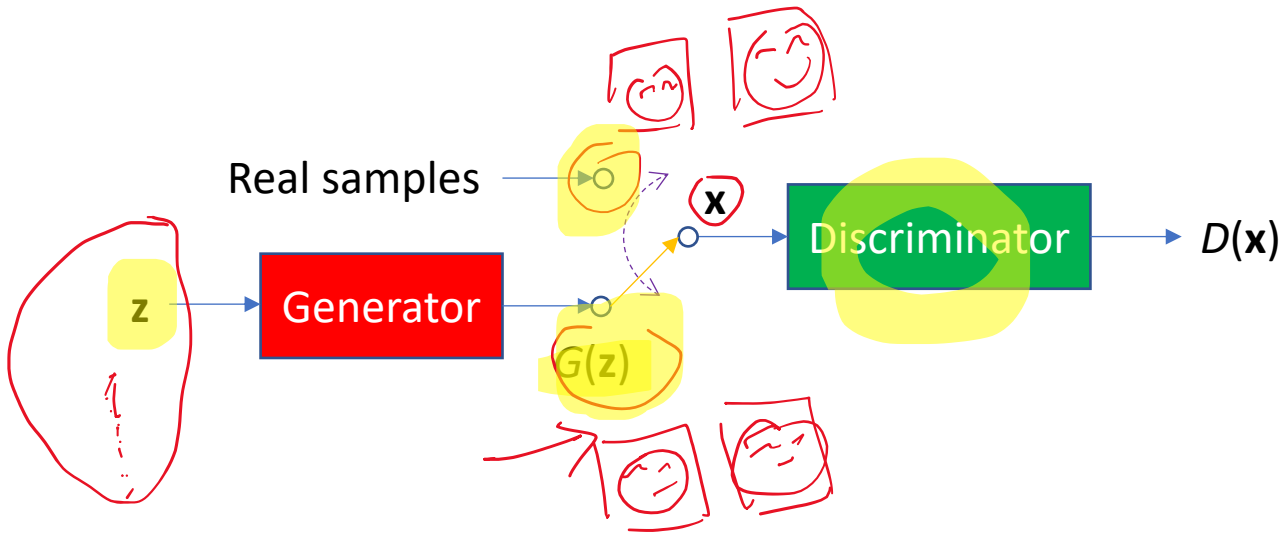
Pattern Recognition, Neural Networks and Deep Learning (7CCSMPNN)

Tutorial 4: Solutions

Q1. Diminished gradient is an issue when training Generative Adversarial Networks (GANs). In the literature, when training the Generator, $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log(D(G(\mathbf{z}))]$ is recommended to be an alternative cost function.

a. What is the advantage of using this alternative cost function over the original one, i.e., $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$?

b. Write the Pseudo code for the training of GAN with this alternative cost function.

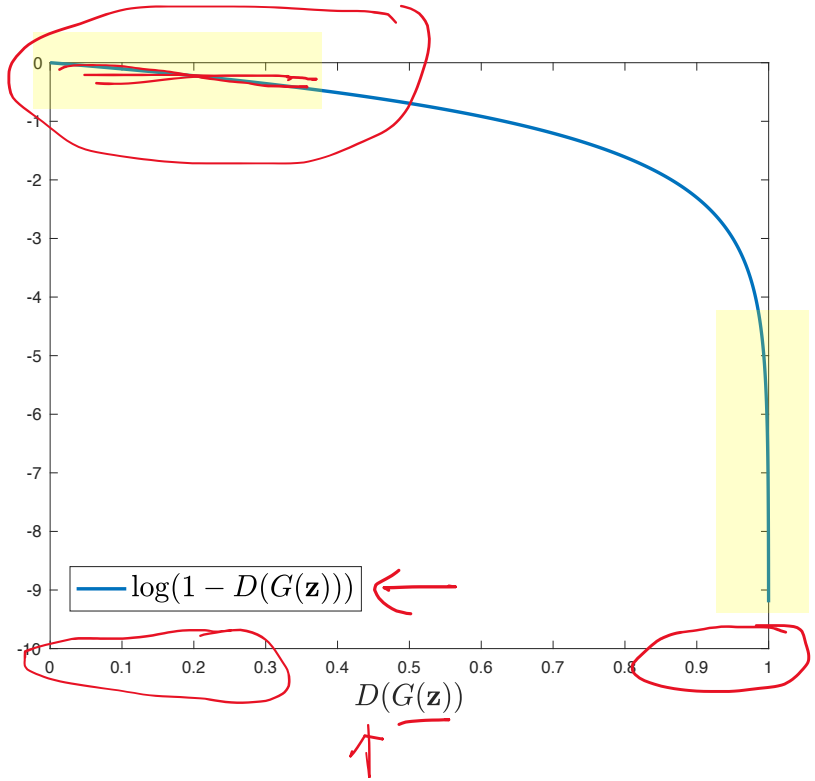


a. What is the advantage of using this alternative cost function over the original one, i.e., $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$?

