Assignment 6

Callum Lau, 19102521

Approximate Inference

January 15, 2020

1 EP for sign constraints

1.1 Part a

$$y_1 \sim \mathcal{N}(0, \sigma^2) \tag{1}$$
$$y_i|y_{i-1} \sim \mathcal{N}(y_{i-1}, \sigma^2) \qquad \text{for } i = 2, 3, \ldots \tag{2}$$
$$x_i|y_i \sim \mathcal{N}(y_i, \tau^2) \qquad \text{for } i = 1, 2, \ldots \tag{3}$$

If we define $p(y_1|y_0) = p(y_1)$, then we can write the joint distribution given by the normalised factors with the distribution of equations (1) - (3) as:

$$P(\mathcal{X}, \mathcal{Y}) = \prod_i p(y_i|y_{i-1})p(x_i|y_i)f_i(x_i) \tag{4}$$

Now $p(y_i|y_{i-1})$ and $p(x_i|y_i)$ are Gaussian so we only need to approximate $f_i(x_i)$ with a Gaussian factor $\tilde{f}_i(x_i)$. The approximation takes the form:

$$\tilde{f}_i(x_i) = \underset{\tilde{f}}{\operatorname{argmin}}\, \mathbf{KL}[f_i(x_i)q_{\neg i}(x_i)||\tilde{f}(x_i)q_{\neg i}(x_i)] \tag{5}$$

Firstly, we find the cavity distribution $q_{\neg i}(x_i)$:

$$\begin{aligned}
q_{\neg i}(\mathcal{X}) &= \frac{\prod_j p(y_j|y_{j-1})p(x_j|y_j)\tilde{f}_j(x_j)}{\tilde{f}_i(x_i)} \\
&= \left\{\prod_{j\neq i} p(y_j|y_{j-1})p(x_j|y_j)\tilde{f}_j(x_j)\right\} p(y_i|y_{i-1})p(x_i|y_i) \\
\implies q_{\neg i}(x_i) &= \int\int \left\{\prod_{j\neq i} p(y_j|y_{j-1})p(x_j|y_j)\tilde{f}_j(x_j)\right\} p(y_i|y_{i-1})p(x_i|y_i) d\mathcal{X}_{\neg i} d\mathcal{Y} \\
&= \int\int \left\{\prod_{j\neq i} p(y_j|y_{j-1})p(x_j|y_j)\tilde{f}_j(x_j)\right\} d\mathcal{X}_{\neg i} d\mathcal{Y}_{\neg\{i-1,i\}} p(y_i|y_{i-1})p(x_i|y_i) dy_{i-1} dy_i
\end{aligned}$$

$$= \int \Big(\int \prod_{j<i} p(y_j|y_{j-1}) p(x_j|y_j) \tilde{f}_j(x_j) d\mathcal{X}_{1:j-1} d\mathcal{Y}_{1:j-2}$$
$$\times \int \prod_{j>i} p(y_j|y_{j-1}) p(x_j|y_j) \tilde{f}_j(x_j) d\mathcal{X}_{j+1:n} d\mathcal{Y}_{j+1:n} \Big) p(y_i|y_{i-1}) p(x_i|y_i) dy_{i-1} dy_i$$
$$= \int \int \alpha_{i-1}(y_{i-1}) \beta_i(y_i) p(y_i|y_{i-1}) p(x_i|y_i) dy_{i-1} dy_i \tag{6}$$

Where we have messages $\alpha_{i-1}(y_{i-1})$ and $\beta_i(y_i)$ which take the form:

$$\alpha_{i-1}(y_{i-1}) = \int \int \prod_{j<i} p(y_j|y_{j-1}) p(x_j|y_j) \tilde{f}_j(x_j) d\mathcal{X}_{1:j-1} d\mathcal{Y}_{1:j-2} \tag{7}$$
$$\beta_i(y_i) = \int \int \prod_{j>i} p(y_j|y_{j-1}) p(x_j|y_j) \tilde{f}_j(x_j) d\mathcal{X}_{j+1:n} d\mathcal{Y}_{j+1:n} \tag{8}$$
$$\alpha_{i-1}(y_{i-1}) \cdot \beta_i(y_i) = \int \int \prod_{j \neq i} p(y_j|y_{j-1}) p(x_j|y_j) \tilde{f}_j(x_j) d\mathcal{X}_{\neg i} d\mathcal{Y}_{\neg(i,i-1)} \tag{9}$$

Since these messages are just products of Gaussians they can be computed exactly. Denote their mean and variances as:

$$\alpha_{i-1}(y_{i-1}) = \mathcal{N}(y_{i-1}; \mu_\alpha, \sigma_\alpha^2) \tag{10}$$
$$\beta_i(y_i) = \mathcal{N}(y_i; \mu_\beta, \sigma_\beta^2) \tag{11}$$

Therefore $q_{\neg i}(x_i)$ becomes:

$$q_{\neg i}(x_i) = \int \int \mathcal{N}(y_{i-1}; \mu_\alpha, \sigma_\alpha^2) \mathcal{N}(y_i; \mu_\beta, \sigma_\beta^2) \mathcal{N}(y_i; y_{i-1}, \sigma^2) \mathcal{N}(x_i; y_i, \tau^2) dy_{i-1} dy_i \tag{12}$$

Firstly, considering only terms with y_{i-1}:

$$\int \mathcal{N}(y_{i-1}; \mu_\alpha, \sigma_\alpha^2) \mathcal{N}(y_i; y_{i-1}, \sigma^2) dy_{i-1} = \mathcal{N}(y_i; \mu_\alpha, \sigma^2 + \sigma_\alpha^2)$$

Then

$$q_{\neg i}(x_i) = \int \mathcal{N}(y_i; \mu_\alpha, \sigma^2 + \sigma_\alpha^2) \mathcal{N}(y_i; \mu_\beta, \sigma_\beta^2) \mathcal{N}(x_i; y_i, \tau^2) dy_i$$

Now we note that:

$$\mathcal{N}(y_i; \mu_\alpha, \sigma^2 + \sigma_\alpha^2) \mathcal{N}(y_i; \mu_\beta, \sigma_\beta^2) = \mathcal{N}(y_i; , \hat{\mu}, \hat{\sigma}^2)$$

