VARIATIONAL INFERENCE : FOUNDATIONS & APPLICATIONS

Albert Zhao, John Kim

CS 159

Overview

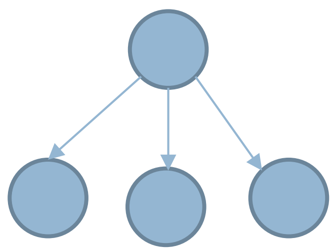
- Foundations
  - Probablistic Pipeline
  - Brief History
  - Problem with exact inference
  - Evidence lower bound
  - Simple model example

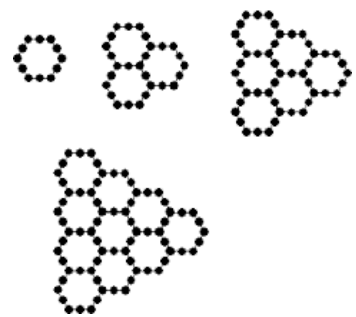
- Applications
  - Crowd Clustering
  - Variational Autoencoders

Foundations

The Probablistic Pipeline

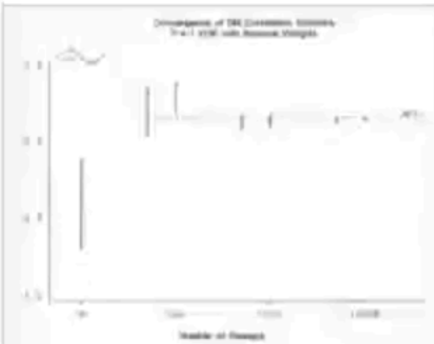
Problem/Question Model Selection Find Patterns Inference & Explore



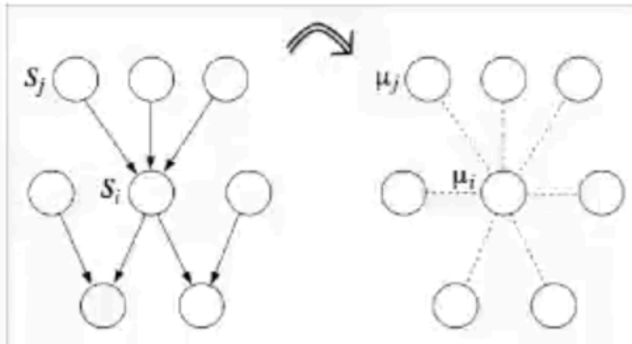


- Inference is the main algorithmic problem
- What does this model say about this data?
- Goal: Find Scalable and General inference algorithms
- **Variational Inference** is one solution to approximate tractable inference

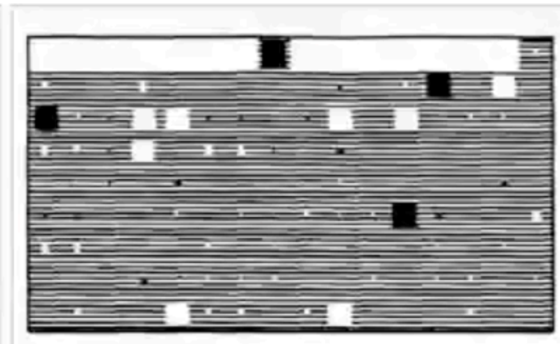
Brief History of Variational Inference



[Peterson and Anderson 1987]



[Jordan et al. 1999]



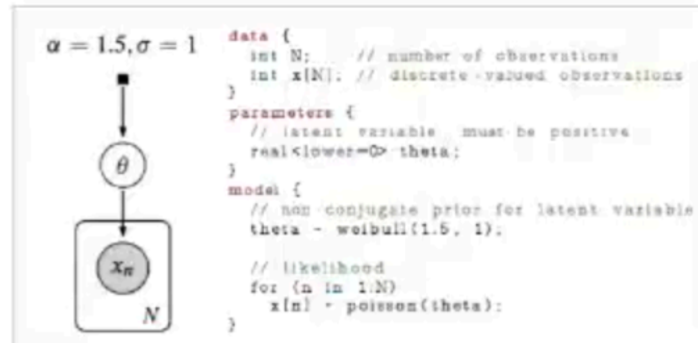
[Hinton and van Camp 1993]

- Variational Inference adapted its ideas from statistical physics.
- Concepts first emerged in late 80s with Peterson and Anderson (1987) who used mean-field methods to fit a neural-network
- Hinton and Van Camp (1993) furthered mean-field methods for neural networks.
- Michael Jordan's lab at MIT generalized V.I. to many probablistic models (Jordan et al., 1999)

Recent applications

[Kingma and Welling 2013]

[Rezende et al. 2014]



[Kucukelbir et al. 2015]

- V. I. has become more scalable and easy to derive (even automated in some cases)
- Variational inference has been extended to probablistic programming, reinforcement learning, and etc.
- Today we'll introduce the basic pipeline of V.I.

Problems with exact inference

- Suppose we have some posterior distribution that we would like to compute by Bayesian inference:

$$p^*(\mathbf{x}) \triangleq p(\mathbf{x} \mid D) = \frac{1}{Z} p(D|\mathbf{x}) p(\mathbf{x})$$

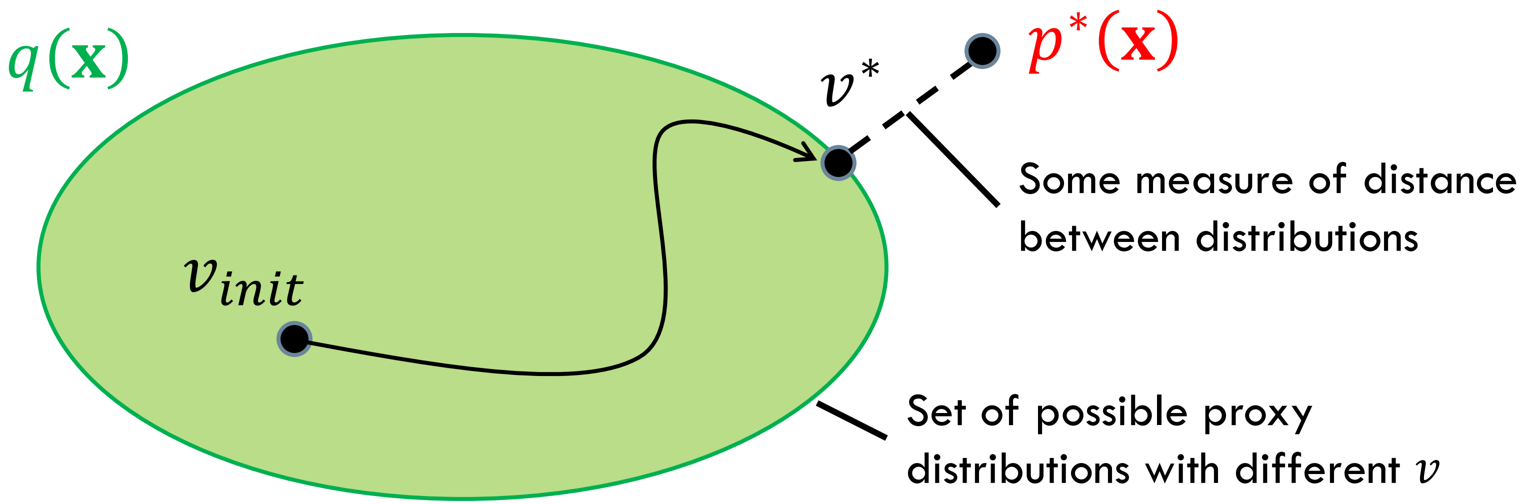
Latent variable

Observed Data

Called the "evidence"
Often intractable to compute

- We want to approximate the true posterior with some close proxy distribution q

Variational Inference in a picture



- Variational Inference converts the inference problem into an optimization problem
- User defines a family of proxy distributions $q(\mathbf{x}; v)$
- Optimize the variational parameters v to bring $q(\mathbf{x})$ as "close" to $p^*(\mathbf{x})$ as possible

Theory: KL divergence

- Measure of distance between distributions
- Forward KL divergence

$$\mathbb{KL}\left(p^*||q\right) = \sum_{\mathbf{x}} p^*\left(\mathbf{x}\right) \log \frac{p^*\left(\mathbf{x}\right)}{q(\mathbf{x})}$$

