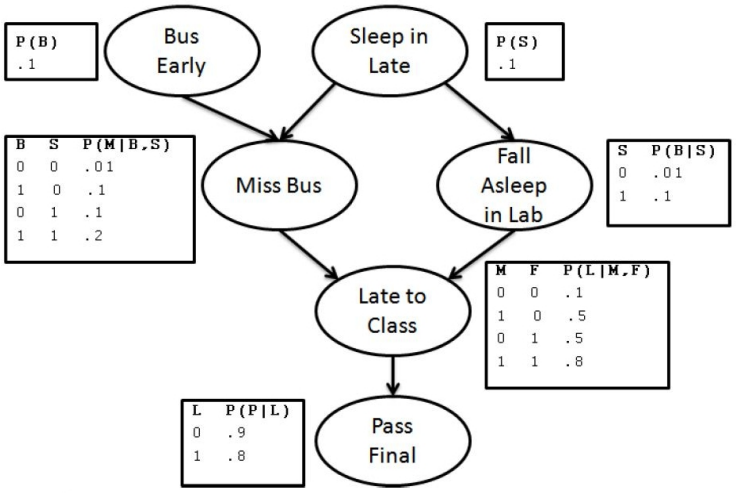
CSEP 573 Applications of Artificial Intelligence Winter 2011
Assignment 3
Due: Wednesday March 9, 6:30PM

Q 1: [20 points] In the following Bayesian network



q1.a: **[3 points]** Are Bus_Early and Sleep_in_Late independent?

q1.b: **[3 points]** Is Bus_Early conditionally independent of Pass_Final given Late to Class?

q1.c: **[3 points]** Is Bus_Early conditionally independent of Fall_Asleep_in_Lab given Miss_Bus and Sleep_in_Late?

q1.d: **[4 points]** Use variable elimination to compute probability of Late_to_Class given Fall_Asleep = true and Bus_Early = false.

q1.e: **[3 points]** We are using likelihood weighting for computing probability distribution of Late_To_Class given Fall asleep in lab = true and Bus_Early = false. What would be the weight of the sample with the values:

```
Bus_Early = false
Sleep_in_Late = true
Miss_Bus = false
Fall asleep in lab = true
Late_To_Class = false
Pass_final = true
```

q1.f: **[4 points]** We are using Gibbs sampling to compute probability distribution for Fall_Asleep_in_Lab given Bus_Early = true. Our current sample has the values:

```
Bus_Early = true
Sleep_in_Late = false
Miss_Bus = true
Fall_Asleep_in_Lab = false
Late to Class = false
Pass_Final = true
```

Lets say our sampler selects Miss_Bus. What is the probability we flip the value of this variable?

Q 2: [25 points] Each day the professor observes whether the students sleep in class and whether they have red eyes. The professor has the following theory.

The prior probability of getting enough sleep, with no observations, is 0.7.

Probability of getting enough sleep at night is 0.8, given that the student got enough sleep the previous night, and 0.3 if not.

Probability of having red eyes is 0.2 if student got enough sleep and 0.7 if not.

Probability of sleeping in class is 0.1 if the student got enough sleep, and 0.3 if not.

q2.a: **[5 points]** Model this problem as a hidden Markov Model.

q2.b: **[5 points]** The professor observes a student for three days and finds the evidence as follows.

```
Day 1: no red eyes, not sleeping,
Day 2: red eyes, not sleeping,
Day 3: red eyes, sleeping.
```

What is the probability that the student actually got enough sleep **on the third day** given these observations.

q2.c: **[15 points]** Simulate Viterbi algorithm and compute the most likely sequence of student's sleep? Show all your steps in simulating the algorithm.

Q 3: [5 points]

Suppose you are running a learning experiment on a new algorithm. You have a data set consisting of 25 examples of each of two classes. You plan to use leave-one-out cross validation.

