Latent Variable Models

Volodymyr Kuleshov

Cornell Tech

Lecture 5

Announcements

- Glitches with Google Hangout link should be resolved. Will be checking email at the beginning of each office hours session to make sure there are no more glitches.
- Homework template is available.
- Extra lecture notes have been posted.
- Good luck with ICML deadline!

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
   - Definition
   - Motivation
2. Warm-up: Shallow mixture models
3. Deep latent-variable models
   - Representation: Variational autoencoder
   - Learning: Variational inference

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: Definition

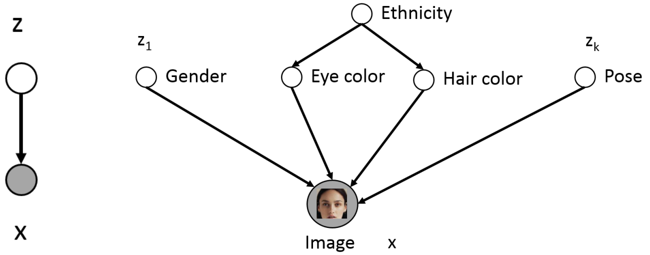
A latent variable model defines a probability distribution

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

containing two sets of variables:

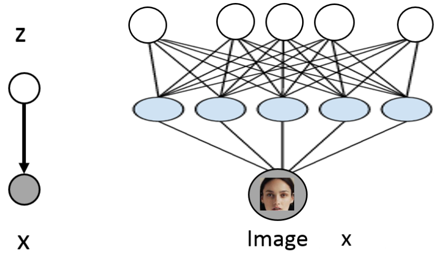
1. Observed variables $\mathbf{x}$ that represent the high-dimensional object we are trying to model.
2. Latent variables $\mathbf{z}$ that are not in the training set, but that are associated with the x via $p(z|x)$ and can encode the structure of the data.

Latent Variable Models: Example



1. Only shaded variables **x** are observed in the data (pixel values)
2. Latent variables **z** correspond to high level features
   - If **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

Deep Latent Variable Models: Example

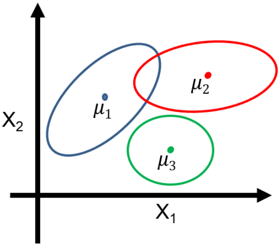


1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. *Hope* that after training, **z** will correspond to meaningful latent factors of variation (*features*). Unsupervised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

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. $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$



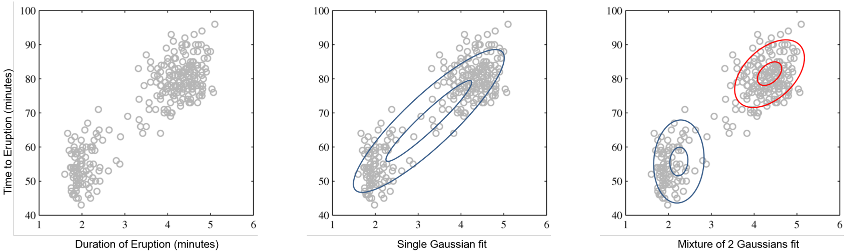
Generative process

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:

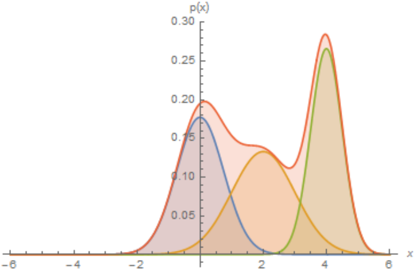
1. $\mathbf{z} \sim \text{Categorical}(1, \cdots, K)$
2. $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$



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)

Representational Power of Mixture models

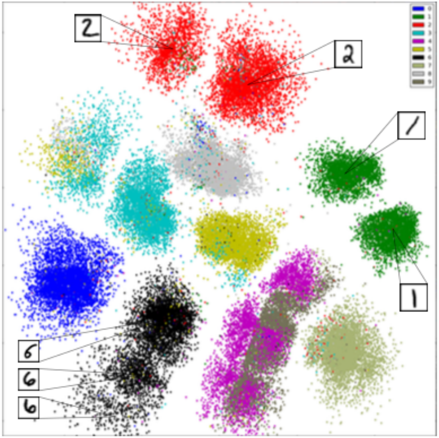
Combine simple models into a more complex and expressive one



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

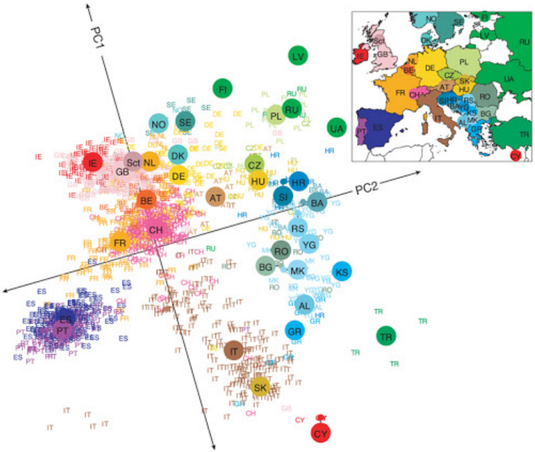
The likelihood is non-convex: this increases representational power, but makes inference more challenging.

