Deep Generative Models

6. Latent variable models

• 국가수리과학연구소 산업수학혁신센터 김민중

Plan for today

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

Variational inference

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

$$\log p_\theta(\boldsymbol{x}) \geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})}$$

$$= \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log q(\boldsymbol{z})$$

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

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

Variational inference(continued)

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

$$D\big(q(\boldsymbol{z}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\big) = -\sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log p_\theta(\boldsymbol{x}, \boldsymbol{z}) + \log p_\theta(\boldsymbol{x}) - H(q) \geq 0$$

- Evidence lower bound (ELBO) holds for any q

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

- Equality holds if $q(\boldsymbol{z}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})$ because $D\big(q(\boldsymbol{z}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\big) = 0$
- Confirms our intuition that we seek likely completions $\boldsymbol{z}$ given the observed values (evidence) $\boldsymbol{x}$

The Evidence Lower bound

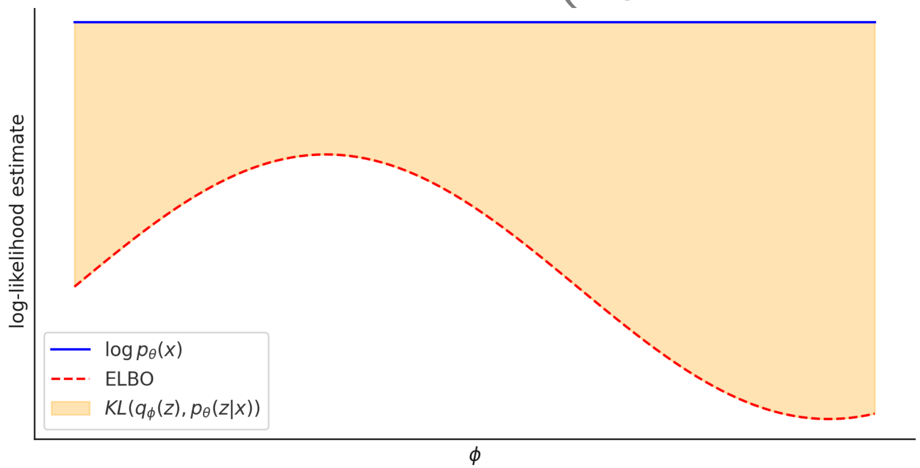
- What if the posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is intractable to compute?
- In a VAE, this corresponds to "inverting " the neural networks $\mu_\theta, \Sigma_\theta$ defining $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}|\mu_\theta(\boldsymbol{z}), \Sigma_\theta(\boldsymbol{z}))$
- Suppose is $q_\phi(\boldsymbol{z})$ a (tractable) probability distribution over the hidden variables parameterized by ϕ (variational parameters)
  - For example, a Gaussian with mean and covariance specified by ϕ

$$q_\phi(\boldsymbol{z}) = N\left(\boldsymbol{z}\middle|\boldsymbol{\mu}_\phi, \Sigma_\phi\right)$$

- Variational inference: pick ϕ so that $q_\phi(\boldsymbol{z})$ is as close as possible to $p_\theta(\boldsymbol{z}|\boldsymbol{x})$

The Evidence Lower bound

$$\log p_\theta(\boldsymbol{x}) = \text{ELBO} + D\left(q_\phi(\boldsymbol{z}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)$$



- The better $q_\phi(\boldsymbol{z})$ can approximate the posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, the smaller $D\left(q_\phi(\boldsymbol{z}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)$ we can achieve, the closer ELBO will be to $\log p_\theta(\boldsymbol{x})$
- We want to jointly optimize over θ and ϕ to maximize the ELBO over a dataset D

The Evidence Lower bound applied to the dataset

- Evidence lower bound (ELBO) holds for any $q_\phi(\boldsymbol{z})$

$$\log p_\theta(\boldsymbol{x}) \geq \sum_{\boldsymbol{z}} q_\phi(\boldsymbol{z}) \log p_\theta(\boldsymbol{x}, \boldsymbol{z}) + H\left(q_\phi(\boldsymbol{z})\right) =: \mathcal{L}(\boldsymbol{x}; \theta, \phi)$$

