Latent Variable Models

Stefano Ermon, Aditya Grover

Stanford University

Lecture 5

Recap of last lecture

1. Autoregressive models:
   - Chain rule based factorization is fully general
   - Compact representation via *conditional independence* and/or *neural parameterizations*
2. Autoregressive models Pros:
   - Easy to evaluate likelihoods
   - Easy to train
3. Autoregressive models Cons:
   - Requires an ordering
   - Generation is sequential
   - Cannot learn features in an unsupervised way

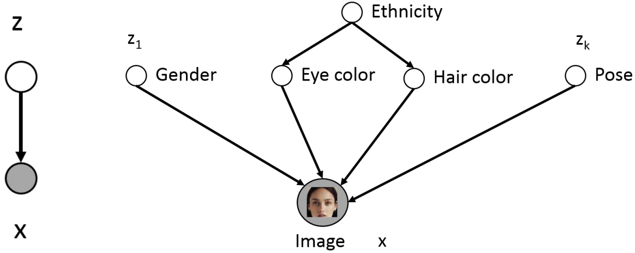
Plan for today

1. Latent Variable Models
    - Variational EM

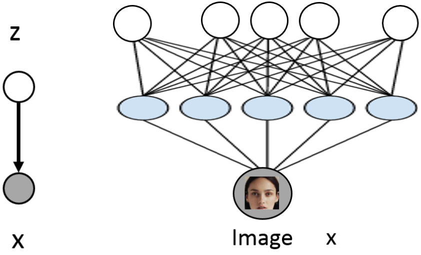
Latent Variable Models: Motivation

1. Lots of variability in images $\mathbf{x}$ due to gender, eye color, hair color, pose, etc. However, unless images are annotated, these factors of variation are not explicitly available (latent).
2. **Idea**: explicitly model these factors using latent variables $\mathbf{z}$

Latent Variable Models: Motivation



1. Only shaded variables $\mathbf{x}$ are observed in the data (pixel values)
2. Latent variables $\mathbf{z}$ correspond to high level features
   - If $\mathbf{z}$ chosen properly, $p(\mathbf{x}|\mathbf{z})$ could be much simpler than $p(\mathbf{x})$
   - If we had trained this model, then we could identify features via $p(\mathbf{z} \mid \mathbf{x})$, e.g., $p(EyeColor = Blue|\mathbf{x})$
3. **Challenge:** Very difficult to specify these conditionals by hand

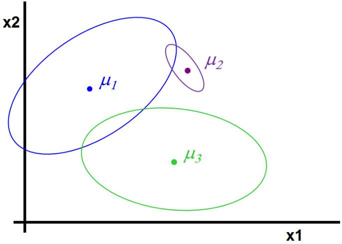


1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. *Hope* that after training, $\mathbf{z}$ will correspond to meaningful latent factors of variation (*features*). Unsupervised representation learning.
4. As before, features can be computed via $p(\mathbf{z} \mid \mathbf{x})$

Mixture of Gaussians: a Shallow Latent Variable Model

Mixture of Gaussians. Bayes net: $\mathbf{z} \rightarrow \mathbf{x}$.

1. $\mathbf{z} \sim \text{Categorical}(1, \cdots, K)$
2. $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$

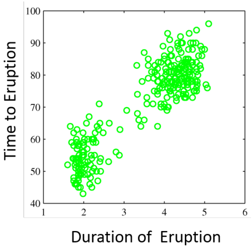


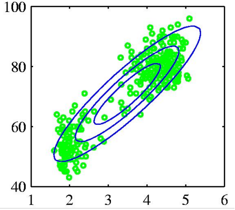
Generative process

1. Pick a mixture component k by sampling z
2. Generate a data point by sampling from that Gaussian

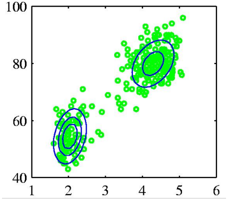
Mixture of Gaussians:

1. $\mathbf{z} \sim \mathrm{Categorical}(1, \cdots, K)$
2. $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$





