13. Introduction to Generative Models

"Deep Learning" Course (2020)

Takyoung Kim

This is an informal lecture review of Prof. Sungroh Yoon @ Seoul National University.
I would like to express my appreciation to the professor for providing an insightful lecture.
Lectures are publicly opened in this YouTube.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Supervised learning vs. Unsupervised learning

The goal between supervised and unsupervised learning differs as follows:

- Supervised: learning a function to map $x \rightarrow y$
- Unsupervised: learning inherent structure of the data

1.2 Discriminative model vs. Generative model

In this section, we assume only supervised learning. The various difference between the two models are indicated on Table 1.

Table 1. The difference between discriminative and generative model

	Discriminative	**Generative**
Goal	directly estimate $p(y \mid x)$	estimate $p(x \mid y)$, then deduce $p(y \mid x)$
Should evaluate	$f(x) = argmax_y p(y \mid x)$	$f(x) = argmax_y p(x \mid y)p(y)$
What's learned	decision boundary	probability distribution of data
Examples	SVM	Gaussian mixture, Bayes nets

1.3 Generative modeling in unsupervised learning

Generative models can be applied to many tasks, especially density estimation and sample generation. The goal of density estimation is to learn $p(x)$, the data distribution. It means finding the parameter θ which explains the training data well. If the density of the data is well estimated, the sample generation will work well. Sample generation literally means sampling a data from the distribution of the trained model. The intuitive comparison of sampled data and its distribution is indicated below:

- training data (observation) $\sim p_{data}(x)$
- generated samples $\sim p_{model}(x)$

We can check the taxonomy of generative models in Figure 1[1]. In this figure, the `Explicit density` means explicitly defining the density and solving for $p_{model}(x)$, whereas the `Implicit density` means learning a model that can sample from $p_{model}(x)$ without clearly defining the distribution. A Generative Adversarial Network (GAN) is a representative example of generative model with implicit density.

[1] Figure from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

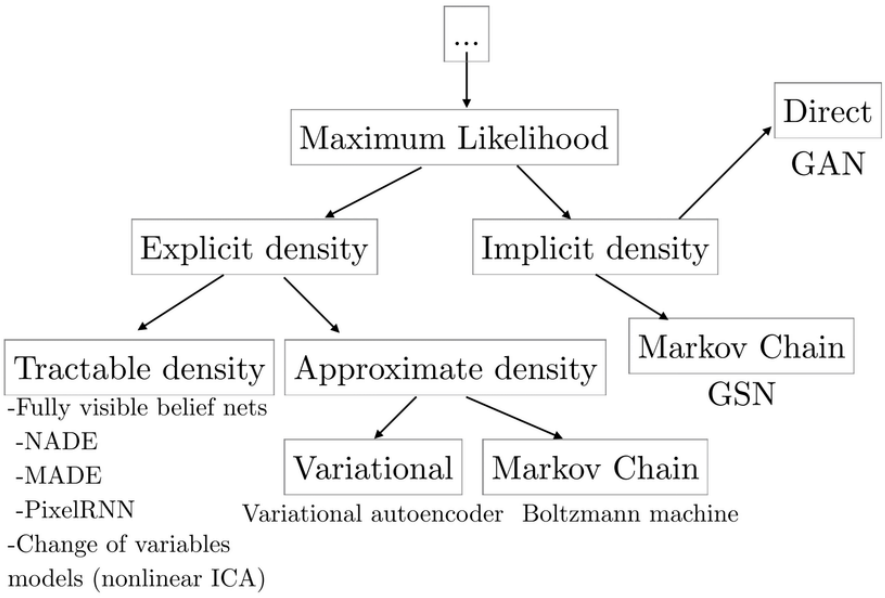

Fig. 1. Taxonomy of generative models.

Further looking types of the `Approximate density`, there are two types: `Variational` and `Markov Chain`. The former, `Variational`, says deterministic approximation, which has advantage of scalability but has relatively low quality. However the `Markov Chain` is a stochastic approximation which guarantees high quality if given enough time and resources (but it cannot be given infinite costs in practice).