Original Cost: $V(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\underbrace{\log D(\mathbf{x})}_{\text{For real samples}}] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\underbrace{\log(1 - D(G(\mathbf{z})))}_{\text{For fake samples}}]$

Diminished gradient: $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(\mathbf{z}^{(i)}))\right)$

Flat: When the Generator does NOT work well



Steep: When the Generator works very well

Alternative Cost:

$$V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\underbrace{\log D(\mathbf{x})}_{\text{For real samples}}] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\underbrace{-\log(D(G(\mathbf{z}))}_{\text{For fake samples}}]$$

This cost $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log(D(G(\mathbf{z}))]$ helps alleviate the gradient vanishing problem.

The figure below shows the original and alternative costs.

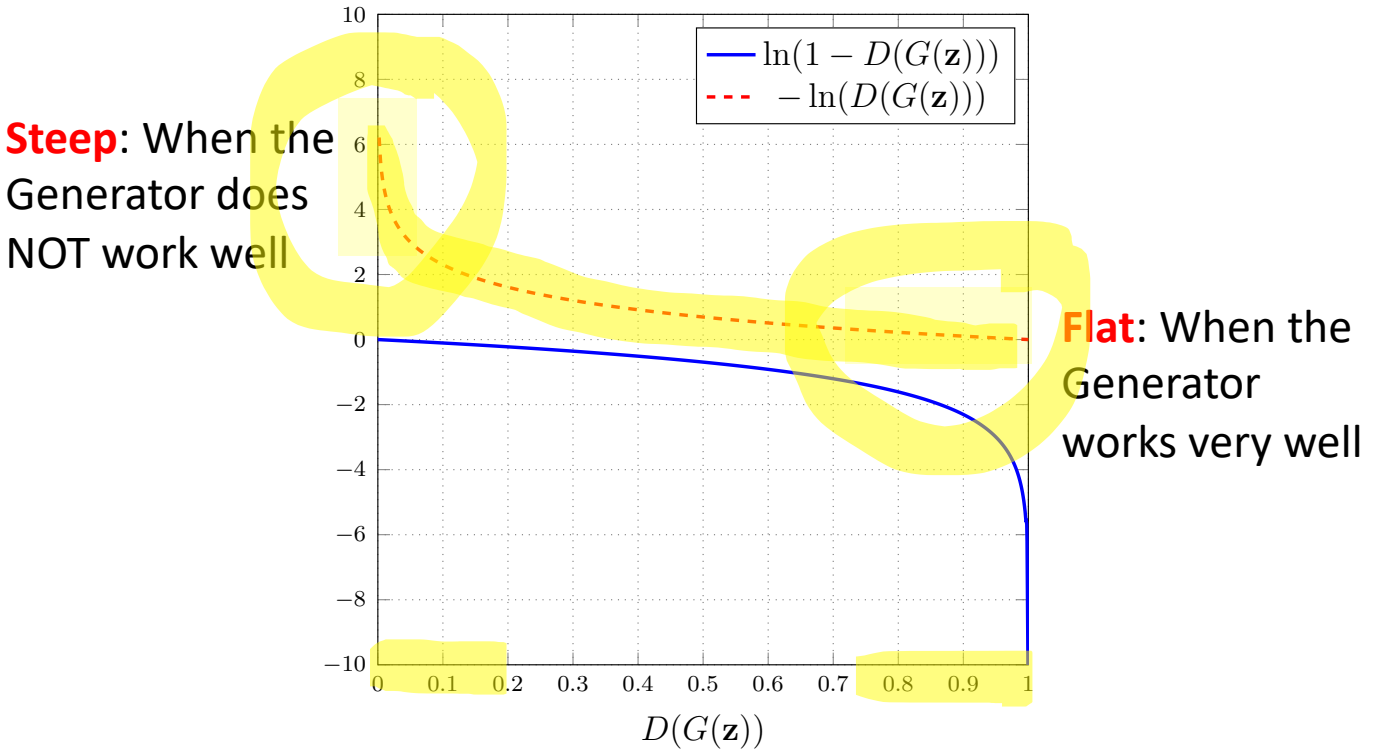


Figure 1: Original ($\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$) and alternative ($\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log(D(G(\mathbf{z}))]$) costs.

