STAT 375 Homework 3 Solutions

Problem (1)

The belief propagation update equations for the grid are, up to normalization, as below:

$$\nu_{i\to j}^{(t+1)}(x_i) \cong e^{\theta_i x_i} \prod_{k\in\partial i\setminus j} \left\{ \sum_{x_k} e^{\theta_{ik} x_i x_k} \nu_{k\to i}^{(t)}(x_k) \right\}$$

Since the variables take two values only, it is somewhat easier to use the log-likelihood ratio instead of the beliefs as messages:

$$L_{i\to j}^{(t)} = \frac{1}{2}\log\left(\frac{\nu_{i\to j}^{(t)}(+1)}{\nu_{i\to j}^{(t)}(-1)}\right)$$

The update equations then become:

$$\begin{aligned}
L_{i\to j}^{(t+1)} &= \frac{1}{2}\log\left(\frac{e^{\theta_i}\prod_{k\in\partial i\setminus j}\sum_{x_k} e^{\theta_{ik}x_k}\nu_{k\to i}^{(t)}(x_k)}{e^{-\theta_i}\prod_{k\in\partial i\setminus j}\sum_{x_k} e^{-\theta_{ik}x_k}\nu_{k\to i}^{(t)}(x_k)}\right) \\
&= \theta_i + \sum_{k\in\partial i\setminus j}\frac{1}{2}\log\left(\frac{\sum_{x_k} e^{\theta_{ik}x_k}\nu_{k\to i}^{(t)}(x_k)}{\sum_{x_k} e^{-\theta_{ik}x_k}\nu_{k\to i}^{(t)}(x_k)}\right) \\
&= \theta_i + \sum_{k\in\partial i\setminus j}\frac{1}{2}\log\left(\frac{e^{(\theta_{ik}+L_{k\to i}^{(t)})} + e^{-(\theta_{ik}+L_{k\to i}^{(t)})}}{e^{(\theta_{ik}-L_{k\to i}^{(t)})} + e^{-(\theta_{ik}-L_{k\to i}^{(t)})}}\right)
\end{aligned}$$

Since $\frac{1}{2}\log(z) = \operatorname{arctanh}\left(\frac{z-1}{z+1}\right)$, the above simplifies to:

$$L_{i\to j}^{(t)} = \theta_i + \sum_{k\in\partial i\setminus j} \operatorname{arctanh}\left\{\tanh(\theta_{ik})\tanh(L_{k\to i}^{(t)})\right\}$$

Problem (2)

