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

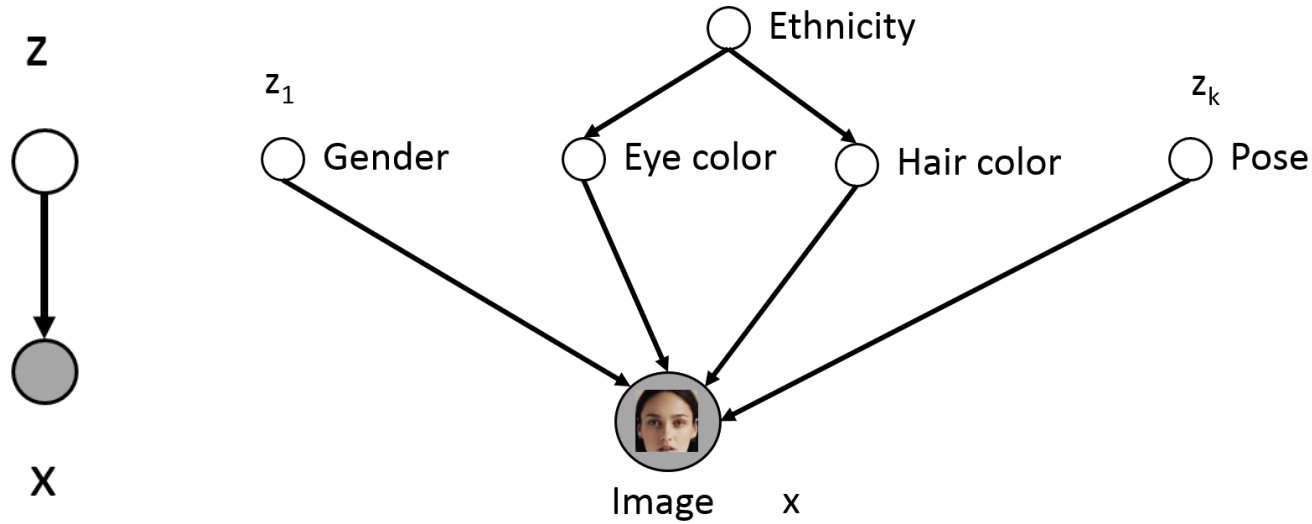
Plan for today

1. Latent Variable Models
   - Mixture models
   - Variational autoencoder
   - Variational inference and learning

Latent Variable Models: Motivation

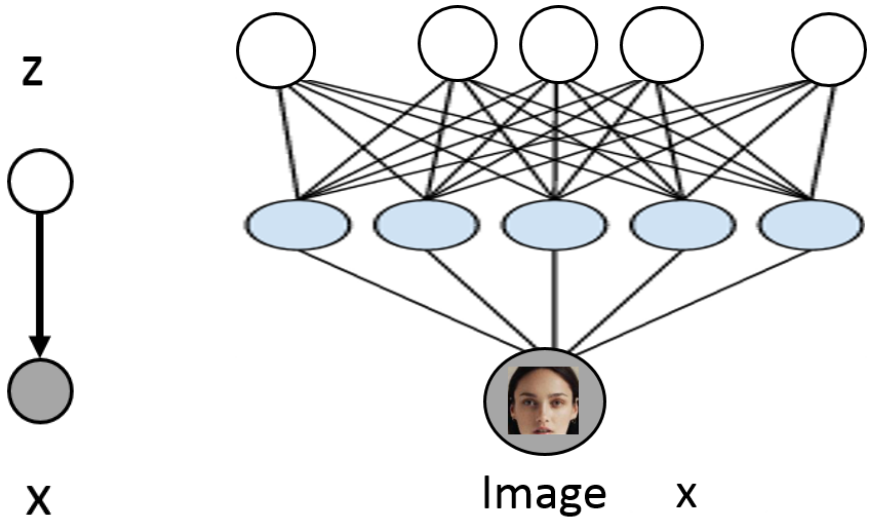
1. Lots of variability in images **x** due to gender, eye color, hair color, pose, etc. However, unless images are annotated, these factors of variation are not explicitly available (latent).
2. **Idea**: explicitly model these factors using latent variables **z**

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., $p(\mathit{EyeColor} = \mathit{Blue}|\mathbf{x})$
3. **Challenge:** Very difficult to specify these conditionals by hand

Deep Latent Variable Models

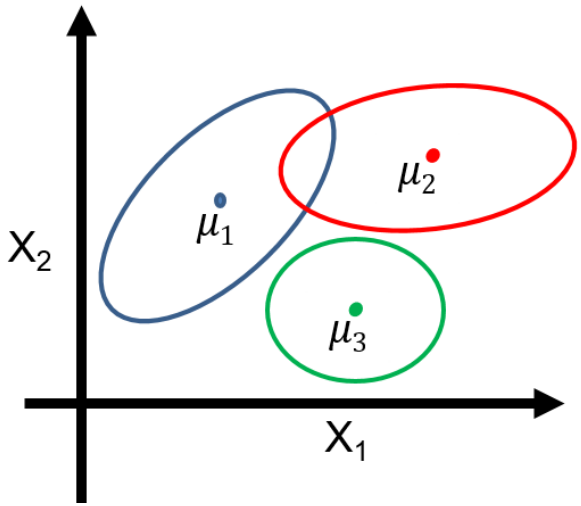


1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. *Hope* that after training, $\mathbf{z}$ will correspond to meaningful latent factors of variation (*features*). Unsupervised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



