Machine Learning II: Fractal 3

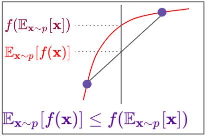
Rajendra Nagar

Assistant Professor
Department of Electircal Engineering
Indian Institute of Technology Jodhpur
http://home.iitj.ac.in/∼rn/

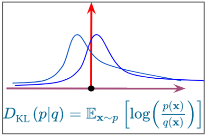
October 31, 2021

Variational Auto-encoders

Learn p_{model} from $\{\mathbf{x}\}_{i=1}^{n}$, s.t. $\mathrm{D}_{\mathrm{KL}}\left(p_{\text{model}}(x) \| p_{\text{data}}(x)\right)$ is minimized.

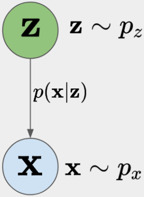


Jensen's Inequality



KL Divergence

Generative Modeling



$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$\min D_{\mathrm{KL}}\left(p_{\text{data}}(\mathbf{x}) \| p(\mathbf{x})\right)$$

$$\log(p(\mathbf{x})) = \log\left(\textstyle\sum_{\mathbf{z}\in\mathcal{Z}} p(\mathbf{x},\mathbf{z})\right)$$

Intractable: $k = 100$ requires 2^{100} '+' ops.

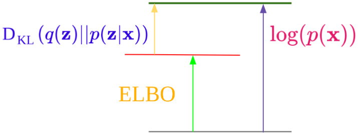
$$\log(p(\mathbf{x})) = \text{ELBO}$$

$$D_{\mathrm{KL}}\left(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})\right) = 0$$

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$$

$$q_\phi(\mathbf{z}|\mathbf{x})$$

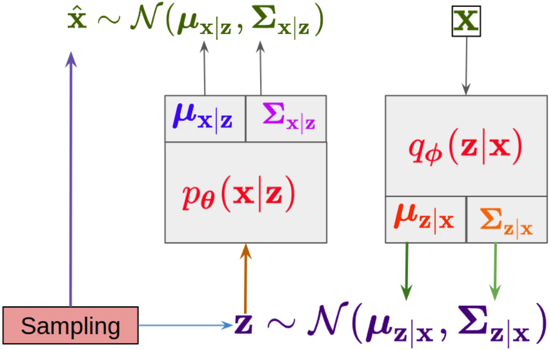
Tractable approximation to posterior distribution



$$\log(p(\mathbf{x})) = \text{ELBO} + D_{\mathrm{KL}}\left(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})\right)$$

Realization of Variational Auto-encoders using NNs

So far we have considered only abstract representations of $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, and $q(\mathbf{z})$. Let us assume that $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{z}})$, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}})$ and $q(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

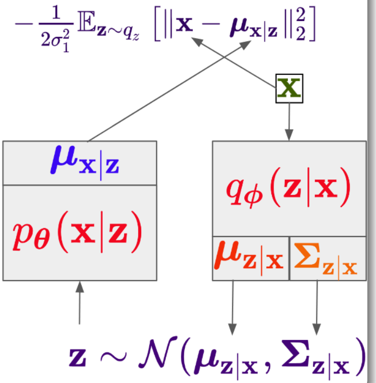


Realization of Variational Auto-encoders using NNs

Now, consider the ELBO function $\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right]$ and use $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}, \sigma_1^2\mathbf{I})$.

Then, the ELBO function becomes:

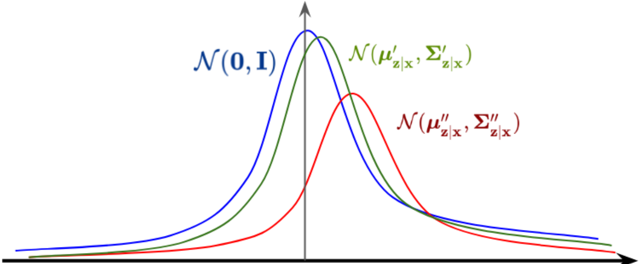
$$
\begin{aligned}
\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right] &= \mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}, \sigma_1^2\mathbf{I})\right)\right] \\
\log\left(\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}, \sigma_1^2\mathbf{I})\right) &= -\frac{1}{2\sigma_1^2}\|\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}\|_2^2 + \text{const.} \\
\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right] &= -\frac{1}{2\sigma_1^2}\mathbb{E}_{\mathbf{z}\sim q_z}\left[\|\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}\|_2^2\right] \\
&\Rightarrow \text{Reconstruction loss.}
\end{aligned}
$$



