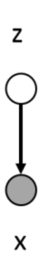
Deep Generative Models (Fall 2024)

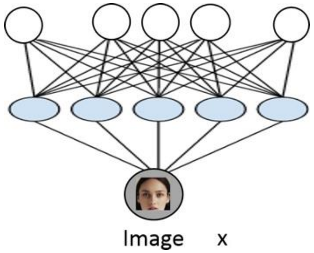
CS HUFS

Variational Autoencoder II

- Latent Variable Models
  - Learning deep generative models
  - Stochastic optimization (Reparameterization trick)
  - Inference amortization

Variational Autoencoder





A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim N(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = N(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where $\mu_\theta, \Sigma_\theta$ 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 because $p(\mathbf{x})$ is hard to evaluate (and optimize)

Variational inference

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

$$
\begin{aligned}
\log p(\mathbf{x};\theta) &\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right) \\
&= \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z})}_{\text{Loglikelihood as if fully observed}} \underbrace{- \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})}_{\text{Entropy } H(q) \text{ of } q} \\
&= \sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z}) + H(q)
\end{aligned}
$$

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

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

- (Aside: This is what we compute in the E-step of the EM algorithm)

Variational Inference

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

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

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

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

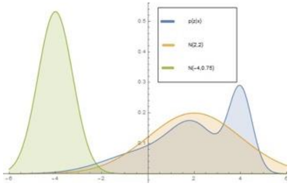
- Equality holds if $q = p(\mathbf{z}|\mathbf{x};\theta)$ because $D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) = 0$

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

- Confirms our intuition that we seek likely completions $\mathbf{z}$ given the observed values (evidence) $\mathbf{x}$.

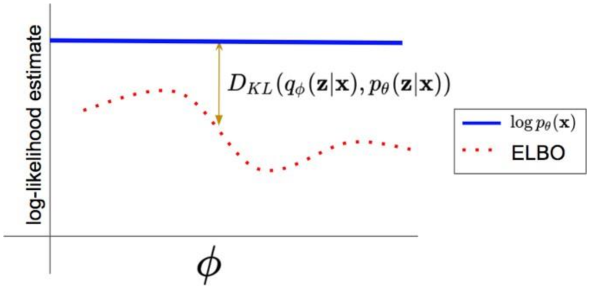
Intractable Posteriors

- What if the posterior $p(\mathbf{z}|\mathbf{x};\theta)$ is intractable to compute? In a VAE this corresponds to "inverting" the neural networks $\mu_\theta, \Sigma_\theta$ defining $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$
- 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)

The Evidence Lower bound

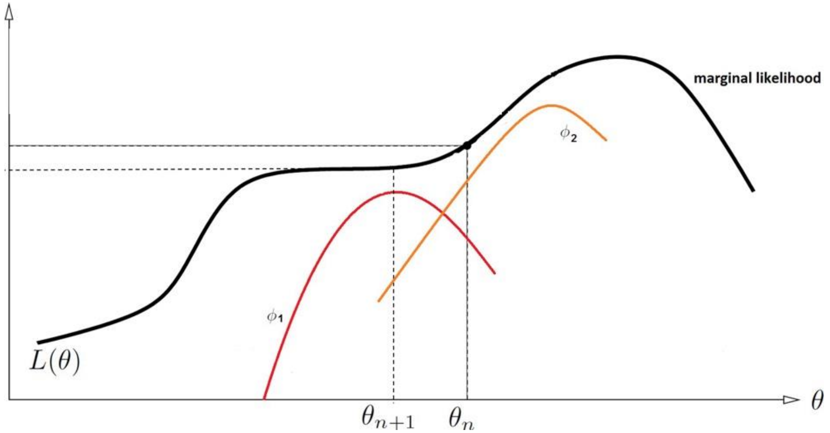


$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

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

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

Variational learning



$L(\mathbf{x}; \theta, \phi_1)$ and $L(\mathbf{x}; \theta, \phi_2)$ 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):

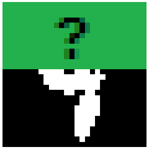
$$\ell(\theta;\mathcal{D}) = \sum_{\mathbf{x}^i\in\mathcal{D}} \log p(\mathbf{x}^i;\theta) \geq \sum_{\mathbf{x}^i\in\mathcal{D}} \mathcal{L}(\mathbf{x}^i;\theta,\phi^i)$$

- 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 $p(\mathbf{z}, \mathbf{x}; \theta)$ is close to $p_{\mathrm{data}}(\mathbf{z}, \mathbf{x})$. $\mathbf{z}$ denotes the top half of the image (assumed to be latent)
- Suppose $q(\mathbf{z}; \phi)$ is a (tractable) probability distribution over the hidden variables $\mathbf{z}$ parameterized by ϕ (variational parameters)

$$q(\mathbf{z}; \phi) = \prod_{\text{unobserved variables } z_i} (\phi_i)^{z_i} (1 - \phi_i)^{(1-z_i)}$$

- Is $\phi_i = 0.5 \ \forall i$ a good approximation to the posterior $p(\mathbf{z}|\mathbf{x}; \theta)$? No
- Is $\phi_i = 1 \ \forall i$ a good approximation to the posterior $p(\mathbf{z}|\mathbf{x}; \theta)$? No
- Is $\phi_i \approx 1$ for pixels i corresponding to the top part of digit **9** a good approximation? Yes
- Note: not true if $p(\mathbf{z}, \mathbf{x}; \theta)$ is far from $p_{\mathrm{data}}(\mathbf{z}, \mathbf{x})$, i.e., at the beginning of learning