Where

$$\hat{\sigma}^2 = ((\sigma^2 + \sigma_\alpha^2)^{-1} + (\sigma_\beta^2)^{-1})^{-1}$$
$$\hat{\mu} = \hat{\sigma}^2 (\frac{\mu_\alpha}{\sigma^2 + \sigma_\alpha^2} + \frac{\mu_\beta}{\sigma_\beta^2})$$

Finally, we have

$$q_{\neg i}(x_i) = \int \mathcal{N}(y_i; \hat{\mu}, \hat{\sigma}^2)\mathcal{N}(x_i; y_i, \tau^2)dy_i$$

Which evaluates to

$$q_{\neg i}(x_i) = \mathcal{N}(x_i; \mu_q, \sigma_q^2) \tag{13}$$

$$\mu_q = ((\sigma^2 + \sigma_\alpha^2)^{-1} + (\sigma_\beta^2)^{-1})^{-1}(\frac{\mu_\alpha}{\sigma^2 + \sigma_\alpha^2} + \frac{\mu_\beta}{\sigma_\beta^2}) \tag{14}$$

$$\sigma_q^2 = ((\sigma^2 + \sigma_\alpha^2)^{-1} + (\sigma_\beta^2)^{-1})^{-1} + \tau^2 \tag{15}$$

Now we are able to evaluate equation (5): $\tilde{f}_i(x_i) = \text{argmin}_{\tilde{f}} \mathbf{KL}[f_i(x_i)q_{\neg i}(x_i)||\tilde{f}(x_i)q_{\neg i}(x_i)]$. Note this is minimised via moment matching - i.e. we want the to set the expected moments of the proposed distribution to the target distribution. We use the `proj` operator to denote the Gaussian parametrised by the moments of the given distribution. Then

$$\begin{aligned}
\tilde{f}(x_i)q_{\neg i}(x_i) &= \texttt{proj}[f_i(x_i)q_{\neg i}(x_i)] \\
&= \texttt{proj}[\mathbb{1}_{\{s_i x_i > 0\}}\mathcal{N}(x_i; \mu_q, \sigma_q^2)] \\
&= \mathcal{N}(x_i; \mathbb{E}(\mu_q, \sigma_q^2), \mathbb{V}(\mu_q, \sigma_q^2)) \qquad \text{(formula for truncated Gaussian)} \\
\implies \tilde{f}(x_i) &= \frac{\mathcal{N}(x_i; \mathbb{E}(\mu_q, \sigma_q^2), \mathbb{V}(\mu_q, \sigma_q^2))}{\mathcal{N}(x_i; \mu_q, \sigma_q^2)}
\end{aligned}$$

Therefore we get the final update equation:

$$\tilde{f}(x_i) = \mathcal{N}(x_i; \mu_f, \sigma_f^2) \tag{16}$$

$$\text{where } \sigma_f^2 = (\mathbb{V}(\mu_q, \sigma_q^2)^{-1} - (\sigma_q^2)^{-1})^{-1} \tag{17}$$

$$\mu_f = \sigma_f^2 \left(\frac{\mathbb{E}(\mu_q, \sigma_q^2)}{\mathbb{V}(\mu_q, \sigma_q^2)} - \frac{\mu_q}{\sigma_q^2} \right) \tag{18}$$

We can therefore approximate the resulting posterior by iterating through the factors using these updates until (un-guarenteed) convergence. The resulting marginal of $P(\mathcal{Y})$ can then be found via marginalisation:

$$P(\mathcal{Y}) = \int P(\mathcal{X}, \mathcal{Y})dx_{1:n} = \prod_i p(y_i|y_{i-1}) \int p(x_i|y_i)\tilde{f}_i(x_i)dx_i$$

1.2 Part b

In this version we first compute the sign probabilities:

$$g_i(y_i) = P(\texttt{sign}(x_i) = s_i|y_i) \tag{19}$$

$$\begin{aligned}
&= \int_{-\infty}^{\infty} \mathbb{1}_{\{x_i s_i > 0\}}\mathcal{N}(x_i; y_i, \tau^2)dx_i \\
&= \Phi_{s_i, \tau^2}(y_i) \qquad (20)
\end{aligned}$$

With the joint given by

$$P(\mathcal{Y}) = \prod_i p(y_i|y_{i-1})g_i(y_i) \tag{21}$$

where again we have defined $p(y_1|y_0) = p(y_1)$. We approximate the non-gaussian $g_i(y_i)$ terms with the gaussian $\tilde{g}_i(y_i)$, where using EP the updates are given by

$$\tilde{g}_i(y_i) = \underset{\tilde{g}}{\operatorname{argmin}}\, \mathbf{KL}[g_i(y_i)q_{\neg i}(y_i)||\tilde{g}(y_i)q_{\neg i}(y_i)] \tag{22}$$

To evaluate this expression we first find the cavity distribution $q_{\neg i}(y_i)$:

$$\begin{aligned}
q_{\neg i}(\mathcal{Y}) &= \frac{\prod_j p(y_j|y_{j-1})\tilde{g}_j(y_j)}{\tilde{g}_i(y_i)} = p(y_i|y_{i-1}) \prod_{j\neq i} p(y_j|y_{j-1})\tilde{g}_j(y_j) \\
\implies q_{\neg i}(y_i) &= \int p(y_i|y_{i-1}) \prod_{j\neq i} p(y_j|y_{j-1})\tilde{g}_j(y_j) d\mathcal{Y}_{\neg i} \\
&= \int p(y_i|y_{i-1}) \prod_{j<i} p(y_j|y_{j-1})\tilde{g}_j(y_j) d\mathcal{Y}_{1:i-1} \int \prod_{j>i} p(y_j|y_{j-1})\tilde{g}_j(y_j) d\mathcal{Y}_{i+1:n} \\
&= \int p(y_i|y_{i-1}) A_{i-1}(y_{i-1}) dy_{i-1} B_i(y_i)
\end{aligned} \tag{23}$$

where

$$A_{i-1}(y_{i-1}) = \int \prod_{j<i} p(y_j|y_{j-1})\tilde{g}_j(y_j) d\mathcal{Y}_{1:i-2} = \mathcal{N}(y_{i-1}; \mu_A, \sigma_A^2) \tag{24}$$

$$B_i(y_i) = \int \prod_{j>i} p(y_j|y_{j-1})\tilde{g}_j(y_j) d\mathcal{Y}_{i+1:n} = \mathcal{N}(y_i; \mu_B, \sigma_B^2) \tag{25}$$

$$A_{i-1}(y_{i-1}) \cdot B_i(y_i) = \int \prod_{j\neq i} p(y_j|y_{j-1})\tilde{g}_i(y_i) d\mathcal{Y}_{\neg(i,i-1)} \tag{26}$$

