Lecture 9: Explicit Generative Models
Efstratios Gavves
Lecture overview

- Gentle intro to Bayesian Modelling and Variational Inference
- Restricted Boltzmann Machines
- Deep Boltzmann Machines
- Deep Belief Network
- Contrastive Divergence
- Variational Autoencoders
- Normalizing Flows
Explicit density models

- Plug in the model density function to likelihood
- Then maximize the likelihood

Problems
- Design complex enough model that meets data complexity
- At the same time, make sure model is computationally tractable
- More details in the next lecture
Bayesian Modelling
Variational Inference
How to define a generative model?

- We can define an explicit density function over all possible relations $\psi_c$ between the input variables $x_c$
  
  $$p(x) = \prod_c \psi_c(x_c)$$

- Quite inefficient $\Rightarrow$ think of all possible relations (not just pairwise) between $256 \times 256 = 65K$ input variables

- Solution: Define an energy function to model the relations between the inputs variables
Boltzmann (or Gibbs) distribution defined over a free energy function $E(x)$

$$p(x) = \frac{1}{Z} \exp(-E(x))$$

- $Z$ is the normalization factor that makes sure $\int_x p(x) \, dx = 1$
- Very expensive to compute $\rightarrow$ if $x = \{0, 1\}$ computing $Z$ requires $2^d$ computations

Better restrict the model further to a bottleneck

$$E(x) = -x^T W h - b^T x - c^T h$$
Why Boltzmann?

- In statistical mechanics and mathematics, a Boltzmann distribution (also called Gibbs distribution) is a probability distribution, probability measure, or frequency distribution of particles in a system over various possible states. The distribution is expressed in the form

\[ F(\text{state}) \propto \exp\left(-\frac{E}{kT}\right) \]

- \( E \) is the state energy, \( k \) is the Boltzmann constant, \( T \) is the thermodynamic temperature

Restricted Boltzmann Machines

- $E(x) = -x^T Wh - b^T x - c^T h$
- The $x^T Wh$ models correlations between $x$ and the latent activations via the parameter matrix $W$
- The $b^T x, c^T h$ model the priors
- Restricted Boltzmann Machines (RBM) assume $x, h$ to be binary

Restricted Boltzmann Machines (RBM) assume $x, h$ to be binary
Restricted Boltzmann Machines

- \( E(x) = -x^T Wh - b^T x - c^T h, \ \theta = \{W, b, c\} \)

- The free energy function \( F(x) = -\log \sum_h \exp(-E(x, h)) \) defines a bipartite graph with undirected connections.
  - Information flows forward and backward.
Restricted Boltzmann Machines

- The hidden units $h_j$ are independent to each other conditioned on the visible units
  \[ p(h|x) = \prod_j p(h_j|x, \theta) \]

- The hidden units $x_i$ are independent to each other conditioned on the visible units
  \[ p(x|h) = \prod_i p(x_i|h, \theta) \]
Training RBMs

- The conditional probabilities are defined as sigmoids:
  \[ p(h_j|x, \theta) = \sigma(W_{jx} + b_j) \]
  \[ p(x_i|h, \theta) = \sigma(W_{ix} + c_i) \]

- Maximize log-likelihood:
  \[ \mathcal{L}(\theta) = \frac{1}{N} \sum_n \log p(x_n|\theta) \]

- Let’s take the gradients:
  \[
  \frac{\partial \log p(x_n|\theta)}{\partial \theta} = -\frac{\partial F(x_n)}{\partial \theta} - \frac{\partial \log Z}{\partial \theta}
  \]
  \[
  = -\sum_h p(h|x_n, \theta) \frac{\partial E(x_n|h, \theta)}{\partial \theta} + \sum_{\tilde{x},h} p(\tilde{x}, h|\theta) \frac{\partial E(\tilde{x}, h|\theta)}{\partial \theta}
  \]
Training RBMs

Let’s take the gradients
\[
\frac{\partial \log p(x_n|\theta)}{\partial \theta} = - \frac{\partial F(x_n)}{\partial \theta} - \frac{\partial \log Z}{\partial \theta}
\]
\[
= - \sum_h p(h|x_n, \theta) \frac{\partial E(x_n|h, \theta)}{\partial \theta} + \sum_{\tilde{x}, h} p(\tilde{x}, h|\theta) \frac{\partial E(\tilde{x}, h|\theta)}{\partial \theta}
\]

