Introduction to

Generative Adversarial Network(GAN)

Arsham Gholamzadeh Khoee
Alireza Javaheri

Overview

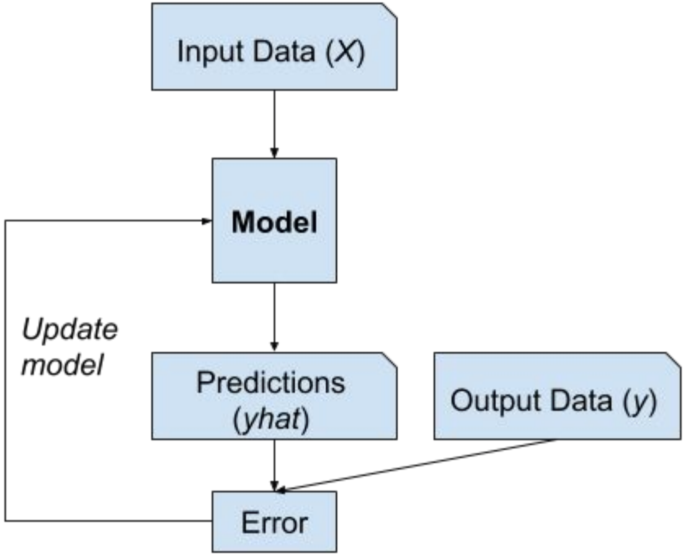
- What Are Generative Models ?
- Why Generative Adversarial Networks ?
- What Are Generative Adversarial Network ?

What are Generative Model ?

Supervised vs. Unsupervised Learning

Supervised Learning

In the predictive or supervised learning approach, the goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs ...



Supervised vs. Unsupervised Learning

Supervised Learning

Data : (x, y)

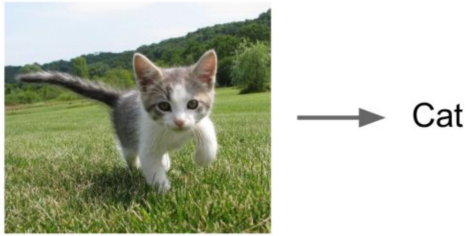
x is data, y label

Goal : Learn a function to map x -> y

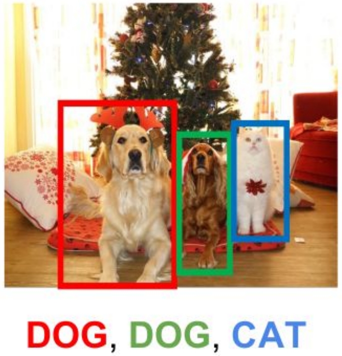
Examples : Classification, regression, object detection, semantic segmentation, image captioning, etc.

Supervised vs Unsupervised Learning

Supervised Learning



classification



Object detection

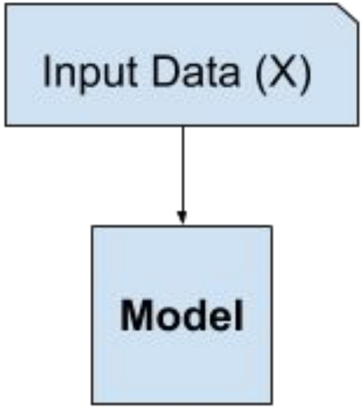
segmentation

Image captioning

Supervised vs Unsupervised Learning

Unsupervised Learning

In the descriptive or unsupervised learning we are only given inputs, and the goal is to find "interesting patterns" in the data. This is a much less well-defined problem, since we are not told what kinds of patterns to look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given x to the observed value).



Supervised vs Unsupervised Learning

Unsupervised Learning

Data : x

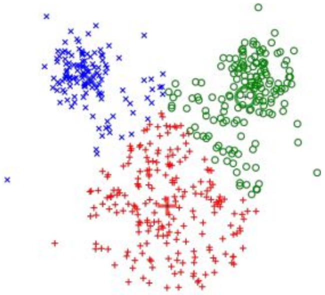
Just data, no labels !

Goal : Learn some underlying hidden structure of the data

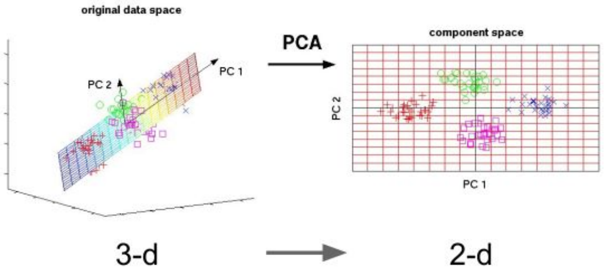
Examples : Clustering, Dimensionality reduction, etc.