Therefore we get that

$$\begin{aligned}
\int p(y_i|y_{i-1}) A_{i-1}(y_{i-1}) dy_{i-1} &= \int \mathcal{N}(y_i; y_{i-1}, \sigma^2)\mathcal{N}(y_{i-1}; \mu_A, \sigma_A^2) dy_{i-1} \\
&= \mathcal{N}(y_i; \mu_A, \sigma^2 + \sigma_A^2)
\end{aligned}$$

Hence

$$\begin{aligned}
q_{\neg i}(y_i) &= \mathcal{N}(y_i; \mu_A, \sigma^2 + \sigma_A^2)\mathcal{N}(y_i; \mu_B, \sigma_B^2) \\
\implies q_{\neg i}(y_i) &= \mathcal{N}(y_i; \mu_q, \sigma_q^2) && (27) \\
\text{where } \sigma_q^2 &= ((\sigma^2 + \sigma_A^2)^{-1} + (\sigma_B^2)^{-1})^{-1} && (28) \\
\mu_q &= \sigma_q^2 \left(\frac{\mu_A}{\sigma^2 + \sigma_A^2} + \frac{\mu_B}{\sigma_B^2} \right) && (29)
\end{aligned}$$

Now we are set to evaluate equation (22): $\text{argmin}_{\tilde{g}} \mathbf{KL}[g_i(y_i)q_{\neg i}(y_i)||\tilde{g}(y_i)q_{\neg i}(y_i)]$ as we did in (a) via moment matching:

$$\tilde{g}_i(y_i) = \frac{\texttt{proj}[g_i(y_i)q_{\neg i}(y_i)]}{q_{\neg i}(y_i)} = \frac{\texttt{proj}[\Phi_{s_i,\tau^2}(y_i)\mathcal{N}(y_i;\mu_q,\sigma_q^2)]}{\mathcal{N}(y_i;\mu_q,\sigma_q^2)} \tag{30}$$

Now we wish to show that the fixed points of (a) and (b) are the same. Note that we can write:

$$g_i(y_i) = \int f_i(x_i)p(x_i|y_i)dx_i \tag{31}$$

And therefore the fixed points are the same if after an update we have:

$$\tilde{g}_i(y_i) = \int \tilde{f}_i(x_i)p(x_i|y_i)dx_i \tag{32}$$

Assuming this is true before an iteration, and going back to equation (9):

$$\begin{aligned}
\alpha_{i-1}(y_{i-1}) \cdot \beta_i(y_i) &= \int\int \prod_{j\neq i} p(y_j|y_{j-1})p(x_j|y_j)\tilde{f}_j(x_j)d\mathcal{X}_{\neg i}d\mathcal{Y}_{\neg(i,i-1)} \qquad \text{eqn (9)} \\
&= \int \prod_{j\neq i} p(y_j|y_{j-1}) \left(\int p(x_j|y_j)\tilde{f}_j(x_j)d\mathcal{X}_{\neg i}\right) d\mathcal{Y}_{\neg(i,i-1)} \\
&= \int \prod_{j\neq i} p(y_j|y_{j-1})\tilde{g}_i(y_i)d\mathcal{Y}_{\neg(i,i-1)}
\end{aligned}$$

substituting equation (26) gives

$$\boxed{\alpha_{i-1}(y_{i-1}) \cdot \beta_i(y_i) = A_{i-1}(y_{i-1}) \cdot B_i(y_i)} \tag{33}$$

Equivalently (and for future reference) this means:

$$\begin{aligned}
q_{\neg i}(x_i) &= \int\int \alpha_{i-1}\beta_i(y_i)p(y_i|y_{i-1})p(x_i|y_i)dy_{i-1}dy_i \\
&= \int\int A_{i-1}(y_{i-1})B_i(y_i)p(y_i|y_{i-1})p(x_i|y_i)dy_{i-1}dy_i \qquad (A_{i-1} = \alpha_{i-1}, B_i = \beta_i) \\
&= \int \left(\int A_{i-1}(y_{i-1}B_i(y_i)p(y_i|y_{i-1})dy_{i-1}\right) p(x_i|y_i)dy_i
\end{aligned}$$

substituting equation (23):

$$\boxed{q_{\neg i}(x_i) = \int q_{\neg i}(y_i)p(x_i|y_i)dy_i} \tag{34}$$

Therefore the messages are the same. We now return to the KL divergence equations (5) and (22):

$$\tilde{f}_i^{\text{new}}(x_i) = \underset{\tilde{f}}{\text{argmin}}\, \mathbf{KL}[f_i(x_i)q_{\neg i}(x_i)||\tilde{f}(x_i)q_{\neg i}(x_i)]$$

$$\tilde{g}_i^{\text{new}}(y_i) = \underset{\tilde{g}}{\text{argmin}}\, \mathbf{KL}[g_i(y_i)q_{\neg i}(y_i)||\tilde{g}(y_i)q_{\neg i}(y_i)]$$

In order to show the fixed points are the same we now show the Zeroth moments Z_f, Z_g of LHS terms of the KL divergences are the same:

$$Z_f \overset{def}{=} \int q_{\neg i}(x_i)f_i(x_i)dx_i \overset{(34)}{=} \int \left(\int q_{\neg i}(y_i)p(x_i|y_i)dy_i \right) f_i(x_i)dx_i$$

changing the order of integration:

$$\begin{aligned} &= \int q_{\neg i}(y_i) \left(\int p(x_i|y_i)f_i(x_i)dx_i \right) dy_i \\ &\overset{(31)}{=} \int q_{\neg i}(y_i)g_i(y_i)dy_i \overset{def}{=} Z_g \end{aligned} \tag{35}$$