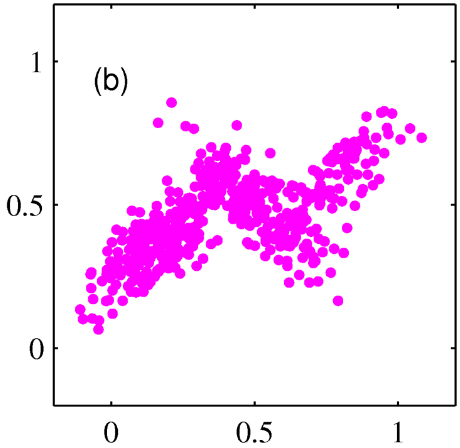
Single Gaussian

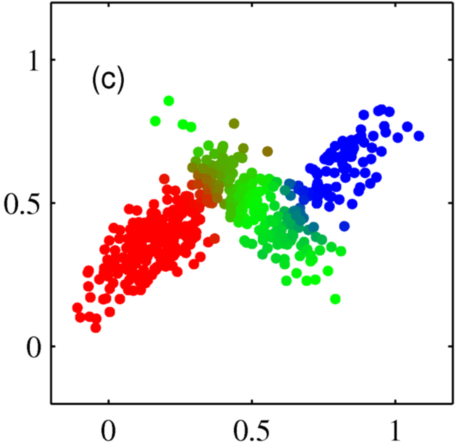


Mixture of two Gaussians

3. **Clustering:** The posterior $p(\mathbf{z} \mid \mathbf{x})$ identifies the mixture component
4. **Unsupervised learning:** We are hoping to learn from unlabeled data (ill-posed problem)

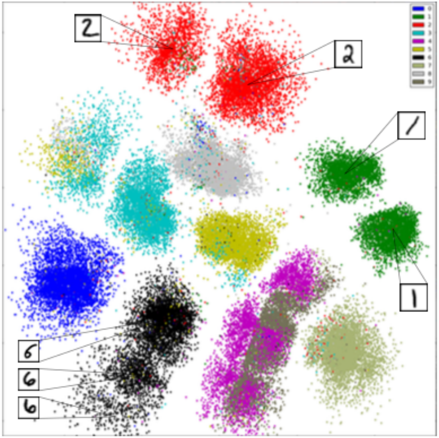
Unsupervised learning





Shown is the posterior probability that a data point was generated by the i-th mixture component, $P(z = i|x)$

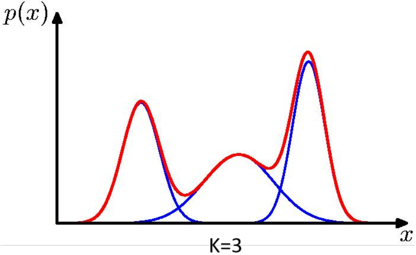
Unsupervised learning



Unsupervised clustering of handwritten digits.

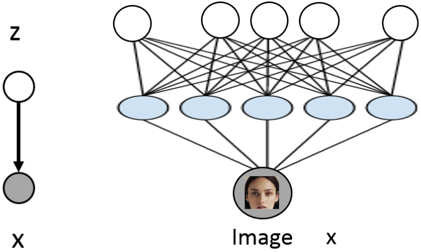
Unsupervised learning

Combine simple models into a more complex and expressive one



$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) = \sum_{k=1}^{K} p(\mathbf{z} = k) \underbrace{\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)}_{\text{component}}$$

Variational Autoencoder



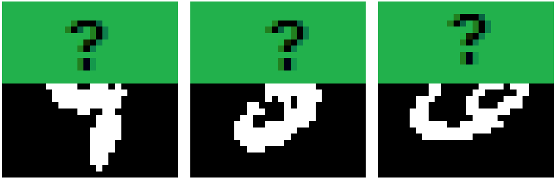
A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
   - $\mu_\theta(\mathbf{z}) = \sigma(A\mathbf{z} + c) = (\sigma(a_1\mathbf{z} + c_1), \sigma(a_2\mathbf{z} + c_2)) = (\mu_1(\mathbf{z}), \mu_2(\mathbf{z}))$
   - $\Sigma_\theta(\mathbf{z}) = diag(\exp(\sigma(B\mathbf{z} + d))) = \begin{pmatrix} \exp(\sigma(b_1\mathbf{z}+d_1)) & 0 \\ 0 & \exp(\sigma(b_2\mathbf{z}+d_2)) \end{pmatrix}$
   - $\theta = (A, B, c, d)$
3. Even though $p(\mathbf{x} \mid \mathbf{z})$ is simple, the marginal $p(\mathbf{x})$ is very complex/flexible

