Assignment 3

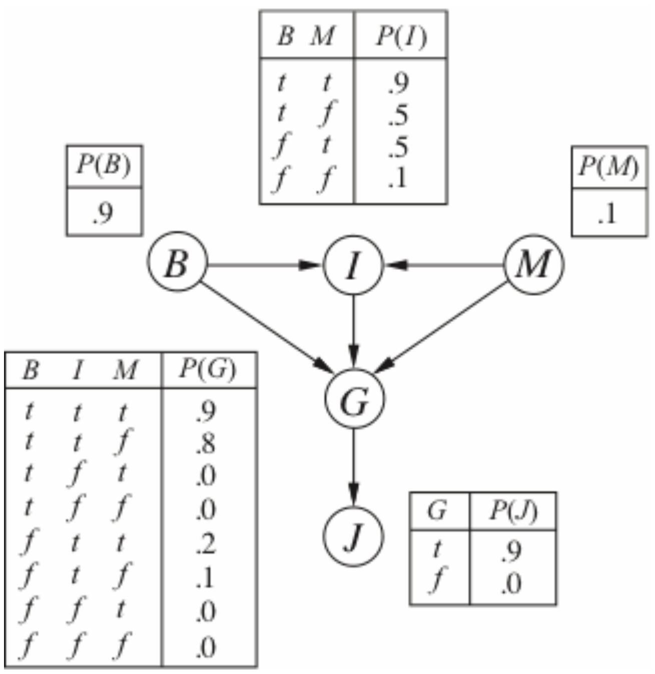
Comp 560 Artificial Intelligence

Due: Oct 29, 2015, 11:59pm

In this assignment, you will answer written questions and implement algorithms related to Probability and Bayes Networks.

You are free to use any high-level programming language you are comfortable with and to work in groups of up to 3 people.

Bayes Nets____(written questions)______________________________



$P(B)$
.9

B	M	$P(I)$
t	t	.9
t	f	.5
f	t	.5
f	f	.1

$P(M)$
.1

B	I	M	$P(G)$
t	t	t	.9
t	t	f	.8
t	f	t	.0
t	f	f	.0
f	t	t	.2
f	t	f	.1
f	f	t	.0
f	f	f	.0

G	$P(J)$
t	.9
f	.0

Consider this simple Bayes net with Boolean variables, indicating whether someone broke an election law (**B**), was indicted (**I**), whether the prosecutor was politically motivated (**M**), if the person was found guilty (**G**), and if they were ultimately put in jail (**J**).

A (2 points): Which of the following are asserted by the network structure:

- P(B,I,M) = P(B) P(I) P(M)
- P(J|G) = P(J|G,I)

- P(M|G,B,I) = P(M|G,B,I,J)

B (1 points): What is P(b,i,¬m,g,j)?

C (2 points): Use variable elimination to calculate the probability someone goes to jail, given that he or she broke the law and the prosecutor was not politically motivated. Try to keep the intermediate factors as small as possible. Describe all of the assumptions/decisions that you make at each step.

D (2 points): Describe a process by which you might sample a joint assignment to the variables of this Bayes net, conditioned on the fact that there was a guilty verdict. You can use any of the techniques we described in class, but you should justify your choice.

Naive Bayes Spam Classification

In this question you will be implementing a Naive Bayes spam classification system for predicting whether an email is spam or ham.

Data - Pre-processed spam and ham email data: emails.tar.gz
(http://www.tamaraberg.com/teaching/Fall_15/HW/HW3/emails.tar.gz)

To open this file under mac or linux, use the command: "tar zxvf emails.tar.gz".

To perform classification, we split the data into 4 parts:
1) spam training documents (http://www.tamaraberg.com/teaching/Fall_15/HW/HW3/spamtraining.txt)
2) ham training documents (http://www.tamaraberg.com/teaching/Fall_15/HW/HW3/hamtraining.txt)
3) spam testing documents (http://www.tamaraberg.com/teaching/Fall_15/HW/HW3/spamtesting.txt)
4) ham testing documents (http://www.tamaraberg.com/teaching/Fall_15/HW/HW3/hamtesting.txt).

The spam/ham training documents will form your labeled training set (the documents whose label you know) and the spam/ham testing documents will form your testing set (the documents whose label you need to predict).

a: Computing Features. (4 points)

As features for spam/ham classification we will use a lexicon of words. Compute a lexicon consisting of the set of unique words occurring more than k times in the training document collection (k is a number you should set experimentally by examining the lexicon and classification accuracy produced for a few values of k, e.g. 1,2,10,...).

b. Training. (5 points)

The goal of the training stage is to estimate the parameters of the model from the training set. Parameters include the **likelihoods** P(Word | class) for every word (w) in your lexicon for each class (spam and ham). The likelihood estimate is defined as:

P(Word = w | class) = (# of times word w occurs in training examples from this class) / (Total # of words in training examples from this class).