Hence we find Zeroth moments are the same. Since the first and second moments can be found by taking the first and second derivatives of these normalising constants Z_f and Z_g, the moments of the left hand side KL divergence terms remain the same. By moment matching, the RHS of each KL term is just an unnormalised Gaussian parametrised by the same first and second moments. Therefore their normalising constants must also be the same:

$$\begin{aligned} &\int \tilde{g}_i^{\text{new}}(y_i)q_{\neg i}(y_i)dy_i = \int \tilde{f}_i^{\text{new}}(x_i)q_{\neg i}(x_i)dx_i \\ \implies &\int \tilde{g}_i^{\text{new}}(y_i)q_{\neg i}(y_i)dy_i \overset{(34)}{=} \int \tilde{f}_i^{\text{new}}(x_i) \left(\int q_{\neg i}(y_i)p(x_i|y_i)dy_i \right) dx_i \end{aligned}$$

changing the order of integration:

$$\begin{aligned} \implies &\int \tilde{g}_i^{\text{new}}(y_i)q_{\neg i}(y_i)dy_i = \int \left(\int \tilde{f}_i^{\text{new}}(x_i)p(x_i|y_i)dx_i \right) q_{\neg i}(y_i)dy_i \\ \implies &\tilde{g}_i^{\text{new}}(y_i) = \int \tilde{f}_i^{\text{new}}(x_i)p(x_i|y_i)dx_i \end{aligned}$$

Hence we have shown the relation $\tilde{g}_i(y_i) = \int \tilde{f}_i(x_i)p(x_i|y_i)dx_i$ holds true after an iteration. In total this means that the algorithms are equivalent in that they will have the same fixed points since the resulting marginal is the same as the marginal found in Part a:

$$P(\mathcal{Y}) = \prod_i p(y_i|y_{i-1})\tilde{g}_i(y_i) = \prod_i p(y_i|y_{i-1}) \int p(x_i|y_i)\tilde{f}_i(x_i)dx_i = \int P(\mathcal{X}, \mathcal{Y})dx_{1:n}$$

2 EP for binary latent factor model

2.1 Part a

The binary latent factor model is defined as

$$\boldsymbol{x}|\boldsymbol{s} \sim \mathcal{N}(\sum_{i=1}^{K} s_i \mu_i, \sigma^2 \mathbb{I}) \tag{36}$$

$$\boldsymbol{s} \sim \prod_{i=1}^{K} \pi_i^{s_i} (1-\pi_i)^{(1-s_i)} \tag{37}$$

Therefore we can write down the log-joint probability:

$$\begin{aligned}
\log p(\boldsymbol{x}, \boldsymbol{s}) &= \log p(\boldsymbol{x}|\boldsymbol{s}) + \log p(\boldsymbol{s}) \\
&= \sum_{k=1}^{K} (s_i \log \pi_i + (1-s_i)\log(1-\pi_i) + \log\left(\frac{1}{(2\pi)^{\frac{D}{2}}|\sigma^2\mathbf{I}|}\right) \\
&\quad - \frac{1}{2\sigma^2}(\boldsymbol{x} - \sum_i s_i \boldsymbol{\mu}_i)^\top \mathbf{I} (\boldsymbol{x} - \sum_i s_i \boldsymbol{\mu}_i) \\
&= \sum_i s_i \log \frac{\pi_i}{1-\pi_i} - \frac{1}{2\sigma^2}(\boldsymbol{x}^\top \boldsymbol{x} - 2\boldsymbol{x}^\top \sum_i s_i \boldsymbol{\mu}_i + \sum_i \sum_j \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \\
&\quad + \sum_i \log(1-\pi_i) - \frac{D}{2}\log(2\pi\sigma^2) \\
&= \sum_i s_i \left(\log \frac{\pi_i}{1-\pi_i} + \frac{\boldsymbol{x}^\top \boldsymbol{\mu}_i}{\sigma^2}\right) - \frac{1}{2\sigma^2} \sum_i \sum_j s_i s_j \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j + C \qquad (38) \\
&= \sum_i \log f_i(s_i) + \sum_{ij} \log g_{ij}(s_i, s_j)
\end{aligned}$$

note that $s_i^2 = s_i$ and hence (39) is equivalent to:

$$= \sum_i s_i \left(\log \frac{\pi_i}{1-\pi_i} + \frac{\boldsymbol{x}^\top \boldsymbol{\mu}_i}{\sigma^2} - \frac{\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i}{2\sigma^2}\right) - \frac{1}{\sigma^2} \sum_{i<j} s_i s_j \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j + C \tag{39}$$

$$= \sum_i b_i s_i + \sum_{i<j} W_{ij} s_i s_j + \log Z \tag{40}$$

Which is the exact same form as the Boltzmann Machine with coefficients:

$$b_i = \log \frac{\pi_i}{1-\pi_i} + \frac{\boldsymbol{x}^\top \boldsymbol{\mu}_i}{\sigma^2} - \frac{\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i}{2\sigma^2} \tag{41}$$

$$W_{ij} = -\frac{1}{\sigma^2} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \tag{42}$$

2.2 Part b

We have

$$\begin{aligned}\log p(\boldsymbol{s}, \boldsymbol{x}) &\propto \sum_i b_i s_i + \sum_{i<j} W_{ij} s_i s_j \\ p(\boldsymbol{s}, \boldsymbol{x}) &\propto e^{\sum_i b_i s_i + \sum_{i<j} W_{ij} s_i s_j} \\ &= \prod_i e^{b_i s_i} \prod_{i<j} e^{W_{ij} s_i s_j} \\ &= \prod_i f_i(s_i) \prod_{i<j} g_{ij}(s_i, s_j)\end{aligned}$$

where

$$f_i(s_i) = e^{b_i s_i} \quad \text{and} \quad g_{ij}(s_i, s_j) = e^{W_{ij} s_i s_j} \tag{43}$$

Hence $f_i(s_i)$ is clearly in the expfam, and $g_{ij}(s_i, s_j)$ is not. Therefore we do not need to approximate $f_i(s_i)$ and can design a tractable message passing scheme by making the approximation $g_{ij}(s_i, s_j) = \tilde{g}_{ij}(s_i, s_j)$, where we make the approximation:

$$\tilde{g}_{ij}(s_i, s_j) = e^{\theta_i s_i + \theta_j s_j} \tag{44}$$

In order to find the update for $\tilde{g}_{ij}$ we use EP:

$$\tilde{g}_{ij}(s_i, s_j) = \underset{\tilde{g}}{\operatorname{argmin}} \mathbf{KL}[g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j) || \tilde{g}(s_i, s_j) q_{\neg ij}(s_i, s_j)] \tag{45}$$