Recap

- Latent Variable Models
  - Allow us to define complex models $p(\mathbf{x})$ in terms of simple building blocks $p(\mathbf{x} \mid \mathbf{z})$
  - Natural for unsupervised learning tasks (clustering, unsupervised representation learning, etc.)
  - No free lunch: much more difficult to learn compared to fully observed, autoregressive models

Marginal Likelihood



- Suppose some pixel values are missing at train time (e.g., top half)
- Let $\mathbf{X}$ denote observed random variables, and $\mathbf{Z}$ the unobserved ones (also called hidden or latent)
- Suppose we have a model for the joint distribution (e.g., PixelCNN)

$$p(\mathbf{X}, \mathbf{Z}; \theta)$$

What is the probability $p(\mathbf{X} = \bar{\mathbf{x}}; \theta)$ of observing a training data point $\bar{\mathbf{x}}$?

$$\sum_{\mathbf{z}} p(\mathbf{X} = \bar{\mathbf{x}}, \mathbf{Z} = \mathbf{z}; \theta) = \sum_{\mathbf{z}} p(\bar{\mathbf{x}}, \mathbf{z}; \theta)$$

- Need to consider all possible ways to complete the image (fill green part)

Partially observed data

- Suppose that our joint distribution is

$$p(\mathbf{X}, \mathbf{Z}; \theta)$$

- We have a dataset $\mathcal{D}$, where for each datapoint the $\mathbf{X}$ variables are observed (e.g., pixel values) and the variables $\mathbf{Z}$ are never observed (e.g., cluster or class id.). $\mathcal{D} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}\}$.
- Maximum likelihood learning:

$$\begin{aligned} \ell(\theta; \mathcal{D}) &= \log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}; \theta) \\ &= \sum_{\mathbf{x} \in \mathcal{D}} \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta) \end{aligned}$$

- Evaluating $\log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta)$ can be intractable. Suppose we have 30 binary latent features, $\mathbf{z} \in \{0, 1\}^{30}$. Evaluating $\sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta)$ involves a sum with 2^{30} terms. For continuous variables, $\log \int_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z}$ is often intractable.

Why is parameter learning in presence of Partially Observed Data challenging?

Likelihood function for Fully Observed Data:

$$\underbrace{p_\theta(\mathbf{x})}_{\text{easy to compute}} = \prod_i \log p(x_i \mid \mathbf{x}_{pa(i)})$$

Compare with Likelihood function for Partially Observed Data:

$$\underbrace{\sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z})}_{\text{hard to compute}}$$

Likelihood function for Partially Observed Data:

- is not decomposable (by variable and parent assignment) and not unimodal as a function of θ. Could still try gradient descent.
- Hard to compute and take gradients ∇_θ (too many completions)

First attempt: Naive Monte Carlo

Likelihood function for Partially Observed Data is hard to compute:

$$\sum_{\text{All possible values of } \mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) = |\mathcal{Z}| \sum_{\mathbf{z} \in \mathcal{Z}} \frac{1}{|\mathcal{Z}|} p_\theta(\mathbf{x}, \mathbf{z}) = |\mathcal{Z}| \mathbb{E}_{\mathbf{z} \sim Uniform(\mathcal{Z})} [p_\theta(\mathbf{x}, \mathbf{z})]$$

We can think of it as an (intractable) expectation. Monte Carlo to the rescue:

1. Sample $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(k)}$ uniformly at random
2. Approximate expectation with sample average

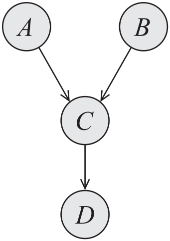
$$\sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) \approx |\mathcal{Z}| \frac{1}{k} \sum_{j=1}^{k} p_\theta(\mathbf{x}, \mathbf{z}^{(j)})$$

Works in theory but not in practice. For most $\mathbf{z}$, $p_\theta(\mathbf{x}, \mathbf{z})$ is very low (most completions don't make sense). Some are very large but will never "hit" likely completions by uniform random sampling. Need a clever way to select $\mathbf{z}^{(j)}$ to reduce variance of the estimator.