Example: Unsupervised learning over hand-written digits

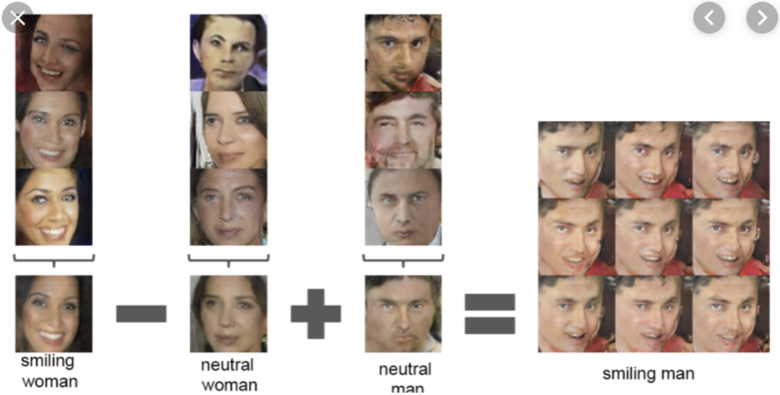


Unsupervised clustering of handwritten digits.

Example: Unsupervised learning over DNA sequence data



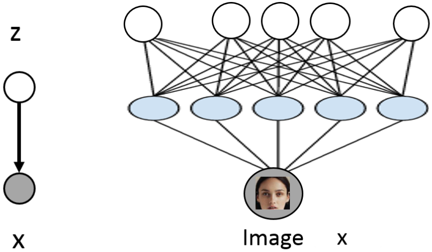
Example: Unsupervised learning over face images



Plan for today

1. Latent variable models
   - Definition
   - Motivation
2. Warm-up: Shallow mixture models
3. **Deep latent-variable models**
   - Representation: Variational autoencoder
   - Learning: Variational inference

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

Benefits of the Latent-Variable Approach

- 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

Partially observed data

- Suppose that our joint distribution is

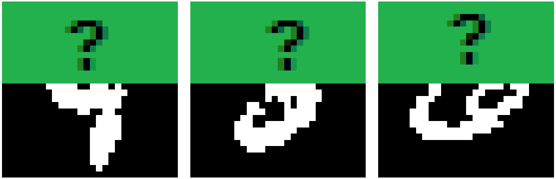
$$p(\mathbf{X}, \mathbf{Z}; \theta)$$

- 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 hard!

Example: Learning with Missing Values



- 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)

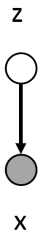
$$p(\mathbf{X}, \mathbf{Z}; \theta)$$

  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)

Example: Variational Autoencoder



A mixture of an infinite number of Gaussians:

- $\mathbf{z} \sim \mathcal{N}(0, I)$. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
- $\mathbf{Z}$ are unobserved at train time (also called hidden or latent)
- 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.
Challenges: deriving algorithms for choosing q and extending this approximation to the marginal log-likelihood.

Approximating the Marginal Log LIkelihood

We can approximate marginal probabilities with importance sampling:

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

However, what we want to approximate is 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

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

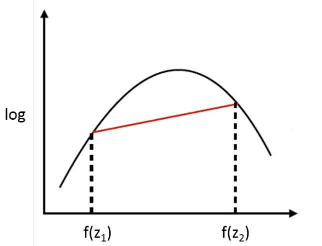
Jensen's Inequality

What we want to approximate is 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

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)

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

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 $q = p(\mathbf{z}|\mathbf{x};\theta)$

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

- **Variational Inference:** Optimize over the possible q's to make bound as tight as possible.

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.
- In practice, 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 $D_{KL}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x};\theta)) = 0$

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

Variational inference continued

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

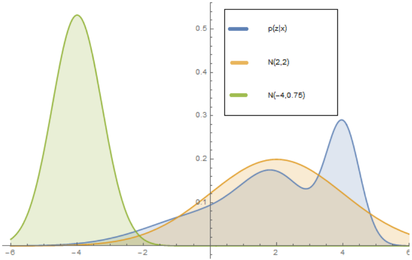
$$D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) = -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q) \geq 0$$

- Rearranging, we 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

Variational Inference Optimizes the Evidence Lower Bound

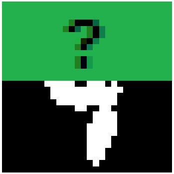


- **Variational inference**: Optimize q to approximate the intractable posterior $p(\mathbf{z}|\mathbf{x};\theta)$.
- Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - For example, a Gaussian with mean and 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

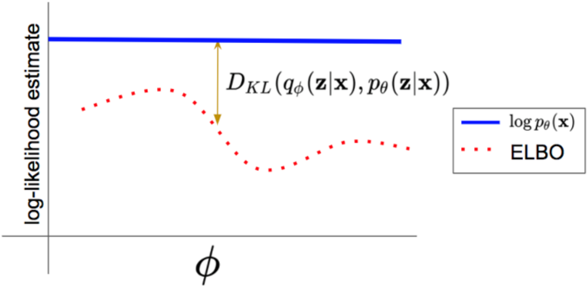


- Assume $p(\mathbf{x}^{top}, \mathbf{x}^{bottom}; \theta)$ assigns high probability to images that look like digits. In this example, we assume $\mathbf{z} = \mathbf{x}^{top}$ are unobserved (latent)
- Suppose $q(\mathbf{x}^{top}; \phi)$ is a (tractable) probability distribution over the hidden variables (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: The Evidence Lower bound



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

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

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
- Next steps: scale-up variational inference to large datasets and neural networks
  - Amortized variational inference
  - Low variance gradient estimators and the reparametrization trick
- Alternative: give up on KL-divergence and likelihood (GANs)