# Expectation-Maximization (Contd) and Introduction to Variational Inference

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

Feb 11, 2019

# Recap: The Expectation Maximization (EM) Algorithm

Used for doing parameter estimation in latent variable models

$$\Theta_{MLE} = \arg\max_{\Theta} \log p(\mathbf{X}|\Theta) = \arg\max_{\Theta} \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

# Recap: The Expectation Maximization (EM) Algorithm

Used for doing parameter estimation in latent variable models

$$\Theta_{MLE} = \arg\max_{\Theta} \log p(\mathbf{X}|\Theta) = \arg\max_{\Theta} \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

## The EM Algorithm

- Initialize $\Theta$ as $\Theta^{(0)}$, set $t = 1$
- Step 1: Compute conditional posterior of latent vars given current params $\Theta^{(t-1)}$

$$p(\boldsymbol{z}_n^{(t)}|\boldsymbol{x}_n, \Theta^{(t-1)}) = \frac{p(\boldsymbol{z}_n^{(t)}|\Theta^{(t-1)})p(\boldsymbol{x}_n|\boldsymbol{z}_n^{(t)}, \Theta^{(t-1)})}{p(\boldsymbol{x}_n|\Theta^{(t-1)})} \propto \text{prior} \times \text{likelihood}$$

- Step 2: Now maximize the expected complete data log-likelihood w.r.t. $\Theta$

$$\Theta^{(t)} = \arg\max_{\Theta} \mathcal{Q}(\Theta, \Theta^{(t-1)}) = \arg\max_{\Theta} \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{z}_n^{(t)}|\boldsymbol{x}_n, \Theta^{(t-1)})}[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(t)}|\Theta)]$$

- If not yet converged, set $t = t + 1$ and go to Step 1.

# Making EM Faster: Online EM

- Needn't compute $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for every $\boldsymbol{x}_n$ in each EM iteration (computational/storage efficiency)
  - Recall that the expected CLL is often a sum over all data points

$$\mathcal{Q}(\Theta, \Theta^{old}) = \mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{n=1}^{N} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]$$

# Making EM Faster: Online EM

- Needn't compute $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for every $\boldsymbol{x}_n$ in each EM iteration (computational/storage efficiency)
  - Recall that the expected CLL is often a sum over all data points

$$\mathcal{Q}(\Theta, \Theta^{old}) = \mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{n=1}^{N} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]$$

  - Can compute this quantity recursively using small minibatches of data

$$\mathcal{Q}_t = (1-\gamma_t)\mathcal{Q}_{t-1} + \gamma_t \left[\sum_{n=1}^{N_t} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]\right]$$

.. where $\gamma_t = (1+t)^{-\kappa}, 0.5 < \kappa \leq 1$ is a decaying learning rate

# Making EM Faster: Online EM

- Needn't compute $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for every $\boldsymbol{x}_n$ in each EM iteration (computational/storage efficiency)
  - Recall that the expected CLL is often a sum over all data points

$$\mathcal{Q}(\Theta, \Theta^{old}) = \mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{n=1}^{N} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]$$

  - Can compute this quantity recursively using small minibatches of data

$$\mathcal{Q}_t = (1-\gamma_t)\mathcal{Q}_{t-1} + \gamma_t \left[ \sum_{n=1}^{N_t} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)] \right]$$

  .. where $\gamma_t = (1+t)^{-\kappa}, 0.5 < \kappa \leq 1$ is a decaying learning rate

  - Requires computing $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ only for data in current mini-batch (computational/storage efficiency)

# Making EM Faster: Online EM

- Needn't compute $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for every $\boldsymbol{x}_n$ in each EM iteration (computational/storage efficiency)
  - Recall that the expected CLL is often a sum over all data points

$$\mathcal{Q}(\Theta, \Theta^{old}) = \mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{n=1}^{N} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]$$

  - Can compute this quantity recursively using small minibatches of data

$$\mathcal{Q}_t = (1-\gamma_t)\mathcal{Q}_{t-1} + \gamma_t \left[\sum_{n=1}^{N_t} \mathbb{E}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)] + \mathbb{E}[\log p(\boldsymbol{z}_n|\phi)]\right]$$

