Various Technical Details

Andrew Ott

1 Definitions

Kullback-Leibler:

$$\mathrm{KL}(p(x)||q(y)) = -\mathrm{E}_{p(x)}[log\frac{q(y)}{q(x)}] \tag{1}$$

Note: The Kullback-Leibler is non-negative.
The Variational Lower Bound, frequently called the evidence lower bound(ELBO):

$$\mathcal{L}(q) = -\mathrm{KL}(q(z|x)||p(x,z)) \tag{2}$$

A little algebra changes (2) into the following:

$$\mathcal{L}(q) = -\mathrm{KL}(q(z|x)||p(z)) + \mathrm{E}_{q(z|x)}[logp(x|z)] \tag{3}$$

This is the form used most frequently in the literature. The KL term in (3) measures the difference between the variational posterior $p(z|x)$ and the prior $p(z)$. The expectation in (3) is called the reconstruction error. While the term is itself intractible, it lends itself to easy MC sampling by sampling some $z \sim q(z|x)$ then putting it into the log. See (Kingma & Welling, 2013) section 2.3 for more details. In fact, a thorough understanding of this paper and its implementation will help you understand much of VAEs, as it is the paper that created the VAE, alongside (Rezende, et al., 2014). I have included a fly-by summary of (Kingma & Welling, 2013) in section 5 of this paper.

2 Background

Variational Lower Bound Identity:

$$logp(x) = \mathcal{L}(q) + \mathrm{KL}(q(z|x)||p(z|x)) \tag{4}$$

Proof.

$$\begin{aligned} logp(x) &= \mathrm{E}_{q(z|x)}[logp(x)] \\ &= \mathrm{E}_{q(z|x)}[logp(x,z) - logp(z|x)] \\ &= \mathrm{E}_{q(z|x)}[(logp(x,z) - logq(z|x)) - (logp(z|x) - logq(z|x))] \\ &= \mathrm{E}_{q(z|x)}[(logp(x,z) - logq(z|x))] - \mathrm{E}_{q(z|x)}[logp(z|x) - logq(z|x)] \\ &= \mathcal{L}(q) + \mathrm{KL}(q(z|x)||p(z|x)) \qquad \square \end{aligned}$$

The KL term in (5) is intractible in our case. Fortunately, the nonnegativity of the KL allows the following inequality:

$$logp(x) \geq \mathcal{L}(q) \tag{5}$$

By maximizing $\mathcal{L}(q)$, $logp(x)$ is also maximized, (Casella & Berger, 2002), (Bishop, 2006). Hence, we will use $\mathcal{L}(q)$ for the VAE.

3 The Variational Autoencoder

The variational autoencoder (VAE) is a deep latent variable model. VAEs describe data as being generated from a distribution parameterized by neural networks (Kingma and Welling, 2014; Rezende, et al., 2014). By using neural networks to map the data to and from the hidden variables, VAEs can capture intricate relationships between hidden variables without hand coding. This makes them useful for dimensionality reduction as well as data generation.

4 Motivation for the Variational Autoencoder

Consider dataset $X = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$. For our purposes, a dataset is a collection of graded student assessments and a datapoint is one graded student test. Each iid datapoint $x^{(i)}$ is an nx1 dimensional vector distributed as

$$x^{(i)} \sim p_\theta(x^{(i)}) \tag{6}$$

θ are the parameters of distribution $p(.)$ We also assume that $x^{(i)}$ is created by some hidden vector $z^{(i)}$ distributed as

$$z^{(i)} \sim p_\theta(z^{(i)}) \tag{7}$$

with

$$x^{(i)} \sim p_\theta(x^{(i)}|z^{(i)}) \tag{8}$$

and

$$p_\theta(x^{(i)}, z^{(i)}) = p_\theta(x^{(i)}|z^{(i)})p_\theta(z^{(i)}) \tag{9}$$

For the remainder of the paper we will be dealing with only single datapoints at a time. Thus for notational convenience I will refer to $x^{(i)}$ and $z^{(i)}$ as x and z respectively. We would like to find out

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz \tag{10}$$

$$p_\theta(z|x) = \frac{p_\theta(z|x)}{\int p_\theta(x|z)p_\theta(z)dz} \tag{11}$$

Solving (10) will allow us to maximize the probability that the data has occurred (Doersch, 2016). Solving (11) and allows us to probabalistically encode datapoint x as hidden variable z. With well chosen z, one can generate realistic data. Real life data is often very complex, having difficult to specify relationships between z and x. It is thus very desireable that (8) be very flexible in order to capture the relationships. This is accomplished by feeding z through some flavor of a neural network. As neural networks are universal approximators, they can theoretically capture all the dependencies from z to x.

