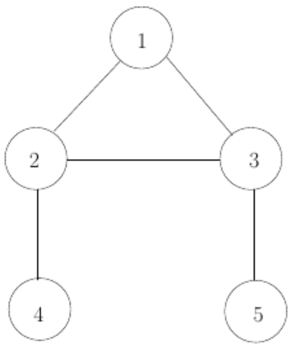
IE 598 Inference in Graphical Models

Homework 3

Due-11/06/2012

Problem 3.1 Consider the following graphical model.



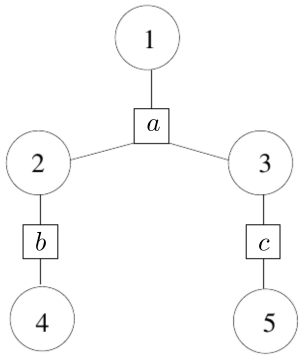
(a) Draw a factor graph representing the graphical model and specify the factor graph message-passing equations. For this particular example, explain why the factor graph message-passing equations can be used to compute the marginals, but the sum-product equations cannot be used.

(b) Define a new random variable $x_6 = \{x_1, x_2, x_3\}$, i.e., we group variables x_1, x_2, and x_3 into one variable. Draw an undirected graph which captures the relationship between x_4, x_5, and x_6. Explain why you can apply the sum-product algorithm to your new graph to compute the marginals. Compare the belief propagation equations for the new graph with the factor graph message-passing equations you obtained in part (a).

(c) If we take the approach from part (b) to the extreme, we can simply define a random variable $x_7 = \{x_1, x_2, x_3, x_4, x_5\}$, i.e., define a new random variable which groups all five original random variables together. Explain what running the sum-product algorithm on the corresponding one vertex graph means. Assuming that we only care about the marginals for $x_1, x_2, \ldots, x_5$, can you think of a reason why we would prefer the method in part (b) to the method in this part, i.e., why it might be preferable to group a smaller number of variables together?

Solution 3.1

(a) The factor graph is a tree, so factor graph message-passing gives the exact answer. Since the original undirected graph has a cycle, sum-product algorithm is not guaranteed to converge to the correct answer.

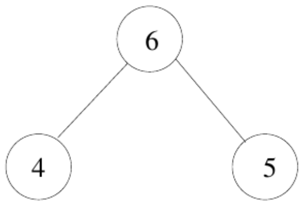


The message update rules are:

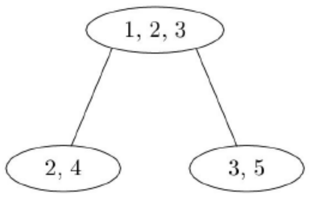
$$\begin{aligned}
\nu_{1\to a}(x_1) &= 1/|\mathcal{X}| \\
\nu_{4\to b}(x_4) &= 1/|\mathcal{X}| \\
\nu_{5\to c}(x_5) &= 1/|\mathcal{X}| \\
\nu_{b\to 2}(x_2) &= \sum_{x_4} (1/|\mathcal{X}|)\psi_{24}(x_2,x_4) \\
\nu_{c\to 3}(x_3) &= \sum_{x_5} (1/|\mathcal{X}|)\psi_{35}(x_3,x_5) \\
\nu_{2\to a}(x_2) &= \nu_{b\to 2}(x_2) \\
\nu_{3\to a}(x_3) &= \nu_{c\to 3}(x_3) \\
\nu_{a\to 1}(x_1) &= \sum_{x_2,x_3} \psi_{123}(x_1,x_2,x_3)\nu_{2\to a}(x_2)\nu_{3\to a}(x_3) \\
\nu_{a\to 2}(x_2) &= \sum_{x_1,x_3} \psi_{123}(x_1,x_2,x_3)\nu_{1\to a}(x_1)\nu_{3\to a}(x_3) \\
\nu_{a\to 3}(x_3) &= \sum_{x_1,x_2} \psi_{123}(x_1,x_2,x_3)\nu_{1\to a}(x_1)\nu_{2\to a}(x_2) \\
\nu_{2\to b}(x_2) &= \nu_{a\to 2}(x_2) \\
\nu_{3\to c}(x_3) &= \nu_{a\to 3}(x_3) \\
\nu_{b\to 4}(x_4) &= \sum_{x_2} \psi_{24}(x_2,x_4) \\
\nu_{c\to 5}(x_5) &= \sum_{x_3} \psi_{35}(x_3,x_5)
\end{aligned}$$

(b) Grouping $x_6 = \{x_1, x_2, x_3\}$, we get a tree. Then, we can apply sum-product algorithm with $\psi_{46}(x_1,x_2,x_3,x_4) = \psi_{24}(x_2,x_4)$, $\psi_{56}(x_1,x_2,x_3,x_5) = \psi_{35}(x_3,x_5)$, and $\psi_6(x_1,x_2,x_3) = \psi_{123}(x_1,x_2,x_3)$.

$$\begin{aligned}
\nu_{4\to 6}(x_4) &= 1/|\mathcal{X}| \\
\nu_{5\to 6}(x_5) &= 1/|\mathcal{X}| \\
\nu_{6\to 5}(x_6) &= \sum_{x_4} (1/|\mathcal{X}|)\psi_6(x_6)\psi_{46}(x_4,x_6) \\
\nu_{6\to 4}(x_6) &= \sum_{x_5} (1/|\mathcal{X}|)\psi_6(x_6)\psi_{56}(x_5,x_6)
\end{aligned}$$



These equations involve similar computations to those done by the factor graph message-passing equations in part (a). However, there is one minor difference. Specifically, in factor graph message-passing equations, we never had to deal with more than 3 variables at a time. The sum-product equations above have several steps where four variables are involved (we count x_6 are three variables since summing over x_6 is more costly than summing over x_4 or x_5, assuming that all the original variables have the same alphabet size). Thus, the factor graph message-passing equations are slightly more efficient than the sum-product equations. We note that if the variables are grouped a slightly different fashion, we can make the sum-product algorithm have the same complexity as the factor graph message-passing algorithm from part (a). The idea is to form a better junction tree as shown below:



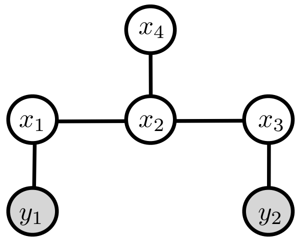
This is a junction tree, and the sum-product algorithm with trivial modifications involves only three variables, matching the factor graph complexity. Using our standard notation, assume that the original graph has clique potentials $\psi_{123}, \psi_{24}, \psi_{35}$. Then, we absorb ψ's into node potentials of the junction tree and let the edge potentials enforce the consistency on the common nodes, i.e., $\psi_{123,24} = \mathbb{I}(x_2 = x_2')$ and $\psi_{123,35} = \mathbb{I}(x_3 = x_3')$, which means that the x_2 in the node $\{2,4\}$ must be identical to the x_2 in the node $\{1,2,3\}$ node, etc.

