UC Berkeley
Department of Electrical Engineering and Computer Science
Department of Statistics

EECS 281A / STAT 241A Statistical Learning Theory

Solution to Problem Set 4

Fall 2012

Issued: Tues. Oct. 2, 2012 **Due:** Tues. Oct. 16, 2012

Reading: Chapters 4, 17

Problem 4.1

Sum-product algorithm: Consider the sum-product algorithm on an undirected tree with compatibility functions ψ_s and ψ_{st}. For the tree in Figure 1(a), suppose that we have ternary random variables (i.e., $X_s \in \{-1, 0, 1\}$), where the compatibility functions are of the form:

$$\psi_{st}(x_s, x_t) = \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix}, \quad \psi_s(x_s) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ for } s \text{ odd and,} \quad \psi_s(x_s) = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \text{ for } s \text{ even}$$

(Here entry (i, j) in the 3×3 matrix notation for ψ_{st} gives the value $\psi_{st}(i, j)$, whereas entry i in the 3-vector ψ_s gives the number $\psi_s(i)$).

Throughout this problem, use the serial ordering of message-passing, in which node s updates its message to t only when it has received all other incoming messages.

(a) Implement the sum-product algorithm for a general tree. Please document (meaning describe in comments within the code) what each step is doing, and hand in your documented code. Use it to compute marginal distributions for the tree in Figure 1(a), using the specified compatibility functions with

(i) $a = 1$ and $b = 0.5$

(ii) $a = 1$ and $b = 2$.

Solution: The marginals for $a = 1$, $b = 0.5$ are:

	$\mathbb{P}(X_s) = -1$	$\mathbb{P}(X_s) = 0$	$\mathbb{P}(X_s) = 1$
X_1	0.1617	0.2712	0.5671
X_2	0.4220	0.1435	0.4345
X_3	0.1639	0.2767	0.5594
X_4	0.5058	0.1422	0.3520
X_5	0.1868	0.2889	0.5243
X_6	0.4376	0.1672	0.3952

And the marginals for $a = 1$, $b = 2$ are:

	$\mathbb{P}(X_s) = -1$	$\mathbb{P}(X_s) = 0$	$\mathbb{P}(X_s) = 1$
X_1	0.1690	0.3630	0.4680
X_2	0.5395	0.1835	0.2769
X_3	0.1704	0.3695	0.4601
X_4	0.4461	0.1920	0.3619
X_5	0.1473	0.3560	0.4967
X_6	0.5344	0.1635	0.3021