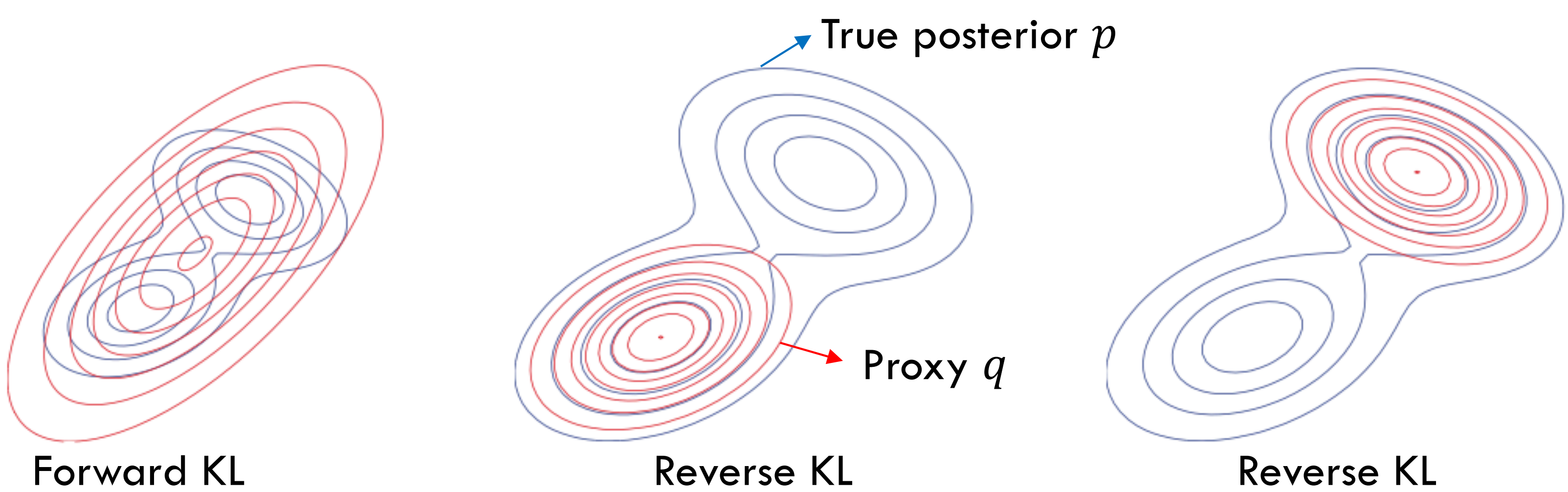
- Reverse KL divergence

$$\mathbb{KL}\left(q||p^*\right) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*\left(\mathbf{x}\right)}$$

- *Note that KL(a||b) divergence is 0 iff a = b

Theory: KL divergence

- Minimizing reverse KL pushes q to underestimates the support of p
- Minimizing forward KL pushes q to overestimate the support of p
- Open times we want to accurately estimate a single mode of the true posterior – Minimize reverse KL
- Minimizing forward KL is referred to as expectation propagation



Kevin Murphy, “Machine Learning: A Probablistic Perspective”, Chp 21, pp. 734

Theory: Modifying reverse KL

- Normalized posterior in reverse KL is intractable to compute:

$$\mathbb{KL}\left(q||p^*\right) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*(\mathbf{x})}$$

- The un-normalized posterior is often tractable to compute:

$$\tilde{p}(\mathbf{x}) \triangleq \boxed{p(\mathbf{x}, \mathcal{D})} = p^*(\mathbf{x})Z$$

- A modified KL divergence objective is then:

$$J(q) \triangleq \boxed{\mathbb{KL}\left(q||\tilde{p}\right)}$$ Computable

Theory: Formulating J(q)

- Following the definition of KL divergence:

$$
\begin{aligned}
J(q) &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\tilde{p}(\mathbf{x})} \\
&= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{Z p^*(\mathbf{x})} \\
&= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*(\mathbf{x})} - \log Z \\
&= \mathbb{KL}\left(q \| p^*\right) - \underbrace{\boxed{\log Z}}_{\text{Constant w.r.t } q}
\end{aligned}
$$

Theory: Interpreting J(q)

- Since KL divergence is strictly non-negative:

$$J(q) = \underbrace{\mathbb{KL}\left(q||p^*\right)}_{\text{Positive}} - \log Z \geq -\log Z = -\underbrace{\log p(\mathcal{D})}_{\text{Log likelihood of data}}$$

- A couple observations:
  - Minimizing J(q) is equivalent to minimizing KL(q||p*)
  - -J(q) lower bounds the log-likelihood of the dataset
  - Hence our objective is to minimize J(q)

Theory: Evidence Lower Bound

- Alternatively, we can maximize the additive inverse:

$$\boxed{L(q)} \triangleq -J(q) = -\mathbb{KL}\left(q||p^*\right) + \log Z \leq \log Z = \log p(\mathcal{D})$$

Variational Lower Bound
Evidence Lower Bound (ELBO)

- We will further discuss how to maximize the ELBO

Theory: Evidence Lower Bound

- We may formulate J(q) in various ways:

$$
\begin{aligned}
J(q) &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\tilde{p}(\mathbf{x})} \\
&= \mathbb{E}_q \left[\log q(\mathbf{x})\right] + \mathbb{E}_q \left[-\log \tilde{p}(\mathbf{x})\right] \\
&= -\underbrace{\boxed{\mathbb{H}\left(q\right)}}_{\text{Entropy}} + \underbrace{\boxed{\mathbb{E}_q \left[E(\mathbf{x})\right]}}_{\text{Expected energy}}
\end{aligned}
$$

- First term wants q to be more diffuse (sort of regularization)
- Second term wants q to place its mass on the MAP estimate
- ELBO is not convex! (optimizing converges to local optimum)

Theory: Evidence Lower Bound

- We may formulate J(q) in various ways:

$$
\begin{aligned}
J(q) &= \mathbb{E}_q\left[\log q(\mathbf{x}) - \log p(\mathbf{x})p(\mathcal{D}|\mathbf{x})\right] \\
&= \mathbb{E}_q\left[\log q(\mathbf{x}) - \log p(\mathbf{x}) - \log p(\mathcal{D}|\mathbf{x})\right] \\
&= \boxed{\mathbb{E}_q\left[-\log p(\mathcal{D}|\mathbf{x})\right]} + \boxed{\mathbb{KL}\left(q(\mathbf{x})||p(\mathbf{x})\right)}
\end{aligned}
$$

Expected Negative Log-Likelihood

Distance between q and exact prior

- So how do we maximize the ELBO?

$$L(q) \triangleq -J(q) = -\mathbb{KL}\left(q||p^*\right) + \log Z \leq \log Z = \log p(\mathcal{D})$$

Theory: Mean Field Method

- Let's consider a popular form of variational inference called Mean Field approximation. (Opper and Saad 2001)
- Assume the proxy q fully factorizes.

$$q(\mathbf{x}) = \prod_i \underbrace{q_i(\mathbf{x}_i)}_{\text{One proxy distribution per dimension}}$$

- Recall the ELBO

$$L(q) \triangleq -J(q) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{\tilde{p}(\mathbf{x})}{q(\mathbf{x})} \leq \log p(\mathcal{D})$$

Theory: Mean Field Method

- Let's single out terms involving one factorized q_j

$$
\begin{aligned}
L(q_j) &= \sum_{\mathbf{x}} \prod_i q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] \\
&= \sum_{\mathbf{x}_j} \sum_{\mathbf{x}_{-j}} q_j(\mathbf{x}_j) \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] \\
&= \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) \\
&\quad - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\sum_{k \neq j} \log q_k(\mathbf{x}_k) + q_j(\mathbf{x}_j) \right] \\
&= \underbrace{\sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log f_j(\mathbf{x}_j) - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log q_j(\mathbf{x}_j)}_{\text{Terms including } q_j} + \text{const}
\end{aligned}
$$

Notation: excluding q_j

Theory: Mean Field Method

- From the previous result:

$$L(q_j) = \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log f_j(\mathbf{x}_j) - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log q_j(\mathbf{x}_j) + \text{const}$$

- The resulting equation can be simplified as:

$$L(q_j) = -\mathbb{KL}\left(q_j || f_j\right)$$

$$\log f_j(\mathbf{x}_j) \triangleq \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) = \mathbb{E}_{-q_j}\left[\log \tilde{p}(\mathbf{x})\right]$$

Theory: Mean Field Method

- Recall KL(a||b) = 0 iff a = b
- Hence, we may maximize $L(q_j)$ by setting $q_j = f_j$

$$q_j(\mathbf{x}_j) = \underline{\frac{1}{Z_j}} \exp\left(\mathbb{E}_{-q_j}\left[\log \tilde{p}(\mathbf{x})\right]\right)$$

Normalization constant

- Nice exact expression for Coordinate Ascent!