Let us see what happens in the sum-product belief propagation. $\nu_{24\to123}(x_{2,4}) = \psi_{24}(x_{2,4})$, and $\nu_{123\to35}(x_{1,2,3}) = \psi_{123}(x_1,x_2,x_3)\sum_{x_4}\psi_{24}(x_2,x_4)$. We can avoid repeating the same computation by only summing out x_4. Now, $\nu_{35\to123}(x_{3,5}) = \psi_{35}(x_{3,5})$, and $\nu_{123\to24}(x_{1,2,3}) = \psi_{123}(x_1,x_2,x_3)\sum_{x_5}\psi_{35}(x_3,x_5)$. Again, we can avoid repeating the same computation by summing out x_5 first. Thus, the junction tree can match the factor graph in terms of complexity. In fact, as we have seen, the sum-product with a few trivial modifications (i.e., don't repeat the same computation/don't sum over stuff we know is zero) gets the same complexity as factor graph message-passing. The modifications are essentially what is known as the Shafer-Shenoy algorithm.

(c) Because the graph has a single node, sum-product does nothing. If we want to get the marginal for an individual variable, say x_1, then this approach would use brute-force, i.e., sum over all values for the remaining variables. Thus, we have a computation involving all 5 variables, whereas in part (b) we never had to deal with more than 4 variables at a time. Thus, although grouping variables can produce a tree, there is a cost to be paid - the more variables that are put into a single group, the higher the computational cost for the sum-product algorithm.

Problem 3.2 Let $x \sim \mathcal{N}^{-1}(h_x, J_x)$, and $y = Cx + v$, where $v \sim \mathcal{N}(0, R)$.

1. Find the potential vector $h_{y|x}$ and the information matrix $J_{y|x}$ of $p(y|x)$.
2. Find the potential vector $h_{x,y}$ and the information matrix $J_{x,y}$ of $p(x,y)$.
3. Find the potential vector $h_{x|y}$ and the information matrix $J_{x|y}$ of $p(x|y)$.
4. Consider the following Gaussian graphical model.



   Let $y_1 = x_1 + v_1$, $y_2 = x_3 + v_2$, and $R = I$ is the identity matrix. Find C. Represent messages $h_{x_3 \to x_2}$ and $J_{x_3 \to x_2}$ in terms of y_2 and the elements of h_x and J_x.

5. Now assume that we have an additional measurement $y_3 = x_3 + v_3$, where v_3 is a zero-mean Gaussian variable with variance 1 and is independent from all other variables. Find the new C. Represent messages $h_{x_3 \to x_2}$ and $J_{x_3 \to x_2}$ in terms of y_2, y_3 and the elements of h_x and J_x.
6. The BP message from x_3 to x_2 define a Gaussian distribution with mean $m_{x_3 \to x_2} = J^{-1}_{x_3 \to x_2} h_{x_3 \to x_2}$ and variance $\sigma_{x_3 \to x_2} = J^{-1}_{x_3 \to x_2}$. Comment on the difference in the mean and the variance of this message when computed using a single observation y_2 versus when computed using multiple observations (y_2, y_3). Can you guess the mean and variance of the BP message when the number of observations grows to infinity?

Solution 3.2

1.

$$p(y|x) \quad \propto \quad \exp\Big\{ -\frac{1}{2}(y - Cx)^T R^{-1}(y - Cx)\Big\}$$

   Then, $h_{y|x} = x^T C^T R^{-1}$ and $J_{y|x} = R^{-1}$.

2.

$$\begin{aligned} p(x,y) \quad &\propto \quad \exp\Big\{ -\frac{1}{2}(y - Cx)^T R^{-1}(y - Cx)\Big\} \exp\Big\{ -\frac{1}{2}x^T J_x x + h_x^T x\Big\} \\ &= \quad \exp\Big\{ -\frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} J_x + C^T R^{-1} C & -C^T R^{-1} \\ -R^{-1}C & R^{-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h_x \\ 0 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} \Big\} \end{aligned}$$

   Then, $h_{x,y} = \begin{bmatrix} h_x \\ 0 \end{bmatrix}$ and $J_{x,y} = \begin{bmatrix} J_x + C^T R^{-1} C & -C^T R^{-1} \\ -R^{-1}C & R^{-1} \end{bmatrix}$.

3. Computing conditionals from the information form is easy. $h_{x|y} = h_x + C^T R^{-1} y$ and $J_{x|y} = J_x + C^T R^{-1} C$.

4. For the given graphical model, $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, since $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$. From the Gaussian BP update, we know that

$$\begin{aligned} h_{x_3 \to x_2} &= h_{x_3|y_2} &= h_{x_3} + y_2 \\ J_{x_3 \to x_2} &= J_{x_3|y_2} &= J_{x_3,x_3} + 1 \end{aligned}$$

$h_{x_3 \to x_2}$ corresponds to our belief about the potential vector of random variable x_3 given the observations, and $J_{x_3 \to x_2}$ corresponds to our belief about the information matrix of random variable x_3 given the observations.

5. With the new observation, $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. From the Gaussian BP update, we get that

$$\begin{aligned} h_{x_3 \to x_2} &= h_{x_3|y_2} &= h_{x_3} + y_2 + y_3 \\ J_{x_3 \to x_2} &= J_{x_3|y_2} &= J_{x_3,x_3} + 2 \end{aligned}$$

6. With one observation, the message is $\mathcal{N}(\frac{h_{x_3}+y_2}{J_{x_3,x_3}+1}, \frac{1}{J_{x_3,x_3}+1})$. With two observations, the message is $\mathcal{N}(\frac{h_{x_3}+y_2+y_3}{J_{x_3,x_3}+2}, \frac{1}{J_{x_3,x_3}+2})$. As we get more observations, we get more accurate beliefs with smaller variance. As we increase the number of observations m, our belief converges to a deterministic scalar value $\frac{1}{m}\sum_{i=1}^{m} y_i$.

Problem 3.3 In this exercise, you will construct an undirected graphical model for the problem of segmenting foreground and background in an image, and use loopy belief propagation to solve it. Load the image `flower.bmp` into MATLAB using `imread`. (The command `imshow` may also come in handy.) Partial labeling of the foreground and background pixels are given in the mask images `foreground.bmp` and `background.bmp`, respectively. In each mask, the white pixels indicate positions of representative samples of foreground or background pixels in the image. Let $y = \{y_i\}$ be an observed color image, so each y_i is a 3-vector (of RGB values between 0 and 1) representing the pixel indexed by i. Let $x = \{x_i\}$, where $x_i \in \{0, 1\}$ 2 is a foreground(1)/background(0) labeling of the image at pixel i. Let us say the probabilistic model for x and y given by their joint distribution can be factored in the form

$$\mu(x, y) = \frac{1}{Z} \prod_i \phi(x_i, y_i) \prod_{(j,k) \in E} \psi(x_j, x_k) \qquad (1)$$

where E is the set of all pairs of adjacent pixels in the same row or column as in 2-dimensional grid. Suppose that we choose

$$\psi(x_j, x_k) = \begin{cases} 0.9 & \text{if } x_j = x_k \\ 0.1 & \text{if } x_j \neq x_k \end{cases}$$

This encourages neighboring pixels to have the same label–a reasonable assumption. Suppose further that we use a simple model for the conditional distribution $\phi(x_i, y_i) = \mathbb{P}_{Y_i|X_i}(y_i|x_i)$:

$$\mathbb{P}(y_i | x_i = \alpha) \propto \frac{1}{(2\pi)^{3/2}\sqrt{\det \Lambda_\alpha}} \exp\Big\{ -\frac{1}{2}(y_i - \mu_\alpha)^T \Lambda_\alpha^{-1} (y_i - \mu_\alpha) \Big\} + \epsilon$$

for $y_i \in [0, 1]^3$. That is, the distribution of color pixel values over the same type of image region is a modified Gaussian distribution, where ϵ accounts for outliers. Set $\epsilon = 0.01$ in this problem.

