Scientific Python CSE2682

Coursework 1. The Naive Bayesian Network

Coursework Files

The files that you need for the coursework are available from the course web page under directory`Рубежный контроль 1` 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 coursework, to which you will add your own code. You should look through these to work out what they are doing.
- File `data.txt` of discrete data. This is a highly dependent data set for some kind of process with 4 stages and five measured variables.

The format of the data file is as follows: - Number of Variables - Number of Root nodes - For each variable The number of states of each variable - Number of Data Points - Data points - for each variable `the state`

All data items are integers and are separated either by a space or a carriage return. The states are numbered consecutively from 0 upwards. Example:

Row	Description
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	data point 3
...	etc.

Tasks

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 CourseworkLibrary.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. 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.

Task 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 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 3: 5 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 4: 5 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(B)`.

Task 5: 6 marks

Finally complete the function Query which calculates the probability distribution over the root node of a naive Bayesian network. To represent a naive network 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`. A query is a list of the instantiated states of the child nodes, for example `[1,0,3,2,0]`. The returned value is a list (or vector) giving the posterior probability distribution over the states of the root node, for example `[0.1,0.3,0.4,0.2]`.

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 `data.txt` data set, and create a results file containing:

1. A title giving your full name
2. The prior probability distribution of node 0 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 `[4,0,0,0,5]` and `[6, 5, 2, 5, 5]` on the naive network

Rename your python file “Coursework01.py” and submit it and your results file named “Results01.txt” using GitHub.