# Combining GAN and VAE

Jee Dong Jun

05.12.2020

# Latent Variable Model

- Consider dataset $X$ consisting of N iid samples of variable x. Assume that it is generated by unobserved latent variable z.
- Suppose z is generated by prior distribution $p_\theta(z)$
- Value x is generated by conditional distribution $p(x|z)$

# ELBO

- We use evidence lower bound(ELBO) L instead of actual likelihood.
- And we also use $q(z|x)$ to approximate posterior distribution
- $L(x, \theta, q) = \log p(x; \theta) - D_{KL}(q(z|x)||p(z|x; \theta)))$
- Note, KL divergence term is non-negative so ELBO is lower bound of actual likelihood
- Two are equal if our approximation matches true distribution.

# Evidence Lower Bound ELBO

$$\begin{aligned}
L(x,\theta,q) &= \log p(x;\theta) - D_{KL}(q(z|x)||p(z|x;\theta))) \\
&= \log p(x;\theta) - E_{z\sim q}(\log q(z|x) - \log p(z|x)) \\
&= \log p(x;\theta) - E_{z\sim q}(\log q(z|x) - \log p(z,x) + \log p(x)) \\
&= -E_{z\sim q}(\log q(z|x) - \log p(z,x)) \\
&= -D_{KL}(q(z|x)||p(z;\theta)) + E_{q}(\log p(x|z))
\end{aligned}$$

# Optimizing ELBO

- Suppose our choice of q has parameters $\phi$ and $p(x|z)$ has parameters $\theta$
- We want to maximize $L(\theta, \phi, x) = -D_{KL}(q_\phi(z|x)||p(z;\theta)) + E_{q_\phi}(\log p_\theta(x|z))$ respect to $\theta, \phi$
- For many cases, it is theoretically possible to calculate KL divergence term.
- Therefore, main issue is with second part $E_{q_\phi}(\log p)$ terms.

# The actual Algorithm

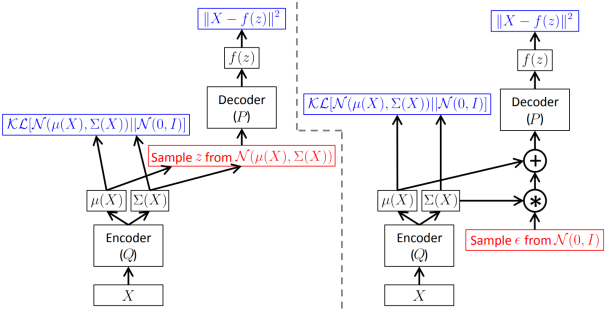

$\|X - f(z)\|^2$
$f(z)$
Decoder
($P$)
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))||\mathcal{N}(0,I)]$
Sample $z$ from $\mathcal{N}(\mu(X),\Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder
($Q$)
$X$
$\|X - f(z)\|^2$
$f(z)$
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))||\mathcal{N}(0,I)]$
Decoder
($P$)
$+$
$*$
$\mu(X)$
$\Sigma(X)$
Encoder
($Q$)
Sample $\epsilon$ from $\mathcal{N}(0,I)$
$X$

Figure 1: Summary

# Generative Adversarial Network

- There are two different networks.
- One is called Discriminator D and the other is called Generator G
- These two networks work against each other for training.

# Cost Function of GAN

- The cost function of GAN can be expressed as

$$E_{x \sim p_{data}} \log D(x) + E_{z \sim p_{prior}} \log(1 - D(G(z))$$

- However, Discriminator and Generator are acting against each other.
- We are trying to find D that maximize the cost function while G tries to minimize the cost function.

# GAN

- Generator G tries to make the fake sample from the random noise.
- Discriminator D tries to differentiate the fake sample generated from the generator G and the real sample from the data.
- As both of them becomes more and more accurate, Generator G will learn the $p_{data}$ distribution.
- Note G defines a distribution.
- GAN is a method to train the generator distribution to be the $p_{data}$ distribution by comparing the sample from the distribution.

# GAN And VAE

- GAN is difficult to train while VAE is comparatively easier to train.
- VAE tends to produce blurry image compared to GAN.
- VAE naturally yields encoder network that gives the representation of data.
- VAE is carrying out Maximum Likelihood Estimation. (Approximately)

# Latent Variable Model Again

- Let z be the latent variable with prior $p(z)$ and conditional distribution $p(x|z)$
- $q(z|x)$ be encoding distribution and $p'(x|z)$ be decoding distribution
- Note, here we are not assuming encoding distribution is approximating posterior distribution like VAE.