Realization of Variational Auto-encoders using NNs

Our goal is to maximize the ELBO function $\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right]$ such that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is as close as possible to $q(\mathbf{z})$. That is, we have to minimize $D_{\mathsf{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|q(\mathbf{z}))$. Now, use $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}})$ and $q(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

$$
\begin{aligned}
D_{\mathsf{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|q(\mathbf{z})) &= D_{\mathsf{KL}}(\mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}})\|\mathcal{N}(\mathbf{0}, \mathbf{I})) \\
&= \frac{1}{2}\sum_{i=1}^{k}\left(\mu_i^2 + \sigma_i^2 - 1 - \log_e(\sigma_i^2)\right)
\end{aligned}
$$

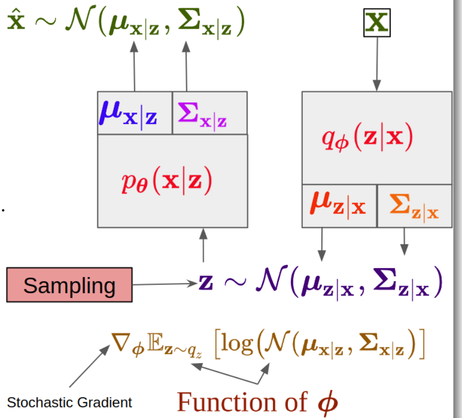


Sampling makes our life difficult

In order to find the optimal weights of the network, we have to find the gradient of the loss function with respect to the parameters ϕ and $\boldsymbol{\theta}$. Let us consider finding the gradient of ELBO function $\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right]$. We can easily find the gradient of ELBO w.r.t. $\boldsymbol{\theta}$ as

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right] = \mathbb{E}_{\mathbf{z}\sim q_z}\left[\nabla_{\boldsymbol{\theta}}\log\left(p(\mathbf{x}|\mathbf{z})\right)\right].$$

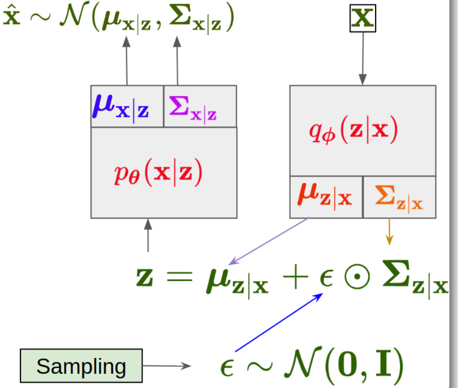
However, here we observe that $\mathbf{z}$ and $\log\left(p(\mathbf{x}|\mathbf{z})\right)$ are functions of ϕ. Therefore, finding $\nabla_{\phi}\mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p(\mathbf{x}|\mathbf{z})\right)\right]$ is not easy as we can not take gradient inside the expectation and this is stochastic gradient.



Re-parametrization Trick to rescue us

Let ϵ be a random variable such that $\epsilon \sim \mathcal{N}(0,1)$ and let μ and σ be two constants. Now, let us define a random variable z as $z = \mu + \epsilon\sigma$. Then, find the distribution of the random variable z.

$$
\begin{aligned}
\mathbb{E}[z] &= \mathbb{E}[\mu + \epsilon\sigma] \\
&= \mathbb{E}[\mu] + \mathbb{E}[\epsilon\sigma] \\
&= \mu + \sigma\mathbb{E}[\epsilon] \\
&= \mu \\
\text{var}(z) &= \text{var}(\mu + \epsilon\sigma) \\
&= \text{var}(\mu) + \text{var}(\epsilon\sigma) \\
&= 0 + \sigma^2\text{var}(\epsilon) \\
&= \sigma^2 \\
z &\sim \mathcal{N}(\mu, \sigma^2).
\end{aligned}
$$



