Learning Latent Variable Models

Volodymyr Kuleshov

Cornell Tech

Lecture 6

Announcements

- Thank you for signing up for presentation slots. Papers will be out by end of the week.
- Assignment submission system will be up by the end of the week.
- Good luck with ICML deadline!

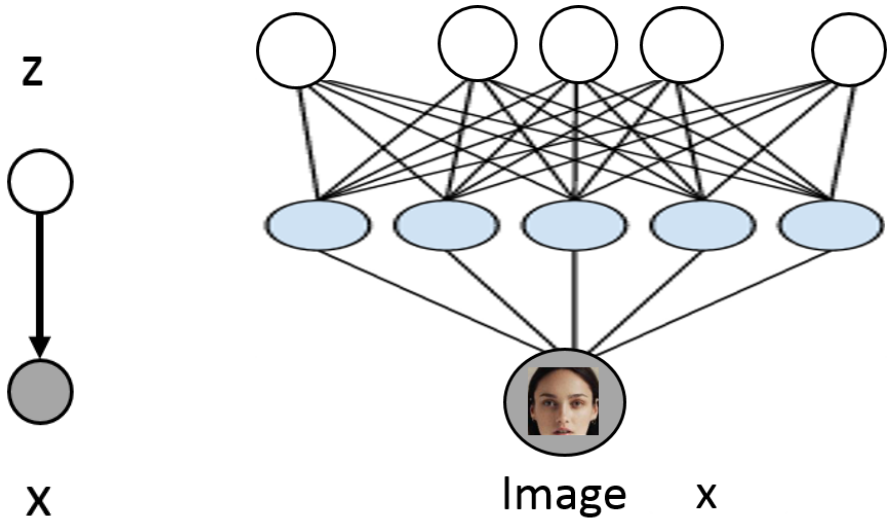
Plan for today

1. Deep latent variable models: a recap
2. Learning deep latent variable generative models
   - Stochastic optimization: gradient estimators
     - REINFORCE estimator
     - Reparameterization trick
   - Inference amortization

Recap: 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**

Recap: 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

- 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

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

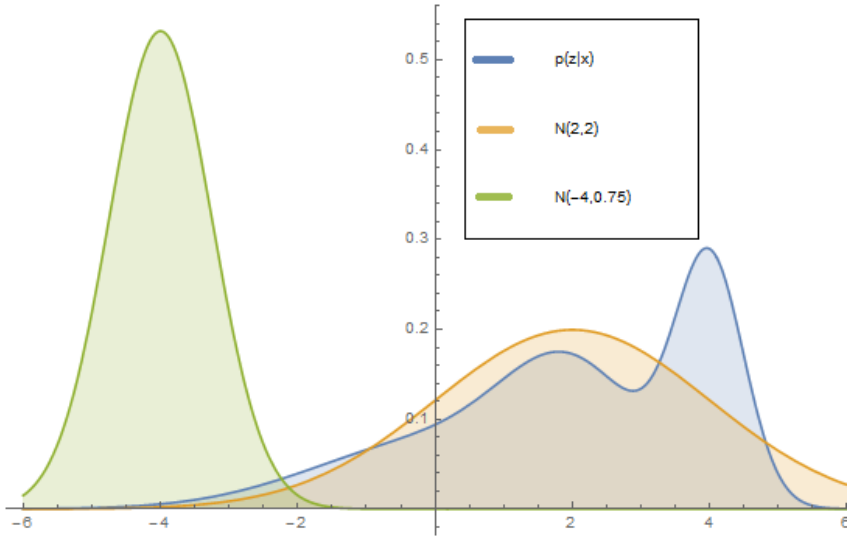
- **Evidence lower bound** (ELBO) holds for any q

