Latent Variable Models

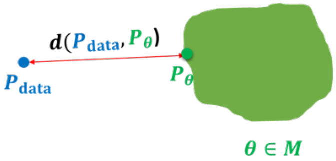
Volodymyr Kuleshov

Cornell Tech

Lecture 5

The Task of Generative Modeling

Suppose we are given a training set of examples, e.g., images of dogs



Model family

Our Goal: define a probability distribution $p(x)$ over images x such that

- **Generation:** If we sample $x_{new} \sim p(x)$, x_{new} should look like a dog.
- **Representation Learning:** We should be able to learn what these images have in common, e.g., ears, tail, etc.
- **Density Estimation:** $p(x)$ should be high if x looks like a dog, and low otherwise (*anomaly detection*)

Previously: Generation and density estimation. Today: Representation learning.

Recap of Previous Lectures

1. Autoregressive models:
   - Probability distributions factorize into a product of factors:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{x}_{<i})$$

   - We can efficiently represent p via *conditional independence* or via compact *neural parameterizations* $p_{\text{Neural}}(x_i \mid \mathbf{x}_{<i})$ of each factor.
2. Pros of autoregressive models:
   - It is computationally tractable to evaluate likelihoods
   - It is tractable to train $p(\mathbf{x})$ via maximum likelihood & gradient descent
3. Cons of autoregressive models:
   - They require choosing an ordering over variables
   - Generation is sequential (hence usually slow)
   - Cannot learn features in an unsupervised way

Lecture Outline

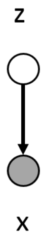
Latent Variable Models: Motivation

Human faces $\mathbf{x}$ can feature a lot of interesting characteristics: gender, age, hair color, eye color, pose, etc.

Challenge: How to automatically learn these characteristics from data?

- Unless the images are annotated, these factors of variation are not explicitly available (latent).
- **Idea**: Explicitly model these factors using latent variables $\mathbf{z}$ and learn them using unsupervised learning.

Latent Variable Models: Definition



A latent variable model defines a probability distribution

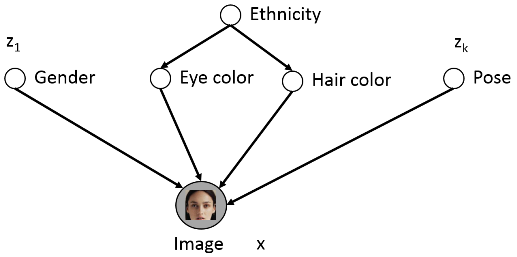
$$p(x, z) = p(x|z)p(z)$$

containing two sets of variables:

1. Observed variables $\mathbf{x}$ that represent the high-dimensional objects we are trying to model and that are in our training set.
2. Latent variables $\mathbf{z}$ that are not in the dataset, but that are associated with $\mathbf{x}$ as specified by $p(\mathbf{x}, \mathbf{z})$. We will learn $\mathbf{z}$ and $p(\mathbf{x}, \mathbf{z})$ jointly.

Latent Variable Models: Example

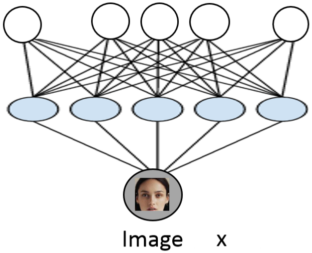
Consider the following distribution $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ over faces:



- The observed variables $\mathbf{x}$ are the images (their pixel values)
- The latent variables $\mathbf{z}$ are discrete features: gender, pose, etc.
  - Note that we can extract features via $p(\mathbf{z} \mid \mathbf{x})$, e.g., $p(Gender = F|\mathbf{x})$
- **Challenges:**
  1. Very difficult to specify conditional $p(\mathbf{x}|\mathbf{z})$ by hand
  2. Unsupervised learning in this model can be intractable

Continuous Latent Variable Representations

Our solution to Challenge #1 involves choosing a continuous ("distributed") representation for $\mathbf{z}$:

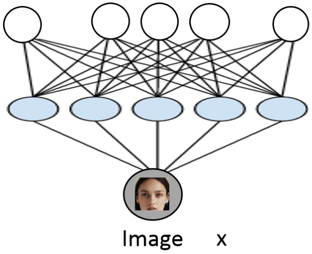


1. The variable $\mathbf{z} \in \mathbb{R}^p$ is a continuous real-valued vector
2. The $\mathbf{z} \to \mathbf{x}$ mapping is specified by a neural net and learned from data

The $\mathbf{z}$ form a smooth parameterization of faces $\mathbf{x}$. Similar faces (same age, gender) have similar $\mathbf{z}$. We can map $\mathbf{x}$ to $\mathbf{z}$, interpolate between $\mathbf{z}$, generate faces from new $\mathbf{z}$, etc.