Supervised vs Unsupervised Learning

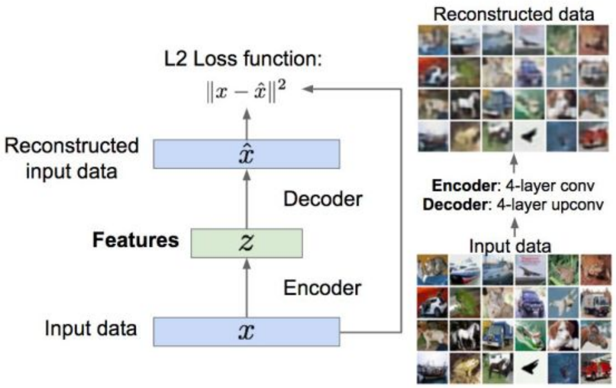
Unsupervised Learning



clustering



PCA (Dimensionality reduction)



Autoencoder

Supervised vs Unsupervised Learning

Supervised Learning

Data : (x, y)

x is data, y label

Goal : Learn a function to map x -> y

Examples : Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data : x

Just data, no labels !

Goal : Learn some underlying hidden structure of the data

Examples : Clustering, Dimensionality reduction, etc.

Generative Models

Given training data, generate new samples from same distribution.

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Generative Models

Given training data, generate new samples from same distribution.

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

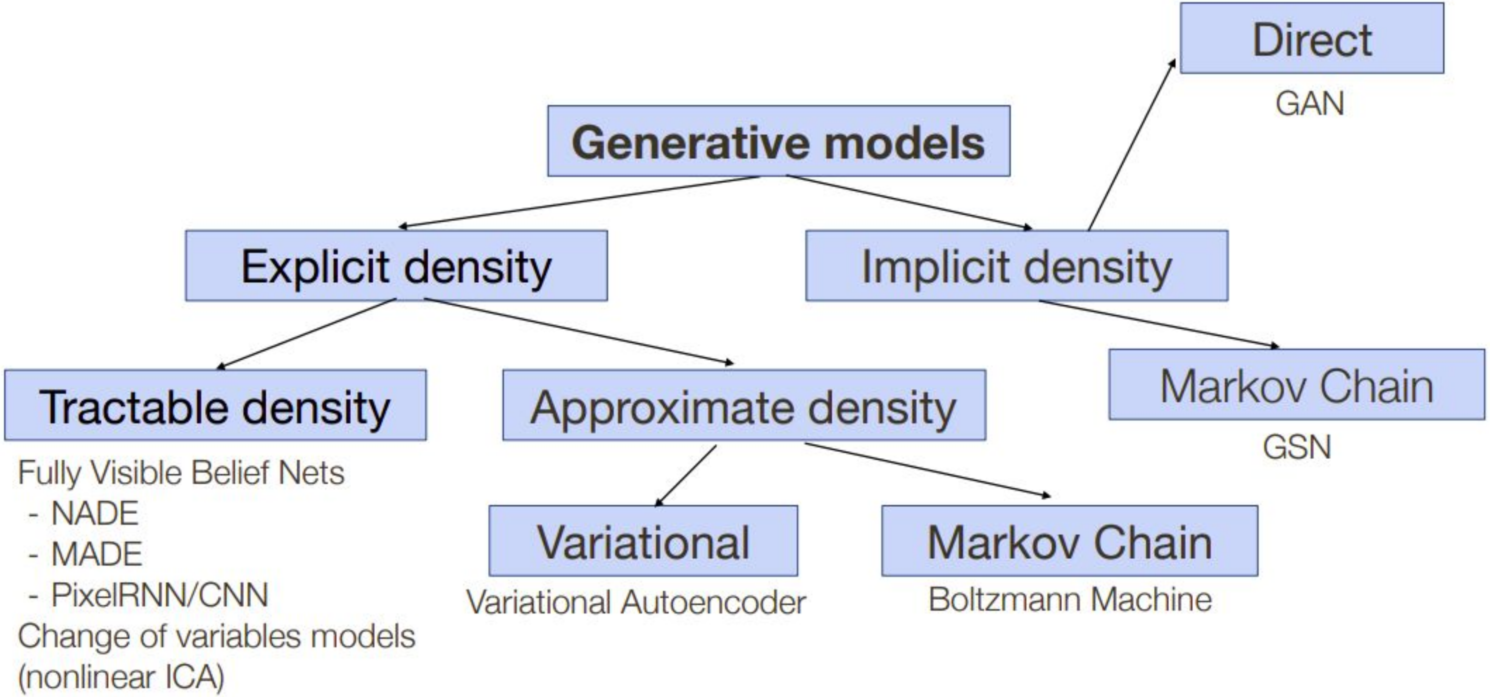
Why Generative Models ?

- Realistic samples for artwork, super-resolution, colorization, etc.
- Generative models of time-series data can be used for simulation and planning
- Anomaly detection
- Etc .

Why Generative Models ?

CycleGAN

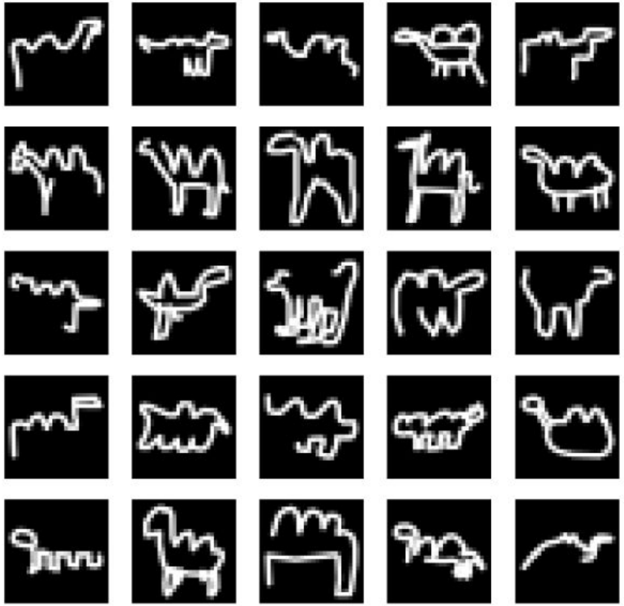
Taxonomy of Generative Models



What Are Generative Adversarial Network?

