Comp 0085 Assignment 6

Alexandra Maria Proca (SN: 20047328)

January 20, 2021

Question 1.

a. Let $p(y_1|y_0) = p(y_1)$. The joint probability of $\boldsymbol{x}, \boldsymbol{y}$ can written as

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} p(y_i|y_{i-1})p(x_i|y_i)p(s_i|x_i)$$

$$= \prod_{i=1}^{N} p(y_i|y_{i-1})p(x_i|y_i)f_i(x_i)$$

$f_i(x_i) \approx \tilde{f}_i(x_i)$, where $\tilde{f}_i(x_i)$ is Gaussian. Thus, the cavity can be written as

$$q_{\neg i}(x_i) = \int\limits_{y_1,\dots,y_N} \prod_{i=1}^{N} p(y_i|y_{i-1})p(x_i|y_i) \int\limits_{x_{i'}\neg x_i} \prod_{i'\neq i} \tilde{f}_{i'}(x_{i'})$$

$$= \int\limits_{y_{i-1},y_i} \left(p(y_i|y_{i-1})p(x_i|y_i) \int\limits_{y_{i'-1},y_{i'},x_{i'}\neg y_{i-1},y_i,x_i} \prod_{i'\neq i} p(y_{i'}|y_{i'-1})p(x_{i'}|y_{i'})\tilde{f}_{i'}(x_{i'}) \right)$$

The expression can be factored in terms of forward and backward messages, yielding

$$\int\limits_{y_{i'-1},y_{i'},x_{i'}\neg y_{i-1},y_i,x_i} p(y_{i'}|y_{i'-1})p(x_{i'}|y_{i'})\tilde{f}_{i'}(x_{i'}) = \alpha_{i-1}(y_{i-1})\beta_i(y_i)$$

where $\alpha_{i-1}(y_{i-1}) \sim \mathcal{N}(\mu_\alpha, \Sigma_\alpha),\ \beta_i(y_i) \sim \mathcal{N}(\mu_\beta, \Sigma_\beta)$.

Thus, substituting into the cavity yields

$$q_{\neg i}(x_i) = \int\limits_{y_{i-1},y_i} \underbrace{p(y_i|y_{i-1})}_{\mathcal{N}(y_{i-1},\sigma^2)} \underbrace{p(x_i|y_i)}_{\mathcal{N}(y_i,\tau^2)} \underbrace{\alpha_{i-1}(y_{i-1})}_{\mathcal{N}(\mu_\alpha,\Sigma_\alpha)} \underbrace{\beta_i(y_i)}_{\mathcal{N}(\mu_\beta,\Sigma_\beta)}$$

The product of Gaussian distributions is a Gaussian of the form
$\mathcal{N}(a, A)\cdot\mathcal{N}(b, B) \propto \mathcal{N}(c, C)$
where
$C = \left(A^{-1} + B^{-1}\right)^{-1}$
$c = CA^{-1}a + CB^{-1}b$

Thus, simplifying the cavity yields,
$q_{\neg i}(x_i) \sim \mathcal{N}(\mu_{\neg i}, \Sigma_{\neg i})$
where $\Sigma_{\neg i} = \left(\left(\Sigma_\alpha + \sigma^2\right)^{-1} + \Sigma_\beta^{-1}\right)^{-1} + \tau^2$

$$\mu_{\neg i} = \frac{\Sigma_{\neg i}\mu_\alpha}{\Sigma_\alpha + \sigma^2} + \frac{\Sigma_{\neg i}\mu_\beta}{\Sigma_\beta}$$

To minimize $\text{KL}[f_i(x_i)q_{\neg i}(x_i)\|\tilde{f}_i(x_i)\ q_{\neg i}(x_i)]$, $\tilde{f}(x)q(x)$ is first solved for by computing the expected mean and variance of the Gaussian $f_i(x_i)q_{\neg i}(x_i)$.

For continuous random variables,

$$\mu = \int x f(x)$$
$$\Sigma = \int (x-\mu) f(x)$$

Thus,

$$\mu_i = \frac{1}{Z}\int_{x_i} x_i f_i(x_i) q_{\neg i}(x_i) = E(\mu_{\neg i}, \Sigma_{\neg i})$$
$$\Sigma_i = \frac{1}{Z}\int_{x_i} (x_i - \mu_i) f_i(x_i) q_{\neg i}(x_i) = V(\mu_{\neg i}, \Sigma_{\neg i})$$
$$\tilde{f}_i(x_i)\ q_{\neg i}(x_i) \sim \mathcal{N}(E(\mu_{\neg i}, \Sigma_{\neg i}), V(\mu_{\neg i}, \Sigma_{\neg i}))$$