The Expectation Maximization (EM) Algorithm

- Start with an initial guess (random) of the parameters $\theta^{(0)}$
- Repeat until convergence
  1. Complete ("hallucinate") the incomplete data (the $\mathbf{z}$ part) using current parameters (E-step)
  2. Train: Update the parameters based on the completed data (M-step)

The Expectation Maximization Algorithm: Example



$\theta_a = .3$
$\theta_b = .9$
$\theta_{c|\bar{a},\bar{b}} = .83$
$\theta_{c|\bar{a},b} = .09$
$\theta_{c|a,\bar{b}} = .6$
$\theta_{c|a,b} = .2$
$\theta_{d|\bar{c}} = .1$
$\theta_{d|c} = .8$

Data instance: $(a, ?, ?, \bar{d})$

How to complete this example?

For each possible completion

- STEP 1: Compute how likely the completion is (given the observed part).
- Compute $P(\mathbf{z}|\mathbf{x})$. In this example $P(B, C \mid A = a, D = \bar{d})$.

For example,

$$P(b, c \mid a, \bar{d}) = \frac{P(a, b, c, \bar{d})}{P(a, \bar{d})} = \frac{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8)}{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8) + \cdots + 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9} = .0492$$

$$P(\bar{b}, c \mid a, \bar{d}) = \frac{P(a, \bar{b}, c, \bar{d})}{P(a, \bar{d})} = \frac{0.3 \cdot 0.1 \cdot 0.6 \cdot (1 - 0.8)}{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8) + \cdots + 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9} = .0164$$

The Expectation Maximization Algorithm

- Data set is now bigger and weighted
- If binary variables, $(a, ?, ?, \bar{d})$ corresponds to four weighted examples
  - $(a, b, c, \bar{d})$, weight $= .0492 = P(b, c \mid a, \bar{d})$
  - $(a, b, \bar{c}, \bar{d})$, weight $= .8852 = P(b, \bar{c} \mid a, \bar{d})$
  - $(a, \bar{b}, c, \bar{d})$, weight $= .0164 = P(\bar{b}, c, \mid a, \bar{d})$
  - $(a, \bar{b}, \bar{c}, \bar{d})$, weight $= .0492 = P(\bar{b}, \bar{c} \mid a, \bar{d})$
- weight = probability according to current parameter estimates
- After completion, the dataset is fully observed, so we can train as usual via gradient descent or closed-form (M-Step)

$$\theta_{t+1} = \arg\max_{\theta} \sum_{m=1}^{M} E_{p(\mathbf{z}_m|\mathbf{x}_m;\theta_t)}[\log p(\mathbf{x}_m, \mathbf{z}_m; \theta)]$$

- Two problems:
  1. $p(\mathbf{z}_m \mid \mathbf{x}_m; \theta_t) = p(\mathbf{x}_m, \mathbf{z}_m; \theta_t)/p(\mathbf{x}_m; \theta_t)$ requires $p(\mathbf{x}_m; \theta_t) = \sum_{\mathbf{z}} p(\mathbf{x}_m, \mathbf{z}; \theta_t)$ (what you'd need for gradient descent)
  2. Still requires looking at all possible completions
- Solution: replace $p(\mathbf{z} \mid \mathbf{x}; \theta)$ with a tractable $q(\mathbf{z})$ and do Monte Carlo

The EM Algorithm

- Suppose $q(\mathbf{z})$ is **any** probability distribution over the hidden variables

$$
\begin{aligned}
D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) &= \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x};\theta)} \\
&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z}|\mathbf{x};\theta) + \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})}_{-H(q)} \\
&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z}|\mathbf{x};\theta) - H(q) \\
&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{z},\mathbf{x};\theta)}{p(\mathbf{x};\theta)} - H(q) \\
&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{x};\theta) - H(q) \\
&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q) \geq 0
\end{aligned}
$$

The EM Algorithm

- Suppose $q(\mathbf{z})$ is **any** probability distribution over the hidden variables

$$D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) = -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q) \geq 0$$

- **Evidence lower bound** (ELBO) holds for any q

$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q)$$

- Equality holds if $q = p(\mathbf{z}|\mathbf{x};\theta)$

$$\log p(\mathbf{x};\theta) {\color{red}=} \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q)$$