As a baseline, you run your experimental setup on a simple majority classifier (i.e. a `dumb' classifier which ignores the input features and uniformly outputs the class that is in the majority in the training set).

You expect the majority classifier to score about 50% on leave-one-out cross validation, but to your surprise, it scores zero. Can you explain why?

Q 4: [extra credit points] Odd cookie Out. You have twelve cookies. Eleven of them weigh the same however the twelfth is the odd one, either lighter or heavier, but we don't know which one. You have a weighing pan which can tell you which side is heavier or whether the two sides have the same weight.

q4.a: Use only three weighings to find the odd cookie out and whether it is heavier or lighter.

q4.b: Can you track the entropy of each situation in each weighing? Can you use this idea to find a lower bound on the number of weighings if there are a total of n cookies and you want to find the odd one out.

Q 5: [extra credit points]

You are given results from a set of experiments. In each experiment, one of the two dice were rolled 20 times and observations taken. Unfortunately, the experimenter forgot to annotate which die he rolled in each experiment.

Each die is biased with different unknown probabilities of getting 1-6. Output the best estimate of probabilities of 1-6 for both the dice.

Also, annotate the experiments with the most likely die used to generate them. Explain how you generated these values. The observations are:

51124513353231115451
55215666134526666626
15124666435461666263
42416454331556332252
11215135454446361114
22142633516334223116
66643666566156412646
43612443113354266262
53532642445163432412
31312666216431144511
46616616616231566666
52612646232644253311
15153611342143525326
22333361145115464221
14345244352633242333
52664436514666255154
24315122534251143452
52436561466251355556
65131361615432213621
44166445456333551414
13532156653365261424
56131565132532626412
64564634133641125445
21461423355564661123

Q 6: [50 points] Programming Project: Text Classification

Description of the Programming Project

Your Goal

This assignment will give you some experience with Text Classification using Weka, an open source Machine learning tool.

We provide you with training and development sets from the popular 20 newsgroups dataset. The goal is to predict for a given post, which of the newsgroups it belongs to. This is a multi-class classification problem, which is in general a more difficult than concept learning. The headers have been stripped out, since they contain the names of the newsgroups.

We also provide you with some existing code that runs Naive Bayes over the dataset using cross validation. Treat this as baseline try and improve your classifier to get the best model. Some suggestions on how you can improve the classifier are:

a. Insights from text: remove words with no information content (e.g., the, this), and very infrequent words. You can find a list of stop words online. You can also reduce sparsity by running a stemmer on the words (use any freely available stemmer, e.g. opennlp)

b. Insights from NLP: instead of words use bigrams or trigrams. Make sure you know how to handle the sparsity, because many of the trigrams/bigrams may be rare.

c. More insights from NLP: run a part of speech tagger and annotate a word with its POS. For example if 'list' is used in noun sense use 'list_NN' as a feature as opposed to 'list_VP' which represents 'list' used in the verb phrase sense. You may use any freely available part of speech tagger (e.g., opennlp).

d. Insights from information retrieval: instead of simply using the presence or count of a word in the document, use its tfidf score.

e. Learning algorithms: try different algorithms for learning classifiers. As a start try SVM, random forests and multilayer perceptron.

f. Parameter search: almost all algorithms have many parameters. Try and do some parameter search and find the best parameters to your algorithm of choice.

g. Haha! More NLP: Use features from Wordnet to add semantic information about words in the features. You can download Wordnet online as well as the java library which implements Wordnet functions. This is tricky -- so pick this only if you have prior NLP/Wordnet experience.

h. Training Data Size: vary the amount of training data you give to the classifier

i. Feature selection: do a greedy feature selection (or rejection) to automatically choose the right subset of features.

j. Your favorite.

Pick any **three** of the above and perform a basic evaluation on how adding this feature affects performance. For example, for learning algorithms output precision, recall, f-measure of the different algorithms you try. For parameter search make a graph of performance versus different parameter values. For bigrams/trigrams see how n-grams' performance varies for n=1,2 and 3. Besides reporting the results interpret these hopefully to explain/justify the performance variations that you see. You could also give anecdotal examples to prove your points better.

