Bil 470 / YAP 470

Introduction to Machine Learning (Yapay Öğrenme)

Batuhan Bardak

Unsupervised Learning (PCA & K-means)

Date: 14.11.2022

Plan for today

- Dimensionality Reduction
  - PCA
- Unsupervised Learning
  - Hierarchical Clustering
  - K-Means

Learning Paradigms

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, r^{(n)}\right\}_{n=1}^{N}$

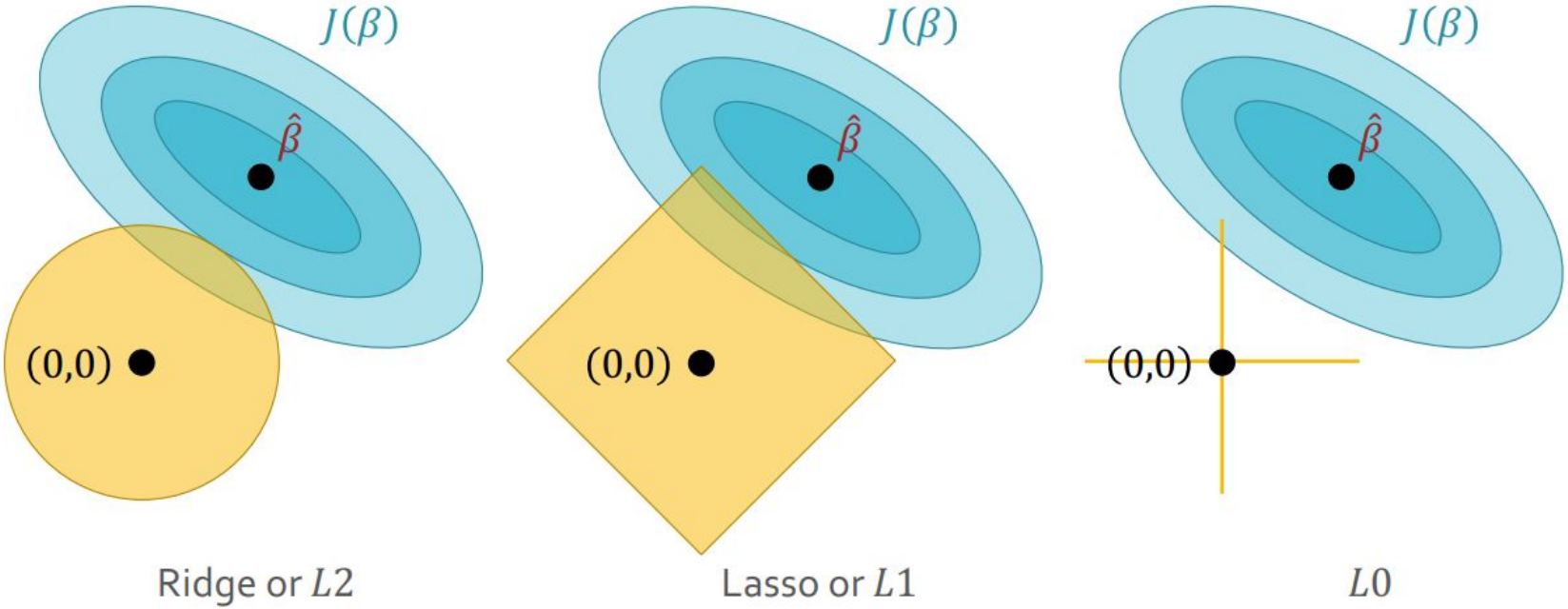
Slide Credit: Henry Chai

Dimensionality Reduction

- Goal: given some unlabeled data set, learn a latent (typically lower-dimensional) representation
- Use cases:
  - Reducing computational cost (runtime, storage, etc…)
  - Improving generalization
  - Visualizing data
- Applications:
  - High-resolution images/videos
  - Text data
  - Financial or transaction data