Overview of GAN Structure

- Photographer
- Painter



Overview of GAN Structure

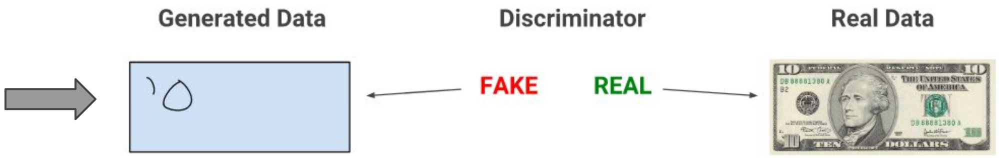
Overview of GAN Structure

A generative adversarial network (GAN) has two parts:

- The **generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.
- The **discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

Overview of GAN Structure

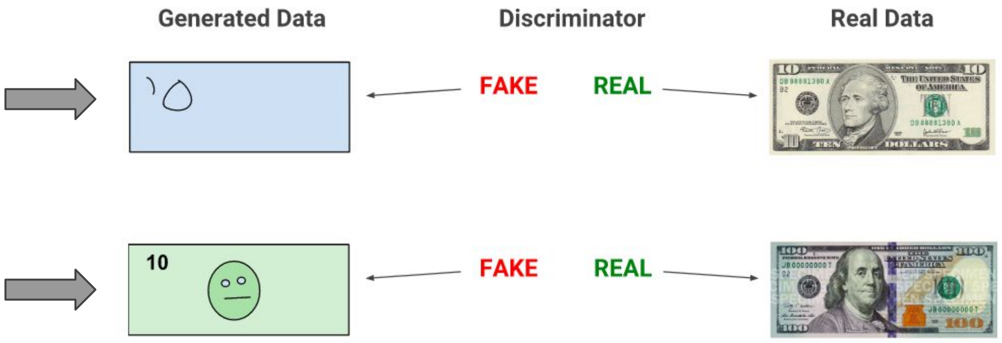
When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake



Overview of GAN Structure

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake

As training progresses, the generator gets closer to producing output that can fool the discriminator