To solve for $\tilde{f}_i(x_i)$

$$\tilde{f}_i(x_i)\ = \frac{\mathcal{N}(E(\mu_{\neg i}, \Sigma_{\neg i}), V(\mu_{\neg i}, \Sigma_{\neg i}))}{q_{\neg i}(x_i)}$$
$$= \frac{\mathcal{N}(E(\mu_{\neg i}, \Sigma_{\neg i}), V(\mu_{\neg i}, \Sigma_{\neg i}))}{\mathcal{N}(\mu_{\neg i}, \Sigma_{\neg i})} = \mathcal{N}(\mu_i, \Sigma_i)$$

where

$$\Sigma_i = \left(V(\mu_{\neg i}, \Sigma_{\neg i})^{-1} - \Sigma_{\neg i}^{-1}\right)^{-1}$$
$$\mu_i = \frac{\Sigma_i E(\mu_{\neg i}, \Sigma_{\neg i})}{V(\mu_{\neg i}, \Sigma_{\neg i})} - \frac{\Sigma_i \mu_{\neg i},}{\Sigma_{\neg i}}$$

b. An alternative approach to the expectation propagation algorithm in (a) is to first compute the probabilities:

$$g_i(y_i) = p(\text{sign}(x_i) = s_i | y_i)$$
$$= \int_{x_i} p(s_i|x_i) p(x_i|y_i)$$
$$= \int_{x_i} f_i(x_i) p(x_i|y_i)$$

Thus, the joint probability of $\boldsymbol{x}$ and $\boldsymbol{y}$ is

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} p(y_i|y_{i-1}) g_i(y_i)$$

$g_i(y_i) \approx \tilde{g}_i(y_i)$, where $\tilde{g}_i(y_i)$ is Gaussian. Thus, the cavity can be written as

$$q_{\neg i}(y_i) = \int\limits_{y_1,\ldots,y_N} \left(\prod_{i=1}^{N} p(y_i|y_{i-1}) \int\limits_{y_{i'} \neg y_i} \prod_{i' \neq i} \tilde{g}_{i'}(y_{i'}) \right)$$
$$= \int\limits_{y_{i-1}} \left(p(y_i|y_{i-1}) \int\limits_{y_{i'-1}, y_{i'} \neg y_{i-1}, y_i} \prod_{i' \neq i} p(y_{i'}|y_{i'-1}) \tilde{g}_{i'}(y_{i'}) \right)$$

The expression can be factored in terms of forward and backward messages, yielding

$$\int\limits_{y_{i'-1}, y_{i'} \neg y_{i-1}, y_i} \prod_{i' \neq i} p(y_{i'}|y_{i'-1}) \tilde{g}_{i'}(y_{i'}) = \gamma_{i-1}(y_{i-1}) \delta_i(y_i)$$

where $\gamma_{i-1}(y_{i-1}) \sim \mathcal{N}(\mu_\gamma, \Sigma_\gamma),\ \delta_i(y_i) \sim \mathcal{N}(\mu_\delta, \Sigma_\delta)$

Thus, substituting into the cavity yields

$$q_{\neg i}(y_i) = \int\limits_{y_{i-1}} \underbrace{p(y_i|y_{i-1})}_{\mathcal{N}(y_{i-1}, \sigma^2)} \underbrace{\gamma_{i-1}(y_{i-1})}_{\mathcal{N}(\mu_\gamma, \Sigma_\gamma)} \ \underbrace{\delta_i(y_i)}_{\mathcal{N}(\mu_\delta, \Sigma_\delta)}$$

Simplifying the cavity by multiplying and integrating the Gaussians yields

$q_{\neg i}(y_i) = \mathcal{N}\left(\mu^g_{\neg i}, \Sigma^g_{\neg i}\right)$
where
$\Sigma^g_{\neg i} = \left(\left(\Sigma_\gamma + \sigma^2\right)^{-1} + \Sigma_\delta^{-1}\right)^{-1}$
$\mu^g_{\neg i} = \dfrac{\Sigma^g_{\neg i}\mu_\gamma}{\Sigma_\gamma + \sigma^2} + \dfrac{\Sigma^g_{\neg i}\mu_\delta}{\Sigma_\delta}$

$\tilde{g}_i(y_i)$ can now be solved for by proving the EP algorithm in (a) and (b) have the same fixed points and leveraging the solution of $\tilde{f}_i(x_i)$ to find $\tilde{g}_i(y_i)$.

By definition,
$g_i(y_i) = \displaystyle\int_{x_i} f_i(x_i)p(x_i|y_i)$
Thus, the approximations are assumed to yield
$\tilde{g}_i(y_i) = \displaystyle\int_{x_i} \tilde{f}_i(x_i)p(x_i|y_i)$

To show that the factorizations are equal,
$\gamma_{i-1}(y_{i-1})\delta_i(y_i) = \displaystyle\int\limits_{y_{i'-1}, y_{i'} \neg y_{i-1}, y_i} \prod_{i' \neq i} p(y_{i'}|y_{i'-1})\tilde{g}_{i'}(y_{i'})$
Substituting with $\tilde{f}_i(x_i)$
$= \displaystyle\int\limits_{y_{i'-1}, y_{i'} \neg y_{i-1}, y_i} \prod_{i' \neq i} p(y_{i'}|y_{i'-1}) \int_{x_i} \tilde{f}_i(x_i)p(x_i|y_i)$
$= \displaystyle\int\limits_{y_{i'-1}, y_{i'}, x_{i'} \neg y_{i-1}, y_i, x_i} \prod_{i' \neq i} p(y_{i'}|y_{i'-1})\tilde{f}_i(x_i)p(x_i|y_i)$
$= \alpha_{i-1}(y_{i-1})\beta_i(y_i)$