The code for implementing belief propagation on a grid is given below:

```
clear all
beta_vals = (0.1:0.1:3);
l = 10;
E = 4*l*(l-1);
Tmax_vals = [40 80];

delta = zeros(length(beta_vals), length(Tmax_vals));

b_hor_fixed = (rand(l, l-1)-0.5);
b_vert_fixed = (rand(l-1, l)-0.5);
b_nodes_fixed = (rand(l, l)-0.5);

for iter = 1:length(beta_vals)


    Tmax = max(Tmax_vals);
    beta = beta_vals(iter);
    %initialization
    b_hor = b_hor_fixed*beta*2;
    b_vert = b_vert_fixed*beta*2;
    b_nodes = b_nodes_fixed*beta*2;

    %The following four matrices contain the messages nu_{t}
    h_hor_right = zeros(l, l-1);
    h_hor_left = zeros(l, l-1);
    h_vert_up = zeros(l-1, l);
    h_vert_down = zeros(l-1, l);

    %The following matrices contain the updated messages, ...
        i.e. nu_{t+1}
    h_hor_right_new = zeros(l, l-1);
    h_hor_left_new = zeros(l, l-1);
    h_vert_up_new = zeros(l-1, l);
    h_vert_down_new = zeros(l-1, l);
    for t = 1:Tmax

        %looping over edges with 3 inputs
        for i = 2:(l-1)
            for j = 2:(l-1)
                h_hor_right_new(i, j) = b_nodes(i, j) + ...
                sum(atanh([tanh(b_hor(i, ...
                    j-1))*tanh(h_hor_right(i, j-1)) ...
                       tanh(b_vert(i-1, ...
                           j))*tanh(h_vert_down(i-1, j)) ...
```

```
                        tanh(b_vert(i, j))*  ...
                            tanh(h_vert_up(i, j))]));
                h_vert_down_new(i, j) = b_nodes(i, j) + ...
                sum(atanh([tanh(b_vert(i-1, ...
                    j))*tanh(h_vert_down(i-1, j)) ...
                        tanh(b_hor(i, ...
                            j-1))*tanh(h_hor_right(i, j-1)) ...
                        tanh(b_hor(i, j))*tanh(h_hor_left(i, ...
                            j))]));
            end
        end

        for i = 2:(l-1)
            for j = 1:(l-2)
                h_hor_left_new(i, j) = b_nodes(i, j+1) + ...
                sum(atanh([tanh(b_hor(i, ...
                    j+1))*tanh(h_hor_left(i, j+1)) ...
                         tanh(b_vert(i-1, ...
                             j+1))*tanh(h_vert_down(i-1, ...
                             j+1)) ...
                         tanh(b_vert(i, ...
                             j+1))*tanh(h_vert_up(i, j+1))]));

                h_vert_up_new(j, i) = b_nodes(j+1, i) + ...
                sum(atanh([tanh(b_vert(j+1, ...
                    i))*tanh(h_vert_up(j+1, i)) ...
                         tanh(b_hor(j+1, ...
                             i-1))*tanh(h_hor_right(j+1, ...
                             i-1)) ...
                         tanh(b_hor(j+1, ...
                             i))*tanh(h_hor_left(j+1, i))]));
            end
        end

        %looping over edges with 2 inputs
        for i = 2:(l-1)
            h_vert_up_new(l-1, i) = b_nodes(l, i) + ...
            sum(atanh([tanh(b_hor(l, ...
                i-1))*tanh(h_hor_right(l, i-1)) ...
                        tanh(b_hor(l, i))*tanh(h_hor_left(l, ...
                            i))]));

            h_vert_down_new(1, i) = b_nodes(1, i) + ...
            sum(atanh([tanh(b_hor(1, ...
                i-1))*tanh(h_hor_right(1, i-1)) ...
                        tanh(b_hor(1, i))*tanh(h_hor_left(1, ...
                            i))]));

            h_hor_right_new(i, 1) = b_nodes(i, 1) + ...
```

```
            sum(atanh([tanh(b_vert(i-1, ...
                1))*tanh(h_vert_down(i-1, 1)) ...
                    tanh(b_vert(i, 1))*tanh(h_vert_up(i, ...
                        1))]));

            h_hor_left_new(i, l-1) = b_nodes(i, l) + ...
            sum(atanh([tanh(b_vert(i-1, ...
                l))*tanh(h_vert_down(i-1, l)) ...
                    tanh(b_vert(i, l))*tanh(h_vert_up(i, ...
                        l))]));

        end

        for i = 2:(l-1)
             h_vert_down_new(i, 1) = b_nodes(i, 1) + ...
             sum(atanh([tanh(b_vert(i-1, ...
                 1))*tanh(h_vert_down(i-1, 1)) ...
                    tanh(b_hor(i, 1))*tanh(h_hor_left(i, ...
                        1))]));

             h_vert_down_new(i, l) = b_nodes(i, l) + ...
             sum(atanh([tanh(b_vert(i-1, ...
                 l))*tanh(h_vert_down(i-1, l))
                    tanh(b_hor(i, ...
                        l-1))*tanh(h_hor_right(i-1, ...
                        l-1))]));


             h_hor_right_new(1, i) = b_nodes(1, i) + ...
             sum(atanh([tanh(b_hor(1, ...
                 i-1))*tanh(h_hor_right(1, i-1)) ...
                    tanh(b_vert(1, i))*tanh(h_vert_up(1, ...
                        i))]));

             h_hor_right_new(l, i) = b_nodes(l, i) + ...
             sum(atanh([tanh(b_hor(l, ...
                 i-1))*tanh(h_hor_right(l, i-1)) ...
                    tanh(b_vert(l-1,i))*tanh(h_vert_down(l-1, ...
                        i))]));


        end

        for i = 1:(l-2)
             h_vert_up_new(i, 1) = b_nodes(i+1, 1) + ...
             sum(atanh([tanh(b_vert(i+1, ...
                 1))*tanh(h_vert_up(i+1, 1)) ...
                    tanh(b_hor(i+1, ...
                        1))*tanh(h_hor_left(i+1, 1))]));
```

