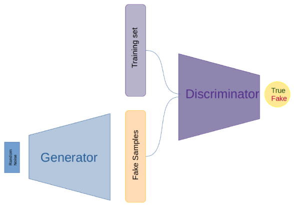
Deep Learning

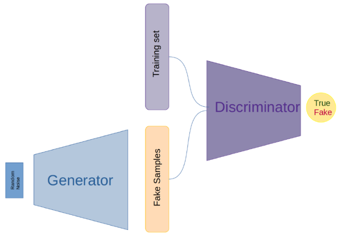
Introduction to Generative Adversarial Networks (GANs)



Learning goals

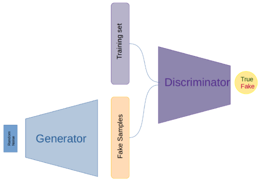
- architecture of a GAN
- minimax loss
- training a GAN

WHAT IS A GAN?



- A *generative adversarial network* (GAN) consists of two DNNs:
    - generator
    - discriminator
- Generator transforms random noise vector into fake sample.
- Discriminator gets real and fake samples as input and outputs probability of the input being real.

WHAT IS A GAN?



- Goal of generator: fool discriminator into thinking that the synthesized samples are real.
- Goal of discriminator: recognize real samples and not being fooled by generator.
- This sets off an arms race. As the generator gets better at producing realistic samples, the discriminator is forced to get better at detecting the fake samples which in turn forces the generator to get even better at producing realistic samples and so on.

FAKE CURRENCY ILLUSTRATION

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles.

-Ian Goodfellow

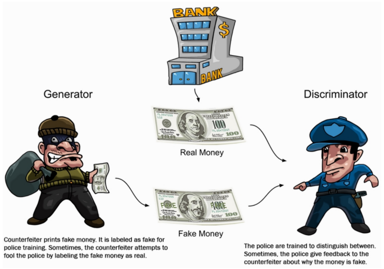

BANK
Generator
Discriminator
Real Money
Fake Money
Counterfeiter prints fake money. It is labeled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labeling the fake money as real.
The police are trained to distinguish between. Sometimes, the police give feedback to the counterfeiter about why the money is fake.

Image created by Mayank Vadsola

GAN Training

MINIMAX LOSS FOR GANS

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

- $p_{\text{data}(\mathbf{x})}$ is our target, the data distribution.
- The generator is a neural network mapping a latend random vector $\mathbf{z}$ to generated sample $G(\mathbf{z})$. Even if the generator is a determinisic function, we have random outputs, i.e. variability.
- $p(\mathbf{z})$ is usually a uniform distribution or an isotropic Gaussian. It is typically fixed and not adapted during training.

MINIMAX LOSS FOR GANS

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

- $G(\mathbf{z})$ is the output of the generator for a given state $\mathbf{z}$ of the latent variables.
- $D(\mathbf{x})$ is the output of the discriminator for a real sample $\mathbf{x}$.
- $D(G(\mathbf{z}))$ is the output of the discriminator for a fake sample $G(\mathbf{z})$ synthesized by the generator.

MINIMAX LOSS FOR GANS

$$\min_{G} \max = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}(\mathbf{x})}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

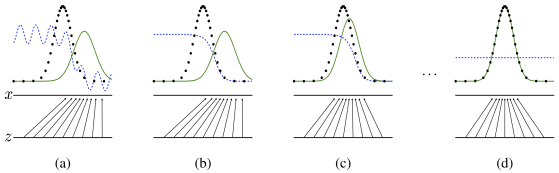
- $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}(\mathbf{x})}}[\log D(\mathbf{x})]$ is the log-probability of correctly classifying real data points as real.
- $\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$ is the log-probability of correctly classifying fake samples as fake.
- With each gradient update, the discriminator tries to push $D(\mathbf{x})$ toward 1 and $D(G(\mathbf{z})))$ toward 0. This is the same as maximizing V(D,G).
- The generator only has control over $D(G(\mathbf{z}))$ and tries to push that toward 1 with each gradient update. This is the same as minimizing V(D,G).

GAN TRAINING : PSEUDOCODE

Algorithm Minibatch stochastic gradient descent training of GANs. Amount of training iterations, amount of discriminator updates *k*

```
1: for number of training iterations do
2:     for k steps do
3:         Sample minibatch of $m$ samples $\{\mathbf{z}^{(1)} \dots \mathbf{z}^{(m)}\}$ from prior $p_g(\mathbf{z})$
4:         Sample minibatch of $m$ examples $\{\mathbf{x}^{(1)} \dots \mathbf{x}^{(m)}\}$ from training data
5:         Update discriminator by ascending the stochastic gradient:
               $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))\right]$
6:     end for
7:     Sample minibatch of $m$ noise samples $\{\mathbf{z}^{(1)} \dots \mathbf{z}^{(m)}\}$ from the noise prior $p_g(\mathbf{z})$
8:     Update generator by descending the stochastic gradient:
               $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)})))$
9: end for
```

GAN TRAINING: ILLUSTRATION



GANs are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black,dotted line) p_x from those of the generative distribution $p_g(G)$ (green, solid line).Source: Goodfellow et al (2017)

- For k steps, G's parameters are frozen and one performs **gradient ascent** on D to increase its accuracy.
- Finally, D's parameters are frozen and one performs **gradient descent** on G to increase its generation performance.
- Note, that G gets to peek at D's internals (from the back-propagated errors) but D does not get to peek at G.