Deep Latent Variable Models: Example

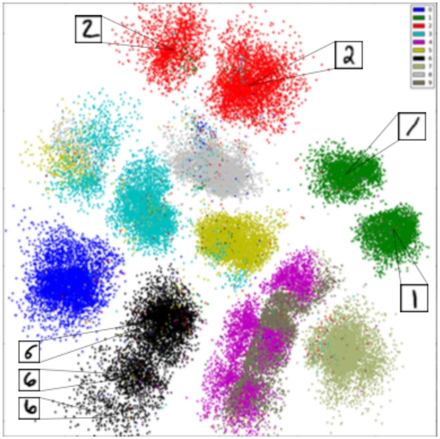
Here is an example of how a deep latent variable model with a continuous latent representation **z** might look like:



1. The prior $p(\mathbf{z})$ is a p-dimensional Gaussian $\mathcal{N}(0, I_p)$
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

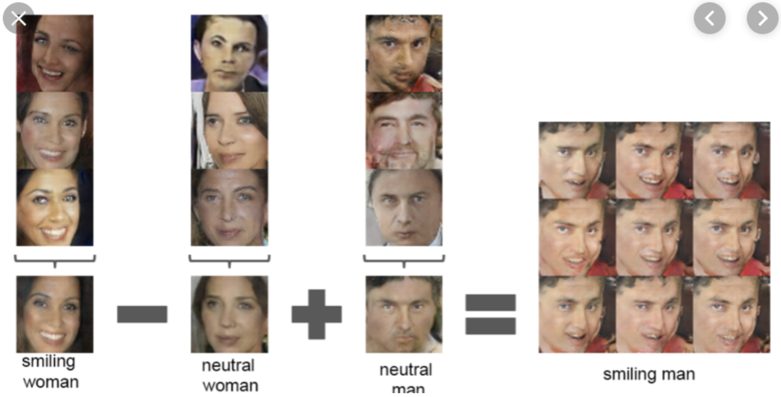
Learning such models is still challenging.

Example: Unsupervised Learning Over Handwritten Digits



Unsupervised clustering of MNIST digits using deep latent variable models.

Example: Unsupervised Learning over Face Images



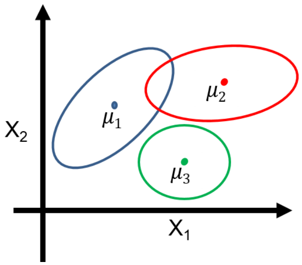
Benefits of the Latent Variable Approach

- Allows 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

Lecture Outline

Shallow Latent-Variable Models: Gaussian Mixture Models

The classical latent variable model $p(\mathbf{x}, z)$ is the mixture of Gaussians:

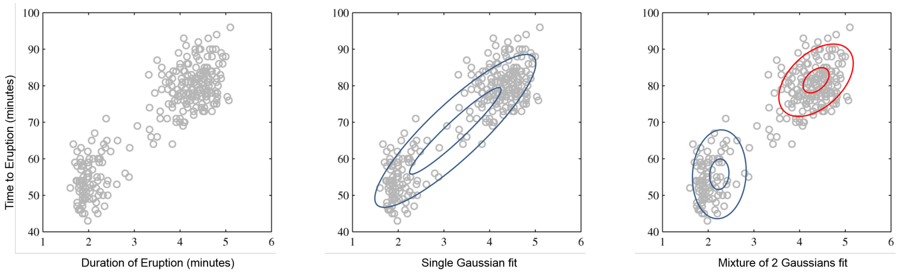


1. The prior is $p(z) = \mathrm{Categorical}(1, \cdots, K)$
2. The conditional is $p(\mathbf{x} \mid z = k) = \mathcal{N}(\mu_k, \Sigma_k)$

This assumes the data is a mixture of K unknown Gaussian clusters.

Representational Power of Mixture Models

Mixtures of Gaussians can represent distributions that single Gaussians cannot, e.g.: the two-component structure of this data:

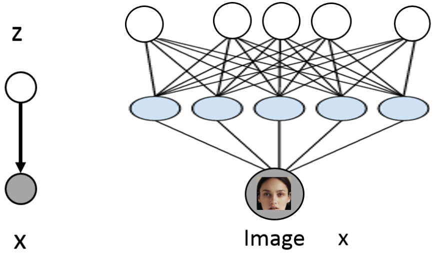


Mixtures of Gaussians can be used for generative tasks:

- **Generation:** First sample z, then sample z-th Gaussian to get $\mathbf{x}$
- **Representation Learning:** Learn cluster components from unlabeled data. The posterior $p(z \mid \mathbf{x})$ identifies the latent cluster.

Deep Latent Gaussian Models

We can extend GMMs to mixtures of an infinite # of Gaussians:



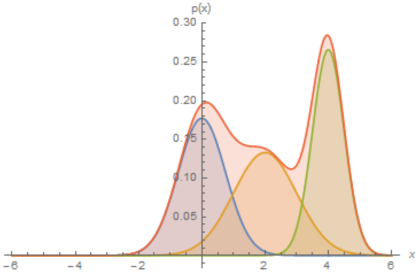
1. The prior is $p(\mathbf{z}) = \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where μ_θ,Σ_θ are neural networks; e.g.:
   - $\mu_\theta(\mathbf{z}) = \sigma(A\mathbf{z} + c) = (\sigma(a_1\mathbf{z} + c_1), \sigma(a_2\mathbf{z} + c_2)) = (\mu_1(\mathbf{z}), \mu_2(\mathbf{z}))$
   - $\Sigma_\theta(\mathbf{z}) = diag(\exp(\sigma(B\mathbf{z} + d))) = \begin{pmatrix} \exp(\sigma(b_1\mathbf{z}+d_1)) & 0 \\ 0 & \exp(\sigma(b_2\mathbf{z}+d_2)) \end{pmatrix}$
   - $\theta = (A, B, c, d)$

Even though $p(\mathbf{x} \mid \mathbf{z})$ is simple, the marginal $p(\mathbf{x})$ is very complex/flexible

Maximum Marginal Likelihood for Latent Variable Models

We may learn latent variable models by choosing p that maximizes the marginal likelihood $p(\mathbf{x})$ averaged over $\mathbf{x}$. For Gaussian mixtures, we have:

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

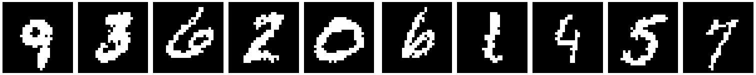


These objectives are non-convex, and in the case of VAEs they're intractable to compute, which motivates approximations we will see next.

Lecture Outline

Running Example: A Generative Model for MNIST

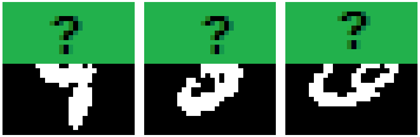
Suppose we are given a dataset $\mathcal{D}$ of handwritten digits (binarized MNIST)



- Each image has $n = 28 \times 28 = 784$ pixels. Each pixel can either be black (0) or white (1).
- **Our Goal:** Learn a probability distribution $p(x) = p(x_1, \cdots, x_{784})$ over $x \in \{0, 1\}^{784}$ such that when $x \sim p(x)$, x looks like a digit.

Running Example: MNIST with Missing Values

One way to introduce latent variables into our running example is to assume that some pixels are missing at train time (e.g., top half):



- Let $\mathbf{x}$ denote the observed random variables, and $\mathbf{z}$ the unobserved ones (also called hidden or latent)
- Suppose we have a model (e.g., an autoregressive model) for the joint

$$p(\mathbf{x}, \mathbf{z}; \theta).$$

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

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

This sums the probabilities of all possible completions of the image (green)

Challenges of Maximum Marginal Likelihood Learning

- Suppose 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.). $\mathcal{D} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}\}$.
- Recall that maximum likelihood learning involves maximizing:

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

- 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.

Naive Approach: Naive Monte Carlo

The marginal likelihood $p_\theta(\mathbf{x})$ for partially observed data is:

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

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, hence we will mis-estimate our objective. We need a clever way to select $\mathbf{z}^{(j)}$.