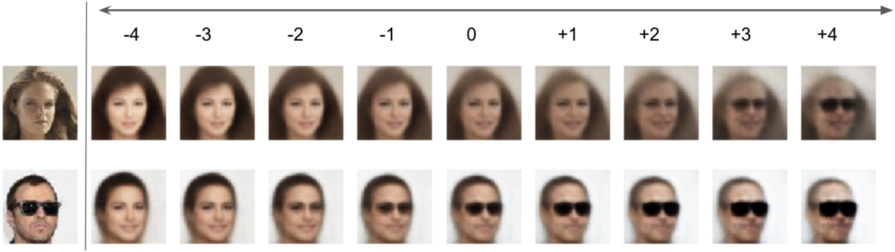
$$\hat{\mathbf{x}}$$

$$p_\theta(\mathbf{x}|\mathbf{z})$$

$$\mathbf{z} = \boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} + \epsilon \odot \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}}$$

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

Kingma and Welling 2014



Murphy 2021

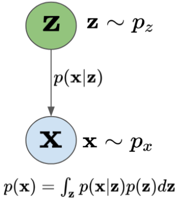
Previously on Generative Models

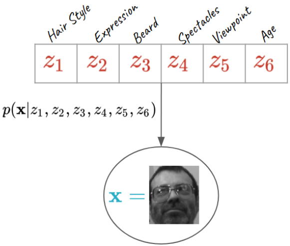
Problem Statement

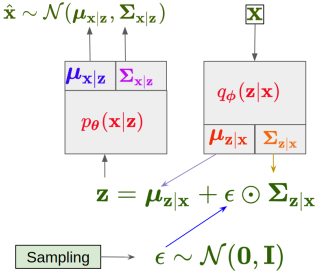
- Given a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ containing samples drawn from an unknown data distribution $p_{\text{data}}(\mathbf{x})$, learn a distribution $p_{\text{model}}(\mathbf{x})$ that is close as possible to the true distribution $p_{\text{data}}(\mathbf{x})$.
- Draw new samples from the distribution p_{data} by using its approximation p_{model}.

Variational Autoencoders

$$\max \mathbb{E}_{\mathbf{z}\sim q_z}\left[\log\left(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right)\right] \text{ such that } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = q(\mathbf{z}).$$







Today on Generative Models

Problem Statement

- Given a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ containing samples drawn from an unknown data distribution $p_{\text{data}}(\mathbf{x})$, learn a distribution $p_{\text{model}}(\mathbf{x})$ that is close as possible to the true distribution $p_{\text{data}}(\mathbf{x})$.
- Draw new points from p_{data} by using its approximation p_{model}.

Generative Adversarial Networks

- Our ultimate goal is to draw new samples from the distribution p_{data}.
- Is it necessary to learn p_{model} that approximate p_{data}?
- Can we draw points from p_{model} without explicitly learning p_{model}?
- Generative Adversarial Networks [Goodfellow et al. 2014] enables us to draw new samples from p_{data} without explicitly learning p_{model}.
- Hence, *circumvent maximization of the log-likelihood*.

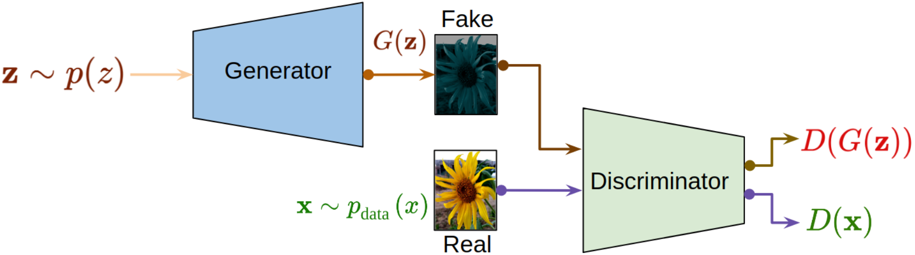
Generative Adversarial Networks

Modified Problem Statement: Given a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ containing samples drawn from an unknown data distribution $p_{\text{data}}(\mathbf{x})$, draw new samples from p_{data} with explicitly modeling p_{model}.

