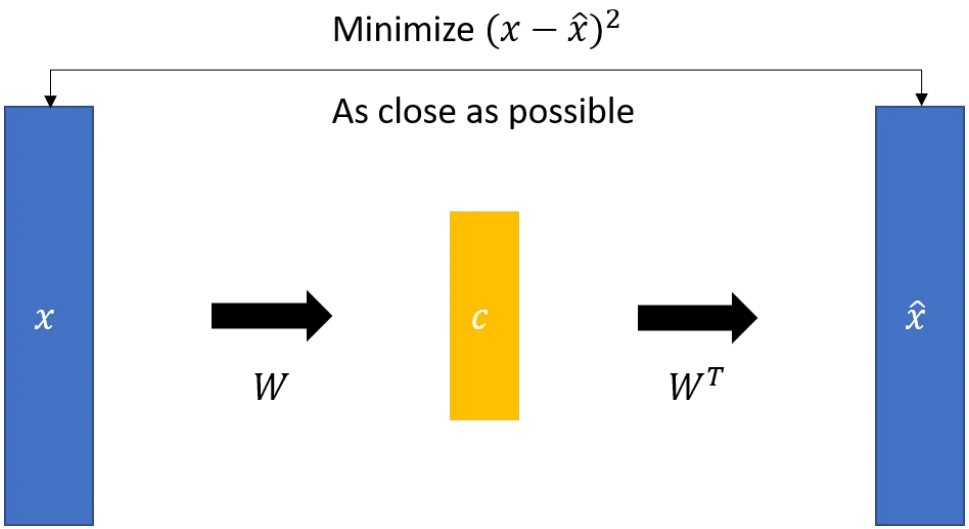
Variational Autoencoder(VAE)

Roger Yong · Follow
Published in Geek Culture
6 min read · Jul 8, 2021

As a generative model, the basic idea of VAE is easy to understand: the real sample is transformed into an ideal data distribution through the encoder network, and then this data distribution is passed to a decoder network to obtain the generated sample. If the generated samples and the real samples are close enough, an autoencoder model is trained.

Dimensionality reduction, PCA and autoencoders (AE)

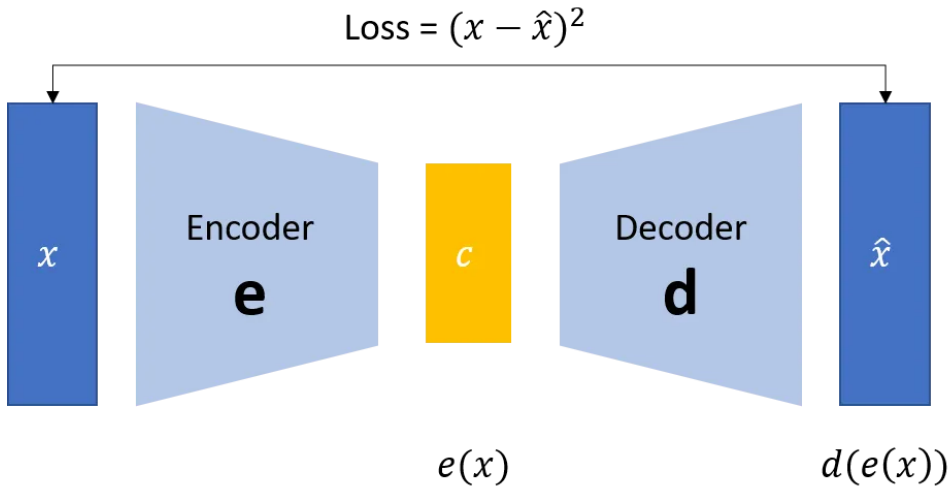
PCA



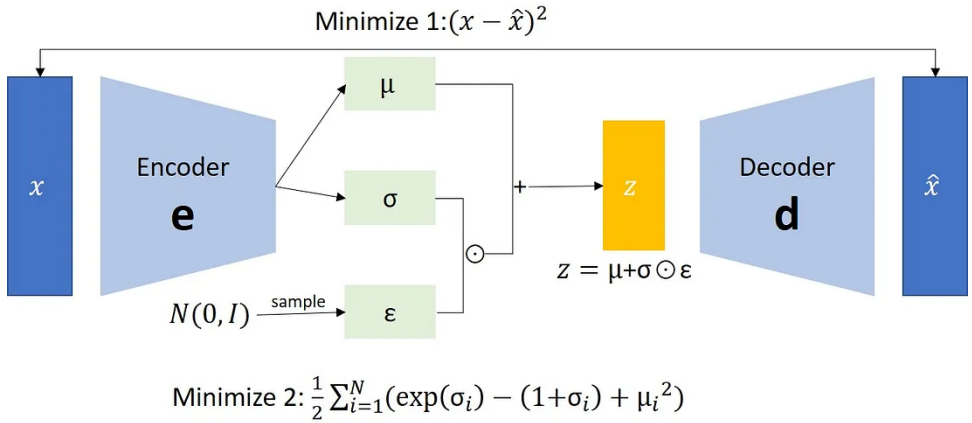
As shown in the figure, x is a matrix which can becomes a low-dimensional matrix c through a transformation W. Because this process is linear, the transpose of W can be used to restore an x hat. PCA is to find a W through SVD (singular value decomposition) so that the matrices x and x hat be as consistent as possible.