Thus,
$\gamma_{i-1}(y_{i-1})\delta_i(y_i) = \alpha_{i-1}(y_{i-1})\beta_i(y_i)$

To show that the joint probabilities are equal,
$g_i(y_i)q_{\neg i}(y_i) = \displaystyle\int\limits_{y_{i-1}} g_i(y_i)p(y_i|y_{i-1})\gamma_{i-1}(y_{i-1})\delta_i(y_i)$
Substituting with $f_i(x_i)$,
$= \displaystyle\int\limits_{y_{i-1}} \left(\int_{x_i} f_i(x_i)p(x_i|y_i)\right) p(y_i|y_{i-1})\gamma_{i-1}(y_{i-1})\delta_i(y_i)$
$= \displaystyle\int\limits_{y_{i-1}, x_i} f_i(x_i)p(x_i|y_i)p(y_i|y_{i-1})\gamma_{i-1}(y_{i-1})\delta_i(y_i)$
Substituting with α_{i-1} and β_i,
$= \displaystyle\int\limits_{y_{i-1}, x_i} f_i(x_i)p(x_i|y_i)p(y_i|y_{i-1})\alpha_{i-1}(y_{i-1})\beta_i(y_i)$
$= f_i(x_i)q_{\neg i}(x_i)$

Thus, $g_i(y_i)q_{\neg i}(y_i) = f_i(x_i)q_{\neg i}(x_i)$.

Finally, solving for $\tilde{g}_i(y_i)q_{\neg i}(y_i)$,
$\tilde{g}_i(y_i)q_{\neg i}(y_i) = \displaystyle\int\limits_{y_{i-1}} \tilde{g}_i(y_i)p(y_i|y_{i-1})\gamma_{i-1}(y_{i-1})\delta_i(y_i)$

Marginalizing with respect to y_{i-1},

$$\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{y_{i-1}, y_i} \tilde{g}_i(y_i) p(y_i|y_{i-1}) \gamma_{i-1}(y_{i-1}) \delta_i(y_i)$$

Substituting with $\tilde{f}_i(x_i)$,

$$= \int_{y_{i-1}, y_i} \left(\int_{x_i} \tilde{f}_i(x_i) p(x_i|y_i) \right) p(y_i|y_{i-1}) \gamma_{i-1}(y_{i-1}) \delta_i(y_i)$$

$$= \int_{y_{i-1}, y_i, x_i} \tilde{f}_i(x_i) p(x_i|y_i) p(y_i|y_{i-1}) \gamma_{i-1}(y_{i-1}) \delta_i(y_i)$$

Substituting with α_{i-1} and β_i,

$$= \int_{y_{i-1}, y_i, x_i} \tilde{f}_i(x_i) p(x_i|y_i) p(y_i|y_{i-1}) \alpha_{i-1}(y_{i-1}) \beta_i(y_i)$$

$$= \int_{x_i} \tilde{f}_i(x_i) q_{\neg i}(x_i)$$

Thus, $\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{x_i} \tilde{f}_i(x_i) q_{\neg i}(x_i)$

Solving for $\tilde{g}_i(y_i)$,

$$\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{x_i} \tilde{f}_i(x_i) q_{\neg i}(x_i)$$

$$\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{y_{i-1}, y_i} \left(\int_{x_i} \tilde{f}_i(x_i) p(x_i|y_i) \right) p(y_i|y_{i-1}) \alpha_{i-1}(y_{i-1}) \beta_i(y_i)$$

$$\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{y_{i-1}, y_i} \left(\int_{x_i} \tilde{f}_i(x_i) p(x_i|y_i) \right) p(y_i|y_{i-1}) \gamma_{i-1}(y_{i-1}) \delta_i(y_i)$$

$$\int_{y_i} \tilde{g}_i(y_i) q_{\neg i}(y_i) = \int_{y_{i-1}, y_i} \left(\int_{x_i} \tilde{f}_i(x_i) p(x_i|y_i) \right) q_{\neg i}(y_i)$$

$$\tilde{g}_i(y_i) = \int_{x_i} \underbrace{\tilde{f}_i(x_i)}_{\mathcal{N}(\mu_i, \Sigma_i)} \underbrace{p(x_i|y_i)}_{\mathcal{N}(y_i, \tau^2)}$$

Thus, $\tilde{g}_i(y_i) \sim \mathcal{N}(\mu_i^g, \Sigma_i^g)$
where $\Sigma_i^g = \Sigma_i + \tau^2$
$\mu_i^g = \mu_i$

Both EP algorithms are thus equivalent in that they have the same fixed points.