Firstly we find the cavity distribution:

$$\begin{aligned}q_{\neg ij}(\mathcal{S}) &= \frac{q(\mathcal{S})}{\tilde{g}_{ij}(s_i, s_j)} = \frac{\prod_m f_m(s_m) \prod_{m<n} \tilde{g}_{mn}(s_m, s_n)}{\tilde{g}_{ij}(s_i, s_j)} \\ &= \prod_m f_m(s_m) \prod_{\substack{m<n \\ m,n \neq i,j}} \tilde{g}_{mn}(s_m, s_n) \\ \implies q_{\neg ij}(s_i, s_j) &= \int \prod_m f_m(s_m) \prod_{\substack{m<n \\ m,n \neq i,j}} \tilde{g}_{mn}(s_m, s_n) d\mathcal{S}_{\neg ij} \\ &= \alpha_i(s_i) \cdot \beta_j(s_j) = e^{T_\alpha s_i + T_\beta s_j}\end{aligned}$$

where the last line represents the messages received into nodes s_i and s_j, whose information is parametrised by T_α, T_β. The messages clearly take this form because all the factors have the same form: $\tilde{g}_{mn}(s_m, s_n) = e^{\theta_i s_i + \theta_j s_j}$ and $f_m(s_m) = e^{b_m s_m}$. We our now equipped to find $\tilde{g}_{ij}$ via moment matching. Note that the sufficient statistics are simply s_i and s_j, and so we find the expected value of s_i under each distribution $\tilde{g}_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j)$ and $g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j)$

$$\begin{aligned}\tilde{g}_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j) &= e^{(T_\alpha + \theta_i) s_i + (T_\beta + \theta_j) s_j} \\ \overset{\text{marginalise}}{\implies} \tilde{g}_{ij}(s_i) q_{\neg ij}(s_i) &\propto e^{(T_\alpha + \theta_i) s_i}\end{aligned}$$

$$\overset{\text{normalise}}{\Longrightarrow} \tilde{g}_{ij}(s_i)q_{\neg ij}(s_i) = \frac{e^{(T_\alpha+\theta_i)s_i}}{1+e^{(T_\alpha+\theta_i)}}$$

And so the moment of s_i under $\tilde{g}_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j)$ is

$$\boxed{\mathbb{E}_{\tilde{g}_{ij}}[s_i] = \frac{1}{1+e^{-(T_\alpha+\theta_i)}}} \tag{46}$$

We also have

$$\begin{aligned} g_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j) &= e^{W_{ij}s_is_j+T_\alpha s_i+T_\beta s_j} \\ \overset{\text{marginalise}}{\Longrightarrow} g_i(s_i)q_{\neg ij}(s_i) &\propto e^{T_\alpha s_i} + e^{W_{ij}s_i+T_\alpha s_i+T_\beta} = e^{T_\alpha s_i}(1+e^{W_{ij}s_i+T_\beta}) \\ \overset{\text{normalise}}{\Longrightarrow} g_i(s_i)q_{\neg ij}(s_i) &= \frac{e^{T_\alpha s_i}(1+e^{W_{ij}s_i+T_\beta})}{1+e^{T_\beta}+e^{T_\alpha s_i}(1+e^{W_{ij}s_i+T_\beta})} \end{aligned}$$

Therefore (using standard expectation formula):

$$\boxed{\mathbb{E}_{g_{ij}}[s_i] = \frac{1}{1+e^{-T_\alpha}\left(\frac{1+e^{T_\beta}}{1+e^{W_{ij}+T_\beta}}\right)}} \tag{47}$$

Equating moments - i.e. equations of (46) and (47) - we find the parameter θ_i:

$$\begin{aligned} \frac{1}{1+e^{-(T_\alpha+\theta_i)}} &= \frac{1}{1+e^{-T_\alpha}\left(\frac{1+e^{T_\beta}}{1+e^{W_{ij}+T_\beta}}\right)} \\ \implies e^{-(T_\alpha+\theta_i)} &= e^{-T_\alpha}\left(\frac{1+e^{T_\beta}}{1+e^{W_{ij}+T_\beta}}\right) \\ \implies -(T_\alpha+\theta_i) &= -T_\alpha + \log\left(\frac{1+e^{T_\beta}}{1+e^{W_{ij}+T_\beta}}\right) \\ \boxed{\theta_i = \log\left(\frac{1+e^{W_{ij}+T_\beta}}{1+e^{T_\beta}}\right)} \end{aligned} \tag{48}$$

and by symmetry

$$\boxed{\theta_j = \log\left(\frac{1+e^{W_{ij}+T_\alpha}}{1+e^{T_\alpha}}\right)} \tag{49}$$

2.3 Part c

Now let $\tilde{g}_{ij} = M_{i\to j}(s_j)M_{j\to i}(s_i)$, then the full posterior is:

$$q(\mathcal{S}) = \prod_m f_m(s_m) \prod_{n\in ne(m)} M_{n\to m}(s_m)$$

and therefore, following the derivations in the lectures

$$q_{\neg ij}(s_i, s_j) = f(s_i)f(s_j) \prod_{J \in ne(i)/j} M_{J\to i}(s_i) \prod_{I \in ne(j)/i} M_{I\to j}(s_j)$$

Then the update equation for the new messages is:

$$\{M^{\text{new}}_{i\to j}(s_j), M^{\text{new}}_{j\to i}(s_i)\} = \text{argmin}\, \mathbf{KL}[g_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j)||M_{i\to j}(s_j)M_{j\to i}(s_i)q_{\neg ij}(s_i, s_j)] \quad (50)$$

Let the `proj` operator denote projection onto the Bernoulli expfam. Then we also know from the lectures that, through marginalisation:

$$M^{\text{new}}_{j\to i}(s_i) = \texttt{proj}\left[\sum_{s_j} g_{ij}(s_i, s_j)f_j(s_j) \prod_{I \in ne(j)/i} M_{I\to j}(s_j)\right]$$