- **Easy** because we just substitute in the definitions the \(x_n\) and sum over \(h\)
- **Hard** because you need to sum over both \(\tilde{x}, h\) which can be huge
  - It requires approximate inference, e.g., MCMC
Training RBMs with Contrastive Divergence

- Approximate the gradient with Contrastive Divergence
- Specifically, apply Gibbs sampler for $k$ steps and approximate the gradient

\[
\frac{\partial \log p(x_n | \theta)}{\partial \theta} = -\frac{\partial E(x_n, h_0 | \theta)}{\partial \theta} - \frac{\partial E(x_k, h_k | \theta)}{\partial \theta}
\]

Deep Belief Network

- RBMs are just one layer
- Use RBM as a building block
- Stack multiple RBMs one on top of the other
  \[ p(x, h_1, h_2) = p(x|h_1) \cdot p(h_1|h_2) \]
- Deep Belief Networks (DBN) are directed models
  - The layers are densely connected and have a single forward flow
  - This is because the RBN is directional, \( p(x_i|h, \theta) = \sigma(W_i x_i + c_i) \), namely the input argument has only variable only from below
Deep Boltzmann Machines

- Stacking layers again, but now with connection from the **above** and from the **below** layers.

- Since it’s a Boltzmann machine, we need an energy function:

  \[
  E(x, h_1, h_2 | \theta) = x^T W_1 h_1 + h_1^T W_2 h_2 + h_2^T W_3 h_3
  \]

- Probability function:

  \[
  p(h_2^k | h_1, h_3) = \sigma(\sum_j W_1^{jk} h_1^j + \sum_l W_3^{kl} h_3^k)
  \]
Deep Boltzmann Machines

- Schematically similar to Deep Belief Networks
- But, Deep Boltzmann Machines (DBM) are undirected models
  - Belong to the Markov Random Field family
- So, two types of relationships: bottom-up and up-bottom

\[
p(h^k_2|h_1, h_3) = \sigma(\sum_j W^{jk}_1 h^j_1 + \sum_l W^{kl}_3 h^k_3)
\]
Training Deep Boltzmann Machines

- Computing gradients is intractable
- Instead, variational methods (mean-field) or sampling methods are used

![Diagram of Deep Boltzmann Machines](image)
Bayesian Modelling
Variational Inference
Bayesian Terminology

- Observed variables $x$
- Latent variables $\theta$
  - Both unobservable model parameters $w$ and unobservable model activations $z$
  - $\theta = \{w, z\}$
- Joint probability density function (pdf): $p(x, \theta)$
- Marginal pdf: $p(x) = \int_\theta p(x, \theta) \, d\theta$
- Prior pdf $\to$ marginal over input: $p(\theta) = \int_x p(x, \theta) \, dx$
  - Usually a user defined pdf
- Posterior pdf: $p(\theta|x)$
- Likelihood pdf: $p(x|\theta)$
Bayesian Terminology

- **Posterior pdf**
  \[
  p(\theta | x) = \frac{p(x | \theta) p(\theta)}{\int p(x, \theta') d\theta'} = \frac{p(x)}{\int p(x | \theta) p(\theta) d\theta} \propto p(x | \theta) p(\theta)
  \]

  - Conditional probability
  - Bayes Rule
  - Marginal probability
  - \( p(x) \) is constant

- **Posterior Predictive pdf**
  \[
  p(x_{\text{new}} | x) = \int_{\theta} p(x_{\text{new}} | \theta) p(\theta | x) d\theta
  \]
**Bayesian Terminology**

- **Conjugate priors**
  - when posterior and prior belong to the same family, so the joint pdf is easy to compute

- **Point estimate approximations of latent variables**
  - instead of computing a distribution over all possible values for the variable, compute one point only, e.g. the most likely (maximum likelihood or max a posteriori estimate)
    \[
    \theta^* = \arg_{\theta} \max p(x|\theta)p(\theta) \quad \text{(MAP)}
    \]
    \[
    \theta^* = \arg_{\theta} \max p(x|\theta) \quad \text{(MLE)}
    \]
  - Quite good when the posterior distribution is peaky (low variance)
Bayesian Modelling