Unfortunately, using a neural network to model the relationships between z and x make (10) intractable. By extension, this makes (11) intractable. This problem can be circumvented through MCMC methods, but they are usually too computationally expensive to be practical (Mnih & Gregor, 2014). Mean field variational Bayes, ala (Bishop, 2006) chapter 10, still has intractable expectations, making that approach undesirable. Another approach in dealing with the intractability of (10) is by using simple Monte-Carlo sampling methods. This is expressed by

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz = \mathrm{E}_{p_\theta(z)}[p_\theta(x|z)] \simeq \frac{1}{B}\sum_{b=1}^{B} p_\theta(x|z^{(b)}) \tag{12}$$

where $z^{(b)} \sim p_\theta(z^b)$.
As B becomes bigger, (12) becomes a better approximation. Unfortunately, if z has high dimensionality, B may need to be infeasibly large. If better samples of z can be drawn(i.e. through $p_\theta(z|x))$) then B would not need to be so large. Yet $p_\theta(z|x))$ is intractable, so this is not possible (Doersch, 2016).

5 Auto-Encoding Variational Bayes by Kingma and Welling, 2013

5.1 Lower Bounds

In order to deal with the those problems, Kingma and Welling introduce an approximation to the posterior:

$$z \sim q_\phi(z|x) \tag{13}$$

Using standard variational techniques we arrive at a differentiable lower bound

$$log p_\theta(x) = KL(q_\phi(z|x)||p_\theta(z|x)) + \mathcal{L}(\theta, \phi; x) \tag{14}$$

where

$$KL(q_\phi(z|x)||p_\theta(z|x)) = -\mathrm{E}_{q_\phi(z|x)}[log\frac{p_\theta(z|x)}{q_\phi(z|x)}] \tag{15}$$

and

$$\mathcal{L}(\theta, \phi; x) = -KL(q_\phi(z|x)||p_\theta(z)) + \mathrm{E}_{q_\phi(z|x)}[log p_\theta(x|z)] \tag{16}$$

Noting that (15) is always positive, we see that

$$logp_\theta(x) \geq \mathcal{L}(\theta, \phi; x) \tag{17}$$

We call (16) the variational lower bound, as it uses variational inference to provide a (hopefully) reasonable lower bound on $logp_\theta(x)$. Interestingly,(15) disappears when $q_\phi(z|x) = p_\theta(z|x)$. Thus if $q_\phi(z|x)$ is very flexible, then (15) will go to 0, making (17) an equality. The authors make $q_\phi(z|x)$ flexible by making its parameters dependant on neural networks.

5.2 The Model

This is is achieved as the Kingma and Welling set up their model as follows:

$$z \sim q_\phi(z|x) = MVN(z|\mu_2, \sigma_2^2 I) \tag{18}$$

$$\mu_2 = W_4 h_2 + b_4 \tag{19}$$

$$log\sigma_2^2 = W_5 h_2 + b_5 \tag{20}$$

$$h_2 = \psi_1(W_6 x + b_6) \tag{21}$$

$\sigma_2^2 I$ is a diagonal matrix with diagonals as the vector σ_2^2. In (21) the ψ_1 is a nonlinearity and for both equations (20) and (21) the log and ψ_1 are applied elementwise. We can put (18), (19), (20), and (21) into words as the following: datapoint x is fed into a neural network with nonlinearity ψ to turn it into vector h_2. After this h_2 is fed into (19) and (20) simultaneously to get the parameters for the multivariate normal in (18). Those parameters are then used in (18) to randomly generate hidden variables z. $q_\phi(z|x)$ is known variously as the probabalistic encoder, the approximate posterior, or the recognition model.

It is important to note that z does not have to be generated via the MVN. Any other continuous distribution will function, so long as the parameters are functions of a neural network.

