Variational Autoencoder

Hongming Shan
hmshan@ieee.org

Rensselaer Polytechnic Institute

January 22, 2020

1 Introduction

2 Preliminary

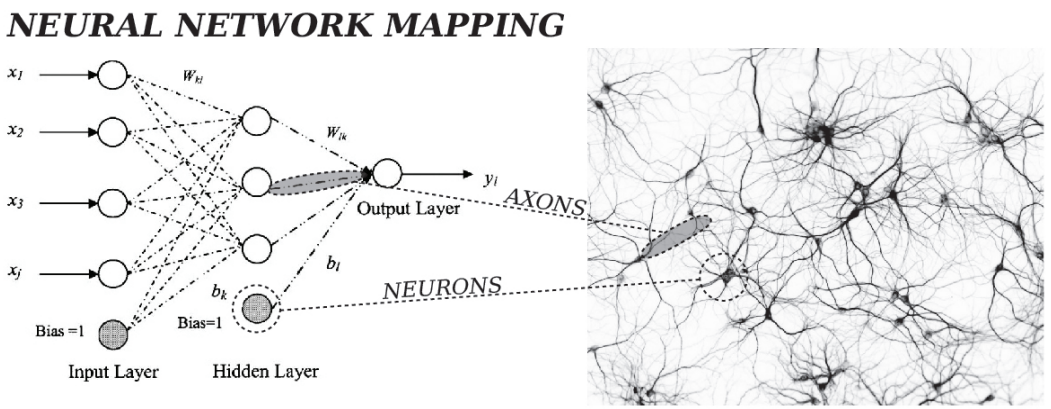
3 Variational Autoencoder

4 Extension of VAE

Two tribes of machine learning

- **Statistical learning**

- **Artificial neural networks**



Topics in this presentation

Generative vs. Discriminative model [Jordan, 2002]

- **Generative model**
  Learn a model of the joint probability, $p(x, y)$, of the inputs x and the label y, and make their prediction by using Bayes rules to calculate $p(y|x)$, and picking the most likely label y

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

- **Discriminant model**
  Model the posterior $p(y|x)$ directly, or learn a direct map from inputs to the class labels.

$$p(y|x)$$

Representative methods

- **Generative model**
  Gaussian mixture model, Hidden Markov model, Naive Bayes, Latent Dirichlet allocation, Generative adversarial networks....
- **Discriminant model**
  Logistic regression, Support vector machine, Boosting, Conditional random fields, Linear regression, Neural networks, Random Forests,...

Quotation from Vapnik

> " One should solve the [classification] problem directly and never solve a more general problem as an intermediate step [such as modeling $p(x|y)$]"
>
> — Vapnik

Which one is better?

Suppose your goal is predicting Y given X. You will get better accuracy on training data by using discriminative objective. If your training set is big enough, this means better accuracy on future data. For small training sets, good performance on training data does not transfer to future data, and you get better accuracy by training your model generatively. In generative training, the marginal part of the objective essentially works as a regularizer. *From Quora*

The advantage of generative model

Inference and generate new sample

1 Introduction

2 Preliminary

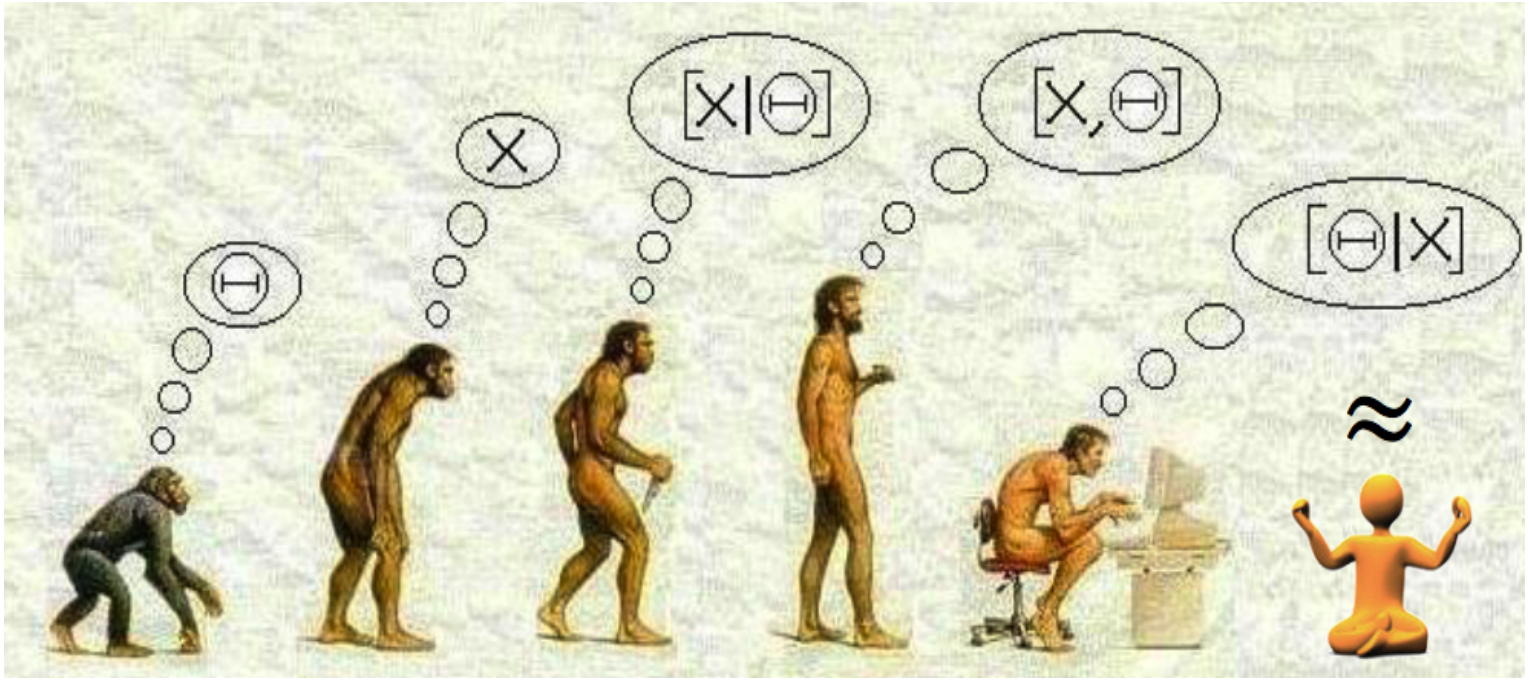
3 Variational Autoencoder

4 Extension of VAE

Preliminary

- **Variational Bayesian (VB) inference**
- **Autoencoder (AE)**

Variational Bayesian inference



Variational Bayesian inference

> “ An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem”
>
> — John W. Tukey, 1915-2000

Recall Bayes rule

Likelihood

How probable is the evidence given that our hypothesis is true?

Prior

How probable was our hypothesis *before* observing the evidence?

$$P(H \mid e) = \frac{P(e \mid H)\, P(H)}{P(e)}$$

Posterior

How probable is our hypothesis given the observed evidence? (Not directly computable)

Marginal

How probable is the new evidence under all possible hypotheses?

$$P(e) = \sum P(e \mid H_i)\, P(H_i)$$

Approximate Bayesian inference

Bayesian inference formalizes **model inversion**, the process of passing from a prior to a posterior in light of data

$$\underbrace{p(\theta|y)}_{\text{posterior}} = \frac{\overbrace{p(y|\theta)}^{\text{likelihood}}\ \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{\int p(y,\theta)\mathrm{d}\theta}_{\text{marginal likelihood } p(y) \text{ (model evidence)}}}$$

In practice, evaluating the posterior is usually difficult because we cannot easily evaluate $p(y)$, especially when:

- analytical solutions are not available
- numerical integration is too expensive

Approximate Bayesian inference

There are two approaches to approximate inference. They have complementary strengths and weaknesses.

Approximate Bayesian inference

There are two approaches to approximate inference. They have complementary strengths and weaknesses.

Stochastic approximate inference

in particular sampling

1. design an algorithm that draws samples $\theta^{(1)}, \ldots, \theta^{(m)}$ from $p(\theta|y)$
2. inspect sample statistics (e.g., histogram, sample quantiles, ...)

☑ asymptotically exact
☒ computationally expensive
☒ tricky engineering concerns

Structural approximate inference

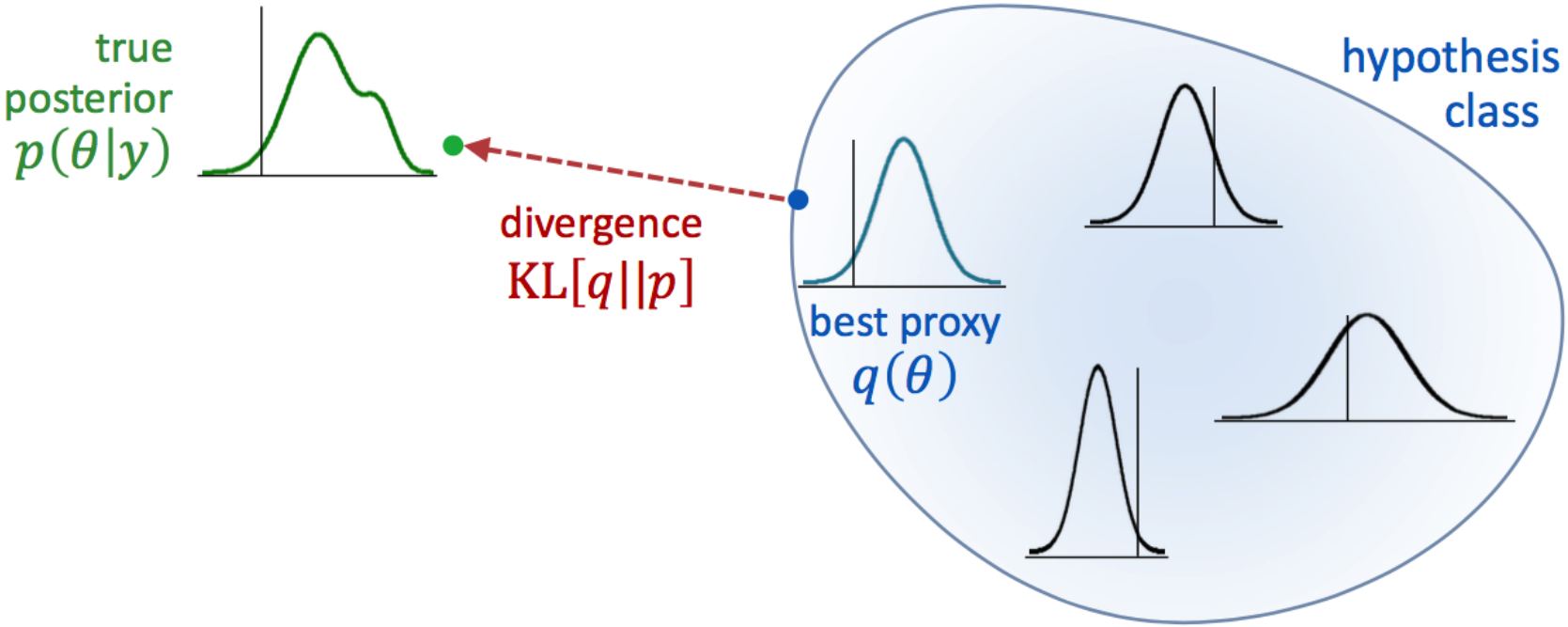
in particular variational Bayes

1. find an analytical proxy $q(\theta)$ that is maximally similar to $p(\theta|y)$
2. inspect distribution statistics of $q(\theta)$ (e.g., mean, quantiles, intervals, ...)

☑ often insightful – and lightning-fast!
☒ often hard work to derive
☒ requires validation via sampling

Variational Bayesian (VB)

In VB, we wishes to find an approximate density that is maximally similar to the true posterior.



Variational calculus

Variational Bayesian inference is based on variational calculus.