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

- 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

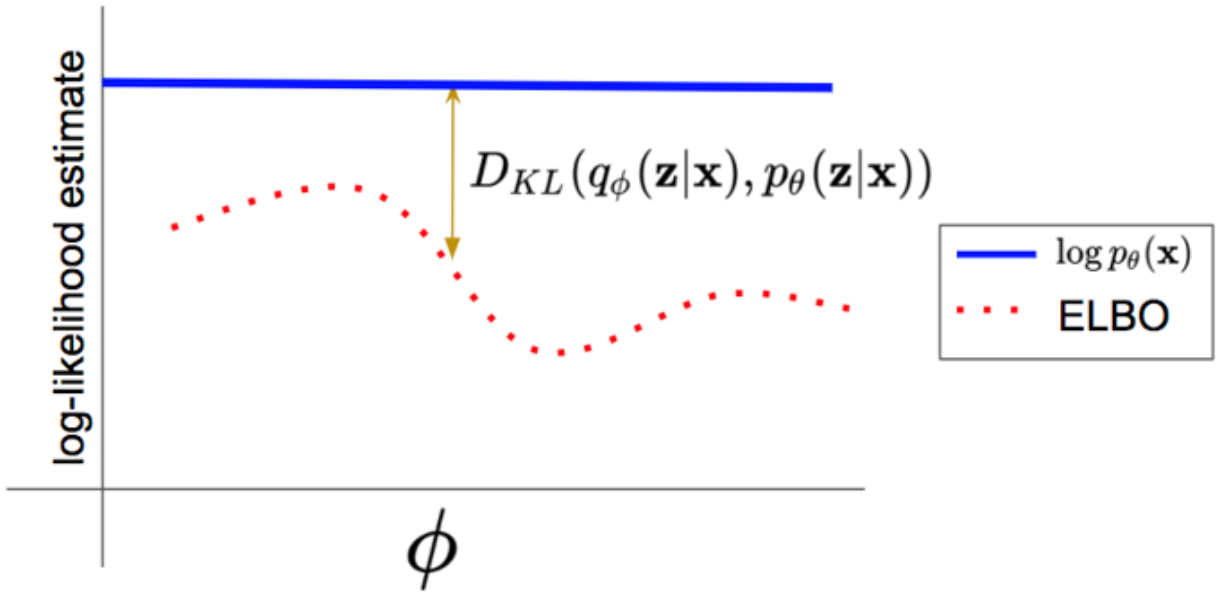


- What if the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute?
- Suppose $q(\mathbf{z};\phi)$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - For example, a Gaussian with mean and covariance specified by ϕ