The code implementing the sum product algorithm is attached below, along with the script to run the algorithm on the tree given in this problem.

```
function marginals = SumProduct(adj, psi_s, psi_st)
% This function implements the recursive SumProduct algorithm
% (without evidence) as outlined in the course reader,
% Figure 4.5.
%
% Input:
%  * adj    = n-by-n matrix with entries in {0, 1}, representing
%             the adjacency matrix of the tree.
%  * psi_s  = d-by-n matrix whose columns represent the vertex
%             potentials. We assume each random variable X_s can
%             take d possible values.
%  * psi_st = n-by-n cell, whose (s,t)-entry is a d-by-d matrix
%             representing the potential function for edge (s,t)
%             in the tree. For pairs (s,t) that are not edges
%             in the tree, psi_st(s,t) is empty.
%
% Output:
%  * marginals = d-by-n matrix whose columns are the marginals
%                of the nodes in the tree.
%

n = size(adj,1);    % number of nodes in the graph
d = size(psi_s,1);  % number of states in each variable

% Create a list of neighbors for each node
neighbors = cell(n,1);
for i = 1:n
    neighbors{i} = find(adj(i,:) == 1);
end

% Initialize messages to be all 1.
% For each edge (i,j), M{i,j} contains the message from node i
% to node j, and M{j,i} contains the message from node j to
% node i. If (i,j) is not an edge, M{i,j} is empty.
M = cell(n,n);
for i = 1:n
    for j = i+1:n
        if (adj(i,j) == 1)
            M{i,j} = ones(d,1);
            M{j,i} = ones(d,1);
        end
    end
end

% Arbitrarily choose root
root = 1;

% Collect messages from root's neighbors
for e = neighbors{root}
    Collect(root, e);
end

% Distribute messages to root's neighbors
for e = neighbors{root}
    Distribute(root, e);
end

% Compute marginals for each node
marginals = zeros(d,n);
for i = 1:n
    % Multiply the potential of node i with the messages
    % from its neighbors
    psi_i = psi_s(:,i);
    for j = neighbors{i}
        psi_i = psi_i .* M{j,i};
    end

    % Then normalize to get the marginal
    marginals(:,i) = psi_i / sum(psi_i);
end


% Helper function for collecting message from node j to
% node i. We first collect messages from node j's
% neighbors, excluding i, then send message from j to i.
function Collect(i,j)
    for k = setdiff(neighbors{j}, i);
        Collect(j,k);
    end
    SendMessage(j,i);
```

```
end


% Helper function for distributing messages from node i
% to node j. We first send message from i to j, then
% forward the message to node j's neighbors, excluding i.
function Distribute(i,j)
    SendMessage(i,j);
    for k = setdiff(neighbors{j}, i);
        Distribute(j,k);
    end
end


% Helper function for sending message from node j to
% node i. We first compute the product of psi_j(x_j)
% and m_kj(x_j) over all neighbors k of j, excluding i,
% then multiply by \psi_{ij}(x_i,x_j) and sum over x_j.
function SendMessage(j,i)
    psi_j = psi_s(:,j);
    for k = setdiff(neighbors{j},i)
        psi_j = psi_j .* M{k,j};
    end
    M{j,i} = sum(psi_st{i,j} .* repmat(psi_j', [d 1]), 2);
end

end
```

```
function marginals = RunSumProduct(a,b)
% Run the sum product algorithm for the tree given in Problem 4.1
% with the specified vertex and edge potentials.

% Adjacency matrix of the tree
adj = [0 1 1 0 0 0; ...
       1 0 0 1 1 0; ...
       1 0 0 0 0 1; ...
       0 1 0 0 0 0; ...
       0 1 0 0 0 0; ...
       0 0 1 0 0 0];

% Vertex potentials
psi_s = [1 3 1 3 1 3; ...
         2 1 2 1 2 1; ...
         3 2 3 2 3 2];

% Edge potentials
psi_template = [a b b; b a b; b b a];
```

```
psi_st = cell(size(adj));
for i = 1:size(adj,1)
    for j = i+1:size(adj,1)
        psi_st{i,j} = psi_template;
        psi_st{j,i} = psi_template;
    end
end

% Run sum product
marginals = SumProduct(adj, psi_s, psi_st);

% Display results
fprintf('Marginals for a = %g, b = %g:\n', a, b);
for i = 1:size(marginals,2)
    fprintf('X%d:  %.4f  %.4f  %.4f\n', i, marginals(:,i)');
end
```

(b) For any tree, prove that sum-product correctly computes the singleton marginals in at most tree diameter iterations. (The tree diameter is the length of the longest path between any two nodes.)

Solution:

Convergence: We first prove that the messages stabilize after D iterations, where D is the diameter of the tree. We use induction on the tree diameter D. For $D = 1$, the result is immediate. Consider a graph of diameter D. Let L be the set of all leaf nodes. For each $v \in L$, let w_v be the only element of N_v, which is a singleton set because v is a leaf node. Thus, at the first time step $t = 1$, the (fixed) message v sends to w_v is

$$M^*_{v\to w_v}(x_{w_v}) = \sum_{x_v} \psi_v(x_v)\psi_{vw_v}(x_v, x_{w_v}).$$

At each subsequent time step, the message each leaf sends to its neighbor is constant, since it does not depend on messages from any other nodes. We construct a new undirected graph G' by stripping each of the leaf nodes from the original graph and redefining the compatibility function for each w_v as

$$\psi'_{w_v}(x_{w_v}) = \psi_{w_v}(x_{w_v}) M_{v\to w_v}(x_{w_v}).$$

This new graphical model has diameter at most $D-2$, which follows because the longest path in a tree is between two leaf nodes. Therefore, we can apply the inductive hypothesis to G'. Thus, after at most $D-2$

time steps, the messages will all converge. Finally, note that replacing the nodes $v \in L$ in G' will not affect any of the fixed-point messages, since the nodes in L always send the same messages to their neighbors. Adding on the first iteration where all leaf nodes send in messages and the last iteration from nodes w_v to the leaf nodes v, we conclude that after D iterations, all messages will converge to a fixed point.