Standard calculus
Newton, Leibniz, and others

- functions $f: x \mapsto f(x)$
- derivatives $\frac{\mathrm{d}f}{\mathrm{d}x}$

Example: maximize the likelihood expression $p(y|\theta)$ w.r.t. θ

Variational calculus
Euler, Lagrange, and others

- functionals $F: f \mapsto F(f)$
- derivatives $\frac{\mathrm{d}F}{\mathrm{d}f}$

Example: maximize the entropy $H[p]$ w.r.t. a probability distribution $p(x)$

Leonhard Euler
(1707 – 1783)
Swiss mathematician
'Elementa Calculi Variationum'

Variational calculus and the free energy

Variational calculus leads itself nicely to approximate Bayesian inference.

$$
\begin{aligned}
\ln p(y) &= \ln \frac{p(y,\theta)}{p(\theta|y)} \\
&= \int q(\theta) \ln \frac{p(y,\theta)}{p(\theta|y)} \, d\theta \\
&= \int q(\theta) \ln \frac{p(y,\theta)}{p(\theta|y)} \frac{q(\theta)}{q(\theta)} \, d\theta \\
&= \int q(\theta) \left(\ln \frac{q(\theta)}{p(\theta|y)} + \ln \frac{p(y,\theta)}{q(\theta)} \right) d\theta \\
&= \underbrace{\int q(\theta) \ln \frac{q(\theta)}{p(\theta|y)} \, d\theta}_{\mathrm{KL}[q||p]} + \underbrace{\int q(\theta) \ln \frac{p(y,\theta)}{q(\theta)} \, d\theta}_{F(q,y)}
\end{aligned}
$$

$\mathrm{KL}[q||p]$: divergence between $q(\theta)$ and $p(\theta|y)$

$F(q,y)$: free energy

Variational calculus and the free energy

In summary, the log model evidence can be expressed as:

$$\ln p(y) = \underbrace{\mathrm{KL}[q||p]}_{\text{divergence}} + \underbrace{F(q,y)}_{\text{free energy}}$$

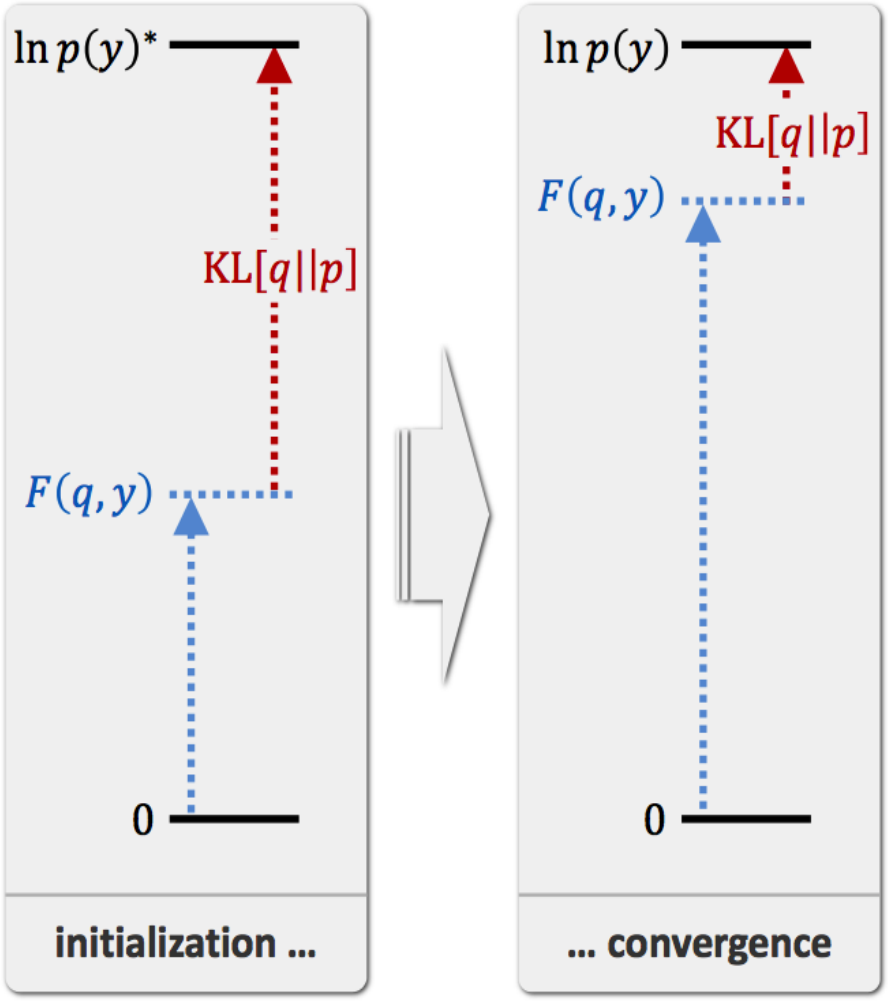
divergence ≥ 0 (unknown)

free energy (easy to evaluate for a given q)

Maximizing $F(q,y)$ is equivalent to:

- minimizing $\mathrm{KL}[q||p]$
- tightening $F(q,y)$ as a lower bound to the log model evidence

* In this illustrative example, the log model evidence and the free energy are positive; but the above equivalences hold just as well when the log model evidence is negative.



Computing the free energy

We can decompose the free energy $F(q, y)$ as follows:

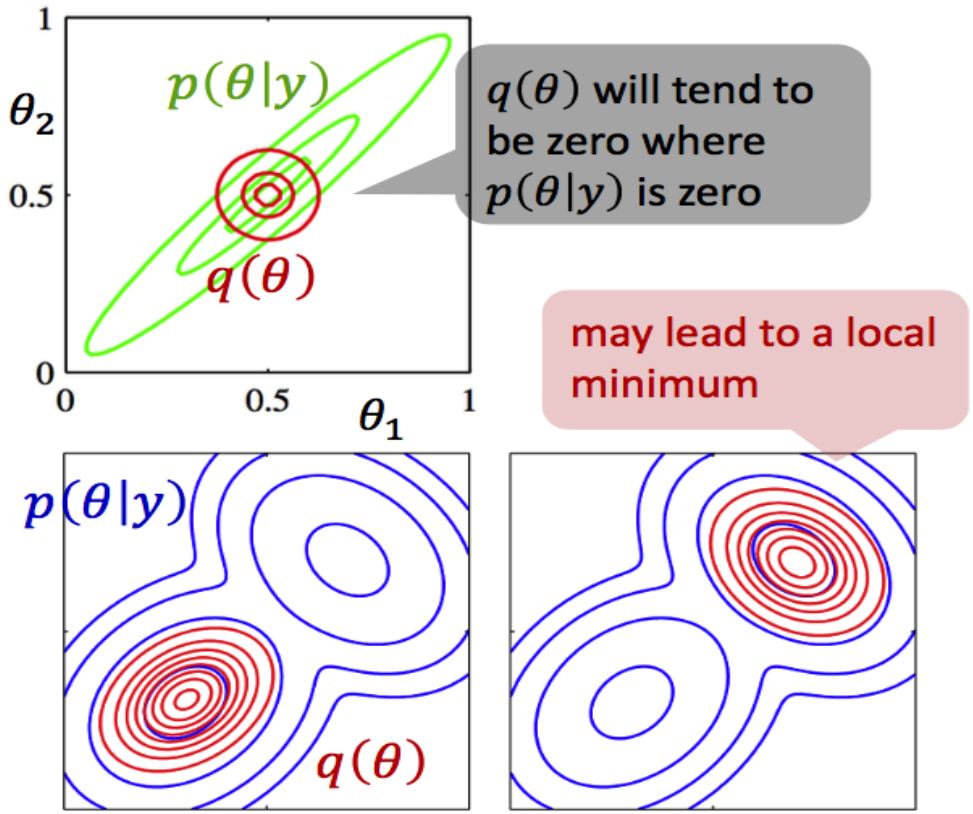
$$F(q,y) = \int q(\theta) \ln \frac{p(y,\theta)}{q(\theta)} \mathrm{d}\theta$$

$$= \int q(\theta) \ln p(y,\theta) \; d\theta - \int q(\theta) \ln q(\theta) \; d\theta$$

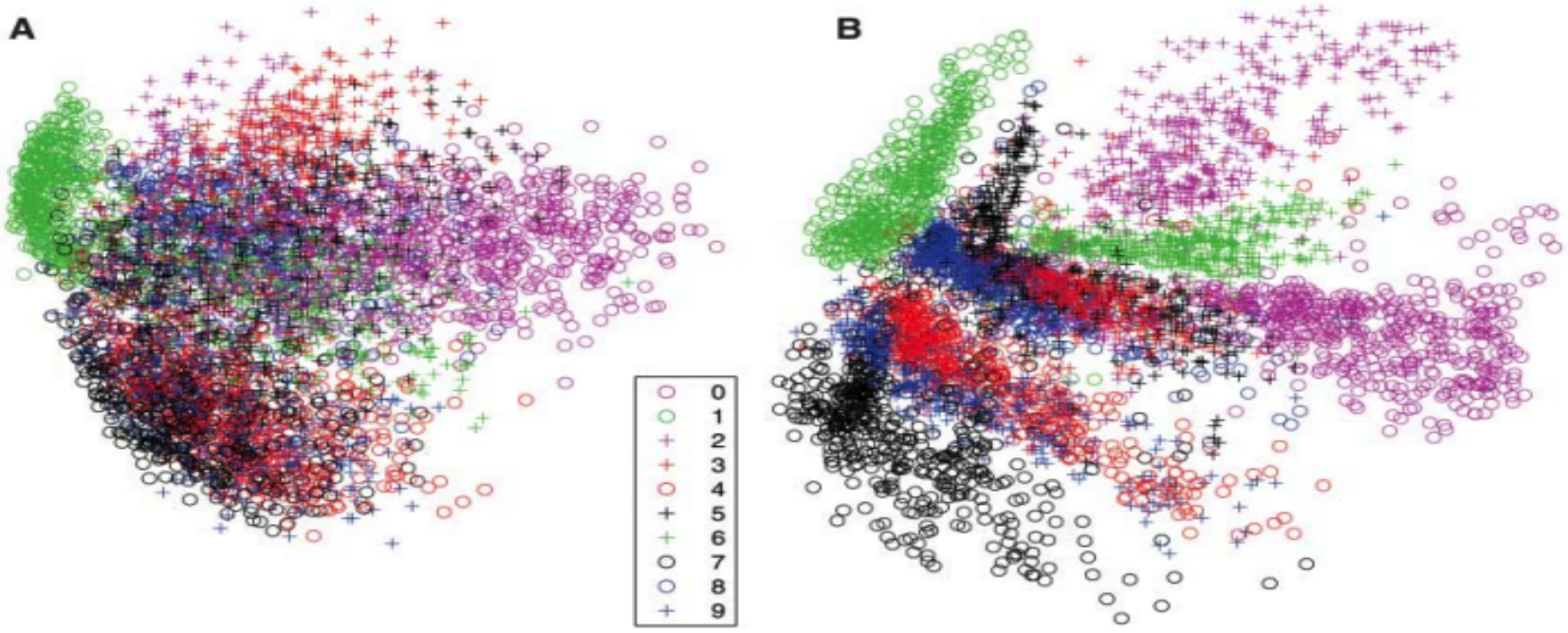
$$= \underbrace{\langle \ln p(y,\theta) \rangle_q}_{\text{expected log-joint}} + \underbrace{H[q]}_{\text{Shannon entropy}}$$

Variational Bayes

$$\min \mathrm{KL}[q(\theta)||p(\theta|y)]$$



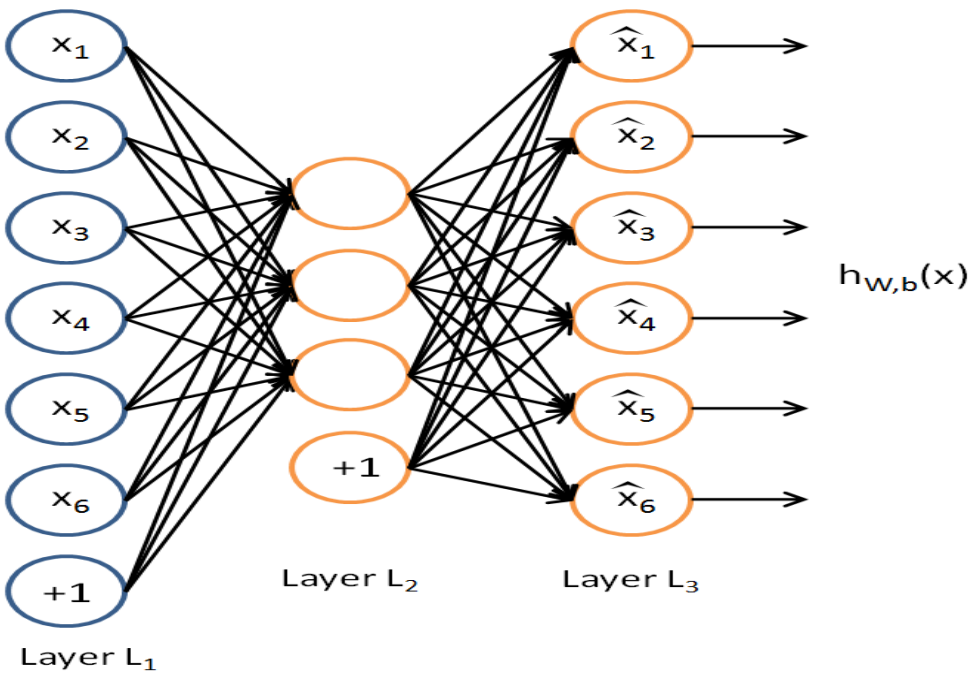
Autoencoder



Autoencoder

An unsupervised multi-layer neural networks, aiming to reconstruct itself.

$$\min \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - h_{w,b}(\boldsymbol{x}_i) \right\|^2$$



Autoencoder

Autoencoder is usually used as

- **Dimension reduction**
- **Initialize the deep neural network**
- **nonlinear, nonconvex, scalable function**

1 Introduction

2 Preliminary

3 Variational Autoencoder

4 Extension of VAE

Variational autoencoder

Autoencoder is used as the nonlinear, nonconvex, scalable function to approximate true posterior distribution in variational Bayesian inference.
Doersch, Carl. “Tutorial on variational autoencoders.” arXiv preprint arXiv:1606.05908 (2016).