However, the difference between AE and PCA is that AE uses neural networks instead of SVD.

Autoencoder



Variational Autoencoder(VAE)



Minimize 2: $\frac{1}{2}\sum_{i=1}^{N}(\exp(\sigma_i)-(1+\sigma_i)+\mu_i{}^2)$

Encoder

This defines the approximate posterior distribution $q(z|x)$, which takes as input an observation and outputs a set of parameters for specifying the conditional distribution of the latent representation z. In this example, simply model the distribution as a diagonal Gaussian, and the network outputs the mean and log-variance parameters of a factorized Gaussian. Output log-variance instead of the variance directly for numerical stability.

Decoder

This defines the conditional distribution of the observation $q(x|z)$, which takes a latent sample as input and outputs the parameters for a conditional distribution of the observation. Model the latent distribution prior $P(z)$ as a unit Gaussian.

Reparameterization trick

To generate a sample z for the decoder during training, you can sample from the latent distribution defined by the parameters outputted by the encoder, given an input observation x. However, this

parameters and another parameter ε as follows:

$$z = \mu + \sigma \odot \varepsilon$$

where μ and σ represent the mean and standard deviation of a Gaussian distribution respectively. They can be derived from the decoder output. The ε can be thought of as a random noise used to maintain stochasticity of z. Generate ε from a standard normal distribution.
The latent variable z is now generated by a function of μ, σ and ε, which would enable the model to backpropagate gradients in the encoder through μ and σ respectively, while maintaining stochasticity through ε.

How VAE works?

The theoretical basis of VAE is the Gaussian mixture model (GMM). The difference is that our code is replaced by a continuous variable z, and z follow standard normal distribution $N(0,1)$.
For each sample z, there will be two variables μ and σ, which respectively determine the mean and standard deviation of the Gaussian distribution corresponding to z, and then the accumulation of all Gaussian distributions in the integration domain becomes the original distribution $P(x)$:

$$P(x) = \int_z P(z)P(x|z)dz$$

Where $z \sim N(0,1)$, $x|z \sim N(\mu(z), \sigma(z))$, Since $P(z)$ is known, $P(x|z)$ is unknown, and $x|z \sim N(\mu(z),\sigma(z))$. What we really need to solve is the expressions of μ and σ, but $P(x)$ is so complex that μ and σ are difficult to be calculated, we need to introduce two neural networks to help us solve it. :

we hope $P(x)$ the bigger the better, then

$$\text{Maximum } L = \sum_x \log P(x)$$

Where $\log P(x)$

$$\begin{aligned}
\log P(x) &= \int_z q(z|x)\log P(x)dz \\
&= \int_z q(z|x)\log\left(\frac{P(z,x)}{P(z|x)}\right)dz \\
&= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\frac{q(z|x)}{P(z|x)}\right)dz \\
&= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz + \int_z q(z|x)\log\left(\frac{q(z|x)}{P(z|x)}\right)dz \\
&= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz + KL\left(q(z|x)||P(z|x)\right)
\end{aligned}$$

The second term of the above formula is a value greater than or equal to 0, so we found a lower bound of $\log P(x)$

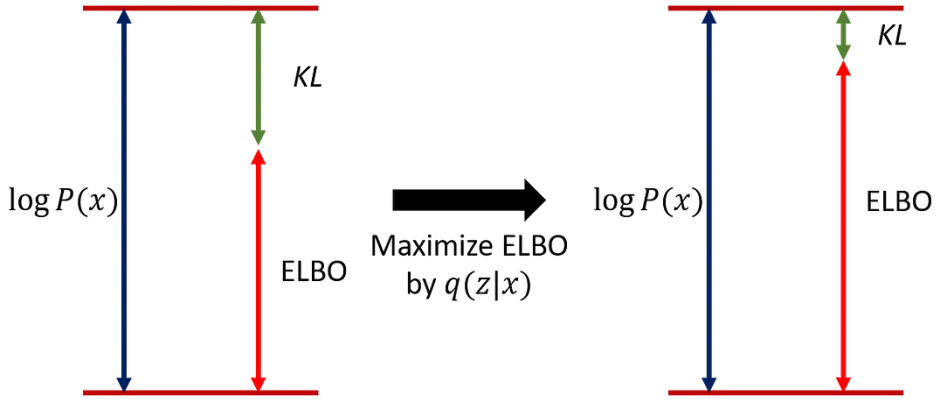
$$\log P(x) \geq \int_z q(z|x)\log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