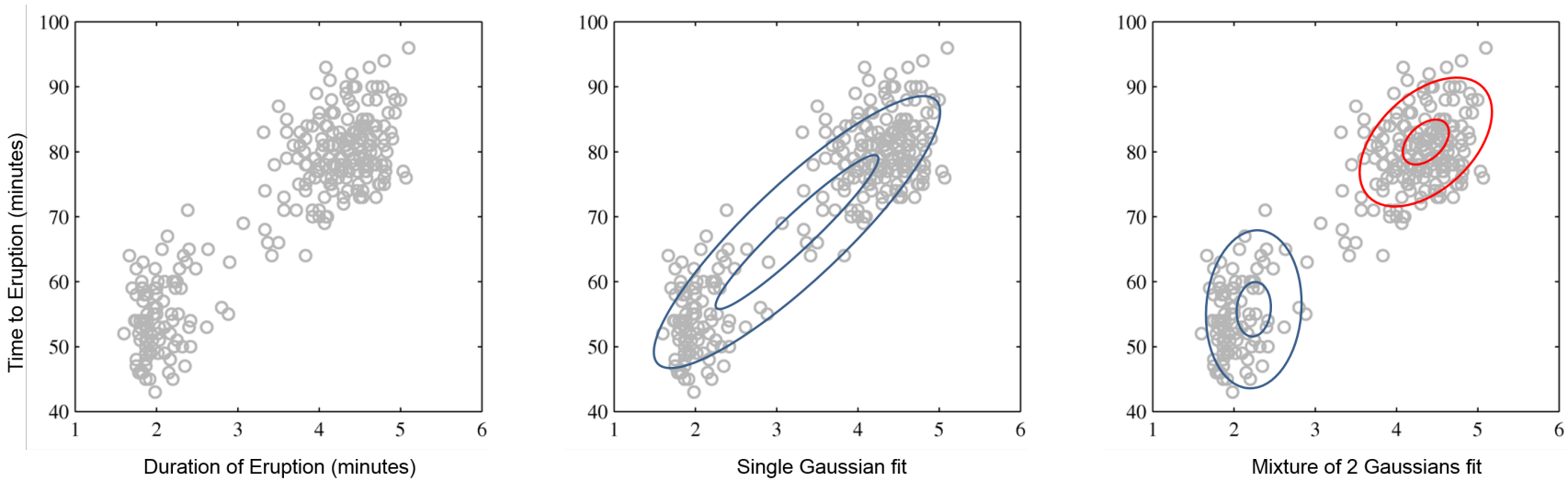
Generative process

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

Mixture of Gaussians: a Shallow Latent Variable Model

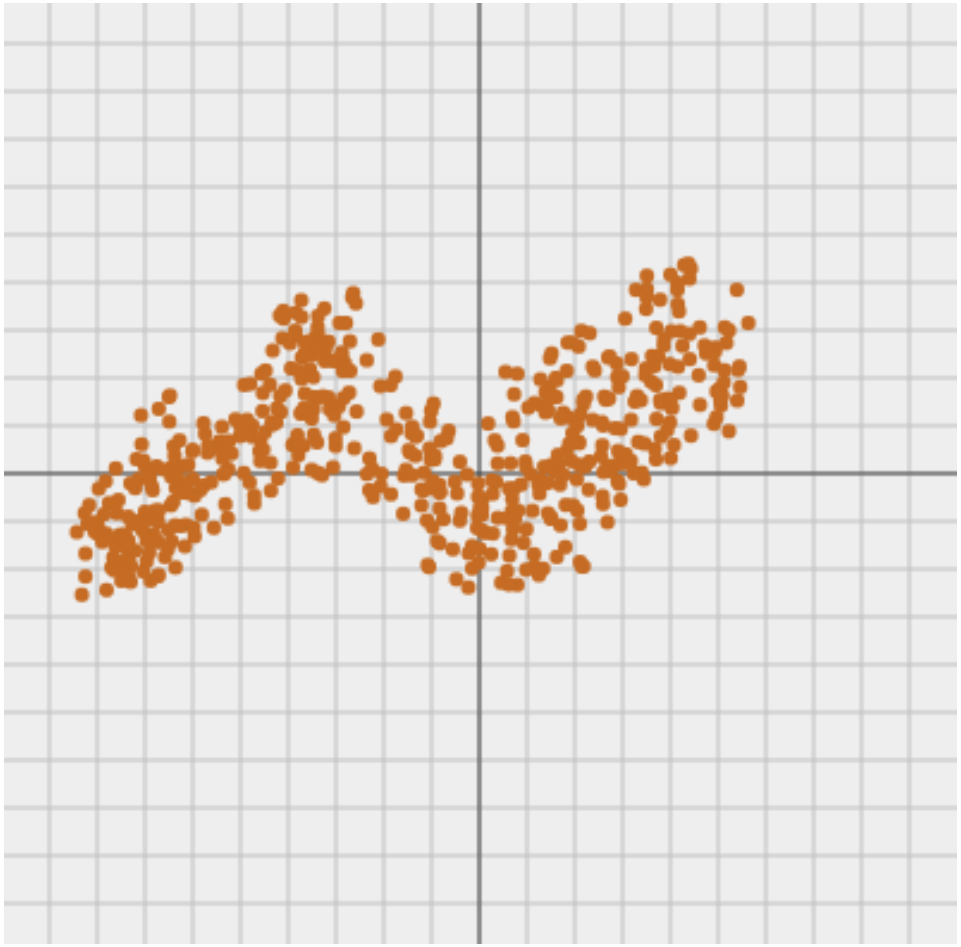
Mixture of Gaussians:

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

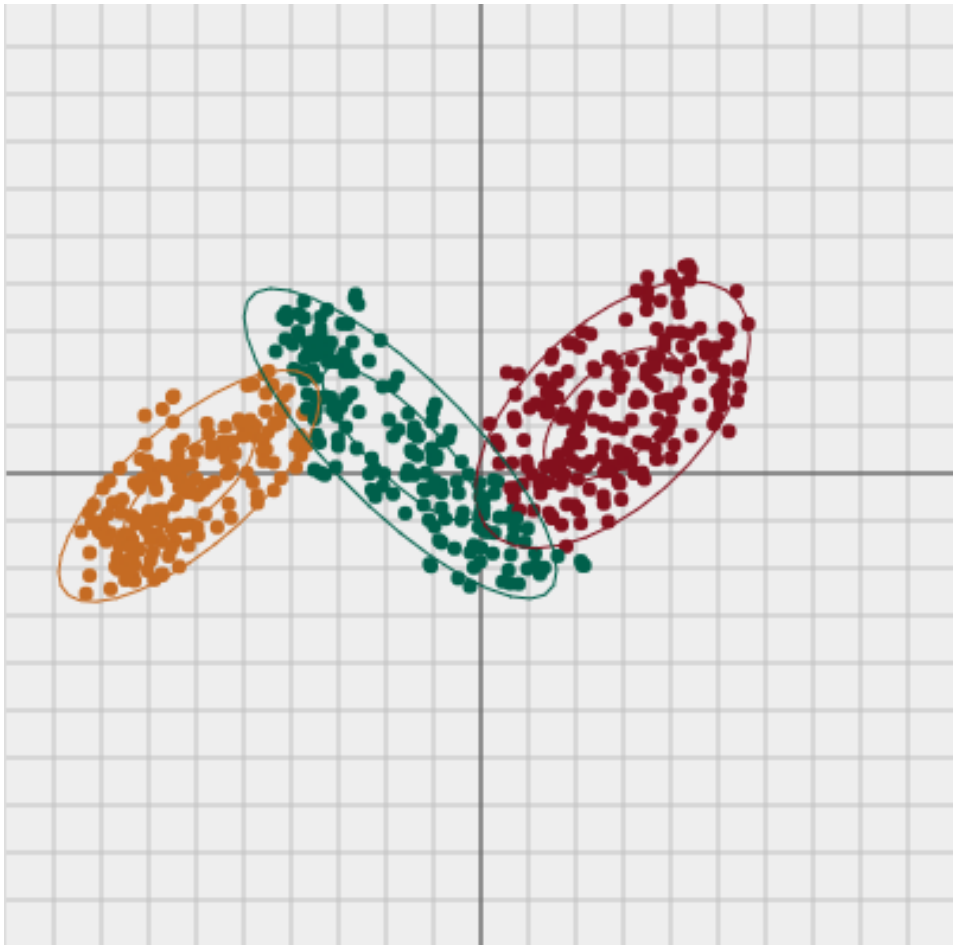


3. **Clustering:** The posterior $p(\mathbf{z} \mid \mathbf{x})$ identifies the mixture component
4. **Unsupervised learning:** We are hoping to learn from unlabeled data (ill-posed problem)