Latent variable models

Let's start with a latent variable models

In statistics, **latent variables** (as opposed to **observable variables**), are variables that are not directly observed but are rather inferred (through a mathematical model) from other variables that are observed (directly measured)

Experimental setup

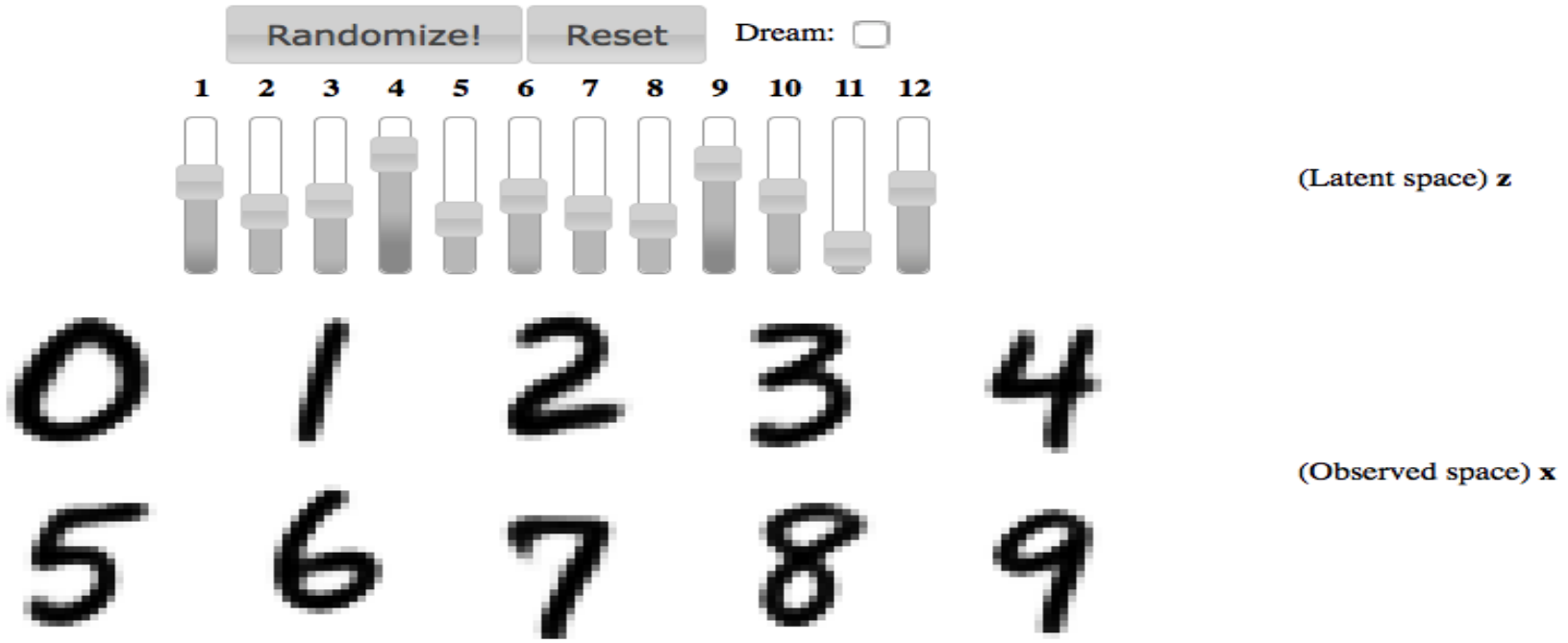
- Assume X is a data point in the dataset.
- Latent variables z in a high-dimensional space $\mathcal{Z}$ drawn from $P(z)$ defined over $\mathcal{Z}$.
- Have a family of deterministic functions $f(z;\theta)$, parameterized by a vector θ in some space Θ, such that

$$f : \mathcal{Z} \times \Theta \to \mathcal{X}$$

- f is deterministic, but if z is random and θ is fixed, then $f(z;\theta)$ is a random variable in the space $\mathcal{X}$.
- We wish 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

Demo on MNIST

Digit Fantasies by a Deep Generative Model



http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Latent variable models

Aiming to maximize the probability of each X in the training set under the entire generative process, according to:

$$P(X) = \int P(X|z;\theta)P(z)dz. \tag{1}$$

Here, $f(z;\theta)$ has been replaced by a distribution $P(X|z;\theta)$, which allows us to make the dependence of X on z explicit by using the law of total probability.

The intuition behind this framework—called "maximum likelihood"—is that if the model is likely to produce training set samples, then it is also likely to produce similar samples, and unlikely to produce dissimilar ones.

Output distribution

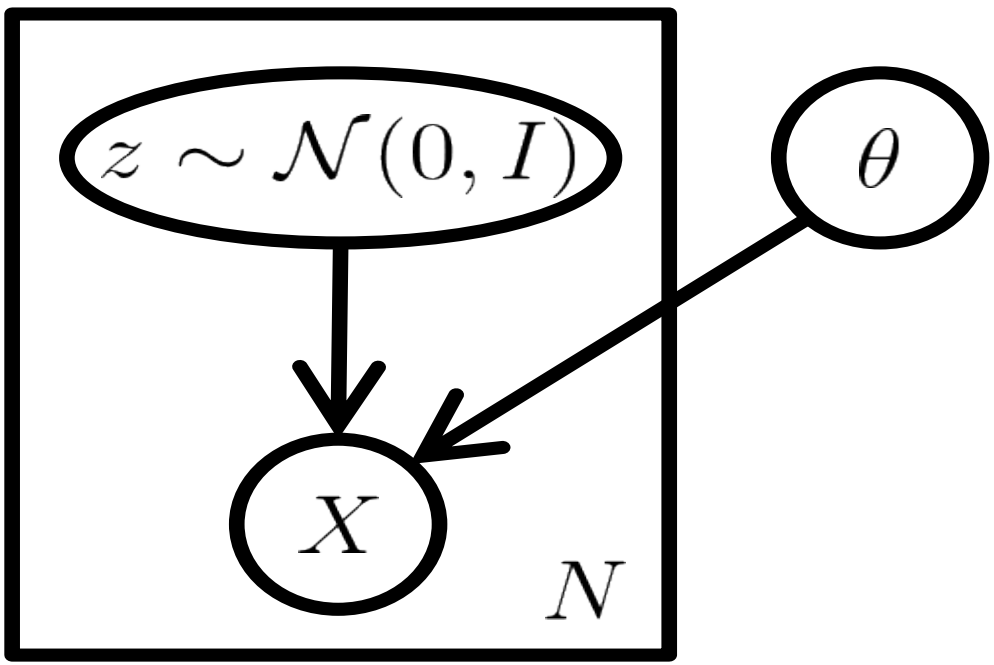
- if X is real-valued variable, usually, $P(X|z;\theta)$ is parameterized by Gaussian distribution

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

- if X is binary,
  $P(X|z)$ might be a Bernoulli parameterized by $f(z;\theta)$

Variational autoencoder [Doersch, 2016, Kingma and Welling, 2013, Rezende et al., 2014]

VAEs approximately maximize Equation (1), according to following graphical model



Two problems when solving Equation 1

$$P(X) = \int P(X|z;\theta)P(z)dz.$$

Two problems that VAE must deal with...

- How to define the latent variables z (i.e., decide what information they represent)
- How to deal with the integral over z

Q1: Define the latent variable

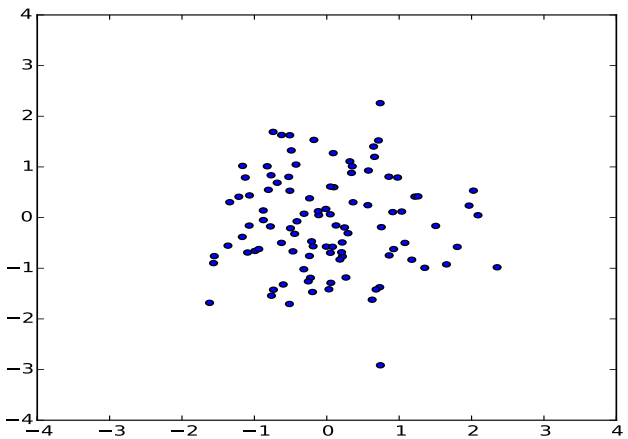
- Unusual approach to deal with this problem.
- Assume that there is no interpretation of the dimension of z.
- Just define z drawn from a simple distribution,

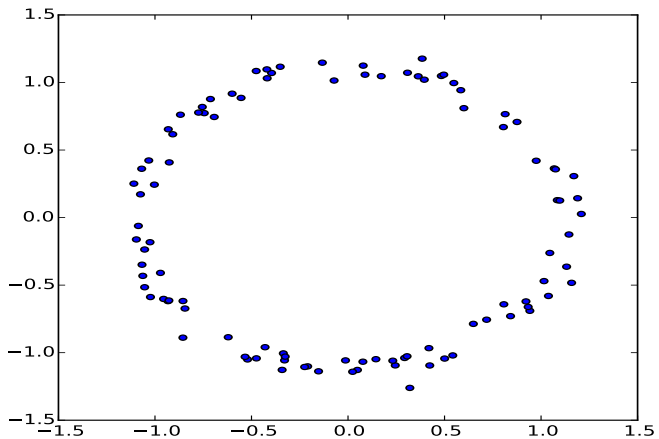
$$z \sim \mathcal{N}(0, I)$$

where I is the identity matrix.

Explanation to Q1

Any distribution in d dimensions can be generated by taking a set of d variables that are normally distributed and mapping them through a sufficiently complicated function





Gaussian distribution→Ring : $g(z) = z/10 + z/||z||$

Explanation to Q1

- Map independent normally-distributed z values to X by a **powerful function approximators**.
- Recall that

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

If $f(z;\theta)$ is a **multi-layer neural network**, then we can imagine the network using its first few layers to map the normally distributed z's to the latent values (like digit identity, stroke weight, angle, etc.) with exactly the right statistics.

Q2: Deal with integral over z

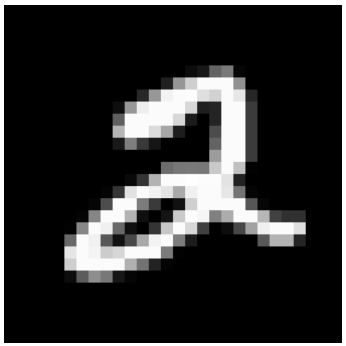
Now to optimize

$$\max P(X) = \int P(X|z;\theta)P(z)dz, \quad \text{s.t.} \quad P(z) = \mathcal{N}(z|0, I)$$

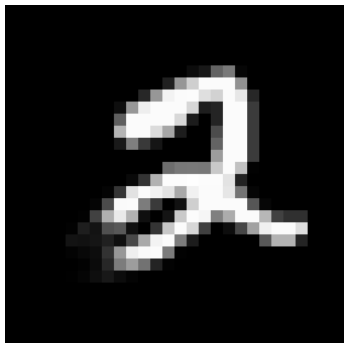
Q2: Deal with integral over z

- Conceptually straightforward: Sample a large number of z values $\{z_1, ..., z_n\}$, and compute $P(X) \approx \frac{1}{n} \sum_i P(X|z_i)$. n needs to be large.
- Since $P(X|z)$ is an isotropic Gaussian, the negative log probability of X is proportional squared Euclidean distance between $f(z)$ and X.