  .. where $\gamma_t = (1+t)^{-\kappa}, 0.5 < \kappa \leq 1$ is a decaying learning rate
  - Requires computing $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ only for data in current mini-batch (computational/storage efficiency)
  - MLE on above $\mathcal{Q}_t$ can be shown to be equivalent to a simple recursive updates for $\Theta$

$$\Theta^{(t)} = (1-\gamma_t) \times \Theta^{(t-1)} + \gamma_t \times \arg\max_{\Theta} \underbrace{\mathcal{Q}(\Theta, \Theta^{t-1})}_{\text{computed using only the } N_t \text{ examples from this minibatch}}$$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}^{(t)} \boldsymbol{x}_n$$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)}(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top$$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$
\begin{aligned}
\boldsymbol{\mu}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top \\
\pi_k^{(t)} &= \frac{\sum_{n=1}^N \gamma_{nk}^{(t)}}{N}
\end{aligned}
$$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\begin{aligned}
\boldsymbol{\mu}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top \\
\pi_k^{(t)} &= \frac{\sum_{n=1}^N \gamma_{nk}^{(t)}}{N}
\end{aligned}$$

- Each update depends on sum of expected sufficient statistics (ESS)

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\begin{aligned}
\boldsymbol{\mu}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top \\
\pi_k^{(t)} &= \frac{\sum_{n=1}^N \gamma_{nk}^{(t)}}{N}
\end{aligned}$$

- Each update depends on sum of expected sufficient statistics (ESS). For each data point $\boldsymbol{x}_n, \boldsymbol{z}_n$
  - ESS for $\boldsymbol{\mu}_k$ is $\gamma_{nk}^{(t)} \boldsymbol{x}_n$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\begin{aligned}
\boldsymbol{\mu}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k^{(t)} &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top \\
\pi_k^{(t)} &= \frac{\sum_{n=1}^N \gamma_{nk}^{(t)}}{N}
\end{aligned}$$

- Each update depends on sum of expected sufficient statistics (ESS). For each data point $\boldsymbol{x}_n, \boldsymbol{z}_n$
  - ESS for $\boldsymbol{\mu}_k$ is $\gamma_{nk}^{(t)} \boldsymbol{x}_n$; ESS for $\boldsymbol{\Sigma}_k$ is $\gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top$

# How M Step uses Sufficient Statistics

- First recall the **batch EM** algorithm for a $K$ component Gaussian mixture model
  - Cluster id $\boldsymbol{z}_n$ s.t. $z_{nk} = 1$ if $\boldsymbol{x}_n$ belongs to cluster $k$, and zero otherwise
  - The conditional posterior of $z_{nk}$ is $p(z_{nk} = 1|\boldsymbol{x}_n, \Theta) \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Denoting current iteration by $t$, and the expectation computed in E step: $\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)}$
- The M step updates for params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ are

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)}\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}^{(t)}(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top$$

$$\pi_k^{(t)} = \frac{\sum_{n=1}^N \gamma_{nk}^{(t)}}{N}$$

- Each update depends on sum of expected sufficient statistics (ESS). For each data point $\boldsymbol{x}_n, \boldsymbol{z}_n$
  - ESS for $\boldsymbol{\mu}_k$ is $\gamma_{nk}^{(t)}\boldsymbol{x}_n$; ESS for $\boldsymbol{\Sigma}_k$ is $\gamma_{nk}^{(t)}(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top$; ESS for $\pi_k$ is $\gamma_{nk}^{(t)}$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

**Batch EM in terms of ESS**

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n | \boldsymbol{x}_n, \Theta) \phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

$$\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) = \mathbb{E}[\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)]$$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \mathbf{s}_n$ where each ESS $\mathbf{s}_n = \sum_{\mathbf{z}_n} p(\mathbf{z}_n|\mathbf{x}_n,\Theta)\phi(\mathbf{x}_n,\mathbf{z}_n)$
- Here $\phi(\mathbf{x}_n,\mathbf{z}_n)$ is the SS associated with one observation $\mathbf{x}_n$ and its latent variable $\mathbf{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

$$
\begin{aligned}
\mathbf{s}_n &= \sum_{\mathbf{z}_n} p(\mathbf{z}_n|\mathbf{x}_n,\Theta)\phi(\mathbf{x}_n,\mathbf{z}_n) = \mathbb{E}[\phi(\mathbf{x}_n,\mathbf{z}_n)] \\
\mathbf{S}^{new} &= \mathbf{S}^{new} + \mathbf{s}_n
\end{aligned}
$$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

$$
\begin{aligned}
\boldsymbol{s}_n &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) = \mathbb{E}[\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)] \\
\mathbf{S}^{new} &= \mathbf{S}^{new} + \boldsymbol{s}_n
\end{aligned}
$$