Question 2.

a. The log-joint probability for a single observation-source pair can be written

$$\log(p(\boldsymbol{s}, \boldsymbol{x})) = \sum_i \log f_i(s_i) + \sum_{ij} \log g_{ij}(s_i, s_j)$$

The joint probability is
$p(\boldsymbol{s}, \boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{s}) p(\boldsymbol{s})$
As defined in question (1) on assignment (5),

$$p(\boldsymbol{x}|\boldsymbol{s}) = \mathcal{N}\left(\sum_{i=1}^{K} s_i \boldsymbol{\mu}_i, \sigma^2 I \right)$$

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

Thus,

$$\log p(\boldsymbol{x},\boldsymbol{s}) = \log p(\boldsymbol{s}) + \log p(\boldsymbol{x}|\boldsymbol{s})$$

$$= \sum_{i=1}^{K} s_i \log \pi_i + (1-s_i)\log(1-\pi_i) - \frac{D}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(\boldsymbol{x} - \sum_{i=1}^{K} s_i\boldsymbol{\mu}_i\right)^{\mathsf{T}}\left(\boldsymbol{x} - \sum_{i=1}^{K} s_i\boldsymbol{\mu}_i\right)$$

$$= \sum_{i=1}^{K} s_i \log\frac{\pi_i}{1-\pi_i} + \log(1-\pi_i) - \frac{D}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} - 2\boldsymbol{x}^{\mathsf{T}}\sum_{i=1}^{K} s_i\boldsymbol{\mu}_i + \sum_{i,j=1}^{K} s_i s_j \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j\right)$$

Because the sources s are binary, when $s_i = s_j$, $s_i s_j = s_i = s_j$:

$$= \sum_{i=1}^{K}\left(s_i \log\frac{\pi_i}{1-\pi_i} + \log(1-\pi_i)\right) - \frac{D}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x} - 2\boldsymbol{x}^{\mathsf{T}}\sum_{i=1}^{K} s_i\boldsymbol{\mu}_i + \sum_{i=1}^{K} s_i\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \sum_{i<j}^{K} s_i s_j \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j\right)$$

$$= \sum_{i=1}^{K}\left(s_i \log\frac{\pi_i}{1-\pi_i} + \frac{1}{\sigma^2}\boldsymbol{x}^{\mathsf{T}} s_i\boldsymbol{\mu}_i - \frac{1}{2\sigma^2} s_i\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \log(1-\pi_i)\right) + \sum_{i<j}^{K} -\frac{1}{2\sigma^2} s_i s_j \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j - \frac{D}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}$$

Isolating terms with s_i yields

$$\sum_{i=1}^{K}\log f_i(s_i) = \sum_{i=1}^{K} s_i\log\frac{\pi_i}{1-\pi_i} + \frac{1}{\sigma^2}\boldsymbol{x}^{\mathsf{T}} s_i\boldsymbol{\mu}_i - \frac{1}{2\sigma^2} s_i\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i$$

$$f_i(s_i) = \exp\left\{s_i\log\frac{\pi_i}{1-\pi_i} + \frac{1}{\sigma^2} s_i\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\mu}_i - \frac{1}{2\sigma^2} s_i\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right\}$$

Isolating terms with $s_i s_j$ yields

$$\sum_{i<j}^{K}\log g_{ij}(s_i,s_j) = \sum_{i<j}^{K} -\frac{1}{2\sigma^2} s_i s_j \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j$$

$$g_{ij}(s_i,s_j) = \exp\left\{-\frac{1}{2\sigma^2} s_i s_j \boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j\right\}$$

The rest of the terms are constants. Thus,

$$\log(p(\boldsymbol{s})) \propto \sum_i \log f_i(s_i) + \sum_{ij}\log g_{ij}(s_i,s_j)$$

$$p(\boldsymbol{s}) = \frac{1}{Z}\prod_{i=1}^{K} f_i(s_i)\prod_{i<j}^{K} g_{ij}(s_i,s_j)$$

A Boltzmann Machine has the form $p(\mathcal{X},\mathcal{Z}) = \frac{1}{Z}\exp\left\{\sum_{i,j} W_{ij}s_i s_j + \sum_i b_i s_i\right\}$

The joint $p(\boldsymbol{s})$ can be written in the same way by representing the natural parameters of $f_i(s_i)$ and $g_{ij}(s_i,s_j)$ as b_i and W_{ij}, respectively.

$$f_i(s_i) = \exp\left\{\log\left(\frac{\pi_i}{1-\pi_i}\right)s_i + \frac{1}{\sigma^2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\mu}_i s_i - \frac{1}{2\sigma^2}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i s_i\right\}$$

$$= \exp\{b_i s_i\}$$

where

$$b_i = \left[\log\tfrac{\pi_i}{1-\pi_i} \quad \tfrac{1}{\sigma^2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\mu}_i \quad -\tfrac{1}{2\sigma^2}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_i\right]$$

$$g_{ij}(s_i,s_j) = \exp\left\{-\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j s_i s_j\right\}$$

$$= \exp\{W_{ij}s_i s_j\}$$

where

$$W_{ij} = \left[-\tfrac{1}{2\sigma^2}\boldsymbol{\mu}_i^{\mathsf{T}}\boldsymbol{\mu}_j\right]$$

By these definitions, the joint $p(\boldsymbol{s})$ is a Boltzmann machine.

b. From (a),

$$p(\boldsymbol{s}) = \frac{1}{Z}\prod_i f_i(s_i)\prod_{i<j} g_{ij}(s_i, s_j)$$

Not all of the EP approximations are needed. Only g_{ij} must be approximated; f_i does not need to be approximated.

$$p(\boldsymbol{s}) \approx q(\boldsymbol{s}) = \frac{1}{Z}\prod_i f_i(s_i)\prod_{i<j} \tilde{g}_{ij}(s_i, s_j)$$

Factoring the approximation, $\tilde{g}_{ij}(s_i, s_j) = \alpha_i(s_i)\beta_j(s_j)$ with natural parameters η_α and η_β, respectively.