Now remember that $f_j(s_j) = e^{b_j s_j}, g_{ij} = e^{W_{ij}s_i s_j}$ and let $M_{I\to j}(s_j)$ have natural parameter n_{Ij} - i.e. $M_{I\to j} = e^{n_{Ij}s_j}$. Then

$$\begin{aligned}
M^{\text{new}}_{j\to i}(s_i) &= \texttt{proj}\left[\sum_{s_j} e^{W_{ij}s_i s_j} e^{b_j s_j} \prod_{I \in ne(j)/i} e^{n_{Ij}s_j}\right] = \texttt{proj}\left[\sum_{s_j} e^{W_{ij}s_i s_j + b_j s_j + \sum_{I \in ne(j)/i} n_{Ij}s_j}\right] \\
&= \texttt{proj}\left[1 + e^{W_{ij}s_i + b_j + \sum_{I \in ne(j)/i} n_{Ij}}\right] \\
&\overset{\text{normalise}}{=} \texttt{proj}\left[\frac{1 + e^{W_{ij}s_i + b_j + \sum_{I \in ne(j)/i} n_{Ij}}}{(1 + e^{W_{ij} + b_j + \sum_{I \in ne(j)/i} n_{Ij}}) + (1 + e^{+b_j + \sum_{I \in ne(j)/i} n_{Ij}})}\right] \\
&\overset{\text{def}}{=} \texttt{proj}\left[\phi(s_i)\right]
\end{aligned}$$

Now note that we want to project to the bernoulli expfam, i.e. we want

$$M^{\text{new}}_{j\to i}(s_i) = e^{s_i \log \frac{p}{1-p}} = e^{s_i n_{j\to i}}$$

which has moment $\langle s_i \rangle_{M_{j\to i}} = p$. Therefore we want to perform moment matching with

$$\langle s_i \rangle_{M_{j\to i}} = \langle s_i \rangle_\phi$$

Now,

$$\langle s_i \rangle_\phi = \sum_{s_i} s_i \phi(s_i) = \frac{1 + e^{W_{ij} + b_j + \sum_{I \in ne(j)/i} n_{Ij}}}{(1 + e^{W_{ij} + b_j + \sum_{I \in ne(j)/i} n_{Ij}}) + (1 + e^{b_j + \sum_{I \in ne(j)/i} n_{Ij}})} \overset{def}{=} \frac{A}{A + B}$$

$$\langle s_i \rangle_{M_{j\to i}} = p$$

therefore since $\langle s_i \rangle_{M_{j\to i}} = \langle s_i \rangle_\phi$

$$\implies n_{j\to i} = \log\left(\frac{\langle s_i \rangle_\phi}{1 - \langle s_i \rangle_\phi}\right) = \log\left(\frac{\frac{A}{A+B}}{1 - \frac{A}{A+B}}\right) = \log\left(\frac{A}{B}\right)$$

$$\boxed{n_{j\to i} = \log\left(\frac{1+e^{W_{ij}+b_j+\sum_{I\in ne(j)/i} n_{Ij}}}{1+e^{b_j+\sum_{I\in ne(j)/i} n_{Ij}}}\right)} \tag{51}$$

Hence we see that the message $M^{\text{new}}_{j\to i}(s_i)$ has natural parameter $n_{j\to i}$ given by the formula above. By symmetry then, the message $M^{\text{new}}_{i\to j}(s_j)$ has natural parameter $n_{i\to j}$ given by:

$$\boxed{n_{i\to j} = \log\left(\frac{1+e^{W_{ij}+b_i+\sum_{J\in ne(i)/j} n_{Ji}}}{1+e^{b_i+\sum_{J\in ne(i)/j} n_{Ji}}}\right)} \tag{52}$$

In fact we see that the message passing scheme for part (b) has the exact same form except with:

$$T_\alpha = b_i + \sum_{J\in ne(i)/j} n_{Ji}$$
$$T_\beta = b_j + \sum_{I\in ne(j)/i} n_{Ij}$$

$$\tilde{g}_{ij} = M_{i\to j}(s_j)M_{j\to i}(s_i) =$$

In this case we see:

$$\begin{aligned} q_{\neg ij}(s_i, s_j) &= f(s_i)f(s_j)\prod_{J\in ne(i)/j} M_{J\to i}(s_i)\prod_{I\in ne(j)/i} M_{I\to j}(s_j) \\ &= e^{s_i b_i + s_j b_j + s_i\sum_{J\in ne(i)/j} n_{Ji} + s_j\sum_{I\in ne(j)/i} n_{Ij}} \\ &= e^{s_i(b_i+\sum_{J\in ne(i)/j} n_{Ji}) + s_j(b_j+\sum_{I\in ne(j)/i} n_{Ij})} \\ &= e^{s_i T_\alpha + s_j T_\beta} \end{aligned}$$

So the cavity distributions are exactly the same, which means the moments of $\tilde{g}_{ij}q_{\neg ij}(s_i, s_j)$ and $M_{i\to j}(s_j)M_{j\to i}(s_i)q_{\neg ij}(s_i, s_j)$ are the same and so we have a loopy BP version of the EP algorithm.

2.4 Part d

One way of selecting K would be to run the EP algorithm with a large range of K values. This would then produce a set of models $\mathcal{M}_k$ indexed by their K value. Each model will have a model evidence given by:

$$P(\mathcal{D}|\mathcal{M}_k) = \int d\theta_k P(\mathcal{D}|\theta_k, \mathcal{M}_k)P(\theta_k|\mathcal{M}_k) \tag{53}$$

We could then conclude that the model with the highest model evidence will give the best K value. This would pose computational difficulties since for every model we would have to integrate out the model parameters.