1. Estimate the posterior density $p(\theta|x)$ for your training data $x$
2. To do so, need to define the prior $p(\theta)$ and likelihood $p(x|\theta)$ distributions
3. Once the $p(\theta|x)$ density is estimated, Bayesian Inference is possible
   - $p(\theta|x)$ is a (density) function, not just a single number (point estimate)

4. But how to estimate the posterior density?
   - Markov Chain Monte Carlo (MCMC) \(\rightarrow\) Simulation-like estimation
   - Variational Inference \(\rightarrow\) Turn estimation to optimization
Variational Inference

- Estimating the true posterior $p(\theta|x)$ is not always possible
  - especially for complicated models like neural networks

- Variational Inference assumes another function $q(\theta|\phi)$ with which we want to approximate the true posterior $p(\theta|x)$
  - $q(\theta|\phi)$ is the approximate posterior
  - Note that the approximate posterior does not depend on the observable variables $x$

- We approximate by minimizing the reverse KL-divergence w.r.t. $\phi$
  \[
  \varphi^* = \arg \min_{\varphi} KL(q(\theta|\phi) \| p(\theta|x))
  \]

- Turn inference into optimization
Variational Inference (graphically)

\[ q(z; \psi) \]

\[ p(z | x) \]

\[ \text{KL}(q(z; \psi^*) || p(z | x)) = \sum q \log \frac{q}{p} \]

Underestimating variance. Why?

if \( p = 0 \) and \( q \)
which is an approximate is not \( 0 \log \frac{q}{p} \)

\[ p(d|x) \] becomes the prior to set

\[ q = 0 \]

when it camps approximately enough \( p \).
Variational Inference (graphically)

$q(z; \nu)$

$p(z | x)$

KL$(q(z; \nu^*) || p(z | x))$

Underestimating variance. Why?
Variational Inference (graphically)

How to overestimate variance?

Underestimating variance. Why?
Variational Inference (graphically)

How to overestimate variance? Forward KL

Underestimating variance. Why?
Mean-Field Approximation and CAVI Optimization

- To make the optimization of the VI easier, one can assume the latent variables are independent of each other
  \[ q(\theta | \varphi) = \prod_j q_j(\theta_j | \varphi_j) \]

- The optimization is often done with CAVI
  - Coordinate-Ascent Variational Inference
  - Initially set \( \varphi \) randomly
  - For each \( j \) in turn you set \( q_j(\theta_j | \varphi_j) = \mathbb{E}_{g_{-j}}[\log p(\theta | x)] \)
Variational Inference - Evidence Lower Bound (ELBO)

- Given latent variables $\theta$ and the approximate posterior
  $$q_\phi(\theta) = q(\theta | \phi)$$

- The log marginal is
  $$\log p(x) = \log \int_\theta p(x, \theta) \, d\theta$$
  $$= \log \int_\theta p(x, \theta) \frac{q_\phi(\theta)}{q_\phi(\theta)} \, d\theta$$
  $$= \log \mathbb{E}_{q_\phi(\theta)} \left[ \frac{p(x, \theta)}{q_\phi(\theta)} \right]$$
  $$\leq \mathbb{E}_{q_\phi(\theta)} \left[ \log \frac{p(x, \theta)}{q_\phi(\theta)} \right]$$

$$= \mathbb{E}_{q_\phi(\theta)}[ \log p(x, \theta) ] - \mathbb{E}_{q_\phi(\theta)}[ \log q_\phi(\theta) ]$$
$$= \mathbb{E}_{q_\phi(\theta)}[ \log p(x, \theta) ] + H(\theta)$$
$$= \text{ELBO}_{\theta, \phi}(x)$$

or

$$= \mathbb{E}_{q_\phi(\theta)}[ \log p(x|\theta) ] - \mathbb{E}_{q_\phi(\theta)}[ \log p(\theta) ]$$
$$+ \mathbb{E}_{q_\phi(\theta)}[ \log q_\phi(\theta) ]$$
$$= \mathbb{E}_{q_\phi(\theta)}[ \log p(x|\theta) ] - \text{KL}(q_\phi(\theta)||p(\theta))$$
$$= \text{ELBO}_{\theta, \phi}(x)$$
Variational Inference - Evidence Lower Bound (ELBO)