After the hidden vector z has been generated, the network then reconstructs the original datapoint x as follows:

$$x \sim p_\theta(x|z) = MVN(x|\mu_1, \sigma_1^2 I) \tag{22}$$

$$\mu_2 = W1 h_1 + b_1 \tag{23}$$

$$log\sigma_1^2 = W_2 h_1 + b_2 \tag{24}$$

$$h_1 = \psi_2(W_3 z + b_3) \tag{25}$$

Just as in the previous equations, in (25) the ψ_2 is a nonlinearity and for both equations (24) and (25) the log and ψ_2 are applied elementwise. (22) - (25) perform the following operations: z is fed into a neural network with nonlinearity ψ. After this, h_1 is fed into (24) and (23) to create the parameters for

(22). The MVN in (22) then is generates the sample x. While (22) uses the MVN, any reasonable distribution, continuous or discrete can be used to generate x. Whichever distribution the researcher thinks most plausible for creating the data can work for (22) provided that its parameters are generated through transformations of h_1. $p_\theta(x|z)$ is known as the probabalistic decoder.

Looking at equations (18) - (25) together, we see that datapoint x is transformed by a neural network in order to create the parameters for a distribution which generates z. Once z is randomly generated, this is fed into another neural network to create the parameters for another distribution which recreates the original datapoint x.

If the researcher wants to randomly generate a sample datapoint, the researcher only needs to take a random sample of z and run it through (22) - (25). This will create a realistic sample datapoint if z, ϕ, and θ have been trained.

5.3 Training the Model

With the model thus defined, it is left to train the model. From the discussion in the first section (16) provides a reasonable lower bound to $logp_\theta(x)$ Therefore we set (16) as the cost function to be minimized in this model. Kingma and Welling note that this can be solved using the following method:

$$\nabla_\phi \mathrm{E}_{q(z|x)}[f(z)] = \nabla_\phi \int q_\phi(z|x) f(z) dz = \int \nabla_\phi q_\phi(z|x) f(z) dz = \int f(z) \nabla_\phi q_\phi(z|x) dz \tag{26}$$

$$= \int f(z) q_\phi(z|x) \nabla_\phi log q_\phi(z|x) dz = \mathrm{E}_{q_\phi(z|x)}[f(z) \nabla_\phi log q_\phi(z|x)] \tag{27}$$

$$\simeq \frac{1}{M} \sum_{l=1}^{M} f(z) \nabla_\phi log q_\phi(z^{(m)}|x) \tag{28}$$

where $z^{(m)} \simeq p_\theta(z^{(m)})$ Unfortunately this gradient estimator yields too high varience. (Blei, et al., 2012) give a proof for why this is so. (Mnih & Gregor, 2014) give experimental results of training with (28) showing it to be useless. With this in mind Kingma and Welling set out to create a gradient with less variance. They do so by reparameterizing z.

$$z = g_\phi(\epsilon, x) \tag{29}$$

with $\epsilon \sim p(\epsilon)$ where

$$g_\phi(e, x) = q_\phi(z|x) \tag{30}$$

Here $g_\phi(.)$ is a deterministic, differentiable mapping from the noise variable ϵ to z parameterized by ϕ. To give a specific example, in (18) z was parameterized by $MVN(z|\mu_2, \sigma_2^2)$. If $p(\epsilon) \sim MVN(\epsilon|0, I)$ then clearly

$$z = \mu_2 + (\sigma_2^2 I)\epsilon \tag{31}$$

$\sigma_2^2 I$ is a diagonal matrix with the diagonals being the elements of σ_2^2. Here $g_\phi(.)$ is thus parameterized as (31). This new definition of z enables for different gradient estimators to be taken of $\mathcal{L}(\theta, \phi; x)$.