Unsupervised learning

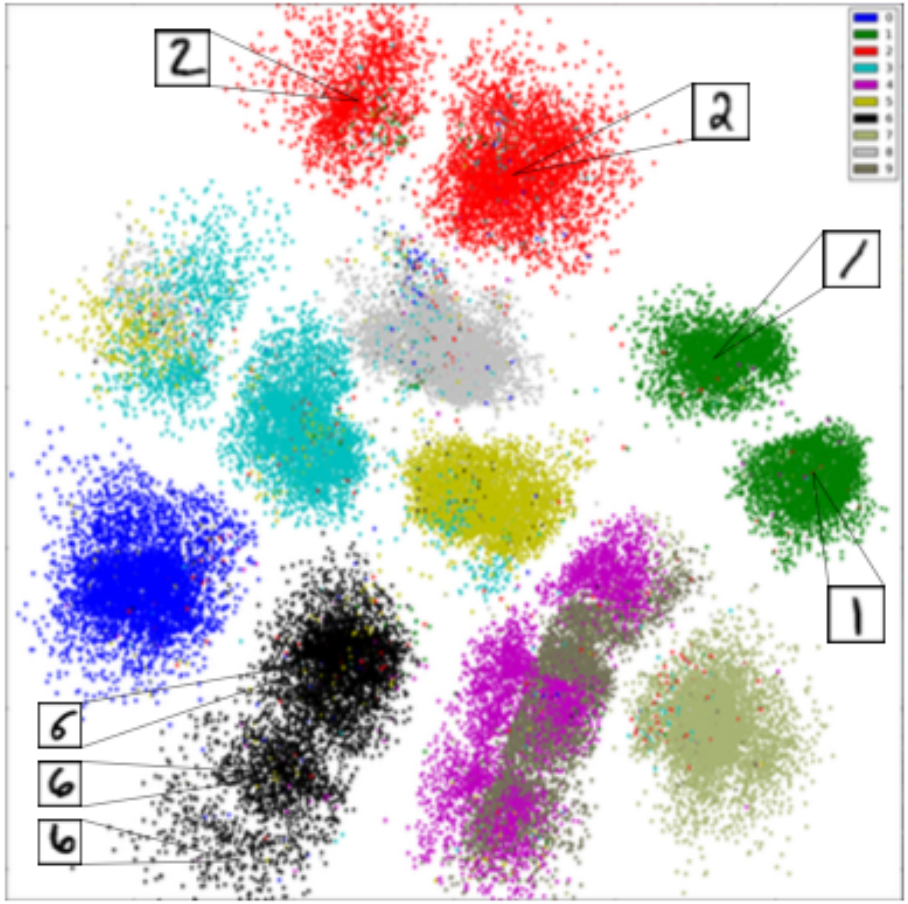


Unsupervised learning



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

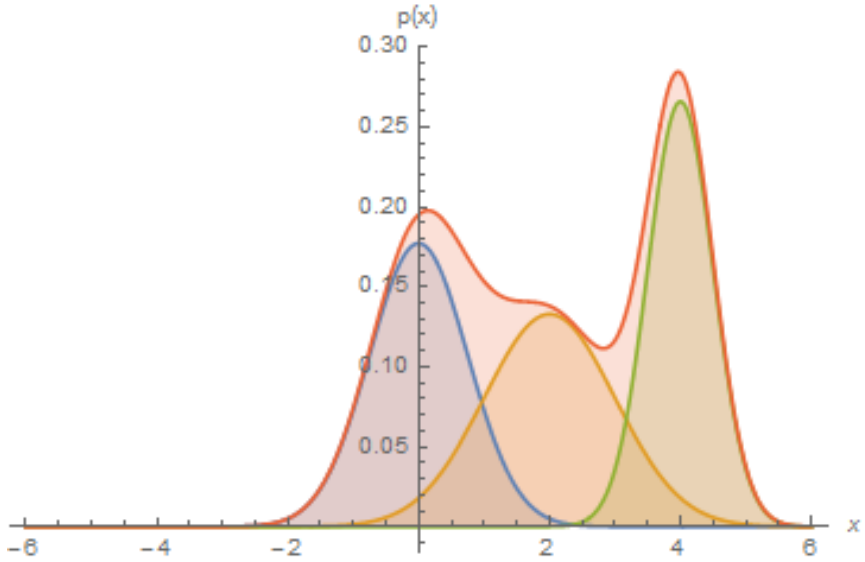
Unsupervised learning



Unsupervised clustering of handwritten digits.

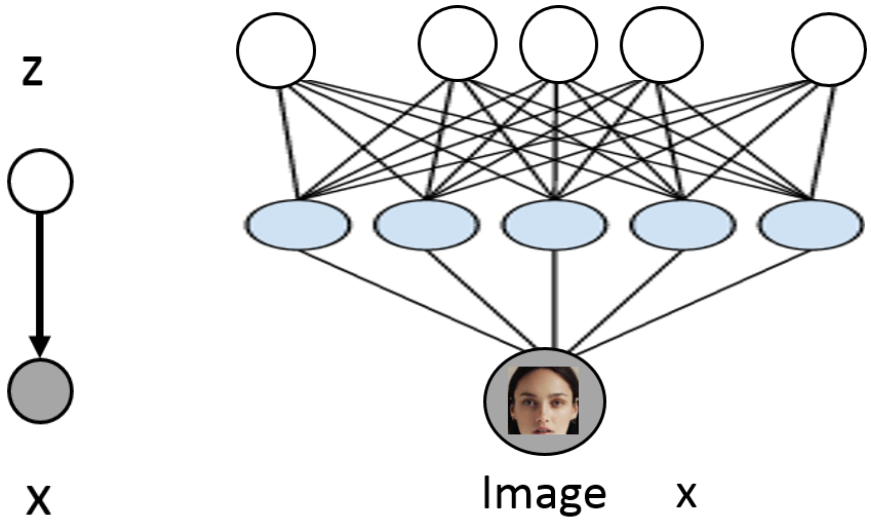
Mixture models

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. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ 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)

Variational Autoencoder Marginal Likelihood

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
3. $\mathbf{Z}$ are unobserved at train time (also called hidden or latent)
4. Suppose we have a model for the joint distribution. What is the probability $p(\mathbf{X} = \bar{\mathbf{x}}; \theta)$ of observing a training data point $\bar{\mathbf{x}}$?

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

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:

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

- 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. Gradients ∇_θ also hard to compute.
- Need **approximations**. One gradient evaluation per training data point $\mathbf{x} \in \mathcal{D}$, so approximation needs to be cheap.

First attempt: Naive Monte Carlo

Likelihood function $p_\theta(\mathbf{x})$ for Partially Observed Data is hard to compute:

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

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.

Second attempt: Importance Sampling

Likelihood function $p_\theta(\mathbf{x})$ for Partially Observed Data is hard to compute:

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

Monte Carlo to the rescue:

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

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

What is a good choice for $q(\mathbf{z})$? Intuitively, choose likely completions. It would then be tempting to estimate the *log*-likelihood as:

$$\log\left(p_\theta(\mathbf{x})\right) \approx \log\left(\frac{1}{k} \sum_{j=1}^{k} \frac{p_\theta(\mathbf{x}, \mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})} \right) \overset{k=1}{\approx} \log\left(\frac{p_\theta(\mathbf{x}, \mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)})} \right)$$

However, it's clear that $\mathbb{E}_{\mathbf{z}^{(1)} \sim q(\mathbf{z})} \left[\log\left(\frac{p_\theta(\mathbf{x}, \mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)})} \right) \right] \neq \log\left(\mathbb{E}_{\mathbf{z}^{(1)} \sim q(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)})} \right] \right)$

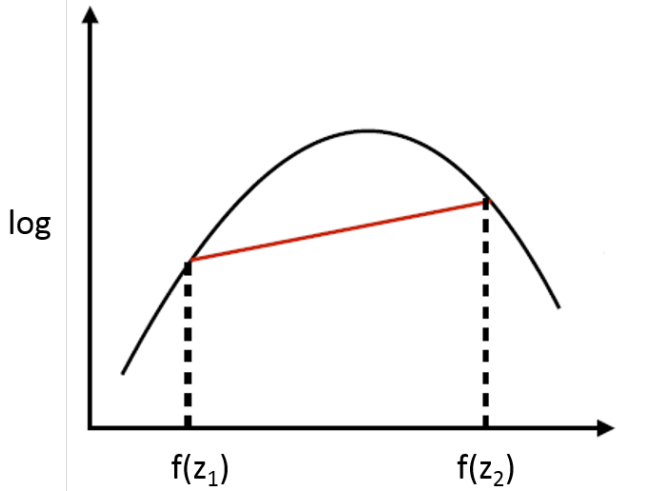
Evidence Lower Bound

Log-Likelihood function for Partially Observed Data is hard to compute:

$$\log\left(\sum_{\mathbf{z}\in\mathcal{Z}} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\sum_{\mathbf{z}\in\mathcal{Z}} \frac{q(\mathbf{z})}{q(\mathbf{z})} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right)$$