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

- **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



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

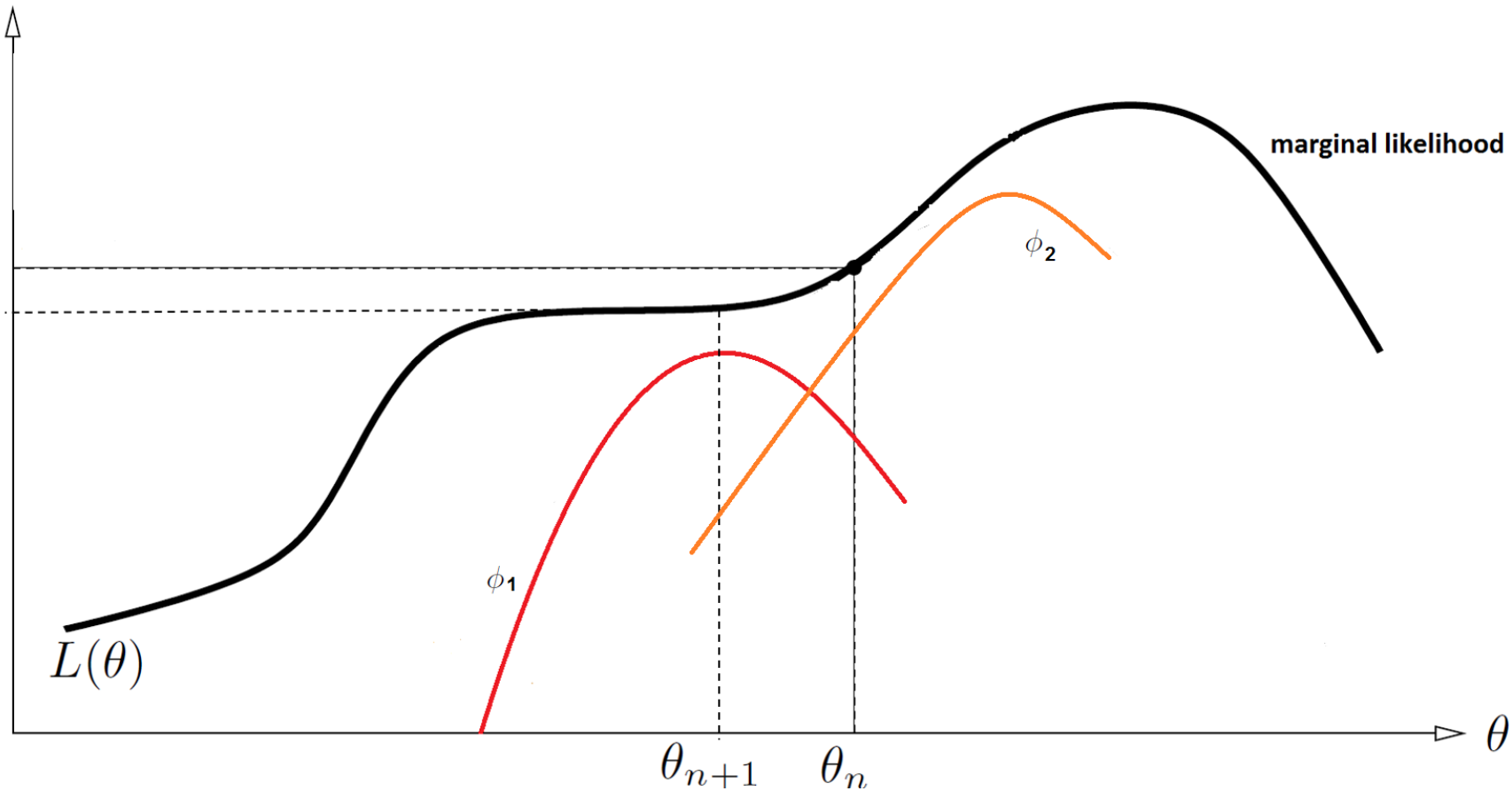
$$= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$$

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

Plan for today

1. Deep latent variable models: a recap
2. **Learning deep latent variable generative models**
   - Stochastic optimization: gradient estimators
     - REINFORCE estimator
     - Reparameterization trick
   - Inference amortization

Variational learning



$\mathcal{L}(\mathbf{x}; \theta, \phi_1)$ 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):

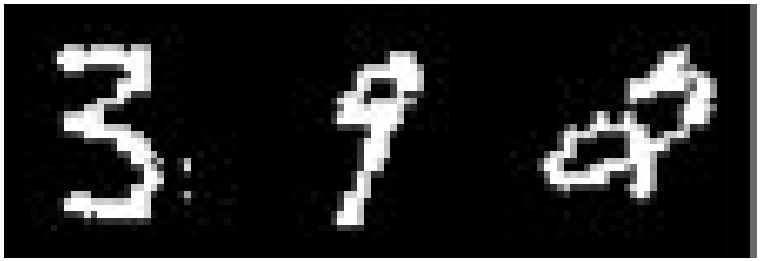
$$\ell(\theta;\mathcal{D}) = \sum_{\mathbf{x}^i\in\mathcal{D}} \log p(\mathbf{x}^i;\theta) \geq \sum_{\mathbf{x}^i\in\mathcal{D}} \mathcal{L}(\mathbf{x}^i;\theta,\phi^i)$$

