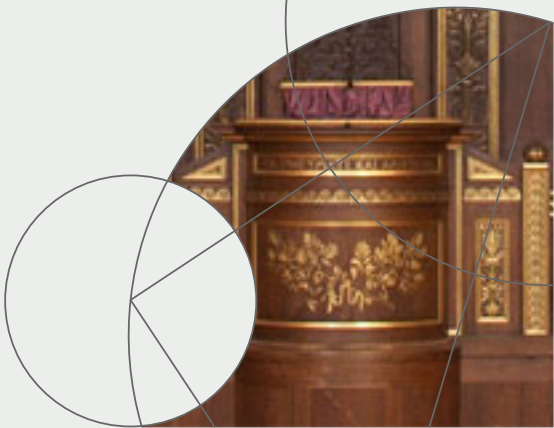
Deep Learning: Variational Auto-Encoder

Rensheng Wang,
https://sit.instructure.com/courses/29876
November 8, 2019



Variational Auto-Encoder

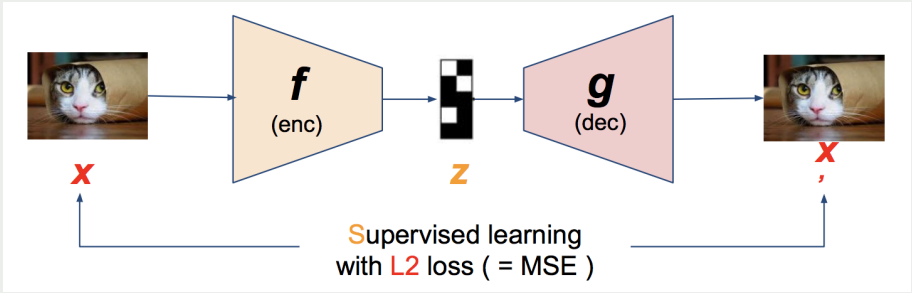
- There are two generative models facing neck to neck in the data generation business right now:
  - ☞ Generative Adversarial Nets (GAN)
  - ☞ Variational Autoencoder (VAE).
- These two models have different take on how the models are trained.
- GAN is rooted in game theory, its objective is to find the Nash Equilibrium between discriminator net and generator net.
- On the other hand, VAE is rooted in bayesian inference, i.e. it wants to model the underlying probability distribution of data so that it could sample new data from that distribution.

Variational Auto-Encoder (VAE)

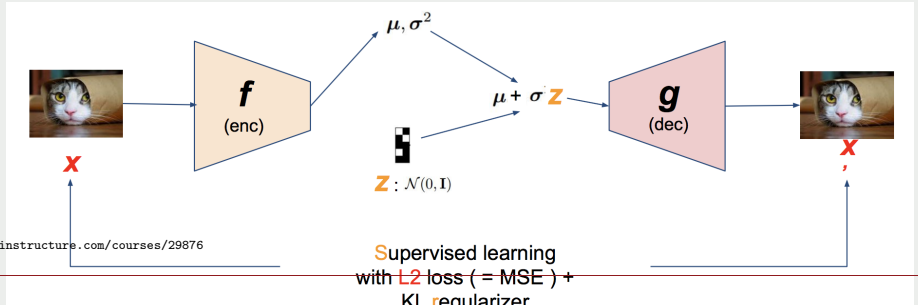
- Kingma et al, Auto-Encoding Variational Bayes, 2013
- Variational auto-encoders have a similar structure with an encoder followed by a decoder.
- Differences compared with vanilla auto-encoders:
  1. Variational Auto-Encoder is probabilistic auto-encoders.
  2. The input data and hidden codes are random variables.
  3. The neural network outputs are partly determined by chance, even after training.
  4. Variational auto-encoders are generative auto-encoders: they can generate new instances which are similar to the samples of the training set.
  5. De-noising auto-encoders only use randomness during training.

Variational Auto-Encoder

- Auto-Encoder

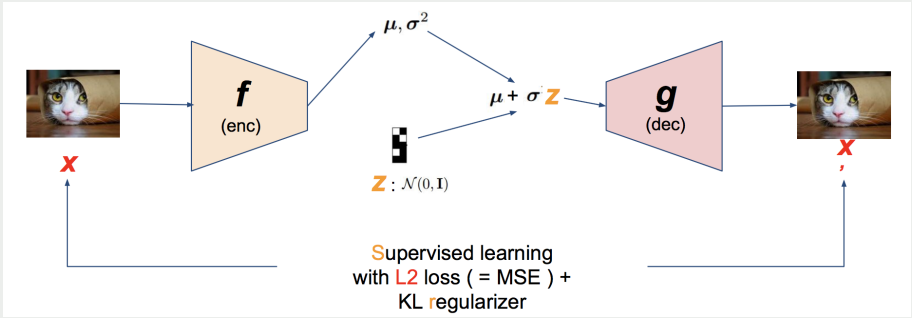


- Variational Auto-Encoder

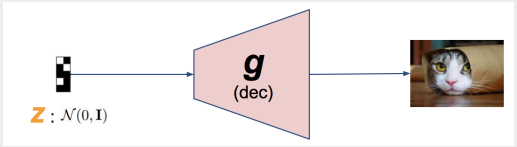


Variational Auto-Encoder

- Training



- Generating (after training, one can very easily generate new instances)



Variational Auto-Encoder

- Different from vanilla auto-encoders, variational auto-encoders add a twist with the hidden coding layer.
- Instead of directly producing a coding for a giving input, the encoder produces a mean coding $\boldsymbol{\mu}$ and standard deviation σ.
- The actual coding is then sample randomly from a Gaussian distribution with mean $\boldsymbol{\mu}$ + standard deviation σ.
- After that the decoder just decodes the sampled coding normally.
- The cost function consists of two parts.
  - the usual reconstruction loss (how similar the auto-encoders output was to its input) $\Delta = \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2$
  - latent loss (how close its hidden nodes were to a normal distribution)

Latent Loss

- From encoding of input data $\{\boldsymbol{x}_i\}$, we have hidden coding layer values $\{\boldsymbol{e}_i\}$

$$f(\boldsymbol{x}_i) = \boldsymbol{e}_i, \quad i = 1, 2, \cdots$$

- Treat both $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{e}_i\}$ as random variables, instead of $\{\boldsymbol{e}_i\}$, we keep the mean and variance of $\{\boldsymbol{e}_i\}$ as the hidden coding layer

$$\boldsymbol{\mu} = \text{mean}\{\boldsymbol{e}_i\}; \qquad \sigma = \text{Var}\{\boldsymbol{e}_i\}$$

- To generate different copies as the input $\{\boldsymbol{x}_i\}$, we sample values from an actual normal distribution with the same dimension as $\boldsymbol{\mu}$ and σ

$$\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$$

- The new hidden coding layer will be $\boldsymbol{z}$ shifted with $\boldsymbol{\mu}$ and scaled by σ

$$\boldsymbol{\mu} + \sigma * \boldsymbol{z}$$

- The latent loss is how close the hidden layer nodes are to normal distribution

$$(\boldsymbol{\mu} + \sigma * \boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{I})$$

Reconstruction Loss vs. Latent Loss

- The variational auto-encoder is to minimize both reconstruction loss and latent loss.
- The smaller the latent loss, the less information can be encoded, and therefore the reconstruction loss goes up.
- As a result, the variational auto-encoder is locked in a trade-off between the latent loss and the reconstruction loss.
- If the latent loss is small, our novel generated images will look a lot like the images at train time, but they will both look really bad.
- If the reconstruction loss is small, then the reconstructed images at train time will look really nice, but our novel generated images will look nothing like the reconstructed images. Obviously we want both, so its important to find a nice equilibrium.

Remarks

- Auto-encoders are artificial neural networks of learning efficient representations of the input data, called codings, without any supervision (i.e., the training set is unlabeled).
- These codings typically have a much lower dimensionality than the input data, making auto-encoders useful for dimensionality reduction.
- More importantly, auto-encoders act as powerful feature detectors, and they can be used for unsupervised pre-training of deep neural networks.
- Lastly, they are capable of randomly generating new data that looks very similar to the training data; this is called a generative model.