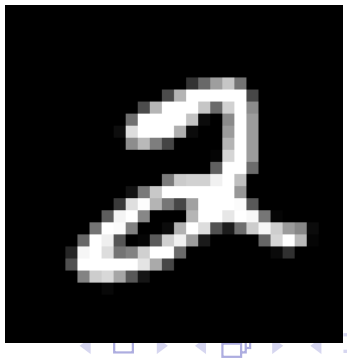
Given an image X (a), the middle sample (b) is much closer in Euclidean distance than the one on the right (c).



(a)



(b)



(c)

Q2: Deal with integral over z

- Should set the σ hyperparameter of our Gaussian distribution such that this kind of erroneous digit does not contribute to $P(X)$.
- A model which produces (c) (identical to X but shifted down and to the right by half a pixel) might be a good model. We would hope that this sample would contribute to $P(X)$

Problem: We cannot have it both ways.

Instead, VAEs alter the sampling procedure to make it faster, without changing the similarity metric.

Setting up the objective

Is there a shortcut we can take when using sampling to compute Equation 1?

Setting up the objective

- In practice, for most z, $P(X|z)$ will be nearly zero, and hence contribute almost nothing to our estimate of $P(X)$.
- **Key idea behind the variational autoencoder** is to attempt to sample values of z that are likely to have produced X, and compute $P(X)$ just from those.
- Need a new function $Q(z|X)$ which can take a value of X and give us a distribution over z values that are likely to produce X.
- Hopefully the space of z values that are likely under Q will be much smaller than the space of all z's that are likely under the prior $P(z)$.
- This lets us compute $E_{z\sim Q}P(X|z)$ relatively easily.

Setting up the objective

if z is sampled from an arbitrary distribution with PDF $Q(z)$, which is not $\mathcal{N}(0, I)$, then how does that help us optimize $P(X)$?

Setting up the objective

if z is sampled from an arbitrary distribution with PDF $Q(z)$, which is not $\mathcal{N}(0, I)$, then how does that help us optimize $P(X)$?

Relate $E_{z\sim Q}P(X|z)$ and $P(X)$

Setting up the objective

Firstly, employ KL divergence to measure the difference between $P(z|X)$ and $Q(z)$.

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

By applying Bayes rules, $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$

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

Setting up the objective

Firstly, employ KL divergence to measure the difference between $P(z|X)$ and $Q(z)$.

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

By applying Bayes rules, $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$

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

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

Inferring $P(X)$

- X is fixed, and Q can be *any* distribution, not just a distribution which does a good job mapping X to the z's that can produce X.
- **Inferring** $P(X)$, construct a Q which does depend on X, making $\mathcal{D}\left[Q(z)||P(z|X)\right]$ small

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

Inferring $P(X)$

- X is fixed, and Q can be *any* distribution, not just a distribution which does a good job mapping X to the z's that can produce X.
- **Inferring** $P(X)$, construct a Q which does depend on X, making $\mathcal{D}\left[Q(z)||P(z|X)\right]$ small

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

the core of the variational autoencoder

Understanding the core of VAE

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

Understanding the core of VAE

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

- **Left side**: maximize: $\log P(X)$ (plus an error term, which makes Q produce z's that can reproduce a given X; this term will become small if Q is high-capacity).

Understanding the core of VAE

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

- **Left side**: maximize: $\log P(X)$ (plus an error term, which makes Q produce z's that can reproduce a given X; this term will become small if Q is high-capacity).
- **Right side**: Optimize via stochastic gradient descent given the right choice of Q

Understanding the core of VAE

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

- **Left side**: maximize: $\log P(X)$ (plus an error term, which makes Q produce z's that can reproduce a given X; this term will become small if Q is high-capacity).
- **Right side**: Optimize via stochastic gradient descent given the right choice of Q

Q is "encoding" X into z, and P is "decoding" it to reconstruct X.

Optimize the objective

Usual choice for $Q(z|X)$

$$Q(z|X) = \mathcal{N}(z|\mu(X;\vartheta), \Sigma(X;\vartheta))$$

where μ and Σ are arbitrary deterministic functions with parameters ϑ that can be learned from data

In practice, μ and Σ are again implemented via neural networks, and Σ is constrained to be a diagonal matrix.

Why this choice?

- The advantages of this choice are computational, as they make it clear how to compute the right hand side.
- The last term—$\mathcal{D}\left[Q(z|X)\|P(z)\right]$—is now a KL-divergence between two multivariate Gaussian distributions, which can be computed in closed form as:

$$\mathcal{D}[\mathcal{N}(\mu_0, \Sigma_0)\|\mathcal{N}(\mu_1, \Sigma_1)] =$$
$$\frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

Why this choice?

- The advantages of this choice are computational, as they make it clear how to compute the right hand side.
- The last term—$\mathcal{D}\left[Q(z|X)\|P(z)\right]$—is now a KL-divergence between two multivariate Gaussian distributions, which can be computed in closed form as:

$$\mathcal{D}[\mathcal{N}(\mu_0, \Sigma_0)\|\mathcal{N}(\mu_1, \Sigma_1)] =$$
$$\frac{1}{2}\left(\operatorname{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

- In our case, this simplifies to:

$$\mathcal{D}[\mathcal{N}(\mu(X), \Sigma(X))\|\mathcal{N}(0, I)] =$$
$$\frac{1}{2}\left(\operatorname{tr}\left(\Sigma(X)\right) + \left(\mu(X)\right)^\top \left(\mu(X)\right) - k - \log\det\left(\Sigma(X)\right)\right).$$

First term on the right hand

Use sampling to estimate $E_{z\sim Q}\left[\log P(X|z)\right]$

As autoencoder is optimized by stochastic gradient descent, take one sample of z and treat $P(X|z)$ for that z as an approximation of $E_{z\sim Q}\left[\log P(X|z)\right]$

Objective

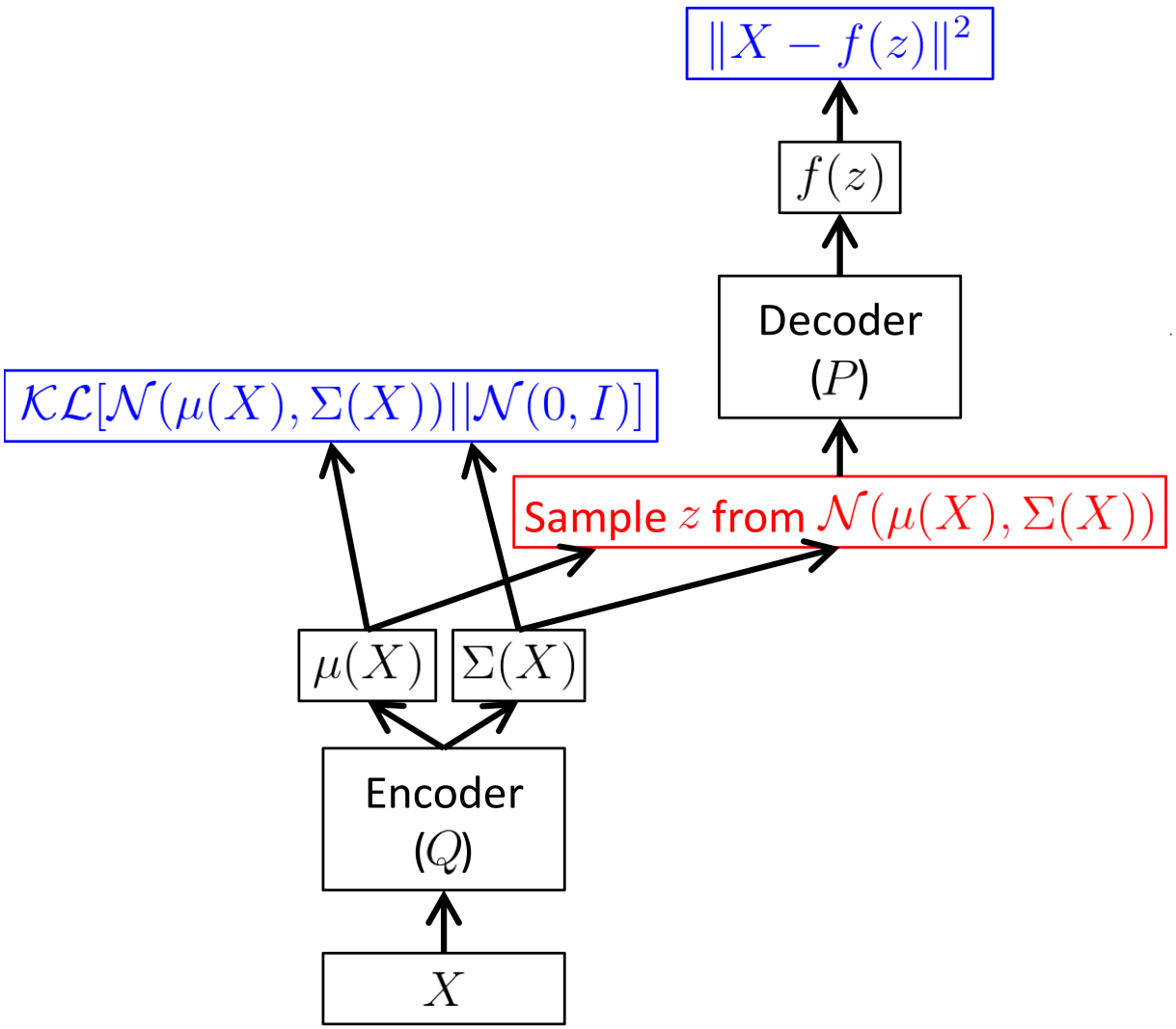
Assume X sampled from a dataset D,
Objective is formulated as

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

Sample a single value of X and a single value of z from $Q(z|X)$, compute the gradient of

$$\log P(X|z) - \mathcal{D}\left[Q(z|X)\|P(z)\right]$$

Variational autoencoder as a feed-forward NN



How to back-propagate error?

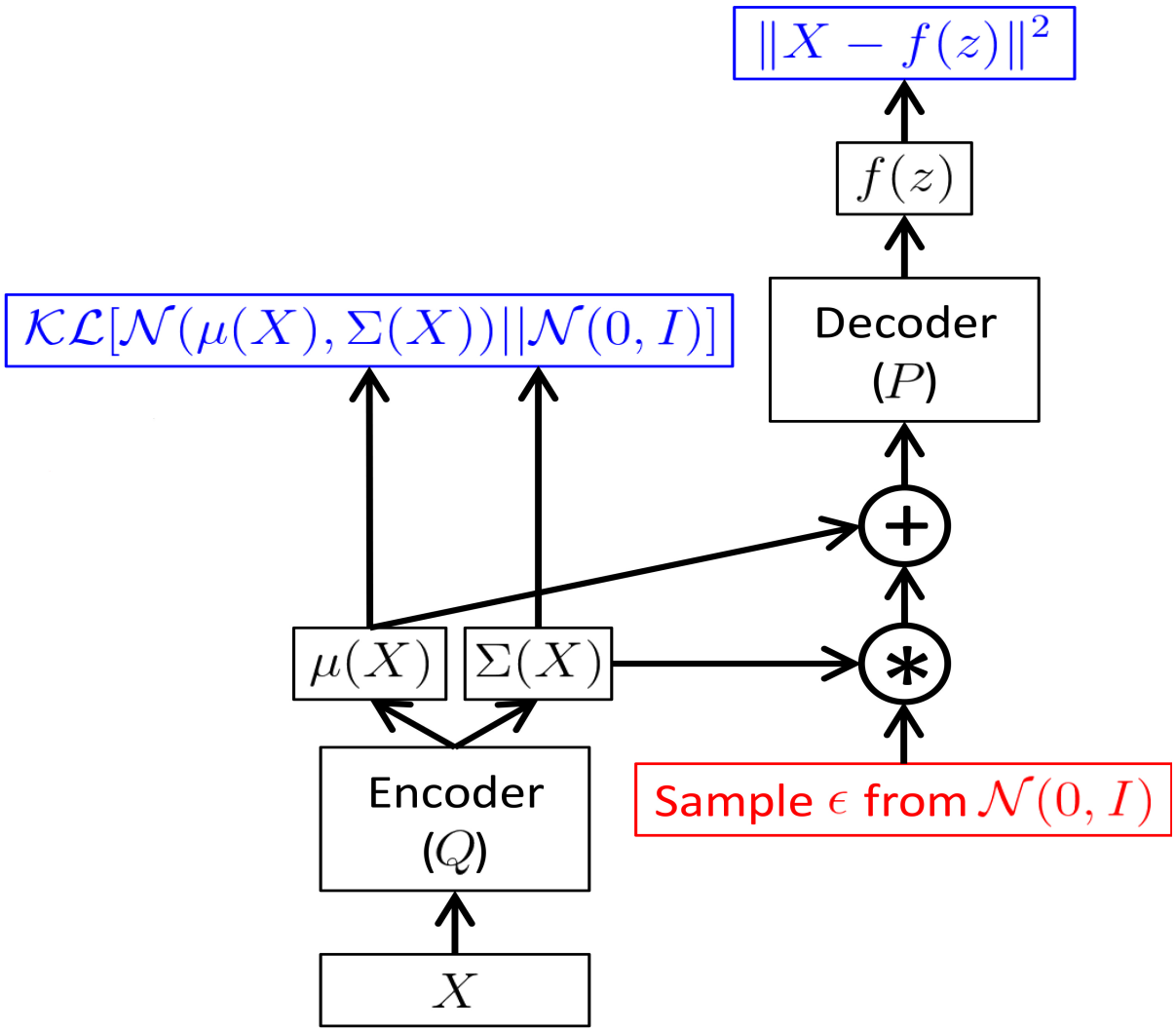
A layer that samples z from $Q(z|X)$, which is non-continuous operation and has no gradient..