  - $\mathbf{S} = \mathbf{S}^{new}$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

$$\begin{aligned} \boldsymbol{s}_n &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) = \mathbb{E}[\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)] \\ \mathbf{S}^{new} &= \mathbf{S}^{new} + \boldsymbol{s}_n \end{aligned}$$

  - $\mathbf{S} = \mathbf{S}^{new}$
  - Recompute parameters $\Theta = f(\mathbf{S})$

# Batch EM Algorithm in terms of Sufficient Statistics

- Denote the sum of ESS as $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$ where each ESS $\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$
- Here $\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is the SS associated with one observation $\boldsymbol{x}_n$ and its latent variable $\boldsymbol{z}_n$
- M step updates of $\Theta$ are like computing a function of $\mathbf{S}$, i.e., $\Theta = f(\mathbf{S})$

## Batch EM in terms of ESS

- Initialize $\mathbf{S}$ and compute parameters $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - $\mathbf{S}^{new} = 0$ (fresh sum of ESS; will be computed in this iteration)
  - For $n = 1 : N$

$$\begin{aligned} \boldsymbol{s}_n &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) = \mathbb{E}[\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)] \\ \mathbf{S}^{new} &= \mathbf{S}^{new} + \boldsymbol{s}_n \end{aligned}$$

  - $\mathbf{S} = \mathbf{S}^{new}$
  - Recompute parameters $\Theta = f(\mathbf{S})$

- Note: In general, there may be more than one sum of ESS (one for each parameter update)

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update **S**

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update **S**
- Can be done in one of the two manners (Liang and Klein, 2009)

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update **S**
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update **S**
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Set "learning rate" $\gamma_t$, pick a random example $n$ and compute its sufficient statistics

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Set "learning rate" $\gamma_t$, pick a random example $n$ and compute its sufficient statistics

$$\boldsymbol{s}_n = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Set "learning rate" $\gamma_t$, pick a random example $n$ and compute its sufficient statistics

$$\begin{aligned} \boldsymbol{s}_n &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n | \boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\ \mathbf{S} &= (1-\gamma_t)\mathbf{S} + \gamma_t \boldsymbol{s}_n \end{aligned}$$

# Online EM Algorithm in terms of Sufficient Statistics

- Works in a similar way as batch EM except we need an online way to update $\mathbf{S}$
- Can be done in one of the two manners (Liang and Klein, 2009)
  - Stepwise EM (based on recursively updating the sum of ESS)
  - Incremental EM (based on deleting old and adding new ESS of each data point)

## Online EM as Stepwise EM

- Initialize the sum of ESS $\mathbf{S}$ and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Set "learning rate" $\gamma_t$, pick a random example $n$ and compute its sufficient statistics

$$\begin{aligned} \boldsymbol{s}_n &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\ \mathbf{S} &= (1-\gamma_t)\mathbf{S} + \gamma_t \boldsymbol{s}_n \end{aligned}$$

  - Recompute $\Theta = f(\mathbf{S})$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

### Online EM as Incremental EM

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \dots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$\boldsymbol{s}_n^{new} = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n | \boldsymbol{x}_n, \Theta) \phi(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$\begin{aligned} \boldsymbol{s}_n^{new} &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\ \mathbf{S} &= \mathbf{S} + \boldsymbol{s}_n^{new} - \boldsymbol{s}_n \end{aligned}$$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$
\begin{aligned}
\boldsymbol{s}_n^{new} &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n | \boldsymbol{x}_n, \Theta) \phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\
\mathbf{S} &= \mathbf{S} + \boldsymbol{s}_n^{new} - \boldsymbol{s}_n \\
\boldsymbol{s}_n &= \boldsymbol{s}_n^{new}
\end{aligned}
$$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^N \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$
\begin{aligned}
\boldsymbol{s}_n^{new} &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\
\mathbf{S} &= \mathbf{S} + \boldsymbol{s}_n^{new} - \boldsymbol{s}_n \\
\boldsymbol{s}_n &= \boldsymbol{s}_n^{new}
\end{aligned}
$$

  - Recompute $\Theta = f(\mathbf{S})$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$\begin{aligned} \boldsymbol{s}_n^{new} &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)\phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\ \mathbf{S} &= \mathbf{S} + \boldsymbol{s}_n^{new} - \boldsymbol{s}_n \\ \boldsymbol{s}_n &= \boldsymbol{s}_n^{new} \end{aligned}$$