VAE: Formulation and Intuition

- Suppose we want to generate a data. Good way to do it is first to decide what kind of data we want to generate, then actually generate the data. For example, say, we want to generate an animal image.
- First, we imagine the animal: it must have four legs, and it must be able to swim. Having those criteria, we could then actually generate the animal by sampling from the animal kingdom. Lo and behold, we get Platypus!
- From the story above, our imagination is analogous to latent variable. It is often useful to decide the latent variable first in generative models, as latent variable could describe our data. Without latent variable, it is as if we just generate data blindly.
- This is the difference between GAN and VAE: VAE uses latent variable, hence its an expressive model.

VAE: Formulation and Intuition

- Before how do we model VAE, we first talk about probability distribution.
- Lets define some notions:
  1. X: data that we want to model a.k.a the animal
  2. z: latent variable a.k.a our imagination
  3. $P(X)$: probability distribution of the data, i.e. that animal kingdom
  4. $P(z)$: probability distribution of latent variable, i.e. our brain, the source of our imagination
  5. $P(X|z)$: distribution of generating data given latent variable, e.g. turning imagination into real animal
- Our objective here is to model the data, hence we want to find $P(X)$. Using the law of probability, we could find it in relation with z as follows:

$$P(X) = \int P(X|z)P(z)dz$$

We marginalize out z from the joint probability distribution $P(X, z)$.

VAE: Formulation and Intuition

- The idea of VAE is to infer $P(z)$ using $P(z|X)$.
- This is making a lot of sense if we think about it: we want to make our latent variable likely under our data.
- But the problem is, we have to infer that distribution $P(z|X)$, as we dont know it yet. In VAE, as it name suggests, we infer $P(z|X)$ using a method called Variational Inference (VI).
- VI is one of the popular choice of method in Bayesian inference, the other one being MCMC method. The main idea of VI is to pose the inference by approach it as an optimization problem.
- By modeling the true distribution $P(z|X)$ using simpler distribution that is easy to evaluate, e.g. Gaussian, and minimize the difference between those two distribution using KL divergence metric, which tells us how difference it is P and Q.

VAE: Formulation and Intuition

- For example, if we want to infer $P(z|X)$ using $Q(z|X)$. The KL divergence then formulated as follows:

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] =& \sum_z Q(z|X) \log \frac{Q(z|X)}{P(z|X)} \\ =& E\left[\log \frac{Q(z|X)}{P(z|X)}\right] \\ =& E\left[\log Q(z|X) - \log P(z|X)\right] \end{aligned}$$

- Recall the aforementioned notations, with Bayes' rule,

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] =& E\left[\log Q(z|X) - \log \frac{P(X|z)P(z)}{P(X)}\right] \\ =& E\left[\log Q(z|X) - (\log P(X|z) + \log P(z)) + \log P(X)\right] \end{aligned}$$

  Notice that the expectation is over z and $P(X)$ doesnt depend on z, so we could move it outside of the expectation.

VAE: Formulation and Intuition

- After removing the term $\log P(X)$,

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] &= E\left[\log Q(z|X) - (\log P(X|z) + \log P(z)) + \log P(X)\right] \\ D_{KL}[Q(z|X)\|P(z|X)] - \log P(X) &= E\left[\log Q(z|X) - (\log P(X|z) + \log P(z))\right] \end{aligned}$$

- If we look carefully at the right hand side of the equation, we would notice that it could be rewritten as another KL divergence.

