

CS 228, Winter 2009

Problem Set #2

1. [5 points] MAP Dirichlet

Suppose that a prior on a parameter vector is $p(\boldsymbol{\theta}) \sim \text{Dirichlet}(\alpha_1, \dots, \alpha_k)$. What is the MAP value of the parameters, that is $\text{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D})$? Assume M_1, M_2, \dots, M_k are the sufficient statistics from the data set \mathcal{D} .

Estimate: 1/2 page, 1/2 hour

2. [12 points] Search in Structure Learning

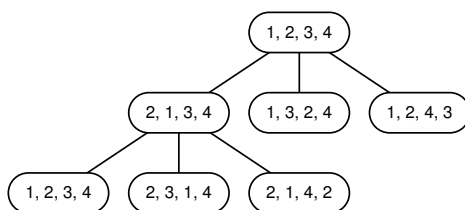


Figure 1: Partial search tree example for orderings over variables X_1, X_2, X_3, X_4 . Successors to $\prec = (1, 2, 3, 4)$ and $\prec' = (2, 1, 3, 4)$ shown.

Consider learning the structure of a Bayesian network for some given ordering, \prec , of the variables, X_1, \dots, X_n . This can be done efficiently as described in Section 14.3.2 of the course reader. Now assume that we want to perform search over the space of orderings, i.e. we are searching for the 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 searching 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 $i = 1, \dots, n-1$. Starting from some given ordering, \prec , we evaluate the BIC-score of all successor orderings, \prec' , where a successor ordering is found by applying o to \prec (see Figure 1). We now choose a particular successor, \prec' . Provide an algorithm for computing as efficiently as possible the BIC-score for the successors of the new ordering, \prec' , given that we have already computed the scores for successors of \prec .

Estimate: 1 page, 1 hour

3. [15 points] Module Network

In this problem, we will consider the task of learning a generalized type of Bayesian networks that involves shared structure and parameters. Let \mathcal{X} be a set of variables, which we assume are all binary-valued. A *module network* over \mathcal{X} partitions the variables \mathcal{X} into K disjoint clusters, for $K \ll n = |\mathcal{X}|$. All of the variables assigned to the same cluster have precisely the same parents and CPD. More precisely, such a network defines:

- An assignment function \mathcal{A} , which defines for each variable X , a cluster assignment $\mathcal{A}(X) \in \{C_1, \dots, C_K\}$.
- For each cluster $C_k (k = 1, \dots, K)$, a graph \mathcal{G} which defines a set of parents $\mathbf{Pa}_{C_k} = \mathbf{U}_k \subset \mathcal{X}$ and a CPD $P_k(X | \mathbf{U}_k)$.

The cluster network structure defines a ground Bayesian network where, for each variable X , we have the parents \mathbf{U}_k for $k = \mathcal{A}(X)$ and the CPD $P_k(X | \mathbf{U}_k)$. Figure 2 shows an example of such a network.

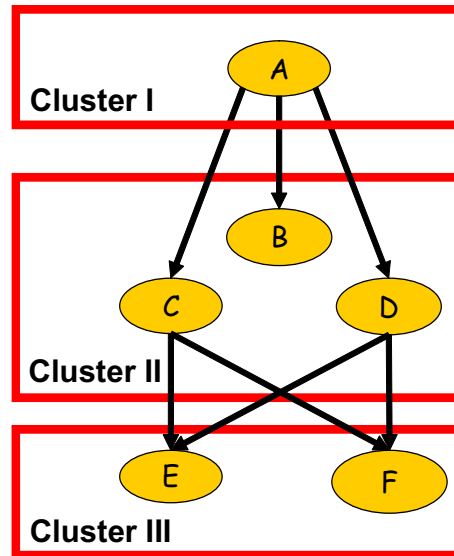


Figure 2: A simple module network

Assume that our goal is to learn a cluster network that maximizes the BIC score given a data set \mathcal{D} , where we need to learn both the assignment of variables to clusters and the graph structure.

- [5 points] Define an appropriate set of parameters and an appropriate notion of sufficient statistics for this class of models, and write down a precise formula for the likelihood function of a pair $(\mathcal{A}, \mathcal{G})$ in terms of the parameters and sufficient statistics.
- We use greedy local search to learn the structure of the cluster network. We will use the following types of operators (each operation should remain the graph acyclic):
 - **Add** operators that add a parent for a cluster;
 - **Delete** operators that delete a parent for a cluster;
 - **Node-Move** operators $o_{k \rightarrow k'}(X)$ that change from $\mathcal{A}(X) = k$ to $\mathcal{A}(X) = k'$. (If $X \in \mathbf{Pa}_{C_{k'}}$, moving X to k' is not allowed.)

As usual, we want to reduce the computational cost by caching our evaluations of operators and reusing them from step to step.

- [2 points] Why did we not include edge reversal in our set of operators?
- [2 points] Describe an efficient implementation for the update associated with the **Node-Move** operator.