1.4 Trends in generative models

There are some drawbacks of conventional generative models. It can be summarized as follows:

1. It requires strong assumptions about structures in data because of calculation availability.
2. It results in suboptimal ones if not approximated well.
3. It relies on computationally expensive inference procedures
   (e.g. Markov Chain Monte Carlo; MCMC)

Recently, training neural nets through backpropagation is a powerful function approximators. Variational Autoencoder (VAE) is a nice example of weak structural assumption and fast training.

1.5 Generative models based on neural nets

There are three featured types of generative models based on neural nets: autoregressive models, Helmholtz machines, and Generative Adversarial Networks (GAN).

First, autoregressive models, also known as fully visible belief nets, model $p(x)$ as Equation 1. Models such as PixelRNN, PixelCNN, and WaveNet are examples of autoregressive models.

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}) \tag{1}$$

Second, Helmholtz machines model $p(x)$ as Equation 2. It assumes a latent variable z, and it can be decompose with recognition net (i.e., encoder) and generative net (i.e., decoder). These models use variational inference to maximize ELBO, which will handle later. Variational Autoencoder belongs to here.

$$p(x) = \int p(z)p(x \mid z)\, dz \tag{2}$$

Lastly, GAN does not model the density explicitly. It consists of generator and discriminator, and trains model by solving minimax problem.

1.6 Why study generative models?

There are many reasons:

- It is an excellent opportunity to test our ability to represent/manipulate high-dimensional or complicated probability distributions.
- It can be incorporated into reinforcement learning (e.g., simulate possible features for planning).
- It can be trained with missing data. That is, it can provide predictions on inputs that are missing and extend to semi-supervised learning.
- It enables machine learning to work with multimodal outputs.
  (e.g., a single input $\rightarrow$ many different correction answers)
- It enables inference of latent representations, which can be useful as general features.
- Many tasks require realistic generation of samples.
  (e.g., single image super-resolution, art creation, image-to-image translation)

2. Autoregressive Models

2.1 Fully visible belief network (FVBN)

It is the explicit density model, as shown as Figure 1. Its process is listed below:

1. Decompose likelihood of each input x (i.e., $p(x)$) into a product of 1-D distributions (See Equation 1).
2. Maximize likelihood of the training data.

It has complex distribution over feature values because of condition term, so we express it using nerual net. The main issue of FVBN is that we need to define the ordering of "previous feature".

2.2 PixelRNN

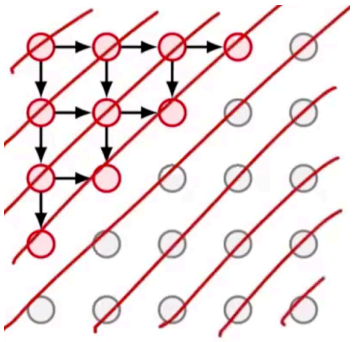

Fig. 2. The process of PixelRNN.

Its core idea is the same as language modeling applied to image. It generates image pixels starting from the corner, and models dependency on previous pixels using 2-D LSTM. Figure 2[2] shows the sequential process of PixelRNN. As illustrated in the figure, it processes pixels sequentially, so has a limitation of speed.

2.3 PixelCNN

PixelCNN still generates image pixels starting from the corner. Differences are just using CNN (not LSTM) to model dependency on previous pixels, and using two convolutional stacks to remove blind spots. Figure 3[3] illustrates changes above. It is a bit faster than PixelRNN, but still slow. The training objective is to maximize likelihood of training images by minimizing the softmax loss at each pixel (0 to 255).

[2]Capture from the lecture.
[3]Capture from the lecture.

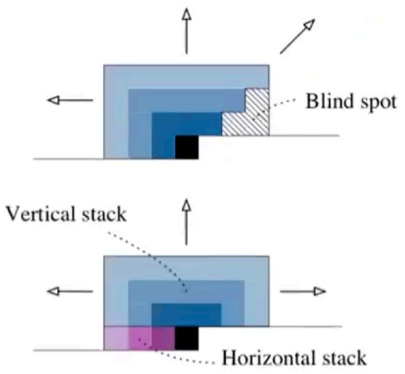

Fig. 3. The process of PixelCNN.