Below are some observations:

- The Generator $G(\mathbf{z})$ is to be trained that it is able to fool the Discriminator $D(\mathbf{x})$. So that $D(G(\mathbf{z}))$ should produce 1 (when the sample is fake).
- In the original cost, it can be seen that the slope for $\log(1 - D(G(\mathbf{z})))$ when $D(G(\mathbf{z})) \approx 0$ is nearly flat. In that region, i.e., $D(G(\mathbf{z})) \approx 0$ needs more attention as the Generator does work well to fool the Discriminator, especially in the initial training stage. However, the nearly-flat slope implies the learning is very slow.
- When considering the alternative cost, $\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[-\log(D(G(\mathbf{z}))]$, it can been seen from the figure that its slope is much obvious in the region $D(G(\mathbf{z})) \approx 0$. Consequently, the gradient vanishing issue is alleviated.
- Both the original and alternative cost function have the same trend, i.e., monotonic decreasing.

b. Write the Pseudo code for the training of GAN with this alternative cost function.

Alternative Cost: $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log(D(G(\mathbf{z})))]$

For real samples | For fake samples

Algorithm 1: (**With Alternative Cost**) Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

for *number of training iterations* **do**

  for k *steps* **do**

  - Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(z)$.
  - Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{z})$.
  - Update the Discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D(\mathbf{x}^{(i)}) \right) - \log \left(D(G(\mathbf{z}^{(i)})) \right) \right].$$

a = ?

  end

  - Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
  - Update the Generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [- \log(D(G(\mathbf{z}^{(i)}))] \Big).$$

b = ?

end

Remark: The gradient-based updates can use any standard gradient-based learning rule.

Q2. The training of Generative Adversarial Networks (GANs) can be formulated as an optimisation problem shown below:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))].$$

a. Find the optimal Discriminator $D(\mathbf{x})$, denoted as $D^*(\mathbf{x})$. Note that $\mathbf{x}$ denotes the sample taken by the Discriminator $D(\mathbf{x})$, which could represent the real or generated sample.

b. Find the optimal $V(D,G)$.

a. Find the optimal Discriminator $D(\mathbf{x})$, denoted as $D^*(\mathbf{x})$. Note that $\mathbf{x}$ denotes the sample taken by the Discriminator $D(\mathbf{x})$, which could represent the real or generated sample.

Given the minimax game as below:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Recall that the distribution for:

- Real samples: $p_{data}(\mathbf{x})$
- Fake (generated) samples: $p_g(\mathbf{x})$

When $\mathbf{x}$ is a continuous variable, the cost can be written as below:

$$\begin{aligned} V(D,G) &= \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[\log(1 - D(\mathbf{x}))] \\ &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbf{x}} \Big(p_{data}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) \Big) d\mathbf{x} \end{aligned}$$

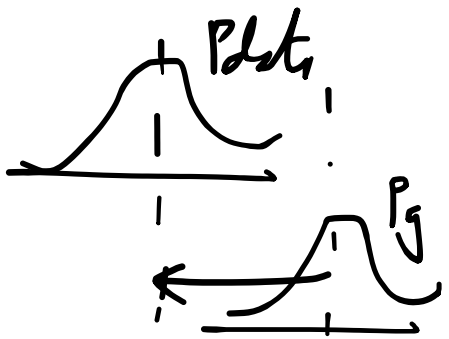
The optimal $V(D,G)$ is obtained when:

What condition ?

$$\frac{d}{dD(\mathbf{x})}\Big(p_{data}(\mathbf{x})\log D(\mathbf{x}) + p_g(\mathbf{x})\log(1-D(\mathbf{x}))\Big) = 0.$$

Remark: The $\log(\cdot)$ operator is referred to $\ln(\cdot)$ (Natural Logarithm).
In the following, we replace $\log(\cdot)$ by $\ln(\cdot)$.

$$\frac{d}{dD(\mathbf{x})}\Big(p_{data}(\mathbf{x})\ln D(\mathbf{x}) + p_g(\mathbf{x})\ln(1-D(\mathbf{x}))\Big) = p_{data}(\mathbf{x})\frac{1}{D(\mathbf{x})} - p_g(\mathbf{x})\frac{1}{1-D(\mathbf{x})}$$
$$= \frac{p_{data}(\mathbf{x})(1-D(\mathbf{x})) - p_g(\mathbf{x})D(\mathbf{x})}{D(\mathbf{x})(1-D(\mathbf{x}))}$$



Setting $\frac{d}{dD(\mathbf{x})}\Big(p_{data}(\mathbf{x})\log D(\mathbf{x}) + p_g(\mathbf{x})\log(1-D(\mathbf{x}))\Big) = 0$, we obtain the best Discriminator as

$$D(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}.$$

$D(\mathbf{x}) = ?$

The ideal distribution $p_g(\mathbf{x})$ is $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$. It happens when the Generator reproduces the real data distribution. Consequently, we have the optimal Discriminator as

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{1}{2}.$$

?

b. Find the optimal $V(D, G)$.

When both Generator and Discriminator are optimal (denoted as D^* and G^* below), we have

$$
\begin{aligned}
V(D^*, G^*) &= \int_{\mathbf{x}} \Big(p_{data}(\mathbf{x}) \ln D^*(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D^*(\mathbf{x})) \Big) d\mathbf{x} \\
&= \int_{\mathbf{x}} \left(p_{data}(\mathbf{x}) \log \left(\frac{1}{2} \right) + p_g(\mathbf{x}) \log \left(1 - \frac{1}{2} \right) \right) d\mathbf{x} \\
&= \log \left(\frac{1}{2} \right) \int_{\mathbf{x}} p_{data}(\mathbf{x}) d\mathbf{x} + \log \left(\frac{1}{2} \right) \int_{\mathbf{x}} p_g(\mathbf{x}) d\mathbf{x} \\
&= 2 \log \left(\frac{1}{2} \right) \\
&= -2 \log(2).
\end{aligned}
$$

?

Q4. When training a Generative Adversarial Network (GAN), we consider a dataset of real samples denoted as $\mathbf{X}_{real} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and the generated samples as $\mathbf{X}_{fake} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, where $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \tilde{\mathbf{x}}_1 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \tilde{\mathbf{x}}_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$.

The Discriminator is given as

$$D(\mathbf{x}) = \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}$$

where θ_{d_1} and θ_{d_2} are parameters of Discriminator, and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Given $\theta_{d_1} = 0.1$ and $\theta_{d_2} = 0.2$. Each sample from the (real and fake) dataset has equal probability to be selected.

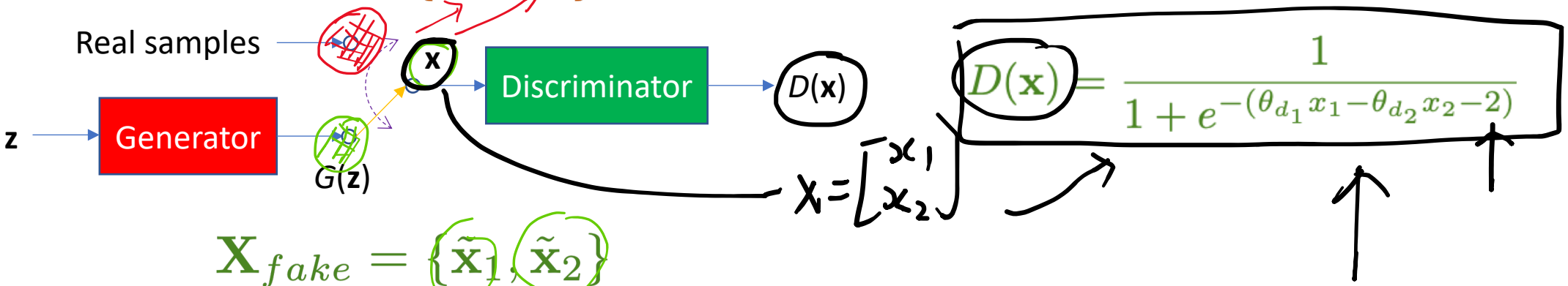
a. Given the datasets $\mathbf{X}_{real}$ and $\mathbf{X}_{fake}$, compute $V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$.

b. Assuming all real and fake samples are selected into the minibatch and $k = 1$ for GAN training, compute $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$ and determine the updated θ_{d_1} and θ_{d_2} for the next iteration using the learning rate $\eta = 0.02$.

a. Given the datasets $\mathbf{X}_{real}$ and $\mathbf{X}_{fake}$, compute $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$.