  - Recompute $\Theta = f(\mathbf{S})$

- However, incremental EM requires keeping a track of sum of ESS $\mathbf{S}$ as well as each $\boldsymbol{s}_n$

# Online EM Algorithm in terms of Sufficient Statistics

- The other Online EM approach "Incremental EM" needs no learning rate (unlike "Stepwise EM")

## Online EM as Incremental EM

- Initialize each ESS $\boldsymbol{s}_n$, $n = 1, \ldots, N$, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{s}_n$, and compute $\Theta = f(\mathbf{S})$
- For $t = 1 : T$ (or until convergence)
  - Pick a random example $n$ and update its exp. sufficient statistics

$$
\begin{aligned}
\boldsymbol{s}_n^{new} &= \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n | \boldsymbol{x}_n, \Theta) \phi(\boldsymbol{x}_n, \boldsymbol{z}_n) \\
\mathbf{S} &= \mathbf{S} + \boldsymbol{s}_n^{new} - \boldsymbol{s}_n \\
\boldsymbol{s}_n &= \boldsymbol{s}_n^{new}
\end{aligned}
$$

  - Recompute $\Theta = f(\mathbf{S})$

- However, incremental EM requires keeping a track of sum of ESS $\mathbf{S}$ as well as each $\boldsymbol{s}_n$
- In practice, stepwise EM outperforms batch EM as well as incremental EM on many problems (can refer to Liang and Klein, 2009 for some examples of models where these algos were tried)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to the form of updates

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables **Z**, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to the form of updates
  - In some cases†, EM usually converges faster (and often like second-order methods like Newton's)

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to the form of updates
  - In some cases[†], EM usually converges faster (and often like second-order methods like Newton's)
    - Example: Mixture of Gaussians with when the data is reasonably well-clustered

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to the form of updates
  - In some cases†, EM usually converges faster (and often like second-order methods like Newton's)
    - Example: Mixture of Gaussians with when the data is reasonably well-clustered
  - EM applies even when the explicit summing over is expensive or integrating out isn't tractable

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# EM vs Gradient-based Methods

- Can also estimate params use gradient-based optimization (or backprop in general) instead of EM
  - Reason: We can usually explicitly sum over or integrate out the latent variables $\mathbf{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal $\Theta$
- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters $\Theta$
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to the form of updates
  - In some cases[†], EM usually converges faster (and often like second-order methods like Newton's)
    - Example: Mixture of Gaussians with when the data is reasonably well-clustered
  - EM applies even when the explicit summing over is expensive or integrating out isn't tractable
  - EM also provides the conditional posterior over the latent variables $\mathbf{Z}$ (from E step)

† Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

# Variational Bayes (VB)
# a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

# Variational Bayes (VB) a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

- Origins of VB/VI were in Statistical Physics (mainly "mean-field" methods; early 80s)

# Variational Bayes (VB)
# a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

- Origins of VB/VI were in Statistical Physics (mainly "mean-field" methods; early 80s)
- Some of the early applications of VB/VI were for neural networks (late 80s)

# Variational Bayes (VB)
# a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

- Origins of VB/VI were in Statistical Physics (mainly "mean-field" methods; early 80s)
- Some of the early applications of VB/VI were for neural networks (late 80s)
- Became very popular in ML community in late 90s (and continues to remain so)

# Variational Bayes (VB)
# a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

- Origins of VB/VI were in Statistical Physics (mainly "mean-field" methods; early 80s)
- Some of the early applications of VB/VI were for neural networks (late 80s)
- Became very popular in ML community in late 90s (and continues to remain so)
    - Primary reason: Faster than MCMC methods

# Variational Bayes (VB) a.k.a. Variational Inference (VI)

(Note: "variational" here refers to optimization of functions of distributions)

- Origins of VB/VI were in Statistical Physics (mainly "mean-field" methods; early 80s)
- Some of the early applications of VB/VI were for neural networks (late 80s)
- Became very popular in ML community in late 90s (and continues to remain so)
  - Primary reason: Faster than MCMC methods
- An aside: Statistics researchers were somewhat skeptical of VB/VI (but that is changing now) and continued their allegiance towards MCMC methods for approximate posterior inference

# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$

# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$
- Suppose $p(\mathbf{Z}|\mathbf{X})$ is intractable. VB/VI approximates it using a distribution $q(\mathbf{Z}|\phi)$ or $q_\phi(\mathbf{Z})$

# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$
- Suppose $p(\mathbf{Z}|\mathbf{X})$ is intractable. VB/VI approximates it using a distribution $q(\mathbf{Z}|\phi)$ or $q_\phi(\mathbf{Z})$
- VB/VI finds the $q(\mathbf{Z}|\phi)$ that is "closest" to $p(\mathbf{Z}|\mathbf{X})$ by finding the "optimal" value of $\phi$

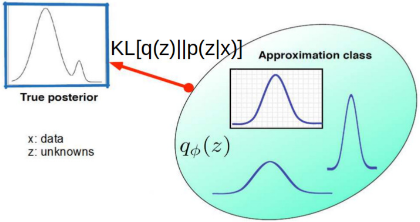
$$\phi^* = \arg\min_{\phi} \text{KL}[q_\phi(\mathbf{Z})||p(\mathbf{Z}|\mathbf{X})]$$

# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$
- Suppose $p(\mathbf{Z}|\mathbf{X})$ is intractable. VB/VI approximates it using a distribution $q(\mathbf{Z}|\phi)$ or $q_\phi(\mathbf{Z})$
- VB/VI finds the $q(\mathbf{Z}|\phi)$ that is "closest" to $p(\mathbf{Z}|\mathbf{X})$ by finding the "optimal" value of $\phi$

$$\phi^* = \arg\min_\phi \text{KL}[q_\phi(\mathbf{Z})||p(\mathbf{Z}|\mathbf{X})]$$

- This amounts of finding the best distribution from a class of distributions parametrized by $\phi$

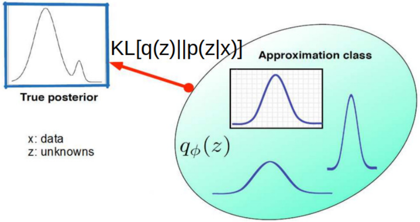


# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$
- Suppose $p(\mathbf{Z}|\mathbf{X})$ is intractable. VB/VI approximates it using a distribution $q(\mathbf{Z}|\phi)$ or $q_\phi(\mathbf{Z})$
- VB/VI finds the $q(\mathbf{Z}|\phi)$ that is "closest" to $p(\mathbf{Z}|\mathbf{X})$ by finding the "optimal" value of $\phi$

$$\phi^* = \arg\min_\phi \mathrm{KL}[q_\phi(\mathbf{Z})||p(\mathbf{Z}|\mathbf{X})]$$

- This amounts of finding the best distribution from a class of distributions parametrized by $\phi$



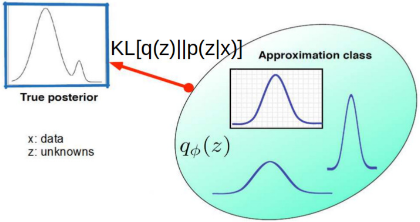

- VB/VI refers to the free parameters $\phi$ as variational parameters (w.r.t. which we optimize)

# Variational Bayes (VB) or Variational Inference (VI)

- Consider a model with data $\mathbf{X}$ and unknowns $\mathbf{Z}$. Goal: Compute the posterior $p(\mathbf{Z}|\mathbf{X})$
- Suppose $p(\mathbf{Z}|\mathbf{X})$ is intractable. VB/VI approximates it using a distribution $q(\mathbf{Z}|\phi)$ or $q_\phi(\mathbf{Z})$
- VB/VI finds the $q(\mathbf{Z}|\phi)$ that is "closest" to $p(\mathbf{Z}|\mathbf{X})$ by finding the "optimal" value of $\phi$

$$\phi^* = \arg\min_{\phi} \text{KL}[q_\phi(\mathbf{Z})||p(\mathbf{Z}|\mathbf{X})]$$

- This amounts of finding the best distribution from a class of distributions parametrized by $\phi$




- VB/VI refers to the free parameters $\phi$ as variational parameters (w.r.t. which we optimize)
- **But wait!** If $p(\mathbf{Z}|\mathbf{X})$ itself is intractable, can we (easily) solve the above KL minimization problem?