Blueprint algorithm for mean field V. I.

- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

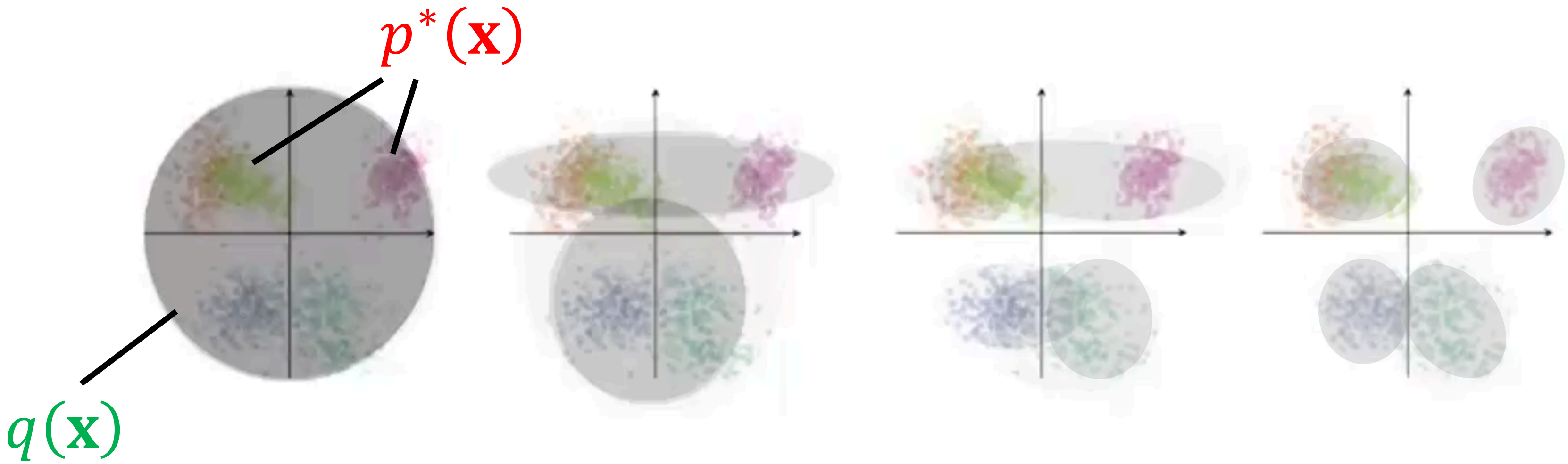
- Problem: Difficult to handle large datasets since each update of the algorithm requires full iteration through the dataset => Stochastic V. I. (won't get into)

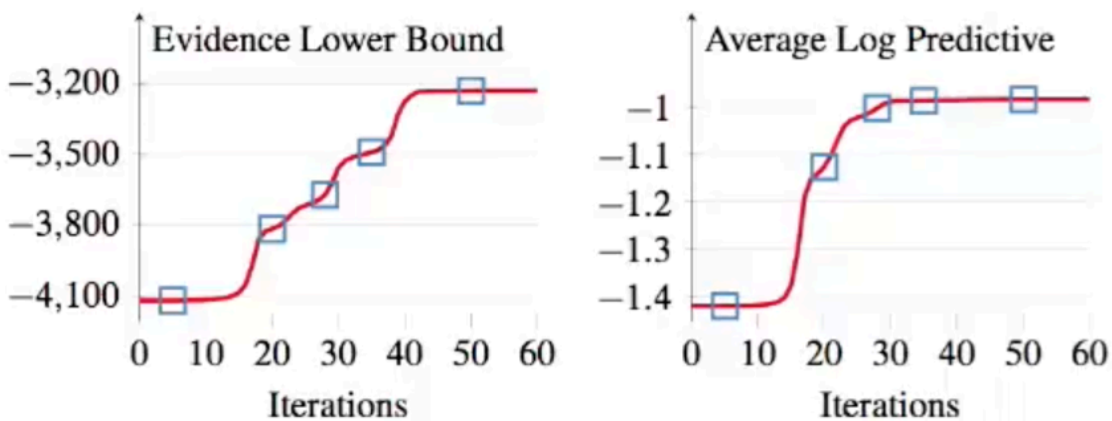
Pros and Cons

- Pros
  - Principled method to trade complexity for bias
  - Possible to assess convergence

- Cons
  - Biased estimate of the true posterior
  - Model-specific algorithms need to be derived by hand

Example: Gaussian Mixture Model





[Images by Alp Kucukelbir]

A Sample Model

- Take a sample hierarchical clustering model that is based on a mixture of K 1D Gaussians with variance 1
- Generating data is a hierarchical process
- We first sample K values μ_k from a Gaussian $\mathcal{N}(0, \sigma^2)$
- These values serve as the means of the Gaussians to be mixed

A Sample Model

- Then, we sample the "cluster" c_i the data point belongs to from $\text{Categorical}(1/K, \ldots, 1/K)$
- We treat c_i as a one-hot encoded vector indicating the cluster and define $\boldsymbol{\mu}$ to be the vector of all the means
- We then sample the data point x_i from $\mathcal{N}\left(c_i^{\top}\boldsymbol{\mu}, 1\right)$
- So the process goes like

Means -> Cluster of data point -> Sample from Gaussian with corresponding mean

A Sample Model

- Let **x** be the vector of sampled data points and **c** be the matrix containing all c_i
- The joint probability distribution of this model over all variables is

$$p(\boldsymbol{\mu}, \mathbf{c}, \mathbf{x}) = p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(c_i) p(x_i \mid c_i, \boldsymbol{\mu})$$

- Note that $p(x_i \mid c_i, \boldsymbol{\mu})$ follows $\mathcal{N}\left(c_i^{\top} \boldsymbol{\mu}, 1\right)$

Inference Difficulties with Sample Model

- We want to infer p($\boldsymbol{\mu}$, $\mathbf{c}$ | $\mathbf{x}$) since $\boldsymbol{\mu}$, $\mathbf{c}$ are the latent variables
- p($\boldsymbol{\mu}$, $\mathbf{c}$ | $\mathbf{x}$) = p($\boldsymbol{\mu}$, $\mathbf{c}$, $\mathbf{x}$) / p($\mathbf{x}$)
- However,

$$p(\mathbf{x}) = \int p(\boldsymbol{\mu}) \prod_{i=1}^{n} \sum_{c_i} p(c_i) p(x_i \mid c_i, \boldsymbol{\mu}) \, \mathrm{d}\boldsymbol{\mu}.$$

- This integral is K-dimensional and isn't equal to a product of 1-dimensional integrals and hence is intractable ($O(K^n)$ to evaluate numerically)

Inference Difficulties with Sample Model

- We could try to rewrite p(x)

$$p(\mathbf{x}) = \sum_{\mathbf{c}} p(\mathbf{c}) \int p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(x_i \,|\, c_i, \boldsymbol{\mu})\, \mathrm{d}\boldsymbol{\mu}$$

- However, the sum over c sums over K^n possible elements, so calculating this formula is also intractable

Variational Inference with Sample Model

- Variational Inference to the Rescue!
- We want to find $q(\boldsymbol{\mu}, \mathbf{c})$ that approximates

$p(\boldsymbol{\mu}, \mathbf{c} \mid \mathbf{x})$

Variational Updates For Sample Model

- Let's define the form of q($\boldsymbol{\mu}$, $\mathbf{c}$)

$$q(\boldsymbol{\mu}, \mathbf{c}) = \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i)$$

- $q(\mu_k; m_k, s_k^2)$ is a Gaussian with mean m_k and variance s_k^2
- $q(c_i; \varphi_i)$ assigns probabilities to c_i based on a vector φ_i of probabilities
- q($\boldsymbol{\mu}$, $\mathbf{c}$) decomposes into a product due to the mean field assumption; the parametric forms of the factors are chosen based on the form of p($\boldsymbol{\mu}$, $\mathbf{c}$, $\mathbf{x}$)