3 Implementing the loopy-BP algorithm

In the loopy-BP algorithm we find the expected moments of s_i. Therefore we can re-use much of the VB algorithm (specifically the M-step for the parameters μ, σ and π remains exactly the same) and we only have to change the E-step for calculating these expected moments of s_i via the message passing scheme rather than through the mean-field approximation. We see from Part c that the marginal on a site s_i is:

$$\begin{aligned} q(s_i) &= \int q(\mathcal{S}) d\mathcal{S}_{\neg i} = \int \prod_m f_m(s_m) \prod_{n \in ne(m)} M_{n \to m}(s_m) d\mathcal{S}_{\neg i} \\ &\propto f_i(s_i) \prod_{j>i} M_{j \to i}(s_i) = e^{s_i(b_i + \sum_{j \in ne(i)} n_{j \to i})} \end{aligned} \tag{54}$$

The natural parameter of a bernoulli distribution has the form $\log \frac{p}{1-p}$ with $\langle s_i \rangle = p$, and therefore:

$$\begin{aligned} (b_i + \sum_{j \in ne(i)} n_{j \to i}) &= \log \frac{p}{1-p} \\ \implies e^{(b_i + \sum_{j \in ne(i)} n_{j \to i})} &= \frac{p}{1-p} \\ \implies \boxed{\langle s_i \rangle = p = \frac{1}{1 + e^{-(b_i + \sum_{j \in ne(i)} n_{j \to i})}}} \end{aligned}$$

After computing the messages to each site, we can then find the expected value of each s_i using this formula, and proceed with the rest of the algorithm unchanged. Specifically the code for this loopy-BP 'E-step' is given below. The b_i and W_{ij} values are first instantiated as $(N \times K)$ and $(N \times K \times K)$ matrices called `bs` and `Ws` respectively. The messages are represented as an $(N \times K \times K)$ matrix called `messages`. This message matrix holds the natural parameters $n_{i \to j}$ of each message $M_{i \to j}$. The message passing scheme (equations (51) and (52)) is iterated for a fixed number of iteration or until the free energy stops increasing. The message passing scheme is implemented below in the function `loopyBP`:

```
def loopyBP(X, mu, sigma, pie, lambda0, messages, maxsteps):

    N, D = X.shape
    K = mu.shape[1]
    Lambda = lambda0.copy()
    fs = []
    eps = 1
    step = 0
    constant = 1e-15
    messages = messages.copy()

    """
    bs is (NxK)
    Ws is (KxK)
    messages is (N, K, K)
    """

    bs = np.log(pie/(1-pie)) + X @ mu/sigma**2 - np.diag(mu.T @ mu)/(2*sigma**2)
    assert bs.shape == (N,K)

    Ws = -(mu.T @ mu)/(sigma**2)

    while abs(eps) > 1e-10 and step < maxsteps:

        for i in range(K):
```

```
            for j in range(i+1,K):

                    ### Messages j -> i
                    term_1 = bs[:,j] + np.sum(messages[:,:,j], axis=1) - messages[:,i,j]
                    term_2 = term_1 + Ws[i,j]
                    messages[:,j,i] = np.log((1+np.exp(term_2))/(1 + np.exp(term_1)))

                    ### Messages i -> j
                    term_1 = bs[:,i] + np.sum(messages[:,:,i], axis=1) - messages[:,j,i]
                    term_2 = term_1 + Ws[j,i]
                    messages[:,i,j] = np.log((1+np.exp(term_2))/(1+np.exp(term_1)))

        ### Find expected values of s_i
        Lambda = sigmoid(bs + np.sum(messages, axis = 1))
        assert Lambda.shape == (N, K)

        f_ = compute_free_energy(X, mu, sigma, pie, Lambda, constant)

        if(step > 0):
            eps = f_ - fs[step - 1]
        fs.append(f_)
        step += 1

    return Lambda, fs[-1], fs
```

This is the only part of the VB algorithm which substantially changes, and the rest of the code remains very similar to the VB algorithm:

```
def compute_free_energy(X, mu, sigma, pie, Lambda, constant):
    Lambda[np.isclose(Lambda, 0)] = constant
    Lambda[np.isclose(Lambda, 1)] = 1 - constant
    term_1 = np.sum(np.multiply(Lambda, np.log(pie/Lambda)))
    term_2 = np.sum(np.multiply((1-Lambda), np.log((1-pie)/(1-Lambda))))
    term_3 = -N*D*np.log(sigma)
    weights = (X - Lambda @ mu.T)
    term_4 = -np.trace((2*sigma**2)**-1*(weights @ weights.T))
    term_5 = -np.sum((2*sigma**2)**-1*(Lambda - Lambda**2) @ np.diag(mu.T @ mu))
    term_6 = -N*(D/2)*np.log(2*np.pi)
    f_ = np.sum(term_1 + term_2 + term_3 + term_4 + term_5 + term_6)

    return f_

def m_step(X, ES, ESS):
    """
    mu, sigma, pie = MStep(X,ES,ESS)

    Inputs:
    -----------------
           X: shape (N, D) data matrix
          ES: shape (N, K) E_q[s]
         ESS: shape (K, K) sum over data points of E_q[ss'] (N, K, K)
                           if E_q[ss'] is provided, the sum over N is done for you.

    Outputs:
    --------
          mu: shape (D, K) matrix of means in p(y|{s_i},mu,sigma)
       sigma: shape (,)    standard deviation in same
         pie: shape (1, K) vector of parameters specifying generative distribution for s
    """
    N, D = X.shape
    if ES.shape[0] != N:
        raise TypeError('ES must have the same number of rows as X')
    K = ES.shape[1]
    if ESS.shape == (N, K, K):
        ESS = np.sum(ESS, axis=0)
    if ESS.shape != (K, K):
        raise TypeError('ESS must be square and have the same number of columns as ES')

    mu = np.dot(np.dot(np.linalg.inv(ESS), ES.T), X).T
    sigma = np.sqrt((np.trace(np.dot(X.T, X)) + np.trace(np.dot(np.dot(mu.T, mu), ESS))
                     - 2 * np.trace(np.dot(np.dot(ES.T, X), mu))) / (N * D))
    pie = np.mean(ES, axis=0, keepdims=True)

    return mu, sigma, pie

def compute_ESS(Lambda):
    """
    Computes E_q[ss'] (N, K, K), using Lambda matrix.
    """
    N, K = Lambda.shape
    ESS = np.zeros(shape=(N,K,K))

    for n in range(N):
        lambda_n = Lambda[n,:]
        for k_1 in range(K):
            for k_2 in range(K):
                ESS[n,k_1,k_2] = lambda_n[k_1]*lambda_n[k_2]
        diag = lambda_n - lambda_n**2
```

```
            ESS[n,:,:] += np.diag(diag)

        return ESS

def init_params(N, D, K):
        Lambda = 0.25 + np.random.rand(N,K)/2.0
        mu = 0.25 + np.random.rand(D,K)/2.0
        pie = 0.25 + np.random.rand(1,K)/2.0
        sigma = 1
        messages = 0.25 + np.random.rand(N,K,K)*0.5
        for i in range(K):
            messages[:,i,i] = 0.0

        return Lambda, mu, pie, sigma, messages

def learn_bin_factors(X, K, iterations):

        N, D = X.shape
        Lambda, mu, pie, sigma, messages = init_params(N,D,K)
        maxsteps = 100
        free_energies = []

        for cIter in range(iterations):

            messages = 0.25 + np.random.rand(N,K,K)*0.5
            for i in range(K):
                messages[:,i,i] = 0.0

            Lambda, f, f_1 = loopyBP(X, mu, sigma, pie, Lambda, messages, maxsteps)
            free_energies.append(f)
            print(f)
            ESS = compute_ESS(Lambda)

            mu, sigma, pie = m_step(X, Lambda, ESS)

        return mu, sigma, pie, Lambda, free_energies
```

