Latent Variable Models

Distributions $P(X)$

Dashpoint X in high dimensional space $\mathcal{X}$

Generative model task

↓

X distributed according unknown $P_{gt}(X)$

↓ Goal

A model P is as similar to P_{gt}

Situations:

Every dashpoints X in the data set, there is one or more settings of the latent variables which causes the model to generate something similar to X

↓ creating

A vector of latent variable in high dimensional space $\mathcal{Z}$, which can be easily sampled according to probability density function $P(z)$ defined over $\mathcal{Z}$

→ shows $P = \int P(z)\, dz$ ← variable

(↓ probability distribution; ↓ density function)

Then, creat a family of deterministic functions $f(z;\theta)$, parametered by θ in some space (H)

where $f : \mathcal{Z} \times (H) \to \mathcal{X}$

Then $f(z;\theta)$ is a random variable in the space $\mathcal{X}$

From all of above, we want to optimize θ such that we can sample z from $P(z)$ and, with high probability, $f(z;\theta)$ will be like the X's in our dataset

↓ maximize the probability of each X according to

$$P(X) = \int f(z;\theta) P(z) \, dz$$

↓

$f(z;\theta)$ can be replaced by a distribution $P(X|z;\theta)$, which allows us to make the dependence of X on z

↓ maximum likelihood ⇒ 1. training sets samples 2. Similar 2. dissimilar

The output distribution is often Gaussian in VAE

$$P(X|z;\theta) = N(X|f(z;\theta), \sigma^2 * I)$$

means it has mean $f(z;\theta)$

covariance $= I * \sigma^2$

why use Gaussian

↓ use gradient descent to increase $P(X)$

↓ by making $f(z;\theta)$ approach X for the variable z

$$P(z) = N(z|0, I)$$

setting up the objective.

we need to introduce a new function $Q(z|X)$ which can take a value of X and give us a distribution over z values that likely to produce X.

which means the space of z values under Q will be much smaller than $P(z)$

why? → for most of z, $P(X|z)$ will be zero, hence contribute nothing to our estimate of $P(X)$

Kullback–Leibler divergence (KL divergence or D)

$$D[Q(z)||P(z|X)] = E_{z\sim Q}[\log Q(z) - \log P(z|X)]$$

↓ by applying Bayes rule to $P(z|X)$

$$D[Q(z)||P(z|X)] = E_{z\sim Q}[\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)$$

↓ $\log P(X)$ is not depend on z, rearrange the equation

$$\log P(X) - D[Q(z)||P(z|X)] = E_{z\sim Q}[\log P(X|z)] - D[Q(z)||P(z)]$$

$Q(z) = Q(z|X)$

$$\min D[Q(z|X)||P(z|X)]$$

$\log P(X)$ plus an error term (which make's Q produce z's that can reproduce a given X; this term should be small, if Q is high capacity

something we can optimise using stochastic descent

The framework:
Q is 'encoding' X into z
P is 'decoding' z into X

$P(z|X)$ is not something we can compute analytically: it describe the values of z that are likely to give rise to a sample like X

because $Q(z|X)$ is pulling to match $P(z|X)$, make intractable $p(z|X)$ tractable

Optimizing the objective.

how to use stochastic gradient descent on the right hand side? → $Q(z|x)$ need to be specific

$\therefore Q(z|x) = N(z|u(x;\vartheta), \Sigma(x;\vartheta))$ u, Σ are arbitrary deterministic functions with parameter ϑ → can be learned from data.

Σ is constrained to diagonal matrix to make it computational and clear.

