Unsupervised Learning

Wenhu Chen
Lecture 17

Readings: RN 21.7.1, PM 10.2, GBC 14.1, 20.10.4

Outline

Learning Goals

- Understanding what is unsupervised Learning
- Understanding K-Means clustering algorithm
- Knowing how to perform PCA
- Understanding the basic idea of Auto-Encoder and GAN

Unsupervised Learning Tasks

2 major types of tasks:

- *Representation learning*: learning low-dimensional representations of examples
- *Generative modelling*: learning probability distribution from which new examples can be drawn as samples

Unsupervised Learning - Clustering

Clustering is a common unsupervised representation learning task

→ Goal is to group training examples into *clusters*.

→ Clusters can be thought of as classes/categories.

Unsupervised Learning - Clustering

2 types of clustering tasks

- *Hard clustering*: each example is assigned to 1 cluster with certainty
  $\rightarrow \texttt{class}(x) = c$
- *Soft clustering*: each example has a probability distribution over all clusters
  $\rightarrow \texttt{class}(x) \sim P(C|x)$

k-Means Clustering - Overview

- A hard clustering algorithm
- Learns to definitively assign examples to classes
- Input: number of clusters k, training examples X
- Goal is to learn a representation that assigns examples to the appropriate class $c \in \{1, 2, \ldots, k\}$

k-Means Clustering - Centroids

Suppose each example contains n features: $x = \langle x_1, x_2, \ldots, x_n \rangle$

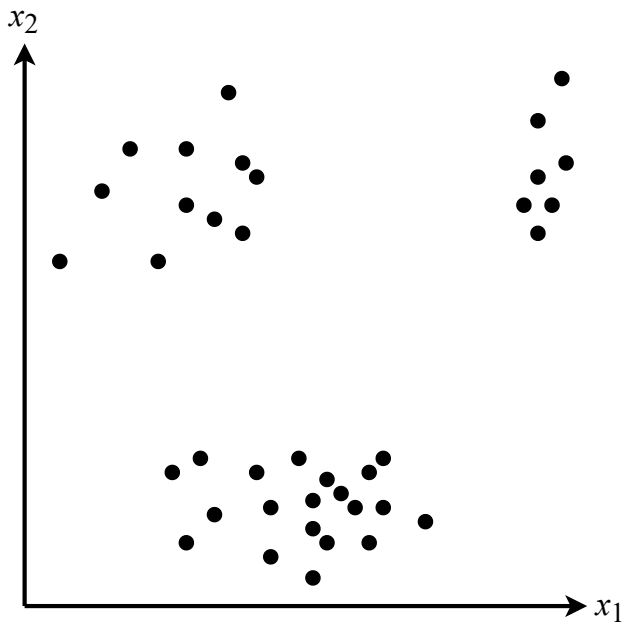
Each feature x_j is real-valued.

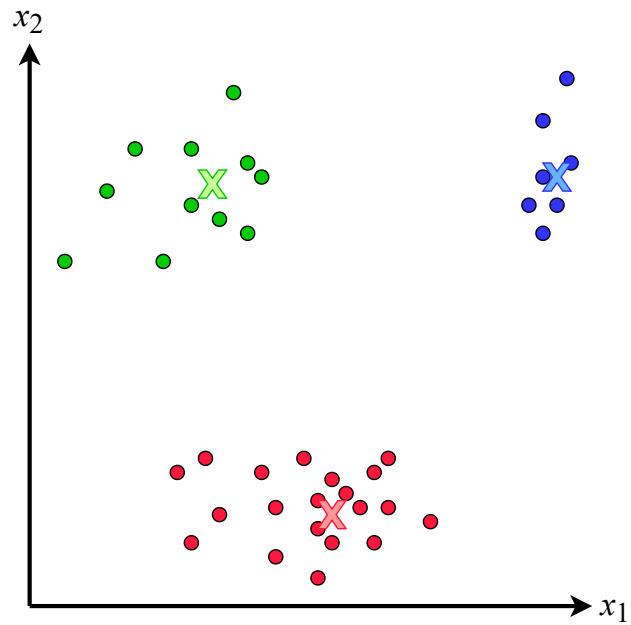
k-Means learns a *centroid* for each cluster and assigns examples to the closest centroid