- iii. [6 points] For each type of operator, specify which other operators need to be reevaluated once the operator has been taken. Briefly justify your response. (Suppose that we cache and update evaluations of operators and reuse them to save the computation.)

Estimate: 1 1/2 pages, 1 1/2 hours

4. [15 points] **Sampling on a Tree**

Suppose we have a distribution $P(\mathbf{X}, \mathbf{E})$ over two sets of variables \mathbf{X} and \mathbf{E} . Our distribution is represented by a nasty Bayes Net with very dense connectivity, and our sets of variables \mathbf{X} and \mathbf{E} are spread arbitrarily throughout the network. In this problem our goal is to use the sampling methods we learned in class to estimate the posterior probability $P(\mathbf{X} = \mathbf{x} \mid \mathbf{E} = \mathbf{e})$. More specifically, we will use a tree-structured Bayes Net as the proposal distribution for use in the importance sampling algorithm.

- (a) [1 points] For a particular value of \mathbf{x} and \mathbf{e} , can we compute $P(\mathbf{x} \mid \mathbf{e})$ exactly, in a tractable way? Can we sample directly from the distribution $P(\mathbf{X} \mid \mathbf{e})$? Can we compute $\hat{P}(\mathbf{x} \mid \mathbf{e}) = P(\mathbf{x}, \mathbf{e})$ exactly, in a tractable way? For each question, provide a Yes/No answer and a single sentence explanation or description.
- (b) [7 points] Now, suppose your friendly TAs have given you a tree network, where X_1 is the root and each X_i for $i \neq 1$ has exactly one parent $X_{p(i)}$. They tell you that the distribution $Q(\mathbf{X}, \mathbf{E})$ defined by this network is “close” to the distribution $P(\mathbf{X}, \mathbf{E})$. You now want to use *the posterior* in Q as your proposal distribution for importance sampling.
- Show how to sample from the posterior in Q . More specifically, provide an explicit construction for a clique tree over this network, and show how to use it to compute the posterior $Q(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$. Describe how to sample from this posterior, once it has been computed.
 - Now you must reweight the samples according to the rules of importance sampling. You want your weighted samples to accurately represent the actual posterior in the original network $P(\mathbf{X} \mid \mathbf{E} = \mathbf{e})$. Show precisely how you determine the weights $w[m]$ for the samples.
 - Show the form of the final estimator $\hat{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{E} = \mathbf{e})$ for $P(\mathbf{X} = \mathbf{x} \mid \mathbf{E} = \mathbf{e})$, in terms of the samples from part i, and the weights from part ii.
- (c) [7 points] Examining the results of your sampling, you find that your TAs were wrong, and the original tree from which you sampled did not provide very good estimates for the posterior. You therefore decide to produce a better tree, and rerun the importance sampling again. This better tree should represent a distribution which is “closer” to the posterior from which you are trying to sample. Show how you can use the samples produced in the previous step to learn the best tree (i.e., a tree that best represents the posterior distribution as estimated in part b, according to the likelihood score). Assume you have access to a maximum spanning tree algorithm, and your job is to define the weights for the edges in terms of the estimator computed above.

Estimate: 2 pages, 2 hours

5. [8 points] **EM with Uniform Initialization**

Consider the Naive Bayes model with class variable C and discrete evidence variables X_1, \dots, X_n . The CPDs for the model are parameterized by $P(C = c) = \theta_c$ and $P(X_i = x \mid C = c) = \theta_{x_i|c}$ for $i = 1, \dots, n$, and for all assignments $x_i \in \text{Val}(X_i)$ and classes $c \in \text{Val}(C)$.

Now given a data set $\mathcal{D} = \{\mathbf{x}[1], \dots, \mathbf{x}[M]\}$, where each $\mathbf{x}[m]$ is a complete assignment to the evidence variables, X_1, \dots, X_n , we can use EM to learn the parameters of our model. Note that the class variable, C , is never observed.

Show that if we initialize the parameters uniformly,

$$\theta_c^0 = \frac{1}{|\text{Val}(C)|} \quad \text{and} \quad \theta_{x_i|c}^0 = \frac{1}{|\text{Val}(X_i)|},$$

for all x_i, c , then the EM algorithm converges in one iteration, and give a closed form expression for the parameter values at this convergence point.

Estimate: 1 1/2 pages, 1 hour

6. [15 points] **Message passing with EM**

Suppose that we have an incomplete data set \mathcal{D} , and network structure \mathcal{G} and matching parameters. Moreover, suppose that we are interested in learning the parameters of a single CPD $P(X_i \mid \mathbf{U}_i)$. That is, we assume that the parameters we were given for all other families are frozen and do not change during the learning. This scenario can arise for several reasons: we might have good prior knowledge about these parameters; or we might be using an incremental approach (for example, see the section in Box 15.C on accelerating convergence).

We now consider how this scenario can change the computational cost of the EM algorithm.

