. A title giving your group members
2. The prior probability distribution of node 1 in the data set

3. The conditional probability matrix $P(2|0)$ calculated from the data.

4. The joint probability matrix $P(2\&0)$ calculated from the data.

5. The conditional probability matrix $P(2|0)$ calculated from the joint probability matrix $P(2\&0)$.

6. The results of queries [0,4,0,0,0,5] and [3, 6, 5, 2, 5, 5] on the naive network

Rename your python file "IDAPICoursework01.py" and submit it and your results file named "IDAPIResults01.txt" using CATE. (NB Currently CATE does not recognise the extension .py. I hope to get this fixed by the first hand in date, but if not you may need to name your file "IDAPICoursework01.txt"

## Some confusing things when starting python

Python uses the list as its primary data structure, and this provides a very flexible mechanism for representing and manipulating data. However, in numerical analysis we are primarily concerned with using vectors and matrices. The python module "numpy" provides support for this activity. For example, if we want to use a 2 by 3 array we would define it as a list of lists:

myList = [ [3,4,7], [2,1,3]]

We can convert it to a Matrix by using

myMatrix = numpy.array(myList)

This function creates a new data structure with attributes like rows (myMatrix.shape[0]) and columns (myMatrix.shape[1]), and methods, for example:

myMatrix.transpose()

Numpy contains many other useful functions, for example to compute a matrix product:

myMatrixProd = numpy.dot(myMatrix, myMatrix.transpose())

A python module is simply a set of python function definitions. In order to use a module you must import it. There are different ways of doing this. For example:

import numpy

allows you to use all the functions defined in numpy, but you must explicitly reference them as numpy functions. For example we write *numpy.dot* for the dot product. Numpy contains a reference to the linear algebra module named *linalg* and in coursework part four you need the eigenvector function *eig*. You reference this by writing *numpy.linalg.eig* You can alternatively write:

from numpy import *

If you do this you can use the functions in numpy without the reference, ie just *dot* for the dot product.

## Coursework Part 2: The Maximally Weighted Spanning Tree

This coursework is concerned with the spanning tree algorithm for finding a singly connected Bayesian network from a data set. The material you need will be completely covered by Lecture 6 of the course. You can use the same skeleton program that was set up for Coursework 1, continuing to fill in code where indicated. You will need to use the function that you wrote to compute the joint probability distribution of a pair of variables. If your original solution was wrong and you are stuck then email me and I will give you some working code to do this (after the hand in date for coursework 1).

### Task 2.1: 6 marks

Complete the function "MutualInformation" which calculates the mutual information (or Kullback Leibler divergence) of two variables from their joint probability table. The process involves marginalising the joint probability table and then applying the formula as described in Lecture 6.

### Task 2.2: 5 marks

Complete the function "DependencyMatrix" which uses mutual information as a measure and creates a symmetric matrix showing the pairwise dependencies between the variables in a data set.

### Task 2.3: 6 marks

Complete the function "DependencyList" which turns the dependency matrix into a list or arcs ordered by their dependency. Your list items should be triplets: [dependency, node1, node2]. Using your dependency list draw by hand the network for the HepatitisC data set. Make an image file of your network (either by scanning your hand drawing or using a drawing package (eg powerpoint or open office)).

### Task 2.4: 6 marks (very difficult)

Starting with a dependency list find the maximally weighted spanning tree automatically, and append it as a list to your results file. Don't wory about finding the causal directions.

### Results File

You should finish by writing a main program part at the end of the skeleton file which will add the following items to your results file:

1. A title giving your group members
2. The dependency matrix for the HepatitisC data set
3. The dependency list for the HepatitisC data set.
4. Ths spanning tree found for the HepatitisC data set (if you attempted task 2.4).

Assemble the results listed above with the network picture into a .pdf file (You can do this with word, open office or latex). Rename your python file "IDAPICoursework02.py" and submit it and your results file named "IDAPIResults02.pdf" using CATE.

## Coursework Part 3: The Minimum Description Length Metric

This coursework is concerned with the minimum description length measure of a Bayesian network and an associated data set. The course material will have been covered by lecture 8. Continue filling up the skeleton solution you used in courseworks 1 and 2. You will need to use your function CPT from coursework 1, so if you haven't got a correct version email me and I will give you one.

### Task 3.1: 4 marks

Complete the function CPT_2 which computes a conditional probability table (or link matrix) for variables with two parents.

