10701/15781 Machine Learning, Fall 2007: Homework 4

Due: Monday, November 19, beginning of the class

Instructions

There are 5 questions on this assignment. Problem 5 involves coding. Do not attach your code to the writeup. Instead, copy your implementation to

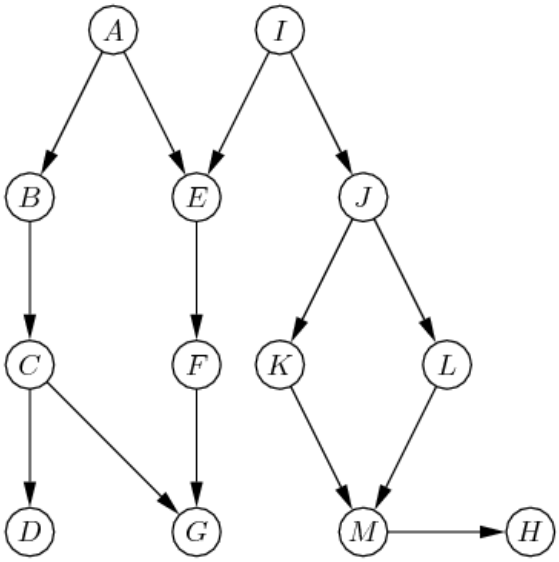
`/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW4`

To write in this directory, you need a kerberos instance for andrew, or you can log into, for example, unix.andrew.cmu.edu.

Please submit each problem seperately with your name and Andrew ID on each problem. Refer to the webpage for policies regarding collaboration, due dates, and extensions.

1 [14 points] Independence [Jingrui]

Which of the following statements are true with respect to the following graphical model, regardless of the conditional probability distributions?



(a) $P(D, H) = P(D)P(H)$

(b) $P(A, I) = P(A)P(I)$

(c) $P(A, I|G) = P(A|G)P(I|G)$

(d) $P(J, G|F) = P(J|F)P(G|F)$

(e) $P(J, M|K, L) = P(J|K, L)P(M|K, L)$

(f) $P(E, C|A, G) = P(E|A, G)P(C|A, G)$

(g) $P(E, C|A) = P(E|A)P(C|A)$

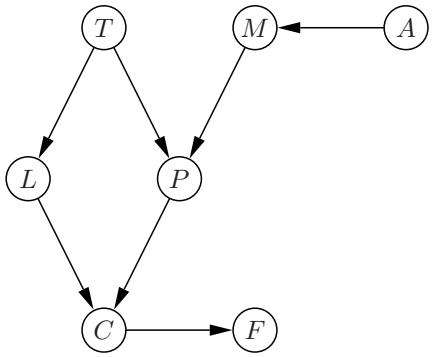
2 [6 points] Bayes Nets [Jingrui]

Consider three binary variables x, y, and z with the joint distribution:

x	y	z	$p(x, y, z)$
0	0	0	0.135
0	0	1	0.09
0	1	0	0.005
0	1	1	0.02
1	0	0	0.1125
1	0	1	0.075
1	1	0	0.1125
1	1	1	0.45

Show that the joint distribution $p(x, y, z)$ can be represented by a Bayes net that has just two edges.

3 [25 points] Variable Elimination [Joseph]



The secrets to fame and fortune are locked away in a Bayes net. Unfortunately, we cannot use these secrets within our lifetimes if we resort to marginalization to perform inference. Using the power of Variable Elimination, we can unlock the wisdom of the Bayes net to become rich and famous, and still have time to enjoy it.