We denote this lower bound as ELBO:

$$\log P(x) \geq \text{ELBO} = \int_z q(z|x)\log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz = \mathrm{E}_{q(z|x)}\left[\log\frac{p(x,z)}{q(z|x)}\right]$$

So we can revise the original form as:

$$\log P(x) = \text{ELBO} + KL\left(q(z|x)||P(z|x)\right)$$



Through adjusting $q(z|x)$ to make ELBO higher and higher, KL divergence is getting smaller and smaller. When we adjust $q(z|x)$ to make $q(z|x)$ and $P(z|x)$ to be the same, KL divergence disappears to 0, ELBO and $\log P(x)$ are fully consistent. It can be concluded that we can always adjust ELBO to be equal to $\log P(x)$, and because ELBO is the lower bound of $\log P(x)$, solving for Maximum $\log P(x)$ is equivalent to solving Maximum ELBO.
Adjusting $P(x|z)$ is adjusting the Decoder, and adjusting $q(z|x)$ is adjusting the Encoder. Every time the decoder advances, the Encoder is adjusted to be consistent with it, so that the decoder will only be better after next training epoch.

$$\begin{aligned}\text{ELBO} &= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz \\ &= \int_z q(z|x)\log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz \\ &= \int_z q(z|x)\log\left(\frac{P(z)}{q(z|x)}\right)dz + \int_z q(z|x)\log P(x|z)dz \\ &= -KL\left(q(z|x)||P(z)\right) + \int_z q(z|x)\log P(x|z)dz\end{aligned}$$

Therefore, maximize ELBO is equivalent to minimize $\text{KL}(q(z|x)||P(z))$ and maximize the integral equation in the second term of the above figure.

First, check the second term of the above figure:

$$\text{Maximum} \int_z q(z|x)\log P(x|z)dz$$

$$\text{Maximum } \mathrm{E}_{q(z|x)}\left[\log P(x|z)\right]$$

The above expectation means $P(x|z)$ (Decoder's output) given that $q(z|x)$ (Encoder's output) is as high as possible. This is similar to AutoEncoder's loss function(reconstruction error):

$$(x - \hat{x})^2$$

Let's discuss $-\text{KL}(q(z|x)||P(z))$:

$$\begin{aligned}\int_z q(z|x)\log P(z)\,dz &= \int_z N(z;\mu,\sigma^2)\log N(z;0,I)\,dz \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{J}(\mu_i^2 + \sigma_i^2)\end{aligned}$$

$$\begin{aligned}\int_z q(z|x)\log q(z|x)\,dz &= \int_z N(z;\mu,\sigma^2)\log N(z;\mu,\sigma^2)\,dz \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{J}(1 + \log\sigma_i^2)\end{aligned}$$

Then, we can write $-\text{KL}(q(z|x)||P(z))$ as:

$$-KL\left(q(z|x)||P(z)\right) = \int_z q(z|x)\left(\log P(z) - \log q\left(z|x\right)\right) dz$$

$$= -\frac{1}{2}\sum_{i=1}^{J}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)$$

Conclusion

both EM and VAE are machine learning techniques/algorithms to find the latent variables **z**. However, despite the overall goal and even the **objective function** being the same, there are differences because of the complexities of the model.
There are 2 issues at hand where EM (and its variants) have limitations. These are mentioned in the original VAE paper by Kingma.

> Very importantly, we *do not* make the common simplifying assumptions about the marginal or posterior probabilities. Conversely, we are here interested in a general algorithm that even works efficiently in the case of:
>
> 1. *Intractability*: the case where the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is intractable (so we cannot evaluate or differentiate the marginal likelihood), where the true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable (so the EM algorithm cannot be used), and where the required integrals for any reasonable mean-field VB algorithm are also intractable. These intractabilities are quite common and appear in cases of moderately complicated likelihood functions $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, e.g. a neural network with a nonlinear hidden layer.
> 2. *A large dataset*: we have so much data that batch optimization is too costly; we would like to make parameter updates using small minibatches or even single datapoints. Sampling-based solutions, e.g. Monte Carlo EM, would in general be too slow, since it involves a typically expensive sampling loop per datapoint.

In the EM algorithm, we can calculate the posterior probability, and in the problem solved by VAE, our posterior is intractable, that is, it cannot be calculated. So we have to approximate this posterior in VAE, which is why we use KL divergence in our formula , and this method is actually a variational approximation to the posterior.

Variational Autoencoder | Em Algorithm | Autoencoder

Follow