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned}\mathcal{L}(q) &= \int q(\mathbf{Z}) \log\left[\frac{p(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log\left[\frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})}\right] d\mathbf{Z}\end{aligned}$$

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned}\mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})}\right] d\mathbf{Z}\end{aligned}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned}\mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})}\right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})}\right] d\mathbf{Z}\end{aligned}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction
- We would like to infer the posterior for all the unknowns (denoted collectively as $\mathbf{Z}$)

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned} \mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right] d\mathbf{Z} \end{aligned}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction
- We would like to infer the posterior for all the unknowns (denoted collectively as $\mathbf{Z}$)
- Since $\log p(\mathbf{X})$ is a constant w.r.t. $\mathbf{Z}$, the following must hold

$$\color{blue}{\arg\min_q \text{KL}(q||p)} = \color{red}{\arg\max_q \mathcal{L}(q)}$$

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned} \mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right] d\mathbf{Z} \end{aligned}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction
- We would like to infer the posterior for all the unknowns (denoted collectively as $\mathbf{Z}$)
- Since $\log p(\mathbf{X})$ is a constant w.r.t. $\mathbf{Z}$, the following must hold

$$\arg\min_q \text{KL}(q||p) = \arg\max_q \mathcal{L}(q)$$

- Since $\text{KL}(q||p) \geq 0$, $\log p(\mathbf{X}) \geq \mathcal{L}(q)$

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\begin{aligned}\mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right] d\mathbf{Z} \\ \text{KL}(q||p) &= -\int q(\mathbf{Z}) \log \left[\frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})}\right] d\mathbf{Z}\end{aligned}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction
- We would like to infer the posterior for all the unknowns (denoted collectively as $\mathbf{Z}$)
- Since $\log p(\mathbf{X})$ is a constant w.r.t. $\mathbf{Z}$, the following must hold

$$\arg\min_q \text{KL}(q||p) = \arg\max_q \mathcal{L}(q)$$

- Since $\text{KL}(q||p) \geq 0$, $\log p(\mathbf{X}) \geq \mathcal{L}(q)$
- $\mathcal{L}(q)$ is also known as the **Evidence Lower Bound (ELBO)**

# Variational Bayes (VB) or Variational Inference (VI)

- The following holds for any $q$: $\log p(\mathbf{X}|m) = \mathcal{L}(q) + \text{KL}(q||p)$ where

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

$$\text{KL}(q||p) = -\int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Above is similar to what we had in EM, but now no $\Theta$ (param) vs $\mathbf{Z}$ (latent var) distinction
- We would like to infer the posterior for all the unknowns (denoted collectively as $\mathbf{Z}$)
- Since $\log p(\mathbf{X})$ is a constant w.r.t. $\mathbf{Z}$, the following must hold

$$\arg\min_q \text{KL}(q||p) = \arg\max_q \mathcal{L}(q)$$

- Since $\text{KL}(q||p) \geq 0$, $\log p(\mathbf{X}) \geq \mathcal{L}(q)$
- $\mathcal{L}(q)$ is also known as the **Evidence Lower Bound (ELBO)**
    - Reason for the name "ELBO": $\log p(\mathbf{X})$ or $\log p(\mathbf{X}|m)$ is the log-evidence of model $m$

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})]$$

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior
- Maximizing $\mathcal{L}(q)$ w.r.t. $q$ can still be hard in general (note the expectation w.r.t. $q$)

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \mathrm{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior
- Maximizing $\mathcal{L}(q)$ w.r.t. $q$ can still be hard in general (note the expectation w.r.t. $q$)
- Some of the ways to make this problem easier

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior
- Maximizing $\mathcal{L}(q)$ w.r.t. $q$ can still be hard in general (note the expectation w.r.t. $q$)
- Some of the ways to make this problem easier
  1. Restricting the form of our approximation $q(\mathbf{Z})$, e.g., mean-field VB (today's discussion)

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior
- Maximizing $\mathcal{L}(q)$ w.r.t. $q$ can still be hard in general (note the expectation w.r.t. $q$)
- Some of the ways to make this problem easier
  1. Restricting the form of our approximation $q(\mathbf{Z})$, e.g., mean-field VB (today's discussion)
  2. Using Monte-Carlo approximation of the expectation/gradient of the ELBO (later)

# VB/VI = Maximizing the ELBO

- Notation: $q(\mathbf{Z})$, $q(\mathbf{Z}|\phi)$, $q_\phi(\mathbf{Z})$, all will refer to the same thing
- VB/VI finds an approximating distribution $q(\mathbf{Z})$ that maximizes the ELBO

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

- Since $q(\mathbf{Z})$ depends on $\phi$, the ELBO is essentially a function of $\phi$

$$\mathcal{L}(q) = \mathcal{L}(\phi) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})] = \mathbb{E}_q[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Z})||p(\mathbf{Z}))$$