- Given latent variables $\theta$ and the approximate posterior $q_\phi(\theta) = q(\theta | \phi)$
- The log marginal is

$$
\log p(x) = \log \int p(x, \theta) \, d\theta = \log \int \frac{p(x, \theta)}{q(\theta | \phi)} \, q(\theta | \phi) \, d\theta
$$

by definition of $p(x)$

Jensen's inequality

$$
\log \left[ \left( \frac{p(x, \theta)}{q(\theta | \phi)} \right) \right] \leq \mathbb{E}_{q(\theta | \phi)} \left[ \log \left( \frac{p(x, \theta)}{q(\theta | \phi)} \right) \right]
$$

multiply and divide by $q(\theta | \phi)$

by definition of $q(\theta | \phi)$

= $\mathbb{E}_{q(\theta | \phi)} \left[ \log p(x, \theta) \right] - \mathbb{H} (q(\theta | \phi))$

$\mathbb{H}(q(\theta | \phi))$

Note:

$$
\mathbb{E}_{q(\theta | \phi)} \left[ \log q(\theta | \phi) \right] = \mathbb{E}_{q(\theta | \phi)} \left[ \log q(\theta | \phi) \right] - \mathbb{E}_{q(\theta | \phi)} \left[ \log q(\theta | \phi) \right]
$$

$\mathbb{E}_{q(\theta | \phi)} \left[ \log q(\theta | \phi) \right]$
It is easy to see that the ELBO is directly related to the marginal

\[
\text{ELBO}_{\theta, \varphi}(x) =
\]

\[
= \mathbb{E}_{q_\varphi(\theta)}[\log p(x, \theta)] - \mathbb{E}_{q_\varphi(\theta)}[\log q_\varphi(\theta)]
\]

\[
= \mathbb{E}_{q_\varphi(\theta)}[\log p(\theta|x)] + \mathbb{E}_{q_\varphi(\theta)}[\log p(x)] - \mathbb{E}_{q_\varphi(\theta)}[\log q_\varphi(\theta)]
\]

\[
= \mathbb{E}_{q_\varphi(\theta)}[\log p(x)] - KL(q_\varphi(\theta)||p(\theta|x))
\]

\[
= \log p(x) - KL(q_\varphi(\theta)||p(\theta|x)) \quad \iff \log p(x) \text{ does not depend on } q_\varphi(\theta)
\]

\[
\Rightarrow \quad \log p(x) = \text{ELBO}_{\theta, \varphi}(x) + KL(q_\varphi(\theta)||p(\theta|x)) \quad \iff \mathbb{E}_{q_\varphi(\theta)[1]}=1
\]

You can also see \(\text{ELBO}_{\theta, \varphi}(x)\) as Variational Free Energy
It is easy to see that the ELBO is directly related to the marginal:

$$\text{ELBO}_{\theta,\varphi}(x) =$$
ELBO interpretations

- \( \log p(x) = \text{ELBO}_{\theta, \varphi}(x) + KL(q_\varphi(\theta) || p(\theta | x)) \)
- The log-likelihood is constant, as it does not depend on any parameter
- Also, both \( \text{ELBO}_{\theta, \varphi}(x) > 0 \) and \( KL(q_\varphi(\theta) || p(\theta | x)) > 0 \)

1. The higher the Variational Lower Bound \( \text{ELBO}_{\theta, \varphi}(x) \), the smaller the difference between the approximate posterior \( q_\varphi(\theta) \) and the true posterior \( p(\theta | x) \) \( \Rightarrow \) better latent representation

2. The Variational Lower Bound \( \text{ELBO}_{\theta, \varphi}(x) \) approaches the log-likelihood \( \Rightarrow \) better density model
Amortized Inference

- The variational distribution $q(\theta|\phi)$ does not depend directly on data.
  - Only indirectly, via minimizing its distance to the true posterior $KL(q(\theta|\phi) || p(\theta|x))$.
- So, with $q(\theta|\phi)$ we have a major optimization problem, as the approximate posterior must approximate the whole dataset $x = [x_1, x_2, ..., x_N]$ jointly.
- As this is obviously quite complex, one can amortize the optimization on individual data points by setting $q(\theta|\phi) = q_{\phi}(\theta|x)$.
- Predict model parameters $\theta$ using a $\phi$-parameterized model of the input $x$.
- Use it for parameters that depend on data, such as the latent activations.
Amortized Inference (Intuitively)

- Originally, Variational Inference assumed that $q(\theta|\varphi)$ describes the approximate posterior of the dataset as a whole.
  - Think of $\theta$ not as the latent activations $z$, but only the latent model variables $w$. 