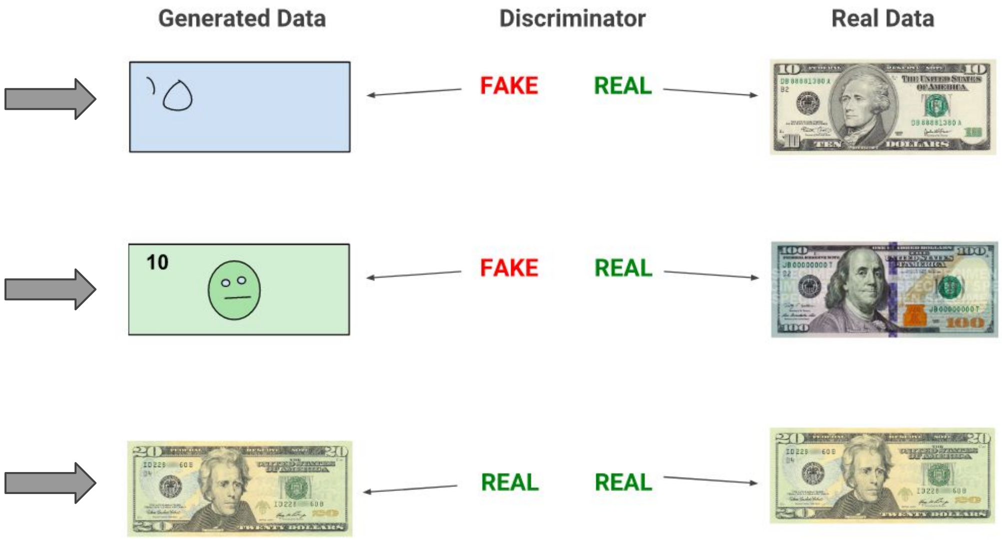


Overview of GAN Structure

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake

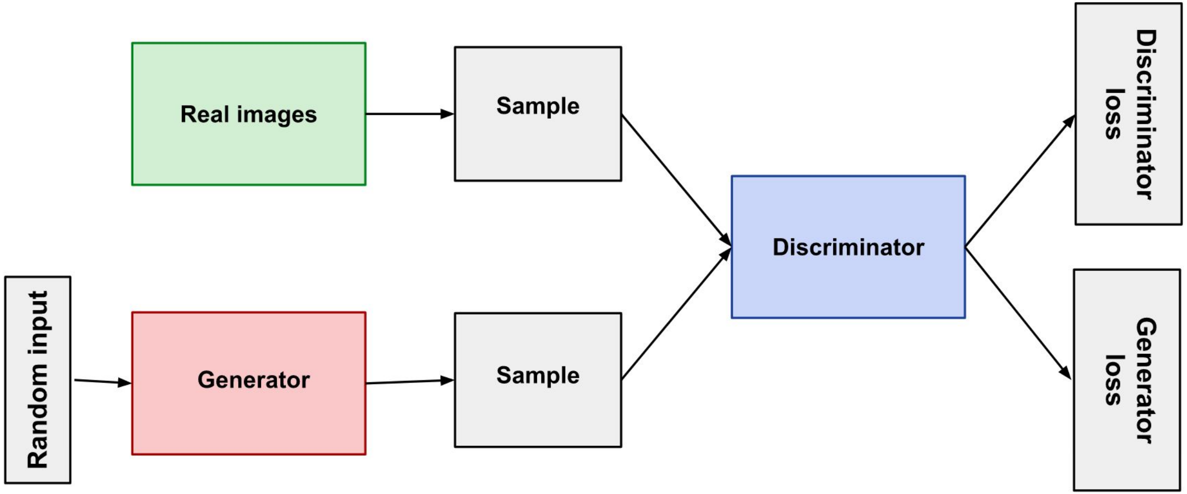
As training progresses, the generator gets closer to producing output that can fool the discriminator

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.



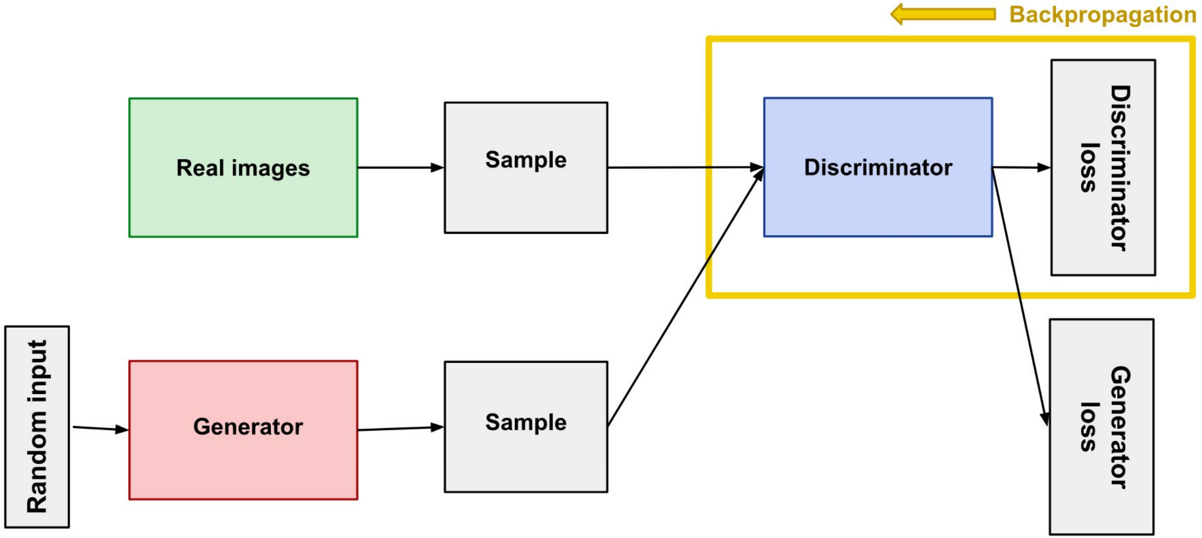
Overview of GAN Structure

Both the generator and the discriminator are neural networks. The generator output is connected directly to the discriminator input. Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights.



The Discriminator

The discriminator in a GAN is simply a classifier. **It tries to distinguish real data from the data created by the generator**. It could use any network architecture appropriate to the type of data it's classifying.

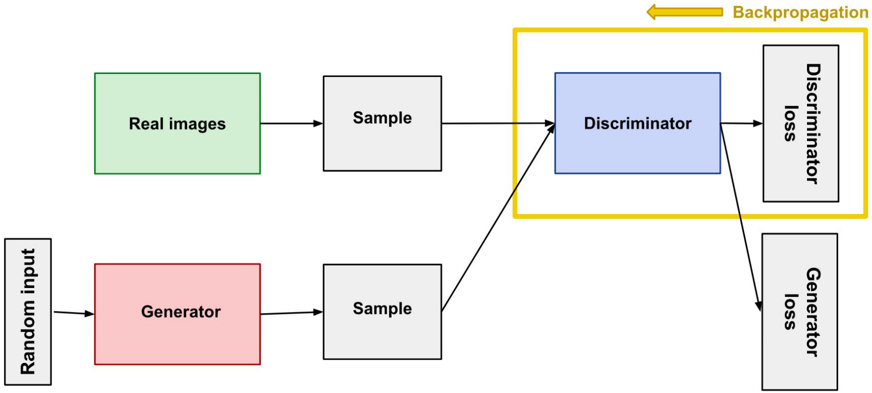


The Discriminator

Discriminator Training Data

The discriminator's training data comes from two sources:

- **Real data** instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
- **Fake data** instances created by the generator. The discriminator uses these instances as negative examples during training.

