This study note is heavily based on this EMNLP 2018 tutorial. It covers the math behind VAEs.

Bridging Latent Varaible Models and Deep Learning

Deep Learning

- Broadly construed, deep learning is a toolbox for learning rich representations of data through numercial optimization.
- Deep learning makes it possible to parameterize conditional likelihoods with powerful function approximations.

Latent Variable Models

- LVMs make it easy to explicitly specify model constraints trough conditional independence properties.
- LVMs objectives often complicate backpropagation by introducing points of non-differentiability.

Targeted Issue and Main Focuses

How to combine the complementary strengths of the both worlds, and address the issues?
Variational inference a key technique for performing approximate posterior inference

The main focus of the tutorial is on training an inference network (deep inference) to output (*latent variable inference*) the parameters of an approximate posterior distribution given the set of variables to be conditioned upon. Also, it focuses on **learning LVMs whose joint distribution can be expressed as a directed graphical model** (DGM), which is done through variational inference.

Learning and Inference

We are interested in two things after the model is defined:

- Learning model parameter θ
- Performing *inference* after learning θ (computing the posterior distribution $p(z|x;\theta)$, or approximated, over the latent variables, given some data x)

These two tasks are intimately connecte because learning often uses inference as a subroutine. **Learning often involves alternatively inferring likely z values, and optimizing the model assuming these inferred z's**. The dominanting approach to train a latent variable model is through maximizing likelihood.

Tractable case

Assuming tractable log marginal likelihood, i.e.

$$\log p(x;\theta) = \log \sum_z p(x,z;\theta) = \log \sum_z p(z|x;\theta)p(x;\theta)$$

is tractable to evaluate (which is equivalent to assuming posterior inference to be tractable).

- Categorical LVMs where K is not too big
- HMMs wehre dynamic programs allow us to efficiently sum over all the z assignmens

Using maximum likelihood training, learning θ then corresponds to solving the following problem:

$$argmax_\theta \sum_{n=1}^{N} \log p(x^{(n)};\theta)$$

where we have assumed N examples in our training set.

Directly Optimizing with Gradient Descent

$$L(\theta) = \log p(x^{(1:N)};\theta) = \sum_{n=1}^{N} \log p(x^{(n)};\theta) = \sum_{n=1}^{N} \log \sum_z p(x^{(n)}, z;\theta)$$

where the gradient is

$$\nabla_\theta L(\theta) = \sum_{n=1}^{N} \mathbb{E}_{p(z|x^{(n)};\theta)}[\nabla_\theta p(x^{(n)}, z;\theta)]$$

which is the same form of the M-step in EM algorithm (to be confirmed). Note that the gradient expression involves an expectation over the posterior $p(z|x^{(n)};\theta)$, and *is an example of how inference is used as a subroutine in learning*. By gradient descent

$$\theta^{(i+1)} = \theta^{(i)} + \eta\nabla_\theta L(\theta^{(i)}).$$

Expectation Maximization (EM) Algorithm

EM is an iterative method for learning LVMs with tractable posterior inference. It maximizes a lower bound on the log marginal likelihood at each iteration. Given randomly-initialized starting parameters $\theta^{(0)}$, the algorithm updates the parameters via the followign alternating procedure:

1. E-step: Derive the posterior under current paramters $\theta^{(i)}$, i.e.,

   $$p(z|x^{(n)};\theta^{(i)}) = \frac{p(x^{(n)}, z;\theta^{(i)})}{p(x^{(n)})}$$

   for all $n = 1, \dots, N$.
2. M-step: Define the *expected complete data likelihood* as

   $$Q(\theta, \theta^{(i)}) = \sum_{n=1}^{N} \mathbb{E}_{p(z|x^{(n)};\theta^{(i)})}[\log p(x^{(n)}, z;\theta)]$$

Maximize this w.r.t. θ while holding $\theta^{(i)}$ fixed

$$\theta^{(i+1)} = argmax_\theta Q(\theta, \theta^{(i)})$$

It can be shown that EM improves the log marginal likelihood at each iteration, i.e.

$$\sum_{n=1}^{N} \log p(x^{(n)}; \theta^{(i+1)}) \geq \sum_{n=1}^{N} \log p(x^{(n)}; \theta^{(i)})$$

In some cases, there is an exact solution to M-step (e.g., GMMs); otherwise, one can use gradient descent and the expression is the same as directly optimizing the log likelihood. This variant of EM (no exact M-step solution) is referred to as *generalized EM*.

