Deep Generative Models

Lecture 5: Variational Autoencoders

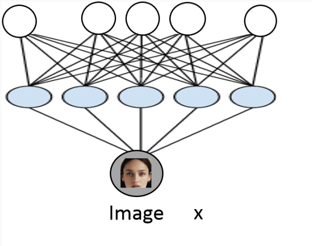
Aditya Grover

UCLA

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

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

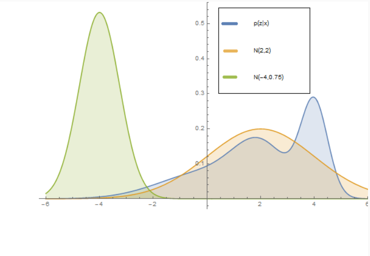
- **Evidence lower bound** (ELBO) holds for any q

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

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

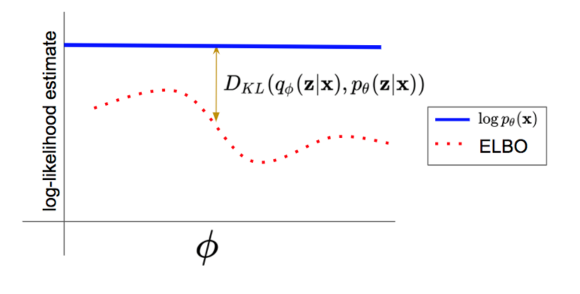
$$\log p_\theta(\mathbf{x}) {\color{red}=} \sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{z}, \mathbf{x}) + H(q)$$

Recap: The Evidence Lower bound



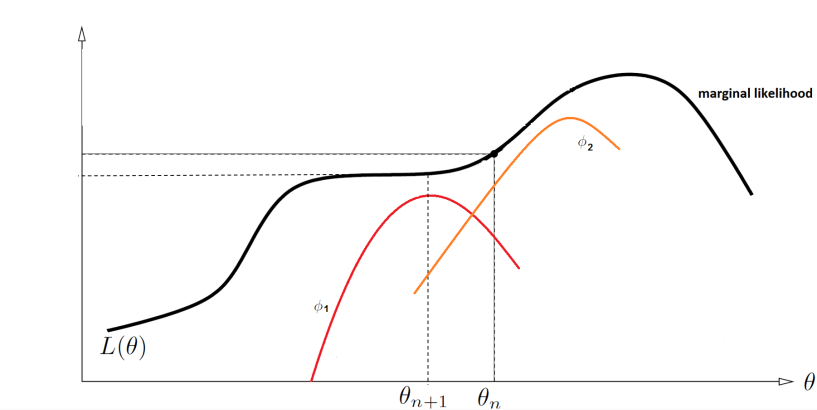
- What if the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable to compute?
- Suppose $q_\phi(\mathbf{z})$ is a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - E.g., a Gaussian with mean and covariance specified by ϕ
$$q_\phi(\mathbf{z}) = \mathcal{N}(\phi_1, \phi_2)$$
- **Variational inference**: pick ϕ so that $q_\phi(\mathbf{z})$ is as close as possible to $p_\theta(\mathbf{z}|\mathbf{x})$. In the figure, the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ (blue) is better approximated by $\mathcal{N}(2, 2)$ (orange) than $\mathcal{N}(-4, 0.75)$ (green)

Recap: The Evidence Lower bound



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

The better $q_\phi(\mathbf{z})$ can approximate the posterior $p_\theta(\mathbf{z}|\mathbf{x})$, the smaller $D_{KL}(q_\phi(\mathbf{z}) \| p_\theta(\mathbf{z}|\mathbf{x}))$ we can achieve, the closer ELBO will be to $\log p_\theta(\mathbf{x})$. Next: jointly optimize over θ and ϕ to maximize the ELBO over a dataset



$\mathcal{L}(\mathbf{x};\theta,\phi_1)$ 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 $q_\phi(\mathbf{z})$

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

- Maximum likelihood learning (over the entire dataset):

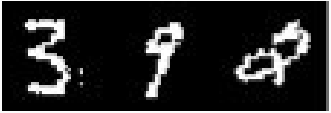
$$\ell(\theta; \mathcal{D}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log p(\mathbf{x}^i; \theta) \geq \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

- 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 $p_\theta(\mathbf{z}|\mathbf{x}^i)$ is different across datapoints $\mathbf{x}^i$