- By "closest" we mean the centroid that is the shortest distance from x
- Need to define a distance function $d(c, x)$
  $\rightarrow$ E.g. Euclidean distance (L2): $d(c, x) = \sqrt{\sum_{j=1}^{n} (c_j - x_j)^2}$

k-Means Clustering - Centroids

Example: $k = 3$, $x = \langle x_1, x_2 \rangle$





k-Means Clustering - Algorithm Overview

k-means alternates between 2 steps:

1. *Centroid update:* Set the centroid of each cluster as the feature-wise mean of each example currently assigned to the cluster.
2. *Cluster assignment:* Assign each training example x to the cluster with the closest centroid.

k-Means Clustering - Algorithm

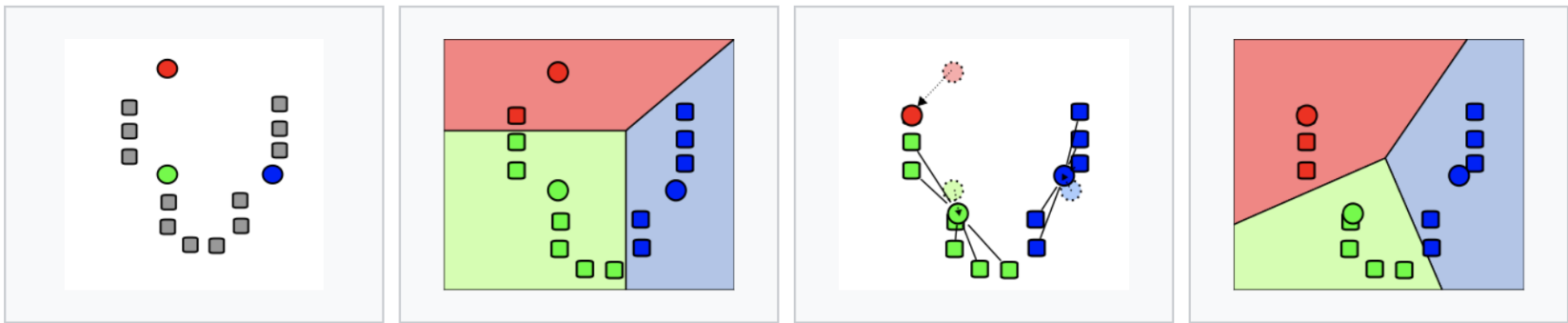
Input: $X \in \mathbb{R}^{m \times n}, k \in \mathbb{N}, d(c, x)$

1. Initialization:
   Randomly initialize k centroids: $C \in \mathbb{R}^{k \times n}$
2. While not converged, do:
   - Assign each example to the cluster whose centroid is closest.
     $Y[i] \leftarrow \arg\min_c d(C[c], X[i])$
   - Calculate the centroid for each cluster c by calculating the average feature value for each exmaple currently classified as cluster c.
     $C[c] \leftarrow \frac{1}{n_c} \sum_{j=1}^{n_c} X_c[j]$

Visualization of Clustering Algorithm

The clustering algorithm visualization:

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

k-Means Clustering - Example Iteration

Let's perform 1 iteration of k-means with $k = 2$, using Euclidean distance. Use the following dataset:

Example	x_1,	x_2	x_3
1	0.2	0.5	0
2	-0.6	2.1	1.2
3	-0.5	1.9	1.3
4	0.1	0.5	-0.3

Assume the current values for the centroids are as follows:

c	c_1,	c_2	c_3
1	0.3	0.8	-0.5
2	-0.1	-0.5	1.0

k-Means Clustering - Example Iteration

Let's perform 1 iteration of k-means with $k = 2$, using Euclidean distance. Use the following dataset:

Example	x_1,	x_2	x_3
1	0.2	0.5	0
2	-0.6	2.1	1.2
3	-0.5	1.9	1.3
4	0.1	0.5	-0.3

$c1 = [0.3, 0.8, -0.5]$, $c2 = [-0.1, -0.5, 1.0]$

Example 1 to $c1$: $0.1^2 + 0.3^2 + 0.5^2$, to $c2$: $0.3^2 + 1.0^2 + 1.0^2$: $c1$
Example 2 to $c1$: $0.9^2 + 1.3^2 + 1.7^2$, to $c2$: $0.4^2 + 2.6^2 + 0.2^2$: $c1$
Example 3 to $c1$: $0.8^2 + 1.1^2 + 1.8^2$, to $c2$: $0.4^2 + 2.4^2 + 0.3^2$: $c1$
Example 4 to $c1$: $0.2^2 + 0.3^2 + 0.2^2$, to $c2$: $0.2^2 + 1.0^2 + 1.3^2$: $c1$

k-Means Clustering - Example Iteration

Let's perform 1 iteration of k-means with $k = 2$, using Euclidean distance. Use the following dataset:

Example	x_1,	x_2	x_3
1	0.2	0.5	0
2	-0.6	2.1	1.2
3	-0.5	1.9	1.3
4	0.1	0.5	-0.3

Computing the new centroid:

$c1 = [0.2, 1.25, 0.55]$ $c2 = []$

$\rightarrow$ ou need to re-initialize the centroid.

k-Means Clustering - Finding the Best Solution

- k-means is guaranteed to converge (with L2 distance)
- Solution not guaranteed to be optimal
- To increase chance of finding better solution, you could:
  - Run multiple times with different random initial cluster assignments
  - Scale the features so that their domains are similar

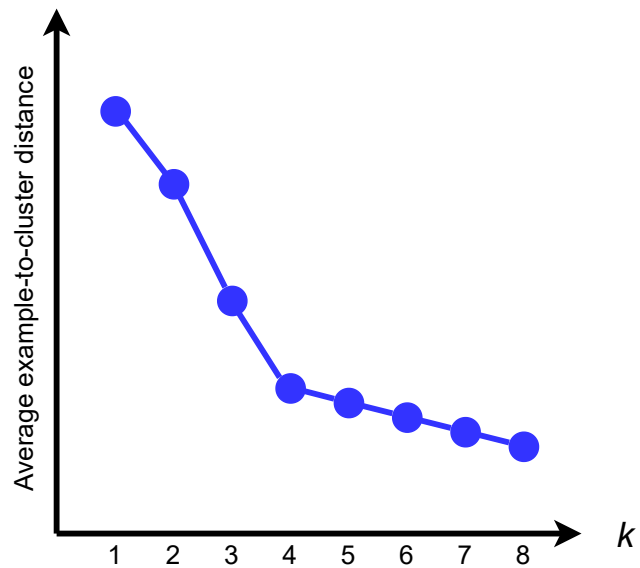
k-Means Clustering - Choosing proper k

The choice of k greatly determines the outcome of the clustering.

- As long as there are $\leq k+1$ examples, running k-means with $k+1$ clusters will result in lower error than running with k clusters
- But using too large k will defeat the point of representation learning...

k-Means Clustering - The Elbow Method

1. Execute k-means with multiple values of $k \in \{1, 2, \ldots, k_{max}\}$.
2. Plot average distance across all examples and assigned clusters.
3. Select k where there is drastic reduction in error improvement on the plot (i.e. "elbow point")



$\rightarrow$ Can be ambiguous, since it is manual

k-Means Clustering - Silhouette Analysis

1. Execute k-means with multiple values of $k \in \{1, 2, \ldots, k_{max}\}$.
2. Calculate average silhouette score $s(x)$ for each k across the dataset
3. Select k that maximizes average $s(x)$

$$s(x) = \begin{cases} \frac{b(x)-a(x)}{\max(a(x),b(x))} & \text{if } |C_x| > 1 \\ 0 & \text{if } |C_x| = 1 \end{cases}$$

- $a(x)$ is the average distance from example x to all other examples in its own cluster
- $b(x)$ is the smallest of the average distance of x to examples in any other cluster