$$\nabla_\phi \mathrm{E}_{q(z|x)}[f(z)] = \nabla_\phi \mathrm{E}_{p(\epsilon)}[f(g_\phi(x,\epsilon))] = \mathrm{E}_{p(\epsilon)}[\nabla_\phi f(g_\phi(x,\epsilon))] \simeq \frac{1}{M}\sum_{m=1}^{M} \nabla_\phi f(g_\phi(x,\epsilon^{(m)})) \tag{32}$$

where $\epsilon^{(m)} \sim p(\epsilon^{(m)})$. Thus by sampling from noise parameter ϵ one can attain a reasonable Monte-Carlo estimation of the gradient. By applying (32) to (16) we can attain the following estimators of $\mathcal{L}$

$$\tilde{\mathcal{L}}^A \simeq \frac{1}{M}\sum_{m=1}^{M}[logp_\theta(x, z^{(m)}) - logq_\phi(z^{(m)}|x)] \tag{33}$$

$$\tilde{\mathcal{L}}^B \simeq -KL(q_\phi(z|x)||p_\theta(z)) + \frac{1}{M}\sum_{m=1}^{M}[logp_\theta(x|z^{(m)})] \tag{34}$$

If both $q_\phi(z|x)$ and $p_\theta(z)$ are Normal, then $-KL(q_\phi(z|x)||p_\theta(z))$ can be solved analytically. Incidentally, it is (34) where the name variational autoencoders come from. In the parlance of autoencoders, the $-KL$ is a regularizing term whereas $\frac{1}{M}\sum_{m=1}^{M}[logp_\theta(x|z^{(m)})]$ is the negative reconstruction error.

Future Directions in the Simulation Study

(Converse, et al., 2019) used variational autoencoders in order to extract latent student knowledge from student test scores. The study showed that the VAE was able to extract student knowledge scores that highly correlated with ground truth student knowledge. The paper had several assumptions about the student knowledge that are problematic: 1) the student latent knowledge was independent and 2) the student latent knowledge did not have causal relationships.

The first is a problem because many student skills are correlated. For example, knowledge about reading correlates with knowledge about writing. This can be addressed by having the latent variables by MVNs with correlation matrices. I have begun working on this. Sadly, there is still a bit of work to be done here. It is very challenging to code nonindependent random variables for VAEs. The theoretical work, thankfully, has been solved. (Ruiz, et al., 2016) give the necessary equation with equation 5 on page 3.

The second assumption poses a problem because some knowledge is built 'ontop of' other knowledge. For instance, a difficult probability problem often entails many other knowledges than just probability. You may need Calculus or Algebra to solve the probability problem. The knowledge of probability is built ontop of other knowledges. While I have not done the testing in this, I have derived the equations for VAEs designed to test this. The derivations are loosely inspired from (Edwards and Storkey, 2017).

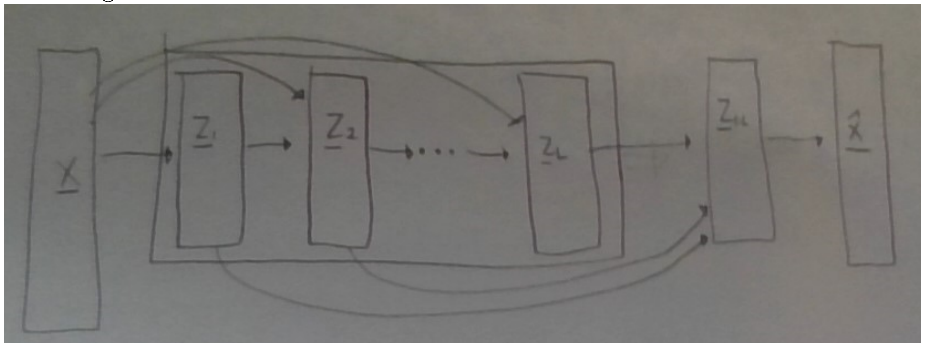
VAEs with Multiple Layer of Stochastic Variables

Suppose that the data, x, is generated by a ladder of stochastic hidden layers in the spirit of (Storkey and Edwards, 2017). More specifically, $z_{1:L}$ is a series of stochastic layers whose variational encoder is as follows:

$$q(z_{1:L}|x) = q(z_1|x)\prod_{i=2}^{L} q(z_i|z_{i-1}, x) \tag{35}$$

where each $q(z_i|z_{i-1}, x)$ is a probability distribution parameterized by neural networks. Due to the constraint of interpretibility, the network will not be able to use multiple layers of representation for the decoder.

Figure 1: The Architecture of the VAE to be used in the research.