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\mathbf{X}_{real} = \{\mathbf{x}_1, \mathbf{x}_2\}$$



$$D(\mathbf{x}) = \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}$$

$$\mathbf{X}_{fake} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$$

$$\tilde{\mathbf{x}}_1 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \tilde{\mathbf{x}}_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Cost: $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$

a. Given the datasets $\mathbf{X}_{real}$ and $\mathbf{X}_{fake}$, compute $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$.

Cost: $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$

Considering the first term $\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})]$, its expectation is:

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] = \sum_{i=1}^{2} \ln D(\mathbf{x}_i) p_{data}(\mathbf{x}_i)$$

$$= \ln D(\mathbf{x}_1) p_{data}(\mathbf{x}_1) + \ln D(\mathbf{x}_2) p_{data}(\mathbf{x}_2)$$

$$= \ln D(\mathbf{x}_1)\frac{1}{2} + \ln D(\mathbf{x}_2)\frac{1}{2}$$

$$= \underbrace{0.5 \ln\left(\frac{1}{1+e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}\right)}_{\text{For sample } \mathbf{x}_1 \text{ in } \mathbf{X}_{real}} + \underbrace{0.5 \ln\left(\frac{1}{1+e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}\right)}_{\text{For sample } \mathbf{x}_2 \text{ in } \mathbf{X}_{real}}$$

$$= 0.5 \ln\left(\frac{1}{1+e^{-(0.1\times 1 - 0.2\times 2 - 2)}}\right) + 0.5 \ln\left(\frac{1}{1+e^{-(0.1\times 3 - 0.2\times 4 - 2)}}\right)$$

$$= -2.4872$$

a. Given the datasets $\mathbf{X}_{real}$ and $\mathbf{X}_{fake}$, compute $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$.

$\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2$

Considering the second term $\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\ln(1 - D(\mathbf{G}(\mathbf{z})))]$, its expectation is:

$$
\begin{aligned}
\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\ln(1 - D(\mathbf{G}(\mathbf{z})))] &= \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\ln(1 - D(\tilde{\mathbf{x}}_i))] \\
&= \ln(1 - D(\tilde{\mathbf{x}}_1))p_g(\tilde{\mathbf{x}}_1) + \ln(1 - D(\tilde{\mathbf{x}}_2))p_g(\tilde{\mathbf{x}}_2) \\
&= \ln(1 - D(\tilde{\mathbf{x}}_1))\frac{1}{2} + \ln(1 - D(\tilde{\mathbf{x}}_2))\frac{1}{2} \\
&= \underbrace{0.5 \ln\left(1 - \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}\right)}_{\text{For sample } \tilde{\mathbf{x}}_1 \text{ in } \mathbf{X}_{fake}} \\
&+ \underbrace{0.5 \ln\left(1 - \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}\right)}_{\text{For sample } \tilde{\mathbf{x}}_2 \text{ in } \mathbf{X}_{fake}} \\
&= 0.5 \ln\left(1 - \frac{1}{1 + e^{-(0.1\times 5 - 0.2\times 6 - 2)}}\right) \\
&+ 0.5 \ln\left(1 - \frac{1}{1 + e^{-(0.1\times 7 - 0.2\times 8 - 2)}}\right) \\
&= -0.0593
\end{aligned}
$$

a. Given the datasets $\mathbf{X}_{real}$ and $\mathbf{X}_{fake}$, compute $V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))]$.