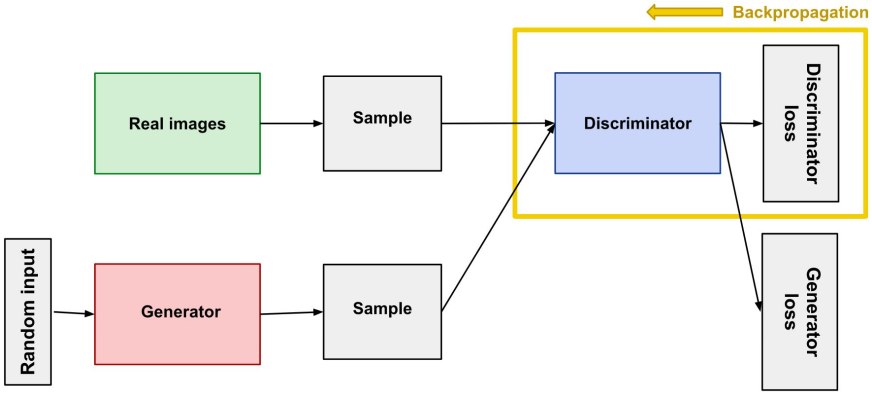


The Discriminator

Discriminator Training Data

The discriminator's training data comes from two sources:

- **Real data** instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
- **Fake data** instances created by the generator. The discriminator uses these instances as negative examples during training.



During discriminator training the generator does not train. Its weights remain constant while it produces examples for the discriminator to train on.

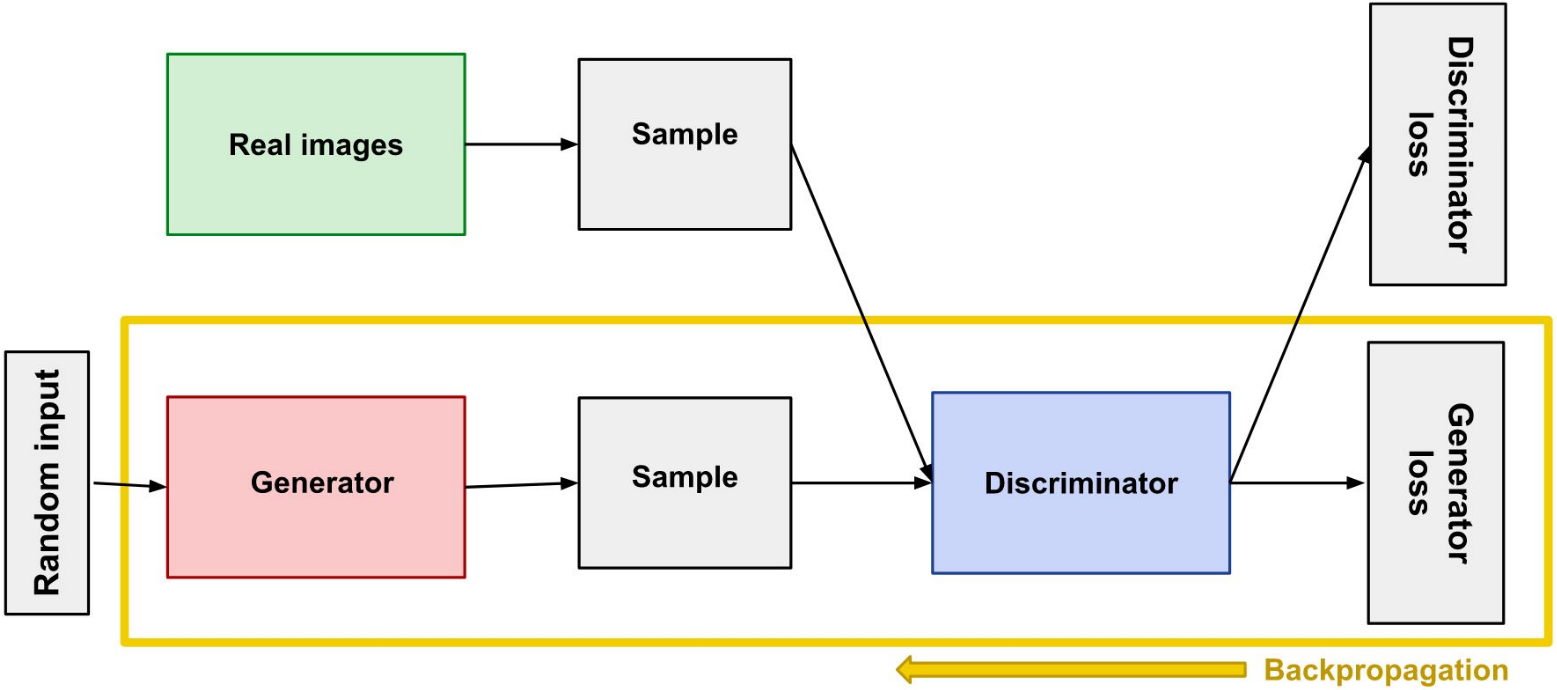
The Discriminator

Training the Discriminator

The discriminator connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss. We use the generator loss during generator training.