Correctness: We then prove that the fixed-point messages can be used to compute the marginals via

$$p(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in N(s)} M^*_{t \to s}(x_s).$$

We use induction on the number of nodes in the tree. When the tree has one node, the conclusion is trivial. Now let $m \geq 2$, and suppose the conclusion holds for any tree with $m-1$ nodes. We will show that the result holds for a tree on m nodes, as well.

Renumber the nodes of the tree so that node m is a leaf, with node 1 as its neighbor. In the first iteration, node m sends the message $M^*_{m \to 1}(x_1) = \sum_{x_m} \psi_m(x_m)\psi_{1m}(x_1, x_m)$ to node 1. Note that by marginalizing out x_m and using the factorization over potentials, we have

$$\begin{aligned} p(x_1, \ldots, x_{m-1}) &= \sum_{x_m} p(x_1, \ldots, x_m) \\ &\propto \; M^*_{m \to 1}(x_1) \prod_{i=1}^{m-1} \psi_i(x_i) \prod_{(j,k) \in E \setminus \{(1,m)\}} \psi_{jk}(x_j, x_k). \end{aligned}$$

In particular, all subsequent messages sent throughout the rest of the graph behave as the sum-product algorithm on the nodes $\{1, \ldots, m-1\}$, with the potential on node 1 replaced by $\psi'_1(x_1) = M^*_{m \to 1}(x_1)\psi_1(x_1)$. By the induction hypothesis, the messages on all edges between nodes $\{1, \ldots, m-1\}$ converge to the proper messages $M^*_{t \to s}(x_s)$, such that the marginal probability of each node is proportional to the product of the node potential and all incoming messages.

It remains to show that the sum-product algorithm computes the correct marginal on node m; i.e., $p(x_m) \propto \psi_m(x_m) M^*_{1 \to m}(x_m)$. Since $p(x_1)$ is proportional to the product of $\psi_1(x_1)$ and all incoming messages, and $M_{1 \to m}(x_m)$ is a sum of the product of $\psi_1(x_1)\psi_{1m}(x_1, x_m)$ and the incoming messages from nodes in $N(1)\setminus\{m\}$, we have

$$M^*_{1 \to m}(x_m) \;\propto\; \sum_{x_1} \frac{p(x_1)}{M^*_{m \to 1}(x_1)} \psi_{1m}(x_1, x_m).$$

Hence, we see that

$$\psi_m(x_m) M^*_{1\to m}(x_m) \;\propto\; \psi_m(x_m) \sum_{x_1} \frac{p(x_1)}{\sum_{x'_m} \psi_m(x'_m)\psi_{1m}(x_1, x'_m)} \psi_{1m}(x_1, x_m),$$

where we have substituted in the form of $M^*_{m\to 1}(x_1)$. Finally, note that marginalizing out the factorized form of $p(x)$ with potentials (summing over all variables except x_1 and x_m in the numerator, and all variables except x_1 in the denominator), we have

$$p(x_m|x_1) = \frac{p(x_1, x_m)}{p(x_1)} = \frac{\psi_m(x_m)\psi_{1m}(x_1, x_m)}{\sum_{x'_m} \psi_m(x'_m)\psi_{1m}(x_1, x'_m)}.$$

It follows that

$$\psi_m(x_m) M^*_{1\to m}(x_m) \;\propto\; \sum_{x_1} p(x_1) p(x_m|x_1) = p(x_m),$$

as wanted.