Hence,

$$\begin{aligned} V(D,G) &= \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\ln D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\ln(1 - D(G(\mathbf{z})))] \\ &= -2.4872 - 0.0593 \\ &= -2.5465 \end{aligned}$$

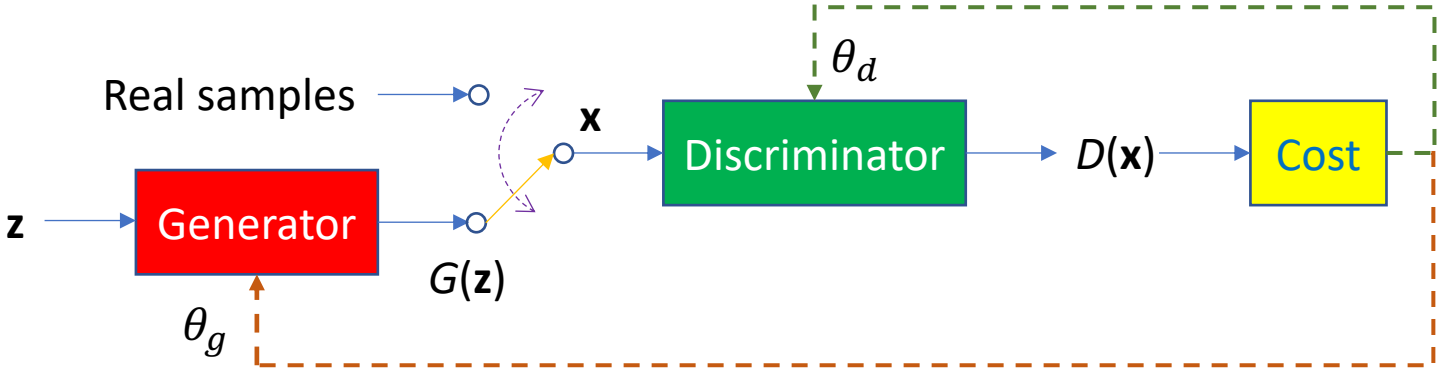
b. Assuming all real and fake samples are selected into the minibatch and $k = 1$ for GAN training, compute $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$ and determine the updated θ_{d_1} and θ_{d_2} for the next iteration using the learning rate $\eta = 0.02$.

b. Assuming all real and fake samples are selected into the minibatch and $k = 1$ for GAN training, compute $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\ln D(\mathbf{x}^{(i)}) + \ln \left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$ and determine the updated θ_{d_1} and θ_{d_2} for the next iteration using the learning rate $\eta = 0.02$.

When training a Generative Adversarial Network (GAN), we consider a dataset of real samples denoted as $\mathbf{X}_{real} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and the generated samples as $\mathbf{X}_{fake} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, where $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\tilde{\mathbf{x}}_1 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$, $\tilde{\mathbf{x}}_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$.

$$D(\mathbf{x}) = \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}$$

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(\mathbf{x}^{(i)}) + \log \left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$$



$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(\mathbf{z}^{(i)}))\right)$$

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

for *number of training iterations* **do**
  for k *steps* **do**
    - Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(z)$.
    - Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{z})$.
    - Update the Discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D(\mathbf{x}^{(i)}) \right) + \log \left(1 - D(G(\mathbf{z}^{(i)})) \right) \right].$$

  end
  - Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
  - Update the Generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(\mathbf{z}^{(i)}))\right).$$

end

Remark: The gradient-based updates can use any standard gradient-based learning rule.

$$\nabla_y f = \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \frac{\partial f}{\partial y_2} \end{bmatrix} \qquad f(y) = y_1^2 + y_2^2$$

$$= \begin{bmatrix} 2y_1 \\ 2y_2 \end{bmatrix}$$

When training a Generative Adversarial Network (GAN), we consider a dataset of real samples denoted as $\mathbf{X}_{real} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and the generated samples as $\mathbf{X}_{fake} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, where $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\tilde{\mathbf{x}}_1 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$, $\tilde{\mathbf{x}}_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$.

Denote the generated samples as $\mathbf{x}_{fake}^{(i)} = \mathbf{z}^{(i)}$.

$m=2$

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{bmatrix} \frac{\partial}{\partial \theta_{d_1}} \left(\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right) \\ \frac{\partial}{\partial \theta_{d_2}} \left(\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right) \end{bmatrix}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{bmatrix} \frac{\partial}{\partial \theta_{d_1}} \ln D\left(\mathbf{x}^{(i)}\right) + \frac{\partial}{\partial \theta_{d_1}} \ln\left(1 - D(\mathbf{x}_{fake}^{(i)})\right) \\ \frac{\partial}{\partial \theta_{d_2}} \ln D\left(\mathbf{x}^{(i)}\right) + \frac{\partial}{\partial \theta_{d_2}} \ln\left(1 - D(\mathbf{x}_{fake}^{(i)})\right) \end{bmatrix}$$