Reparameterization trick

Given $\mu(X)$ and $\Sigma(X)$—the mean and covariance of $Q(z|X)$—we can sample from $\mathcal{N}(\mu(X), \Sigma(X))$ by first sampling $\epsilon \sim \mathcal{N}(0, I)$, then computing $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$. Thus, the equation we actually take the gradient of is:

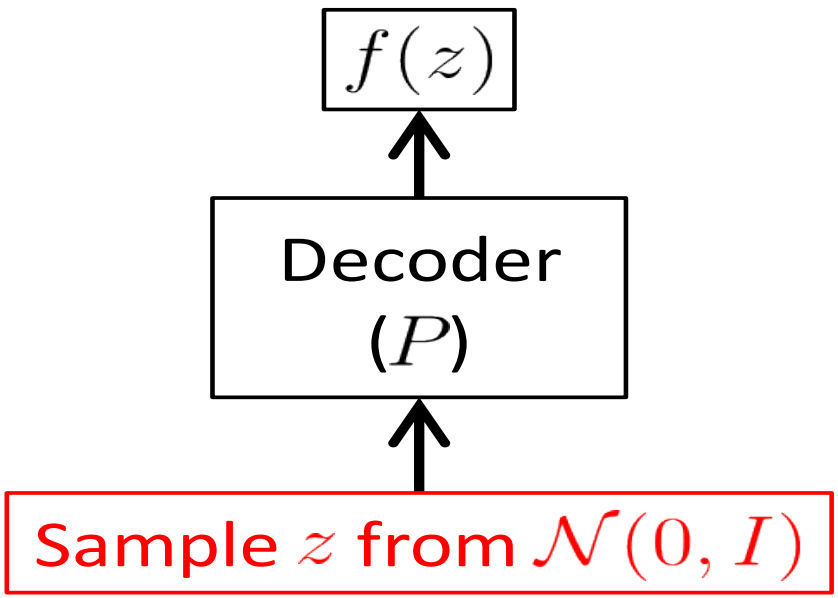
$$E_{X\sim D}\left[E_{\epsilon\sim\mathcal{N}(0,I)}[\log P(X|z=\mu(X)+\Sigma^{1/2}(X)*\epsilon)]-\mathcal{D}\left[Q(z|X)\|P(z)\right]\right]$$

Variational autoencoder with reparameterization trick

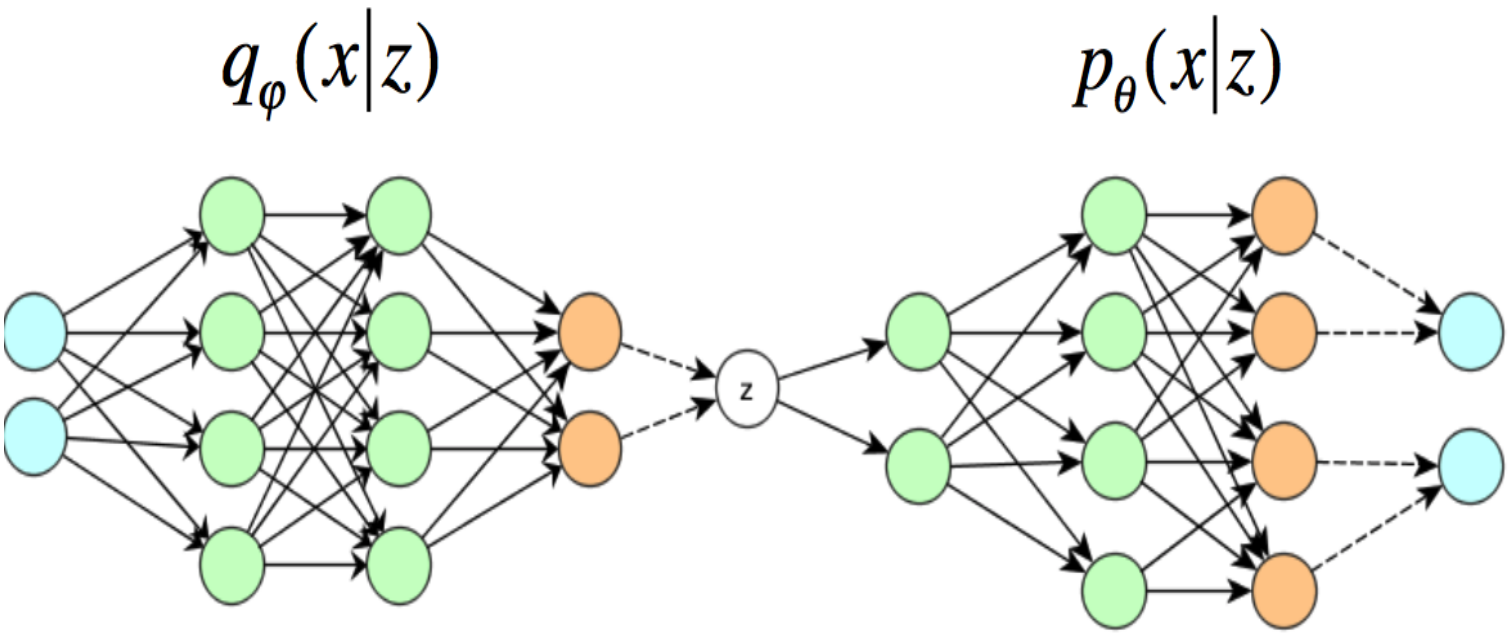


Test

At test time, generate new samples by simply inputting value of $z \sim \mathcal{N}(0, I)$



Network structure



Interpreting the objective

Interpreting the objective from the following three aspects

- How much error is introduced by optimizing $\mathcal{D}[Q(z|X)||P(z|X)]$ in addition to $\log P(X)$
- Describe the VAE framework in terms of information theory
- Investigate whether VAEs have "regularization parameters" analogous to the sparsity penalty in sparse autoencoders

Error from $\mathcal{D}[Q(z|X)\|P(z|X)]$

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

- Rely on assumption that $Q(z|X)$ can be modeled as a Gaussian with some mean $\mu(X)$ and variance $\Sigma(X)$.
- $P(X)$ converges (in distribution) to the true distribution if only if $\mathcal{D}[Q(z|X)\|P(z|X)]$ goes to zero.
- $\mathcal{D}[Q(z|X)\|P(z|X)]$ never goes to zero if the posterior $P(z|X)$ is not necessarily Gaussian for an arbitrary f function.
- **Good news:** there are infinitely many f functions that result in our model generating any given output distribution. Any of these functions will maximize $\log P(X)$ equally well.
- No theory guarantees this function does exist.

The information-theoretic interpretation

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

- $\log P(X)$ can be seen as the total number of bits required to construct X using an ideal encoding.
- Measure the bits required to construct z using a $\mathcal{D}[Q(z|X)||P(z)]$ because under our model, we assume that any z contain no information about X. Hence, we need to measure the amount of *extra* information that we get about X when z comes from $Q(z|X)$ instead of from $P(z)$
- $P(X|z)$ measures the amount of information required to reconstruct X from z under an ideal encoding.
- The total number of bits ($\log P(X)$) is the sum of these two steps, minus a penalty we pay for Q being a sub-optimal encoding ($\mathcal{D}[P(z|X)||Q(z|X)]$)

VAEs and regularization parameter

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

- Whether VAE like sparse autoencoder has regularization parameter as follows:

$$\|\phi(\psi(X)) - X\|^2 + \lambda\|\psi(X)\|_0 \tag{2}$$

- $\mathcal{D}[Q(z|X)||P(z)]$ is not a regularization term...

VAE in PyTorch: Network structure

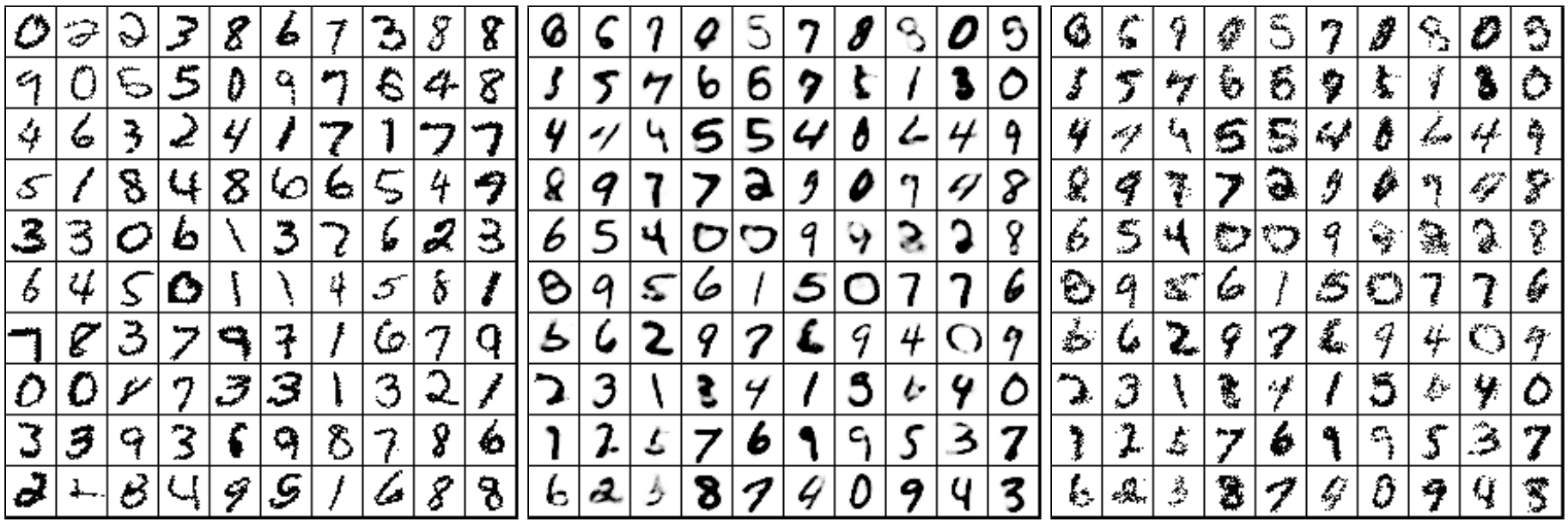
```
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.fc1 = nn.Linear(784, 400)
        self.fc21 = nn.Linear(400, 20)
        self.fc22 = nn.Linear(400, 20)
        self.fc3 = nn.Linear(20, 400)
        self.fc4 = nn.Linear(400, 784)

    def encode(self, x):
        h1 = F.relu(self.fc1(x))
        return self.fc21(h1), self.fc22(h1)

    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5*logvar)
        eps = torch.randn_like(std)
        return mu + eps*std

    def decode(self, z):
        h3 = F.relu(self.fc3(z))
        return torch.sigmoid(self.fc4(h3))

    def forward(self, x):
        mu, logvar = self.encode(x.view(-1, 784))
        z = self.reparameterize(mu, logvar)
        return self.decode(z), mu, logvar
```

VAE in PyTorch: Loss function

```
model = VAE().to(device)
optimizer = optim.Adam(model.parameters(), lr=1e-3)


# Reconstruction + KL divergence losses summed over all elements and batch
def loss_function(recon_x, x, mu, logvar):
    BCE = F.binary_cross_entropy(recon_x, x.view(-1, 784), reduction='sum')

    # see Appendix B from VAE paper:
    # Kingma and Welling. Auto-Encoding Variational Bayes. ICLR, 2014
    # https://arxiv.org/abs/1312.6114
    # 0.5 * sum(1 + log(sigma^2) - mu^2 - sigma^2)
    KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())

    return BCE + KLD
```

Experiments...