(a) Sketch an undirected graphical model that represents $\mu(x, y)$.

(*b*) Compute $\mu_\alpha \in \mathbb{R}^3$ and $\Lambda_\alpha \in \mathbb{R}^{3\times 3}$ for each $\alpha \in \{0,1\}$ from the labeled masks by finding the sample mean and covariance of the RGB values of those pixels for which the label $x_i = \alpha$ is known from `foreground.bmp` and `background.bmp`. The sample mean of samples $\{y_1, \ldots, y_N\}$ is $\bar{y} = \frac{1}{N}\sum_{i=1}^N y_i$ and the sample covariance is $C_y = \frac{1}{N-1}\sum_{i=1}^N (y_i - \bar{y})(y_i - \bar{y})^T$.

(*c*) We want to run the sum-product algorithm on the graph iteratively to find (approximately) the marginal distribution $\mu(x_i|y)$ at every i. For a joint distribution of the form (1) with pairwise compatibility functions and singleton compatibility functions, the local message update rule for passing the message $\nu_{j\to k}(x_j)$ from x_j to x_k, is represented in terms of the messages from the other neighbors of x_j, the potential functions.

$$\nu_{j\to k}(x_j) \propto \phi(x_j, y_j) \prod_{u\in\partial j\setminus k} \sum_{x_u} \psi(x_j, x_u)\nu_{u\to j}(x_u)$$

Then the final belief on x_j is computed as

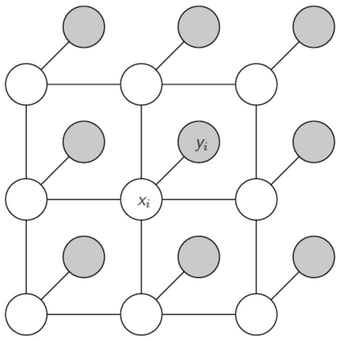
$$\nu_j(x_j) \propto \phi(x_j, y_j) \prod_{u\in\partial j} \sum_{x_u} \psi(x_j, x_u)\nu_{u\to j}(x_u)$$

Implement the sum-product algorithm for this problem. There are four directional messages: down, up, left, and right, coming into and out of each x_i (except at the boundaries). Use a parallel update schedule, so all messages at all x_i are updated at once. Run for 30 iterations (or you can state and use some other reasonable termination criterion). Since we are working with binary random variables, perhaps it is easier to pass messages in log-likelihood. Feel free to start from `gridbp.m` and fill in the missing functions (most of the algorothm is already implemented).
After the marginal distributions at the pixels are estimated, visualize their expectation. Where are the beliefs "weak"?
Visualize the expectation after 1, 2, 3, and 4 iterations. Qualitatively, discuss where the loopy belief propagation converge first and last.
Run BP with a different value of $\epsilon = 0$ and comment on the result.
Run BP with a different pairwise potential and comment on the result.

$$\psi(x_j, x_k) = \begin{cases} 0.6 & \text{if } x_j = x_k \\ 0.4 & \text{if } x_j \neq x_k \end{cases}$$

Solution 3.3

(a) We can use the following graphical model.



(b) The sample statistics are:

$$\mu_1 = \begin{bmatrix} 0.6600 \\ 0.3733 \\ 0.0333 \end{bmatrix}, \quad \Lambda_1 = \begin{bmatrix} 0.0558 & 0.0632 & 0.0009 \\ 0.0632 & 0.0807 & 0.0009 \\ 0.0009 & 0.0009 & 0.0004 \end{bmatrix}$$

$$\mu_0 = \begin{bmatrix} 0.3090 \\ 0.3102 \\ 0.1617 \end{bmatrix}, \quad \Lambda_0 = \begin{bmatrix} 0.0787 & 0.0712 & 0.0482 \\ 0.0712 & 0.0682 & 0.0463 \\ 0.0482 & 0.0463 & 0.0340 \end{bmatrix}$$

(c) Note that the messages can be represented by two-vectors, as they are functions of x_k, which take on binary values. During the first iteration, the messages from x neighbors should be uninformative (equivalent to not being in the product of the sum-product), so these messages are initialized to [0.5 0.5] in the algorithm. Here is the expectation of the marginal beliefs, or well just call them beliefs, since they are binary. We see that the beliefs are "weakest" (or rather, most ambivalent) along the

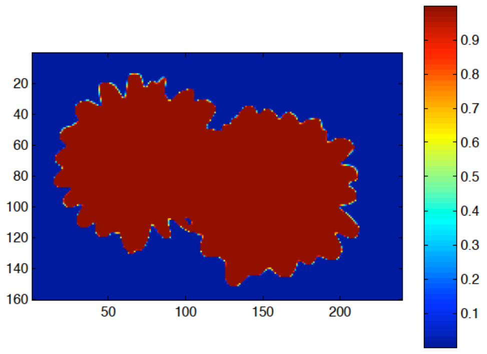


Figure 1: Visualized is $E[x|y]$. Values nearer to 0.5 are more ambivalent beliefs.

boundary between foreground and background.

Next, we show the beliefs based on local evidence alone, that is, the values you have before any messages are passed.

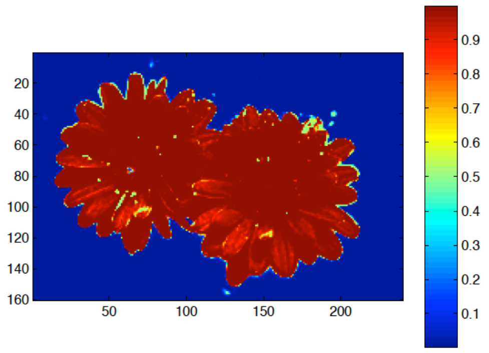


Figure 2: Beliefs at iteration 0.

The absolute-value changes to the beliefs for the first 4 iterations are shown next. As the iterative algorithm runs, more and more variable nodes “settle” as the messages coming into them “settle.”

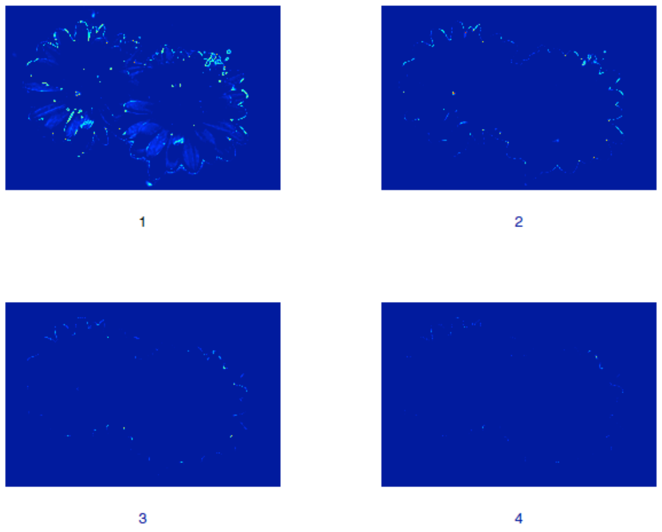


Figure 3: Most beliefs become firm very quickly.

