10708 Graphical Models: Homework 2

Due October 11th, beginning of class

September 27, 2006

Instructions: There are seven questions on this assignment. Each question has the name of one of the TAs beside it, to whom you should direct any inquiries regarding the question. The last problem involves coding, which should be done in MATLAB. Do *not* attach your code to the writeup. Instead, copy your implementation to

`/afs/andrew.cmu.edu/course/10/708/Submit/your_andrew_id/HW2`

Refer to the web page for policies regarding collaboration, due dates, and extensions.

1 I-equivalence [10 pts] [Khalid]

1. Prove that two network structures $\mathcal{G}_1$ and $\mathcal{G}_2$ are I-equivalent if and only if the following two conditions hold:
   (a) The two graphs have the same set of trails, and
   (b) A trail is active in $\mathcal{G}_1$ iff it is active in $\mathcal{G}_2$.
2. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs over $\mathcal{X}$. Prove that if $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and the same set of v-structures then they are I-equivalent.
   (Hint: use the result from part 1)

2 Decomposable Scores [10 pts] [Ajit]

Decomposable scoring functions are those where the score of a network given data $\mathcal{D}$ can be represented as the sum of scores of each node given its parents and the data:

$$\text{score}(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}(X_i | \mathbf{Pa}_i^{\mathcal{G}} : \mathcal{D})$$

In greedy structure search we explore the space of structures by applying a local operator to an existing Bayes net. Examples of local operators include adding an edge, deleting an edge, and reversing an edge. In this question you will show that if the scoring function is decomposable, then computing the change in score caused by a local operator can be computed efficiently.

1. Prove proposition 15.4.5 (Koller & Friedman p. 656)
2. Prove proposition 15.4.6 (Koller & Friedman p. 656)

3 Learning Edge Directions [15 pts] [Ajit]

In this question, we consider a simpler form of structure learning for BNs: Assume we have a skeleton and want to build a BN from it. For each edge, we want to either assign a direction to this edge or delete it from the graph. For this problem, you can assume you are using some decomposable score, FamScore($X_i|\mathbf{Pa}_{X_i}$).

1. Consider the skeleton $X_1 - - X_2 - - X_3$, what are the possible BNs that we are considering in this problem? What is the score of each of the graphs?
2. Now, consider the skeleton $X_1 - - X_2 - - X_3 - - X_4$. Does the decision about the edge $X_1 - - X_2$ affect the family score of X_3? Justify your answer.
3. Using the intuitions above, design a linear time dynamic programming algorithm for finding the optimal BN from a chain skeleton $X_1 - - X_2 - - X_3 - - X_4 - - \cdots - - X_n$.

4 Greedy Structure Search [15 pts] [Khalid]

Suppose we have a general network structure search algorithm, A, that takes a set of basic operators on network structures as a parameter. This set of operators defines the search space for A, as it defines the candidate network structures that are the "immediate successors" of any current candidate network structure, *i.e.*, the successor states of any state reached in the search. Thus, for example, if the set of operators is [add an edge not currently in the network], then the successor states of any candidate network $\mathcal{G}$ is the set of structures obtained by adding a single edge anywhere in $\mathcal{G}$ (so long as acyclicity is maintained).
Given a set of operators, A does a simple greedy search over the set of network structures, starting from the empty network (no edges), using the BIC scoring function. Now, consider two sets of operators we can use in A. Let $A_{[add]}$ be A using the set of operations [add an edge not currently in the network], and let $A_{[add,delete]}$ be A using the set of operations [add an edge not currently in the network, delete an edge currently in the network].

1. Show a distribution where, regardless of the amount of data in our training set (*i.e.*, even with infinitely many samples), the answer produced by $A_{[add]}$ is worse (*i.e.*, has a lower BIC score) than the answer produced by $A_{[add,delete]}$. (It's easiest to represent your true distribution in the form of a Bayesian network, *i.e.*, a network from which sample data is generated.)

2. Show a distribution where, regardless of the amount of data in our training set, $A_{[add,delete]}$ will converge to a local maximum. In other words, the answer returned by the algorithm has a lower score than the optimal (highest-scoring) network. What can we conclude about the ability of our algorithm to find the optimal structure?