(c) Show that for each edge $(s, t) \in E$, the message fixed point M^* can be used to compute the pairwise joint distribution over (x_s, x_t) as follows:

$$p(x_s, x_t) \propto \psi_s(x_s)\, \psi_t(x_t)\, \psi_{st}(x_s, x_t) \prod_{u \in N(s)\backslash t} M^*_{us}(x_s) \prod_{u \in N(t)\backslash s} M^*_{ut}(x_t).$$

Solution: Contract the edge (s, t) into a supernode st with vertex potential $\widetilde{\psi}_{st}(x_s, x_t) = \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s, x_t)$, and consider running the sum product algorithm on this new tree. It is easy to see that the fixed-point messages $\widetilde{M}^*_{uv}(x_v)$ in this new tree are exactly the same as the old fixed-point messages $M^*_{uv}(x_v)$ if $v \notin \{s, t\}$. Furthermore, if u is a neighbor of st in the new tree, then $\widetilde{M}^*_{u(st)}(x_s, x_t) = M^*_{us}(x_s)$ if (s, u) is an edge in the original tree, and $\widetilde{M}^*_{u(st)}(x_s, x_t) = M^*_{ut}(x_t)$ otherwise. Therefore, using the result of part (b) for the marginal of the supernode st, we conclude that

$$\begin{aligned} p(x_s, x_t) &\propto \widetilde{\psi}_{st}(x_s, x_t) \prod_{u \in N(st)} \widetilde{M}_{u(st)}(x_s, x_t) \\ &= \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s, x_t) \prod_{u \in N(s)\backslash t} M^*_{us}(x_s) \prod_{u \in N(t)\backslash s} M^*_{ut}(x_t). \end{aligned}$$

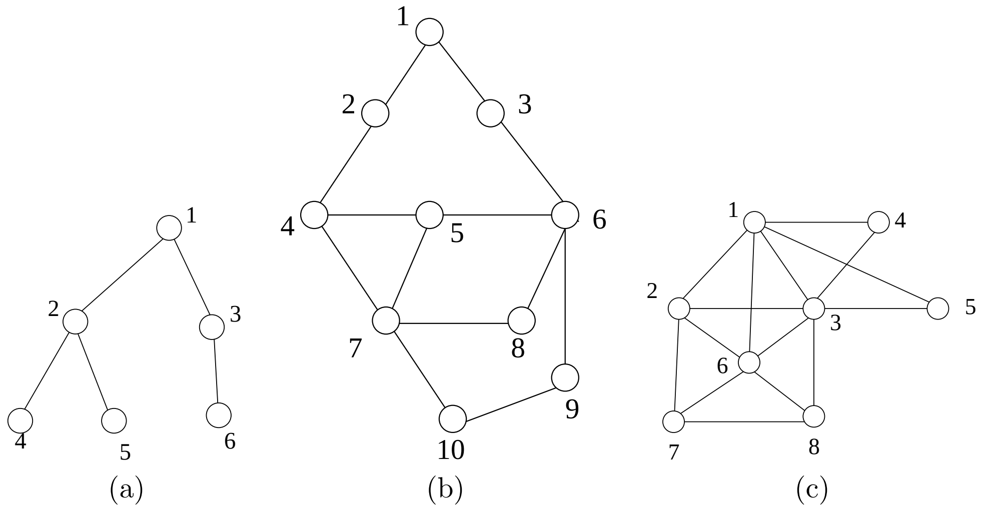

Figure 1: (a) Undirected tree (for Problem 4.1(a)). (b), (c) Some undirected graphs to triangulate (Problem 4.4).

Problem 4.2
Undirected trees and marginals: Let $G = (V, E)$ be an undirected graph. For each vertex $i \in V$, let μ_i be a strictly positive function such that $\sum_{x_i} \mu_i(x_i) = 1$. For each edge, let μ_{ij} be a strictly positive function such that $\sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j)$ for all x_j, and $\sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i)$ for all x_i. Given integers $k_1, \dots, k_d$, consider the function

$$\mathbb{Q}(x_1, \dots, x_d) \quad = \quad \prod_{i=1}^{d} \left[\mu_i(x_i)\right]^{k_i} \prod_{(i,j) \in E} \mu_{ij}(x_i, x_j).$$