[Diagram showing distributions $P(\theta(x))$, $q(\theta|\varphi)$, and $P(x)$]
Variational Autoencoders

- Let’s rewrite the ELBO a bit more explicitly
  \[
  \text{ELBO}_{\theta, \phi}(x) = \mathbb{E}_{q(\theta) \phi}(\log p(x|\theta)) - \text{KL}(q(\theta) \phi||p(\theta)) \\
  = \mathbb{E}_{q(\phi)(z|x)}(\log p(\theta)(x|z)) - \text{KL}(q(\phi)(z|x)||p(\lambda)(z))
  \]

- Instead of \( p(x|\theta) \) we have \( p(\theta)(x|z) \) to indicate that the model for the posterior density has weights parameterized by \( \theta \) and latent model activations parameterized by \( z \).

- Instead of \( p(\theta) \) we have \( p(\lambda)(z) \), namely we put a \( \lambda \)-parameterized prior only on the latent activations \( z \) and not the model weights.

- Instead of \( q(\theta|\phi) \) we have \( q(\phi)(z|x) \) to indicate that the model approximates the posterior density of the latent activations, and the model weights are parameterized by \( \phi \).
Variational Autoencoders

- So, we have $\text{ELBO}_{\theta, \varphi}(x) = \mathbb{E}_{q_{\varphi}(z|x)}[\log p_{\theta}(x|z)] - \text{KL}(q_{\varphi}(z|x) \| p_{\lambda}(z))$

- What if we model the densities $p_{\theta}(x|z)$ and $q_{\varphi}(z|x)$ as neural networks?

- The approximate posterior looks like a standard ConvNet (or MLP), which receives an image input $x$ and returns a feature map/latent variable $z$
  - Also known as encoder or inference network

- The likelihood term $p_{\theta}(x|z)$ looks like an inverted ConvNet (deconvolutions), which given a latent feature map $z$ reconstructs the input $x$
  - Also known as decoder or generator network, because it recognizes the input given the latent variable

- A difference from a standard autoencoder is we now have an opinion of what the distribution of the latents $z$ should look like, with $p_{\lambda}(z)$
Training Variational Autoencoders

- Maximize the Evidence Lower Bound (ELBO)
  - Or minimize the negative ELBO
    \[
    \mathcal{L}(\theta, \varphi) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\varphi(z|x) \| p_\lambda(z))
    \]
- How do we optimize the ELBO?
Training Variational Autoencoders

- Maximize the Evidence Lower Bound (ELBO)
  - Or minimize the negative ELBO
    \[ \mathcal{L}(\theta, \varphi) = \mathbb{E}_{q_\varphi(Z|x)}[\log p_\theta(x|Z)] - \text{KL}(q_\varphi(Z|x)||p_\lambda(Z)) \]
    \[ = \int_Z q_\varphi(z|x) \log p_\theta(x|z) \, dz - \int_Z q_\varphi(z|x) \log \frac{q_\varphi(z|x)}{p_\lambda(z)} \, dz \]

- Forward propagation → compute the two terms

- The first term is an integral (expectation) that we cannot solve analytically. So, we need to sample from the pdf instead
  - When \( p_\theta(x|z) \) contains even a few nonlinearities, like in a neural network, the integral is hard to compute analytically
Complex integrals
Training Variational Autoencoders

- Maximize the Evidence Lower Bound (ELBO)
  - Or minimize the negative ELBO

\[
\mathcal{L}(\theta, \varphi) = \mathbb{E}_{q_{\varphi}(Z|x)}[\log p_{\theta}(x|Z)] - \text{KL}(q_{\varphi}(Z|x)||p_{\lambda}(Z))
\]

\[
= \int q_{\varphi}(z|x) \log p_{\theta}(x|z) \, dz - \int q_{\varphi}(z|x) \log \frac{q_{\varphi}(z|x)}{p_{\lambda}(z)} \, dz
\]