Second Approach: Importance Sampling

Importance sampling is a clever way of applying Monte Carlo. First, we express $p_\theta(\mathbf{x})$ as a different kind of expectation:

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

where q can be any distribution over $\mathbf{z}$. We then apply Monte Carlo:

1. Sample $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(k)}$ from $q(\mathbf{z})$ (instead of uniformly at random!)
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. When $q(z)$ is "good", this approximation works very well! There are two challenges:

1. How to extend this to approximate the marginal log-likelihood $\log p_\theta(x)$?
2. How to choose a good q?

Approximating the Marginal Log-Likelihood

We've seen a way to approximate marginal probabilities with importance sampling:

$$p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right] \approx \frac{1}{k}\sum_{j=1}^{k}\frac{p_\theta(\mathbf{x},\mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})}$$

Next, we will extend this approach to approximate the marginal log-likelihood:

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

It's clear that

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

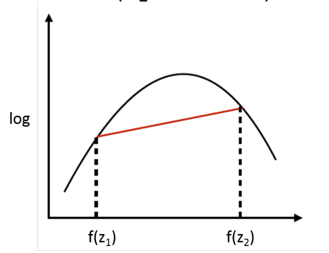
Jensen's Inequality

We want to approximate the marginal log-likelihood:

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

- 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)

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



Evidence Lower Bound via Jensen's Inequality

We want to approximate the marginal log-likelihood:

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

- We will use Jensen's 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 $f(\mathbf{z}) = \frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}$, we obtain:

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

This is called the Evidence Lower Bound (**ELBO**).

The Evidence Lower Bound (ELBO)

The **evidence lower bound** (ELBO) holds for any q:

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

Variational Inference: We will maximize the log-likelihood by maximizing the ELBO while automatically choosing a good q.

How to Choose a Good q?

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

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

Rearranging, we get that

$$\log p(\mathbf{x};\theta) = \mathrm{ELBO} + D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)).$$

- The closer $q(\mathbf{z})$ is to $p(\mathbf{z}|\mathbf{x};\theta)$, the closer the ELBO is to the true log-likelihood.
- Confirms our previous importance sampling intuition: we should choose likely completions.
- In practice, the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute.
- **Strategy**: Find an approximate q close to $p(\mathbf{z}|\mathbf{x};\theta)$ via optimization.

Variational Inference and Learning

We have shown that at any datapoint $\mathbf{x}$, we have

$$\log p(\mathbf{x};\theta) = \mathrm{ELBO}(\theta, q) + D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) \geq \mathrm{ELBO}(\theta, q)$$

Variational inference produces a tight approximation for $\log p(\mathbf{x};\theta)$ by finding a q that makes $\mathrm{ELBO}(q)$ tight:

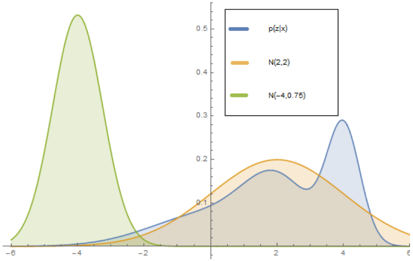
1. We choose $q(\mathbf{z};\phi)$ to be a (tractable) probability distribution over $\mathbf{z}$ parameterized by ϕ (variational parameters).
2. We find a good ϕ^* by maximizing the ELBO and making it tight:

$$\log p(\mathbf{x}^{(i)};\theta) \geq \max_{\phi} \mathrm{ELBO}(\theta, \phi)$$

Once we have a good $\mathrm{ELBO}(\theta, \phi^*)$ that is tight around $\log p(\mathbf{x};\theta)$ we can do interesting things like optimize θ (next lecture!).

Example: Variational Inference Over Gaussians

Variational Inference: Optimize q to approximate the intractable $p(\mathbf{z}|\mathbf{x};\theta)$.

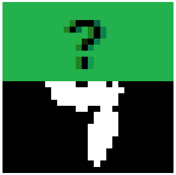


- Example: Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
    - For example, a Gaussian with mean and covariance specified by ϕ

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

- 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)

Example: Optimizing Likelihood with Missing Data



- 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

Summary

- Pros of Latent Variable Models:
  - Easy to build flexible models
  - Suitable for unsupervised learning
- Cons: Latent Variable Models:
  - Not tractable to evaluate likelihoods
  - Not tractable to train via maximum-likelihood
- Computing $p(\mathbf{x}; \theta)$ for arbitrary $\mathbf{x}$ is hard in latent variable models
  - Variational inference produces a tight lower bound on $\log p(\mathbf{x}; \theta)$:

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

    for some good $q(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$ found by optimization
  - Still, this is a complex approximation that only holds at one $\mathbf{x}$.
- Next lecture, we will look at how to scale this procedure to large datasets of $\mathbf{x}$.