1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

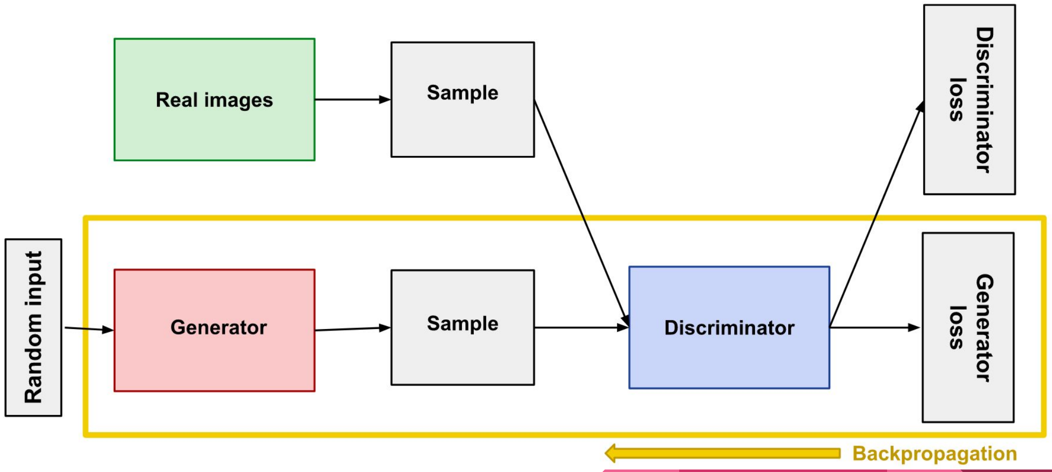
The Generator



The Generator

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.

1. Sample random noise
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Backpropagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.



GAN Training

- The discriminator trains for one or more epochs
- The generator trains for one or more epochs
- Repeats steps 1 and 2 to continue to train the generator and discriminator networks.

Convergence

- GAN convergence is hard to identify
- The generator improves with training, the discriminator performance gets worse

GAN Training

- The discriminator trains for one or more epochs
- The generator trains for one or more epochs
- Repeats steps 1 and 2 to continue to train the generator and discriminator networks.

Convergence

- GAN convergence is hard to identify
- The generator improves with training, the discriminator performance gets worse
- If the generator succeeds perfectly, then the discriminator has a 50% accuracy
- This progression poses a problem for convergence of the GAN as a whole

GAN Training

- The discriminator trains for one or more epochs
- The generator trains for one or more epochs
- Repeats steps 1 and 2 to continue to train the generator and discriminator networks.

Convergence

- GAN convergence is hard to identify
- The generator improves with training, the discriminator performance gets worse
- If the generator succeeds perfectly, then the discriminator has a 50% accuracy
- This progression poses a problem for convergence of the GAN as a whole
- The discriminator is giving completely random feedback
- Convergence is often a fleeting, rather than stable state

Loss Function

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

Loss Function

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

- **D(x)** is the discriminator's estimate of the probability that real data instance x is real

Loss Function

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

- **D(x)** is the discriminator's estimate of the probability that real data instance x is real
- **E_x** is the expected value over all real data instances

Loss Function

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

- **D(x)** is the discriminator's estimate of the probability that real data instance x is real
- **Ex** is the expected value over all real data instances
- **G(z)** is the generatore's output when given noise z.

Loss Function

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

- **D(x)** is the discriminator’s estimate of the probability that real data instance x is real
- **Ex** is the expected value over all real data instances
- **G(z)** is the generatore’s output when given noise z.
- **D(G(z))** is the discriminator’s estimate of the probability that a fake instance is real

Loss Function

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

- **D(x)** is the discriminator’s estimate of the probability that real data instance x is real
- **Ex** is the expected value over all real data instances
- **G(z)** is the generatore’s output when given noise z.
- **D(G(z))** is the discriminator’s estimate of the probability that a fake instance is real
- **Ez** is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instance G(z))

Loss Function

Review

$$h_\theta(x^i) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_j x_j$$

$$x_0 = 1, \quad j = \textit{the number of features}$$

Loss Function

Review

$$h_\theta(x^i) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_j x_j$$

$$x_0 = 1, \quad j = \textit{the number of features}$$

$$\textit{Least Squared Error} = \frac{1}{2}\left(h_\theta(x^i) - y^i\right)^2$$

Loss Function

Review

$$h_\theta(x^i) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_j x_j$$

$$x_0 = 1, \quad j = \textit{the number of features}$$

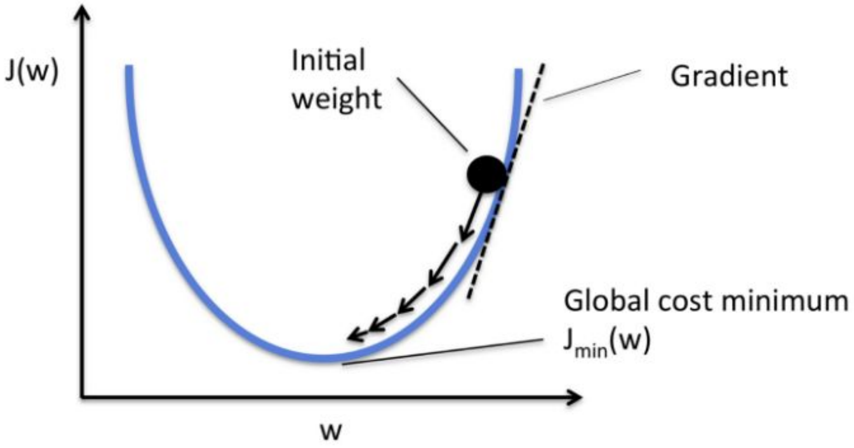
$$\textit{Least Squared Error} = \frac{1}{2}\left(h_\theta(x^i) - y^i\right)^2$$