- Forward propagation → compute the two terms
- The **first term** is an integral (expectation) that we cannot solve analytically. So, we need to sample from the pdf instead
  - When \( p_{\theta}(x|z) \) contains even a few nonlinearities, like in a neural network, the integral is hard to compute analytically
- The **second term** is the KL divergence between two distributions that we know
Typical VAE

- We set the prior $p_\lambda(Z)$ to be the unit Gaussian
  $p(Z) \sim N(0, 1)$

- We set the likelihood to be a Bernoulli for binary data
  $p(X|Z) \sim Bernoulli(\pi)$

- We set $q_\phi(Z|x)$ to be a neural network (MLP, ConvNet), which maps an input $x$ to the Gaussian distribution, specifically it’s mean and variance
  - $\mu_z, \sigma_z \sim q_\phi(Z|x)$
  - The neural network has two outputs, one is the mean $\mu_x$ and the other the $\sigma_x$, which corresponds to the covariance of the Gaussian

- We set $p_\theta(X|Z)$ to be an inverse neural network, which maps $Z$ to the Bernoulli distribution if our outputs binary (e.g. Binary MNIST)
VAE: Interpolation in the latent space
Forward propagation in VAE

- Sample $z$ from the approximate posterior density $z \sim q_\phi(Z|x)$
  - As $q_\phi$ is a neural network that outputs values from a specific and known parametric pdf, e.g. a Gaussian, sampling from it is rather easy
  - Often even a single draw is enough

- Second, compute the $\log p_\theta(x|Z)$
  - As $p_\theta$ is a a neural network that outputs values from a specific and known parametric pdf, e.g. a Bernoulli for white/black pixels, computing the log-prob is easy

- Computing the ELBO is rather straightforward in the standard case

- How should we optimize the ELBO?
Forward propagation in VAE

- Sample $z$ from the approximate posterior density $z \sim q_{\varphi}(Z|x)$
  - As $q_{\varphi}$ is a neural network that outputs values from a specific and known parametric pdf, e.g. a Gaussian, sampling from it is rather easy
  - Often even a single draw is enough

- Second, compute the $\log p_{\theta}(x|Z)$
  - As $p_{\theta}$ is a neural network that outputs values from a specific and known parametric pdf, e.g. a Bernoulli for white/black pixels, computing the log-prob is easy

- Computing the ELBO is rather straightforward in the standard case

- How should we optimize the ELBO? Backpropagation?
Backward propagation in VAE

- Backpropagation \(\Rightarrow\) compute the gradients of
  \[ \mathcal{L}(\theta, \phi) = \mathbb{E}_{z \sim q_{\phi}(z|x)}[\log p_{\theta}(x|z)] - \text{KL}(q_{\phi}(Z|x) || p_{\lambda}(Z)) \]

- \(\nabla_{\theta} \mathcal{L} = \mathbb{E}_{z \sim q_{\phi}(z|x)}[\nabla_{\theta} \log p_{\theta}(x|z)] \)
  - The expectation and sampling in \(\mathbb{E}_{z \sim q_{\phi}(z|x)}\) does not depend on \(\theta\), so no problem!
  - Also, the KL does not depend on \(\theta\), so no gradient from over there!

- \(\nabla_{\phi} \mathcal{L} = \nabla_{\phi} \left[ \mathbb{E}_{z \sim q_{\phi}(z|x)}[\log p_{\theta}(x|z)] \right] - \nabla_{\phi} \left[ \text{KL}(q_{\phi}(Z|x) || p_{\lambda}(Z)) \right] \)
Backward propagation in VAE

- Backpropagation $\Rightarrow$ compute the gradients of
  \[
  \mathcal{L}(\theta, \varphi) = \mathbb{E}_{z \sim q_\varphi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\varphi(Z|x)||p_\lambda(Z))
  \]
Backward propagation in VAE

Backpropagation $\rightarrow$ compute the gradients of

$$\mathcal{L}(\theta, \varphi) = \mathbb{E}_{z \sim q_{\varphi}(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_{\varphi}(Z|x)\|p_\lambda(Z))$$