Variational Updates For Sample Model

- Hence, the variational parameters are φ_i, m_k, and s_k^2
- Recall (where $\mathbf{x} = (\boldsymbol{\mu}, \mathbf{c})$ here) the following two equations

$$q_j(\mathbf{x}_j) \quad = \quad \frac{1}{Z_j} \exp\left(\mathbb{E}_{-q_j}\left[\log \tilde{p}(\mathbf{x})\right]\right)$$

$$p(\boldsymbol{\mu}, \mathbf{c}, \mathbf{x}) = p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(c_i) p(x_i \mid c_i, \boldsymbol{\mu})$$

Variational Updates For Sample Model

- Hence, for the update for $\varphi_{i,}$ we obtain

$$q^*(c_i;\varphi_i) \propto \exp\left\{\log p(c_i) + \mathbb{E}\left[\log p(x_i \mid c_i, \boldsymbol{\mu}); \mathbf{m}, \mathbf{s}^2\right]\right\}$$

- Expectations are over the factors of q related to the variables after the semicolon unless otherwise indicated
- log $p(c_i)$ = log (1/K) = -log K, which is a constant
- For the second term in the sum, note that as c_i is an indicator vector, we have that

$$p(x_i \mid c_i, \boldsymbol{\mu}) = \prod_{k=1}^{K} p(x_i \mid \mu_k)^{c_{ik}}$$

Variational Updates For Sample Model

- Recall that $p(x_i \mid c_i, \boldsymbol{\mu})$ follows $\mathcal{N}\left(c_i^\top \boldsymbol{\mu}, 1\right)$
- Hence,

$$
\begin{aligned}
\mathbb{E}\left[\log p(x_i \mid c_i, \boldsymbol{\mu})\right] &= \sum_k c_{ik} \mathbb{E}\left[\log p(x_i \mid \mu_k); m_k, s_k^2\right] \\
&= \sum_k c_{ik} \mathbb{E}\left[-(x_i - \mu_k)^2/2; m_k, s_k^2\right] + \text{const.} \\
&= \sum_k c_{ik} \left(\mathbb{E}\left[\mu_k; m_k, s_k^2\right] x_i - \mathbb{E}\left[\mu_k^2; m_k, s_k^2\right]/2\right) + \text{const.}
\end{aligned}
$$

Variational Updates For Sample Model

- As c_i is an indicator vector, we obtain for the update for φ_{ik},

$$\varphi_{ik} \propto \exp\left\{\mathbb{E}\left[\mu_k; m_k, s_k^2\right] x_i - \mathbb{E}\left[\mu_k^2; m_k, s_k^2\right]/2\right\}$$

- The expected values here are easy to calculate given the form of $q(\mu_k; m_k, s_k^2)$, which is a Gaussian

Variational Updates For Sample Model

- Recall (where $\mathbf{x} = (\boldsymbol{\mu}, \mathbf{c})$ here)

$$q_j(\mathbf{x}_j) = \frac{1}{Z_j} \exp\left(\mathbb{E}_{-q_j}\left[\log \tilde{p}(\mathbf{x})\right]\right)$$

- Also, recall

$$p(\boldsymbol{\mu}, \mathbf{c}, \mathbf{x}) = p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(c_i) p(x_i \mid c_i, \boldsymbol{\mu})$$

- Hence, for the update for m_k and s_k, we get

$$q(\mu_k) \propto \exp\left\{\log p(\mu_k) + \sum_{i=1}^{n} \mathbb{E}\left[\log p(x_i \mid c_i, \boldsymbol{\mu}); \varphi_i, \mathbf{m}_{-k}, \mathbf{s}^2_{-k}\right]\right\}$$

Variational Updates For Sample Model

- We then recall that $\boldsymbol{\mu}_k$ is drawn from $\mathcal{N}(0,\sigma^2)$ and for the second equality, that $p(x_i \mid c_i, \boldsymbol{\mu}) = \prod_{k=1}^{K} p(x_i \mid \mu_k)^{c_{ik}}$

$$
\begin{aligned}
\log q(\mu_k) &= \log p(\mu_k) + \textstyle\sum_i \mathbb{E}\left[\log p(x_i \mid c_i, \boldsymbol{\mu}); \varphi_i, \mathbf{m}_{-k}, \mathbf{s}^2_{-k}\right] + \text{const.} \\
&= \log p(\mu_k) + \textstyle\sum_i \mathbb{E}\left[c_{ik} \log p(x_i \mid \mu_k); \varphi_i\right] + \text{const.} \\
&= -\mu_k^2/2\sigma^2 + \textstyle\sum_i \mathbb{E}\left[c_{ik}; \varphi_i\right] \log p(x_i \mid \mu_k) + \text{const.} \\
&= -\mu_k^2/2\sigma^2 + \textstyle\sum_i \varphi_{ik} \left(-(x_i - \mu_k)^2/2\right) + \text{const.} \\
&= -\mu_k^2/2\sigma^2 + \textstyle\sum_i \varphi_{ik} x_i \mu_k - \varphi_{ik}\mu_k^2/2 + \text{const.} \\
&= \left(\textstyle\sum_i \varphi_{ik} x_i\right) \mu_k - \left(1/2\sigma^2 + \textstyle\sum_i \varphi_{ik}/2\right) \mu_k^2 + \text{const.}
\end{aligned}
$$

Variational Updates For Sample Model

- We then note that $q(\boldsymbol{\mu}_k)$ is a Gaussian distribution and update m_k and s_k^2 according to the mean and standard deviation of the Gaussian

$$m_k = \frac{\sum_i \varphi_{ik} x_i}{1/\sigma^2 + \sum_i \varphi_{ik}}, \qquad s_k^2 = \frac{1}{1/\sigma^2 + \sum_i \varphi_{ik}}$$

Deriving ELBO for Simple Model

- Recall this equation for the ELBO:

$$L(q) \triangleq -J(q) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{\tilde{p}(\mathbf{x})}{q(\mathbf{x})}$$

- The ELBO can be rewritten as follows, where expectations are taken over q(z) (here, z represents the latent variables instead of x and p(z, x) = $\tilde{p}(\mathbf{x})$)

$$\text{ELBO}(q) = \mathbb{E}\left[\log p(\mathbf{z}, \mathbf{x})\right] - \mathbb{E}\left[\log q(\mathbf{z})\right]$$

Deriving ELBO for Simple Model

- Recall $$p(\boldsymbol{\mu}, \mathbf{c}, \mathbf{x}) = p(\boldsymbol{\mu}) \prod_{i=1}^{n} p(c_i) p(x_i \mid c_i, \boldsymbol{\mu})$$

And $$q(\boldsymbol{\mu}, \mathbf{c}) = \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i).$$

Deriving ELBO for Simple Model

- We then obtain (note that p and q factorize)

$$\begin{aligned}
\mathrm{ELBO}(\mathbf{m}, \mathbf{s}^2, \boldsymbol{\varphi}) = & \sum_{k=1}^{K} \mathbb{E}\left[\log p(\mu_k); m_k, s_k^2\right] \\
& + \sum_{i=1}^{n} \left(\mathbb{E}\left[\log p(c_i); \varphi_i\right] + \mathbb{E}\left[\log p(x_i \mid c_i, \boldsymbol{\mu}); \varphi_i, \mathbf{m}, \mathbf{s}^2\right]\right) \\
& - \sum_{i=1}^{n} \mathbb{E}\left[\log q(c_i; \varphi_i)\right] - \sum_{k=1}^{K} \mathbb{E}\left[\log q(\mu_k; m_k, s_k^2)\right].
\end{aligned}$$