As seen above, the input student data, x, is fed into each stochastic layer, z_i. The stochastic layer z_i has as input x and z_{i-1}, the previous layer. All the random variables are then taken as is to form $z_{1:L}$. Note that nothing new happens to form $z_{1:L}$: the random variables generated from z_1 to z_L are each collected as is to form this layer. No new layers or random processes happen. From here, $z_{1:L}$ is used with the Q-matrix to generate the reconstruction of the input, $\hat{x}$. The question becomes, how does one derive $\mathcal{L}(q)$ for this new network? To derive this, we begin a new variational lower bound identity.

Proof.

$$\begin{aligned}
logp(x) &= \mathrm{E}_{q(z_{1:L}|x)}[logp(x)] \\
&= \mathrm{E}_{q(z_{1:L}|x)}[logp(x, z_{1:L}) - logp(z_{1:L}|x)] \\
&= \mathrm{E}_{q(z_{1:L}|x)}[(logp(x, z_{1:L}) - logq(z_{1:L}|x)) - (logp(z_{1:L}|x) - logq(z_{1:L}|x))] \\
&= \mathrm{E}_{q(z_{1:L}|x)}[(logp(x, z_{1:L}) - logq(z_{1:L}|x))] - \mathrm{E}_{q(z_{1:L}|x)}[logp(z_{1:L}|x) - logq(z_{1:L}|x)] \\
&= \mathcal{L}(q) + \mathrm{KL}(q(z_{1:L}|x)||p(z_{1:L}|x))
\end{aligned}$$

□

where

$$\mathcal{L}(q) = \mathrm{E}_{q(z_{1:L}|x)}[(logp(x, z_{1:L}) - logq(z_{1:L}|x))] \tag{36}$$

As in the previous derivation, the KL term is intractible, so we ignore it and maximize $\mathcal{L}(q)$. Using equation (5) we can simplify $\mathcal{L}(q)$.

Proof.

$$\begin{aligned}
\mathcal{L}(q) &= \mathrm{E}_{q(z_{1:L}|x)}[logp(x, z_{1:L}) - logq(z_{1:L}|x)] \\
&= -\mathrm{E}_{q(z_{1:L}|x)}[logq(z_{1:L}|x)] + \mathrm{E}_{q(z_{1:L}|x)}[logp(x, z_{1:L})] \\
&= -\mathrm{E}_{q(z_{1:L}|x)}[log[q(z_1|x)\prod_{i=2}^{L}(q(z_i|z_{i-1}, x)]] \\
&+ \mathrm{E}_{q(z_{1:L}|x)}[logp(x|z_{1:L})] + \mathrm{E}_{q(z_{1:L}|x)}[logp(z_{1:L})] \\
&= \sum_{i=1}^{i=L} H[p(z_i|z_{i-1}, x)] + \mathrm{E}_{q(z_{1:L}|x)}[p(x|z_{1:L})] + \mathrm{E}_{q(z_{1:L}|x)}[p(z_{1:L})]
\end{aligned}$$

□

where

$$H[p(x)] = -\mathrm{E}_{p(x)}[p(x)] \tag{37}$$

The final line of the proof is the term to be maximized by the VAE. While the math has been done, programming this out will take some effort.

References

[1] Dieng, A., Kim, Y., Rush, A., Blei, D. *Avoiding latent variable collapse with generative skip models.* In ICML, 2018.

[2] Edwards, H. and Storkey, A. *Towards a neural statistician.* International Conference on Learning Representations (ICLR), 2017.

[3] Kingma, D. and Welling, M. *Auto-encoding variational Bayes.* In ICML, 2014.

[4] Kingma, D., Rezende, D., Mohamed, S., and Welling, M. *Semi-supervised Learning with Deep Generative Models.* In Neural Information Processing Systems, 2014.

[5] Mnih, Andriy and Karol Gregor. *Neural Variational Inference and Learning in Belief Networksl* In AISTATS, 2014.

[6] Rezende, D. J., Mohamed, S., and Wierstra, D. *Stochastic backpropagation and approximate inference in deep generative models* In ICML, 2014.

[7] Ruiz, F. J., Titsias, M., and Blei, D. *The generalized reparameterization gradient.* arXiv preprint arXiv:1610.02287, 2016.