# Structure of Adversarial Autoencoders

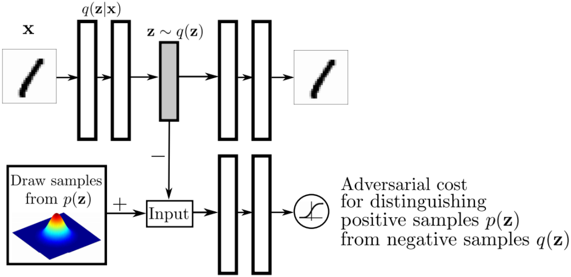

$\mathbf{x}$
$q(\mathbf{z}|\mathbf{x})$
$\mathbf{z} \sim q(\mathbf{z})$
$-$
Draw samples from $p(\mathbf{z})$
$+$
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

Figure 2: Structure of AAE from Adversarial Autoencoders

# Aggregated Posterior

- We want to train $p'(x|z)$ such that it is like conditional distribution $p(x|z)$ and aggregated posterior q(z) to be like true prior p(z)
- We are not performing posterior inference.
- $\int_X q(z|x)p_d(x)dx = q(z)$ is trained to be p(z) true prior.
- $p'(x|z)$ is trained to be true conditional distribution.

# Several Possible Choices for Encoder

- We can use deterministic q. Only source of stochasticity is from the sample.
- We can also use encoder structure of VAE. $N(\mu(x), \sigma(x))$ Similarly implementing reparametrization trick for evaluating the gradient.

# Universal Approximator Posterior

- When we use $N(\mu(x), \sigma(x))$, distribution is very flexible respect to x but it is still just normal distribution when x is given.
- We start from random noise $\eta$ from a fixed distribution.
- Consider $f(x, \eta)$ where f is defined by neural network as the encoder.
- Further discussed with Adversarial Variational Bayes

# Another way to view Variational Autoencoder

- Cost of Variatioanl Autoencoder is $L(\theta, \phi, x) = -D_{KL}(q_\phi(z|x)||p(z;\theta)) + E_{q_\phi}(\log p_\theta(x|z))$
- Note this is for single data x. For mini batch algorithm, after simplification, we ae minimizing

$$-E_x E_{q(z|x)}(\log p_\theta(x|z)) - Entropy + CrossEntropy(q(z), p(z))$$

- Therefore, we can think of VAE as minimizing distance between q(z),p(z) while second term ensure larger entropy of q(z|x).

# AAE compared to VAE

- Once we view VAE as autoencoder with regularization, AAE is also autoencoder with regularization that replaced with different regularization.
- However, note AAE did not replace KL divergence regularization with some random other regularization.
- KL terms in VAE can be decomposed as Entropy and CrossEntropy between aggregated posterior and prior.
- Instead of using cross entropy to train aggregated posterior q(z) to be like p(z), in AAE adversarial cost is used to train aggregated posterior q(z) to be like p(z)

# Comparison of VAE and AEE

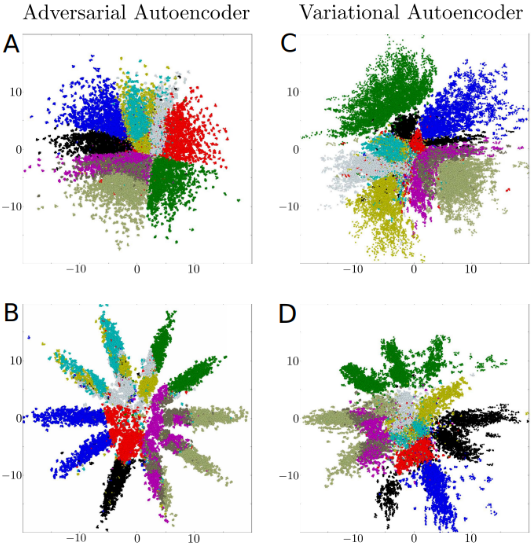


Figure 3: Comparing AAE and VAE on MNIST from Adversarial Autoencoders

# For clustering and semi-supervised learning

- Instead of considering one latent variable (continuous), one can introduce one more latent variable for label.