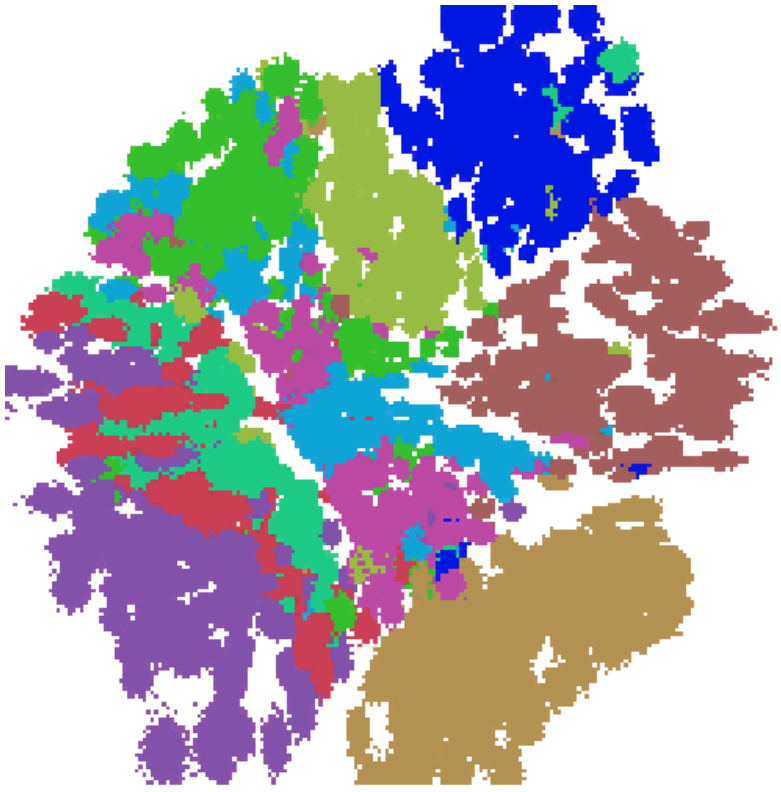
Generate new samples on MNIST dataset.



(a) Left: Training data. Middle: Model samples. Right: Sampled pixel probabilities

2D embedding

Project MNIST to 2D latent variable space



(b) 2D embedding.

Generate samples on NORB dataset

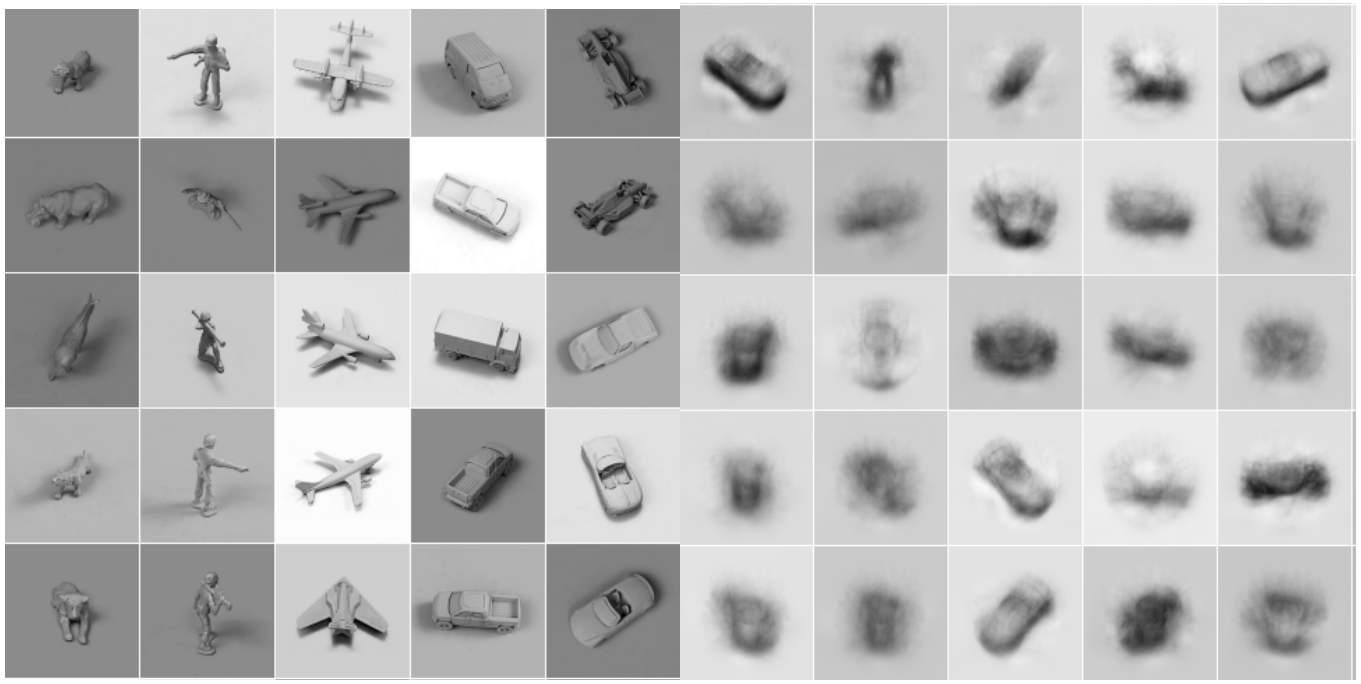


Figure: Left side shows samples from the training data and the right side shows the generated samples

Generate samples on CIFAR dataset

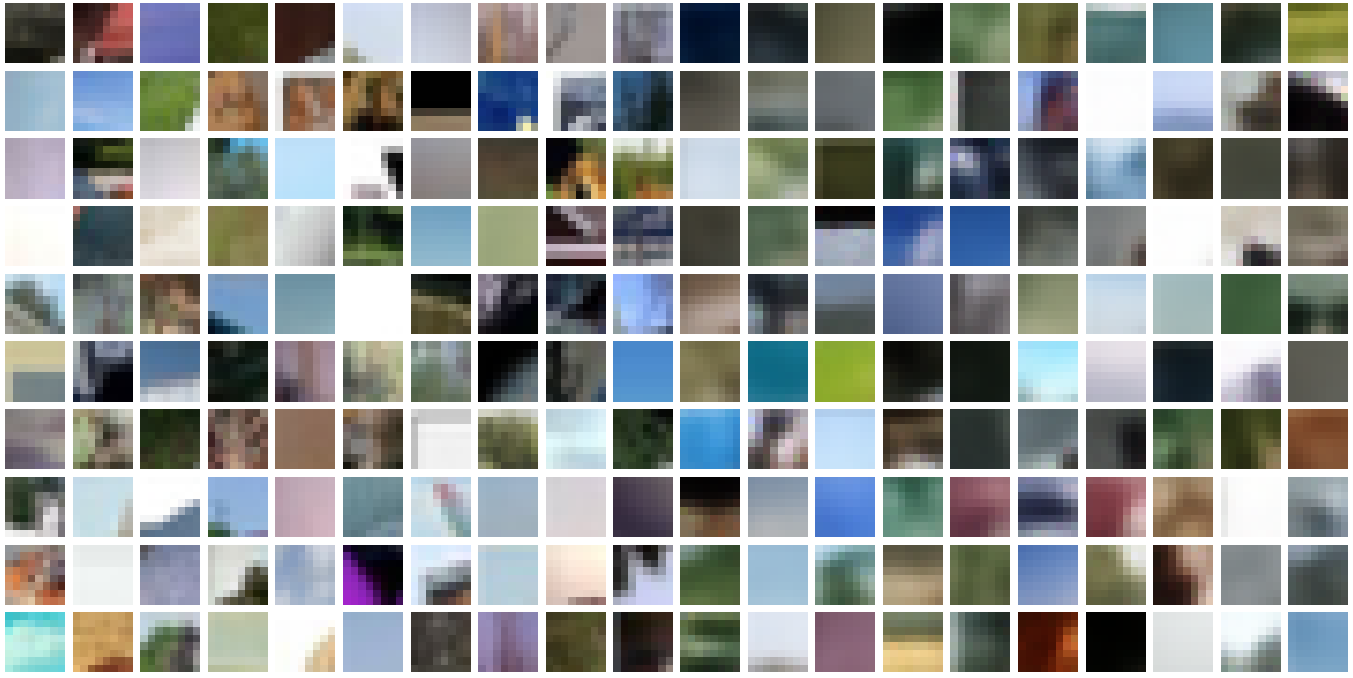
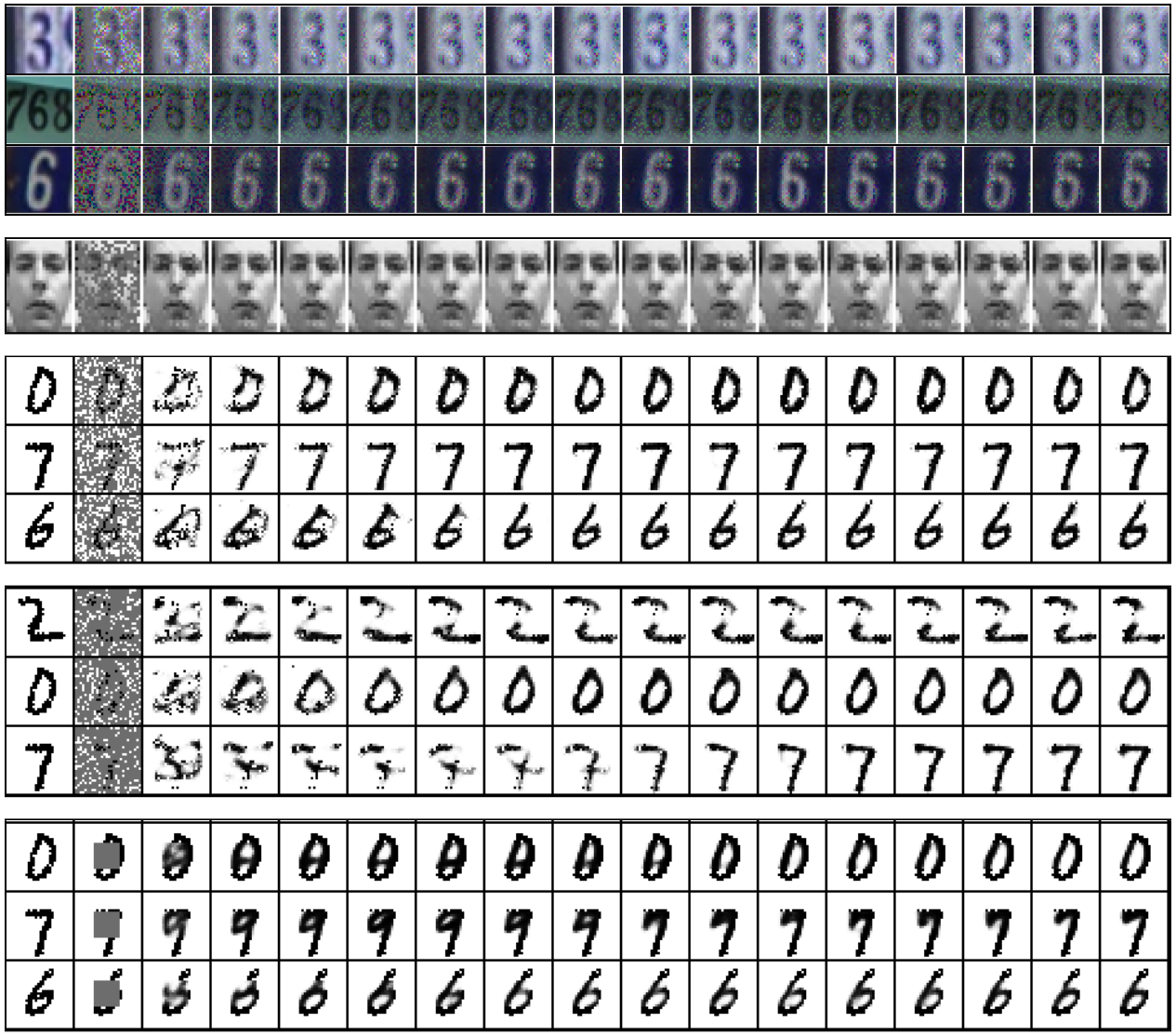


Figure: Left side shows samples from the training data and the right side shows the generated samples

Generate samples on Frey dataset

Figure: Left side shows samples from the training data and the right side shows the generated samples

Imputation results



1 Introduction

2 Preliminary

3 Variational Autoencoder

4 Extension of VAE

Extension of VAE

- **Conditional VAE [Sohn et al., 2015]**
- **Semi-supervised VAE [Kingma et al., 2014]**

Motivation

- We don't just want to generate new digits, but instead we want to add digits to an existing string of digits written by a single person.
- Similar to a truly practical problem in computer graphics called hole filling: given an existing image where a user has removed an unwanted object, the goal is to fill in the hole with plausible-looking pixels.