WEKA and Newsgroup Data on attu

There is a lot of information on Weka available, but it can be difficult to find all the details needed to accomplish a particular task. To make your job easier, we have set up a directory for this assignment on attu:

/projects/instr/11wi/csep573/assignment4

There are subdirectories for the weka code, the newsgroup files, and several scripts. You should have read access, but not write access to these directories. The scripts produce files, so you need to copy them to your own subdirectory in order to use them. You do not need to copy the program or data directories, since the scripts refer to the course assignment directory. Please try out the sample scripts soon, so we can find out if there are unexpected problems.

The `loadData.sh` script should be run first. It uses a Weka program to take all the files in a data directory, and import them into a single Weka `..arff` file. It uses subdirectory names as categories, and saves each file as a single string. To use most classifiers, you have to filter this file to convert the strings to word vectors. This is also done in the `loadData.sh` script. You can examine the resulting `.arff` files with a text editor.

Note that if you are going to classify separate test data, any filters that you apply must filter both the training data and the test data together in batch mode. This is what is being done with the `-b` option in `loadData.sh`

The `crossBayes.sh` script shows a basic classifier in action. To get a description of the options, you can run the `classifierHelp.sh` script.

```
#!/bin/bash

A4=/projects/instr/11wi/csep573/assignment4
WekaInstallDir=$A4/weka
CLASSPATH=$CLASSPATH:$WekaInstallDir/weka.jar

TRAIN="train.arff"

if [ ! -f $TRAIN ] ; then
        echo Training arf does not exist: $TRAIN
else
        time java weka.classifiers.bayes.NaiveBayes -x 4 -o -c 1 -t train.arff
fi
```

The `-x 4` says to do 4-fold cross validation. The `-o` says to output statistics, but not the classifier. The `-c 1` is needed because the filter we needed in `loadData.sh` messed up the order of our categories.

The `runTest.sh` script, along with `loadData.sh` are models for what you will turn in. The main difference between `runTest.sh` and `crossBayes.sh` is that the former includes a `-T` argument to indicate separate test data. Presumably you will not use naive Bayes as your classifier, and you will probably modify or add filters. Make sure an filters are done in batch mode for both training data and test data.

Training, Development and Test Data

The scripts that you turn in should test your classifer on the test data in:

/projects/instr/11wi/csep573/assignment4/Data/573groups-test

This is what the sample scripts do now.

When we run your code, we will replace this test data directory with different data - the real test data. What is there now is just a copy of the 573groups-dev data.

These scripts are specific to the directory layout on attu. If you want to work on a different machine, you will need to change the values of the directory variables. Be sure to turn in the attu version.

WEKA references

Weka Home Page
Weka's Wiki
A Weka Primer
Text Categorization with Weka
Stopwords
Stemmers
Classifiers
Using Weka in Java
Java Classifier Tutorial
Python module to make .arff files from within NLTK
"use the -c option"
"use the -b option"

Grading for the Programming Project

[30 points] The analysis of the three options you choose above.
[20 points] Hidden dataset evaluation. We have a held out test set. Submit your best model and feature generation code. We will evaluate your best classifier using

f-measure on our dataset. Your score will be proportional to the performance of your submitted classifier.
[extra credit points] If you discover insightful features that work well.
[extra credit points] If you choose an interesting experiment that we have not suggested above.
[extra credit points] If your code is one of the top 2 performers in the class.
[extra credit points] If you do the programming assignment without a partner.

What to Turn In

Use the Class Dropbox to turn in a single file: "a4submit.zip". This should contain the files needed to run a classification test - at least your modified versions of loadData.sh and runTest.sh, plus any other support scripts necessary. Also include a single .pdf file called assignment4Report.pdf with your writeup You should not need to include any data files, since we will use those in the course directory on attu.