- Maximum likelihood learning (over the entire dataset)

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

- Therefore,

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

- Note that we use different variational parameters ϕ^i for every data point $\boldsymbol{x}^{(i)}$

Learning via stochastic variational inference(SVI)

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

$$\mathcal{L}\left(\boldsymbol{x}^{(i)}; \theta, \phi^i\right) = \sum_{\boldsymbol{z}} q_{\phi^i}(\boldsymbol{z}) \log p_\theta\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}\right) + H\left(q_{\phi^i}(\boldsymbol{z})\right)$$

$$= E_{q_{\phi^i}(\boldsymbol{z})}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}\right) - \log q_{\phi^i}(\boldsymbol{z})\right]$$

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

Learning Deep Generative models

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = \sum_{\boldsymbol{z}} q_\phi(\boldsymbol{z}) \log p_\theta(\boldsymbol{x},\boldsymbol{z}) + H\left(q_\phi(\boldsymbol{z})\right)$$
$$= E_{q_\phi(\mathbf{z})}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z})\right]$$

- Note: dropped i superscript from ϕ^i for compactness
- To evaluate the bound, sample $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(K)}$ from $q_\phi(\boldsymbol{z})$ and estimate

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

- Key assumption: $q_\phi(\boldsymbol{z})$ is tractable, i.e., easy to sample and evaluate
- Want to compute $\nabla_\phi \mathcal{L}(\boldsymbol{x};\theta,\phi)$ and $\nabla_\theta \mathcal{L}(\boldsymbol{x};\theta,\phi)$

Learning Deep Generative models

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = \sum_{\boldsymbol{z}} q_\phi(\boldsymbol{z}) \log p_\theta(\boldsymbol{x},\boldsymbol{z}) + H\left(q_\phi(\boldsymbol{z})\right)$$
$$= E_{q_\phi(\mathbf{z})}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z})\right]$$

- Want to compute $\nabla_\phi \mathcal{L}(\boldsymbol{x};\theta,\phi)$ and $\nabla_\theta \mathcal{L}(\boldsymbol{x};\theta,\phi)$
- 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
- For now, a better but less general alternative that only works for continuous $\boldsymbol{z}$ (and only some distributions)

Reparameterization

- Want to compute a gradient with respect to ϕ of

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

- where $\mathbf{z}$ is continuous
- Suppose $q_\phi(\mathbf{z}) = N(\mathbf{z}|\boldsymbol{\mu}, \sigma \boldsymbol{I})$ is a Gaussian with parameters $\phi = (\boldsymbol{\mu}, \sigma)$
- These are equivalent ways of sampling
  - Sample $\mathbf{z} \sim N(\boldsymbol{\mu}, \sigma \boldsymbol{I})$
  - Sample $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{I})$, $\mathbf{z} = \boldsymbol{\mu} + \sigma \boldsymbol{\epsilon} = g_\phi(\boldsymbol{\epsilon})$. Here g_ϕ is deterministic

Reparameterization

- Using this equivalence, we compute the expectation in two ways

$$E_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z})}[r(\boldsymbol{z})] = \int q_\phi(\boldsymbol{z})r(\boldsymbol{z})d\boldsymbol{z} = E_{\boldsymbol{\epsilon}\sim N(\boldsymbol{0},\boldsymbol{I})}\left[r\left(g_\phi(\boldsymbol{\epsilon})\right)\right]$$

$$= \int N(\boldsymbol{\epsilon})r(\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$$