- log() is a concave function. $\log(px + (1-p)x') \geq p\log(x) + (1-p)\log(x')$.
- Idea: use Jensen Inequality (for concave functions)

$$\log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[f(\mathbf{z})\right]\right) = \log\left(\sum_{\mathbf{z}} q(\mathbf{z}) f(\mathbf{z})\right) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log f(\mathbf{z})$$



Evidence Lower Bound

Log-Likelihood function for Partially Observed Data is hard to compute:

$$\log\left(\sum_{\mathbf{z}\in\mathcal{Z}} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\sum_{\mathbf{z}\in\mathcal{Z}} \frac{q(\mathbf{z})}{q(\mathbf{z})} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right)$$

- log() is a concave function. $\log(px + (1-p)x') \geq p\log(x) + (1-p)\log(x')$.
- Idea: use Jensen Inequality (for concave functions)

$$\log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[f(\mathbf{z})\right]\right) = \log\left(\sum_{\mathbf{z}} q(\mathbf{z}) f(\mathbf{z})\right) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log f(\mathbf{z})$$

Choosing $f(\mathbf{z}) = \frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}$

$$\log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right) \geq \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right)\right]$$

Called Evidence Lower Bound (**ELBO**).

Variational inference

- Suppose $q(\mathbf{z})$ is **any** probability distribution over the hidden variables
- **Evidence lower bound** (ELBO) holds for any q

$$\begin{aligned}
\log p(\mathbf{x};\theta) &\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} \right) \\
&= \sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z}) \underbrace{- \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})}_{\text{Entropy } H(q) \text{ of } q} \\
&= \sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z}) + H(q)
\end{aligned}$$

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

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

- (Aside: This is what we compute in the E-step of the EM algorithm)

Why is the bound tight

- We derived this lower bound that holds holds for any choice of $q(\mathbf{z})$:

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

- If $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta)$ the bound becomes:

$$\begin{aligned}
\sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta) \log \frac{p(\mathbf{x},\mathbf{z};\theta)}{p(\mathbf{z}|\mathbf{x};\theta)} &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta) \log \frac{p(\mathbf{z}|\mathbf{x};\theta)p(\mathbf{x};\theta)}{p(\mathbf{z}|\mathbf{x};\theta)} \\
&= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta) \log p(\mathbf{x};\theta) \\
&= \log p(\mathbf{x};\theta) \underbrace{\sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta)}_{=1} \\
&= \log p(\mathbf{x};\theta)
\end{aligned}$$

- Confirms our previous importance sampling intuition: we should choose likely completions.
- What if the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute? How loose is the bound?

Variational inference continued

- Suppose $q(\mathbf{z})$ is **any** probability distribution over the hidden variables. A little bit of algebra reveals

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

- Rearranging, we re-derived the **Evidence lower bound** (ELBO)

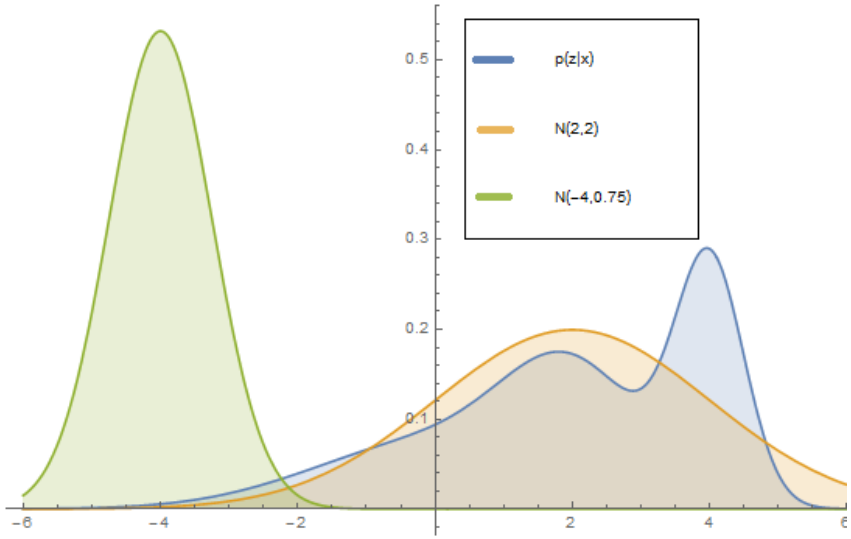
$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q)$$

- Equality holds if $q = p(\mathbf{z}|\mathbf{x};\theta)$ because $D_{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x};\theta)) = 0$

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

- In general, $\log p(\mathbf{x};\theta) = \mathrm{ELBO} + D_{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x};\theta))$. The closer $q(\mathbf{z})$ is to $p(\mathbf{z}|\mathbf{x};\theta)$, the closer the ELBO is to the true log-likelihood

The Evidence Lower bound



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

$$q(\mathbf{z};\phi) = \mathcal{N}(\phi_1, \phi_2)$$

- **Variational inference**: pick ϕ so that $q(\mathbf{z};\phi)$ is as close as possible to $p(\mathbf{z}|\mathbf{x};\theta)$. In the figure, the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ (blue) is better approximated by $\mathcal{N}(2,2)$ (orange) than $\mathcal{N}(-4,0.75)$ (green)

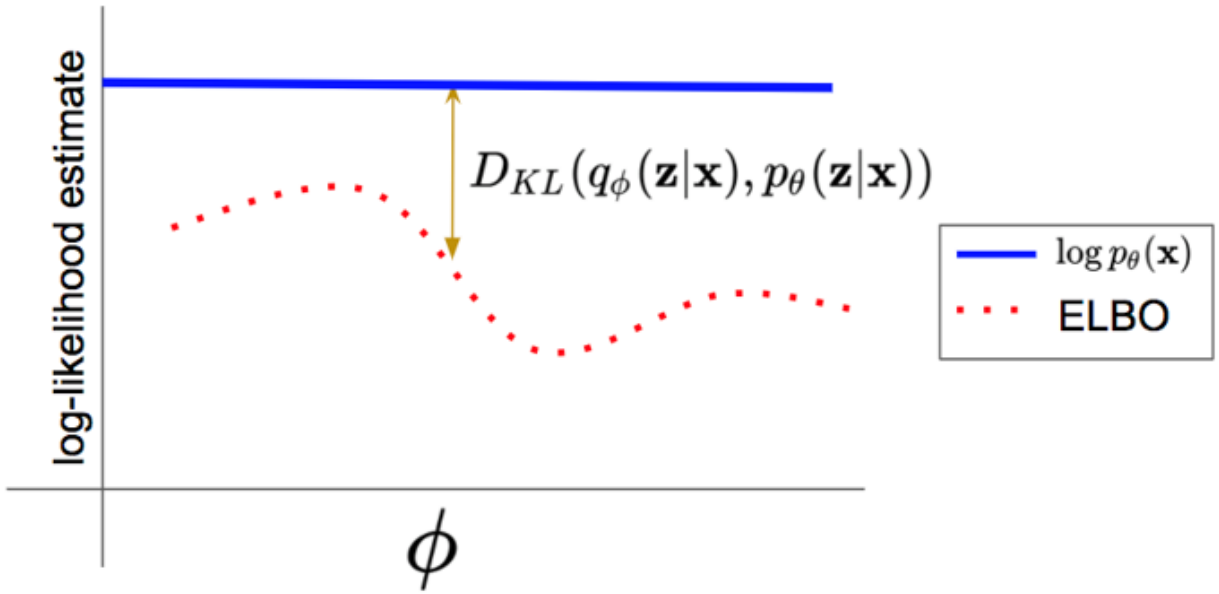
A variational approximation to the posterior