The variables of interest are: Attending machine learning class (A), your mastery of Machine Learning (M), your Time-management skills (T), being Late to the airport (L), writing a good Paper (P), making it to the Conference (C), and achieving Fame (F). All of these variables are binary valued $\{T, F\}$. The conditional probability tables are:

$$P(A = T) = 0.7 \qquad P(T = T) = 0.6$$
$$P(M = T|A = T) = 0.9, \quad P(M = T|A = F) = 0.3$$
$$P(L = T|T = T) = 0.1, \quad P(L = T|T = F) = 0.4$$
$$P(F = T|C = T) = 0.5, \quad P(F = T|C = F) = 0.1$$

$$P(P=T|T=T,M=T)=0.9, \quad P(P=T|T=T,M=F)=0.5$$
$$P(P=T|T=F,M=T)=0.7, \quad P(P=T|T=F,M=F)=0.2$$

$$P(C=T|L=T,P=T)=0.2, \quad P(C=T|L=T,P=F)=0.1$$
$$P(C=T|L=F,P=T)=0.9, \quad P(C=T|L=F,P=F)=0.2$$

3.1 [3 points] Joint Probability

Write down the formula for the joint probability distribution that makes the same conditional independent assumptions as the above graph.

3.2 [7 points] Variable Elimination for Fame

Using variable elimination, compute the probability of being Famous given that you Attended machine learning class [$P(F=T|A=T)=?$]. Show your work.

3.3 [2 points] Moralization

Before we can apply the magic of variable elimination we must first moralize the graph. What two edges must be added to moralize this graph?

3.4 [6 points] Fill Edges

Efficient variable elimination depends on the order in which we eliminate variables. How do we pick a good ordering? Unfortunatley, this is in itself a hard problem, which you will have the opportunity to "solve" here. Many heuristics for solving this problem rely on fill edges. Suppose we eliminate the variable P first. Then we must create a temporary factor which contains the unbound variables T, C, L, and M. This corresponds to removing P (and its edges) from the graph and connecting all of its neighbors (parents and children) in a new graph. The new edges that we add in this process are called fill edges. We now have a new graph in which we have removed P and all the edges that were connected to P, and added fill edges to fully connect the neighbors of P. We can repeat this process on the new graph to "eliminate" the next variable.[1]

3.4.1 [3 points] Counting Fill Edges

Suppose we use the following elimination order $\{P, M, C, L, T, A\}$ where we are trying to compute $P(F)$. After moralization how many new fill edges will we introduce?

3.4.2 [3 points] Minimum Fill Ordering

Suppose we (you) had to do variable elimination on this graph to compute $P(F)$. What ordering would minimize the amount of work you have to do? Specfically, what ordering would require no additional fill edges after moralization? You may want to use this ordering in the future.

3.5 [7 points] Most Likely Explanation for Your Future

We don't know your future but the graphical model does. Suppose you attend class and have good time management skills, what does your future hold? Specifically, will you be famous?

[1] Fill edges can be used to directly estimate the computational complexity of variable elimination: Take the moralized graph you drew in Question 3.3, and draw the additional fill edges you generate (the resulting graph is called the triangulated graph). The running time of variable elimination will be linear in the number of variables, but exponential in the size of the largest clique in the triangulated graph. (A clique is a set of nodes forming a fully connected subgraph.)

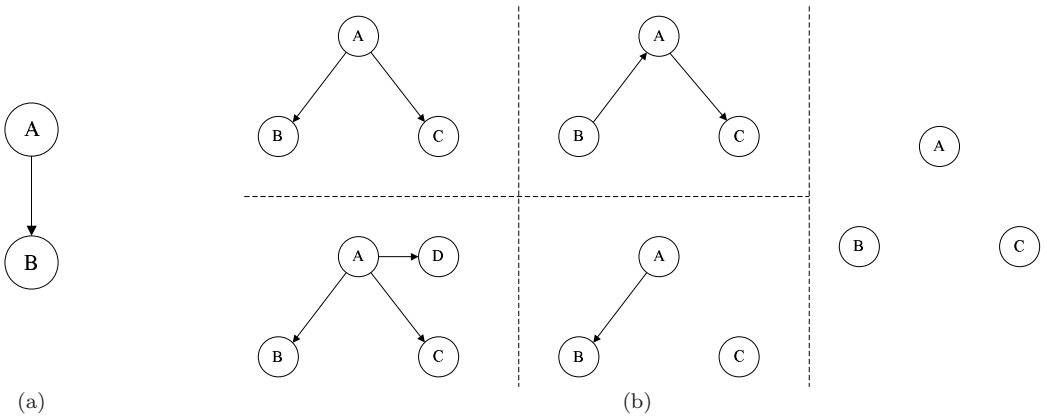


Figure 1: Bayesian networks for Question 4.2.1.