Thus, the cavity is

$$q_{\neg ij}(s_i, s_j) = f_i(s_i)f_j(s_j)\prod_{k\neq i,j}\alpha_k(s_i)\prod_{l\neq i,j}\beta_l(s_j)$$

To derive the message passing scheme to find $\tilde{g}_{ij}$, the KL divergence is minimized:

$$\left\{\alpha_i^{\text{new}}(s_i), \beta_j^{\text{new}}(s_j)\right\} = \arg\min \mathsf{KL}[g_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j)||\tilde{g}_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j)]$$
$$= \arg\min \mathsf{KL}[g_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j)||\alpha_i(s_i)\beta_j(s_j)\ q_{\neg ij}(s_i, s_j)]$$

Marginalizing the cavity yields

$$q_{\neg ij}(s_i) = f_i(s_i)\prod_{k\neq i,j}\alpha_k(s_k)$$

$$q_{\neg ij}(s_j) = f_j(s_j)\prod_{l\neq i,j}\beta_l(s_l)$$

To solve for the natural parameters of $\alpha_i(s_i)$, each side is marginalized with respect to s_i

$$g_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j) = \alpha_i(s_i)\beta_j(s_j)\ q_{\neg ij}(s_i, s_j)$$

$$\Rightarrow \alpha_i(s_i)q_{\neg ij}(s_i) = \sum_{s_j}\frac{1}{Z}g_{ij}(s_i, s_j)q_{\neg ij}(s_i, s_j)$$

$$\alpha_i(s_i)q_{\neg ij}(s_i) = \sum_{s_j}\frac{1}{Z}\left(g_{ij}(s_i, s_j)f_j(s_j)\prod_{l\neq i,j}\beta_l(s_j)\right)f_i(s_i)\prod_{k\neq i,j}\alpha_k(s_i)$$

$$\alpha_i(s_i) = \sum_{s_j}\frac{1}{Z}\left(g_{ij}(s_i, s_j)f_j(s_j)\prod_{l\neq i,j}\beta_l(s_j)\right)$$

$$\exp\{\eta_\alpha s_i\} = \sum_{s_j}\frac{1}{Z}\left(\exp\left\{W_{ij}s_is_j + b_js_j + \sum_{l\neq i,j}\eta_{\beta_l}s_j\right\}\right)$$

Because s_j is binary (0,1):

$$\exp\{\eta_\alpha s_i\} = \frac{1}{Z}\left(\exp\{0\} + \exp\left\{W_{ij}s_i + b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}\right)$$

$$\exp\{\eta_\alpha s_i\} = \frac{1}{Z}\left(1 + \exp\left\{W_{ij}s_i + b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}\right)$$

where Z is the normalizer. Z is defined by setting all natural parameters that take cliques other than s_j (W_{ij} takes s_i) to 0.

$$Z = 1 + \exp\left\{b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}$$

Thus,

$$\exp\{\eta_\alpha s_i\} = \frac{1+\exp\left\{W_{ij}s_i + b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}}{1+\exp\left\{b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}}$$

$$\eta_\alpha = \log\frac{1+\exp\left\{W_{ij} + b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}}{1+\exp\left\{b_j + \sum_{l\neq i,j}\eta_{\beta_l}\right\}}$$

Similarly for $\beta_j(s_j)$,

$g_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j) = \alpha_i(s_i)\beta_j(s_j)\ q_{\neg ij}(s_i,s_j)$

$$\Rightarrow \beta_j(s_j)q_{\neg ij}(s_j) = \sum_{s_i}\frac{1}{Z}g_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j)$$

$$\beta_j(s_j)q_{\neg ij}(s_j) = \sum_{s_i}\frac{1}{Z}\left(g_{ij}(s_i,s_j)f_i(s_i)\prod_{k\neq i,j}\alpha_k(s_i)\right)f_j(s_j)\prod_{l\neq i,j}\beta_l(s_j)$$

$$\beta_j(s_j) = \sum_{s_i}\frac{1}{Z}\left(g_{ij}(s_i,s_j)f_i(s_i)\prod_{k\neq i,j}\alpha_k(s_i)\right)$$

$$\exp\{\eta_\beta s_j\} = \sum_{s_i}\frac{1}{Z}\left(\exp\left\{W_{ij}s_is_j + b_is_i + \sum_{k\neq i,j}\eta_{\alpha_k}s_i\right\}\right)$$

$$\exp\{\eta_\beta s_j\} = \frac{1}{Z}\left(\exp\{0\} + \exp\left\{W_{ij}s_j + b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}\right)$$

$$\exp\{\eta_\beta s_j\} = \frac{1}{Z}\left(1 + \exp\left\{W_{ij}s_j + b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}\right)$$

$$Z = 1+\exp\left\{b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}$$

$$\exp\{\eta_\beta s_j\} = \frac{1+\exp\left\{W_{ij}s_j + b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}}{1+\exp\left\{b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}}$$

$$\eta_\beta = \log\frac{1+\exp\left\{W_{ij} + b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}}{1+\exp\left\{b_i + \sum_{k\neq i,j}\eta_{\alpha_k}\right\}}$$

c. From (a),

$$p(\boldsymbol{s}) = \frac{1}{Z}\prod_i f_i(s_i)\prod_{i<j}g_{ij}(s_i,s_j)$$

Extending to factored approximate messages,

$$p(\boldsymbol{s}) \approx q(\boldsymbol{s}) = \frac{1}{Z}\prod_{\text{nodes } i}f_i(s_i)\prod_{\text{edges }(ij)}\tilde{g}_{ij}(s_i,s_j)$$