$\rightarrow$ Significantly more objective than the Elbow Method

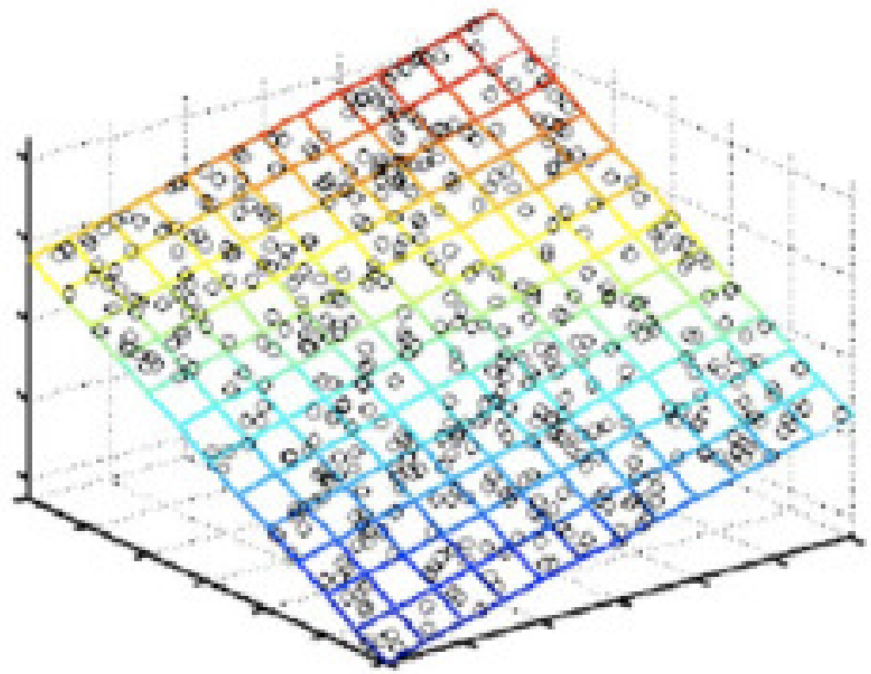
Dimension Reduction

Dimensionality reduction simply refers to the process of reducing the number of attributes in a dataset while keeping as much of the variation in the original dataset as possible.

- High Dimension Data actually resides in an inherent low-dimensional space.
- Additional dimensions are just random noise.
- Goal is to recover these inherent dimension and discard noise dimension.

Dimension Reduction

The observed data point dimensionality is not necessarily the intrinsic dimension of the data.

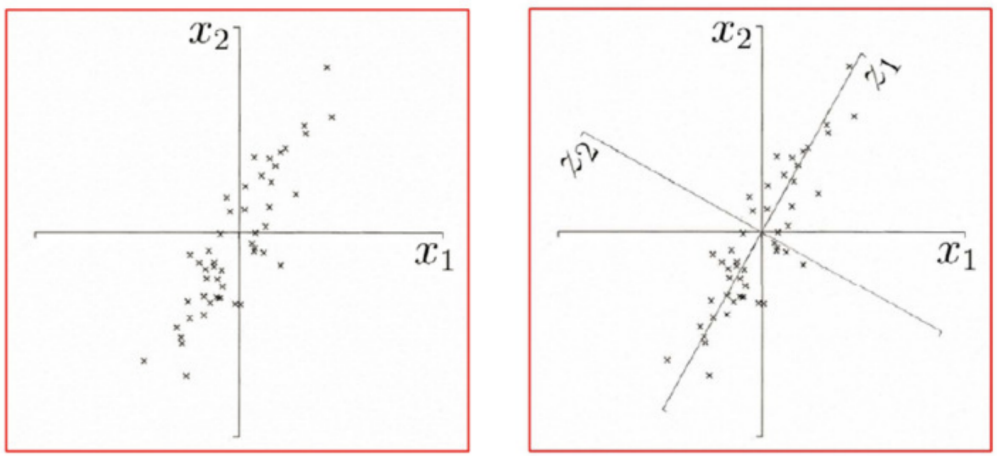


$$3d \Rightarrow 2d$$

By finding the intrinsic dimension, the problem becomes simpler.

Principal Component Analysis

- Widely used method for unsupervised dimensionality reduction
- account for variance of data in as few dimensions as possible
- First PC is the project of direction that maximizes the variance of projected data
- Second PC is the project of direction that is orthogonal to the first PC that maximizes the variance of projected data



Principal Component Analysis

- Mean center the data
- Compute Covariance Matrix Σ
- Calculate the eigen values and eigen vectors of Σ
  - Eigenvector with largest eigen value λ_1 is the first PC
  - Eigenvector with k_{th} largest eigenvaluve λ_k is the k-th PC.
  - $\lambda_k / \sum_k \lambda_k$ is the proportion of variance captured by k-th PC.