The last places to converge are atypical regions, such as what may be mistaken as background in a foreground region, and vice versa, notably, on the boundary where the final beliefs turn out to be closest to [0.5 0.5].

Loopy belief propagation actually works really well here. These beliefs can be used as an alpha map to cut out the foreground, for example:

When $\epsilon = 0$, the algorithm is less robust to outliers, and we see a lot of ‘holes’ in the foreground due to outliers.

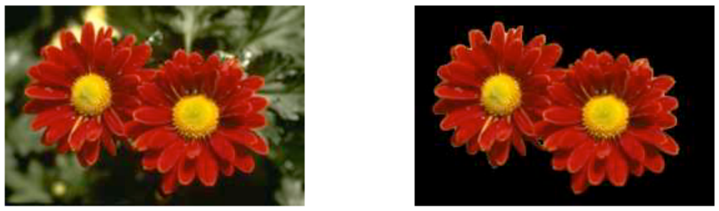


Figure 4: The original image and its alpha-masked foreground cutout.

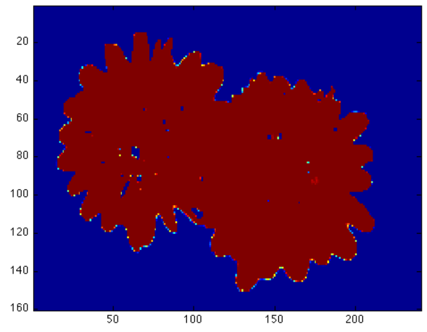


Figure 5: BP estimation after 5 iterations with $\epsilon = 0$.

When $\psi(x_j, x_k) = 0.6$ if $x_j = x_k$ and 0.4 if $x_j \neq x_k$, the correlation between adjacent nodes are weaker and the estimation is closer to what is given by the singleton potentials.

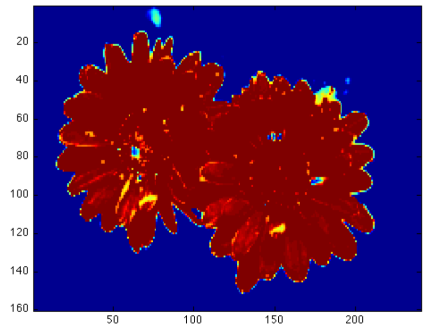
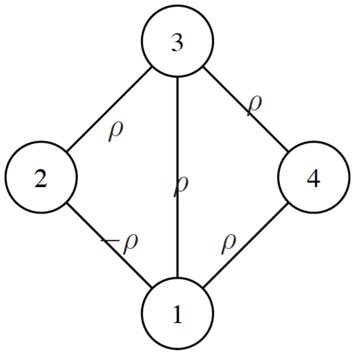


Figure 6: BP estimation with a new pairwise potential.

Problem 3.4 Consider the Gaussian graphical model depicted below. More precisely, if we let x denote the 4-dimensional vector of variables at the 4 nodes (ordered according to the node numbering given), then $x \sim]cN^{-1}(h, J)$, where J has diagonal values all equal to 1 and non-zero off-diagonal entries as indicated in the figure (e.g., $J_{12} = -\rho$).



(a) Confirm (e.g., by checking Sylvesters criterion to see if the determinants of all principal minors are positive) that J is a valid information matrix–i.e., it is positive definite–if $\rho = .39$ or $\rho = .4$. Compute the variances for each of the components (i.e., the diagonal elements of $\Lambda = J^{-1}$)–you can use Matlab to do this if youd like.

(b) We now want to examine Loopy BP for this model, focusing on the recursions for the information matrix parameters. Write out these recursions in detail for this model. Implement these recursions and try for $\rho = .39$ and $\rho = .4$. Describe the behavior that you observe.

(c) Construct the computation tree for this model. Note that the effective "J" – parameters for this model are copies of the corresponding ones for the original model (so that every time the edge $(1, 2)$ appears in the computation tree, the corresponding J-component is $-\rho$). Use Matlab to check the positive-definiteness of these implied models on computation trees for different depths and for our two different values of ρ. What do you observe that would explain the result in part (b)?

Solution 3.4

(a) The information matrix is given by:

$$\begin{bmatrix} 1 & -\rho & \rho & \rho \\ -\rho & 1 & \rho & 0 \\ \rho & \rho & 1 & \rho \\ \rho & 0 & \rho & 1 \end{bmatrix}$$

The eigenvalues are $(2\rho + 1, \rho + 1, -2\rho + 1, -\rho + 1)$. When $-0.5 < \rho < 0.5$, all eigenvalues are positive; thus J is positive definite for $rho = 0.39$ and $\rho = 0.4$. We can compute that $\Lambda_{11} = \Lambda_{33} = \frac{-2\rho^2+1}{4\rho^4-5\rho^2+1}$, $\Lambda_{22} = \frac{\rho-1}{2\rho^2+\rho-1}$, and $\Lambda_{44} = \frac{-\rho-1}{2\rho^2-\rho-1}$

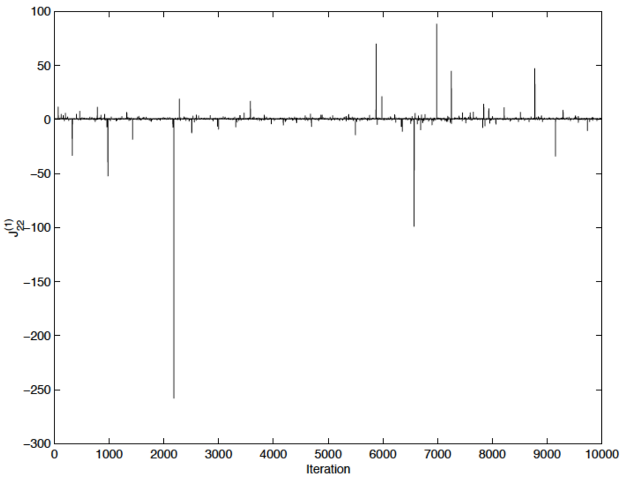
(b) The Gaussian BP update rule for the information matrix is:

$$J_{i\to j} \quad \propto \quad J_{ii} - \sum_{k\in\partial i\setminus j} J_{ik} J^{-1}_{k\to i} J_{ki}$$

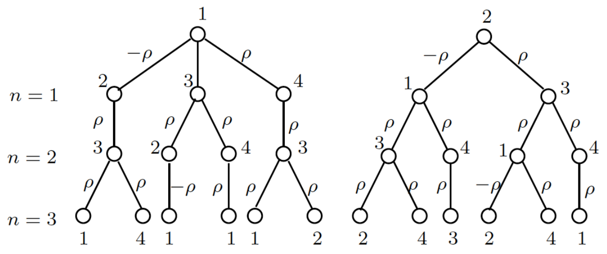
When $\rho = 0.39$, the BP update information matrices converge to the following fixed-point:

$$\begin{aligned}
J_{2\to1} &\simeq 0.7060\\
J_{3\to1} &\simeq 0.5691\\
J_{4\to1} &\simeq 0.7060\\
J_{1\to2} &\simeq 0.5173\\
J_{3\to2} &\simeq 0.5173\\
J_{1\to3} &\simeq 0.5691\\
J_{2\to3} &\simeq 0.7060\\
J_{4\to3} &\simeq 0.7060\\
J_{1\to4} &\simeq 0.5173\\
J_{3\to4} &\simeq 0.5173
\end{aligned}$$

When $\rho = 0.4$, there is no steady state value, and below is shown an example for $J_{2\to1}$.