Supposing that G is a tree, can you give choices of integers $k_1, \dots, k_d$ for which $\mathbb{Q}$ is a valid probability distribution? If so, prove the validity. (*Hint:* It may be easiest to first think about a Markov chain.)
Solution: We claim that we can choose $k_i = 1 - \deg_i$ to make $\mathbb{Q} \equiv \mathbb{Q}_G$ a probability distribution, where $\deg_i$ is the degree of node i in the tree G. We prove this claim by induction on the number of vertices d in the graph. When $d = 1$ the graph G only consists of a single node with degree 0, and $\mathbb{Q}_G(x_1) = \mu_1(x_1)$ is a probability distribution because $\sum_{x_1} \mu_1(x_1) = 1$. Now assume the claim is true for all trees with d vertices, and let $G = (V, E)$ be

a tree with $|V| = d+1$ vertices. We want to prove that

$$\mathbb{Q}_G(x_1, \ldots, x_d, x_{d+1}) = \prod_{i=1}^{d+1} \mu_i(x_i)^{1-\deg_i} \prod_{(i,j)\in E} \mu_{ij}(x_i, x_j)$$

is a probability distribution. Without loss of generality assume node $d+1$ is a leaf in G, and its only neighbor is node d. Since $\mu_{d+1}(x_{d+1})$ appears with a power of $1 - \deg_{d+1} = 0$, the only factor involving x_{d+1} in $\mathbb{Q}_G$ is $\mu_{d,d+1}(x_d, x_{d+1})$. Therefore, summing $\mathbb{Q}_G$ over x_{d+1} gives us

$$\begin{aligned}
\mathbb{Q}'_G(x_1, \ldots, x_d) &= \sum_{x_{d+1}} \mathbb{Q}_G(x_1, \ldots, x_d, x_{d+1}) \\
&= \left[\sum_{x_{d+1}} \mu_{d,d+1}(x_d, x_{d+1})\right] \prod_{i=1}^{d} \mu_i(x_i)^{1-\deg_i} \prod_{\substack{(i,j)\in E \\ (i,j)\neq(d,d+1)}} \mu_{ij}(x_i, x_j) \\
&= \left(\prod_{i=1}^{d-1} \mu_i(x_i)^{1-\deg_i}\right) \mu_d(x_d)^{2-\deg_d} \prod_{(i,j)\in E\setminus\{(d,d+1)\}} \mu_{ij}(x_i, x_j).
\end{aligned}$$

Let $G' = (V', E')$ be the tree that we get after removing vertex $d+1$ from G, where $V' = V \setminus \{d+1\}$ and $E' = E \setminus \{(d, d+1)\}$, and note that the degrees of the nodes in G' now satisfy $\deg'_i = \deg_i$ for $1 \le i \le d-1$, and $\deg'_d = \deg_d - 1$, since node d lost the edge $(d, d+1)$. Now observe that we can write

$$\mathbb{Q}'_G(x_1, \ldots, x_d) = \prod_{i=1}^{d} \mu_i(x_i)^{1-\deg'_i} \prod_{(i,j)\in E'} \mu_{ij}(x_i, x_j) = \mathbb{Q}_{G'}(x_1, \ldots, x_d).$$

Therefore, using the inductive hypothesis that $\mathbb{Q}_{G'}$ is a probability distribution, we conclude that

$$\sum_{x_1,\ldots,x_{d+1}} \mathbb{Q}_G(x_1, \ldots, x_{d+1}) = \sum_{x_1,\ldots,x_d} \mathbb{Q}_{G'}(x_1, \ldots, x_d) = 1,$$

as desired.

Problem 4.3
Max-marginals: For each $i \in \{1, 2, \ldots, d\}$, we define the max-marginal at node i via

$$q_i(x_i) \quad = \quad \max_{x_j, j\neq i} \mathbb{P}(x_1, x_2, \ldots, x_d).$$

When x_i takes on m-states, then q_i specifies a vector of m numbers. This is the analog of an ordinary marginal distribution, with the summation replaced by maximization.

Suppose that $\arg\max_{x_i} q_i(x_i) = \{x_i^*\}$ for each node (i.e., the maximum is uniquely attained at x_i^*). Show that $(x_1^*, x_2^*, \ldots, x_d^*)$ is the unique global optimum, meaning that $\mathbb{P}(x_1^*, \ldots, x_d^*) > \mathbb{P}(x_1, \ldots, x_d)$ for all $x \neq x^*$.