In addition, you should smooth the likelihoods to reduce the effect of small probabilities. *Laplace smoothing* is a very simple method that increases the observation count of every value 'w' by some constant *m*. This corresponds to adding *m* to the numerator above, and m*V to the denominator (where V is the number of words in your lexicon). The higher the value of *m*, the stronger the smoothing. Experiment with different integer values of *m* (sample a few values between 1 and 50) and report how it affects your classification accuracies.

You should also estimate the parameters for the **priors P(class)** as the empirical frequencies of the classes in the training set (ie percentage of spam and ham documents).

c. Testing. (4 points)

You will perform maximum a posteriori (MAP) classification of spam or ham according to your learned Naive Bayes model. Suppose a test document contains words w_1, w_2, ... , w_{200}. According to this model, the posterior probability (up to scale) of each class given spam or ham is:

$P(\text{class}) \cdot P(w_1|\text{class}) \cdot P(w_2|\text{class}) \cdot \ldots \cdot P(w_{200} \mid \text{class})$.

To avoid underflow, you should compute the log of the above quantity:

$\log P(\text{class}) + \log P(w_1|\text{class}) + \log P(w_2|\text{class}) + \ldots + \log P(w_{200} \mid \text{class})$.

For each test document, compute the above probabilities for the spam and ham classes. Classify the test document as the most probable class according to your computation. Also include in your report a discussion of whether it would make any difference to use maximum likelihood (ML) classification for the provided data.

d. Measuring Performance. (2 points)

Compute the overall accuracy -- what percentage of documents in the testing set were classified correctly? Compute the spam accuracy -- what percentage of documents in the spam testing set were classified as spam? Compute the ham accuracy -- what percentage of documents in the ham testing set were classified as ham? Do these accuracies vary with k and/or m?

e. Write-up (3 points)

Clearly mention in your report:

- Description of your Naive Bayes classifier implementation.
- Description of your results -- including overall accuracy, spam accuracy, ham accuracy, and descriptions of your experiments to set values for k and m. Also include discussion of whether MAP and ML classification would produce different results for the provided data.
- Include some example emails showing where your algorithm did well and where it failed (and why you think this behavior occurred).

f. Extra-credit

Describe and implement additional features, e.g. n-gram features like bi-grams or tri-grams, or any other features you think would be useful for spam/ham classification. See whether your proposed features improve classification accuracy.

In general, parameters of the model, such as k and m should be tuned only on the training set. Split your training set into two parts, training and evaluation. Use the evaluation set find good settings for these parameters. Report whether this makes a difference in your final testing set accuracy.

Collect some documents for new target classes (e.g. documents describing different topics or different genres) and train classifiers to predict between these new classes.

Submission Instructions

HW should be uploaded to the classroom.cs.unc.edu server (one submission per group uploaded into the directory of one of the group members). Email Ric (poirson@cs.unc.edu) with the location of your HW submission, your group member names, and your PDF write-up. All submitted code must run on the classroom server for full credit. **Note**, if you edit any portion of your HW after the submission date that time-stamp will become your submission time and count into your late days.

Submissions should include:

1. A report in PDF format called HW2_lastname1_lastname2_lastname3.pdf with the last names of all group members. The report should describe your implemented solution and fully answer the questions for every part of the assignment. Your description should highlight the most "interesting" aspects of your solution, i.e., any non-obvious implementation choices and parameter settings, and what you have found to be especially important for getting good performance. Feel free to include pseudo-code or figures if they are needed to clarify your approach. Your report should be self-contained and it should (ideally) make it possible for us to understand your solution without having to run your source code.

All group reports need to include a brief statement of individual contribution, i.e., which group member(s) was/were responsible for which parts of the solution and submitted material.

2. Your source code and a README file indicating how to run your code. Code should be well commented and it should be easy to see the correspondence between what's in the code and what's in the report.

Late policy: Assignments are due at 11:59pm on the assigned date. Students have 5 free late days to use over the course of the semester (e.g. an assignment submitted at 2am after the deadline will count as 1 day late). After free late days are used, assignments will be accepted up to 1 week late at a penalty of 10% per day.

Academic integrity: All code and written responses for assignments should be original within your group. To protect the integrity of the course, we will be actively checking for code plagiarism (both from current classmates and the internet).