(c) Computation trees from the perspective of node 1 and node 2 are shown below. Let us consider the



information matrix from the perspective of node 1. After one iteration we have:

$$J_1 = \begin{bmatrix} 1 & -\rho & \rho & \rho \\ -\rho & 1 & 0 & 0 \\ \rho & 0 & 1 & 0 \\ \rho & 0 & 0 & 1 \end{bmatrix}$$

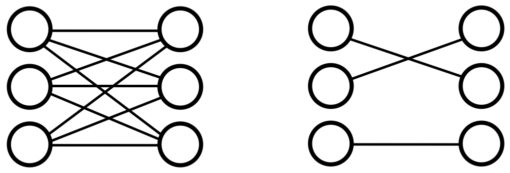
After 2 iterations:

$$J_2 \quad = \quad \begin{bmatrix} 1 & -\rho & \rho & \rho & 0 & 0 & 0 & 0 \\ -\rho & 1 & 0 & 0 & \rho & 0 & 0 & 0 \\ \rho & 0 & 1 & 0 & 0 & \rho & \rho & 0 \\ \rho & 0 & 0 & 1 & 0 & 0 & 0 & \rho \\ 0 & \rho & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \rho & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \rho & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \rho & 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix is not positive definite for $\rho > 0.4760$. After three iterations the following information matrix: Following in this fashion, J_3 after three iterations is not positive definite for $\rho > 0.4473$, and J_4 is not positive definite for $\rho?0.4309$. If we continue like this, after a few more iterations, we will reach a point such that J is not positive definite for $\rho > 0.4$. However, we will never get to a point such that J is not positive definite for $\rho > 0.39$. Since the computation tree implies an invalid information matrix when $\rho > 0.4$, it isnt very surprising that loopy BP behaves so badly in this case.

Problem 3.5 In this problem, we apply inference techniques in graphical models to find *Maximum Weight Matching (MWM)* in a complete bipartite graph. This is one of a few problems where belief propagation converges and is correct on a general graph with loops. Other such examples include Gaussian graphical models studied in class.

Consider an undirected weighted complete bipartite graph $G(X, Y, E)$ where X is a set of n nodes and Y is another set of n nodes: $|X| = |Y| = n$. In a bipartite complete graph all the nodes in X are connected to all the nodes in Y and vice versa, as shown below. Further, each edge in this graph is associated with a real



valued weight $w_{ij} \in \mathbb{R}$. A *matching* in a graph is a subset of edges such that no edges in this matching share a node. A matching is a *perfect matching* if it matches all the nodes in the graph. Let $\pi = (\pi(1), \ldots, \pi(n))$ be a permutation of n nodes. In a bipartite graph, a permutation π defines a perfect matching $\{(i, \pi(i)\}_{i \in \{1,\ldots,n\}}$. From now on, we use a permutation to represent a matching. A *weight* of a (perfect) matching is defined as $W_\pi = \sum_{i=1}^n w_{i,\pi(i)}$. The problem of *maximum weight matching* is to find a matching such that

$$\pi^* \quad = \quad \arg\max_\pi W_\pi \ .$$

We want to solve this maximization by introducing a graphical model with probabilty proportional to the weight of a matching:

$$\mu(\pi) \quad = \quad \frac{1}{Z} e^{CW_\pi} \, \mathbb{I}(\pi \text{ is a perfect matching}) \ ,$$

for some constant C.

(a) The set of matchings can be encoded by a pair of vectors $x \in \{1, \dots, n\}^n$ and $y \in \{1, \dots, n\}^n$, where each node takes an integer value from 1 to n. With these, we can represent the joint distribution as a pari-wise graphical model:

$$\mu(x, y) = \frac{1}{Z} \prod_{(i,j) \in \{1,\dots,n\}^2} \psi_{ij}(x_i, y_j) \prod_{i=1}^{n} e^{w_{i,x_i}} \prod_{i=1}^{n} e^{w_{y_i,i}} ,$$

where $\psi_{ij}(x_i, y_j) = \begin{cases} 0 & x_i = j \text{ and } y_j \neq i \ , \\ 0 & x_i \neq j \text{ and } y_j = i \ , \\ 1 & \text{otherwise} \ . \end{cases}$ Show that for the pairwise graphical model defined above, the joint distribution $\mu(x, y)$ is non-zero if and only if $\pi_x = \{(1, x_1), \dots, (n, x_n)\}$ and $\pi_y = \{(y_1, 1), \dots, (y_n, n)\}$ both are matchings and $\pi_x = \pi_y$. Further, show that when non-zero, the probability is equal to $\frac{1}{Z} e^{2W_{\pi_x}}$.

(b) Let

$$(x^*, y^*) \quad = \quad \arg\max_{x,y} \mu(x, y) \ .$$

Show that $\pi_{x^*} = \pi_{x^*}$ is the maximum weight matching on the given graph G with weights $\{w_{ij}\}$.

(c) Let us denote by l_i the i-th 'left' node in X corresponding to the random variable x_i, and by r_j the j-th 'right' node in Y corresponding to the random variable y_j. We are going to derive *max-product* update rules for this problem. Let $\nu_{l_i \to r_j}(x_i)^{(t)}$ denote the message from a left node l_i to a right node r_j at t-th iteration, which is a vector of size n. Similarly, we let $\nu_{r_j \to l_i}(y_j)^{(t)}$ denote the message from a right node r_j to a leftt node l_i. We initialize all the messages such that

$$\begin{aligned} \nu^{(0)}_{l_i \to r_j}(x_i) &= e^{w_{i,x_i}} \ , \\ \nu^{(0)}_{r_j \to l_i}(y_j) &= e^{w_{y_j,j}} \ . \end{aligned}$$

Write the message update rule for the message $\nu^{(t+1)}_{l_i \to r_j}(x_i)$ and $\nu^{(t+1)}_{r_j \to l_i}(y_j)$ as functions of messages from previous iterations.

Solution 3.5

(a) From the definition of the pair-wise compatibility function, we claim that the probability is non-zero if and only if $\pi_x = \pi_y$. First we prove the 'if' part. Suppose $\pi_x = \pi_y$, that is $y_{x_i} = i$. Then, for a pair (i, j) which is included in the matching π_x, $\psi_i j(x_i, y_j) = 1$ since $x_i = j$ and $y_j = i$. For a pair (i, j) which is not included in the matching π_x, $\psi_i j(x_i, y_j) = 1$ since $x_i \neq j$ and $y_j \neq i$. Hence, if $\pi_x = \pi_y$, then the probability is non-zero.