$$\textit{Cost Function} = J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right)^2, \; m = \textit{number of data}$$

Loss Function

Review

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$



Loss Function

Review

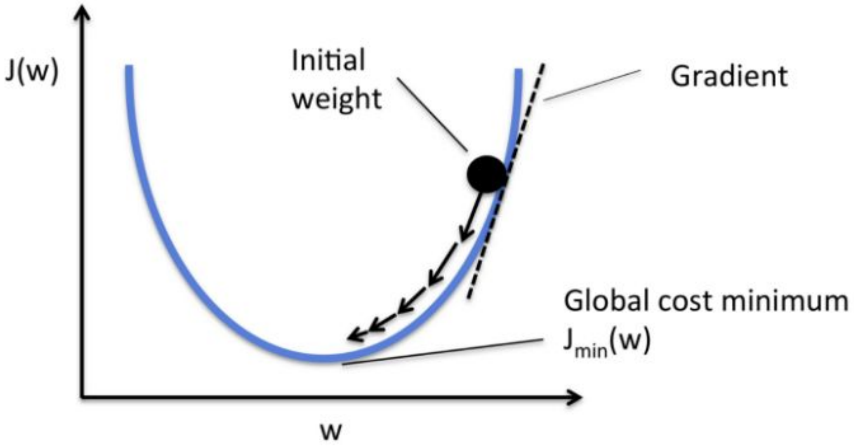
$$\nabla_{\theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right) x_j^i$$

$$\widehat{\theta}_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right) x_0^i$$

$$\widehat{\theta}_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right) x_1^i$$

...

$$\widehat{\theta}_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right) x_j^i$$



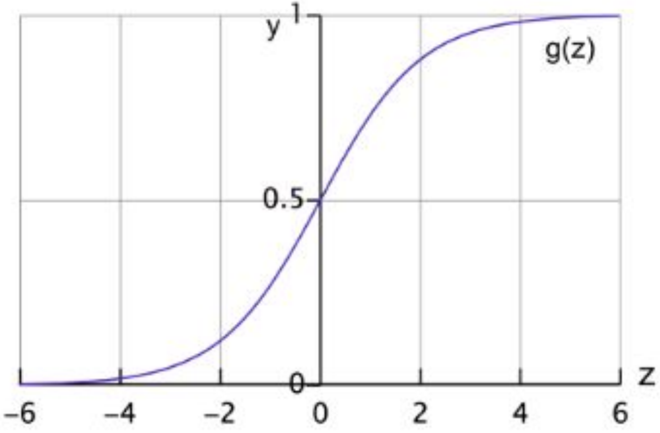
Loss Function

Logistic Regression

$$h_\theta(x^i) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_j x_j = \theta^T x$$

$$\textit{Sigmoid Function}: \quad g(z) = \frac{1}{1 + e^{(-z)}}$$

$$\textit{Hypothesis}: \quad h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}}$$



Loss Function

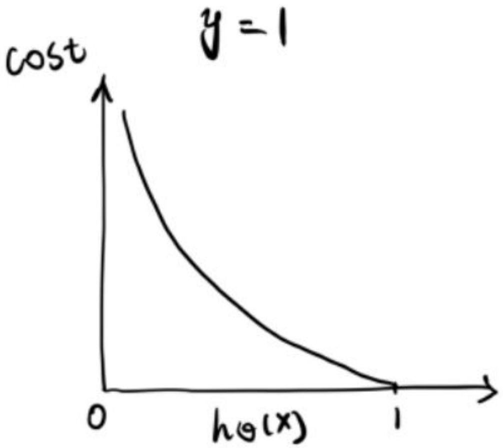
Logistic Regression

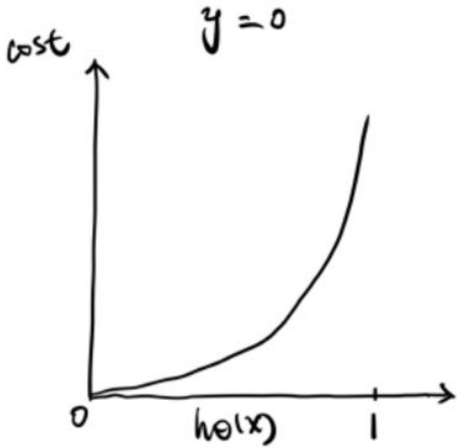
$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Loss Function

