Learning Latent Variable Models

Volodymyr Kuleshov

Cornell Tech

Lecture 6

Announcements

- Project proposal tentatively due on Wednesday Feb 22rd (in two weeks).
- I will be traveling on two dates in the next months. Lectures will be over Zoom.
    - Monday, Feb. 13
    - Monday, Mar. 13

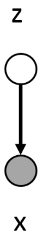
Latent Variable Models: Motivation

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

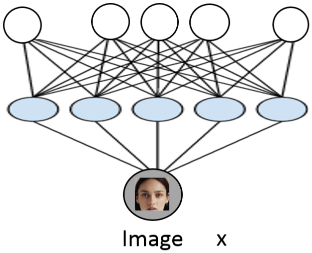
$$p(x, z) = p(x|z)p(z)$$

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.

Deep Gaussian Latent Variable Models

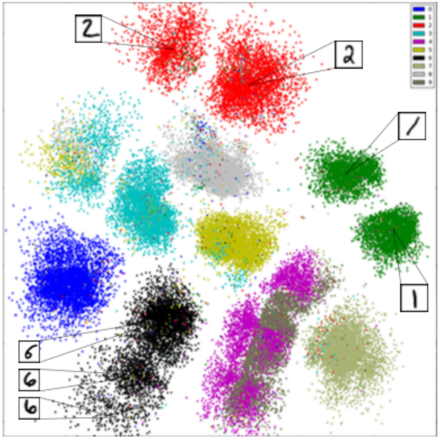
We may form deep latent variable models by parameterizing the $\mathbf{z} \rightarrow \mathbf{x}$ mapping using neural nets. Deep Gaussian models are an example:



1. The prior $p(\mathbf{z})$ is a p-dimensional Gaussian $\mathcal{N}(0, I_p)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where μ_θ,Σ_θ are neural networks

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

Example: Unsupervised Learning Over Handwritten Digits



Unsupervised clustering of MNIST digits using deep latent variable models.

Challenges of Maximum Marginal Likelihood Learning

Maximum likelihood learning of latent variable models is typically intractable.

- 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.

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:

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

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 (this lecture!).

Plan for today

1. **Learning Deep Latent Variable Models**
2. Gradient-Based Optimization of the ELBO
    - The REINFORCE Gradient Estimator
    - The Reparameterization Trick
3. Inference Amortization
4. The Variational Autoencoder

Approximating Marginal Likelihood Over Full Training Set

The evidence lower bound (ELBO) holds for any $\mathbf{x}$ and any $q(\mathbf{z};\phi)$:

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

Suppose we now have a training set $\mathcal{D} = \{\mathbf{x}^1, \cdots, \mathbf{x}^M\}$. We can form the ELBO at each x^i to obtain a lower bound for the marginal likelihood over the entire $\mathcal{D}$:

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

- 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 $\mathbf{x}^i$
- This bound is tightest when each ϕ_i maximizes the ELBO for its $\mathbf{x}^i$:

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

Learning via Stochastic Variational Inference (SVI)

We can now define a training procedure for latent variable models:

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

where $\mathcal{L}(\mathbf{x}^i; \theta, \phi^i) = E_{q(\mathbf{z}; \phi^i)}[\log p(\mathbf{z}, \mathbf{x}^i; \theta) - \log q(\mathbf{z}; \phi^i)]$. We will optimize the above objective using stochastic gradient descent:

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

Estimating Gradients for the ELBO

The above procedures requires us to estimate the gradients $\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$ of the ELBO:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = E_{q(\mathbf{z};\phi)}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q(\mathbf{z};\phi)]$$

- Key assumption: $q(\mathbf{z};\phi)$ is tractable, i.e., easy to sample from and evaluate
- The gradient with respect to θ is easy

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

- The gradient with respect to ϕ is more complicated because the expectation depends on ϕ. We will derive specialized gradient estimators for it next.

Plan for today

1. Learning Deep Latent Variable Models
2. **Gradient-Based Optimization of the ELBO**
    - The REINFORCE Gradient Estimator
    - The Reparameterization Trick
3. Inference Amortization
4. The Variational Autoencoder

The REINFORCE Gradient Trick

We want the gradient with respect to ϕ of the expectation of f:

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

We can use the following trick to obtain an estimator:

$$\begin{aligned}\frac{\partial}{\partial \phi_i} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] &= \textstyle\sum_{\mathbf{z}} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{1}{q_\phi(\mathbf{z})} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) \\ &= \textstyle\sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = E_{q_\phi(\mathbf{z})}\left[\frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z})\right]\end{aligned}$$

REINFORCE Gradient Estimation

We 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 formula for the entire gradient 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 sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$ and estimate using Monte Carlo:

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

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

REINFORCE Gradient Estimation for the ELBO

Recall that we seek the gradient $\nabla_\phi \mathcal{L}(\mathbf{x}; \theta, \phi)$ of the ELBO:

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q(\mathbf{z}; \phi)}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)]$$

- 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

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

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

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

- Monte Carlo estimates of gradients are unbiased

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

- However, its variance is high! The value $\log q(z)$ can greatly vary in magnitude for small z. In practice, there are better estimates that one can use.