- Makes sense: Maximizing $\mathcal{L}(q)$ will give a $q$ that explains data well and is close to the prior
- Maximizing $\mathcal{L}(q)$ w.r.t. $q$ can still be hard in general (note the expectation w.r.t. $q$)
- Some of the ways to make this problem easier
  1. Restricting the form of our approximation $q(\mathbf{Z})$, e.g., mean-field VB (today's discussion)
  2. Using Monte-Carlo approximation of the expectation/gradient of the ELBO (later)
- For locally conjugate models VB/VI is particularly easy to derive

# Mean-Field VB

- One of the simplest ways of doing VB

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$
- Assume our approximation $q(\mathbf{Z})$ factorizes over these groups

$$q(\mathbf{Z}|\phi) = \prod_{i=1}^{M} q(\mathbf{Z}_i|\phi_i)$$

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$
- Assume our approximation $q(\mathbf{Z})$ factorizes over these groups

$$q(\mathbf{Z}|\phi) = \prod_{i=1}^{M} q(\mathbf{Z}_i|\phi_i)$$

- As a short-hand, sometimes we write $q = \prod_{i=1}^{M} q_i$ where $q_i = q(\mathbf{Z}_i|\phi_i)$

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$
- Assume our approximation $q(\mathbf{Z})$ factorizes over these groups

$$q(\mathbf{Z}|\phi) = \prod_{i=1}^{M} q(\mathbf{Z}_i|\phi_i)$$

- As a short-hand, sometimes we write $q = \prod_{i=1}^{M} q_i$ where $q_i = q(\mathbf{Z}_i|\phi_i)$
- In mean-field VB, learning the optimal $q$ reduces to learning the optimal $q_1, \ldots, q_M$

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$
- Assume our approximation $q(\mathbf{Z})$ factorizes over these groups

$$q(\mathbf{Z}|\phi) = \prod_{i=1}^{M} q(\mathbf{Z}_i|\phi_i)$$

- As a short-hand, sometimes we write $q = \prod_{i=1}^{M} q_i$ where $q_i = q(\mathbf{Z}_i|\phi_i)$
- In mean-field VB, learning the optimal $q$ reduces to learning the optimal $q_1, \ldots, q_M$
- The groups are usually chosen based on the model's structure, e.g., in Bayesian linear regression

$$q(\mathbf{Z}|\phi) = q(\boldsymbol{w}, \lambda, \beta|\phi) = q(\boldsymbol{w}|\phi_w)q(\lambda|\phi_\lambda)q(\beta|\phi_\beta)$$

# Mean-Field VB

- One of the simplest ways of doing VB
- In mean-field VB, we define a partition of the latent variables $\mathbf{Z}$ into $M$ groups $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$
- Assume our approximation $q(\mathbf{Z})$ factorizes over these groups

$$q(\mathbf{Z}|\phi) = \prod_{i=1}^{M} q(\mathbf{Z}_i|\phi_i)$$

- As a short-hand, sometimes we write $q = \prod_{i=1}^{M} q_i$ where $q_i = q(\mathbf{Z}_i|\phi_i)$
- In mean-field VB, learning the optimal $q$ reduces to learning the optimal $q_1, \ldots, q_M$
- The groups are usually chosen based on the model's structure, e.g., in Bayesian linear regression

$$q(\mathbf{Z}|\phi) = q(\boldsymbol{w}, \lambda, \beta|\phi) = q(\boldsymbol{w}|\phi_w)q(\lambda|\phi_\lambda)q(\beta|\phi_\beta)$$

- Note: Mean-field is quite a strong assumption (can destroy structure among latent variables)

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z}$$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X}, \mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X}, \mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

- Suppose we wish to find the optimal $q_j$ given all other $q_i$ $(i \neq j)$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X}, \mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

- Suppose we wish to find the optimal $q_j$ given all other $q_i$ $(i \neq j)$. Let's re-express $\mathcal{L}(q)$ as

$$\mathcal{L}(q) \quad = \quad \int q_j \left[ \int \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right] d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j \quad + \quad \text{consts w.r.t. } q_j$$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X}, \mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

- Suppose we wish to find the optimal $q_j$ given all other $q_i$ $(i \neq j)$. Let's re-express $\mathcal{L}(q)$ as

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \left[ \int \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right] d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j \quad + \quad \text{consts w.r.t. } q_j \\ &= \int q_j \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j\end{aligned}$$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_i$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X},\mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

- Suppose we wish to find the optimal $q_j$ given all other $q_i$ $(i \neq j)$. Let's re-express $\mathcal{L}(q)$ as