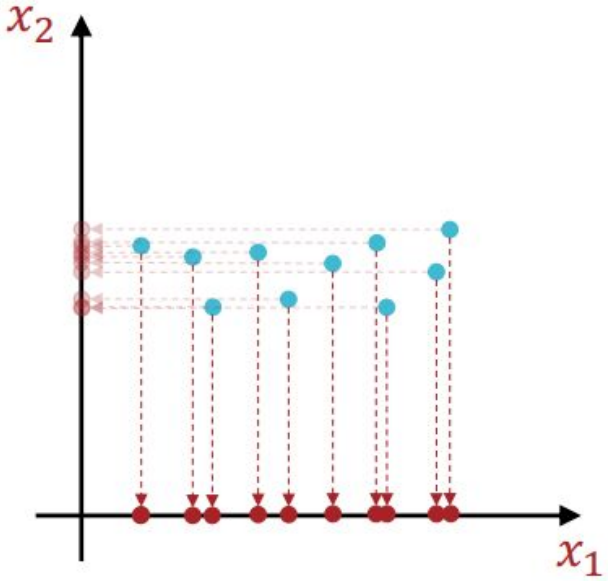
Slide Credit: Henry Chai

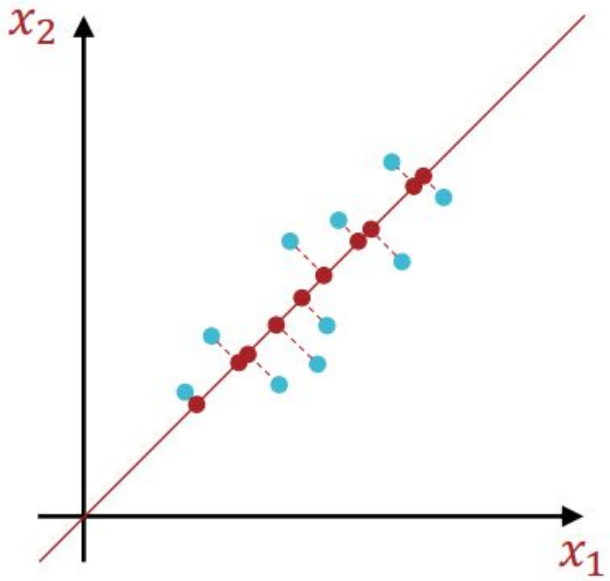
Recall: Regularization (L1)



Slide Credit: Henry Chai

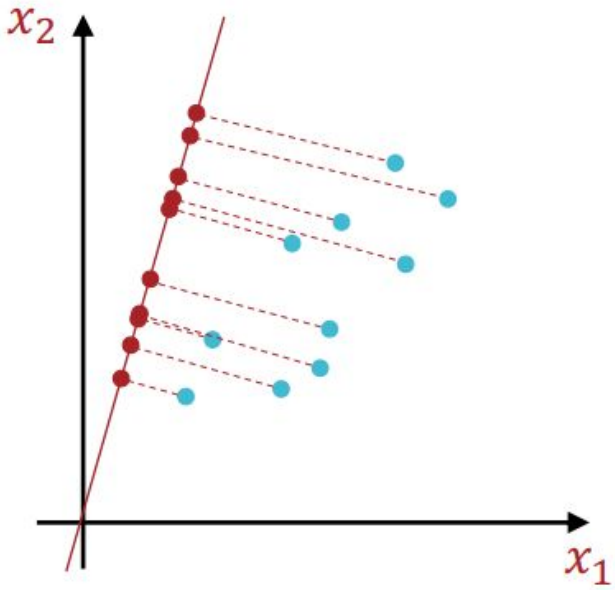
Feature Elimination

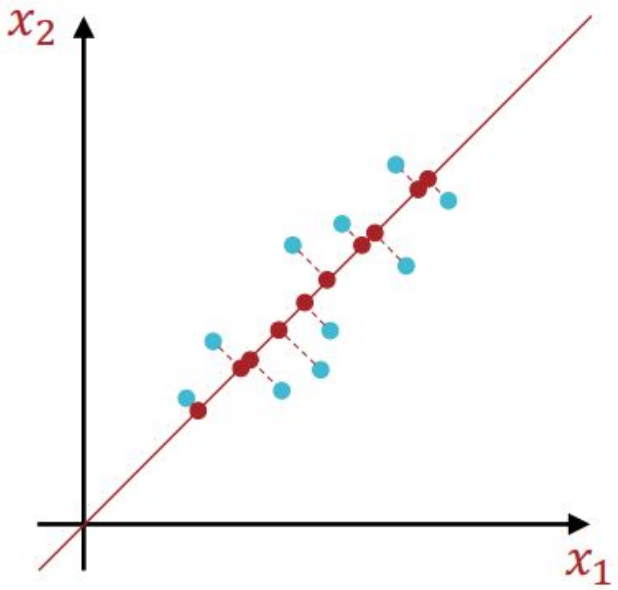




Slide Credit: Henry Chai

Feature Reduction





Slide Credit: Henry Chai

Centering the Data

- To be consistent, we will constrain principal components to be *orthogonal unit vectors* that begin at the origin
- Preprocess data to be centered around the origin:

1. $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$
2. $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{\mu}\ \forall\ n$
3. $X = \begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)^T} \\ \tilde{\boldsymbol{x}}^{(2)^T} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(N)^T} \end{bmatrix}$

Slide Credit: Henry Chai

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a **unit** vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(n)} = \left(\frac{\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}}{\|\boldsymbol{v}\|_2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Length of projection

Direction of projection

Slide Credit: Henry Chai

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a unit vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(n)} = \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}$$

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$\begin{aligned}
&\left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2 \\
&= {\tilde{\boldsymbol{x}}^{(n)}}^T \tilde{\boldsymbol{x}}^{(n)} - 2\left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)} + \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \boldsymbol{v} \\
&= {\tilde{\boldsymbol{x}}^{(n)}}^T \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)} \\
&= \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2
\end{aligned}$$

Slide Credit: Henry Chai

Min. Reconstruction Error ⇔ Max. the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \sum_{n=1}^{N} \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2 \leftarrow \text{Variance of projections } (\tilde{\boldsymbol{x}}^{(n)} \text{ are centered})$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T \left(\sum_{n=1}^{N} \tilde{\boldsymbol{x}}^{(n)} \tilde{\boldsymbol{x}}^{(n)^T} \right) \boldsymbol{v}$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X) \boldsymbol{v}$$

Slide Credit: Henry Chai

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{v}, \lambda) &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\|\boldsymbol{v}\|_2^2 - 1) \\ &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\boldsymbol{v}^T \boldsymbol{v} - 1)\end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = (X^T X)\boldsymbol{v} - \lambda \boldsymbol{v}$$

$$\rightarrow (X^T X)\hat{\boldsymbol{v}} - \lambda \hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda \hat{\boldsymbol{v}}$$

- $\hat{\boldsymbol{v}}$ is an eigenvector of $X^T X$ and λ is the corresponding eigenvalue! But which one?

Slide Credit: Henry Chai

Maximizing the Variance

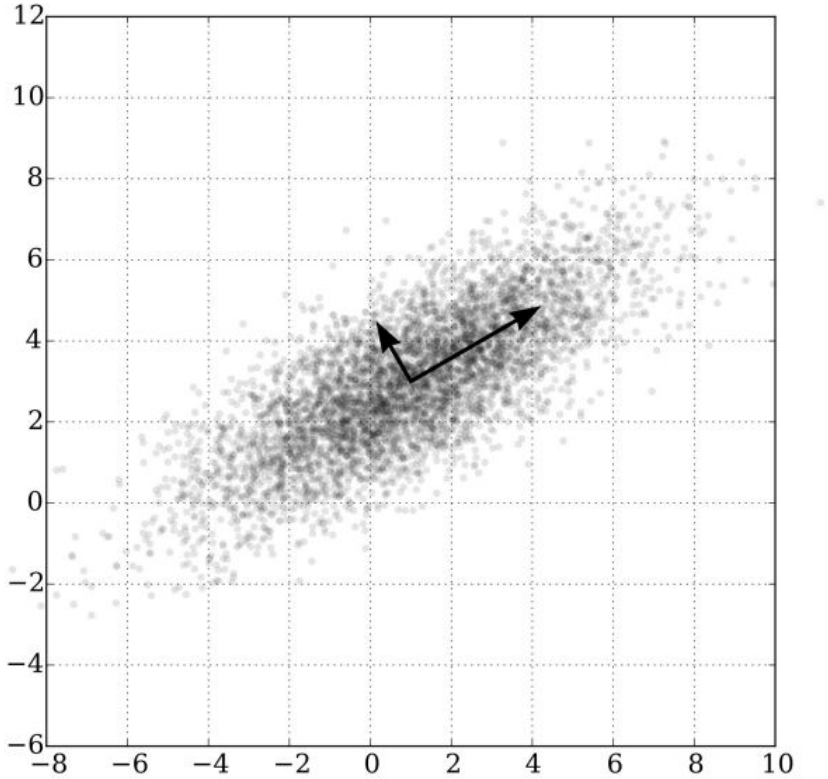
$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$(X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}} \rightarrow \hat{\boldsymbol{v}}^T (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}} = \lambda$$

- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_1
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthogonal
- Etc …
- λ_i is a measure of how much variance falls along $\hat{\boldsymbol{v}}_i$

Slide Credit: Henry Chai

How can we efficiently find principle components (eigenvectos)?



Slide Credit: Henry Chai

Singular Value Decomposition (SVD) for PCA

- Every real-valued matrix $X \in \mathbb{R}^{N \times D}$ can be expressed as

$$X = USV^T$$

where:

1. $U \in \mathbb{R}^{N \times N}$ - columns of U are eigenvectors of XX^T
2. $V \in \mathbb{R}^{D \times D}$ - columns of V are eigenvectors of $X^T X$
3. $S \in \mathbb{R}^{N \times D}$ - diagonal matrix whose entries are the eigenvalues of $X \rightarrow$ squared entries are the eigenvalues of XX^T and $X^T X$

Slide Credit: Henry Chai

PCA Algorithm

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(n)})\}_{n=1}^{N}, \rho$

1. Center the data
2. Use SVD to compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = XV_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

Slide Credit: Henry Chai

How many PCs should we use?

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(n)})\}_{n=1}^{N}, \rho$

1. Center the data
2. Use SVD to compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = XV_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

Slide Credit: Henry Chai

Choosing the number of PCs

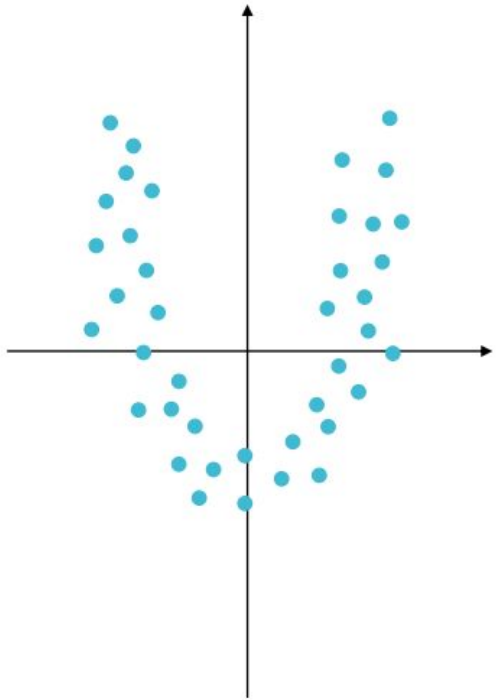
- Define a percentage of explained variance for the i^{th} PC:

$$\lambda_i \Big/ \sum \lambda_j$$

- Select all PCs above some threshold of explained variance, e.g., 5%
- Keep selecting PCs until the total explained variance exceeds some threshold, e.g., 90%
- Evaluate on some downstream metric

Slide Credit: Henry Chai

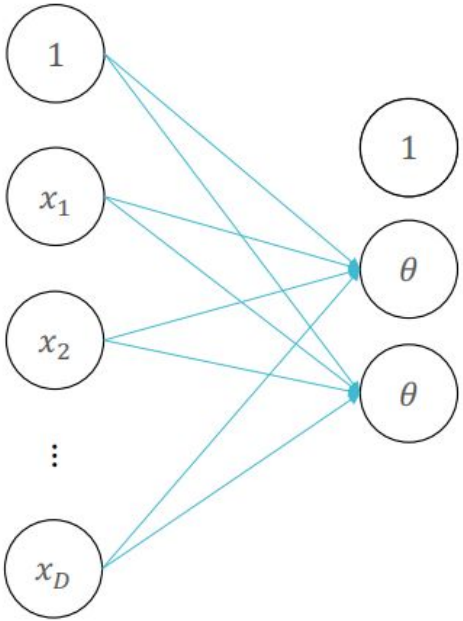
Shortcomings of PCA



- Principal components are orthogonal (unit) vectors
- Principal components can be expressed as linear combinations of the data