Intractable case

What if calculation of posterior inference or log marginal likelihood is intractable? Variational inference is a technique for approximating an intractable posterior distribution $p(z|x;\theta)$ with a tractable surrogate. In the context of learning the parameters of LVMs, VI can be used in optimizing a lower bound on the log marginal likelihood that involves only an approximate posterior over latent variables, rather than the exact posteriors.

$$\begin{aligned}
\log p(x;\theta) &= \int q(z;\lambda) \log p(x;\theta) dz \\
&= \int q(z;\lambda) \log \frac{p(x,z;\theta)}{p(z|x;\theta)} dz \\
&= \int q(z;\lambda) \log \frac{p(x,z;\theta) q(z;\lambda)}{q(z;\lambda) p(z|x;\lambda)} \\
&= \int q(z;\lambda) \log \frac{p(x,z;\theta)}{q(z;\lambda)} dz + \int q(z;\lambda) \log \frac{q(z;\lambda)}{p(z|x;\lambda)} dz \\
&= \mathbb{E}_{q(z;\lambda)} \log \frac{p(x,z;\theta)}{q(z;\lambda)} + \mathrm{KL}[q(z;\lambda) || p(z|x;\lambda)] \\
&= \mathrm{ELBO}(\theta, \lambda; x) + \mathrm{KL}[q(z;\lambda) || p(z|x;\lambda)] \\
&\geq \mathrm{ELBO}(\theta, \lambda; x)
\end{aligned}$$

Given N data, the ELBO over the entire dataset is given by the sum of individual ELBOs ($x^{(n)}$ are assumed to be drawn i.i.d),

$$\mathrm{ELBO}(\theta, \lambda; x^{(1:N)}) = \sum_{n=1}^{N} \mathbb{E}_{q(z;\lambda^{(n)})} [\log \frac{p(x^{(n)}, z; \theta)}{q(z; \lambda^{(n)})}] \leq \log p(x^{(1:N)}; \theta)$$

Note that $\lambda = [\lambda^{(1)}, \ldots, \lambda^{(n)}]$ (i.e., *we have* $\lambda^{(n)}$ *for each data point* $x^{(n)}$), which will be further approximated in the context of variational autoencoders (VAEs) with amortized variational inference. It is this aggregate ELBO that we wish to maximize w.r.t. θ and λ to train our model.

Maximizing the aggregate ELBO

coordinate ascent (variational EM)

1. Variational E-step: Maximize the ELBO for *each* $x^{(n)}$ holding $\theta^{(i)}$ fixed

$$\begin{aligned}\lambda^{(n)} &= argmax_{\lambda}\text{ELBO}(\theta^{(i)}, \lambda; x^{(n)}) \\ &= argmax_{\lambda}\mathbb{E}_{\text{q}(z;\lambda)}[\log \frac{p(z|x^{(n)};\theta)p(x^{(n)};\theta)}{q(z;\lambda)}] \\ &= argmin_{\lambda}\mathbb{E}_{q(z;\lambda)}[\log \frac{q(z;\lambda)}{p(z|x^{(n)};\theta)}] \\ &= argmin_{\lambda}\text{KL}[q(z;\lambda)||p(z|x^{(n)};\theta)],\end{aligned}$$

where the third equality holds since $logp(x;\theta^{(i)})$ is a constant w.r.t. $\lambda^{(n)}$'s.

2. Variational M-step: Maximize the aggregated ELBO w.r.t. θ holding the $\lambda^{(n)}$'s fixed

$$\theta^{(i+1)} = argmax_{\theta}\sum_{n=1}^{N}\text{ELBO}(\theta, \lambda^{(n)}; x^{(n)}) = argmax_{\theta}\sum_{q(z;\lambda^{(n)})}\mathbb{E}_{q(z;\lambda)}[\log p(x^{(n)}, z; \theta)],$$

   where the second equality holds since the term $\mathbb{E}_{q(z;\lambda)}[-\log q(z;\lambda^{(n)})]$ is constant w.r.t. θ.

This is known as **variational expectation maximization**, because the E-step is replaced with variational inference which finds the best variational approximation to the true posterior. The M-step maximizes the expected complete data likelihood where the expectation is taken w.r.t. the variational posterior.