- Each of these expectations can be calculated in closed form

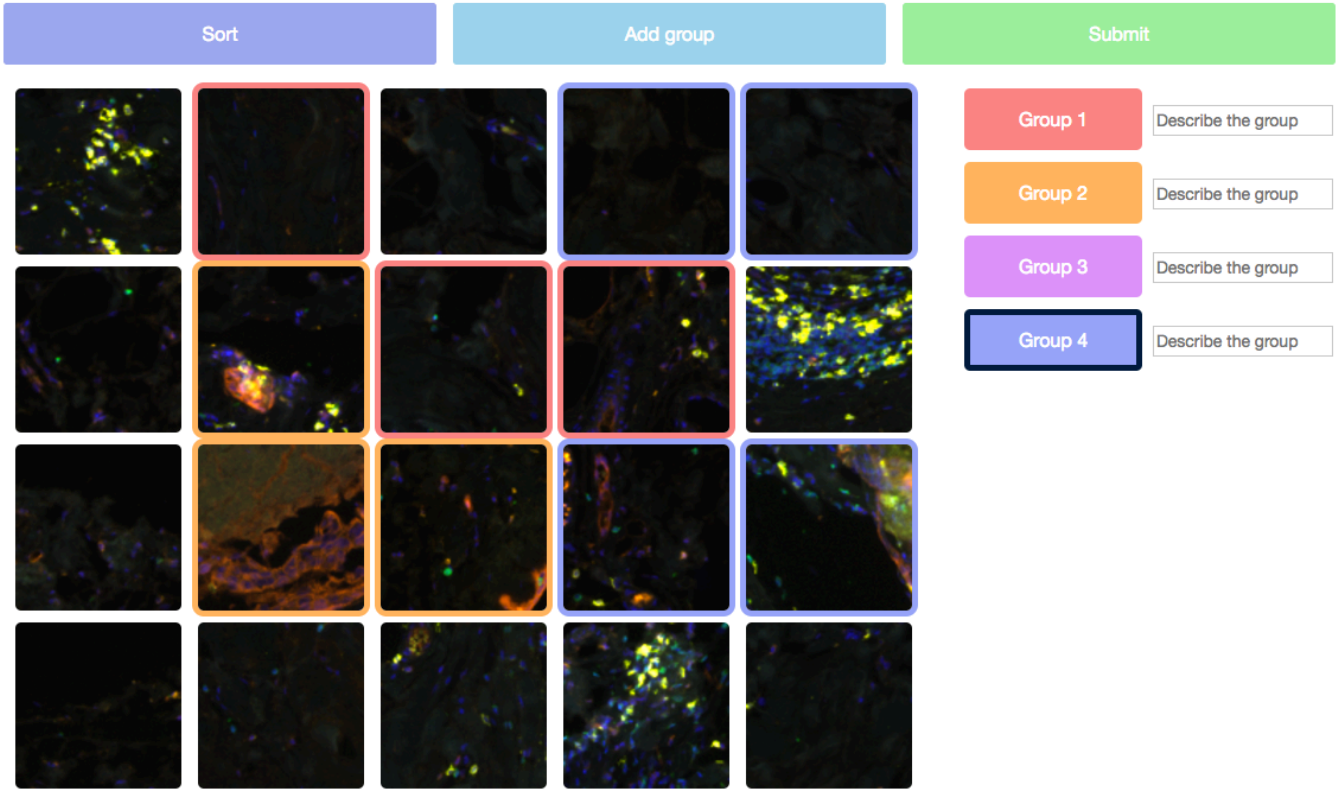
Applications

Application: Crowd Clustering

- Crowd sourcing has been used to label large datasets of data
- Conventionally:
  - Experts provide the *categories*
  - Crowd labels images with *predefined categories*

Application: Crowd Clustering

- Can workers **discover** categories?
- We want to use the crowd to cluster images in an unsupervised manner.



Application: Crowd Clustering

- But how do we aggregate data from multiple workers?
- We extract binary pairwise labels from the worker provided clusterings.

$$(\text{image id } 1, \text{image id } 2, \text{same cluster?}\,) = (a_i, b_i, l)$$

+/- 1

- Then, we want to find some embedding of images that aggregates the information provided by the workers.
- To do so, we first define a graphical model.

Blueprint algorithm for mean field V. I.

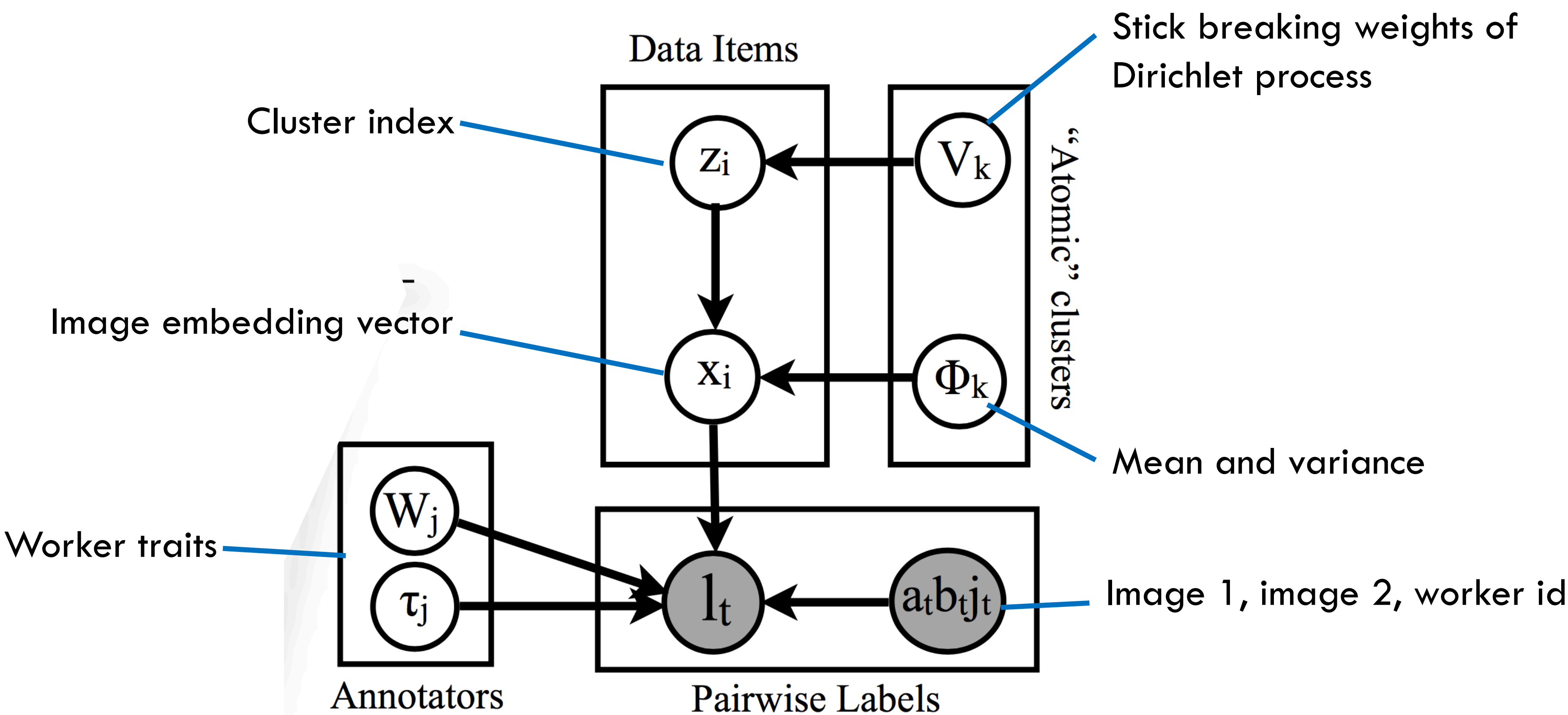
- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

Blueprint algorithm for mean field V. I.

- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

Application: Crowd Clustering

- Clusters are sampled from a Dirichlet process
- Each cluster has a corresponding mean and variance which describe a Multivariate Gaussian from which the images are sampled



Application: Crowd Clustering

- How do we define the likelihood of the data?
- Workers act as a logistic classifier in the embedding space

$$p(l_t | \mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) = \frac{1}{1 + \exp(-l_t A_t)}$$

+/- 1

- The strength of the similarity is defined as

$$A_t = \mathbf{x}_{a_t}^T \mathbf{W}_{j_t} \mathbf{x}_{b_t} + \tau_{j_t}$$

- High A_t: Images will be labeled as the same cluster
- Low A_t: Images will be labeled to be in different cluster

Application: Crowd Clustering

- Then, the joint distribution is defined as

$$p(\Phi, V, Z, X, W, \tau, \mathcal{L}) = \prod_{k=1}^{\infty} p(V_k|\alpha)p(\Phi_k|\mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) \prod_{i=1}^{N} p(z_i|V)p(\mathbf{x}_i|\Phi_{z_i})$$
$$\prod_{j=1}^{J} p(\mathrm{vecp}\{\mathbf{W}_j\}|\sigma_0^w)p(\tau_j|\sigma_0^\tau) \prod_{t=1}^{T} p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}).$$

- Exact inference of the posterior is clearly intractable since we need to integrate over variables with complex dependencies. (Encoded in each of the factorized distributions)

Blueprint algorithm for mean field V. I.

- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

Application: Crowd Clustering

- We instead use Variational Inference.
- Define a factorized proxy posterior which doesn't model the full complexity between the variables. Instead it represents a single mode of the true posterior.

$$q(\Phi, V, Z, X, W, \tau) = \prod_{k=K+1}^{\infty} p(V_k|\alpha)p(\Phi_k|\mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0)$$

$$\prod_{k=1}^{K} q(V_k)q(\Phi_k) \prod_{i=1}^{N} q(z_i)q(\mathbf{x}_i) \prod_{j=1}^{J} q(\text{vecp}\{\mathbf{W}_j\})q(\tau_j)$$

Blueprint algorithm for mean field V. I.

- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

Application: Crowd Clustering

- We define variational distributions for the first K mixture components and fix the remainder to their corresponding priors.
- Then we can define a lower bound to the log evidence

$$\begin{aligned}\log p(\mathcal{L}|\sigma_0^x, \sigma_0^\tau, \sigma_0^w, \alpha, \mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) \\ \geq E_q \log p(\Phi, V, Z, X, W, \tau, \mathcal{L}) + \mathcal{H}\{q(\Phi, V, Z, X, W, \tau)\}\end{aligned}$$

Blueprint algorithm for mean field V. I.

- Derive a probablistic model for problem
- Choose a proxy distribution q
- Derive ELBO
- Coordinate ascent on each q_i (Ghahramani and Beal, 2001)
- Repeat until convergence

Applications: Crowd Clustering

$$q_{ik} = q(z_i = k) \sim \exp\Big(\psi(\xi_{k,1}) - \psi(\xi_{k,1} + \xi_{k,2}) + \sum_{l=1}^{k-1} \psi(\xi_{l,2}) - \psi(\sum_{l=1}^{k-1} \xi_{l,1} + \xi_{l,2})$$

$$- \frac{\eta_k}{2} \mathrm{tr}\{\mathbf{J}_k^{-1} E_q\{\mathbf{x}_i \mathbf{x}_i^T\}\} - \frac{1}{2}(D \log(\pi) - \log|\mathbf{J}_k| + \sum_{d=1}^{D} \psi(1 + \eta_k - \frac{d}{2}) - \frac{D}{\beta_k}\Big)$$

$$[\boldsymbol{\sigma}_i^x]_d = \Big(\sum_k q(z_i = k)\eta_k [\mathbf{J}_k]_{dd} + \sum_{t:a_t=i} 2|\lambda(\Delta_t)| \big[E_q\{\mathbf{W}_{j_t} \mathbf{x}_{b_t} \mathbf{x}_{b_t}^T \mathbf{W}_{j_t}\}\big]_{dd}$$

$$+ \sum_{t:b_t=i} 2|\lambda(\Delta_t)| \big[E_q\{\mathbf{W}_{j_t} \mathbf{x}_{a_t} \mathbf{x}_{a_t}^T \mathbf{W}_{j_t}\}\big]_{dd}\Big)^{-1}$$

$$\boldsymbol{\mu}_i^x = (\mathbf{I} - \mathbf{U}_i \circ (\mathbf{1} - \mathbf{I}))^{-1} \mathbf{v}_i$$

$$\sigma_j^\tau = \Big(1/\sigma_0^\tau + \sum_{t:j_t=j} 2|\lambda(\Delta_t)|\Big)^{-1}$$

$$\mu_j^\tau = \sigma_j^\tau \sum_{t:j_t=j} l_t/2 + 2\lambda(\Delta_t)(\boldsymbol{\mu}_{a_t}^x)^T \boldsymbol{\mu}_j^w \boldsymbol{\mu}_{b_t}^x$$

$$[\boldsymbol{\sigma}_j^w]_{d_1 d_2} = \left(1/\sigma_0^w + \sum_{t:j_t=j} 2|\lambda(\Delta_t)| \big[E_q\{\mathbf{Y}^{a_t b_t}\}\big]_{d_1 d_2 d_1 d_2}\right)^{-1}$$

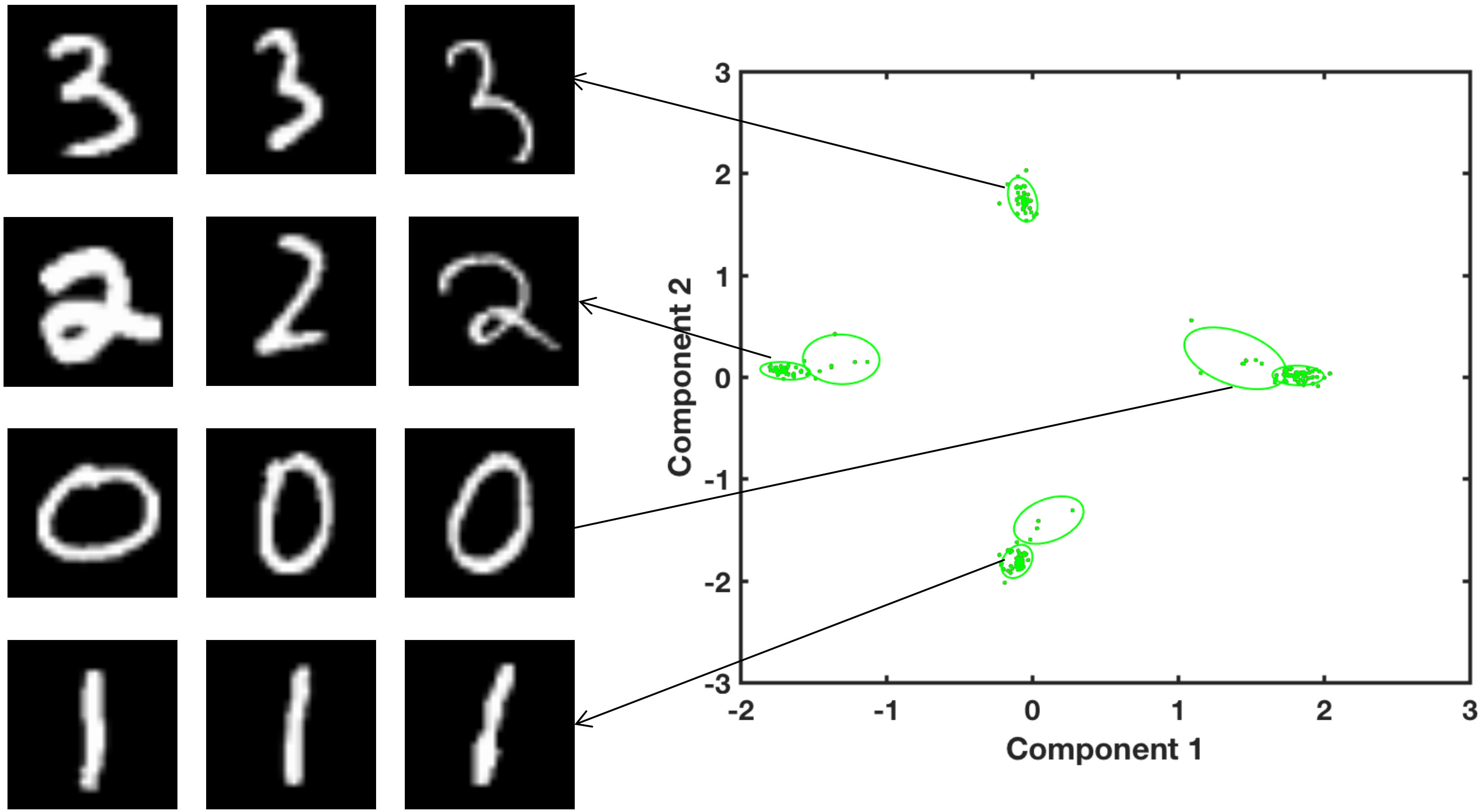
$$\mathrm{vecp}\{\boldsymbol{\mu}_j^w\} = (\mathbf{I} - \mathbf{B}_j \circ (\mathbf{1} - \mathbf{I}))^{-1} \mathbf{c}_j$$