2.4 WaveNet

It is interesting that the author of PixelRNN, PixelCNN, and WaveNet is the same. WaveNet is an audio application of autoregressive model. It can generate raw audios trained on raw audio waveforms. The structure is similar to PixelCNN structure but much more successful – widely used. Its generation ability is amazing, but it takes 2-minute to synthesize a second audio. It is a chronic drawback of autoregressive models slow at inference.

A summarized features of WaveNet is below:

- It uses dilated convolution to vastly increase receptive field.
- It uses classification (not regression) to generate next sample point.
- It converts continuous audio with 256-level quantized classes.

2.5 Remarks

The autoregressive models have the following features:

- It can explicitly compute $p(x)$.
- It generally makes good samples.
- Its sequential generation process is slow.

In order to imporve performance, we can apply some skills:

- Use gated convolutional layers.
- Add shorcut connections.
- Use discretized logistic loss.
- Apply multi-scale methods.

- Apply parallelization.
- etc.

3. Approximate Inference - Introduction

3.1 Probabilistic machine learning

A probabilistic model denotes a joint distribution of hidden variable z and observed variable x. It can be expressed as $p(z,x)$, describing how (a portion of) the world works. By calculating the posterior $p(z \mid x)$, i.e. conditional distribution of hidden variable given the observation, we can inference about the unknowns.

By the way, **why posterior is important?** We can find the answer in Equation 3. According to the equation, the posterior links true data distribution (denominator) and model distribution (numerator). In other words, it represents the relation between data and model, so is used in all downstream analysis. Therefore, posterior inference is a central task in probabilistic models.

$$p(z \mid x) = \frac{p(z,x)}{p(x)} \tag{3}$$

3.2 Posterior inference

Posterior inference refers to the following:

1. Computing posterior distribution $p(z \mid x)$.
2. Taking expectations computed w.r.t. this distribution.

The example of the latter is an expectation-maximization (EM) algorithm. It evaluates expectation of complete-data log likelihood w.r.t. posterior distribution of latent variables (we will not handle in detail). For many models of practical interest, it is intractable. It causes challenge of inference which makes it difficult to train probabilistic models.

3.3 Challenge of inference

General reasons for challenging inference are below:

- The dimensionality of latent space is too high to work with directly.
- The posterior is highly complex, so expectations are not analytically tractable.

In the scope of deep learning, the reasons are below:

- There are too many interactions between hidden variables (e.g. conections between layers).

- Most neural nets with multiple layers of hidden variables have intractable posterior distributions.

A solution for these is **approximate posterior inference**.

3.4 Bayesian view

In Bayesian view, we setup the general problem. First we consider a joint density of latent variables z and observations x. Then we follow Bayes rule, formulating as Equation 4. In this case, latent variables help govern distribution of data in Bayesian models.

$$p(z,x) = p(z)p(x \mid z) \tag{4}$$

To sum up, a Bayesian model draws latent variables from a prior $p(z)$, then relates them to observations through likelihood $p(x \mid z)$. And inference in a Bayesian model conditions on data and computes posterior $p(z \mid x)$, which requires approximate inference.

3.5 Approximate posterior inference

There are two important types of approximate posterior inference.

3.5.1 Stochastic approach (e.g. Markov Chain Monte Carlo; MCMC)

It assumes that the model can generate exact results when given 'infinite' computational resources. Its approximation arises from using a finite amount of time.

These sampling methods can be computationally demanding, so their use is often limited to small-scale problems. And it is also difficult to know whether independent samples are being generated because it depends on random walks. Additionally it yields a large variance, as the same results of random walks.

3.5.2 Deterministic approach (e.g. Variational Inference; VI)

This approach relatively scales well to large applications. It is based on analytical approximation to posterior $p(z \mid x)$, sacrificing accuracy a bit. In this view of inference, some assumptions exist such as the posterior factorizes or has a parametric form like Gaussian.

3.6 Markov Chain Monte Carlo

The MCMC method has been dominant paradigm for decades. Its process is summarized as follows:

1. Construct an ergodic[4] Markov chain on z.

[4]No matter where you start, you will eventually reach any other points when keep running.

2. Sample from the chain to collect samples from the stationary distribution.
3. Approximate the posterior with an empirical estimate, constructed from (a subset of) the collected samples.

It is widely studied, extended, and applied to many methods such as Metropolis-Hastings algorithms or Gibbs Sampling. The goal of this method is to sample from a small subset of high-dimensional region where sampling is hard. The name "Monte Carlo" means performing a random walk, and the name "Markov Chain" supports the random walk according to sequential information. Below is more intuitive process of reaching a goal in MCMC, and MCMC method means doing this process by forming a Markov Chain.

1. Often p has some (connected) structure.
2. Start somewhere, and start moving around.
3. Try to move toward a region of high probability, and then get there.
4. Try to stay in the regions of high probability, doing random walks.

Its critical drawback is that the speed is too slow for large data sets and complex models (variational inference is a good alternative in this settings).

3.7 Variational Inference

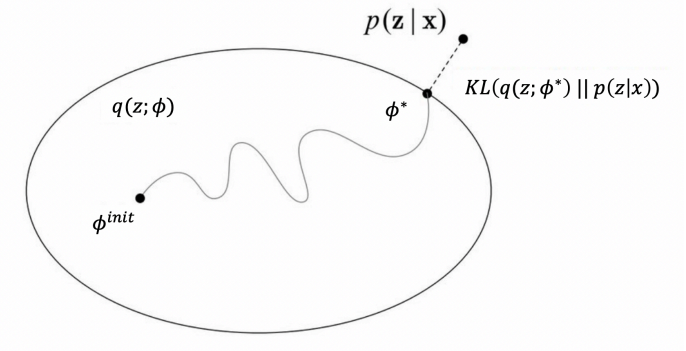

Fig. 4. The main idea of variational inference.

Its main idea is **to turn inference into optimization problem** (rather than sampling). The summarized process of Figure 4[5] is below:

1. Posit Q, a family of approximate densities over z. Each $q(z) \in Q$ indicates a candidate approximation to the exact posterior.

[5]Figure from NeurIPS 2016 Tutorial, "Variational Inference: Foundations and Modern Methods", David Blei et al.

2. Find the member of Q that minimizes D_{KL} to the exact posterior.

$$q^*(z) = argmin_{q(z)\in Q}\ D_{KL}[q(z) \parallel p(z \mid x)] \tag{5}$$

3. Approximate the posterior with q^* by using it in place of $p(z \mid x)$ in downstream modeling.

3.8 Comparison

MCMC and VI are different approaches to solving the same problem, with stochastic and deterministic approach respectively. Table 2 shows the difference.

Table 2. The difference between MCMC and VI

	Markov Chain Monte Carlo	**Variational Inference**
What it does	samples a Markov chain	solves an optimization problem
Approximates posterior with	samples from the chain	optimization results
A tool for from densities	simulating (sampling) from densities	approximating densities

Then, when to use MCMC or VI? The followings are some features of each method, so you should choose the appropriate method in your situation.

- MCMC
  - More computationally intensive than VI.
  - But it guarantees producing (asymptotically) exact samples from the target density.
  - It suits for smaller data sets or scenarios (heavier cost, more precise samples).
- VI
  - It has no such guarantees, it only finds a density close to the target.
  - But it is faster than MCMC, and also can use stochastic/distributed optimization.
  - It suits to large data sets or scenarios, where we want to quickly explore many models.

4. Approximate Inference - Variational Inference

4.1 The phrase "variational"

The term "variational" is an umbrella term that refers to mathematical tools for:

1. optimization-based formulations of problems, or
2. associated techniques for their solution.