- 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

Policy Gradients

- Want to compute a gradient with respect to ϕ of the expected reward

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

$$\frac{\partial}{\partial \phi_i} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{1}{q_\phi(\mathbf{z})} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z})$$

$$= \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = E_{q_\phi(\mathbf{z})} \left[\frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) \right]$$

REINFORCE Gradient Estimation

- Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})}\left[f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})\right]$$

- We can now construct a Monte Carlo estimate
- Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$ and estimate

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k)$$

- Assumption: The distribution $q(\cdot)$ is easy to sample from and evaluate probabilities
- Works for both discrete and continuous distributions

Variational Learning of Latent Variable Models

- To learn the variational approximation we need to compute the gradient with respect to ϕ of

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

- The function inside the brackets also depends on ϕ (and $\theta, \mathbf{x}$). Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\phi,\theta,\mathbf{z},\mathbf{x})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) f(\phi,\theta,\mathbf{z},\mathbf{x})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\phi,\theta,\mathbf{z},\mathbf{x})] = E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[f(\phi,\theta,\mathbf{z},\mathbf{x})\nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) + \nabla_\phi f(\phi,\theta,\mathbf{z},\mathbf{x})\right]$$

- We can now construct a Monte Carlo estimate of $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$

REINFORCE Gradient Estimates have High Variance

- Want to compute a gradient with respect to ϕ of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})}\left[f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})\right]$$

- Monte Carlo estimate: Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k) := f_{\mathrm{MC}}(\mathbf{z}^1, \cdots, \mathbf{z}^K)$$

- Monte Carlo estimates of gradients are unbiased

$$E_{\mathbf{z}^1, \cdots, \mathbf{z}^K \sim q_\phi(\mathbf{z})}\left[f_{\mathrm{MC}}(\mathbf{z}^1, \cdots, \mathbf{z}^K)\right] = \nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

Learning Deep Generative models

- 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. Estimate $\nabla_\theta \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ (Monte Carlo)
4. Update θ, ϕ in the gradient direction

- In practice, gradients estimates can be too noisy. Need to use **control variates** (baselines) or **reparameterization trick**

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

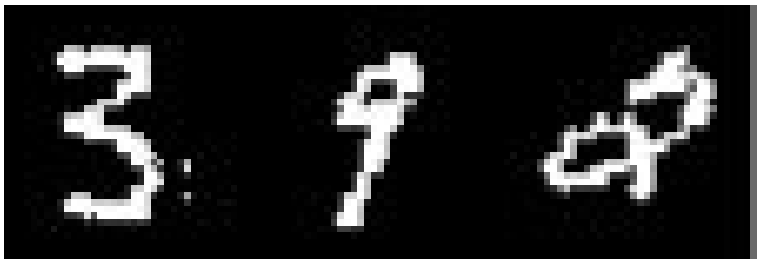
$$\begin{aligned}E_{q(\mathbf{z};\phi)}[r(\mathbf{z},\phi)] &= E_{\epsilon}[r(g(\epsilon;\phi),\phi)] \\ &\approx \frac{1}{k}\sum_{k} r(g(\epsilon^k;\phi),\phi)\end{aligned}$$

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

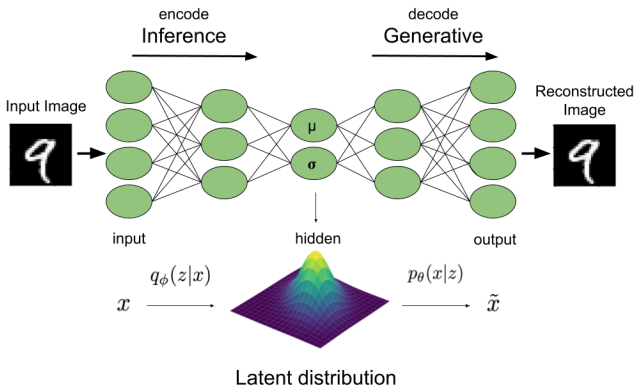
- Optimize $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ as a function of θ, ϕ using (stochastic) gradient descent