Note that before every iteration of loopy-BP we initialise the messages uniformly in the range 0.25 to 0.75 with diagonal elements - i.e. messages $M_{i \to i}$ set to zero (as seen in the function `learn_bin_factors`). Below is a plot of the free energy:

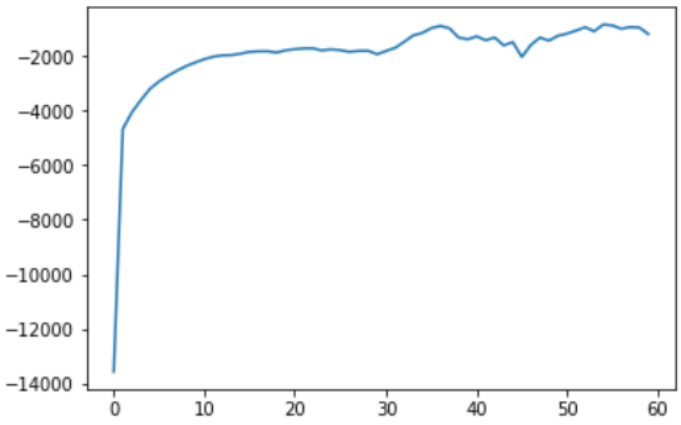


Figure 1: Plot of free energy vs iteration

As we see, unlike the mean-field algorithm, the free energy is not always increasing. This is due to the loopiness in the graphical model which provides no guarantees for an increasing free energy. This is in comparison to the VB mean-field algorithm, which although forms an approximate posterior over the latents, still guarantees an increasing free energy. However, we found that when $K = 8$, the features it found was consistently better than the mean-field algorithm. On a good run, the algorithm was able to find all 8 features:

Figure 2: Loopy-BP Features

Additionally, we provide for comparison the features found on one of the best runs of Mean Field:

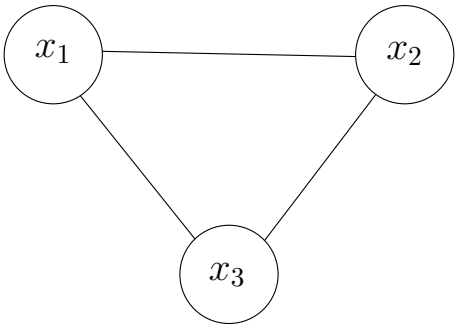
Figure 3: Mean-Field Features

Note that the algorithm was only able to find at most 7 features on the best run. Therefore we conclude that the Loopy BP algorithm seems to perform better than the Mean-Field algorithm. In fact, on more typical runs for both algorithms we saw that Loopy-BP generally found clearer features. This agrees with the analysis in the lectures that although Loopy-BP has no convergence guarantees the results generally outperform the Mean-field algorithm.

4 Inconsistency of Local Marginals

4.1 Part a

Consider the simplest possible case of three binary variables $x_1, x_2, x_3 \in \{0, 1\}$ in the loopy graph structure:



A set of consistent local singleton beliefs is:

$$b_i(x_i) = \begin{Bmatrix} 0.5, & \text{if } x_i = 0 \\ 0.5, & \text{if } x_i = 1 \end{Bmatrix}$$

And pairwise:

$$b_{ij}(x_i, x_j) = \begin{Bmatrix} b_{ij}, & \text{if } x_i = 0, x_j = 0 \\ 0.5 - b_{ij}, & \text{if } x_i = 0, x_j = 1 \\ 0.5 - b_{ij}, & \text{if } x_i = 1, x_j = 0 \\ b_{ij}, & \text{if } x_i = 1, x_j = 1 \end{Bmatrix}$$

i.e. we have that

$$b_i(x_i) = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \quad b_{ij}(x_i, x_j) = \begin{pmatrix} b_{ij} & 0.5 - b_{ij} \\ 0.5 - b_{ij} & b_{ij} \end{pmatrix} \tag{55}$$

Note the singletons are consistent since for all i we clearly have $\sum_{x_i} b_i(x_i) = 1$, and for the pairwise $\sum_{x_i} b_{ij}(x_i, x_j) = b_j(x_j)$ (which follows from the fact that summing over the rows gives $0.5 = b_j$, and summing over the columns gives $0.5 = b_i$). However, if we now set $b_{12} = 0$, $b_{13} = 0.5$ and $b_{23} = 0.5$ then we have that

$$\begin{aligned} p(X_1 = X_2) &= p(x_1 = 0, x_2 = 0) + p(x_1 = 1, x_2 = 1) = b_{12}(0,0) + b_{12}(1,1) = 0 + 0 = 0 \\ p(X_1 = X_3) &= p(x_1 = 0, x_3 = 0) + p(x_1 = 1, x_3 = 1) = b_{13}(0,0) + b_{13}(1,1) = 0.5 + 0.5 = 1 \\ p(X_2 = X_3) &= p(x_2 = 0, x_3 = 0) + p(x_2 = 1, x_3 = 1) = b_{23}(0,0) + b_{23}(1,1) = 0.5 + 0.5 = 1 \end{aligned}$$

This is globally inconsistent since these beliefs imply $x_1 \neq x_2$, $x_2 = x_3$ but $x_3 = x_1$, i.e. we have a contradiction. So we have a set of locally consistent but globally inconsistent beliefs.