### Task 3.2: 3 marks

In the skeleton solution there is an example definition of a singly connected Bayesian network with six variables. It could be used to model the Neurones data set, but it isn't the maximally weighted spanning tree, and assumes two root nodes (0 and 1). It uses two lists, each with an entry for each node, the nodes being ordered numerically. The arcList contains the connectivity. Each variable has an associated sub-list with the first item on the list being the variable index, and the next items the indices of its parents. The cptList list contains probability tables asasociated with each node.

Using your solution to coursework 2 write a network definition for the HepatitisC data set.

### Task 3.3: 4 marks

Complete the definition of function MDLSize which calculates the size of a network, given the definition of the network in the two list form described above.

### Task 3.4: 4 marks

Complete the definition of the function JointProbability which calculates the joint probability of a single point, supplied as a list eg [1,0,3,2,1,0], for a given Bayesian network.

### Task 3.5: 4 marks

Complete the definition of function MDLAccuracy which calculates log likelihood of the network given the data as described in the lectures.

### Task 3.6: 4 marks (hard)

Write a function to find the highset scoring network formed by deleting one arc from the spanning tree.

### Results File

You should finish by replacing the main program part at the end of the skeleton file with one which will add the following items to your results file:

1. A title giving your group members
2. The MDLSize of the your network for Hepatitis C data set
3. The MDLAccuracy of the your network for Hepatitis C data set
4. The MDLAccuracy of the your network for Hepatitis C data set
5. The score of your best network with one arc removed

Rename your python file "IDAPICoursework03.py" and submit it and your results file named "IDAPIResults03.txt" using CATE.

## Coursework Part 4: Principal Component Analysis

This coursework is concerned with covariance estimation and finding principal components. It uses numpy's linear algebra module (linalg) and its image handling capabilities. There is no dependence on courseworks 1-3. The data to be used is the HepatitisC data set, the set of six face images a.pgm .. f.pgm, and the file PrincipalComponents.txt. The file IDAPICourseworkLibrary contains three functions which will read images and turn them into data sets and vice versa.

### Task 4.1: 3 marks

Complete the function Mean which calculates the mean vector of a data set represented (as usual) by a matrix in which the rows are data points and the columns are variables.

### Task 4.2: 4 marks

Complete the function Covariance which calculates the covariance matrix of a data set represented as above.

### Task 4.3: 3 marks

For this part you will need to use the file named PrincipalComponents.txt which is on the course web page. It contains a set of ten principal components (the last having a zero eigenvalue) calculated from ten different images of the subject in image c.pmg. You can read it with the function ReadEigenfaceBasis() which is in the file IDAPICourseworkLibrary.py. It returns an array where each row is an eigenface.

Complete the function CreateEigenfaceFiles to create image files of the principal components (eigenfaces) in the format returned by ReadEigenfaceBasis(). You can use the IDAPICourseworkLibrary file SaveEigenface to do the image handling. Don't use the file foramt .pgm (eg instead choose .jpg or .png) when creating these image files. This is to avoid problems later. You can create a series of file names for the different eigenfaces within a loop using python's powerful string concatination feature, for example: filename = "PrincipalComponent" + str(j) + ".jpg"

### Task 4.4: 3 marks

Complete the function ProjectFace, which reads one image file (use c.pmg for your example) and projects it onto the principal component basis.

### Task 4.5: 3 marks

Complete the function CreatePartialReconstructions which generates and saves image files of the reconstruction of an image from its component magnitudes. Generate files of the mean face, the mean face plus first principal component ... and so on up to the mean face plus all principal components.

### Task 4.6: 7 marks (hard)

Complete the function PrincipalComponents which performs pca on a data set. To do this you will need to use the Kohonen Lowe method outlined towards the end of lecture 15. Your function should return the orthonormal basis as a list of eigenfaces equivalent to the one returned by ReadEigenfaceBasis(). Check out the function numpy.linalg.eig before you try to compute the eigenvectors yourself!

To check out your Principal components you should use the face images a.pgm to f.pgm. You can create an image data set using the IDAPICourseworkLibrary, for example: imageData = array(ReadImages()). You can find the mean image from your solution to 4.1.

### Results

Create a results file as before with the following items.

1. A title giving your group members

2. The mean vector Hepatitis C data set
3. The covariance matrix of the Hepatitis C data set
4. The component magnitudes for image “c.pgm” in the principal component basis used in task 4.4.

Put together a file “IDAPIResults04.pdf” which contains the above results together with the eigenface images you created in task 4.3 and the partial reconstructions of image c.pgm created in task 4.4. If you did Task 4.6 include the eigenfaces found in the basis you calculated from the images and the partial reconstructions of image c.pgm using that basis. Submit it with your Python code named “IDAPICoursework04.py”

Breath a hugh sigh of relief and begin your Christmas celebrations.