Now we prove the the 'only if' part. Suppose $\pi_x \neq \pi_y$, that is there exists a pair of nodes (i, j) such that $j = x_i$ but $i \neq y_j$. It follows that $\psi_{ij}(x_i, y_j) = 0$, and this finishes the proof.

When the probability is non-zero, it follows that x and y define matchings π_x and π_y. Then, all the pairwise compatibility functions are one, and we are left to evaluate the singleton compatibility functions. Notice that $\prod_i e^{w_{i,x_i}} = e^{\sum_i w_{i,x_i}} = e^{W_{\pi_x}}$, and similarly $\prod_i e^{w_{y_i,i}} = e^{W_{\pi_y}}$. This proveos that $\mu(x) = \frac{1}{Z} e^{2W_{\pi_x}}$.

(b) Since the probability is a monotone increasing function of the weight, solving for the maximum probability realization recovers the maximum weight matching.

(c) From the max-product algorithm definition, and noting that this is a complete graph,

$$\begin{aligned}
\nu^{(t+1)}_{l_i\to r_j}(x_i) &= e^{w_{i,x_i}} \prod_{k\neq j} \max_{y_k\in\{1,\dots,n\}} \psi_{i,k}(x_i,y_k)\nu^{(t)}_{r_k\to l_i}(y_k)\,,\\
\nu^{(t+1)}_{r_j\to l_i}(y_j) &= e^{w_{y_j,j}} \prod_{k\neq i} \max_{x_k\in\{1,\dots,n\}} \psi_{k,j}(x_k,y_j)\nu^{(t+1)}_{l_k\to r_j}(x_k)\,.
\end{aligned}$$

We can further simplify this exploiting the structure of $\psi_{ij}()$'s.

$$\nu^{(t+1)}_{l_i\to r_j}(x_i) = \begin{cases} e^{w_{i,x_i}} \prod_{k\neq j} \max_{y_k\neq i} \nu^{(t)}_{r_k\to l_i}(y_k) & \text{if } x_i = j\,,\\ e^{w_{i,x_i}} \nu^{(t)}_{r_{x_i}\to l_i}(y_{x_i}=i) \prod_{k\notin\{j,x_i\}} \max_{y_k\neq i} \psi_{i,k}(x_i,y_k)\,\nu^{(t)}_{r_k\to l_i}(y_k) & \text{otherwise}\,,\end{cases}$$

Now, let's further simplify this formula. Notice that when $x_i \neq j$, $\max_{y_k\neq i}\psi_{i,k}(x_i,y_k)\,\nu^{(t)}_{r_k\to l_i}(y_k) = \max_{y_k\neq i}\nu^{(t)}_{r_k\to l_i}(y_k)$, since $\psi_{i,k}(x_i\neq k, y_k\neq i)=1$.

$$\nu^{(t+1)}_{l_i\to r_j}(x_i) = \begin{cases} e^{w_{i,x_i}} \prod_{k\neq j} \max_{y_k\neq i} \nu^{(t)}_{r_k\to l_i}(y_k) & \text{if } x_i = j\,,\\ e^{w_{i,x_i}} \nu^{(t)}_{r_{x_i}\to l_i}(y_{x_i}=i) \prod_{k\notin\{j,x_i\}} \max_{y_k\neq i} \nu^{(t)}_{r_k\to l_i}(y_k) & \text{otherwise}\,,\end{cases}$$

Then, we can scale the messages by

$$\frac{1}{\left(e^{w_{i,x_i}}\prod_{k\neq j}\max_{y_k\neq i}\nu^{(t)}_{r_k\to l_i}(y_k)\right)}$$

to get

$$\nu^{(t+1)}_{l_i\to r_j}(x_i) = \begin{cases} 1 & \text{if } x_i = j\,,\\ \dfrac{\nu^{(t)}_{r_{x_i}\to l_i}(y_{x_i}=i)}{\max_{y_{x_i}\neq i}\nu^{(t)}_{r_{x_i}\to l_i}(y_{x_i})} & \text{otherwise}\,.\end{cases}$$

Problem 3.6 As mentioned in class, Gaussian BP allows to compute the minimum of a quadratic function

$$\widehat{x} = \arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle\right\}. \tag{2}$$

for $Q\in\mathbb{R}^{n\times n}$ positive definite, where $\langle a,b\rangle = a^Tb$ indicates the standard inner product of two vectors. In this homework we will consider a case in which Q is not positive definite, but is symmetric and has full rank. in this case we can still define

$$\widehat{x} = -Q^{-1}b\,. \tag{3}$$

which is a stationary point (a saddle point) of the above quadratic function. The BP update equations are exactly the same as for the minimization problem with a positive definite Q. We claim that, when BP converges, it still computes the correct solution $\widehat{x}$.

We consider a specific model. An unknown signal $s_0\in\mathbb{R}^n$ is observed in Gaussian noise

$$y = As_0 + w_0\,. \tag{4}$$

Here $y \in \mathbb{R}^m$ is a vector of observations, $A \in \mathbb{R}^{m\times n}$ is a measurement matrix, and $w_0 \in \mathbb{R}^m$ is a vector of Gaussian noise, with i.i.d. entries $w_{0,i} \sim \mathcal{N}(0, \sigma^2)$. We are given y and A, and would like to reconstruct the unknown vector s_0, and hence w_0.

A popular method consists in solving the following quadratic programming problem (known as *ridge regression*):

$$\widehat{s} = \arg\min_{s\in\mathbb{R}^n} \left\{ \frac{1}{2}\|y - As\|_2^2 + \frac{1}{2}\lambda\|s\|_2^2 \right\}. \tag{5}$$

We will do something equivalent. For $x \in \mathbb{R}^{m+n}$, $x = (z, s)$, $z \in \mathbb{R}^m$, $s \in \mathbb{R}^n$, we define a cost function

$$\mathcal{C}_{A,y}(x = (z, s)) \;=\; -\frac{1}{2}\|z\|_2^2 + \frac{1}{2}\lambda\|s\|_2^2 + \langle z, y - As\rangle\,. \tag{6}$$

We will look for the stationary point of $\mathcal{C}_{A,y}$.

(a) Show that the cost function $\mathcal{C}_{A,y}(x)$ can be written in the form

$$\mathcal{C}_{A,y}(x) = \frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle\,. \tag{7}$$

Write explicitly the form of the matrix $Q \in \mathbb{R}^{(m+n)\times(m+n)}$ and the vector $b \in \mathbb{R}^{m+n}$.

(b) Let $\widehat{x} = (\widehat{z}, \widehat{s})$ be the stationary point of $\mathcal{C}_{A,y}(z, s)$. Assuming it is unique, show that $\widehat{s}$ does coincide with the ridge estimator (5).

(c) Write the update rule for the BP algorithm (equivalent to the sum-product algorithm) to compute the stationary point $\widehat{x} = (\widehat{z}, \widehat{s})$ of $\mathcal{C}_{A,y}(x)$. [hint: use the same ideas from the Gaussian belief propagation for positive definite Q.]

(d) Prove the above claim that, if BP converges, then it computes $\widehat{x}$, cf. Eq. (3) even if Q is not positive definite.