To approximate g_{ij},

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

where $M_{i\to j}(s_j)$ has the natural parameters η_{ij} and $M_{j\to i}(s_i)$ has the natural parameters η_{ji}. The cavity can then be written

$$q_{\neg ij}(s_i,s_j) = f_i(s_i)f_j(s_j)\prod_{k\in\text{ne}(i)\setminus j}M_{k\to i}(s_i)\prod_{l\in\text{ne}(j)\setminus i}M_{l\to j}(s_j)$$

To derive the message passing scheme to find $\tilde{g}$, the KL divergence is minimized:

$\left\{M_{i\to j}^{\text{new}}, M_{j\to i}^{\text{new}}\right\} = \arg\min \mathsf{KL}[g_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j)||\tilde{g}_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j)]$

$= \arg\min \mathsf{KL}[g_{ij}(s_i,s_j)q_{\neg ij}(s_i,s_j)||\alpha_i(s_i)\beta_j(s_j)\ q_{\neg ij}(s_i,s_j)]$

Marginalizing the cavity yields

$$q_{\neg ij}(s_i) = f_i(s_i) \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i)$$

$$q_{\neg ij}(s_j) = f_j(s_j) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j)$$

To solve for $M^{\mathrm{new}}_{j \to i}$, each side is marginalized with respect to s_i

$$g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j) = \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j) q_{\neg ij}(s_i, s_j)$$

$$\Rightarrow M^{\mathrm{new}}_{j \to i} q_{\neg ij}(s_i) = \sum_{s_j} \frac{1}{Z} g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j)$$

$$M^{\mathrm{new}}_{j \to i} q_{\neg ij}(s_i) = \sum_{s_j} \frac{1}{Z} \left(g_{ij}(s_i, s_j) f_j(s_j) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j) \right) f_i(s_i) \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i)$$

$$M^{\mathrm{new}}_{j \to i} = \sum_{s_j} \frac{1}{Z} \left(g_{ij}(s_i, s_j) f_j(s_j) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j) \right)$$

$$M^{\mathrm{new}}_{j \to i} = \sum_{s_j} \frac{1}{Z} \left(\exp \left\{ W_{ij} s_i s_j + b_j s_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} s_j \right\} \right)$$

Because s_j is binary (0,1):

$$M^{\mathrm{new}}_{j \to i} = \frac{1}{Z} \left(\exp\{0\} + \exp \left\{ W_{ij} s_i + b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\} \right)$$

$$M^{\mathrm{new}}_{j \to i} = \frac{1}{Z} \left(1 + \exp \left\{ W_{ij} s_i + b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\} \right)$$

$$Z = 1 + \exp \left\{ b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\}$$

Thus,

$$M^{\mathrm{new}}_{j \to i} = \frac{1 + \exp \left\{ W_{ij} s_i + b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\}}{1 + \exp \left\{ b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\}}$$

$$\eta_{ji} = \log \frac{1 + \exp \left\{ W_{ij} + b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\}}{1 + \exp \left\{ b_j + \sum_{l \in \mathrm{ne}(j) \setminus i} \eta_{lj} \right\}}$$

Similarly for $M^{\mathrm{new}}_{i \to j}$,

$$g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j) = \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j) q_{\neg ij}(s_i, s_j)$$

$$\Rightarrow M^{\mathrm{new}}_{i \to j} q_{\neg ij}(s_j) = \sum_{s_i} \frac{1}{Z} g_{ij}(s_i, s_j) q_{\neg ij}(s_i, s_j)$$

$$M^{\mathrm{new}}_{i \to j} q_{\neg ij}(s_j) = \sum_{s_i} \frac{1}{Z} \left(g_{ij}(s_i, s_j) f_i(s_i) \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i) \right) f_j(s_j) \prod_{l \in \mathrm{ne}(j) \setminus i} M_{l \to j}(s_j)$$

$$M^{\mathrm{new}}_{i \to j} = \sum_{s_i} \frac{1}{Z} \left(g_{ij}(s_i, s_j) f_i(s_i) \prod_{k \in \mathrm{ne}(i) \setminus j} M_{k \to i}(s_i) \right)$$

$$M_{i\to j}^{\text{new}} = \sum_{s_i} \frac{1}{Z}\left(\exp\left\{W_{ij}s_is_j + b_is_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}s_i\right\}\right)$$
$$M_{i\to j}^{\text{new}} = \frac{1}{Z}\left(\exp\{0\} + \exp\left\{W_{ij}s_j + b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}\right)$$
$$M_{i\to j}^{\text{new}} = \frac{1}{Z}\left(1 + \exp\left\{W_{ij}s_j + b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}\right)$$
$$Z = 1 + \exp\left\{b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}$$
$$M_{i\to j}^{\text{new}} = \frac{1 + \exp\left\{W_{ij}s_j + b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}}{1 + \exp\left\{b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}}$$
$$\eta_{ij} = \log\frac{1 + \exp\left\{W_{ij} + b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}}{1 + \exp\left\{b_i + \sum_{k\in\text{ne}(i)\setminus j}\eta_{ki}\right\}}$$