DIVERGENCE MEASURES

- The goal of generative modeling is to learn $p_{\text{data}}(\mathbf{x})$.
- The differences between different generative models can be measured in terms of **divergence measures**.
- A divergence measure quantifies the distance between two distributions.
- There are many different divergence measures that one can us (e.g. Kullback-Leibler divergence).
- All such measures always positive and 0 if and only if the two distributions are equal to each other.

DIVERGENCE MEASURES

- One approach to training generative models is to explicitly minimize the distance between $p_{\text{data}}(\mathbf{x})$ and the model distribution $p_\theta(\mathbf{x})$ according to some divergence measure.
- If our generator has the capacity to model $p_{\text{data}}(\mathbf{x})$ perfectly, the choice of divergence does not matter much because they all achieve their minimum (that is 0) when $p_\theta(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$.
- However, it is not likely that that the generator, which is parametrized by the weights of a neural network, is capable of perfectly modelling an arbitrary $p_{\text{data}}(\mathbf{x})$.
- In such a scenario, the choice of divergence measure matters, because the parameters that miniminize the various divergence measures differ.

IMPLICIT DIVERGENCE MEASURE OF GANS

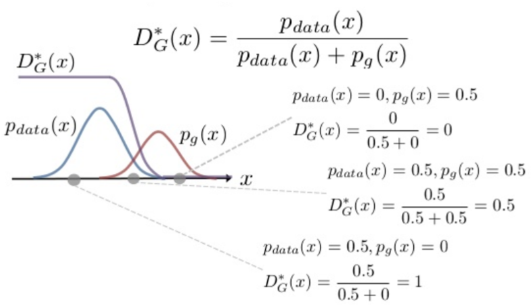
- GANs do not explicitly minimize any divergence measure.
- However, (under some assumptions!) optimizing the minimax loss is equivalent to implicitly minimizing a divergence measure.
- That is, if the optimal discriminator is found in every iteration, the generator minimizes the **Jensen-Shannon divergence (JSD)** (theorem and proof are given by the original GAN paper (Goodfellow et al, 2014)):

$$JS(p_{\text{data}}||p_g) = \frac{1}{2}KL(p_{\text{data}}||\frac{p_{\text{data}} + p_g}{2}) + \frac{1}{2}KL(p_g||\frac{p_{\text{data}} + p_g}{2})$$

$$KL(p_{\text{data}}||p_g) = E_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[log\frac{p_{\text{data}}(\mathbf{x})}{p_g(\mathbf{x})}]$$

OPTIMAL DISCRIMINATOR

For G fixed, the optimal discriminator D_G^* is:



$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$
$D_G^*(x)$
$p_{data}(x)$
$p_g(x)$
x
$p_{data}(x) = 0, p_g(x) = 0.5$
$D_G^*(x) = \frac{0}{0.5 + 0} = 0$
$p_{data}(x) = 0.5, p_g(x) = 0.5$
$D_G^*(x) = \frac{0.5}{0.5 + 0.5} = 0.5$
$p_{data}(x) = 0.5, p_g(x) = 0$
$D_G^*(x) = \frac{0.5}{0.5 + 0} = 1$

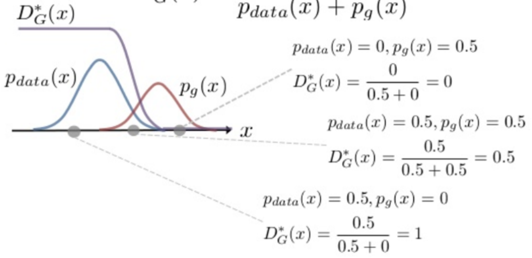
Credit: Mark Chang

- The optimal discriminator returns a value greater than 0.5 if the probability to come from the data ($p_{data}(x)$) is larger than the probability to come from the generator ($p_g(x)$).

OPTIMAL DISCRIMINATOR

For G fixed, the optimal discriminator D_G^* is:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$



$D_G^*(x)$
$p_{data}(x)$
$p_g(x)$
x
$p_{data}(x) = 0, p_g(x) = 0.5$
$D_G^*(x) = \frac{0}{0.5 + 0} = 0$
$p_{data}(x) = 0.5, p_g(x) = 0.5$
$D_G^*(x) = \frac{0.5}{0.5 + 0.5} = 0.5$
$p_{data}(x) = 0.5, p_g(x) = 0$
$D_G^*(x) = \frac{0.5}{0.5 + 0} = 1$

Credit: Mark Chang

- Note: The optimal solution is almost never found in practice, since the discriminator has a finite capacity and is trained on a finite amount of data.
- Therefore, the assumption needed to guarantee that the generator minimizes the JSD does usually not hold in practice.

REFERENCES

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014)
Generative Adversarial Networks
https://arxiv.org/abs/1406.2661

Ian Goodfellow (2016)
NIPS 2016 Tutorial: Generative Adversarial Networks
https://arxiv.org/abs/1701.00160

Mark Chang (2016)
Generative Adversarial Networks
https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks