**Homework Assignment #2  
Due Monday, 02/15, before class**

**Part 1**

For this part of the homework, you are to write a program that implements both **Naive Bayes** and **TAN** (tree-augmented Naive Bayes). Specifically, you should assume:

* Your code is intended for binary classification problems.
* All of the attributes are discrete valued. Your program should be able to handle a variable number of attributes with possibly different numbers of values for each attribute.
* You use Laplace estimates (pseudocounts of 1) when estimating all probabilities.

For the TAN algorithm. Your program should:

* Use [Prims's algorithm](https://en.wikipedia.org/wiki/Prim%27s_algorithm) to find a maximal spanning tree (but choose **maximal** weight edges instead of minimal weight ones). Initialize this process by choosing the first attribute in the input file for *Vnew*. **If there are ties in selecting maximum weight edges, use the following preference criteria: (1) prefer edges emanating from attributes listed earlier in the input file, (2) if there are multiple maximal weight edges emanating from the first such attribute, prefer edges going to attributes listed earlier in the input file.**
* To root the maximal weight spanning tree, pick the first attribute in the input file as the root.

Your program should read files that are in the [ARFF](http://weka.wikispaces.com/ARFF+%28stable+version%29) format. In this format, each instance is described on a single line. The feature values are separated by commas, and the last value on each line is the class label of the instance. Each ARFF file starts with a header section describing the features and the class labels. Lines starting with '%' are comments. See the link above for a brief, but more detailed description of the ARFF format. Your program needs to handle only discrete attributes, and simple ARFF files (i.e. don't worry about sparse ARFF files and instance weights). Example ARFF files are provided below. Your program can assume that the class attribute is named 'class' and it is the last attribute listed in the header section.

The program should be called bayes and should accept four command-line arguments as follows:  
bayes <train-set-file> <test-set-file> <n|t>  
where the last argument is a single character (either 'n' or 't') that indicates whether to use naive Bayes or TAN.

If you are using a language that is not compiled to machine code (e.g. Java), then you should make a small script called bayes that accepts the command-line arguments and invokes the appropriate source-code program and interpreter. More instructions below!

Your program should determine the network structure (in the case of TAN) and estimate the model parameters using the given training set, and then classify the instances in the test set. Your program should output the following:

* The structure of the Bayes net by listing one line per attribute in which you list (i) the name of the attribute, (ii) the names of its parents in the Bayes net (for naive Bayes, this will simply be the 'class' variable for each attribute) separated by whitespace.
* One line for each instance in the test-set (in the same order as this file) indicating (i) the predicted class, (ii) the actual class, (iii) and the posterior probability of the predicted class.
* The number of the test-set examples that were correctly classified.

You can test the correctness of your code using [lymph\_train.arff](http://pages.cs.wisc.edu/~dpage/cs760/lymph_train.arff) and [lymph\_test.arff](http://pages.cs.wisc.edu/~dpage/cs760/lymph_test.arff). [This directory](http://pages.cs.wisc.edu/~dpage/cs760/hw2-outputs/) contains the outputs your code should produce for each data set, and some additional files showing the intermediate calculations for TAN.

**Part 2**

Plot a [learning curve](https://en.wikipedia.org/wiki/Learning_curve)for both methods. You should plot points for training set sizes of 25, 50, and 100 instances. For each training-set size, randomly draw 4 different training sets and evaluate each resulting model on the test set. For each training set size, plot the average test-set accuracy. Be sure to label the axes of your plot. **Submit the learning curves in a PDF file.**

**Submission Instructions**

Create an executable that calls your program. Instructions to create the executable can be found [here](http://pages.cs.wisc.edu/~dpage/cs760/instructions.html).   
You should electronically hand in your runnable program and source code as follows:   
Collect and compress all files needed to run your code as into zip file named <Wisc username>\_HW2.zip.   
Upload this zip file as Assignment 2 at the [course Moodle](https://ay15-16.moodle.wisc.edu/prod/course/view.php?id=1075).

**You need to ensure that your code will run, when called from the command line as described above, on the CS department Linux machines.**