Logistic Regression

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

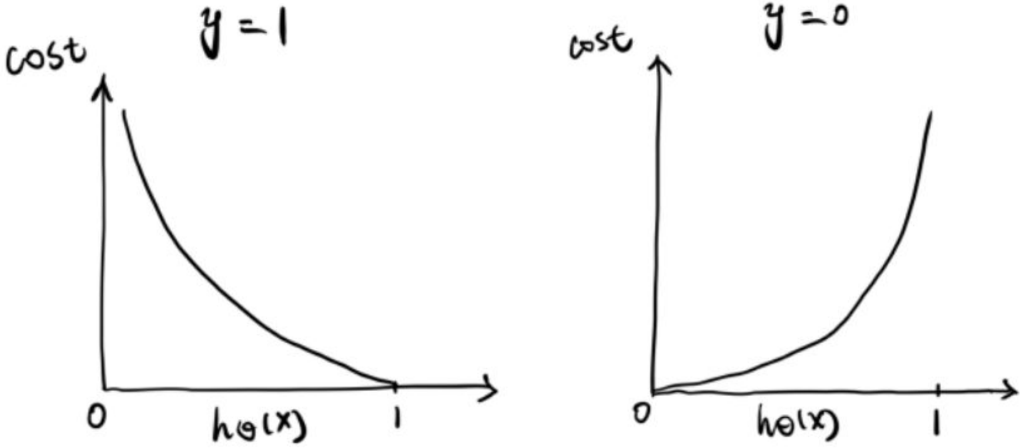




Loss Function

Logistic Regression

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

Loss Function

Logistic Regression

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} -y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)}))]$$

$$m = number\ of\ samples$$

Loss Function

Discriminator's loss function(cost func)

$$L_D = -\sum_{i=1}^{m} y^i \log(D(x_i)) - \sum_{i=1}^{m} (1 - y^i) \log(1 - D(x_i))$$

Loss Function

Discriminator's loss function(cost func)

$$L_D = -\sum_{i=1}^{m} y^i \log(D(x_i)) - \sum_{i=1}^{m} (1 - y^i)\log(1 - D(x_i))$$

$$x \sim \begin{cases} p_{data} & \text{if, } y_1 = 1, \ \textit{with prob } 0.5 \\ p_g & \text{if, } y_1 = 0 \ \ \textit{otherwise} \end{cases}$$

$$L_D = -\left[\sum_{i=1}^{\frac{m}{2}} y^i \log(D(x_i)) + \sum_{i=\frac{m}{2}}^{m} (1 - y^i)\log(1 - D(x_i))\right]$$

Loss Function

Discriminator’s loss function(cost func)

$$L_D = -\left[\sum_{i=1}^{\frac{m}{2}} y^i \log(D(x_i)) + \sum_{i=\frac{m}{2}}^{m} \left(1 - y^i\right)\log(1 - D(x_i))\right]$$

Loss Function

Discriminator’s loss function(cost func)

$$L_D = -\left[\sum_{i=1}^{\frac{m}{2}} y^i \log(D(x_i)) + \sum_{i=\frac{m}{2}}^{m} \left(1 - y^i\right)\log(1 - D(x_i))\right]$$

$$L_D = -\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y \log(D(x)) + \frac{1}{2}\mathbb{E}_{x\sim p_g}(1 - y)\log(1 - D(x))\right]$$

Loss Function

Discriminator's loss function(cost func)

$$L_D = -\left[\sum_{i=1}^{\frac{m}{2}} y^i \log(D(x_i)) + \sum_{i=\frac{m}{2}}^{m} \left(1 - y^i\right)\log(1 - D(x_i))\right]$$

$$L_D = -\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y \log(D(x)) + \frac{1}{2}\mathbb{E}_{x\sim p_g}(1 - y)\log(1 - D(x))\right]$$

$$L_D = -\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y \log(D(x)) + \frac{1}{2}\mathbb{E}_{z\sim p_z}(1 - y)\log(1 - D(G(z)))\right]$$

Loss Function

Discriminator's loss function(cost func)

$$L_D = -\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y\log(D(x)) + \frac{1}{2}\mathbb{E}_{z\sim p_z}(1-y)\log(1-D(G(z)))\right]$$

Loss Function

Discriminator's loss function(cost func)

$$L_D = -\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y\log(D(x)) + \frac{1}{2}\mathbb{E}_{z\sim p_z}(1-y)\log(1-D(G(z)))\right]$$

- The discriminator's goal, through training, is to minimize its loss LD

$$\max_D[-L_D] = \max_D\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y\log(D(x)) + \frac{1}{2}\mathbb{E}_{z\sim p_z}(1-y)\log(1-D(G(z)))\right]$$

Loss Function

Discriminator's loss function(cost func)

$$\max_D[-L_D] = \max_D\left[\frac{1}{2}\mathbb{E}_{x\sim p_{data}} y\log(D(x)) + \frac{1}{2}\mathbb{E}_{z\sim p_z}(1-y)\log(1-D(G(z)))\right]$$

- The generator however, wants to maximize the discriminator's uncertainty (L_D), or equivalently minimize $-L_D$

$$L_G = -L_D$$

Loss Function

Discriminator’s loss function(cost func)

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