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \left[ \int \log p(\mathbf{X},\mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right] d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j \quad + \quad \text{consts w.r.t. } q_j \\ &= \int q_j \log \tilde{p}(\mathbf{X},\mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j\end{aligned}$$

where $\log \tilde{p}(\mathbf{X},\mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\log p(\mathbf{X},\mathbf{Z})] + \text{const}$

# Deriving Mean-Field VB Updates

- With $q = \prod_{i=1}^{M} q_j$, what's each optimal $q_i$ equal to when we do $\arg\max_q \mathcal{L}(q)$?
- Note that under this mean-field assumption, the ELBO simplifies to

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left[ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right] d\mathbf{Z} = \int \prod_i q_i \left[ \log p(\mathbf{X}, \mathbf{Z}) - \sum_i \log q_i \right] d\mathbf{Z}$$

- Suppose we wish to find the optimal $q_j$ given all other $q_i$ $(i \neq j)$. Let's re-express $\mathcal{L}(q)$ as

$$\begin{aligned} \mathcal{L}(q) &= \int q_j \left[ \int \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right] d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j \quad + \quad \text{consts w.r.t. } q_j \\ &= \int q_j \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \log q_j d\mathbf{Z}_j \end{aligned}$$

where $\log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})] + \text{const}$

- Note that $\mathcal{L}(q) = -KL(q_j || \tilde{p}) + \text{const}$. Which $q_j$ will maximize it?

$$\boxed{q_j = \tilde{p}(\mathbf{X}, \mathbf{Z}_j)}$$

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i\neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$
- Important: For estimating $q_j$, the required expectation depends on other $\{q_i\}_{i \neq j}$

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i\neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$
- Important: For estimating $q_j$, the required expectation depends on other $\{q_i\}_{i\neq j}$
- Thus we need to cycle through updating each $q_j$ in turn (similar to co-ordinate ascent, alternating optimization, Gibbs sampling, etc.)

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$
- Important: For estimating $q_j$, the required expectation depends on other $\{q_i\}_{i \neq j}$
- Thus we need to cycle through updating each $q_j$ in turn (similar to co-ordinate ascent, alternating optimization, Gibbs sampling, etc.)
- Guaranteed to converge (to a local optima)

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$
- Important: For estimating $q_j$, the required expectation depends on other $\{q_i\}_{i \neq j}$
- Thus we need to cycle through updating each $q_j$ in turn (similar to co-ordinate ascent, alternating optimization, Gibbs sampling, etc.)
- Guaranteed to converge (to a local optima)
  - We are basically solving a sequence of concave maximization problems

# Deriving Mean-Field VB Updates

- Since $\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}$, we have

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j} \qquad \forall j$$

- Note: Only need to compute the numerator. Denominator can usually be recognized by inspection
- For locally-conjugate models, $q_j^*(\mathbf{Z}_j)$ will have the same form as the prior $p(\mathbf{Z}_j)$
- Important: For estimating $q_j$, the required expectation depends on other $\{q_i\}_{i \neq j}$
- Thus we need to cycle through updating each $q_j$ in turn (similar to co-ordinate ascent, alternating optimization, Gibbs sampling, etc.)
- Guaranteed to converge (to a local optima)
  - We are basically solving a sequence of concave maximization problems
  - Reason: $\mathcal{L}(q) = \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) \mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const}$ is concave w.r.t. each $q_j$

# The Mean-Field VB Algorithm

- Also known as Co-ordinate Ascent Variational Inference (CAVI) Algorithm
- Input: Model $p(\mathbf{X}, \mathbf{Z})$, Data $\mathbf{X}$
- Output: A variational distribution $q(\mathbf{Z}) = \prod_{j=1}^{M} q_j(\mathbf{Z}_j)$
- Initialize: Variational distributions $q_j(\mathbf{Z}_j)$, $j = 1, \ldots, M$
- While the ELBO has not converged
  - For each $j = 1, \ldots, M$, set

$$q_j(\mathbf{Z}_j) \propto \exp(\mathbb{E}_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})])$$

  - Compute ELBO $\mathcal{L}(q) = \mathbb{E}_q[\log p(\mathbf{X}, \mathbf{Z})] - \mathbb{E}_q[\log q(\mathbf{Z})]$

# Next Class

- Continue the discussion on mean-field VI
- Some examples of mean-field VI
- Mean-field VI for models with exponential family distributions
- Some properties of VI
- More general forms of VI (modern VI methods)