$$\Delta_t = (E_q\{A_t^2\})^{1/2}$$

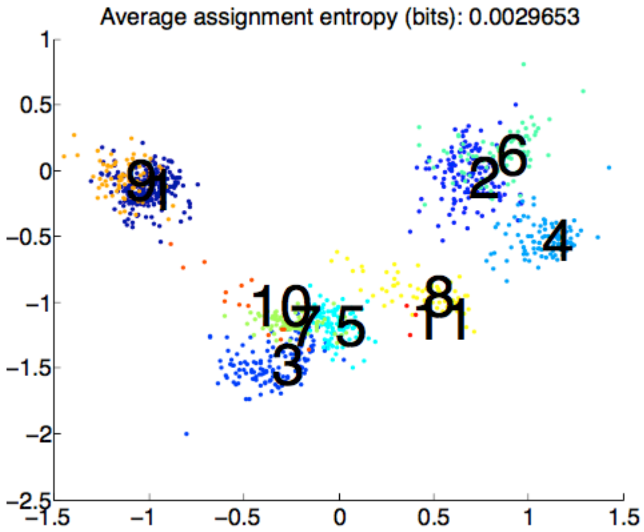
where $N_k = \sum_i q_{ki}$ and $\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_i q_{ik} \mathbf{x}_i$.

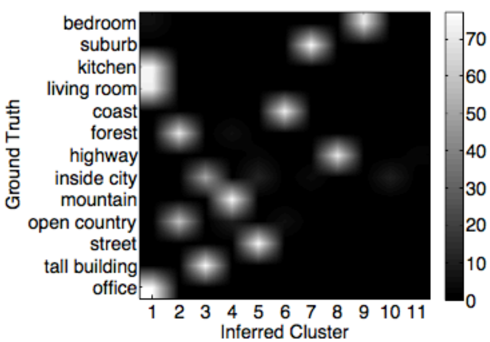
Let's not bother

Application: Crowd Clustering

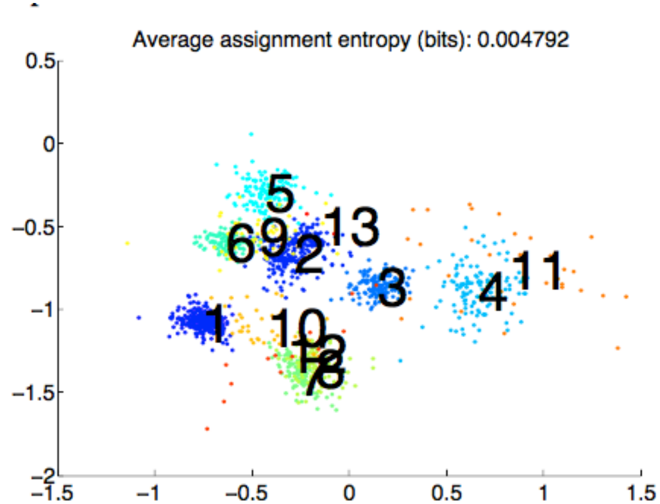


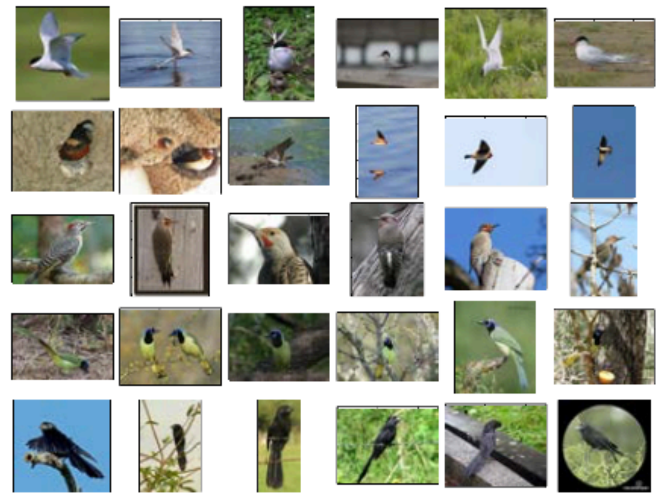
Application: Crowd Clustering

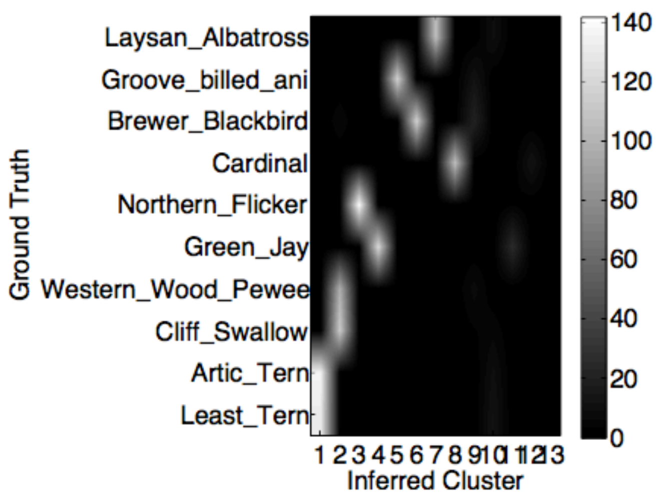


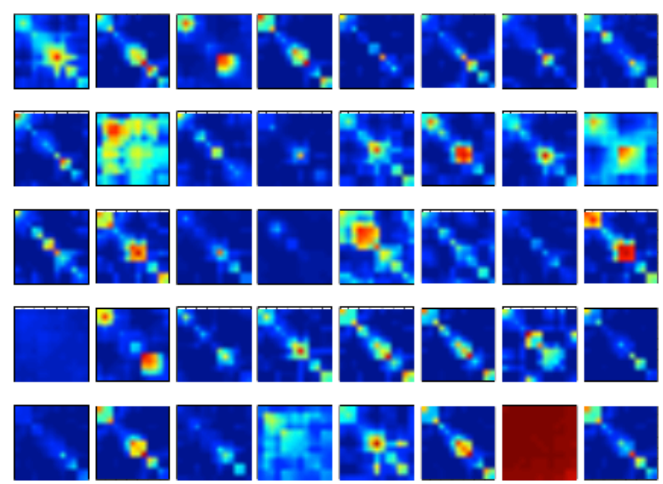


Application: Crowd Clustering









Application: Variational Autoencoders

- An autoencoder uses neural networks to represent its input $\mathbf{x}$ in terms of latent variables $\mathbf{z}$ and then uses this representation to generate output data $\mathbf{x}'$ similar to the input
- So we encode the input $\mathbf{x}$ using f, $z = f(\mathbf{x})$, and then decode $\mathbf{z}$ by some function g, $\mathbf{x}' = g(\mathbf{z})$, where both f and g depend on model parameters

Application: Variational Autoencoders

- For variational autoencoders, we use neural networks to model probability distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ instead of deterministic functions $\mathbf{x}' = g(\mathbf{z})$ and $\mathbf{z} = f(\mathbf{x})$, where θ represents model parameters

Application: Variational Autoencoders

- We assume that the input data $\mathbf{x}$ is generated by first sampling $\mathbf{z}$ from a prior distribution $p_\theta(\mathbf{z})$ and then sampling $\mathbf{x}$ from the distribution $p_\theta(\mathbf{x}|\mathbf{z})$
- $\mathbf{x}$ can be discrete or continuous, but $\mathbf{z}$ is assumed to be continuous
- Note: Discrete versions of variational autoencoders exist where $\mathbf{z}$ is discrete; see Rolfe 2017.