- (a) [10 points] Assume we have a clique tree for the network \mathcal{G} and that the CPD $P(X_i \mid \mathbf{U}_i)$ was assigned to clique \mathcal{C}_j . Analyze which messages change after we update the parameters for $P(X_i \mid \mathbf{U}_i)$. Use this analysis to show how, after an initial pre-computation step, we can perform a single iteration of EM over this single family with a computational cost that depends only on the size of \mathcal{C}_j and not the size of the rest of the cluster tree.
- (b) [5 points] Would this conclusion change if we update the parameters of several families that are all assigned to the same cluster in the cluster tree? Explain.

Estimate: 1 page, 1 1/2 hours

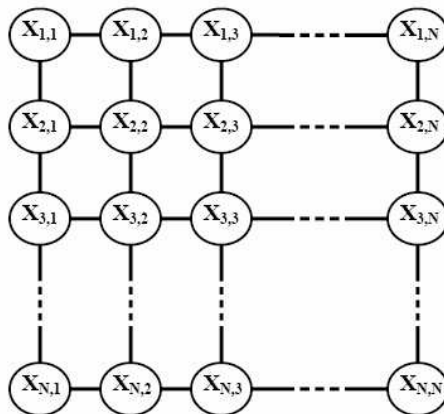
7. [15 points] **Markov Network Learning**

Assume that we are learning Markov Network parameters on a $N \times N$ grid (see Figure 3), given a dataset \mathcal{D} of M examples $\mathcal{D} = \{\xi[m] : m = 1, \dots, M\}$. As usual, we consider a log-linear parameterization of the Markov network, using a set of k features $\Phi = \{f_i : i = 1, \dots, k\}$. Each feature is defined over a pair of neighboring variables, e.g. $f_i(X_{2,2}, X_{2,3})$. There is no parameter sharing.

We define the *generalized pseudo-likelihood* objective as:

$$\ell_{gen-pseudo}(\boldsymbol{\theta} : \mathcal{D}) = \frac{1}{M} \sum_m \sum_j \ln P(\mathbf{X}_j[m] \mid \mathbf{X}_{-j}[m], \boldsymbol{\theta})$$

where $\mathbf{X}_j = \{X_{j,1}, X_{j,2}, \dots, X_{j,N}\}, \forall j = 1, \dots, N$, and \mathbf{X}_{-j} stands for all the variables on other horizontal chains, i.e. $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{j-1}, \mathbf{X}_{j+1}, \dots, \mathbf{X}_N$.

Figure 3: A $N \times N$ grid

- (a) [2 points] Write down the probability $P(\mathbf{X}_j | \mathbf{X}_{-j})$.
- (b) [10 points] Derive the gradient $\frac{\partial}{\partial \theta_i} \ln P(\mathbf{X}_j | \mathbf{X}_{-j})$.
- (c) [3 points] Derive the gradient for the *generalized pseudo-likelihood* objective, that is $\frac{\partial}{\partial \theta_i} \ell_{gen-pseudo}(\boldsymbol{\theta} : \mathcal{D})$.

Estimate: 1 1/2 pages, 1 hour

8. [8 points] Decomposable Utility Functions

Recall that a utility function is a mapping from an outcome (assignment to variables) to a real number. Suppose we have M variables, $X_1 \dots X_M$, each of which has a domain of size $|Val(X_i)| = d$, and a utility function $U(X_1 \dots X_M)$ over these variables. Our goal in this problem is to find the “ideal” outcome (i.e., the one that gives us the highest utility). Concretely, we are searching for:

$$(x_1^* \dots x_M^*) = \arg \max_{x_1 \dots x_M} U(x_1 \dots x_M).$$

Assume that this outcome is unique (that is, there are no ties for first-place outcome).

Suppose that our U is decomposable as a sum of utilities for relatively small subsets of the variables:

$$U(X_1 \dots X_M) = \sum_{i=1}^k U(\mathbf{Y}_i),$$

where $\mathbf{Y}_i \subset \{X_1 \dots X_M\}$.

- (a) [5 points] Describe an algorithm that exactly determines $(x_1^* \dots x_M^*)$ and $U(x_1^* \dots x_M^*)$ and that exploits the structure of the utility function to make it computationally more efficient. You may use as a subroutine any algorithm that we described in class.
- (b) [3 points] Show that your algorithm requires a computational complexity of $O(Md^2)$ for the case of chain decomposability, where:

$$U(X_1 \dots X_M) = \sum_{i=1}^{M-1} U(X_i, X_{i+1}).$$

Estimate: 1/2 page, 1/2 hour

9. [7 points] **Value of Imperfect Information**

In a decision problem, we know how to calculate the value of perfect information of X at decision D . Now imagine that we cannot observe the exact value of X , but we can instead observe a noisy estimate of X .

For this problem, assume X is binary. Also assume the noisy observation has a false positive rate of p and a false negative rate of q . (That is when $X = 0$ we observe 1 with probability p , and when $X = 1$ we observe 0 with probability q .)

Give a simple method by which we can calculate the improvement in MEU from observing this imperfect information. (Your answer should be just a couple lines long, but you should explain exactly how p and q are used.)

Estimate: <1/2 page, 1/2 hour