This leads to a loopy BP algorithm because all factors are "connected": rather than existing in a tree, as all s contribute to x at t. This leads to a loopy BP because the posterior marginals are approximate as evidence is over-counted. Message-based EP in the loopy graph can be seen as a constraint on the approximate sites.

d. A Bayesian method that could be used to select K, the number of hidden binary variables using EP, is that of automatic relevance determination (ARD). ARD could be employed by placing a column-wise Gaussian prior on $\boldsymbol{\mu}$: $\boldsymbol{\mu}_{:i} \sim \mathcal{N}(0, \alpha_i^{-1}I)$
Thus, EP could be employed, updating the approximation of $f_i(s_i)$ until convergence. The evidence of $\boldsymbol{\alpha}$ could then be optimized in a hyper-M step. Optimization with respect to the distribution of $\boldsymbol{\alpha}$ would thus control the relevant hidden binary variables, as $\boldsymbol{\alpha}$ that diverged would be considered irrelevant factors. Through this, $\boldsymbol{\alpha}$ selects the relevant hidden variables, effectively learning the dimensionality K of $\boldsymbol{\alpha}$ and thus of $\boldsymbol{s}$. The computational complexity of the algorithm is $\approx O(NK^2)$. Thus, this method may pose computational difficulties, especially if the number of factors K is very large. A possible solution could be to sequentially update α_i by decomposing the evidence $p(\boldsymbol{x}|\boldsymbol{\alpha})$ into $p(\boldsymbol{x}|\boldsymbol{\alpha}_{\neg i})$ and $p(\boldsymbol{x}|\boldsymbol{\alpha})$. Using this "Fast Marginal Likelihood Maximization", only the number of features to yield the least error for $\boldsymbol{\alpha}_{\neg i}$ can be used in the following update which would allow for more efficient computation of the algorithm, as there would only be a small set of K features in the model during the updates.

Question 3.