- Assume $p(\mathbf{x}^{top}, \mathbf{x}^{bottom}; \theta)$ assigns high probability to images that look like digits. In this example, we assume $\mathbf{z} = \mathbf{x}^{top}$ are unobserved (latent)
- Suppose $q(\mathbf{x}^{top}; \phi)$ is a (tractable) probability distribution over the hidden variables (missing pixels in this example) $\mathbf{x}^{top}$ parameterized by ϕ (variational parameters)

$$q(\mathbf{x}^{top}; \phi) = \prod_{\text{unobserved variables } \mathbf{x}_i^{top}} (\phi_i)^{\mathbf{x}_i^{top}} (1 - \phi_i)^{(1-\mathbf{x}_i^{top})}$$

- Is $\phi_i = 0.5 \ \forall i$ a good approximation to the posterior $p(\mathbf{x}^{top} | \mathbf{x}^{bottom}; \theta)$? No
- Is $\phi_i = 1 \ \forall i$ a good approximation to the posterior $p(\mathbf{x}^{top} | \mathbf{x}^{bottom}; \theta)$? No
- Is $\phi_i \approx 1$ for pixels i corresponding to the top part of digit **9** a good approximation? Yes

The Evidence Lower bound



$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

$$= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$$

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 $\log p(\mathbf{x};\theta)$. Next: jointly optimize over θ and ϕ to maximize the ELBO over a dataset

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)

Latent Variable Models

Stefano Ermon, Aditya Grover

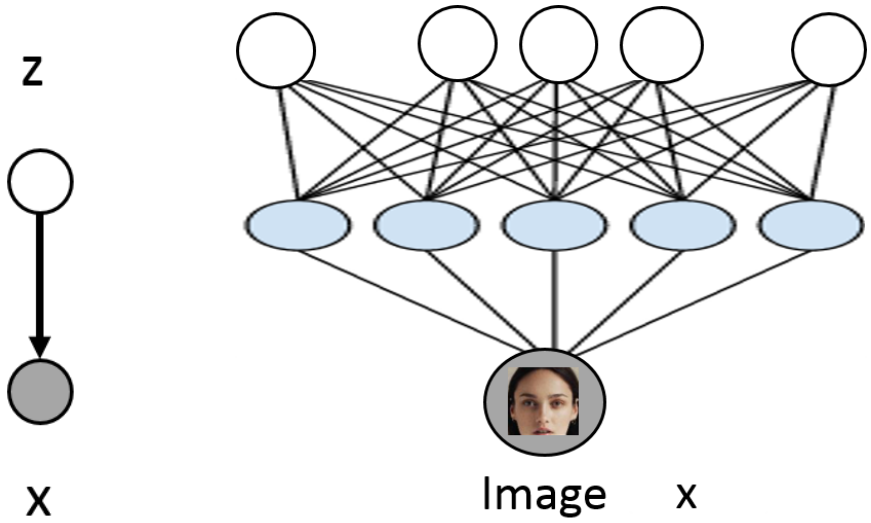
Stanford University

Lecture 6

Plan for today

1. Latent Variable Models
   - Learning deep generative models
   - Stochastic optimization:
     - Reparameterization trick
   - Inference Amortization

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
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

Recap: Variational Inference

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

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

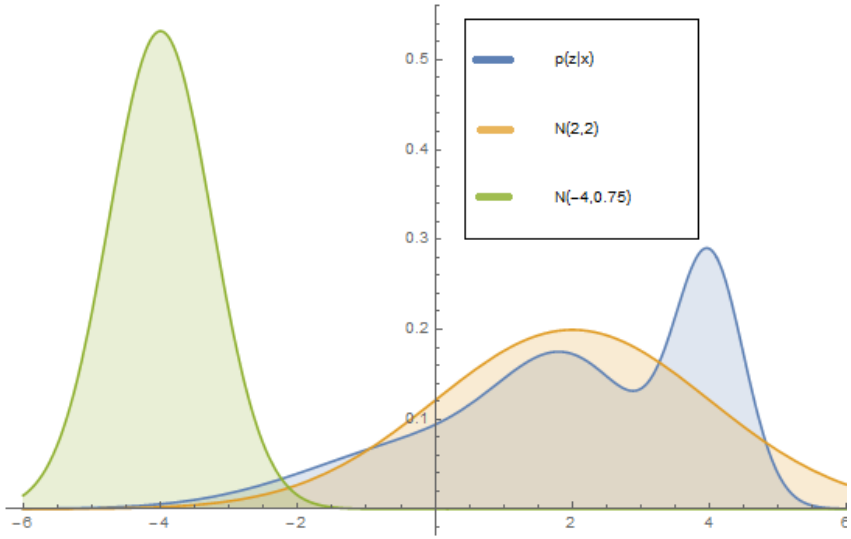
- **Evidence lower bound** (ELBO) holds for any q

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

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

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

Recap: The Evidence Lower bound

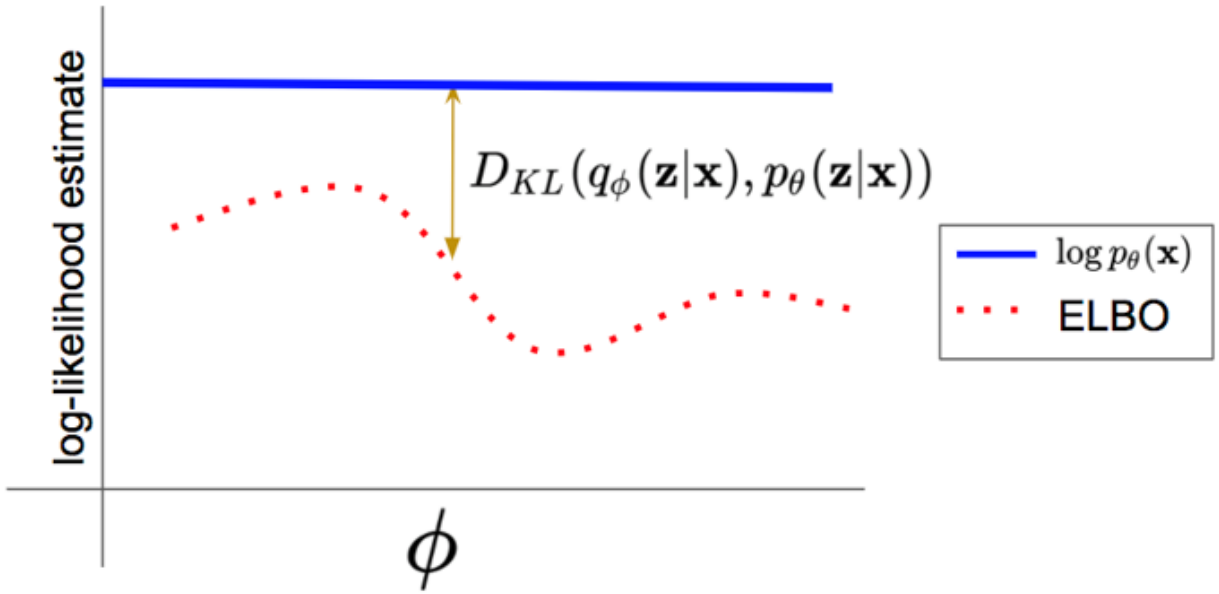


- What if the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute?
- Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - For example, a Gaussian with mean and covariance specified by ϕ

$$q(\mathbf{z};\phi) = \mathcal{N}(\phi_1, \phi_2)$$

- **Variational inference**: pick ϕ so that $q(\mathbf{z};\phi)$ is as close as possible to $p(\mathbf{z}|\mathbf{x};\theta)$. In the figure, the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ (blue) is better approximated by $\mathcal{N}(2,2)$ (orange) than $\mathcal{N}(-4,0.75)$ (green)