ϑ is omitted for easy handwriting

So, the KL divergence $D[Q(z|x) \| P(z)]$ can be expressed by two multivariate Gaussian distributions

$$D(N(u_0, \Sigma_0) \| N(u_1, \Sigma_1)) = \frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (u_1 - u_0)^T \Sigma_1^{-1}(u_1 - u_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

where k is the dimensionality of the distribution

↓ it can be simplified to.

$$D[N(u(x), \Sigma(x)) \| N(0, I)] = \frac{1}{2}\left(\mathrm{tr}\,\Sigma(x) + (u(x))^T(u(x)) - k - \log\det(\Sigma(x))\right)$$

For equation

$$\log P(x) - D[Q(z|x) \| P(z|x)] = E_{z\sim Q}[\log P(x|z)] - D[Q(z|x) \| P(z)]$$

Tricks:

for $E_{z\sim Q}[\log P(x|z)]$, if we use sampling to estimate it, getting a good estimate requires passing many samples of z through f, it's expensive.

So, in std, we take one sample of z and treat $\log P(x|z)$ as an approximation of $E_{z\sim Q}[\log P(x|z)]$, After all, we already doing std over different values of X sampled from dataset. :D

So the equation we need to optimize is

$$E_{x\sim D}[\log P(x) - D[Q(z|x) \| P(z|x)]] = E_{x\sim D}[E_{z\sim Q}[\log P(x|z)] - D[Q(z|x) \| P(z)]]$$

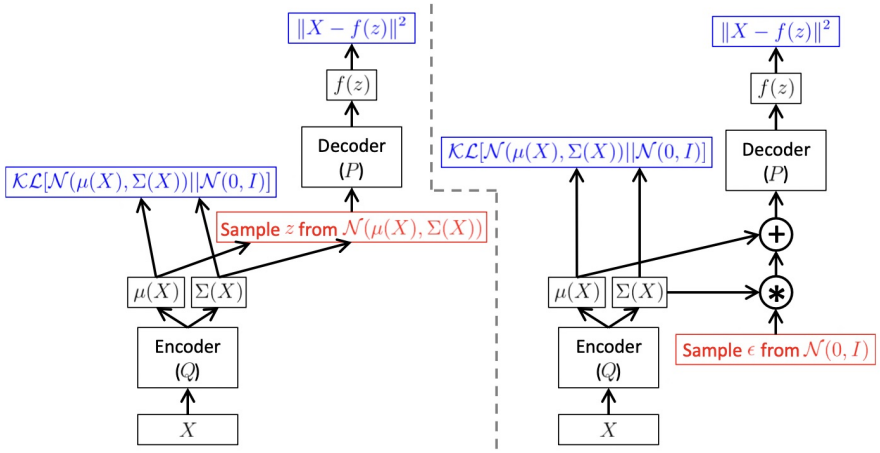

Figure 4: A training-time variational autoencoder implemented as a feedforward neural network, where $P(X|z)$ is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

Probability density function

Definition

value at any given sample (or point) in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a relative likelihood that the value of random variable would be equal to that sample.

probability density is the probability per unit length

↓

So, absolute likelihood for a particular value is 0

↓

Since an infinite set of possible random values.

Absolutely continuous univariate distribution

$Pr[a \le X \le b] = \int_b^a f(x)\,dx$ A random variable has density f_X

And F_X is the cumulative distribution function of X

$$F_x = \int_{-\infty}^{x} f(u)\,du.$$

If f_x is continuous at (x)

$$f_X(x) = \frac{d}{dx}(F_x(x))$$

Formal definition

countable intersection

$\sigma\cap A, \sigma\cup A, (\cup_n (A\cap\sigma)) = \cup_n (A\cap\sigma)$

A random variable X with values in a measurable space (Ω, A) (usually $\mathbb{R}^n$ with the Borel sets as subsets) has as probability distribution the measure X_*P on (Ω, A): the density of X with respect to a reference measure μ on (Ω, A) is:

$$f = \frac{dX_*P}{d\mu}$$

That is, f is any measurable function

$$Pr[X \in A] = \int_{X^{-1}A} dP = \int_A f\,d\mu$$

Covariance

$$Cov(X,Y) = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{n} \longrightarrow \text{for population}$$

$$Cov(X,Y) = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \longrightarrow \text{for a sample.}$$

just show the direction, while correlation can show the strength of relationship

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \times \sigma_Y}$$

σ → standard deviation.