5 Variable Elimination [10 pts] [Khalid]

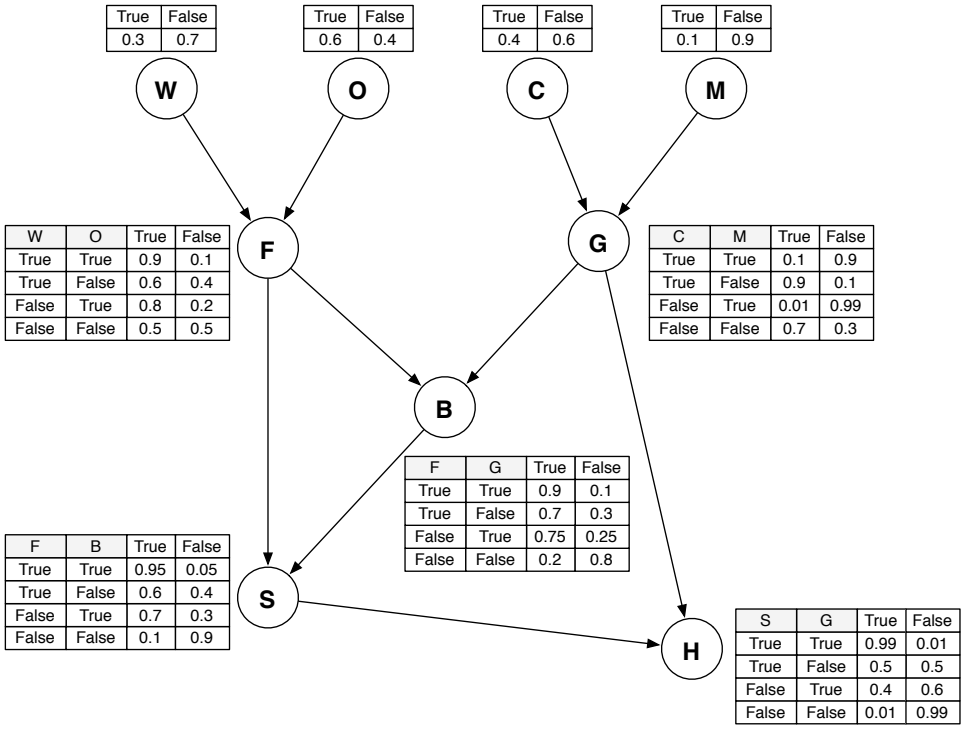

Figure 1: Football network

After two losses in three games to start the season, the world champion Pittsburgh Steelers seem to be in a world of trouble. Not knowing what else to do, Coach Cowher has enlisted your help in trying to turn things around during the bye week. In Figure 1, you will find a Bayes net that hopefully contains the key to a successful season. The variables of interest are: Weather (W), Opponent (O), Fans (F), Big Ben (B), Steelers Win (S), Good Health (G), Chunky Soup (C), Motorcycle (M), and Happy (H).

Coach Cowher would like to know the answers to the following questions, but keep in mind, he does not like to be kept waiting. There's only one thing he hates more than losing, and that is exponential computational complexity.

1. What is the chance that the Steelers win given that Big Ben eats his Chunky Soup? [$P(S = T|C = T) = ?$]
2. How fired up are the fans given that Big Ben is in good health? [$P(F = T|G = T) = ?$]
3. $P(M = T|G = T) = ?$
4. $P(M = T|G = T, S = T) = ?$
5. $P(W = T|G = T, B = F, S = T) = ?$

Additionally, report the ordering used and the factors produced after eliminating each variable for the first query $[P(S = T|C = T)]$.

6 Conditional Probabilities in Variable Elimination [15pts]

[Khalid]

Consider a factor produced as a product of some of the CPDs in a Bayesian network $\mathcal{B}$:

$$\tau(\mathbf{W}) = \prod_{i=1}^{k} P(Y_i|\mathbf{Pa}_{Y_i})$$

where $\mathbf{W} = \cup_{i=1}^{k}(\{Y_i\} \cup \mathbf{Pa}_{Y_i})$.

1. Show that τ is a conditional probability in some network. More precisely, construct another Bayesian network $\mathcal{B}'$ and a disjoint partition $\mathbf{W} = \mathbf{Y} \cup \mathbf{Z}$ such that $\tau(\mathbf{W}) = P_{\mathcal{B}'}(\mathbf{Y}|\mathbf{Z})$.
2. Show that the intermediate factors produced by the variable elimination algorithm are also conditional probabilities in some network.