- This is what we compute in the E-step of the EM algorithm

The EM Algorithm

- $q(\mathbf{z})$ is an arbitrary probability distribution over $\mathbf{z}$

$$D_{KL}(q\|p(\mathbf{z}|\mathbf{x};\theta)) = -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q)$$

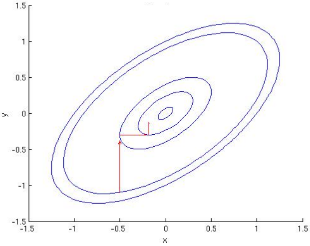
- Re-arranging can rewrite as

$$H(q) + \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) = \log p(\mathbf{x};\theta) - D_{KL}(q\|p(\mathbf{z}|\mathbf{x};\theta)) \triangleq F[q,\theta]$$

- Can interpret EM as coordinate ascent on $F[q,\theta]$
  1. Initialize $\theta^{(0)}$
  2. $q^{(0)} = \arg\max_q F[q,\theta^{(0)}] = p(\mathbf{z}|\mathbf{x};\theta^{(0)})$
  3. $\theta^{(1)} = \arg\max_\theta F[q^{(0)},\theta] = \arg\max_\theta \sum_{\mathbf{z}} q^{(0)}(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta)$
  4. $q^{(1)} = \arg\max_q F[q,\theta^{(1)}]$
  5. ...
- Marginal likelihood never decreases

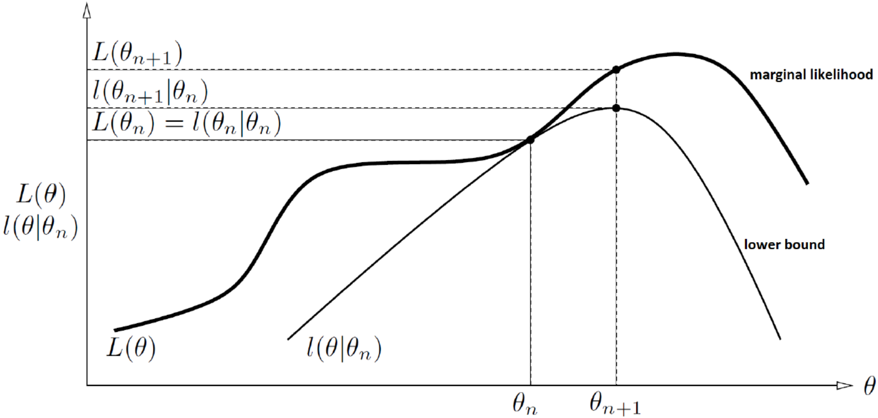
$$\begin{aligned}\log p(\mathbf{x};\theta^{(0)}) = F[q^{(0)},\theta^{(0)}] \le F[q^{(0)},\theta^{(1)}] &= \log p(\mathbf{x};\theta^{(1)}) - D_{KL}(q^{(0)}\|p(\mathbf{z}|\mathbf{x};\theta^{(1)})) \\ &\le \log p(\mathbf{x};\theta^{(1)}) = F[q^{(1)},\theta^{(1)}] = \cdots\end{aligned}$$

Coordinate ascent



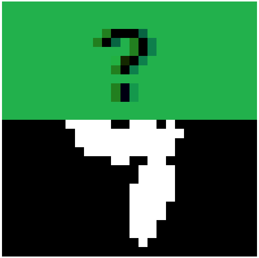
- Maximize along one direction at a time:
  - Initialize x_0, y_0
  - $y_1 = \arg\max_y f(x_0, y)$
  - $x_1 = \arg\max_x f(x, y_1)$
  - $y_2 = \arg\max_y f(x_1, y)$
  - ...
- Objective keeps improving.
  $f(x_0, y_0) \leq f(x_0, y_1) \leq f(x_1, y_1) \leq f(x_1, y_2) \leq \cdots$

Derivation of EM algorithm



(Figure adapted from tutorial by Sean Borman)