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

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



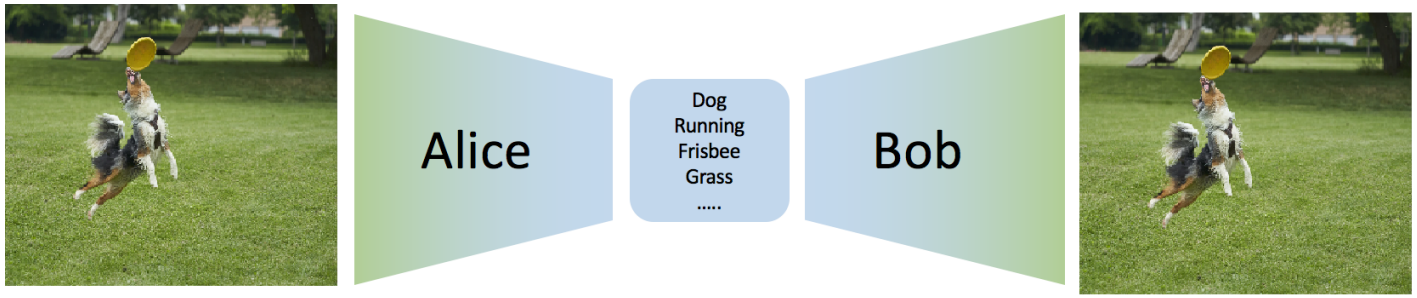
$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))]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

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



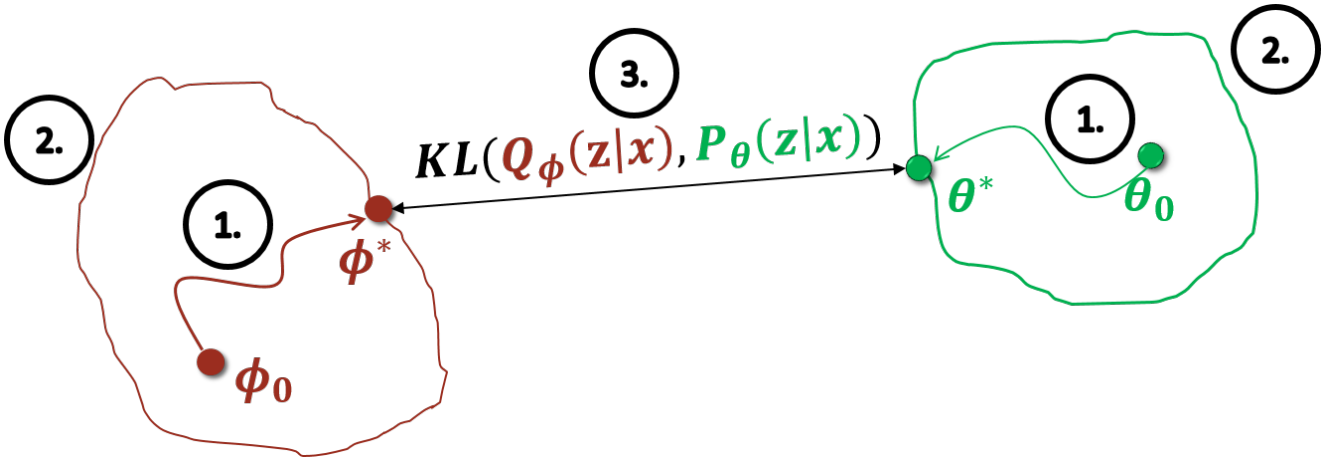
1. Given an image $\mathbf{x}^i$, we (stochastically) compress it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. We send the message $\hat{\mathbf{z}}$ to the decoder
2. Given $\hat{\mathbf{z}}$, the decoder reconstructs the image using $p(\mathbf{x}|\hat{\mathbf{z}}; \theta)$

- 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

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:

- Renyi's 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)