Objective

Given an input X and an output Y, we want to create a model $P(Y|X)$ which maximizes the probability of the ground truth.

We define the model by introducing a latent variable $z \sim \mathcal{N}(0, I)$, such that:

$$P(Y|X) = \mathcal{N}(f(z, X), \sigma^2 * I).$$

Where f is a deterministic function that we can learn from data.

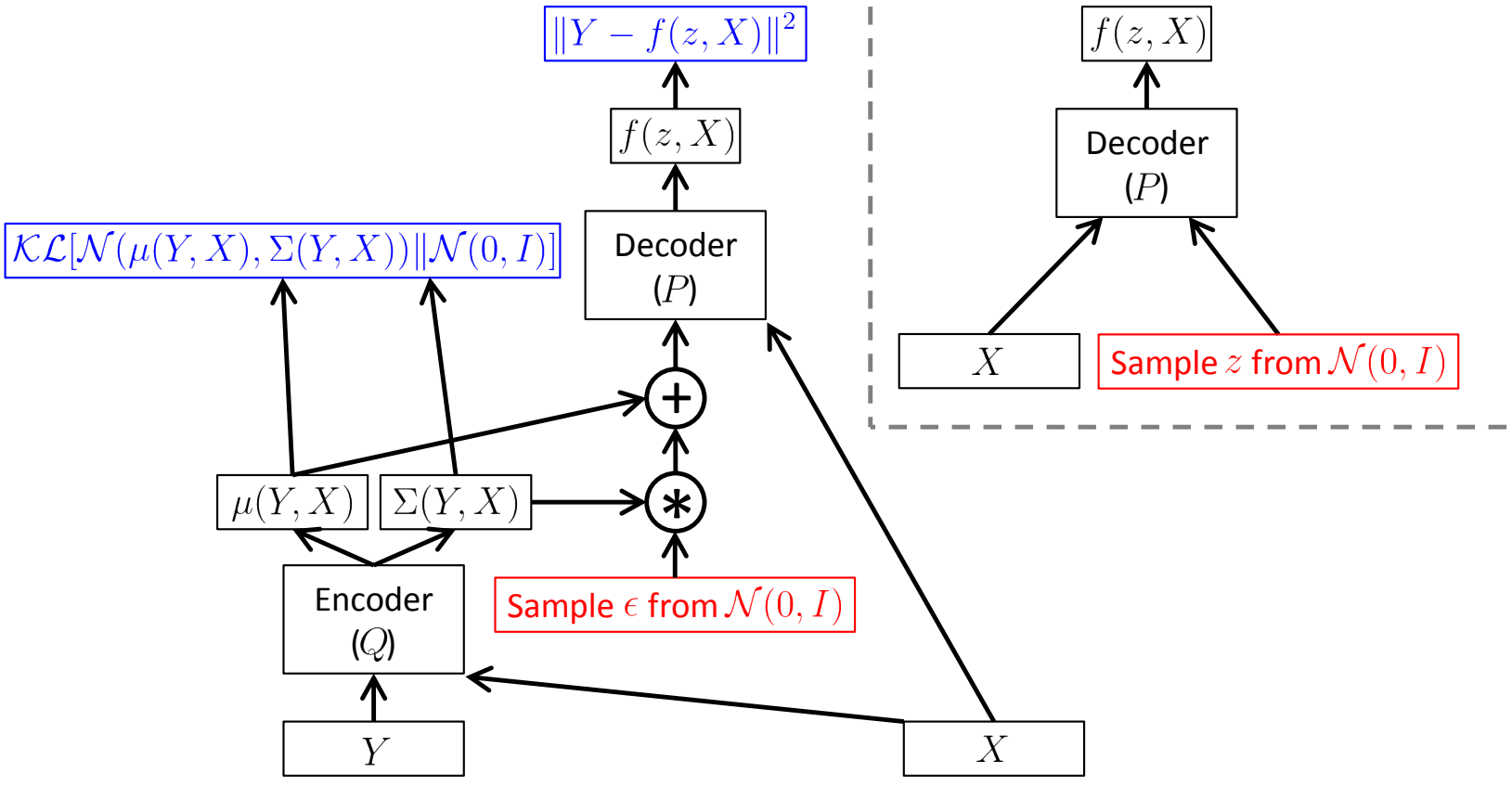
Rewrite some Equations

$$\mathcal{D}\left[Q(z|Y,X)\|P(z|Y,X)\right] = E_{z\sim Q(\cdot|Y,X)}\left[\log Q(z|Y,X) - \log P(z|Y,X)\right]$$

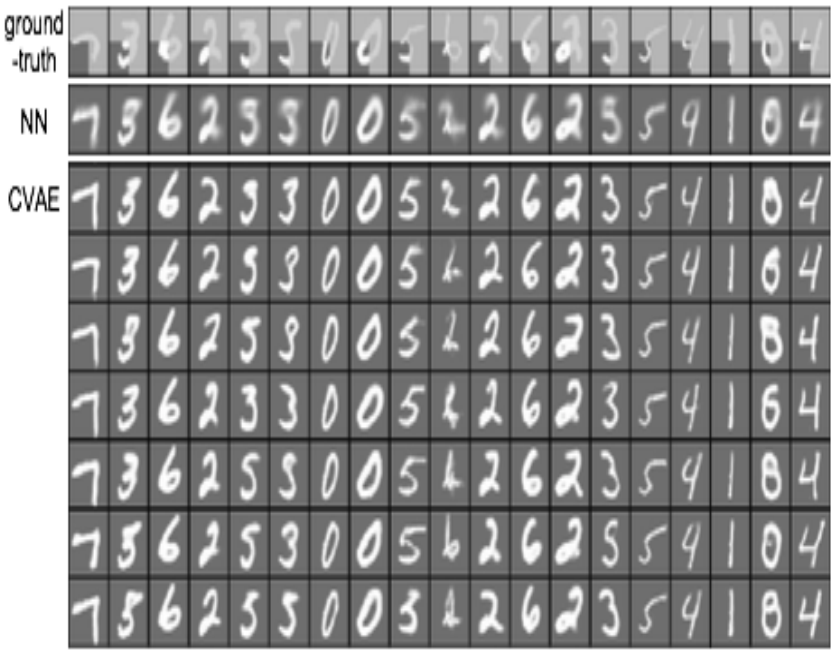
$$\mathcal{D}\left[Q(z|Y,X)\|P(z|Y,X)\right] =$$
$$E_{z\sim Q(\cdot|Y,X)}\left[\log Q(z|Y,X) - \log P(Y|z,X) - \log P(z|X)\right] + \log P(Y|X)$$

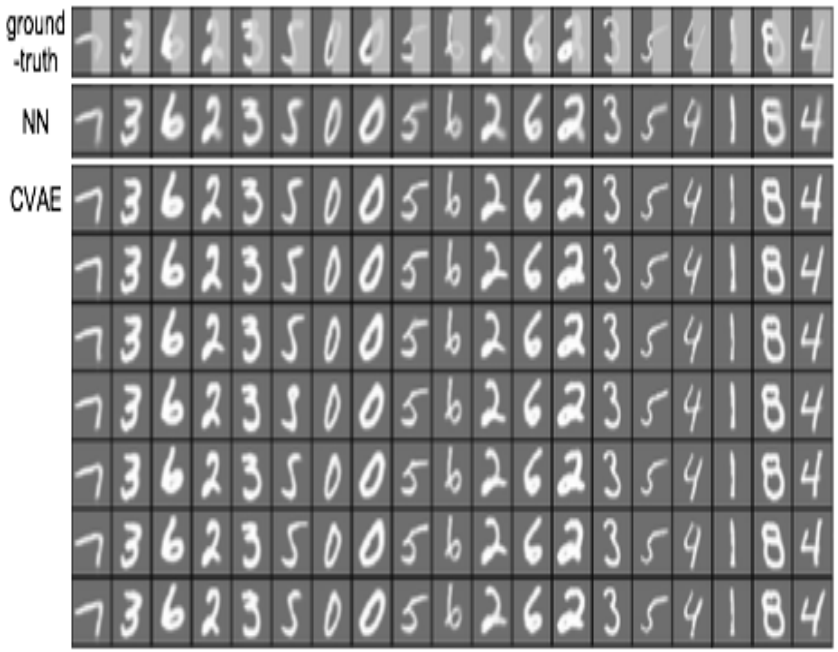
$$\log P(Y|X) - \mathcal{D}\left[Q(z|Y,X)\|P(z|Y,X)\right] =$$
$$E_{z\sim Q(\cdot|Y,X)}\left[\log P(Y|z,X)\right] - \mathcal{D}\left[Q(z|Y,X)\|P(z|X)\right]$$