If for *each* data $x^{(n)}$ there exists $\lambda^{(n)}$ such that $q(z;\lambda^{(n)}) = p(z|x^{(n)};\theta)$, we say that *the variational family is flexible enough to include the true posterior*, and **it reduces to the classic EM algorithm**. However, we are interested in finding a flexible variational family that allows for tractable optimization since we have assumed the exact posterior inference is intractable.

gradient ascent (stochastic variational inference)

In practice, performing coordinate ascent on the entire dataset is usually too expensive. Alternatively, gradient-based optimization can be performed over mini-batches. For each $x^{(n)}$ in the mini-batch (of size B) we initialize $\lambda_0^{(n)}$ and perform gradient ascent on the ELBO w.r.t. λ for K steps,

$$\lambda_k^{(n)} = \lambda_{k-1}^{(n)} + \eta\nabla_{\lambda}\text{ELBO}(\theta, \lambda_k^{(n)}; x^{(n)}),$$

and for M-step, hold fixed the variational parameters $\lambda_K^{(1)}, \ldots, \lambda_K^{(B)}$ and update θ:

$$\theta^{(i+1)} = \theta^{(i)} + \eta\nabla_{\theta}\sum_{n=1}^{B}\mathbb{E}_{q(z|\lambda_K^{(n)})}[\log p(x^{(n)}, z; \theta^{(i)})].$$

This is called *stochastic variational inference*.

Deep inference and VAEs

So far we have looked into two different ways of performing inference (i.e., calculating posterior distributions):

1. calculating the exact posterior distribution $p(z|x;\theta)$ when it is tractable, and
2. approximating posterior distributions by $q(z;\lambda)$ by updating λ.

A third alternative is to train a neural network to predict variational parameters λ, *rather than arriving at $\lambda^{(n)}$'s by optimizing the ELBO w.r.t. to them*.

The idea is that instead of optimizing for each $x^{(n)}$ a $\lambda^{(n)}$, we make z depend on x and parameterize the variational distribution $q(z|x;\phi)$ equally across the entire dataset with ϕ. This style of inference is known as *amortized variational inference*. When both $q(z|x;\phi)$ and $p(x|z;\theta)$ are parameterized using neural networks, we arrive variational auto encoders (VAEs). The term *autoencoder* is obvious when ELBO is rearranged as follows:

$$\begin{aligned}\mathrm{ELBO}(\theta,\phi;x) &= \mathbb{E}_{q(z|x;\phi)}[\log \frac{p(x|z;\theta)p(z;\theta)}{q(z|x;\phi)}] \\ &= \mathbb{E}_{q(z|x;\phi)}[\log p(x|z;\theta)] - \mathrm{KL}[q(z|x;\phi)||p(z;\theta)].\end{aligned}$$

The inference network $q(z|x;\phi)$ and generative network $p(z|x;\theta)$ are jointly trained by maximizing the ELBO with gradient ascent:

$$\begin{aligned}\theta^{(i+1)} &= \theta^{(i)} + \eta\nabla_\theta \mathrm{ELBO}(\theta^{(i)},\phi^{(i)};x^{(n)}) \\ \phi^{(i+1)} &= \phi^{(i)} + \eta\nabla_\phi \mathrm{ELBO}(\theta^{(i)},\phi^{(i)};x^{(n)}).\end{aligned}$$

Unlike the coordinate ascent-style training, θ and ϕ are trained jointly, where

$$\begin{aligned}\nabla_\theta \mathrm{ELBO}(\theta,\phi;x) &= \mathbb{E}_{q(z|x;\phi)}[\nabla_\theta \log p(x,z;\theta)] \\ &= \mathbb{E}_{q(z|x;\phi)}[\nabla_\theta \log p(x|z;\theta)],\end{aligned}$$

can be estimated with Monte Carlo samples with one sample. However, it is not trivial for $\nabla_\phi \mathrm{ELBO}(\theta,\phi;x)$, but with $q(z|x;\phi) = \mathcal{N}(z;\mu, diag(\sigma^2))$ and reparameterization trick,

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q(z|x;\phi)}[\log p(x|z;\theta)] &= \nabla_\phi \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\log p(x|\mu+\sigma\epsilon;\theta)] \\ &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\nabla_\phi \log p(x|\mu+\sigma\epsilon;\theta)],\end{aligned}$$

this again can be estimated with a single sample.