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{z \sim q_{\varphi}(z|x)}[\nabla_\theta \log p_\theta(x|z)]$$

- The expectation and sampling in $\mathbb{E}_{z \sim q_{\varphi}(z|x)}$ does not depend on $\theta$, so no problem!
- Also, the KL does not depend on $\theta$, so no gradient from over there!

$$\nabla_\varphi \mathcal{L} = \nabla_\varphi \left[\mathbb{E}_{z \sim q_{\varphi}(z|x)}[\log p_\theta(x|z)]\right] - \nabla_\varphi \left[\text{KL}(q_{\varphi}(Z|x)\|p_\lambda(Z))\right]$$

- Problem?
Backward propagation in VAE

- Backpropagation $\rightarrow$ compute the gradients of
  \[ L(\theta, \varphi) = \mathbb{E}_{z \sim q_\varphi(z|x)}[\log p_\theta(x|z)] - KL(q_\varphi(Z|x)||p_\lambda(Z)) \]

- $\nabla_\theta L = \mathbb{E}_{z \sim q_\varphi(z|x)}[\nabla_\theta \log p_\theta(x|z)]$
  - The expectation and sampling in $\mathbb{E}_{z \sim q_\varphi(z|x)}$ does not depend on $\theta$, so no problem!
  - Also, the KL does not depend on $\theta$, so no gradient from over there!

- $\nabla_\varphi L = \nabla_\varphi \left[ \mathbb{E}_{z \sim q_\varphi(z|x)}[\log p_\theta(x|z)] \right] - \nabla_\varphi \left[ KL(q_\varphi(Z|x)||p_\lambda(Z)) \right]$

- Problem? Sampling $z \sim q_\varphi(Z|x)$ is not differentiable $\rightarrow$ no gradients
- No gradients $\rightarrow$ No backprop $\rightarrow$ No training! $\rightarrow$ Solution?
So, our latent variable $Z$ is a Gaussian (in the standard VAE) represented by the mean and variance $\mu_Z, \sigma_Z$, which are the output of a neural net.

So, we can train by sampling randomly from that Gaussian:

$$z \sim N(\mu_Z, \sigma_Z)$$

Once we have that $z$, however, it’s a fixed value (not a function), so we cannot backprop.

We could use, however, the REINFORCE algorithm to compute an approximation to the gradient:

- High-variance gradients $\rightarrow$ slow and not very effective learning.
Solution: Reparameterization trick

- Remember, we have a Gaussian output $z \sim N(\mu_Z, \sigma_Z)$.

- For certain pdfs, including the Gaussian, we can rewrite their random variable $z$ as deterministic transformations of a simpler random variable $\varepsilon$.

- For the Gaussian specifically, the following two formulations are equivalent:

$$z \sim N(\mu_Z, \sigma_Z) \iff z = \mu_Z + \varepsilon \cdot \sigma_Z,$$

where $\varepsilon \sim N(0, 1)$ and $\mu_Z, \sigma_Z$ are deterministic values from the NN function.
Instead of sampling from \( z \sim N(\mu_Z, \sigma_Z) \), we sample from \( \epsilon \sim N(0,1) \) and then we compute \( z \).

- Sampling directly from \( z \sim N(\mu_Z, \sigma_Z) \) leads to high-variance estimates.
- Sampling directly from \( \epsilon \sim N(0,1) \) leads to low-variance estimates.

- Remember: since we are sampling for \( z \), we are also sampling gradients.
- More distributions beyond Gaussian possible: Laplace, Student-t, Logistic, Cauchy, Rayleight, Pareto.

**Solution: Reparameterization trick**

- High-variance gradient
- Low-variance gradient
Again, the latent variable is \( z = \mu_z + \epsilon \cdot \sigma_z \)

- \( \mu_z \) and \( \sigma_z \) are deterministic functions (via the neural network encoder)
- \( \epsilon \) is a random variable, which comes \textit{externally}
- The \( z \) as a result is itself a random variable, because of \( \epsilon \)
- However, now the randomness is not associated with the neural network and its parameters that we have to learn
  - The randomness instead comes from the external \( \epsilon \)
  - The gradients flow through \( \mu_z \) and \( \sigma_z \)
Reparameterization Trick (graphically)

Original form

- \( z \sim q(z|\phi, x) \)

