Homework 4

Due at noon on March 9, 2016

GUIDELINES: You can brainstorm with others, but please solve the problems and write up the answers by yourself. You may use textbooks (Koller & Friedman, Russel & Norvig, etc.), lecture notes, and standard programming references (e.g., online Java API documentation). Please do NOT use any other resources or references (e.g., example code, online problem solutions, etc.) without asking.

SUBMISSION INSTRUCTIONS: Submit this assignment by Dropbox. You submission should include: A PDF containing written answers and all your code.

1 LEARNING BAYESIAN NETWORK (25 POINTS)

Consider learning the following Bayesian network: $A \rightarrow B \leftarrow C$. And the following data table, with entries '?1' and '?2' missing at random:

A	B	C
?1	T	T
F	T	F
F	F	F
T	F	T
T	F	T
T	T	T
F	F	T
F	T	T
F	T	T
T	?2	F
T	T	F
T	F	F

(a) Use the data to estimate initial parameters for this network, using maximum likelihood estimation for simplicity

(b) Perform two iterations of the EM algorithm (by hand) to estimate the values of the missing data, reestimate the parameters, reestimate the values of the missing data, and reestimate the parameters once more. Show your calculations.

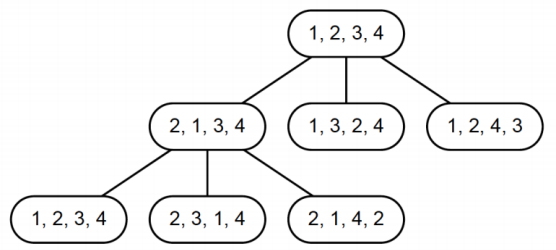

Figure 2.1: Partial search tree example for orderings over variables X_1, X_2, X_3, X_4. Successors to $\prec = (1,2,3,4)$ and $\prec' = (2,1,3,4)$ shown.

2 Learning Structure of Bayesian Network (25 points)

Consider learning the structure of a Bayesian network for some given ordering, $\prec$, of the variables $X_1, \ldots, X_n$. (This can be done efficiently as described in section 18.5.2.1 of the textbook.) Now assume that we want to perform a search over the space of orderings; that is, we are searching for a network (with bounded in-degree k) that has the highest score. We do this by defining the score of an ordering as the score of the (bounded in-degree) network with the maximum score consistent with that ordering, and then we search for the ordering with the highest score. We bound the in-degree so that we have a smaller and smoother search space. We will define our search operator, o, to be "Swap X_i and X_{i+1}" for some $i = 1, \ldots, n-1$. Starting from some given ordering, $\prec$, we evaluate a decomposable structure score of all successor orderings, $\prec'$, where a successor ordering is found by applying o to $\prec$ (see Figure 2.1). We now choose a particular successor, $\prec'$. Provide an algorithm for computing as efficiently as possible the score for all successors of the new ordering, $\prec'$, given that we have already computed the scores for all successors of $\prec$. Note: A structure score score $(G : D)$ is decomposable if the score of a structure G can be written as

$$score(G : D) = \sum_i FamScore(X_i | Pa^G_{X_i} : D),$$

where the *family score* $FamScore(X|U : D)$ is a score measuring how well a set of variables U serves as parents of X in the data set D.

3 Gibbs Sampling (15 points)

Consider the Bayesian network $A \rightarrow B$, where A and B are Boolean variables, $P(A = 1) = 1/2$, and $P(B = 1|A = 1) = P(B = 0|A = 0) = 1$. Will Gibbs sampling applied to this network converge to the correct stationary distribution? Why?

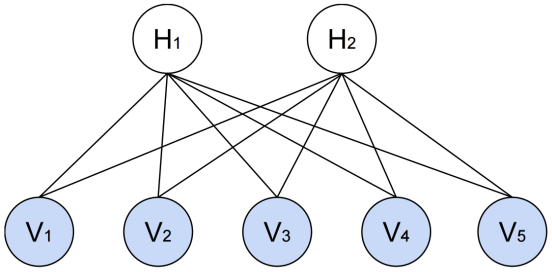

Figure 5.1: An example RBM with 5 visible units and 2 hidden units.

4 Particle Filter (15 points)

At each resampling step, particle filtering usually creates multiple copies of the highest-weight samples. Why does this not result in the number of different samples dwindling over time?

5 Learning Restricted Boltzmann Machines (20 points)

Restricted Boltzmann Machines (RBMs) are a class of Markov networks that have been used in several applications, including image feature extraction, collaborative filtering, and recently in deep belief networks. An RBM is a bipartite Markov network consisting of a visible (observed) layer and a hidden layer, where each node is a binary random variable. One way to look at an RBM is that it models latent factors that can be learned from input features. For example, suppose we have samples of binary user ratings (like vs. dislike) on 5 movies: Finding Nemo (V_1), Avatar (V_2), Star Trek (V_3), Aladdin (V_4), and Frozen (V_5). We can construct the following RBM:
Here, the bottom layer consists of visible nodes $V_1, \ldots, V_5$ that are random variables representing the binary ratings for the 5 movies, and H_1, H_2 are two hidden units that represent latent factors to be learned during training (e.g., H_1 might be associated with Disney movies, and H_2 could represent the adventure genre). If we are using an RBM for image feature extraction, the visible layer could instead denote binary values associated with each pixel, and the hidden layer would represent the latent features. However, for this problem we will stick with the movie example. In the following questions, let $V = (V_1, \ldots, V_5)$ be a vector of ratings (e.g. the observation $v = (1,0,0,0,1)$ implies that a user likes only Finding Nemo and Aladdin). Similarly, let $H = (H_1, H_2)$ be a vector of latent factors. Note that all the random variables are binary and take on states in $\{0,1\}$. The joint distribution of a configuration is given by

$$P(V = v, H = h) = \frac{1}{Z} e^{-E(v,h)}, \tag{5.1}$$

where

$$E(v,h) = -\sum_{ij} w_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \tag{5.2}$$

is the energy function, $\{w_{ij}\}, \{a_i\}, \{b_j\}$ are model parameters, and

$$Z = Z(\{w_{ij}\}, \{a_i\}, \{b_j\}) = \sum_{v,h} e^{-E(v,h)}$$

is the partition function, where the summation runs over all joint assignments to V and H.

(e) We can use the log-likelihood of the visible units, $\log p(V = v)$, as the criterion to learn the model parameters $\{w_{ij}\}, \{a_i\}, \{b_j\}$. However, this maximization problem has no closed form solution. One popular technique for training this model is called "contrastive divergence" and uses an approximate gradient descent method. Compute the gradient of the log-likelihood objective with respect to w_{ij} by showing the following:

$$\frac{\partial \log p(V = v)}{\partial w_{ij}} = \sum_h p(H = h|V = v) v_i h_j - \sum_{v,h} p(V = v, H = h) v_i h_j \tag{5.3}$$

$$= \mathscr{E}[V_i H_j | V = v] - \mathscr{E}[V_i H_j]. \tag{5.4}$$

Hint 1: To save some writing, do not expand $E(v,h)$ until you have $\frac{\partial E(v,h)}{\partial w_{ij}}$.

Hint 2: The partition function, Z, is a function of w_{ij}.

(f) After training, suppose $H_1 = 1$ corresponds to Disney movies, and $H_2 = 1$ corresponds to the adventure genre. Which w_{ij} do you expect to be positive, where i indexes the visible nodes and j indexes the hidden nodes? List all of them.