Solution: Let $x = (x_1, \ldots, x_d) \in \arg\max_y \mathbb{P}(y)$. Then for each $1 \leq i \leq d$, on the one hand we have

$$\mathbb{P}(x) \leq \max_{y_j, j \neq i} \mathbb{P}(y_1, \ldots, y_{i-1},\ x_i,\ y_{i+1}, \ldots, y_d) = q(x_i),$$

and on the other hand, because $\mathbb{P}(x) = \max_y \mathbb{P}(y)$, we also have

$$\mathbb{P}(x) \geq \max_{y_j, j \neq i} \mathbb{P}(y_1, \ldots, y_{i-1}, x_i^*, y_{i+1}, \ldots, y_d) = q(x_i^*) = \max_{y_i} q(y_i).$$

This means $x_i \in \arg\max_{y_i} q(y_i)$, which implies $x_i = x_i^*$. Since this holds for all $1 \leq i \leq d$, we conclude that $x = x^*$.

Problem 4.4
Triangulation/JT: Consider the two graphs shown in panels (b) and (c) of Figure 1. For each graph, first form a triangulated version, and then construct a junction tree using the greedy algorithm. (You can simply implement each step of the greedy algorithm on paper to find a maximum weight spanning tree.)
Solution:

(a) We triangulate the graph by running elimination algorithm on it with ordering $1, 2, 3, 4, 5, 8, 10, 6, 7, 9$ and using the reconstituted graph as the triangulated version of the initial graph (in general, we want to use elimination ordering that results in introducing as few new edges as possible). We then identify the maximum cliques and build a clique graph where the nodes are the maximum cliques and the edge weights are the cardinality of the intersection of the cliques. Finally, we construct the junction tree by finding a maximum weight spanning tree in this clique graph. See Figure 2 for a sample junction tree. (Note: this is just one of many possible solutions).

(b) The tree-building pipeline is the same as in previous part. We use elimination ordering $4, 5, 1, 2, 3, 6, 7, 8,$ which only introduces 1 new edge.

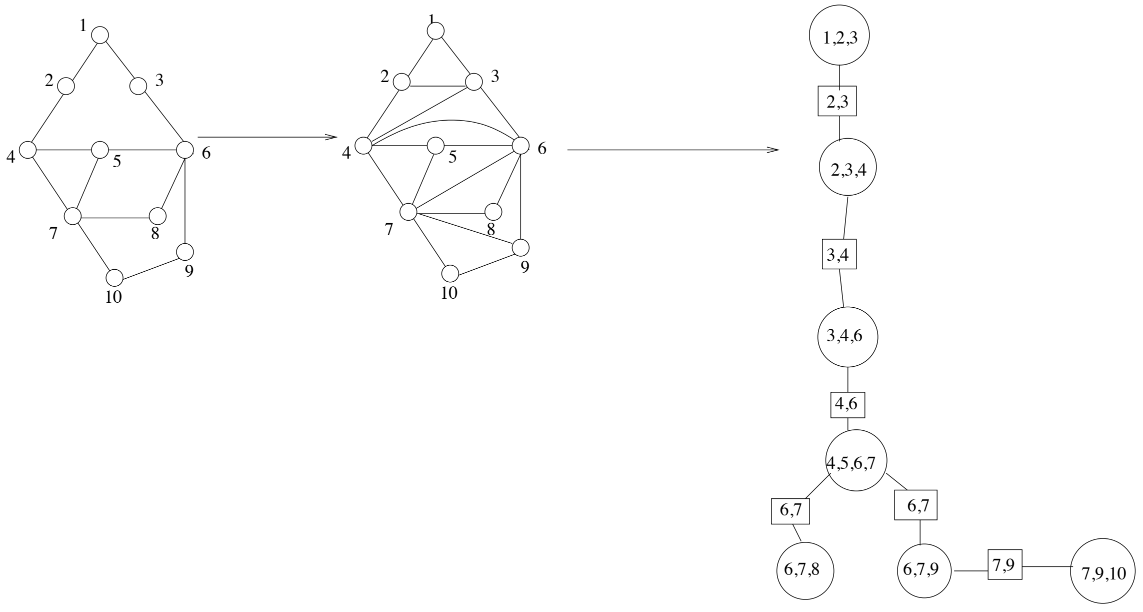

Figure 2: Problem 4.4(a): Initial graph, triangulated graph, and junction tree.