

# Lecture 4: Flows and Latent Variable Models

*Scribed by: Taha Aksu, Shen Ting Ang, Song Kai, Daniel Maung*

*Lecture presented by:*

## Flows

### Autoregressive Flows

The generation path needs to be done sequentially.

The inference path does not depend on any previous inferences i.e. it can be done in parallel.

### Inverse Autoregressive Flows

The inference path is slow this time.

The sampling will be efficient as it can be done in parallel.

Note that: Combining AF and IAF will result in the worse of the two worlds, slow sampling and slow inferencing.

### Continuous flows for discrete data

Problem: Degeneracy due to using discrete data.

Solution: Dequantization

Add noise  $u \sim \text{Uniform}[0,1)$

## Latent Variable Models

Latent Variable Model	Autoregressive Model
Some of the variables are hidden	All variables are observed
More efficient as pixels need not depend on each other (already described by the latent space)	Slow as pixels depend on each other

### The sampling process for LVM:

1) Sample some  $z \sim p(z, B)$

2) Sample  $x$  through another distribution  $\sim q(x, \text{DNN}(z))$

Note that the parameters of the second distribution is learned through a DNN whose input is  $z$ .

The Challenge: the latent variable is usually unknown

### Training LVMs

Use Maximum Likelihood.

$$\log p(x) = \log \left( \sum_z p(x|z)p(z) \right)$$

Computing  $p(x)$  is of  $O(n^2)$  (The sum over all  $z$  makes it expensive) so we are looking at ways to generate it in a computationally efficient manner. One way is to use the following equation:

$$p(x) = \frac{p(x|z)p(z)}{p(z|x)}$$

$p(z)$  and  $p(x|z)$  are  $O(1)$  in complexity, so if we have  $p(z|x)$  we have an efficient computation.

The problem is that we do not have  $p(z|x)$  and it is intractable. So following is another equation to get  $p(z|x)$  :

$$p(z|x) = \frac{p(x, z)}{p(x)} = \frac{p(x, z)}{\sum_z p(x|z)p(z)}$$

But still there is another intractable term here:  $p(x|z)$ . This is where variational inference becomes useful. Variational inference will be used in order to find an approximation of the true posterior  $p(z|x)$ .

### Variational Inference

**Aim: Learn an approximation of the intractable posterior**

Use empirical data to approximate the posterior

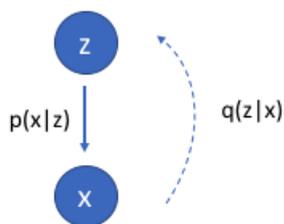
Let  $q(z)$  be the approximation of the true posterior  $p(z|x)$ , then the optimization problem is to minimize the difference between  $q$  and  $p$ , i.e. KL Divergence of  $q$  wrt  $p$  should ideally be 0.

## Variational Lower Bound

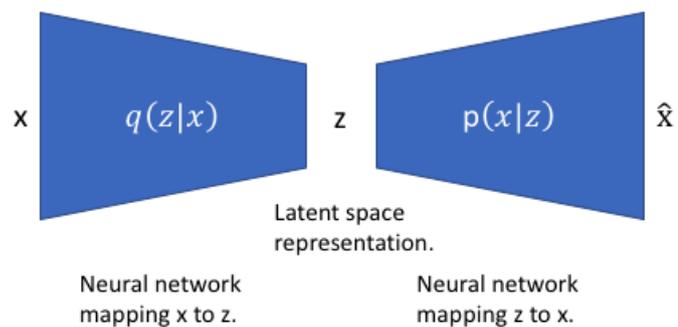
We use an unusual way of defining KL Divergence by taking projection of  $q$  onto  $p$ .

By pulling out terms without  $z$  from the formula of KL Divergence, we are able to get the variational lower bound, and our goal becomes maximizing VLB.

The bound is tight as when  $q = p$ , and the KL divergence equals 0, so the second term disappears and  $\log p(x) = \text{Variational Lower Bound}$



We'd like to use our observations to understand the hidden variable.



## VLB Optimization

### Wake-Sleep algorithm

**Idea: Generate  $z$  from data distribution**

**Intuition: Generate samples from our own "belief", and have posterior to match our belief**

#### Wake Phase

- Sample  $x \sim p_{\text{data}}, z \sim q(z; \phi(x))$
- Maximize VLB wrt.  $\theta, \beta$

#### Sleep Phase

- Sample  $z \sim p(z; \beta), x \sim p(x|z; \theta)$
- Minimize  $\text{KL}(p(z|x) || q(z; \phi(x)))$  wrt.  $\phi$  now that we have samples from  $p_{\text{model}}$ .

Doesn't guarantee bound tightness as  $x \sim p_{\text{model}}$  is not necessarily  $x \sim p_{\text{data}}$   
Flawed but still usable

## Amortized Inference

With the Wake-Sleep algorithm, each data point has its own posterior

- Not scalable to large dataset
- Expensive to evaluate new data points

A neural network could be implemented to map from data points to the noise space

## Helmholtz Machine

See paper by Dayan, Hinton et al., learned with wake-sleep algorithm, thus did not scale due to flaws of wake-sleep.

## Direct Optimization

Not often feasible as the gradients have high variance and require a large sample size to obtain a good estimate.

## Variational Autoencoders

### Pathwise Derivative

Convert  $z$  to be continuous using reparameterization, then use stochastic gradient descent.

Model  $q_{\phi}(z|x)$  as a Gaussian with parameters  $\mu$  and  $\sigma$ , a DNN encoder of  $x$ . The decoder  $p_{\theta}(x|z)$  is differentiable.

SGD can now efficiently compute both  $\nabla_{\phi}[VLB]$  and  $\nabla_{\theta}[VLB]$ .

**Aim: Encourage decoder to learn to reconstruct data**

## Improving VAE

Performance is not as good as AR models on MNIST.

How to bridge the gap:

- Improve Variational Inference

- Flexible Decoder
- More expressive model architectures

Some helpful links:

1. Glow paper: <https://arxiv.org/pdf/1807.03039.pdf>
2. Guide to Variational Inference by David Blei:  
<https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf>
3. <https://jaan.io/what-is-variational-autoencoder-vae-tutorial/>
4. Wake Sleep Algorithm Paper by Geoffery Hinton:  
<https://www.cs.toronto.edu/~hinton/absps/ws.pdf>
5. Application of Variational Inference - Latent Dirichlet Allocation:  
<https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158>
6. Helmholtz Machine by Dayan, Hinton, et al.:  
<http://www.gatsby.ucl.ac.uk/~dayan/papers/hm95.pdf>
7. What is Amortized Variational Inference?:  
<https://www.quora.com/What-is-amortized-variational-inference>
8. <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>
9. Auto-encoding Variational Bayes: <https://arxiv.org/pdf/1312.6114.pdf>
10. (Auto-encoders) Reducing the dimensionality of data with Neural Networks by Hinton and Salakhutdinov: <http://www.cs.toronto.edu/~hinton/science.pdf>