Recap: The Evidence Lower bound

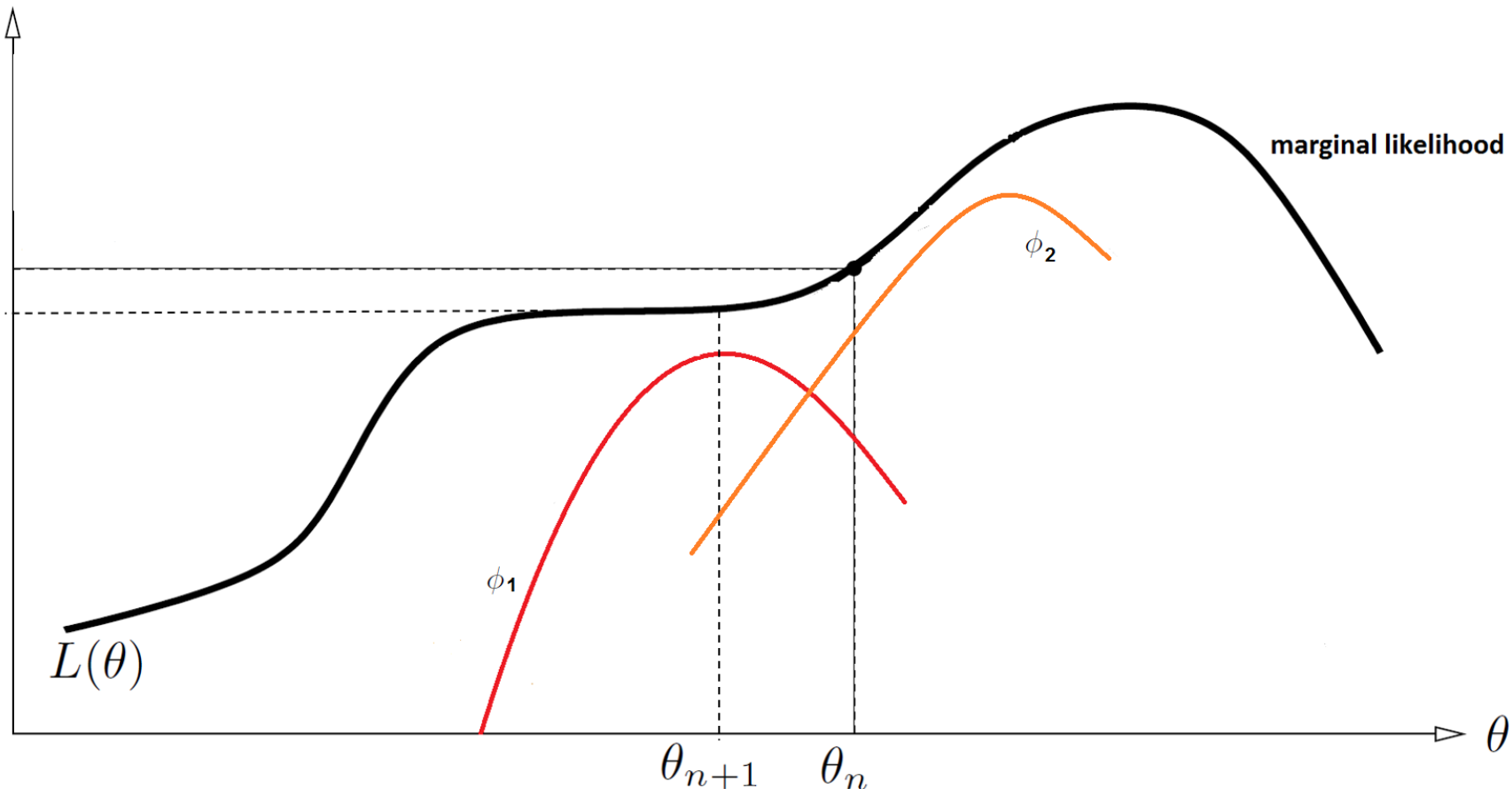


$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

$$= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$$

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 $\log p(\mathbf{x};\theta)$. Next: jointly optimize over θ and ϕ to maximize the ELBO over a dataset

Variational learning



and $\mathcal{L}(\mathbf{x}; \theta, \phi_2)$ are both lower bounds. We want to jointly optimize θ and ϕ

The Evidence Lower bound applied to the entire dataset

- Evidence lower bound (ELBO) holds for any $q(\mathbf{z}; \phi)$

$$\log p(\mathbf{x}; \theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) = \underbrace{\mathcal{L}(\mathbf{x}; \theta, \phi)}_{\text{ELBO}}$$

- Maximum likelihood learning (over the entire dataset):

$$\ell(\theta; \mathcal{D}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log p(\mathbf{x}^i; \theta) \geq \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

- Therefore

$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \cdots, \phi^M} \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

- Note that we use different *variational parameters* ϕ^i for every data point $\mathbf{x}^i$, because the true posterior $p(\mathbf{z}|\mathbf{x}^i; \theta)$ is different across datapoints $\mathbf{x}^i$

A variational approximation to the posterior

- Assume $p(\mathbf{z}, \mathbf{x}^i; \theta)$ is close to $p_{\text{data}}(\mathbf{z}, \mathbf{x}^i)$. Suppose $\mathbf{z}$ captures information such as the digit identity (label), style, etc. For simplicity, assume $\mathbf{z} \in \{0, 1, 2, \cdots, 9\}$.
- Suppose $q(\mathbf{z}; \phi^i)$ is a (categorical) probability distribution over the hidden variable $\mathbf{z}$ parameterized by $\phi^i = [p_0, p_1, \cdots, p_9]$

$$q(\mathbf{z}; \phi^i) = \prod_{k \in \{0,1,2,\cdots,9\}} (\phi_k^i)^{1[\mathbf{z}=k]}$$

- If $\phi^i = [0, 0, 0, 1, 0, \cdots, 0]$, is $q(\mathbf{z}; \phi^i)$ a good approximation of $p(\mathbf{z}|\mathbf{x}^1; \theta)$ ($\mathbf{x}^1$ is the leftmost datapoint)? Yes
- If $\phi^i = [0, 0, 0, 1, 0, \cdots, 0]$, is $q(\mathbf{z}; \phi^i)$ a good approximation of $p(\mathbf{z}|\mathbf{x}^3; \theta)$ ($\mathbf{x}^3$ is the rightmost datapoint)? No
- For each $\mathbf{x}^i$, need to find a good $\phi^{i,*}$ (via optimization, can be expensive).