Its origin is from calculus of variations, an optimization over a space of functions. The general idea is to express a quantity of interest as the solution of an **optimization** problem. Then the optimization problem can be "relaxed" either by approximating:

1. the function to be optimized, or
2. the set over which the optimization takes place.

Such relaxation allows us to approximate the original quantity of interest.

4.2 Variational inference (Variational Bayes)

In variational inference, the "inference" refers to solve the following optimization.

$$q^*(z) = argmin_{q(z)\in Q} \; D_{KL}[q(z) \parallel p(z \mid x)] \tag{6}$$

The complexity of Q determines the complexity of above optimization. Thus choosing Q needs a good balance:

1. We need to restrict Q sufficiently **for efficiency**, because it comprises only tractable distributions.
2. We need to allow Q to be sufficiently rich/flexible **for accuracy**, because it can provide a good approximation to the true posterior.

4.3 Optimization challenge

$$D_{KL} \; [q(z) \parallel p(z \mid x)] = \mathbf{E}_{z\sim q(x)} \; [\log q(z)] - \mathbf{E}_{z\sim q(z)} \; [\log p(z \mid x)] \tag{7}$$

$$= \mathbf{E}_{z\sim q(x)} \; [\log q(z)] - \mathbf{E}_{z\sim q(z)} \; [\log p(z,x)] + \log p(x) \tag{8}$$

The objective of variational inference – Equation 6 – is not computable. If we transform the objective according to Bayes rule, we can draw Equation 8. In the end of Equation 8, the term $\log p(x)$ is intractable because it refers to the true distribution. Variational inference solves this problem by optimizing an alternative objective called **evidence lower bound (ELBO)**, also known as "negative variational free energy".

Then, how to optimize? We rearrange Equation 8, then we can get the following equation.

$$\log p(x) - D_{KL}\ [q(z) \parallel p(z \mid x)] = \mathbf{E}_z\ [\log p(z,x)] - \mathbf{E}_z[\log q(z)] \tag{9}$$

We call the right part of Equation 9 as ELBO or $\mathbf{L}_q$, where **maximizing ELBO is equal to minizing the D_{KL} term** because $\log p(x)$ is a constant term w.r.t. $q(z)$.

Examining ELBO gives intuitions about optimial variational density. Let's transform the ELBO term further.

$$\mathbf{L}_q = \mathbf{E}_z\ [\log p(z,x)] - \mathbf{E}_z[\log q(z)] \tag{10}$$

$$= \mathbf{E}_z\ [\log p(z)] + \mathbf{E}_z\ [\log p(x \mid z)] - \mathbf{E}_z[\log q(z)] \tag{11}$$

$$= \mathbf{E}_z\ [\log p(x \mid z)] - D_{KL}\ [q(z) \parallel p(z)] \tag{12}$$

Equation 12 can be divided with **the expected log likelihood of the data** (first term), and **negative KL divergence between variational density $q(z)$ and prior $p(z)$** (second term).

By the way, which values of z in this objective will encourages $q(z)$ to place its mass on? It can be summarized as follows:

- The first term, $\mathbf{E}_z[\log p(x \mid z]$, encourages densities that explain observed data.
- The second term, $-D_{KL}\ [q(z) \parallel p(z)]$, encourages densities close to prior.

To summarize, the variational objective mirrors usual balance between likelihood and prior.

4.4 Why called ELBO?

ELBO lower bounds the (log) evidence $p(x)$. It means a following inequation satisfies:

$$\text{for any } q(z),\ \log p(x) \geq \mathbf{L}(q) \tag{13}$$

It can be proved easily using the fact that KL divergence term in Equation 9 has always positive value. It can be also proved using Jensen's inequality (not handle in this note).

4.5 Common restrictions on Q

Common restrictions we apply to Q are parametrization and factorization.