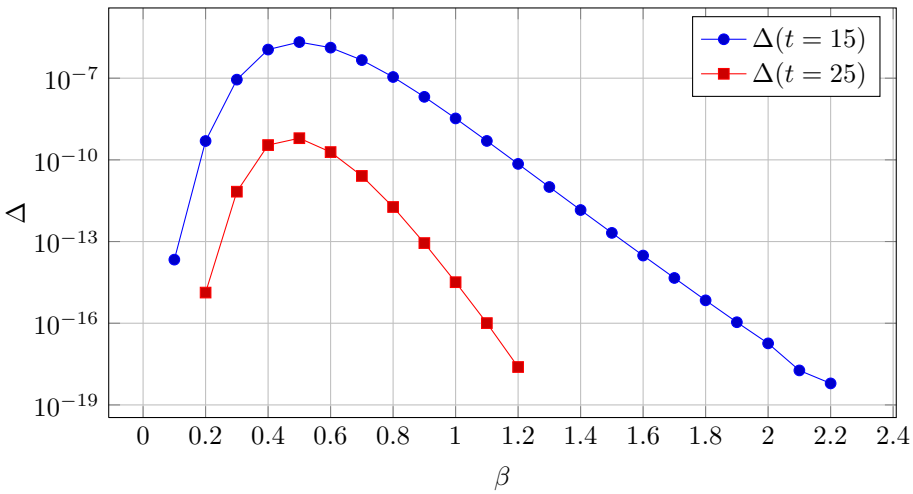
```
            h_vert_up_new(i, l) = b_nodes(i+1, l) + ...
            sum(atanh([tanh(b_vert(i+1, ...
                l))*tanh(h_vert_up(i+1,l)) ...
                    tanh(b_hor(i+1, ...
                        l-1))*tanh(h_hor_right(i+1, ...
                        l-1))]));

            h_hor_left_new(1, i) = b_nodes(1, i+1) + ...
            sum(atanh([tanh(b_hor(1, ...
                i+1))*tanh(h_hor_left(1, i+1)) ...
                    tanh(b_vert(1, ...
                        i+1))*tanh(h_vert_up(1, i+1))]));

            h_hor_left_new(l, i) = b_nodes(l, i+1) + ...
            sum(atanh([tanh(b_hor(l, ...
                i+1))*tanh(h_hor_left(l, i+1))
                    tanh(b_vert(l-1, ...
                        i+1))*tanh(h_vert_down(l-1, ...
                        i+1))]));

        end

        %Edges with single input
        h_vert_down_new(1, 1) = b_nodes(1, 1) + ...
            atanh(tanh(b_hor(1, 1))*tanh(h_hor_left(1, 1)));
        h_vert_down_new(1, l) = b_nodes(1, l) + ...
            atanh(tanh(b_hor(l-1, 1))*tanh(h_hor_right(l-1, ...
                1)));
        h_vert_up_new(l-1, 1) = b_nodes(l, 1) + ...
            atanh(tanh(b_hor(l, 1))*tanh(h_hor_left(l, 1)));
        h_vert_up_new(l-1, l) = b_nodes(l, l) + ...
            atanh(tanh(b_hor(l, l-1))*tanh(h_hor_right(l, ...
                l-1)));

        h_hor_right_new(1, 1) = b_nodes(1, 1) + ...
            atanh(tanh(b_vert(1, 1))*tanh(h_vert_up(1, 1)));
        h_hor_right_new(l, 1) = b_nodes(l, 1) + ...
            atanh(tanh(b_vert(l-1, ...
                1))*tanh(h_vert_down(l-1, l)));
        h_hor_left_new(1, l-1) = b_nodes(1, l) +...
            atanh(tanh(b_vert(1, l-1))*tanh(h_vert_up(1, ...
                l-1)));
        h_hor_left_new(l, l-1) = b_nodes(l, l) + ...
            atanh(tanh(b_vert(l-1, ...
                l))*tanh(h_vert_down(l-1, l)));
```

```
        nu_vert_down_new = myexp(h_vert_down_new);
        nu_vert_up_new = myexp(h_vert_up_new);
        nu_hor_right_new = myexp(h_hor_right_new);
        nu_hor_left_new = myexp(h_hor_left_new);

        nu_vert_down = myexp(h_vert_down);
        nu_vert_up = myexp(h_vert_up);
        nu_hor_right = myexp(h_hor_right);
        nu_hor_left = myexp(h_hor_left);

        h_vert_down = h_vert_down_new;
        h_vert_up = h_vert_up_new;
        h_hor_left = h_hor_left_new;
        h_hor_right = h_hor_right_new;

        if(sum(t==Tmax_vals))
            iter2 = find(t==Tmax_vals);
            delta(iter, iter2) = (mynorm(nu_vert_down_new, ...
                nu_vert_down) ...
                            + mynorm(nu_vert_up_new, ...
                                nu_vert_up) ...
                            + mynorm(nu_hor_right_new, ...
                                nu_hor_right) ...
                            + mynorm(nu_hor_left_new, ...
                                nu_hor_left))/E;

        end
    end



end
semilogy(beta_vals, delta)
grid on
```