3.6 [3 points Extra Credit] Your Own Model

Draw your own simple graphical model. Maybe its relevant to your research. Maybe its just really funny. Describe the variables and the graph structure you chose.

4 [25 points] Bayes Net structure learning [Sue Ann]

4.1 Mutual information

Consider the following dataset with four binary variables A, B, C, D:

A	B	C	D
1	0	1	0
1	0	1	1
1	0	0	1
0	1	0	1
1	0	1	0

Your goal is to learn a Bayesian network structure for this data set.

(a) Compute the mutual information $I(X, Y)$ for each pair of variables X, Y.

(b) Using the computed mutual informations, find a tree Bayesian network that maximizes the likelihood of the training data.

4.2 Learning forests with BIC score

In this part, you will derive a variation of the Chow-Liu algorithm for learning Bayesian networks with forest graph structure using the BIC score.

(a) Recall that the BIC score for M data points is

$$\text{score}_{\text{BIC}}(\mathcal{G}; \mathcal{D}) = \log p(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) - \frac{\text{NumberParams}(\mathcal{G})}{2} \log M, \tag{1}$$

where $\mathcal{G}$ is a Bayesian network, $\mathcal{D}$ is the dataset, $\theta_{\mathcal{G}}$ are the parameters that represent the CPDs in $\mathcal{G}$, and NumberParams($\mathcal{G}$) is the total number of independent parameters in the CPDs of $\mathcal{G}$. For example, for the Bayesian network in Figure 1(a), the number of parameters is $N_A - 1 + N_A(N_B - 1) = N_A N_B - 1$, where $N_X = |\text{Val}[X]|$ is the number of possible values for variable X.

Write down the number of independent parameters in the CPDs for each of the five Bayesian network in Figure 1(b), in terms of N_A, N_B, N_C, and N_D. Show your work.

(b) A forest is a graph that consists of one or more connected components, each of which is a tree (but not a poly-tree, i.e., each node has at most one parent). Prove that the number of independent parameters for any forest $\mathcal{G}$ can be written as

$$\text{NumberParams}(\mathcal{G}) = \sum_{\{X,Y\} \in \text{E}_\mathcal{G}} (N_X - 1)(N_Y - 1) + \left(\sum_{X \in \text{V}_\mathcal{G}} N_X \right) - |\text{V}_\mathcal{G}|. \quad (2)$$

Here, $\text{E}_\mathcal{G}$ are the undirected edges of $\mathcal{G}$ and $\text{V}_\mathcal{G}$ are its vertices (variables).

Hint: In a forest $\mathcal{G}$, $|\text{V}_\mathcal{G}| = |\text{E}_\mathcal{G}| + \#\text{components}_G$, where $\#\text{components}_G$ is the number of connected components in $\mathcal{G}$.

(c) Using (1) and (2), describe an efficient algorithm for learning a forest Bayesian network that maximizes the BIC score. (Hint: This algorithm is a small modification of Chow Liu. Start by writing out the expression for the weight of each possible edge.)

5 [30 points] The Forward-Backward Algorithm [Steve]

In this question, you will implement the Forward-Backward algorithm for Hidden Markov Models and compare it with a fixed-lag smoothing method (an online version of Forward-Backward where you do not get to use the entire sequence of observations but just the recent history).

For all implementation questions, please submit your source code to

`/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW4/`

and provide pseudocode in your answers. The dataset for this question is available on the class website.