The Reparameterization Trick

Suppose we 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**.

- Example: $q(\mathbf{z};\phi) = \mathcal{N}(\mu, \sigma^2 I)$ 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

The Reparameterization Trick for Optimizing the ELBO

Recall that we seek the gradient $\nabla_\phi \mathcal{L}(\mathbf{x}; \theta, \phi)$ of the ELBO:

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q(\mathbf{z};\phi)}[\underbrace{\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)}_{r(\mathbf{z},\phi)}]$$

- 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.

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

and we can take gradients of this Monte Carlo approximation with respect to ϕ using automatic differentiation.

Plan for today

1. Learning Deep Latent Variable Models
2. Gradient-Based Optimization of the ELBO
    - The REINFORCE Gradient Estimator
    - The Reparameterization Trick
3. **Inference Amortization**
4. The Variational Autoencoder

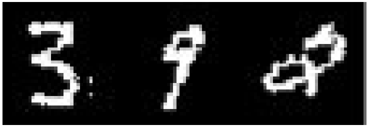
Amortized Inference

Recall that our learning objective can be written as:

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

- 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 $q_\lambda(\mathbf{z}|\mathbf{x})$

Amortized Inference: An Example



- 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 $q_\phi(\mathbf{z}|\mathbf{x})$ and we use ϕ to denote the parameters of the neural network for q.

Learning with Amortized Inference

We now define our final training procedure for latent variable models:

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

where $\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x})]$.

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

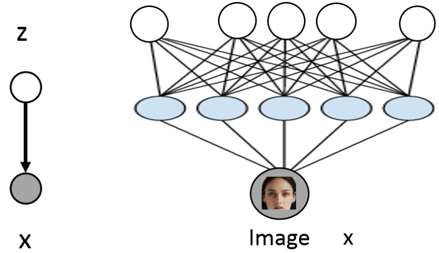
- How to compute the gradients? Use reparameterization like before

Plan for today

1. Learning Deep Latent Variable Models
2. Gradient-Based Optimization of the ELBO
    - The REINFORCE Gradient Estimator
    - The Reparameterization Trick
3. Inference Amortization
4. **The Variational Autoencoder**

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

The Variational Autoencoder

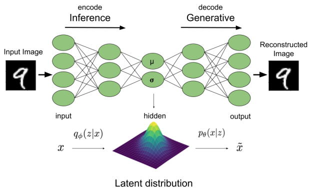
The variational autoencoder is a deep generative modeling algorithm defined by the following characteristics:

- The model $p(\mathbf{x}, \mathbf{z}, \theta)$ is a Deep Latent Gaussian Model
  - The conditional $p(\mathbf{x}|\mathbf{z})$ can also take a non-Gaussian form
  - It is crucial that the latent $\mathbf{z}$ are Gaussian
- The approximate inference network $q(\mathbf{z}|\mathbf{x})$ is a conditional Gaussian
- Both $p(\mathbf{x}|\mathbf{z}), q(\mathbf{z}|\mathbf{x})$ are distributions whose parameters (e.g., $\mu(\mathbf{x}), \Sigma(\mathbf{x})$) are defined by neural networks
- The model is fit using variational inference to maximize the ELBO

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

  where $\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))]$.
- The gradients $\nabla_\theta \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ and $\nabla_\phi \mathcal{L}(\mathbf{x}^i; \theta, \phi)$ are estimated using the reparameterization trick.

Autoencoder Perspective



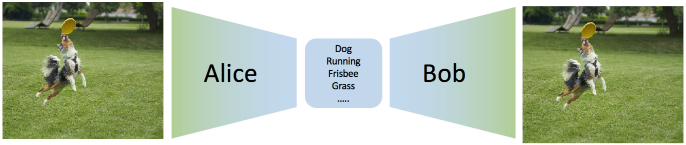
$$\begin{aligned}
\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]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

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



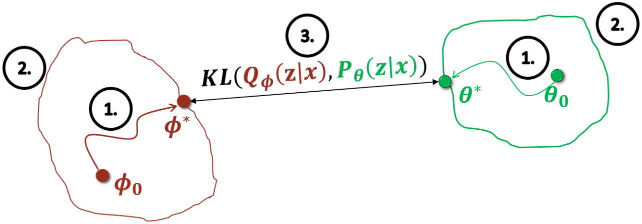
1. Given an image $\mathbf{x}^i$, we (stochastically) compress it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. We send the message $\hat{\mathbf{z}}$ to the decoder
2. Given $\hat{\mathbf{z}}$, the decoder reconstructs the image using $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$

- 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

- Pros of Latent Variable Models
  - Can be used to naturally combine simple models to get a more flexible one (e.g., mixture of Gaussians)
  - Directed model permits ancestral sampling (efficient generation): $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z};\theta)$
  - Latent representations for any $\mathbf{x}$ can be inferred via $q_\phi(\mathbf{z}|\mathbf{x})$
- Cons of Latent Variable Models
  - However, log-likelihood is generally intractable, hence learning is difficult
  - We rely on joint learning of a model (θ) and an amortized inference component (ϕ) to achieve tractability via ELBO optimization

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)