Problem (3)

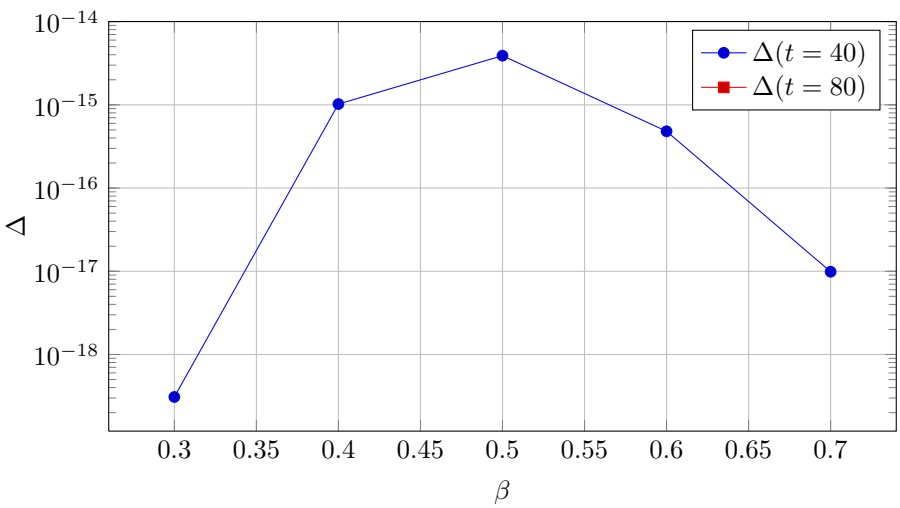
The algorithm converges to a fixed point with high accuracy for the iteration numbers mentioned. Lower iteration numbers yield a slightly better picture.

At low and high values of β, the algorithm converges quickly. For low values this is expected since the interactions along edges are weak and the marginals are approximately the normalized node potentials (corresponding to the graph with no edges). For intermediate values of beta the convergence



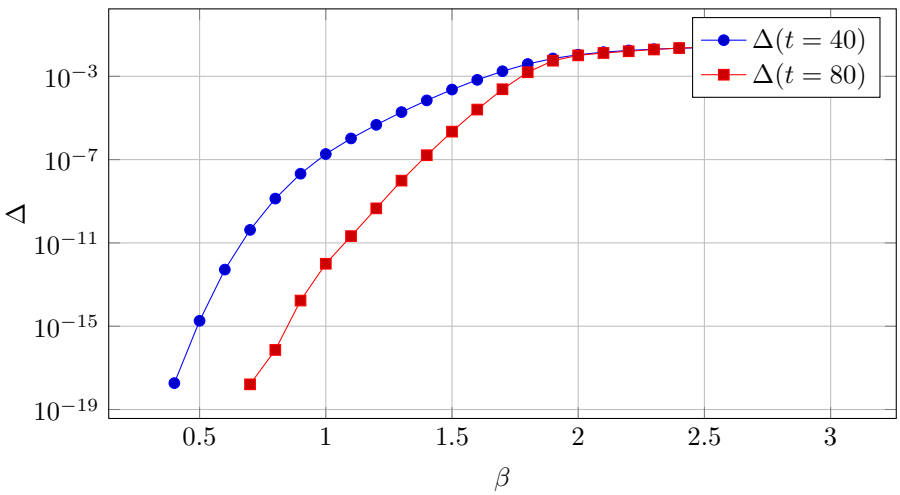
is slower.

For reference, the plot for larger iterations is as below:



Problem (4)

The algorithm does not converge, except at very low values of beta. For these values, the measure is approximately independent over the vertices of the graph.


$\Delta(t = 40)$
$\Delta(t = 80)$
10^{-3}
10^{-7}
10^{-11}
10^{-15}
10^{-19}
Δ
0.5
1
1.5
2
2.5
3
β