The OCR data set that we provide consists of a sequence of words, one character per row.[2] The very first character of each word was capitalized in the original data and has been omitted for simplicity. The columns have the following meaning:

Col. 1: character ID (same as the row number)

Col. 2: character code (1-26), 1='a', 2='b', etc.

Col. 3: the code of the previous character or -1 if this is the first character in a word)

Col. 4: word id

Col. 5: the position of the character in the word

Col. 6: cross-validation fold (ignore this)

Cols. 7-70: pixel value (0/1).

5.1 Learning Parameters

Let X_t denote the t-th letter in a word and O_t^k the value of the k-th pixel for the t-th character. For the first part, you will learn the parameters of the Hidden Markov Model using Maximum Likelihood Estimation. You should learn a stationary model (one that does not depend on t), i.e., you should learn a single distribution $p(X_1)$, a single CPT $p(X_t|X_{t-1})$ and 64 CPTs $p(O_t^k|X_t)$ (one for each pixel).

(a) Using MLE, learn a distribution over the first letter, $p(X_1)$ and the transition model $p(X_t|X_{t-1})$ from the training set. Plot the transition matrix $A = P(X_t|X_{t-1})$.

Hint: This can be done in Matlab with the function: imagesc(A);

(b) Learn an observation model, in which all pixel values are independent, given the character code, i.e.

$$p(\mathbf{O}_t|X_t) = \prod_k p(O_t^k|X_t)$$

[2] This dataset is a modified version of the dataset at http://ai.stanford.edu/~btaskar/ocr/, which has been subsampled and slightly altered to simplify the processing.

5.2 Forward-Backward Algorithm

In this part, you will implement the Forward-Backward Algorithm for HMMs and compare its performance to a Naive Bayes approach which classifies each character independently of all others.

(a) Implement the Forward-Backward Algorithm and report the test set accuracies (pick the class with highest estimated probability). The accuracies should be reported as the proportion of characters which were classified correctly and the proportion of words which were classified completely correctly.

Hint: Testing on the full test set can take a while, depending on the speed of your implementation. You may want to debug your code on a subset of the test data first.

(b) Use the training set to train a single Naive Bayes distribution $p(X_i)$. How would Naive Bayes perform on this dataset? Evaluate the learned Naive Bayes classifier on the test set and compare its performance against the Forward-Backward algorithm. You should report character and word accuracy on the test set and discuss how the two algorithms perform in comparison and why.

5.3 An Online Approach

It is often the case that we do not have the luxury of observing the entire sequence of data before running the Forward-Backward algorithm. This might be due to memory or computation contraints, or it might be that some kind of streaming data must be processed in real time. An example application is real time speech recognition or online activity recognition/patient monitoring. This type of problem is typically referred to as an online problem.

The simplest online approach for HMMs is to generate the forward factors $\alpha_t(X_t)$ in the same way as the Forward-Backward algorithm, but instead of waiting until the end to compute the backward factors, it uses only a few (say m) time steps after t to calculate an approximate backward factor $\beta_t^{(m)}(X_t)$; in other words, to calculate the backward factor for t, it starts at $\min\{t+m, \text{largest t value observed}\}$ instead of always starting at the largest t value observed.

(a) As you might guess, there is some kind of tradeoff between the lag time (m) and the prediction accuracy of the online variant of the Forward-Backward algorithm. In this problem, you will implement the approximate online approach described above and report the test-set character and word accuracy as a function of the lag time (m) for $m = 0, 1, 2, 3, 4, 5$.

Hint: This will be horribly inefficient if you try to run Forward-Backward on the subsequence $O_1, \ldots, O_t$ for each $t \leq T$. Notice that it is not necessary to recompute the entire forward factor table at each time step. Furthermore, although some recalculating of the backward factors is necessary, it is not necessary to perform the backtracking step all the way back to $t = 1$ every time any backward factor is recalculated.