Learning via stochastic variational inference (SVI)

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

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

1. Initialize $\theta, \phi^1, \cdots, \phi^M$
2. Randomly sample a data point $\mathbf{x}^i$ from D
3. Optimize $L(\mathbf{x}^i; \theta, \phi^i)$ as a function of ϕ^i:
    1. Repeat $\phi^i = \phi^i + \eta \nabla_{\phi^i} L(\mathbf{x}^i; \theta, \phi^i)$
    2. until convergence to $\phi^{i,*} \approx \arg\max_\phi L(\mathbf{x}^i; \theta, \phi)$
4. Compute $\nabla_\theta L(\mathbf{x}^i; \theta, \phi^{i,*})$
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

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

- Note: dropped i superscript from ϕ^i for compactness
- To *evaluate* the bound, $\underline{\text{sample } \mathbf{z}^1,\cdots,\mathbf{z}^K \text{ from } q(\mathbf{z};\phi)}$ 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 $\nabla_\theta L(\mathbf{x};\theta,\phi)$ and $\nabla_\phi L(\mathbf{x};\theta,\phi)$
- 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

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

- Want to compute $\nabla_\theta L(\mathbf{x};\theta,\phi)$ and $\nabla_\phi L(\mathbf{x};\theta,\phi)$
- 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
- There is a general technique called REINFORCE (from reinforcement learning) and we might cover it later in the course.
- For now, a better but less general alternative that only works for continuous **z** (and only some distributions)

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 $\mathbf{z} \sim q(\mathbf{z};\phi)$
    - Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon = g(\epsilon;\phi)$. g is deterministic!
- Using this equivalence, we compute the expectation in two ways:

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

$$\nabla_\phi E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \nabla_\phi E_\epsilon[r(g(\epsilon;\phi))] = E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))]$$

- 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

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

- 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

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

and use chain rule for the gradient.

Amortized Inference

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

- 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 $\mathbf{x}^i \rightarrow \phi^{i,*}$
    - 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 $p_{\mathrm{data}}(\mathbf{z}, \mathbf{x}^i)$. 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

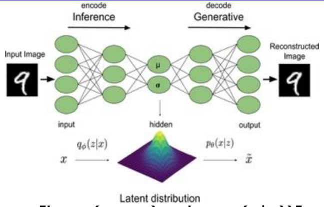
- Optimize $\sum_{\mathbf{x}^i \in D} L(\mathbf{x}^i; \theta, \phi)$ as a function of θ, ϕ using (stochastic) gradient descent

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

1. Initialize $\theta^{(0)}, \phi^{(0)}$
2. Randomly sample a data point $\mathbf{x}^i$ from D
3. Compute $\nabla_\theta L(\mathbf{x}^i; \theta, \phi)$ and $\nabla_\phi L(\mathbf{x}^i; \theta, \phi)$
4. Update θ, ϕ in the gradient direction

- How to compute the gradients? Use reparameterization like before

Autoencoder perspective



$$D_{KL}(P \| Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

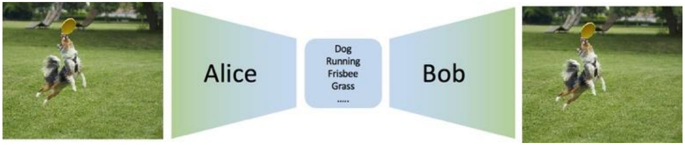
$$\begin{aligned}
L(\mathbf{x}; \theta, \phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]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

1. Take a data point $\mathbf{x}^i$, map it to $\hat{\mathbf{z}}$ by sampling from $q_\phi(\mathbf{z}|\mathbf{x}^i)$ (*encoder*). Sample from a Gaussian with parameters $(\mu, \sigma) = \textit{encoder}_\phi(\mathbf{x}^i)$
2. Reconstruct $\hat{\mathbf{x}}$ by sampling from $p(\mathbf{x}|\hat{\mathbf{z}}; \theta)$ (*decoder*). Sample from a Gaussian with parameters $\textit{decoder}_\theta(\hat{\mathbf{z}})$

What does the training objective $L(\mathbf{x}; \theta, \phi)$ do?

- First term encourages $\hat{\mathbf{x}} \approx \mathbf{x}^i$ ($\mathbf{x}^i$ likely under $p(\mathbf{x}|\hat{\mathbf{z}}; \theta)$). *Autoencoding loss*!
- Second term encourages $\hat{\mathbf{z}}$ to have a distribution similar to the prior $p(\mathbf{z})$

Autoencoder perspective



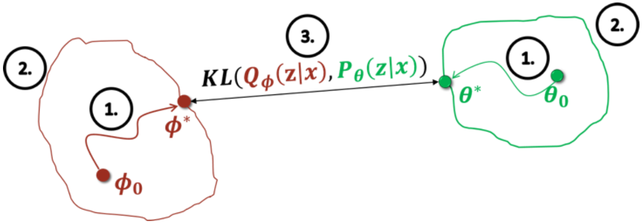
1. Alice goes on a space mission and needs to send images to Bob. Given an image $\mathbf{x}^i$, she (stochastically) compresses it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. Alice sends the message $\hat{\mathbf{z}}$ to Bob
2. Given $\hat{\mathbf{z}}$,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

Model families - Encoder

Amortization (Gershman & Goodman, 2015; Kingma; Rezende; ..)

- Scalability: Efficient learning and inference on massive datasets
- Regularization effect: Because of joint training, it also implicitly regularizes the model θ (Shu et al., 2018)

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)
- Hierarchical models where multiple VAEs are stacked on top of each other (Diffusion models)

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 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)