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] - \log P(X) &= E\left[\log Q(z|X) - (\log P(X|z) + \log P(z))\right] \\ \log P(X) - D_{KL}[Q(z|X)\|P(z|X)] &= E\left[\log P(X|z) - (\log Q(z|X) - \log P(z))\right] \\ &= E\left[\log P(X|z)\right] - E\left[\log Q(z|X) - \log P(z)\right] \\ &= E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right] \end{aligned}$$

- This is the VAE objective function:

$$\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right]$$

VAE: Formulation and Intuition

- VAE objective function:

$$\boxed{\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right]}$$

  - ☞ $Q(z|X)$ that project our data X into latent variable space
  - ☞ z, tha latent variable
  - ☞ $P(X|z)$ that generate data given latent variable

- From above points, we find the similar structure as seen in Auto-Encoder!
- That is, $Q(z|X)$ is the encoder net, z is the encoded representation, and $P(X|z)$ is the decoder net!
- This is why the name of this model is "Variational Auto-Encoder"!

VAE: Dissecting the Objective

$$\boxed{\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right]}$$

- VAE objective function has a very nice interpretation. That is, we want to model our data, which described by $\log P(X)$, under some error $D_{KL}[Q(z|X)\|P(z|X)]$.
- In other words, VAE tries to find the lower bound of $\log P(X)$, which in practice is good enough as trying to find the exact distribution is often untractable.
- That model then could be found by maximizing over some mapping from latent variable to data $\log P(X|z)$ and minimizing the difference between our simple distribution $Q(z|X)$ and the true latent distribution $P(z)$.

VAE: Dissecting the Objective

$$\boxed{\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right]}$$

- As we might already know, maximizing $E[\log P(X|z)]$ is a maximum likelihood estimation. We basically see it all the time in discriminative supervised model, for example Logistic Regression, SVM, or Linear Regression.
- In the other words, given an input z and an output X , we want to maximize the conditional distribution $P(X|z)$ under some model parameters.
- So we could implement it by using any classifier with input z and output X , then optimize the objective function by using for example log loss or regression loss.

VAE: Dissecting the Objective

$$\boxed{\log P(X) - D_{KL}[Q(z|X)\|P(z|X)] = E\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X)\|P(z)\right]}$$

- What about $D_{KL}\left[Q(z|X)\|P(z)\right]$? Here, $P(z)$ is the latent variable distribution. We might want to sample $P(z)$ later, so the easiest choice is $\mathcal{N}(0,1)$. Hence, we want to make $Q(z|X)$ to be as close as possible to $\mathcal{N}(0,1)$ so that we could sample it easily.
- Having $P(z) = \mathcal{N}(0,1)$ also add another benefit. If we also want $Q(z|X)$ to be Gaussian with parameters $\mu(X)$ and $\mathbf{\Sigma}(X)$, i.e. the mean and variance given X, the KL divergence between those two distribution could be computed in closed form!

$$D_{KL}\left[\mathcal{N}(\mu(X), \mathbf{\Sigma}(X))\|\mathcal{N}(0,1)\right] = \frac{1}{2}\left(\mathrm{tr}(\Sigma(X)) + \mu(X)^T\mu(X) - k - \log\det(\Sigma(X))\right)$$

where

- k is the dimension of our Gaussian
- $\mathrm{tr}(X)$ is trace function, i.e., sum of the diagonal of matrix X.
- $\det(X)$ is the determinant of matrix. (For diagonal matrix, $\det(X)$ is the product of its diagonal).

VAE: Dissecting the Objective

- For diagonal matrix, we could implement $\Sigma(X)$ as just a vector,

$$\begin{aligned} D_{KL}\left[\mathcal{N}(\mu(X), \mathbf{\Sigma}(X)) \| \mathcal{N}(0,1)\right] =& \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \log \prod_k \Sigma(X)\right) \\ =& \frac{1}{2}\left(\sum_k \Sigma(X) + \sum_k \mu^2(X) - \sum_k 1 - \sum_k \log \Sigma(X)\right) \\ =& \frac{1}{2}\sum_k \left(\Sigma(X) + \mu^2(X) - 1 - \log \Sigma(X)\right) \end{aligned}$$

- In practice, however, its better to model $\Sigma(X)$ as $\log \Sigma(X)$, as it is more numerically stable to take exponent compared to computing $\log$. Hence, our final KL divergence term is:

$$D_{KL}\left[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0,1)\right] = \frac{1}{2}\sum_k \left(\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X)\right)$$