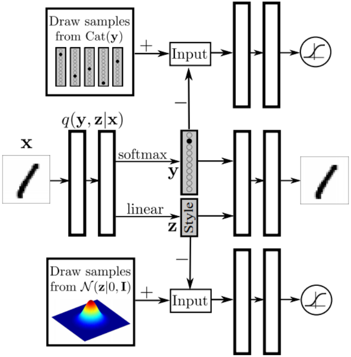

Draw samples from Cat($\mathbf{y}$)
+
Input
−
$q(\mathbf{y},\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
softmax
$\mathbf{y}$
linear
$\mathbf{z}$
Style
−
Draw samples from $\mathcal{N}(\mathbf{z}|0,\mathbf{I})$
+
Input

Figure 4: New Structure

# AAE compared to VAE

- AAE has no restriction on prior distribution. We just need to know how to sample from the prior distribution. We do not need to know the functional form of prior distribution which is required for calculating KL divergence term.
- Similarly, we also do not need to know exact functional form of q(z|x).
- But what is meaning of q(z|x)?
- We are not doing maximum likelihood estimation.
- In the end, AAE is the variant of autoencoder not the variant of VAE.

# AVB from VAE

- Adversarial Variational Bayes starts from VAE not from AAE.

$$\log p(x) \geq L(x, \theta, q) = \log p(x; \theta) - D_{KL}(q(z|x)||p(z|x; \theta)))$$

- Fundamental problem of VAE is that the family of normal distribution used for q(z|x) is not expressive enough and there will be always difference between true log likelihood and ELBO
- KL divergence term becomes 0 iff $q(z|x) = p(z|x)$. Therefore, the more general family we choose for the encoder q(z|x), the performance of the VAE will increase.

# AVB from VAE

- Adversarial Variational Bayes starts from VAE not from AAE.

$$\log p(x) \geq L(x, \theta, q) = \log p(x; \theta) - D_{KL}(q(z|x)||p(z|x; \theta)))$$

- Fundamental problem of VAE is that the family of normal distribution used for q(z|x) is not expressive enough and there will be always difference between true log likelihood and ELBO
- KL divergence term becomes 0 iff $q(z|x) = p(z|x)$. Therefore, the more general family we choose for the encoder q(z|x), the performance of the VAE will increase.

# Use of Neural Network for Encoder Distribution

- Assume $\epsilon$ follow prior distribution and $f_\phi(x, \epsilon)$

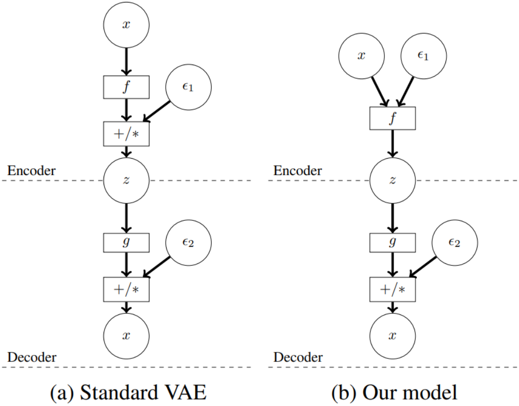


Figure 5: AVB structure from Adversarial Variational Bayes

# Problem with Using Neural Network for the Encoder

- To calculate KL divergence term, reparametrization trick is not enough.

$$\max_{\theta} \max_{\phi} E_x E_{q(z|x)}(\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x))$$

- One needs to know the exact form of $q(z|x)$
- However, if we define q(z|x) by transformation of some prior distribution through neural network, it is impossible to evaluate the exact form of $q(z|x)$

# Introduce the discriminator T(x,z)

- Avoid the problem by representing the term $\log p(z) - \log q_\phi(z|x)$ using the discriminator T.

$$\max_T E_x E_{q_\phi(z|x)} \log \sigma(T(x,z)) + E_x E_{p(z)} \log(1 - \sigma(T(x,z)))$$

- T(x,z) is trying to differentiate whether (x,z) came from $q(z|x)p(x)$ or $p(x)p(z)$
- As T is the neural network, it can approximate any function well.