Solution 3.6

(a) The cost function is

$$\begin{aligned}\mathcal{C}_{A,y}(x = (z, s)) &= -\frac{1}{2}\begin{bmatrix} z & s\end{bmatrix}\begin{bmatrix} I & A \\ A^T & -\lambda I\end{bmatrix}\begin{bmatrix} z \\ s\end{bmatrix} + \begin{bmatrix} y & 0\end{bmatrix}\begin{bmatrix} z \\ s\end{bmatrix} \\ &= \frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle\,,\end{aligned}$$

where $Q = -\begin{bmatrix} I & A \\ A^T & -\lambda I\end{bmatrix}$ and $b = \begin{bmatrix} y \\ 0\end{bmatrix}$.

(b) The stationary point of the quadratic form is given by

$$\nabla\mathcal{C}_{A,y}(x) = Qx + b$$

setting the gradient to zero, we get $\widehat{x} = -Q^{-1}b$. In terms of the original values we get,

$$\begin{aligned}\begin{bmatrix} \widehat{z} \\ \widehat{s}\end{bmatrix} &= \begin{bmatrix} I & A \\ A^T & -\lambda I\end{bmatrix}^{-1}\begin{bmatrix} y & 0\end{bmatrix} \\ &= \begin{bmatrix} (I + \frac{1}{\lambda}AA^T)^{-1} & \frac{1}{\lambda}(I + \frac{1}{\lambda}AA^T)^{-1}A \\ \frac{1}{\lambda}A^T(I + \frac{1}{\lambda}AA^T)^{-1} & -\frac{1}{\lambda}I + \frac{1}{\lambda^2}A^T(I + \frac{1}{\lambda}AA^T)^{-1}A\end{bmatrix}\begin{bmatrix} y & 0\end{bmatrix}\,.\end{aligned}$$

hence,

$$\widehat{s} = A^T(AA^T + \lambda I)^{-1}y$$

equivalently, one can also write

$$\widehat{s} = (A^TA + \lambda I)^{-1}A^Ty$$

both are valid solutions.

(*c*) the BP updates are

$$\begin{aligned} h_{i\to j}^{(t+1)} &= -b_i - \sum_{k\in\partial i\setminus j} \frac{Q_{ik}}{J_{k\to i}^{(t)}} h_{k\to i}^{(t)} \\ J_{i\to j}^{(t+1)} &= Q_{ii} - \sum_{k\in\partial i\setminus j} \frac{Q_{ij}^2}{J_{k\to i}^{(t)}} \end{aligned}$$

It is okay to drop the superscript for the time t, and also change the signs of the messages, as long as they compute the correct marginal in the end. For example, the following is also a valid BP update.

$$\begin{aligned} h_{i\to j}^{(t+1)} &= -b_i + \sum_{k\in\partial i\setminus j} \frac{Q_{ik}}{J_{k\to i}^{(t)}} h_{k\to i}^{(t)} \\ J_{i\to j}^{(t+1)} &= -Q_{ii} - \sum_{k\in\partial i\setminus j} \frac{Q_{ij}^2}{J_{k\to i}^{(t)}} \end{aligned}$$

(*d*) Consider the belief propagation and the corresponding computation tree $T_G(i;\ell)$. We know that the correct solution is

$$\widehat{x} = -Q^{-1}b$$

We will show that assuming the BP converges, the BP estimates $\lim_{\ell\to\infty}\widehat{x}^{(\ell)} = \widehat{x}^{(\infty)}$ is the same as $\widehat{x}$ above.

The only difference in the proof for Q that is not necessarily positive definite is that we don't know if the BP computes the correct solution (the mean) on a tree. Once we prove this claim, then we are done, since all the proof techniques follow from the lecture notes.

Problem 3.7 [Density Evolution] In this problem we consider using Low-Density Parity Check (LDPC) codes to encode bits to be sent over a noisy channel.

Encoding. LDPC codes are defined by a factor graph model over a bipartite graph $G(V, F, E)$, where V is the set of variable nodes, each representing the bit to be transmitted, and F is a set of factor nodes describing the code and E is a west of edges between a bit-node and a factor node. The total number of variable nodes in the graph define the length of the code (also known as the block length), which we denote by $n \triangleq |V|$.

We consider binary variables $x_i \in \{-1, +1\}$ for $i \in V$, and all codewords that are transmitted satisfy

$$\prod_{i\in\partial a} x_i = +1 ,$$

which means that there are even number of -1's in the neighborhood of any factor node.

Channel. We consider a Binary Symmetric Channel, known as BSC(ε), where one bit is transmitted over the channel at each discrete time step, and each transmitted bit is independently flipped with probability ε. Precisely, let $x_i \in \{+1, -1\}$ be a transmitted bit and $y_i \in \{+1, -1\}$ be the received bit (at time i), then

$$\begin{aligned}
\mathbb{P}(y_i = +1 | x_i = +1) &= 1 - \varepsilon \ , \\
\mathbb{P}(y_i = -1 | x_i = +1) &= \varepsilon \ , \\
\mathbb{P}(y_i = -1 | x_i = -1) &= 1 - \varepsilon \ , \\
\mathbb{P}(y_i = +1 | x_i = -1) &= \varepsilon \ .
\end{aligned}$$

The conditional probability distribution over $x_1^n = [x_1, \ldots, x_n]$ given the observed received bits $y_1^n = [y_1, \ldots, y_n]$ is

$$\mu(x_1^n \,|\, y_1^n) \quad = \quad \frac{1}{Z} \prod_{i \in V} \psi_i(x_i, y_i) \prod_{a \in F} \mathbb{I}(\otimes x_{\partial a} = +1) \ ,$$

where $\psi_i(x_i, y_i) = \mathbb{P}(y_i | x_i)$ and $\otimes$ indicates product of binary numbers such that if $x_{\partial a} = \{x_1, x_2, x_3\}$ then $\otimes x_{\partial a} = x_1 \times x_2 \times x_3$ (to be precise we need to take $\psi_i(x_i | y_i) = \mathbb{P}(x_i | y_i)$, but this gives the exactly same conditional distribution as above since any normalization with respect to y_i's are absorbed in the partition function Z). This is naturally a graphical model on a factor graph $G(V, F, E)$ defined by the LDPC code.

(a) Write down the belief propagation updates (also known as the (parallel) sum-product algorithm) for this factor graph model for the messages $\{\nu_{i \to a}^{(t)}(\cdot)\}_{(i,a) \in E}$ and $\{\tilde{\nu}_{a \to i}^{(t)}(\cdot)\}_{(i,a) \in E}$.

(b) What is the computational complexity (how many operations are required in terms of the degrees of the variable and factor nodes) for updating one message $\nu_{i \to a}(\cdot)$ and one message $\tilde{\nu}_{a \to i}(\cdot)$ respectively? Explain how one can improve the computational complexity, to compute the message $\tilde{\nu}_{a \to i}^{(t)}(\cdot)$ exactly in runtime $O(d_a)$, where d_a is the degree of the factor node a.