Learning via stochastic variational inference (SVI)

- Optimize $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$ as a function of $\theta, \phi^1, \cdots, \phi^M$ using (stochastic) gradient descent

$$\begin{aligned}\mathcal{L}(\mathbf{x}^i; \theta, \phi^i) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi^i) \log p(\mathbf{z}, \mathbf{x}^i; \theta) + H(q(\mathbf{z}; \phi^i)) \\ &= E_{q(\mathbf{z};\phi^i)}[\log p(\mathbf{z}, \mathbf{x}^i; \theta) - \log q(\mathbf{z}; \phi^i)]\end{aligned}$$

1. Initialize $\theta, \phi^1, \cdots, \phi^M$
2. Randomly sample a data point $\mathbf{x}^i$ from $\mathcal{D}$
3. Optimize $\mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$ as a function of ϕ^i:
   1. Repeat $\phi^i = \phi^i + \eta \nabla_{\phi^i} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$
   2. until convergence to $\phi^{i,*} \approx \arg\max_\phi \mathcal{L}(\mathbf{x}^i; \theta, \phi)$
4. Compute $\nabla_\theta \mathcal{L}(\mathbf{x}^i; \theta, \phi^{i,*})$
5. Update θ in the gradient direction. Go to step 2

- How to compute the gradients? There might not be a closed form solution for the expectations. So we use Monte Carlo sampling

Learning Deep Generative models

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) \\ &= E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)]\end{aligned}$$

- Note: dropped i superscript from ϕ^i for compactness
- To *evaluate* the bound, sample $\mathbf{z}^1, \cdots, \mathbf{z}^k$ from $q(\mathbf{z};\phi)$ and estimate

$$E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)] \approx \frac{1}{k}\sum_{k} \log p(\mathbf{z}^k,\mathbf{x};\theta) - \log q(\mathbf{z}^k;\phi))$$

- Key assumption: $q(\mathbf{z};\phi)$ is tractable, i.e., easy to sample from and evaluate
- Want to compute $\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$
- The gradient with respect to θ is easy

$$\begin{aligned}\nabla_\theta E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)] &= E_{q(\mathbf{z};\phi)}[\nabla_\theta \log p(\mathbf{z},\mathbf{x};\theta)] \\ &\approx \frac{1}{k}\sum_{k} \nabla_\theta \log p(\mathbf{z}^k,\mathbf{x};\theta)\end{aligned}$$

Learning Deep Generative models

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) \\ &= E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)]\end{aligned}$$

- Want to compute $\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$
- The gradient with respect to ϕ is more complicated because the expectation depends on ϕ
- We still want to estimate with a Monte Carlo average
- Later in the course we'll see a general technique called REINFORCE (from reinforcement learning)
- For now, a better but less general alternative that only works for continuous $\mathbf{z}$ (and only some distributions)

Reparameterization

- Want to compute a gradient with respect to ϕ of

$$E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \int q(\mathbf{z};\phi) r(\mathbf{z}) d\mathbf{z}$$

  where $\mathbf{z}$ is now **continuous**

- Suppose $q(\mathbf{z};\phi) = \mathcal{N}(\mu, \sigma^2 I)$ is Gaussian with parameters $\phi = (\mu, \sigma)$. These are equivalent ways of sampling:
  - Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
  - Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon = g(\epsilon;\phi)$
- Using this equivalence we compute the expectation in two ways:

$$E_{\mathbf{z}\sim q(\mathbf{z};\phi)}[r(\mathbf{z})] = E_{\epsilon\sim\mathcal{N}(0,I)}[r(g(\epsilon;\phi))] = \int p(\epsilon) r(\mu + \sigma\epsilon) d\epsilon$$

$$\nabla_\phi E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \nabla_\phi E_\epsilon[r(g(\epsilon;\phi))] = E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))]$$

- Easy to estimate via Monte Carlo if r and g are differentiable w.r.t. ϕ and ϵ is easy to sample from (backpropagation)
- $E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))] \approx \frac{1}{k}\sum_k \nabla_\phi r(g(\epsilon^k;\phi))$ where $\epsilon^1, \cdots, \epsilon^k \sim \mathcal{N}(0, I)$.
- Typically much lower variance than REINFORCE

Learning Deep Generative models

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) \\ &= E_{q(\mathbf{z};\phi)}[\underbrace{\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)}_{r(\mathbf{z},\phi)}]\end{aligned}$$

- Our case is slightly more complicated because we have $E_{q(\mathbf{z};\phi)}[r(\mathbf{z},\phi)]$ instead of $E_{q(\mathbf{z};\phi)}[r(\mathbf{z})]$. Term inside the expectation also depends on ϕ.
- Can still use reparameterization. Assume $\mathbf{z} = \mu + \sigma\epsilon = g(\epsilon;\phi)$ like before. Then

$$\begin{aligned}E_{q(\mathbf{z};\phi)}[r(\mathbf{z},\phi)] &= E_{\epsilon}[r(g(\epsilon;\phi),\phi)] \\ &\approx \frac{1}{k}\sum_{k} r(g(\epsilon^k;\phi),\phi)\end{aligned}$$

Amortized Inference

$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \cdots, \phi^M} \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

- So far we have used a set of variational parameters ϕ^i for each data point $\mathbf{x}^i$. Does not scale to large datasets.
- **Amortization:** Now we learn a **single** parametric function f_λ that maps each $\mathbf{x}$ to a set of (good) variational parameters. Like doing regression on $\mathbf{x}^i \mapsto \phi^{i,*}$
  - For example, if $q(\mathbf{z}|\mathbf{x}^i)$ are Gaussians with different means $\mu^1, \cdots, \mu^m$, we learn a **single** neural network f_λ mapping $\mathbf{x}^i$ to μ^i
- We approximate the posteriors $q(\mathbf{z}|\mathbf{x}^i)$ using this distribution $q_\lambda(\mathbf{z}|\mathbf{x})$

A variational approximation to the posterior

- Assume $p(\mathbf{z}, \mathbf{x}^i; \theta)$ is close to $p_{\text{data}}(\mathbf{z}, \mathbf{x}^i)$. Suppose $\mathbf{z}$ captures information such as the digit identity (label), style, etc.
- Suppose $q(\mathbf{z}; \phi^i)$ is a (tractable) probability distribution over the hidden variables $\mathbf{z}$ parameterized by ϕ^i
- For each $\mathbf{x}^i$, need to find a good $\phi^{i,*}$ (via optimization, expensive).
- **Amortized inference**: *learn* how to map $\mathbf{x}^i$ to a good set of parameters ϕ^i via $q(\mathbf{z}; f_\lambda(\mathbf{x}^i))$. f_λ learns how to solve the optimization problem for you
- In the literature, $q(\mathbf{z}; f_\lambda(\mathbf{x}^i))$ often denoted $q_\phi(\mathbf{z}|\mathbf{x})$

Learning with amortized inference

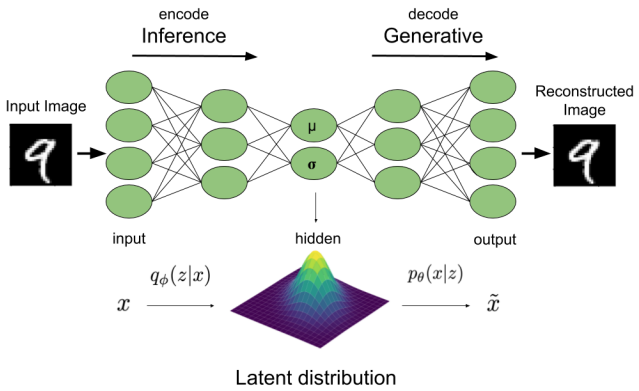
- Optimize $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ as a function of θ, ϕ using (stochastic) gradient descent

$$\begin{aligned} \mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q_\phi(\mathbf{z}|\mathbf{x})) \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \end{aligned}$$