$\beta_1^{(i)}$ $\beta_2^{(i)}$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{bmatrix} \alpha_1^{(i)} + \beta_1^{(i)} \\ \alpha_2^{(i)} + \beta_2^{(i)} \end{bmatrix}$$

$\alpha_2^{(i)}$

$i=1$

$$= \frac{1}{2} \begin{bmatrix} \alpha_1^{(1)} + \beta_1^{(1)} \\ \alpha_2^{(1)} + \beta_2^{(1)} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \alpha_1^{(2)} + \beta_1^{(2)} \\ \alpha_2^{(2)} + \beta_2^{(2)} \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} 0.9089 - 0.3149 \\ -1.8178 + 0.37784 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2.7724 - 0.3651 \\ -3.6966 + 0.4172 \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} 0.5940 \\ -1.4399 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2.4074 \\ -3.2793 \end{bmatrix} = \begin{bmatrix} 1.5007 \\ -2.3596 \end{bmatrix}$$

$i=1$: x_1, $\tilde{x}_1$

$i=2$: x_2, $\tilde{x}_2$

$$D(\mathbf{x}) = \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}$$

$$\alpha_1^{(i)} = \frac{\partial}{\partial \theta_{d_1}} \ln D\left(\mathbf{x}^{(i)}\right) \qquad i = 1, 2$$

$$D(\mathbf{x}) = \frac{1}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}} = \frac{1}{f}$$

$$\frac{\partial}{\partial \theta_{d_1}} \ln D(x^{(i)}) = \frac{\partial \ln(D)}{\partial D} \cdot \frac{\partial D}{\partial f} \cdot \frac{\partial f}{\partial \theta_{d_1}}$$

$$= \frac{1}{D} \cdot \frac{(-1)}{f^2} \cdot e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)} \cdot (-x_1)$$

$$\frac{1}{D} = \frac{1}{1/f}$$

$$= f \cdot \frac{1}{f^2} \cdot e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)} \, x_1$$

$$= \frac{e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}}{1 + e^{-(\theta_{d_1} x_1 - \theta_{d_2} x_2 - 2)}} \, x_1$$

x_1 x_2

$$D = \frac{1}{f}$$

$$\frac{\partial D}{\partial f} = \frac{-1}{f^2}$$

$$\alpha_1^{(i)} = \frac{x_1^{(i)} e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}}{1 + e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}},$$

$\theta_{d_1} = 0.1$

$\theta_{d_2} = 0.2$

When training a Generative Adversarial Network (GAN), we consider a dataset of real samples denoted as $\mathbf{X}_{real} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and the generated samples as $\mathbf{X}_{fake} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, where $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\tilde{\mathbf{x}}_1 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$, $\tilde{\mathbf{x}}_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$.

$\theta_{d_1} = 0.1; \theta_{d_2} = 0.2$

$i = 1 \quad x_1^{(1)} = 1 \quad x_2^{(1)} = 2, \quad x^{(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

where

$$\alpha_1^{(i)} = \frac{x_1^{(i)} e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}}{1 + e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}},$$

$$\alpha_2^{(i)} = -\frac{x_2^{(i)} e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}}{1 + e^{-(\theta_{d_1} x_1^{(i)} - \theta_{d_2} x_2^{(i)} - 2)}},$$

$$\beta_1^{(i)} = -\frac{x_{1_{fake}}^{(i)} e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)}}{e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)} \left(e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)} + 1\right)},$$

$$\beta_2^{(i)} = \frac{x_{2_{fake}}^{(i)} e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)}}{e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)} \left(e^{-(\theta_{d_1} x_{1_{fake}}^{(i)} - \theta_{d_2} x_{2_{fake}}^{(i)} - 2)} + 1\right)}.$$

Update θ_{d_1} and θ_{d_2} using gradient accent rule with $\eta = 0.02$:

$$\begin{bmatrix} \theta_{d_1} \\ \theta_{d_2} \end{bmatrix} = \begin{bmatrix} \theta_{d_1} \\ \theta_{d_2} \end{bmatrix} + \eta \times \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\ln D\left(\mathbf{x}^{(i)}\right) + \ln\left(1 - D(G(\mathbf{z}^{(i)}))\right) \right]$$

$$= \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} + 0.02 \times \begin{bmatrix} 1.5007 \\ -2.3596 \end{bmatrix}$$

$$= \begin{bmatrix} 0.13001 \\ 0.15281 \end{bmatrix}$$