Slide Credit: Henry Chai

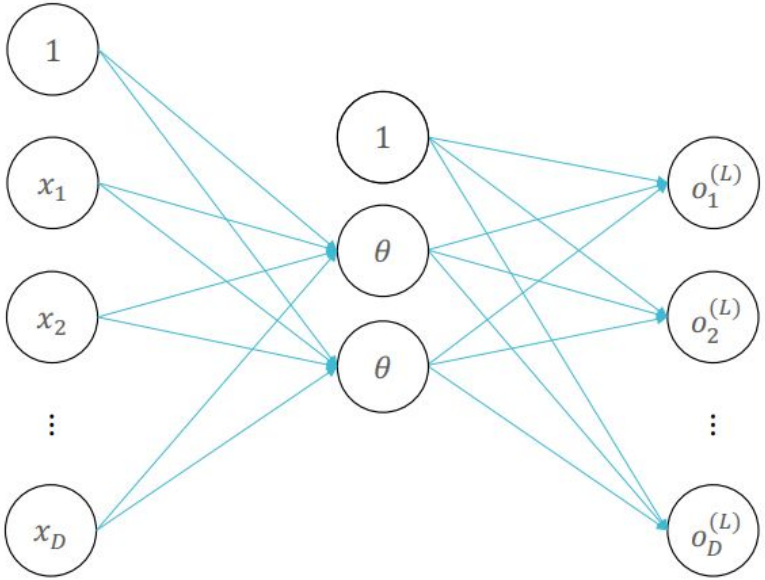
Autoencoders



Insight: neural networks implicitly learn low-dimensional representations of inputs in hidden layers

Slide Credit: Henry Chai

Autoencoders

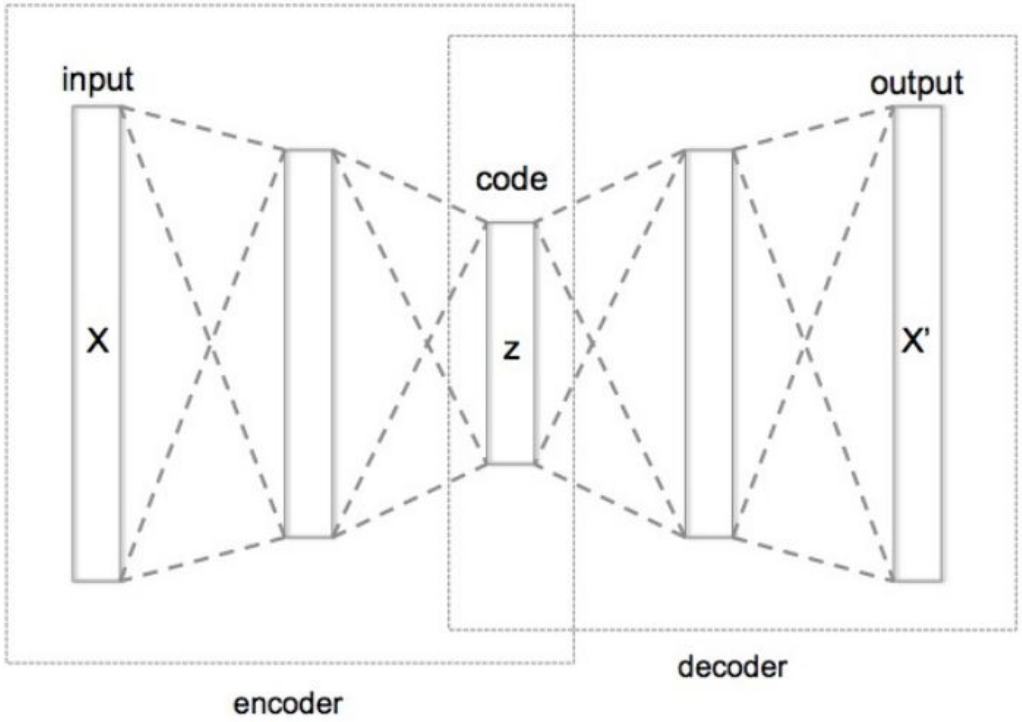


- Learn the weights by minimizing the reconstruction loss:

$$e(\boldsymbol{x}) = \left\| \boldsymbol{x} - \boldsymbol{o}^{(L)} \right\|_2^2$$