(c) Now, we consider a different message passing algorithm introduced by Robert Gallager in 1963. The following update rule is a message passing algorithm known as the **Gallager A algorithm**. Similar to the belief propagation for BEC channels we studied in class, this algorithm also sends discrete messages (as opposed to real-valued messages in part (a)). Both $\nu_{i \to a}^{(t)}$'s and $\tilde{\nu}_{a \to i}^{(t)}$'s are binary, i.e. in $\{+1, -1\}$.

$$\begin{aligned}
\nu_{i \to a}^{(t+1)} &= \begin{cases} +1 & \text{if } \tilde{\nu}_{b \to i}^{(t)} = +1 \text{ for all } b \in \partial i \setminus a \ , \\ -1 & \text{if } \tilde{\nu}_{b \to i}^{(t)} = -1 \text{ for all } b \in \partial i \setminus a \ \ , \\ y_i & \text{otherwise} \ , \end{cases} \\
\tilde{\nu}_{a \to i}^{(t)} &= \prod_{j \in \partial a \setminus i} \nu_{j \to a}^{(t)} \ .
\end{aligned}$$

The interpretation of this update rule is that $\nu_{i \to a}$ messages trust the received bit y_i unless all of the incoming messages disagree with y_i, and $\tilde{\nu}_{a \to i}$ messages make sure that the consistency with respect to $\mathbb{I}(\otimes x_{\partial a})$ is satisfied. In this algorithm, the messages take values in $\{+1, -1\}$ and are the estimated values of x_i's, as opposed to the distribution over those values as in belief propagation.

We assume that random (ℓ, r)-regular bipartite graph is used to generate the LDPC code. In the resulting random graph, all variable nodes have degree ℓ and all factor nodes have degree r. Among all such graphs, a random graph is selected uniformly at random.

Define $W^{(t)}$ to be the (empirical) distribution of the messages $\{\nu_{i \to a}^{(t)}\}_{(i,a) \in E}$ and $Z^{(t)}$ to be the (empirical) distribution of the messages $\{\tilde{\nu}_{a \to i}^{(t)}\}_{(i,a) \in E}$. We assume the messages are initialized in such way that $\nu_{i \to a}^{(0)} = y_i$ for all $i \in V$. We also assume, without loss of generality, that all $+1$ messages were sent, i.e. $x_i = +1$ for all i. Then, let $w^{(t)} = \mathbb{P}(W^{(t)} = -1)$ be the probability that a message $\nu^{(t)}_{i\to a}$ is -1 for a randomly chosen edge (i, a), and let $z^{(t)} = \mathbb{P}(Z^{(t)} = -1)$ be the probability that a message $\tilde{\nu}^{(t)}_{a\to i}$ is -1 for a randomly chosen edge (i, a).

Write the **density evolution** *equations for $w^{(t)}$ and $z^{(t)}$, describing how the random distribution of the messages $w^{(t)}$ and $z^{(t)}$ evolve.* [We are looking for a clean answer. Specifically, the number of operations required to compute $z^{(t)}$ from $w^{(t)}$ should be $O(1)$. The same technique that reduced computation in part (b) should be helpful.]

(d) Write the density evolution equation for a single scalar variable $w^{(t)}$, by substituting $z^{(t)}$. This gives a fixed point equation in the form of $w^{(t)} = F(w^{(t-1)})$ for some F. Plot (using MATLAB to your favorite numerical analysis tool) the function $y = F(x)$ and the identify function $y = x$, for $\ell = 3$ and $r = 4$, and for two values of $\varepsilon = 0.05$ and $\varepsilon = 0.1$ Explain the figure in terms of the error probability of the (3,4)-code on those two BSC(ε)'s.

Solution 3.7

(a)

$$\begin{aligned}
\nu^{(t+1)}_{i\to a}(x_i) &= \mathbb{P}(y_i|x_i) \prod_{b\in\partial i\setminus a} \tilde{\nu}^{(t)}_{b\to i}(x_i) \\
\tilde{\nu}^{(t)}_{a\to i}(x_i) &= \sum_{x_{\partial a\setminus i}} \mathbb{I}(\otimes x_{\partial a} = +1) \prod_{j\in\partial a\setminus i} \nu^{(t)}_{j\to a}(x_j)
\end{aligned}$$

(b) Naive implementation of the above update rule take $O(d_i)$ operations to update $\nu^{(t+1)}_{i\to a}(\cdot)$ and $O(2^{d_a} d_a)$ operations to update $\tilde{\nu}^{(t)}_{a\to i}(\cdot)$, where d_i and d_a are the degrees of the variable node x_i and the factor node a respectively. However, we can use the Fourier transform of the binary variables in order to simplify the computation of updating $\tilde{\nu}^{(t)}_{a\to i}(\cdot)$. Note that we can write the update rule as

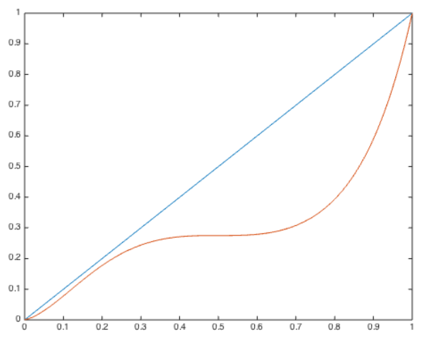
$$\begin{aligned}
\tilde{\nu}^{(t)}_{a\to i}(x_i = +1) &= \sum_{x_{\partial a\setminus i}} \mathbb{I}(\otimes x_{\partial a} = +1) \prod_{j\in\partial a\setminus i} \nu^{(t)}_{j\to a}(x_j) \\
&= \frac{1}{2}\Big(\prod_{j\in\partial a\setminus i} \big(\nu^{(t)}_{j\to a}(+1) + \nu^{(t)}_{j\to a}(-1)\big) + \prod_{j\in\partial a\setminus i} \big(\nu^{(t)}_{j\to a}(+1) - \nu^{(t)}_{j\to a}(-1)\big)\Big) .
\end{aligned}$$

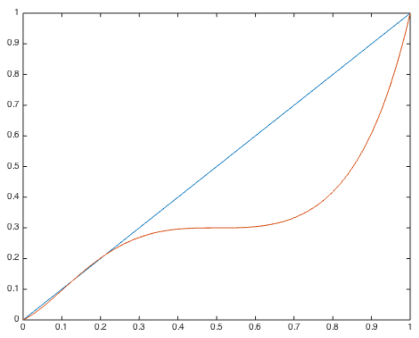
This requires $O(d_a)$ operations to compute, and similar computation of $\tilde{\nu}^{(t)}_{a\to i}(x_i = -1)$ exists.

(c) Given the initialization, we have $w^{(0)} = \varepsilon$. And the update rules are

$$\begin{aligned}
w^{(t+1)} &= (z^{(t)})^{\ell-1} + (1 - (z^{(t)})^{\ell-1} - (1 - z^{(t)})^{\ell-1})\varepsilon , \\
z^{(t)} &= \frac{1}{2}\Big(1 - (1 - 2w^{(t)})^{r-1}\Big) .
\end{aligned}$$

(d) When $\varepsilon = 0.05$, the error probability goes to zero, whereas when $\varepsilon = 0.1$, the error proabability does not decay to zero.


when $\varepsilon = 0.05$


when $\varepsilon = 0.1$