# New Cost

- When q is fixed, the optimal discriminator $T*$ is given by $T^*(x,z) = \log q_\phi(z|x) - \log p(z)$

$$\max_\theta \max_\phi E_x E_{q(z|x)}(\log p_\theta(x|z) - T^*(x,z))$$

- We have to find gradient respect to $\theta$ and $\phi$
- However, note that $T^*$ indirectly depends on $\phi$ too. It is defined as optimization problem depending on $q_\phi$

# How to handle Derivative with Respect to $\phi$

### Proposition 1

$E_q(\nabla_\phi T^*(x,z)) = 0$

$$\begin{aligned} E_q(\nabla_\phi T^*(x,z)) &= E_q(\nabla_\phi \log q_\phi(z|x)) \\ &= \int q_\phi \frac{\nabla_\phi \log q_\phi(z)}{q_\phi} dz \\ &= \nabla_\phi \int q_\phi(z) dz \end{aligned}$$

# The final cost

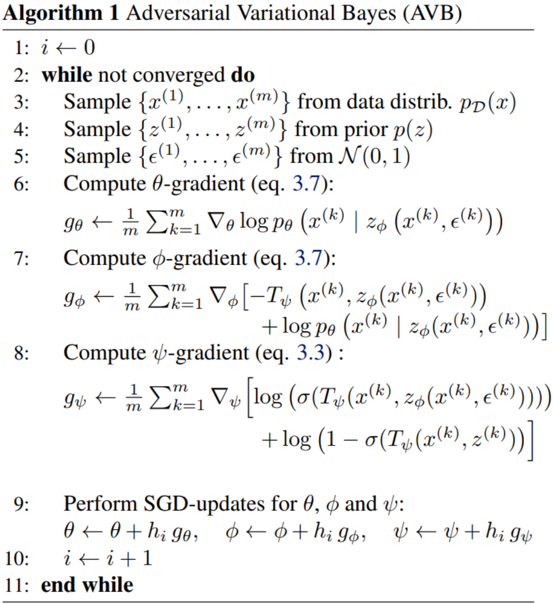
- We can use the reparametrization trick by definition of the encoder function.

$$\max_{\theta} \max_{\phi} E_x E_\epsilon (\log p_\theta(x|z_\phi(x,\epsilon)) - T^*(x, z_\phi(x,\epsilon))$$

- Also, we parametrize discriminator T with $\psi$

$$\max_{\psi} E_x E_\epsilon \log \sigma(T(x, z_\phi(x,\epsilon))) + E_x E_{p(z)} \log(1 - \sigma(T(x,z)))$$

# The final Algorithm



**Algorithm 1** Adversarial Variational Bayes (AVB)

```
$i \leftarrow 0$
while not converged do
    Sample $\{x^{(1)}, \dots, x^{(m)}\}$ from data distrib. $p_{\mathcal{D}}(x)$
    Sample $\{z^{(1)}, \dots, z^{(m)}\}$ from prior $p(z)$
    Sample $\{\epsilon^{(1)}, \dots, \epsilon^{(m)}\}$ from $\mathcal{N}(0, 1)$
    Compute $\theta$-gradient (eq. 3.7):
    $g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\theta \log p_\theta \left( x^{(k)} \mid z_\phi \left( x^{(k)}, \epsilon^{(k)} \right) \right)$
    Compute $\phi$-gradient (eq. 3.7):
    $g_\phi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\phi \big[ -T_\psi \left( x^{(k)}, z_\phi(x^{(k)}, \epsilon^{(k)}) \right)$
        $+ \log p_\theta \left( x^{(k)} \mid z_\phi(x^{(k)}, \epsilon^{(k)}) \right) \big]$
    Compute $\psi$-gradient (eq. 3.3) :
    $g_\psi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\psi \Big[ \log \left( \sigma(T_\psi(x^{(k)}, z_\phi(x^{(k)}, \epsilon^{(k)}))) \right)$
        $+ \log \left( 1 - \sigma(T_\psi(x^{(k)}, z^{(k)})) \right) \Big]$

    Perform SGD-updates for $\theta$, $\phi$ and $\psi$:
    $\theta \leftarrow \theta + h_i \, g_\theta, \quad \phi \leftarrow \phi + h_i \, g_\phi, \quad \psi \leftarrow \psi + h_i \, g_\psi$
    $i \leftarrow i + 1$
end while
```

Figure 6: algorithm from Adversarial Variational Bayes

# Connection with AAE

- AAE can be thought as approximation of discriminator function T(x,z) with T(z) that does not depend on x.
- As mentioned above, this T ensures that $\int q(z|x)p_d(x)dx = p(z)$
- However, this does not guarantee that q(z|x) is approximately posterior.