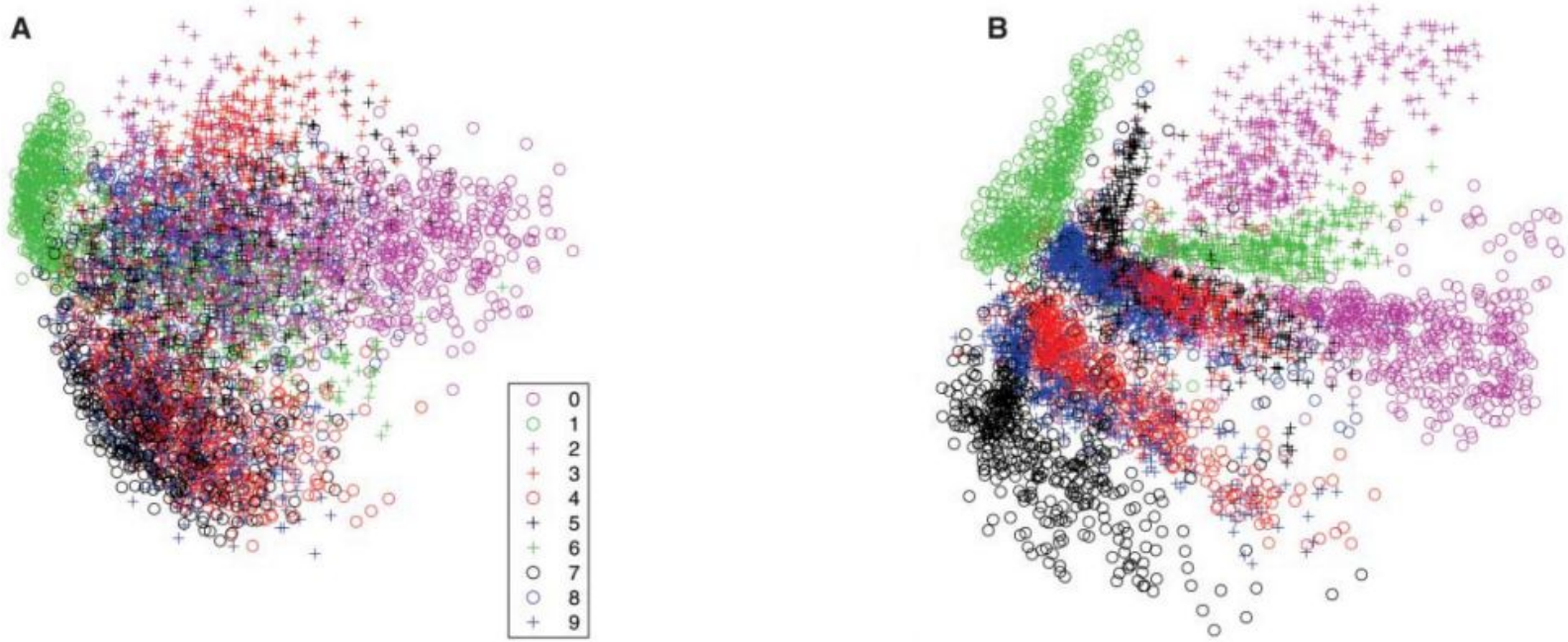
Slide Credit: Henry Chai

Deep Autoencoders



Slide Credit: Henry Chai

PCA (A) vs Autoencoder (B)



Slide Credit: Henry Chai

Takeaways

- PCA finds an orthonormal basis where the first principal component maximizes the variance $\Leftrightarrow$ minimizes the reconstruction error
- Autoencoders use neural networks to automatically learn a latent representation that minimizes the reconstruction error

Slide Credit: Henry Chai

Clustering

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, r^{(n)}\right\}_{n=1}^{N}$