7 Tree-Augmented Naïve Bayes [25pts] [Ajit]

In many classification tasks naïve Bayes is either competitive with, or is, the best method, even though, naïve Bayes ignores dependencies between features. This seems to be an argument against structure learning, until one realizes that most structure learning methods are trying to model the joint distribution, which does not necessarily corresponds to a good estimate of the class-conditional distribution.

Tree-Augmented Naïve Bayes (TAN) is a model which focuses its search on the class conditional distribution; augmenting naïve Bayes by adding correlations between features such that each feature has the class node, and one other feature, as parents. Figure 2 is an example of a TAN model.

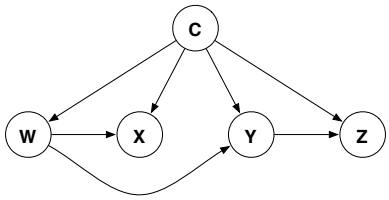

Figure 2: An example of tree-augmented naïve Bayes. Note that the induced graph on the evidence variables (w, x, y, z) forms a tree.

The algorithm for learning TAN models is a variant of the Chow-Liu algorithm for learning tree-structured Bayes nets. Let C represent the class variable, and $\{X_i\}_{i=1}^n$ be the features (non-class variables).

1. Compute the conditional mutual information given C between each pair of distinct variables,
$$I(X_i; X_j|C) = \sum_{x_i,x_j,c} \tilde{P}(x_i, x_j, c) \log \frac{\tilde{P}(x_i, x_j|c)}{\tilde{P}(x_i|c)\tilde{P}(x_j|c)}$$
where $\tilde{P}(\cdot)$ is an empirical distribution (computed using the training data). Intuitively, this quantity represents the gain in information of adding X_i as a parent of X_j given that C is already a parent of X_j.
2. Build a complete undirected graph on the features $X_1, \ldots, X_n$ where the weight of the edge between X_i and X_j is $I(X_i; X_j|C)$. Call this graph $\mathcal{G}_F$.
3. Find a maximum weighted spanning tree[1] on $\mathcal{G}_F$. Call it $\mathcal{T}_F$.
4. Pick an arbitrary node in $\mathcal{T}_F$ as the root, and set the direction of all the edges in $\mathcal{T}_F$ to be outward from the root. Call the directed tree $\mathcal{T}'_F$. (Hint: Use DFS).
5. The structure of the TAN model consists of a naïve Bayes model on $C, X_1, \ldots, X_n$ augmented by the edges in $\mathcal{T}'_F$.

The task is breast cancer typing, classifying a tumor as either malignant or benign. The data is provided in `breast.csv`.

[1]Kruskal's or Prim's algorithm can be used to find a maximum weighted spanning tree

7.1 Structure Learning

Implement the above algorithm for learning the structure of a TAN model, and submit your code as `tanstruct.m`. Using the breast cancer data, draw the structure (directed acyclic graph) produced using this algorithm in your writeup.

7.2 Classification

In this question you will compare the classification accuracy of naïve Bayes and TAN. First, randomly withhold 183 records as a test set. Then, using a training set of size m, for $m = 100, 200, 300, 400, 500$

1. Learn the structure of a TAN model and estimate the parameters using the following smoothing estimator. For the parameter corresponding to $P(x_i|\mathbf{Pa}_i)$ estimate it using

$$\theta_{x_i|\mathbf{Pa}_i} = \alpha\tilde{P}(x_i|\mathbf{Pa}_i) + (1-\alpha)\tilde{P}(x_i)$$
$$\alpha = \frac{m\tilde{P}(\mathbf{Pa}_i)}{m\tilde{P}(\mathbf{Pa}_i) + s}$$

   where s is a smoothing parmeter. For this question, use $s = 5$. This is known as back-off smoothing.

2. Learn a naïve Bayes model and estimate the parameters using back-off smoothing.

3. Compare the classification accuracy of naïve Bayes and TAN on the test set.

Plot classification error vs. the number of training samples in your writeup. Submit the code used to run these experiments as `tancompare.m`.