A variational approximation to the posterior



- Assume $p_\theta(\mathbf{z}, \mathbf{x}^i)$ 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 $q_{\phi^i}(\mathbf{z})$ is a (categorical) probability distribution over the hidden variable $\mathbf{z}$ parameterized by $\phi^i = [p_0, p_1, \cdots, p_9]$

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

- If $\phi^i = [0, 0, 0, 1, 0, \cdots, 0]$, is $q_{\phi^i}(\mathbf{z})$ a good approximation of $p_\theta(\mathbf{z}|\mathbf{x}^1)$ ($\mathbf{x}^1$ is the leftmost datapoint)? Yes
- If $\phi^i = [0, 0, 0, 1, 0, \cdots, 0]$, is $q_{\phi^i}(\mathbf{z})$ a good approximation of $p_\theta(\mathbf{z}|\mathbf{x}^3)$ ($\mathbf{x}^3$ is the rightmost datapoint)? No

Learning latent variable models

Optimize $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$ w.r.t. $\theta, \phi^1, \cdots, \phi^M$ using SGD

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

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:
   3.1 Repeat $\phi^i = \phi^i + \eta \nabla_{\phi^i} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$
   3.2 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 gradients in Steps 4,5? Potentially no closed form solution for the expectations. Use Monte Carlo sampling.

Learning latent variable models

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

- Note: dropped i superscript from $\mathbf{x}^i, \phi^i$ for compactness
- To *evaluate* the bound, use Monte Carlo. Sample $\mathbf{z}^k \sim q_\phi(\mathbf{z})$ and estimate:

$$E_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z})] \approx \frac{1}{k}\sum_k \log p_\theta(\mathbf{z}^k,\mathbf{x}) - \log q_\phi(\mathbf{z}^k))$$

- Key assumption: $q_\phi(\mathbf{z})$ is easy to sample from and evaluate
- How to compute gradients $\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$?

Learning latent variable models

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

- 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 θ is easy with Monte Carlo:

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

  where $\mathbf{z}^k \sim q_\phi(\mathbf{z})$.
- 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

Reparameterization

- Assume $\mathbf{z}$ is now **continuous** [more general solution called REINFORCE later in course]
- Goal compute gradient w.r.t. ϕ of $E_{q_\phi(\mathbf{z})}[r(\mathbf{z})] = \int q_\phi(\mathbf{z}) r(\mathbf{z}) d\mathbf{z}$
- Suppose $q_\phi(\mathbf{z}) = \mathcal{N}(\mu, \sigma^2 I)$ is Gaussian with parameters $\phi = (\mu, \sigma)$. These are equivalent ways of sampling:
  - Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
  - Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon = g_\phi(\epsilon)$
- Using this equivalence we compute the expectation in two ways:

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

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[r(\mathbf{z})] = \nabla_\phi E_\epsilon[r(g_\phi(\epsilon))] = E_\epsilon[\nabla_\phi r(g_\phi(\epsilon))]$$

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

Learning Deep Generative models

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

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

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

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 that maps each $\mathbf{x}$ to a set of (good) variational parameters. Like doing regression on $\mathbf{x}^i \mapsto \phi^{i*}$
  - For example, instead of learning $q_{\phi^i}(\mathbf{z})$ as Gaussians with different means $\mu^1, \cdots, \mu^m$ and covariances $\Sigma^1, \cdots, \Sigma^m$, we can learn a **single** shared neural network $q_\phi(\mathbf{z}|\mathbf{x})$ that maps any $\mathbf{x}^i$ to corresponding mean μ^i and covariance Σ^i

Learning with amortized inference

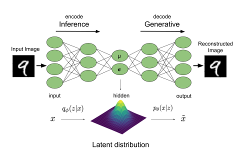
- Optimize $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ w.r.t. θ, ϕ using (stochastic) gradient descent

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

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



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

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

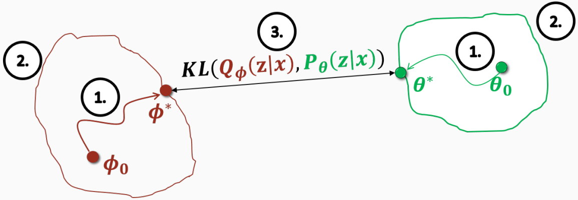
1. Alice goes on a space mission and needs to send images to Bob. Given an image $\mathbf{x}^i$, she (stochastically) compresses it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. Alice sends the message $\hat{\mathbf{z}}$ to Bob
2. Given $\hat{\mathbf{z}}$,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