4.5.1 Parametrization

We use a parametric distribution governed by parameter ϕ. It can be represented as follows:

$$q(z;\phi) = q_\phi(z) \tag{14}$$

The parameter ϕ is called "variational parameter", so it turns the problem of finding the optimal q into **finding the optimal** ϕ. Then ELBO now becomes a function of ϕ : $\mathbf{L}(q,\phi)$, or just $\mathbf{L}(\phi)$. Also in the same way, the "inference" becomes finding the best variational parameter – ϕ^*. Telling in advance, variational autoencoder uses neural nets to model $q_\phi(z)$ and stochastic gradient descent (SGD) to train it.

4.5.2 Factorization

In this view we assume q factorizes (it is called "mean field approach"). It is formulated as follows:

$$q(z) = \prod_{j=1}^{m} q_j(z_j) \tag{15}$$

For example, if z consists of sets of binary variable like $z = (z_1, z_2, \ldots, z_m)$, the density is represented as 2^m table. However we can express it with just $2m$ if we factorize z.

So when assuming factorization, we can say that each latent variable z_j is governed by its own variational factor, the density $q_j(z_j)$. It can take any form appriate to the corresponding random variable; e.g. a continuous variable can take a Gaussian factor and a categorical vector can take a categorical vector.

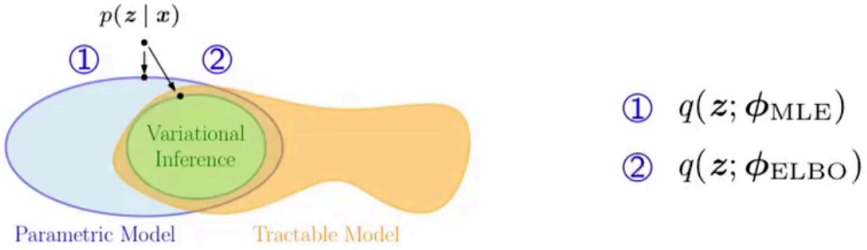

Fig. 5. The position of variational inference.

To summarize, the core idea behind variational inference is to maximize ELBO over a restricted family of distributions q. Figure 5 shows where a variational inference located in model space. We simultaneously handle the parametric and tractable approach to approximate the true posterior.

4.6 Variational inference recipe

Until now we explored what is variational inference, its objective, and optimization approaches using ELBO. Then we can make a handy recipe for using it.

1. Start with a model: $p(x,z)$.
2. Choose a variational approximation: $q(z;\phi) = q_\phi(z)$
3. Write down ELBO: $\mathbf{L}(\phi) = \mathbf{E}_{z\sim q(z)}\ [\log p(x,z) - \log q(z;\phi)]$
4. Compute the expectation/integral: e.g. $\mathbf{L}(\phi) = x\phi^2 + \log\phi$
5. Take derivative: e.g. $\nabla\mathbf{L}(\phi) = 2\phi x + \frac{1}{\phi}$
6. Optimize with gradient descent: $\phi_{t+1} = \phi_t + \varepsilon\nabla_\phi\mathbf{L}(\phi)$

4.7 Summary

The goal of generative models is to estimate density and generate sample. Models can be devided whether they clearly define density:

- explicit density: PixelRNN/CNN, variational autoencoder (VAE)
- implicit density: generative adversarial network (GAN)

In many cases, generative models are versatile and useful for many tasks.

- It can represent/manipulate high-dimensional distributions.
- It can be applied to reinforcement learning and semi-supervised learning.
- It is possible to generate multi-model outputs or infer latent representations.

An approximate inference schemes fall into two classes: stochastic or deterministic. Their strenths and weakness are complementary to each other.

Finally variational inference turns inference problem into optimization problem. Its goal is to approximate posterior $p(z \mid x)$ with $q(z) \in Q$, where z is latent, and x is observed. Q means a family of restricted (parameterized/factorized) distributions over z. And the optimal parameter $q^*(z)$ is equal to $argmin_{q(z)\in Q}\ D_{KL}\ [q(z) \parallel p(z \mid x)]$. In variational autoencoder (VAE) we parametrize q using a neural net and optimize it by backpropagation.