Reparameterised form

- \( \frac{\partial f}{\partial z_j} = g(\phi, x, \epsilon) \)
- \( \frac{\partial f}{\partial \phi_i} = \frac{\partial L}{\partial \phi_i} \)
- \( \epsilon \sim p(\epsilon) \)

Backprop

- \( f \)
- \( z \)
- \( \phi \)
- \( x \)
- \( \epsilon \)
- \( \theta \)
- \( N \)

: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]
VAE Training Pseudocode

Data:
\[ \mathcal{D}: \text{Dataset} \]
\[ q_{\phi}(z|x): \text{Inference model} \]
\[ p_{\theta}(x,z): \text{Generative model} \]

Result:
\[ \theta, \phi: \text{Learned parameters} \]

\[(\theta, \phi) \leftarrow \text{Initialize parameters} \]

while SGD not converged do
  \[ \mathcal{M} \sim \mathcal{D} \text{ (Random minibatch of data)} \]
  \[ \epsilon \sim p(\epsilon) \text{ (Random noise for every datapoint in } \mathcal{M}) \]
  Compute \[ \tilde{\mathcal{L}}_{\theta,\phi}(\mathcal{M}, \epsilon) \] and its gradients \[ \nabla_{\theta,\phi} \tilde{\mathcal{L}}_{\theta,\phi}(\mathcal{M}, \epsilon) \]
  Update \[ \theta \text{ and } \phi \text{ using SGD optimizer} \]
end

The ELBO’s gradients
Figure 2.D.2: An application of VAEs to interpolation between pairs of sentences, from [Bowman et al., 2015]. The intermediate sentences are grammatically correct, and the topic and syntactic structure are typically locally consistent.
Figure 2.D.3: VAEs can be used for image re-synthesis. In this example by White [2016], an original image (left) is modified in a latent space in the direction of a smile vector, producing a range of versions of the original, from smiling to sadness. Notice how changing the image along a single vector in latent space, modifies the image in many subtle and less-subtle ways in pixel space.
Figure 2.D.1: Example application of a VAE in [Gómez-Bombarelli et al., 2016]: design of new molecules with desired chemical properties. (a) A latent continuous representation $\mathbf{z}$ of molecules is learned on a large dataset of molecules. (b) This continuous representation enables gradient-based search of new molecules that maximizes some chosen desired chemical property given by objective function $f(\mathbf{z})$. 
Using simple pdfs, like a Gaussian, for the approximate posterior limits the expressivity of the model.

Better make sure the approximate posterior comes from a class of models that can even contain the true posterior.

Use a series of $K$ invertible transformations to construct the approximate posterior:

$$z_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(z_0)$$

Rule of change for variables:

Changing from the $x$ variable to $y$ using the transformation $y = f(x) = 2x + 1$
Normalizing Flows

Sampling and Entropy

\[ z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0) \]

\[ \log q_K(z_K) = \log q_0(z_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial z_k} \right| \]

Distribution flows through a sequence of invertible transforms

https://blog.evjang.com/2018/01/nf1.html
Normalizing Flows

[Image showing Normalizing Flows with different distributions and transformations]

Normalizing Flows on Non-Euclidean Manifolds

Figure 1: Left: Construction of a complex density on $\mathbb{S}^n$ by first projecting the manifold to $\mathbb{R}^n$, transforming the density and projecting it back to $\mathbb{S}^n$. Right: Illustration of transformed ($\mathbb{S}^2 \rightarrow \mathbb{R}^2$) densities corresponding to an uniform density on the sphere. Blue: empirical density (obtained by Monte Carlo); Red: Analytical density from equation (4); Green: Density computed ignoring the intrinsic dimensionality of $\mathbb{S}^n$.

Gemici et al., 2016

$\log q_K(z_K) = \log q_0(z_0) - \frac{1}{2} \sum_{k=1}^{\infty} \log \det \left| J_\phi^T J_\phi \right|$

Normalizing Flows on Non-Euclidean Manifolds
Summary

- Gentle intro to Bayesian Modelling and Variational Inference
- Restricted Boltzmann Machines
- Deep Boltzmann Machines
- Deep Belief Network
- Contrastive Divergence
- Variational Autoencoders
- Normalizing Flows