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

- Easy to estimate via Monte Carlo if r and g_ϕ are differentiable w.r.t. ϕ and $\boldsymbol{\epsilon}$ is easy to sample from (backpropagation)
- $E_\epsilon\left[\nabla_\phi r\left(g_\phi(\boldsymbol{\epsilon})\right)\right] \approx \frac{1}{K}\sum_k \nabla_\phi r\left(g_\phi(\boldsymbol{\epsilon}^{(k)})\right)$ where $\boldsymbol{\epsilon}^{(1)}, \cdots, \boldsymbol{\epsilon}^{(K)} \sim N(\boldsymbol{0}, \boldsymbol{I})$

Learning Deep Generative models

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = \sum_{\boldsymbol{z}} q_\phi(\boldsymbol{z}) \log p_\theta(\boldsymbol{x},\boldsymbol{z}) + H\left(q_\phi(\boldsymbol{z})\right)$$
$$= E_{q_\phi(\mathbf{z})}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z})\right]$$

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

$$E_{q_\phi(\mathbf{z})}[r(\boldsymbol{z},\phi)] = E_\epsilon\left[r_\phi\left(g_\phi(\boldsymbol{\epsilon})\right)\right] \approx \frac{1}{K}\sum_k r_\phi\left(g_\phi(\boldsymbol{\epsilon}^{(k)})\right)$$

- and use chain rule for the gradient

Learning via stochastic variational inference(SVI)

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

$$\mathcal{L}(\boldsymbol{x}^{(i)}; \theta, \phi^i) = \sum_{\boldsymbol{z}} q_{\phi^i}(\boldsymbol{z}) \log p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{z}) + H\left(q_{\phi^i}(\boldsymbol{z})\right)$$

$$= E_{q_{\phi^i}(\boldsymbol{z})}\left[\log p_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{z}) - \log q_{\phi^i}(\boldsymbol{z})\right]$$

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

Amortized Inference

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

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

A variational approximation to the posterior

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

Learning with amortized inference

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

$$\mathcal{L}(\boldsymbol{x}; \theta, \phi) = \sum_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log p_\theta(\boldsymbol{x}, \boldsymbol{z}) + H\left(q_\phi(\boldsymbol{z}|\boldsymbol{x})\right)$$

$$= E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x})\right]$$

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

Amortized Variational Inference

- **Inference network**: a model that learns an inverse map from observations to latent variables
- Using this, we can compute a set of global variational parameters ϕ valid for infrence at both training and test time
- The simplest inference models: diagonal Gaussian densities

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = N\left(\boldsymbol{z}\middle|\boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x})\right)\right)$$

VAE: Autoencoder perspective

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = E_{q_\phi(\mathbf{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\boldsymbol{x})\right]$$
$$= E_{q_\phi(\mathbf{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x},\mathbf{z}) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\boldsymbol{x})\right]$$
$$= E_{q_\phi(\mathbf{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\mathbf{z})] - D\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p(\mathbf{z})\right)$$

1. Take a data point $\boldsymbol{x}'$, map it to sample $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\boldsymbol{x}')$ (encoder)

- Sample from a Gaussian $q_\phi(\mathbf{z}|\boldsymbol{x}') = N\left(\mathbf{z}\middle|\boldsymbol{\mu}_\phi(\boldsymbol{x}'), \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}')\right)\right)$, $\text{encoder}_\phi(\boldsymbol{x}')$

2. Reconstruct $\hat{\boldsymbol{x}}$ by sampling from $p_\theta(\boldsymbol{x}|\hat{\mathbf{z}})$(decoder)

- Sample from a Gaussian with parameters $\text{decoder}_\theta(\hat{\mathbf{z}})$
- What does the training objective $\mathcal{L}(\boldsymbol{x};\theta,\phi)$ do?
  - First term encourages $\hat{\boldsymbol{x}} \approx \boldsymbol{x}'$ ($\boldsymbol{x}'$ likely under $p_\theta(\boldsymbol{x}|\hat{\mathbf{z}})$)
  - Second term encourages $\hat{\mathbf{z}}$ to have a distribution like the prior $p(\mathbf{z})$

Summary of Latent Variable Models

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

Thanks