The Evidence Lower bound

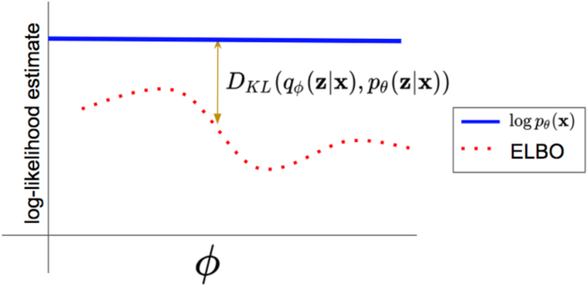


- What if the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute in the E-step?
- Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - E.g., a Gaussian with mean and covariance specified by ϕ, a fully factored probability distribution, a FVSBN, etc.

$$q(\mathbf{z};\phi) = \prod_{\text{unobserved variables } z_i} (\phi_i)^{z_i}(1-\phi_i)^{(1-z_i)}$$

- Note: conditioned on the bottom part ($\mathbf{x}$), choosing pixels independently in $\mathbf{z}$ is not a terrible approximation

The Evidence Lower bound



$$
\begin{aligned}
\log p(\mathbf{x};\theta) &\geq \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}} \\
&= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))
\end{aligned}
$$

The better $q(\mathbf{z};\phi)$ can approximate the posterior $p(\mathbf{z}|\mathbf{x};\theta)$, the smaller $D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$ we can achieve, the closer ELBO will be to $\log p(\mathbf{x};\theta)$

The Variational EM Algorithm

- $q(\mathbf{z};\phi)$ is an arbitrary probability distribution over $\mathbf{z}$

$$D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta)) = -\sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q(\mathbf{z};\phi))$$

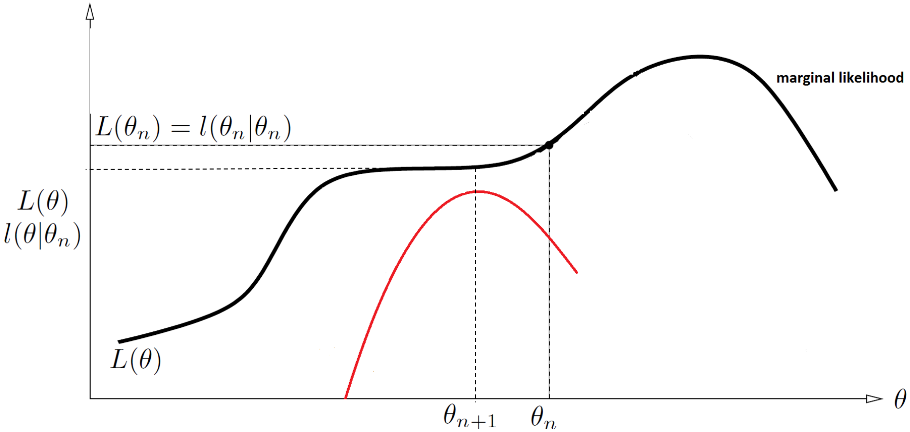
- Re-arranging can rewrite as

$$H(q(\mathbf{z};\phi)) + \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) = \log p(\mathbf{x};\theta) - D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta)) = F[\phi,\theta]$$

- Variational EM as coordinate ascent on $F[\phi,\theta]$
  1. Initialize $\theta^{(0)}$
  2. $\phi^{(1)} = \arg\max_{\phi} F[\phi,\theta^{(0)}] \neq p(\mathbf{z}|\mathbf{x};\theta^{(0)})$ in general
  3. $\theta^{(1)} = \arg\max_{\theta} F[\phi^{(1)},\theta]$ (**can** do Monte Carlo!)
  4. $\phi^{(2)} = \arg\max_{\phi} F[\phi,\theta^{(1)}] \neq p(\mathbf{z}|\mathbf{x};\theta^{(1)})$ in general
  5. ...

- Unlike EM, variational EM **not** guaranteed to reach a local maximum. Marginal likelihood might decrease!

Potential issues with Variational EM algorithm



(Figure adapted from tutorial by Sean Borman)

Summary

- Latent Variable Models Pros:
  - Easy to build flexible models
  - Suitable for unsupervised learning
- Latent Variable Models Cons:
  - Hard to evaluate likelihoods
  - Hard to train via maximum-likelihood
  - Fundamentally, the challenge is that posterior inference $p(\mathbf{z} \mid \mathbf{x})$ is hard. Typically requires variational approximations
- Alternative: give up on KL-divergence and likelihood (GANs)