Solution:

- Let $\mathbf{z}$ be a latent variable with prior distribution $p(\mathbf{z})$.
- Draw a sample $\mathbf{z}$ from $p(\mathbf{z})$ and feed it to a Generator network G.
- Let us assume that the generator G models a distribution p_G.
- Then, the output $G(\mathbf{z})$ is a sample from the generator distribution p_G.
- We want $P_G = p_{\text{data}}$ after learning the weights of the network G.

Generative Adversarial Networks: A Minimax Game



- Discriminator wants $D(\mathbf{x}) = 1$ for real samples.
- Discriminator wants $D(G(\mathbf{z})) = 0$ for fake samples.
- Generator wants $D(G(\mathbf{z})) = 1$ for fake samples.

$$\min_G \max_D \left(\underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})]}_{\text{D wants } D(\mathbf{x})=1 \text{ for real point}} + \overbrace{\underbrace{\mathbb{E}_{\mathbf{z}\sim p(z)}[\log(1-D(G(\mathbf{z})))]}_{\text{D wants } D(G(\mathbf{z}))=0 \text{ for fake point}}}^{\text{G wants } D(G(\mathbf{z}))=1 \text{ for fake point}} \right)$$

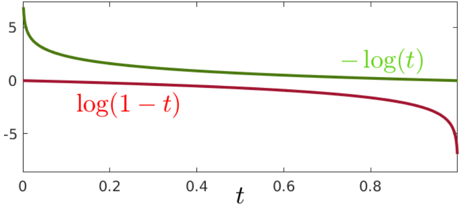
Generative Adversarial Networks: A Minimax Game

Jointly train generator G and discriminator D with a minimax game.

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(z)}[\log(1 - D(G(\mathbf{z})))]\right)$$

1. $D \leftarrow D + \alpha_d \nabla_D f(D, G)$
2. $G \leftarrow G - \alpha_d \nabla_G f(D, G)$

Since at the start of the training generator is very poor the discriminator can easily outperform the generator. Hence, $D(G(\mathbf{z})) \approx 0 \Rightarrow \log(1 - D(G(\mathbf{z}))) \approx 0$. Therefore, the gradient will be almost zero initially.



To overcome this vanishing gradient issue, we can minimize the function $-\log(D(G(\mathbf{z})))$ instead of $-\log(1 - D(G(\mathbf{z})))$.

Generative Adversarial Networks: Optimality Analysis

Jointly train generator G and discriminator D with a minimax game.

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(z)}[\log(1 - D(G(\mathbf{z})))]\right)$$

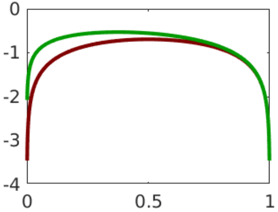
We want to verify that global minimum of this game occurs at $p_G = p_{\text{data}}$.

$$\begin{aligned}
& \min_G \max_D \left(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(z)}[\log(1 - D(G(\mathbf{z})))]\right) \\
= & \min_G \max_D \left(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_G}[\log(1 - D(\mathbf{x}))]\right) \\
= & \min_G \max_D \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x})[\log D(\mathbf{x})] + p_G(\mathbf{x})[\log(1 - D(\mathbf{x}))]\right) d\mathbf{x} \\
= & \min_G \int_{\mathbf{x}} \max_D \left(p_{\text{data}}(\mathbf{x})[\log D(\mathbf{x})] + p_G(\mathbf{x})[\log(1 - D(\mathbf{x}))]\right) d\mathbf{x}
\end{aligned}$$

Generative Adversarial Networks: Optimality Analysis

Consider the problem of finding the point of maximum of the function $f(t) = a\log(t) + b\log(1-t)$. Here $t, a, b \in [0,1]$.

$$\begin{aligned} f(t) &= a\log(t) + b\log(1-t) \\ \frac{df}{dt} &= \frac{a}{t} - \frac{b}{1-t} = 0 \\ t &= \frac{a}{a+b}. \end{aligned}$$



$a = \frac{1}{2}, b = \frac{1}{2}$, $a = \frac{3}{10}, b = \frac{1}{2}$

Now, consider the main problem :

$$\max_D \left(p_{\text{data}}(\mathbf{x})[\log D(\mathbf{x})] + p_G(\mathbf{x})[\log(1 - D(\mathbf{x}))]\right)$$

The optimal discriminator $D^\star(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$