Structure of conditional VAE

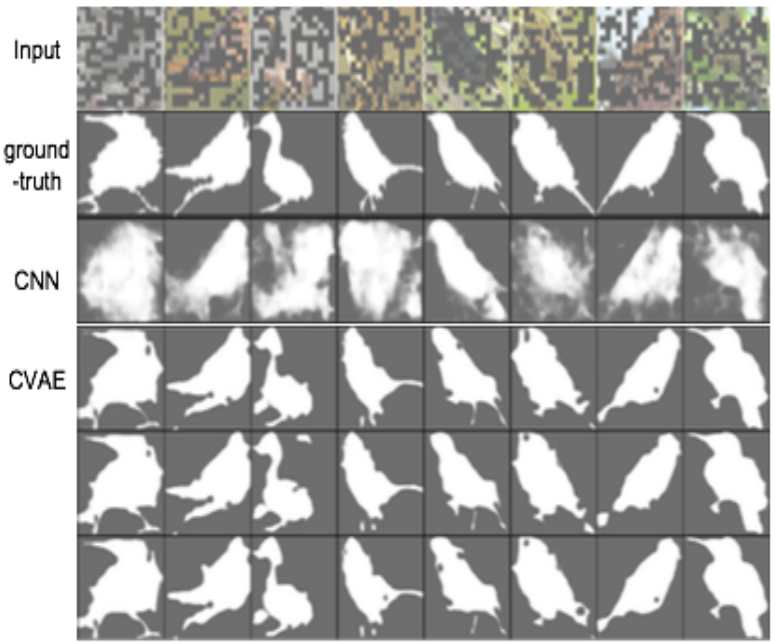


Experimental results

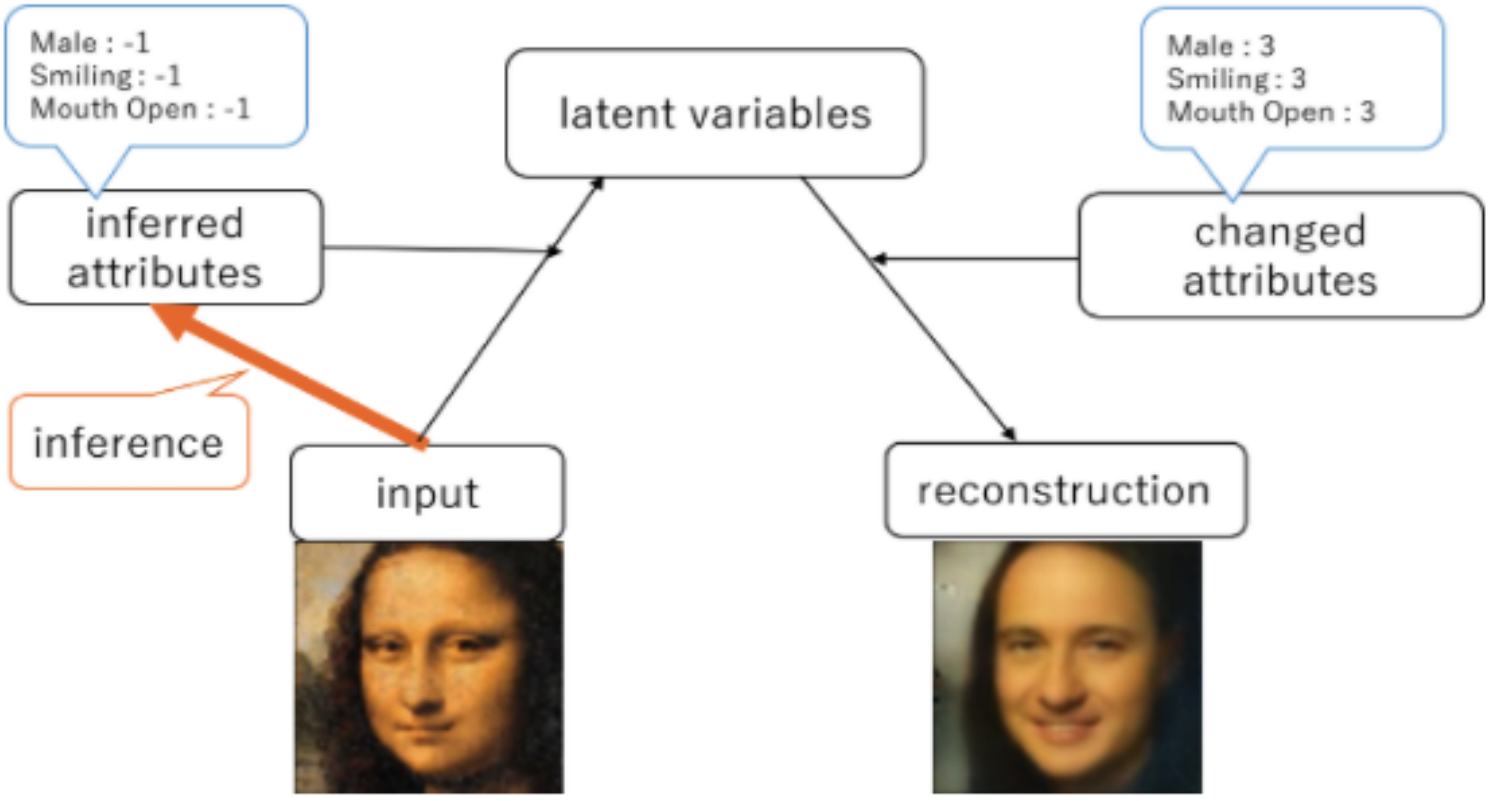




Experimental results



Demo



Demo : http://fvae.ail.tokyo/

Semi-supervised VAE [Kingma et al., 2014]

- We are faced with data that appear as pairs $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, with the i-th observation $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class label $y_i \in \{1, \ldots, L\}$.
- Observations will have corresponding latent variables, which we denote by $\mathbf{z}_i$
- In semi-supervised classification, only a subset of the observations have corresponding class labels; we refer to the empirical distribution over the labeled and unlabeled subsets as $\widetilde{p}_l(\mathbf{x}, y)$ and $\widetilde{p}_u(\mathbf{x})$, respectively.

Latent-feature discriminative model (M1):

M1 : A commonly used approach is to construct a model that provides an embedding or feature representation of the data. Using these features, a separate classifier is thereafter trained.

The generative model we use is:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta}), \tag{3}$$

where $f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta})$ is a suitable likelihood function (e.g., a Gaussian or Bernoulli distribution) whose probabilities are formed by a non-linear transformation, with parameters $\boldsymbol{\theta}$, of a set of latent variables $\mathbf{z}$.

Generative semi-supervised model (M2)

M2: We propose a probabilistic model that describes the data as being generated by a latent class variable y in addition to a continuous latent variable $\mathbf{z}$. The data is explained by the generative process:

$$p(y) = \text{Cat}(y|\boldsymbol{\pi}); \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|y, \mathbf{z}) = f(\mathbf{x}; y, \mathbf{z}, \boldsymbol{\theta}), \tag{4}$$

where $\text{Cat}(y|\boldsymbol{\pi})$ is the multinomial distribution, the class labels y are treated as latent variables if no class label is available and $\mathbf{z}$ are additional latent variables.

Stacked generative semi-supervised model (M1+M2)

M1+M2: We can combine these two approaches by first learning a new latent representation $\mathbf{z}_1$ using the generative model from M1, and subsequently learning a generative semi-supervised model M2, using embeddings from $\mathbf{z}_1$ instead of the raw data $\mathbf{x}$.

The result is a deep generative model with two layers of stochastic variables:

$$p_\theta(\mathbf{x}, y, \mathbf{z}_1, \mathbf{z}_2) = p(y)p(\mathbf{z}_2)p_\theta(\mathbf{z}_1|y, \mathbf{z}_2)p_\theta(\mathbf{x}|\mathbf{z}_1)$$

, where the priors $p(y)$ and $p(\mathbf{z}_2)$ equal those of y and $\mathbf{z}$ above, and both $p_\theta(\mathbf{z}_1|y, \mathbf{z}_2)$ and $p_\theta(\mathbf{x}|\mathbf{z}_1)$ are parameterized as deep neural networks.

Experimental results

Table: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	SVM	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

Conditional Generate

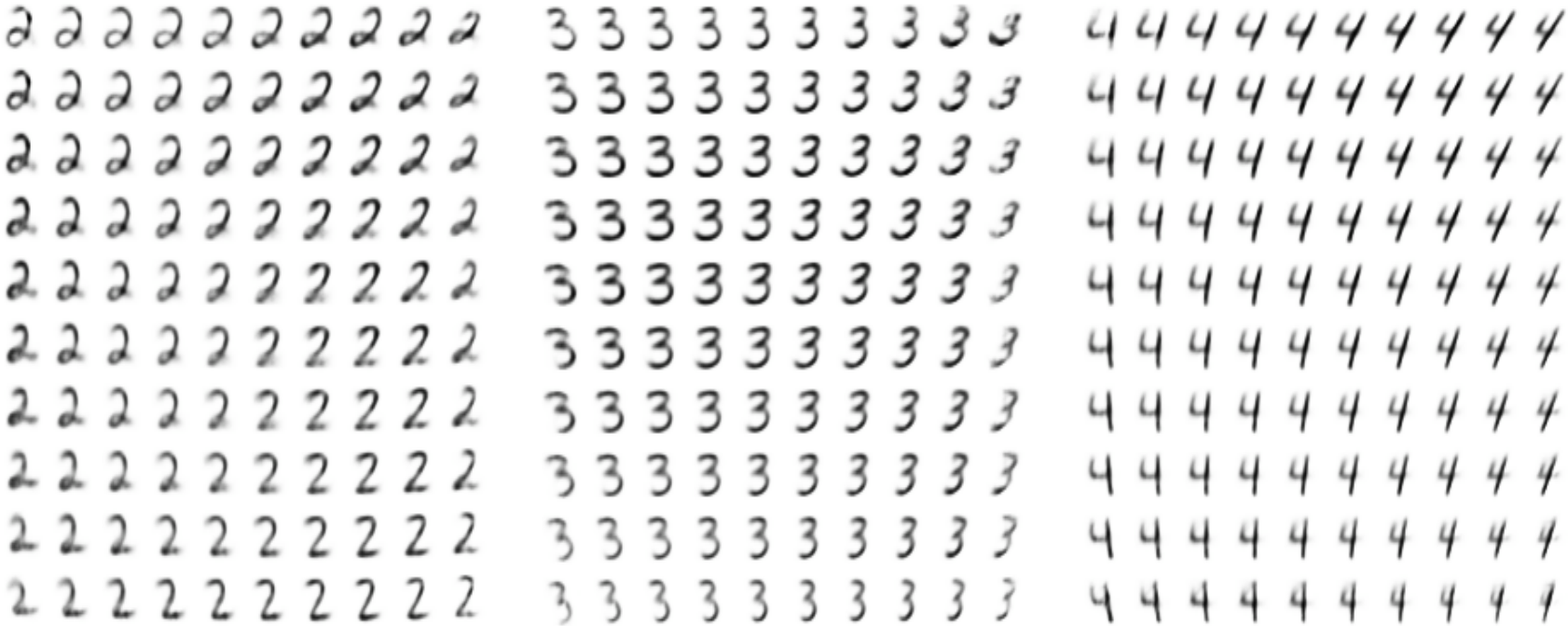
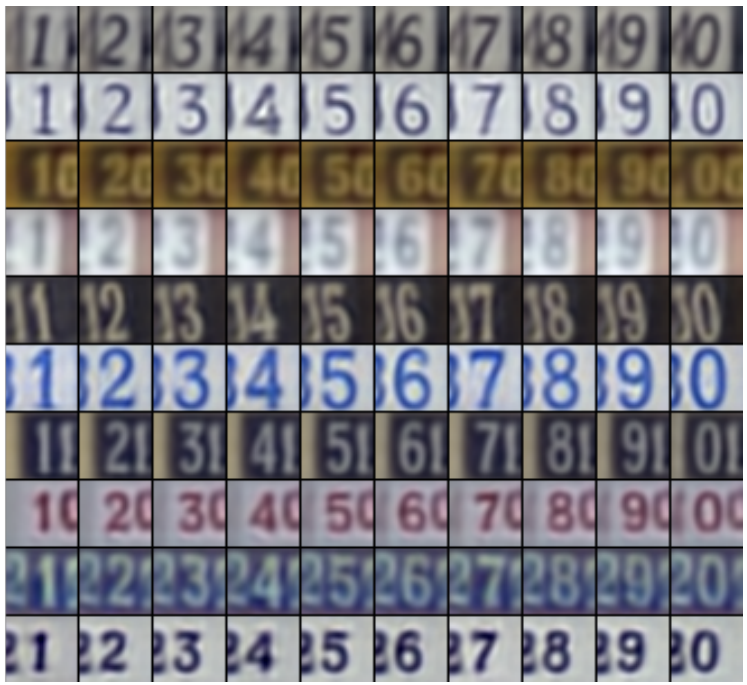


Figure: Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable $\mathbf{z}$

Conditional Generate



Classification

Table: Semi-supervised classification on the SVHN dataset with 1000 labels.

KNN	TSVM	M1+KNN	M1+TSVM	M1+M2
77.93 (± 0.08)	66.55 (± 0.10)	65.63 (± 0.15)	54.33 (± 0.11)	**36.02** (± 0.10)

Table: Semi-supervised classification on the NORB dataset with 1000 labels.

KNN	TSVM	M1+KNN	M1+TSVM
78.71 (± 0.02)	26.00 (± 0.06)	65.39 (± 0.09)	**18.79** (± 0.05)

Conclusion

- Autoencoder plays an nonlinear, nonconvex, scalable function in approximate Bayesian inference.
- The goal of VAE is to generate new sample.
- This presentation covers variational Bayesian inference, autoencoder, variational autoencoder, conditional VAE, semi-supervised VAE.

Discussion

The combination of deep learning (DL) and statistical learning (SL)...

- **DL as nonlinear function for SL**
  VAE [Kingma and Welling, 2013], Deep CCA [Andrew et al., 2013, Wang et al., 2015]
- **SL as regularization for DL**
  Spatial pyramid [He et al., 2014], Fisher layer [Simonyan et al., 2013, Tang et al., 2016],

Thanks.

Andrew, G., Arora, R., Bilmes, J. A., and Livescu, K. (2013). Deep canonical correlation analysis. In *ICML (3)*, pages 1247–1255.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer.

Jordan, A. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep fisher networks for large-scale image classification. In *Advances in neural information processing systems*, pages 163–171.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Tang, P., Wang, X., Shi, B., Bai, X., Liu, W., and Tu, Z. (2016). Deep fishernet for object classification. *arXiv preprint arXiv:1608.00182*.

Wang, W., Arora, R., Livescu, K., and Bilmes, J. (2015). On deep multi-view representation learning. In *Proc. of the 32st Int. Conf. Machine Learning (ICML 2015)*, pages 1083–1092.