Autoencoders - Overview

- A representation learning algorithm
- Learn to map examples to low-dimensional representation

Autoencoders - Components

2 main components

1. *Encoder* $e(x)$: maps x to low-dimensional representation $\hat{z}$
2. *Decoder* $d(\hat{z})$: maps $\hat{z}$ to its original representation x

Autoencoder implements $\hat{x} = d(e(x))$

- $\hat{x}$ is the *reconstruction* of original input x
- Encoder and decoder learned such that $\hat{z}$ contains as much information about x as needed to reconstruct it

Minimize sum of squares of differences between input and prediction:

$$E = \sum_i (x_i - d(e(x_i)))^2$$

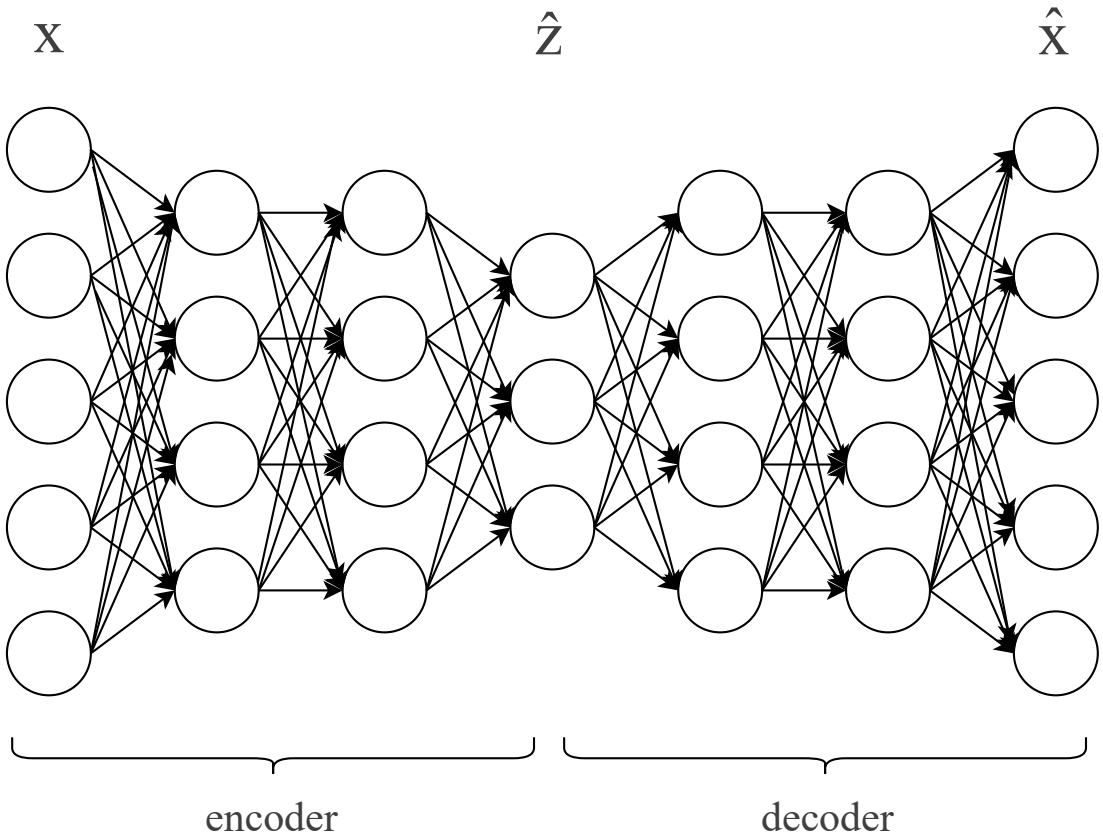
Linear Autoencoders

- Simplest form of autoencoder
- e and d are linear functions with shared weight matrix W

$$\hat{z} = Wx$$
$$\hat{x} = W^\top \hat{z}$$

Deep Neural Network Autoencoders

- Good for complex inputs
- e and d are feedforward neural networks, joined in series
- Train with backpropagation