The EP/loopy-BP algorithm derived in (2) is implemented below, using the same M step as in assignment (5). In `LoopyBP`, `fi` is first computed. Then, loopy BP is used to compute new messages and `lambda` is computed until messages converge or the maximum number of steps is reached. After convergence, the M step is computed. The loopy BP and M step are run for the given number of iterations and `F` is computed at each iteration.

```
import numpy as np
import matplotlib.pyplot as plt

# Computes new message
def compute_message(fi,lambda0,mess,mu,sigma):
    K=lambda0.shape[1]
    a=0.01
    for n in range(lambda0.shape[0]):
        for i in range(K):
            for j in range(i+1,K):
                gij=-(1/(sigma**2))*np.dot(mu[:,i].T,mu[:,j])
                newm=1+np.exp(fi[n,j]+np.sum(mess[:,j,n])-mess[i,j,n]+gij)
                newm=newm/(1+np.exp(fi[n,j]+np.sum(mess[:,j,n])-mess[i,j,n]))
                mess[j,i,n]=a*mess[j,i,n]+(1-a)*np.log(newm)
                #mess[j,i,n]=np.log(newm)

                gij=-(1/(sigma**2))*np.dot(mu[:,j].T,mu[:,i])
                newm=(1+np.exp(fi[n,i]+np.sum(mess[:,i,n])-mess[j,i,n]+gij))
                newm=newm/(1+np.exp(fi[n,i]+np.sum(mess[:,i,n])-mess[j,i,n]))
                mess[i,j,n]=a*mess[i,j,n]+(1-a)*np.log(newm)
                #mess[i,j,n]=np.log(newm)
    return mess

# Updates messages, updates lambda, computes new F;
# halts when messages have converged or maxsteps have been reached
def LoopyBP(X,mu,sigma,pie,lambda0,maxsteps,mess1):
    lambda1=np.copy(lambda0)
    prevlambda=lambda1
    prevF=FreeEnergy(X,mu,sigma,pie,lambda1)
    prevmess=mess1
    fi=np.zeros(np.shape(lambda0))
    for n in range(lambda1.shape[0]):
        fi[n,:]=np.log(pie/(1-pie))+(1/(sigma**2))*np.dot(X[n,:],mu)
            -(1/2*(sigma**2))*np.diag(np.dot(mu.T,mu)).T
    for m in range(maxsteps):
        mess1=compute_message(fi,lambda1,mess1,mu,sigma)
        lambda1=np.zeros(lambda0.shape)
        for n in range(lambda1.shape[0]):
            lambda1[n,:]=1/(1+np.exp(-(fi[n,:]+np.sum(mess1[:,:,n]))))
        F=FreeEnergy(X,mu,sigma,pie,lambda1)
        if np.amax(mess1-prevmess) < 0.0000001:
            break
        prevmess=mess1
    return lambda1,F,mess1

# Computes F
def FreeEnergy(X,mu,sigma,pie,lambda0):
    F=0
    D=lambda0.shape[1]
    for n in range(lambda0.shape[0]):
        tF=X[n]-np.dot(lambda0[n],mu.T)
        tF=np.dot(tF.T,tF)
        tF2=np.sum(np.dot(np.dot(lambda0[n]-lambda0[n]**2,mu.T),mu))
        tF=(-1/(2*sigma**2))*(tF+tF2)-D*np.log(sigma)-(D/2)*np.log(2*np.pi)
        tF2=np.dot(lambda0[n],np.log(pie[0]).T)
            -np.dot(lambda0[n],np.log(lambda0[n]+.0000001).T)
        tF2=tF2+np.sum(np.log(1-pie[0]))-np.sum(np.log(1-lambda0[n]+.0000001))
        tF2=tF2-np.dot(lambda0[n],np.log(1-pie[0]).T)
            + np.dot(lambda0[n],np.log(1-lambda0[n]+.0000001))
        F+=tF+tF2
    return F

# Calculates ESS using ES for m_step
```

```
def compute_ESS(ES):
    N=ES.shape[0]
    K=ES.shape[1]
    ESS=np.zeros((N,K,K))
    for n in range(N):
        tL=np.dot(ES[n].reshape(1,K).T,ES[n].reshape(1,K))
        ind=np.diag_indices(K)
        tL[ind]=ES[n]
        ESS[n]=tL
    return ESS

def init_params(X,K):
    N=X.shape[0]
    D=X.shape[1]
    ES=np.random.rand(N,K)
    ESS=compute_ESS(ES)
    mu,sigma,pie=m_step(X,ES,ESS)
    mess0=np.random.rand(K,K,N)
    for n in range(N):
        mess0[:,:,n]=mess0[:,:,n]-np.diag(np.diag(mess0[:,:,n]))
    return ES,ESS,mu,sigma,pie,mess0

# Initializes parameters and runs EP and M step for given number of iterations;
# calculates F for each iteration
def LearnBinFactors(X,K,iterations):
    maxsteps=10
    ES,ESS,mu,sigma,pie,mess=init_params(X,K)
    F_array=[]
    prevF=FreeEnergy(X,mu,sigma,pie,ES)
    F_array.append(prevF)
    for i in range(iterations):
        ES,F,mess=LoopyBP(X,mu,sigma,pie,ES,maxsteps,mess)
        ESS=compute_ESS(ES)
        mu,sigma,pie=m_step(X,ES,ESS)
        F=FreeEnergy(X,mu,sigma,pie,ES)
        print(F)
        F_array.append(F)
    return mu,sigma,pie,F_array,ES

def m_step(X, ES, ESS):
    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
```

Loopy BP has difficult converging on many runs and thus damping is utilized to encourage convergence. Smaller values for α appear to perform better and thus α is set to 0.01. The free energy for variational mean-field in assignment (5) and for loopy BP are plotted below.

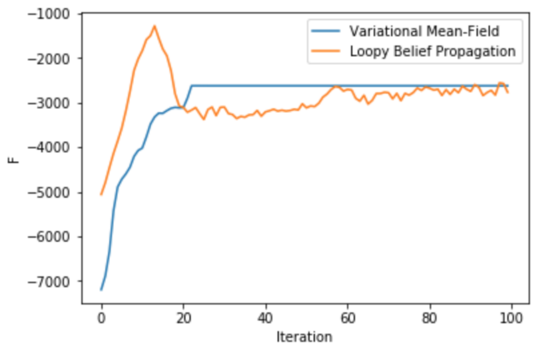

Figure 1: Free Energy

From the figure, it is evident that the free energy of loopy BP fluctuates more than that of variational mean field, having a less steady value of convergence. Additionally, it is possible for free energy to decrease in Loopy BP, unlike regular EM. Notably, both algorithms reach approximately the same value of free energy at convergence.

The values of `mu` are plotted below for both the variational mean-field approximation in assignment (5) and loopy BP.

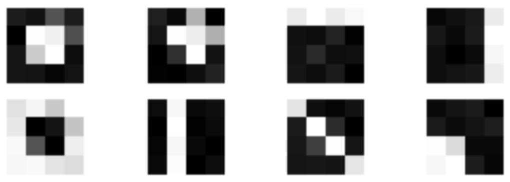

Figure 2: Variational Mean-Field Approximation

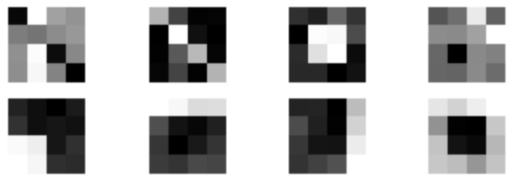

Figure 3: Loopy Belief Propagation

Although both algorithms converge to approximately the same free energy, it appears more obvious that variational mean-field yields more accurate values for `mu` than loopy BP, distinguishing features better. There appears to be more noise in the features predicted by loopy BP and some features are not differentiated.