Application: Variational Autoencoders

- We would like to model $p_\theta(\mathbf{x}|\mathbf{z})$ and infer $p_\theta(\mathbf{z}|\mathbf{x})$ given a large dataset and that the marginal likelihood $p_\theta(\mathbf{x})$ is intractable
- Calculating the expectations for the mean field update equations may be intractable, so we don't use mean field inference
- Again, since $p_\theta(\mathbf{x})$ and hence $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, we introduce a variational family $q_\Phi(\mathbf{z}|\mathbf{x})$ as an approximation

Optimizing ELBO Without Mean-Field

- Recall that the ELBO can be written in terms of the KL divergence

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) || p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

- Optimizing the ELBO can be challenging without the mean field assumption since the gradient of the ELBO may not have a closed form
- We could use the Monte Carlo gradient estimator (below), but it has high variance

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\left[f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z})} \log q_{\boldsymbol{\phi}}(\mathbf{z})\right] \simeq \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}) \nabla_{q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(l)})$$

Reparametrization Trick

- Notably, we can often rewrite a sample $\widetilde{\mathbf{z}}$ from $q_{\Phi}(\mathbf{z} | \mathbf{x})$ using a deterministic differentiable function of a noise variable **ε** that is drawn from a probability distribution p

$$\widetilde{\mathbf{z}} = g_{\phi}(\epsilon, \mathbf{x}) \quad \text{with} \quad \epsilon \sim p(\epsilon)$$

- For this technique to work, **z** must be continuous (otherwise, g wouldn't be differentiable)

Example of Reparametrization Trick

- Let $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$
- We could also rewrite z as $z = \mu + \sigma\epsilon \quad \epsilon \sim \mathcal{N}(0,1)$

Optimizing ELBO Without Mean-Field

- We can now write Monte Carlo estimators of expectations of f(z) as follows

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

- Applying this technique to the ELBO for the variational autoencoder, we obtain

$$\widetilde{\mathcal{L}}^{A}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)}|\mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

Optimizing ELBO Without Mean-Field

- If the K-L divergence can be evaluated analytically, then we need only Monte Carlo estimate the second term (this expression generally has less variance):

$$\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

- Note that KL divergence from the prior acts as a regularizer and the second term is negative reconstruction error

Optimizing ELBO Without Mean-Field

- We can take the gradient of either one of the ELBO estimators and use it in a gradient-based optimizer such as stochastic gradient ascent
- We can also use minibatches with the ELBO estimator using the equation below

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \phi; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)})$$

Setup for Variational Autoencoder

- $p_\theta(\mathbf{z})$ is set to $N(\mathbf{z}; \mathbf{0}, \mathbf{I})$, the multivariate Gaussian with mean **0** and covariance the identity matrix
- $p_\theta(\mathbf{x}|\mathbf{z})$ is set to a multivariate Gaussian for continuous **x** or Bernoulli for binary **x**
- $q_\Phi(\mathbf{z}|\mathbf{x})$ is set to a multivariate Gaussian with diagonal covariance matrix
- I.e. $\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I})$
- Θ and Φ are parameters set by neural networks with one hidden layer

Setup for Variational Autoencoder

- We use the reparametrization trick for the Gaussian distribution similar to the example given before
- The ELBO Monte Carlo estimator for this model is

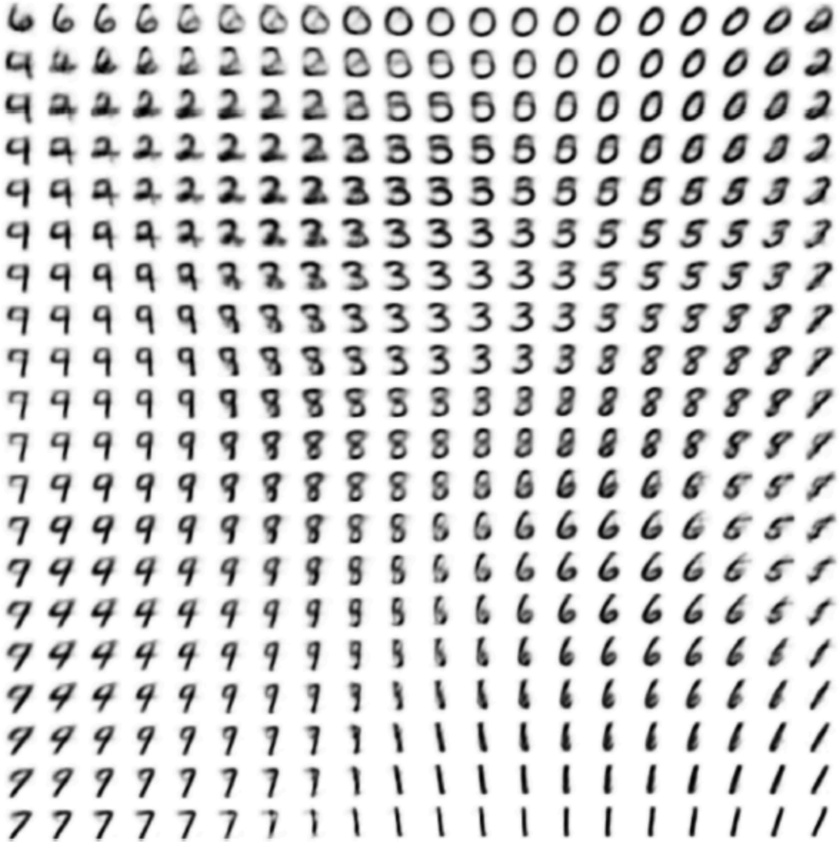
$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$$

- Here, J represents the dimensionality of **z**, $\odot$ represents element-wise product

Applications of Variational Autoencoder

- Generating handwritten digits (learned data manifold for 2D latent space below from Kingma & Welling 2014)

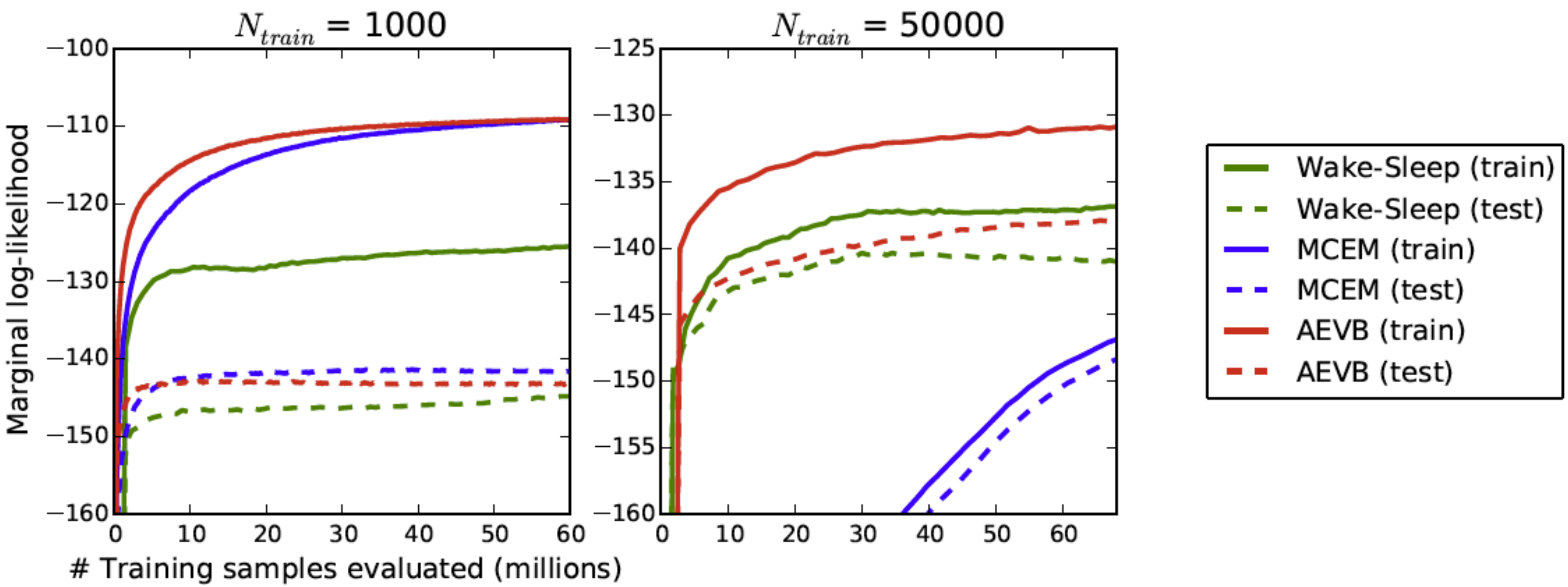


Applications of Variational Autoencoder

- Examples of handwritten digits generated by variational autoencoder using 20 dimensional latent space (also from Kingma & Welling 2014)

Applications of Variational Autoencoder

- Comparison of Variational Autoencoder (labeled as AEVB) compared to other methods (from Kingma & Welling 2014)



Applications of Variational Autoencoder

- Learning representations of images (from Kulkarni et al., 2015)