Generative Adversarial Networks - Overview

a.k.a. GANs

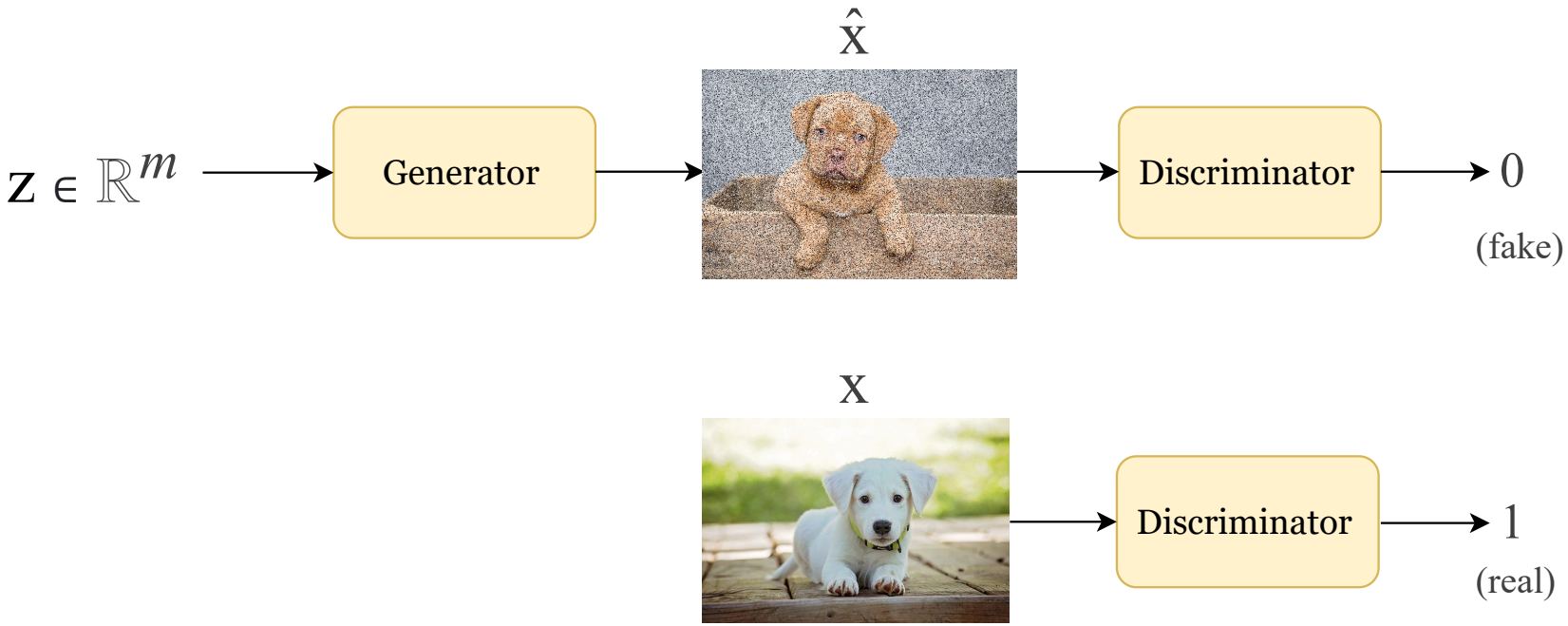
- A generative unsupervised learning algorithm
- Goal is to generate unseen examples that look like training examples

GANs - Components

GANs are actually a pair of neural networks:

- *Generator* $g(z)$: Given vector z in latent space, produces example x drawn from a distribution that approximates the true distribution of training examples
  $\rightarrow$ z usually sampled from a Gaussian distribution
- *Discriminator* $d(x)$: A classifier that predicts whether x is real (from training set) or fake (made by g)

GANs - Illustrative Example



GANs - Training

GANs are trained with a minimax error:

$$E = \mathbb{E}_x[\log(d(x))] + \mathbb{E}_z[\log(1 - d(g(z)))]$$

- Discriminator tries to maximize E
- Generator tries to minimize E

GANs - Training

GANs are trained with a minimax error:

$$E = \mathbb{E}_x[\log(d(x))] + \mathbb{E}_z[\log(1 - d(g(z)))]$$

- Discriminator tries to maximize E
- Generator tries to minimize E

After convergence:

- g should be producing realistic images
- d should output $\frac{1}{2}$, indicating maximal uncertainty

Revisiting Learning Goals

- Understanding what is unsupervised Learning
- Understanding K-Means clustering algorithm
- Knowing how to perform PCA
- Understanding the basic idea of Auto-Encoder and GAN