1. Initialize $\theta^{(0)}, \phi^{(0)}$
2. Randomly sample a data point $\mathbf{x}^i$ from $\mathcal{D}$
3. Compute $\nabla_\theta \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x}^i; \theta, \phi)$
4. Update θ, ϕ in the gradient direction

- How to compute the gradients? Use reparameterization like before

Autoencoder perspective



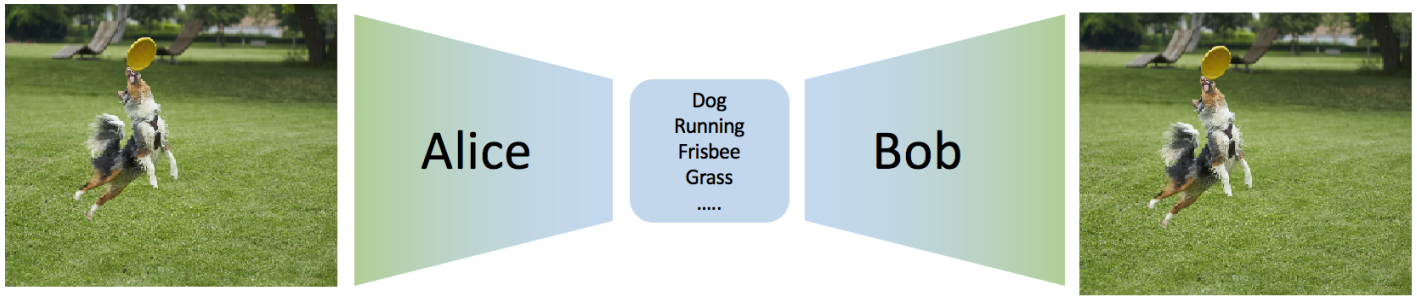
$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\end{aligned}$$

1. Take a data point $\mathbf{x}^i$
2. Map it to $\hat{\mathbf{z}}$ by sampling from $q_\phi(\mathbf{z}|\mathbf{x}^i)$ (*encoder*)
3. Reconstruct $\hat{\mathbf{x}}$ by sampling from $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$ (*decoder*)

What does the training objective $\mathcal{L}(\mathbf{x};\theta,\phi)$ do?

- First term encourages $\hat{\mathbf{x}} \approx \mathbf{x}^i$ ($\mathbf{x}^i$ likely under $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$)
- Second term encourages $\hat{\mathbf{z}}$ to be likely under the prior $p(\mathbf{z})$

Learning Deep Generative models



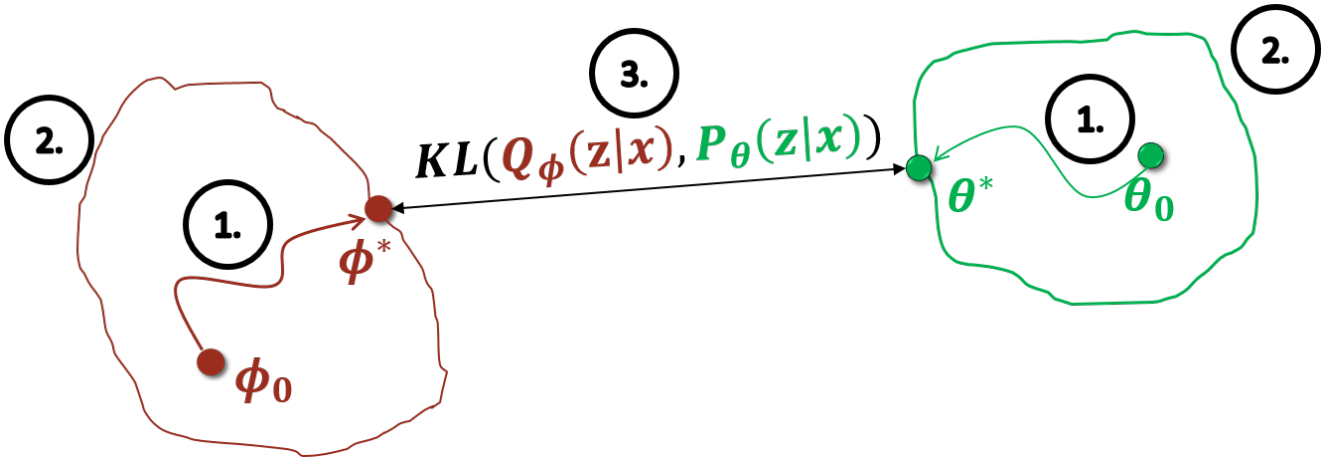
1. Alice goes on a space mission and needs to send images to Bob. Given an image $\mathbf{x}^i$, she (stochastically) compresses it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. Alice sends the message $\hat{\mathbf{z}}$ to Bob
2. Given $\hat{\mathbf{z}}$, Bob tries to reconstruct the image using $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$

- This scheme works well if $E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]$ is large
- The term $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ forces the distribution over messages to have a specific shape $p(\mathbf{z})$. If Bob knows $p(\mathbf{z})$, he can generate realistic messages $\hat{\mathbf{z}} \sim p(\mathbf{z})$ and the corresponding image, as if he had received them from Alice!

Summary of Latent Variable Models

1. Combine simple models to get a more flexible one (e.g., mixture of Gaussians)
2. Directed model permits ancestral sampling (efficient generation): $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z};\theta)$
3. However, log-likelihood is generally intractable, hence learning is difficult
4. Joint learning of a model (θ) and an amortized inference component (ϕ) to achieve tractability via ELBO optimization
5. Latent representations for any $\mathbf{x}$ can be inferred via $q_\phi(\mathbf{z}|\mathbf{x})$

Research Directions



Improving variational learning via:

1. Better optimization techniques
2. More expressive approximating families
3. Alternate loss functions

Model families - Encoder

Amortization (Gershman & Goodman, 2015; Kingma; Rezende; ..)

- Scalability: Efficient learning and inference on massive datasets
- Regularization effect: Because of joint training, it also implicitly regularizes the model θ (Shu et al., 2018)

Augmenting variational posteriors

- Monte Carlo methods: Importance Sampling (Burda et al., 2015), MCMC (Salimans et al., 2015, Hoffman, 2017, Levy et al., 2018), Sequential Monte Carlo (Maddison et al., 2017, Le et al., 2018, Naesseth et al., 2018), Rejection Sampling (Grover et al., 2018)
- Normalizing flows (Rezende & Mohammed, 2015, Kingma et al., 2016)

Model families - Decoder

- Powerful decoders $p(\mathbf{x}|\mathbf{z};\theta)$ such as DRAW (Gregor et al., 2015), PixelCNN (Gulrajani et al., 2016)
- Parameterized, learned priors $p(\mathbf{z};\theta)$ (Nalusnick et al., 2016, Tomczak & Welling, 2018, Graves et al., 2018)

Variational objectives

Tighter ELBO does not imply:

- Better samples: Sample quality and likelihoods are uncorrelated (Theis et al., 2016)
- Informative latent codes: Powerful decoders can ignore latent codes due to tradeoff in minimizing reconstruction error vs. KL prior penalty (Bowman et al., 2015, Chen et al., 2016, Zhao et al., 2017, Alemi et al., 2018)

Alternatives to the reverse-KL divergence:

- Renyis alpha-divergences (Li & Turner, 2016)
- Integral probability metrics such as maximum mean discrepancy, Wasserstein distance (Dziugaite et al., 2015; Zhao et. al, 2017; Tolstikhin et al., 2018)