Slide Credit: Henry Chai

Clustering

- Goal: split an unlabeled data set into groups or clusters of "similar" data points
- Use cases:
  - Organizing data
  - Discovering patterns or structure
  - Preprocessing for downstream machine learning tasks
- Applications:
  - Finding similar customer profiles for marketing
  - Reduce the number of data points via sampling from clusters
  - Use clustering to generate labels
  - Doing visualization

Slide Credit: Henry Chai

Similarity for k-nn

- Intuition: ~~predict the label of a data point to be the label of the "most similar" training point~~ two points are "similar" if the distance between them is small
- Euclidean distance: $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2$

Slide Credit: Henry Chai

Clustering Algorithms

- Hierarchical
  - Top-down (divisive)
  - Bottom-up (agglomerative)
- Partitioning
  - K-means

Slide Credit: Henry Chai

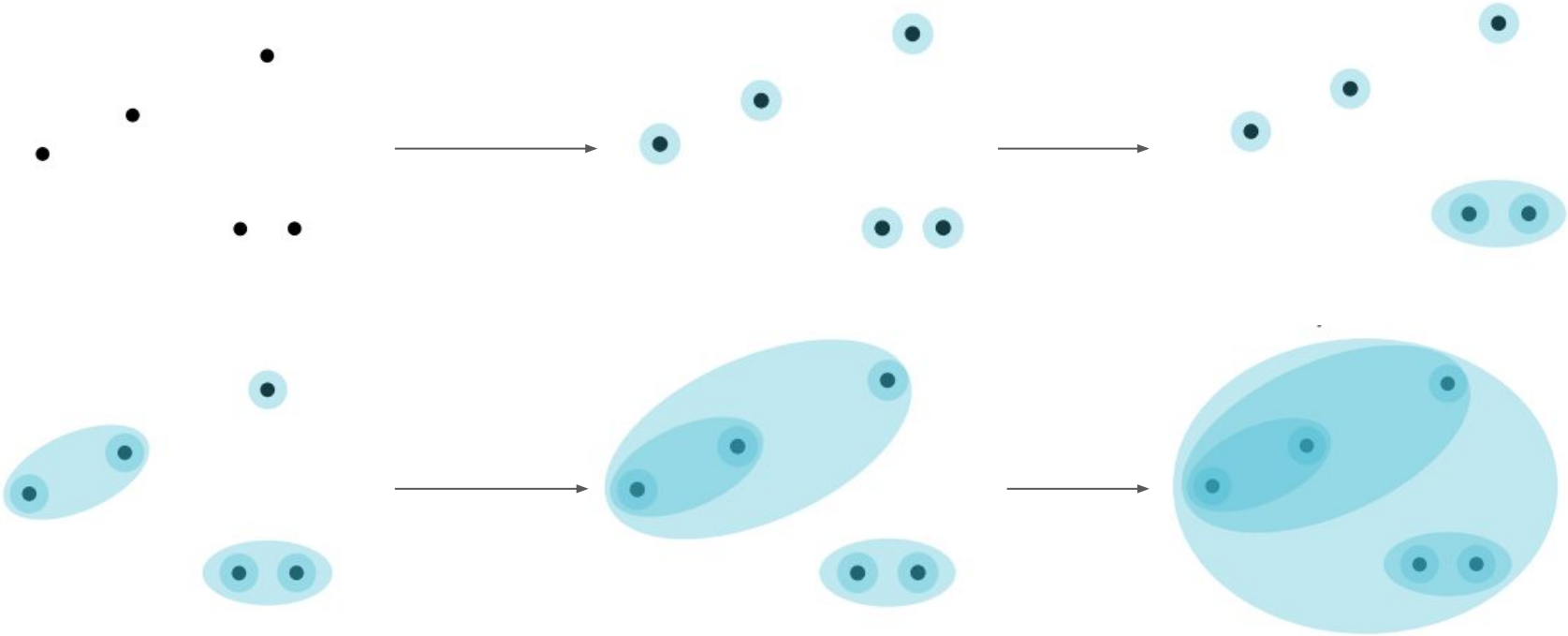
Hierarchical Clustering

- Bottom-up (agglomerative)
  - Start with each data point in its own cluster
  - Iteratively combine the most similar clusters
  - Stop when all data points are in a single cluster

- Top-down (divisive)
  - Start with all data points in one cluster
  - Iteratively split the largest cluster into two clusters
  - Stop when all clusters are single data points

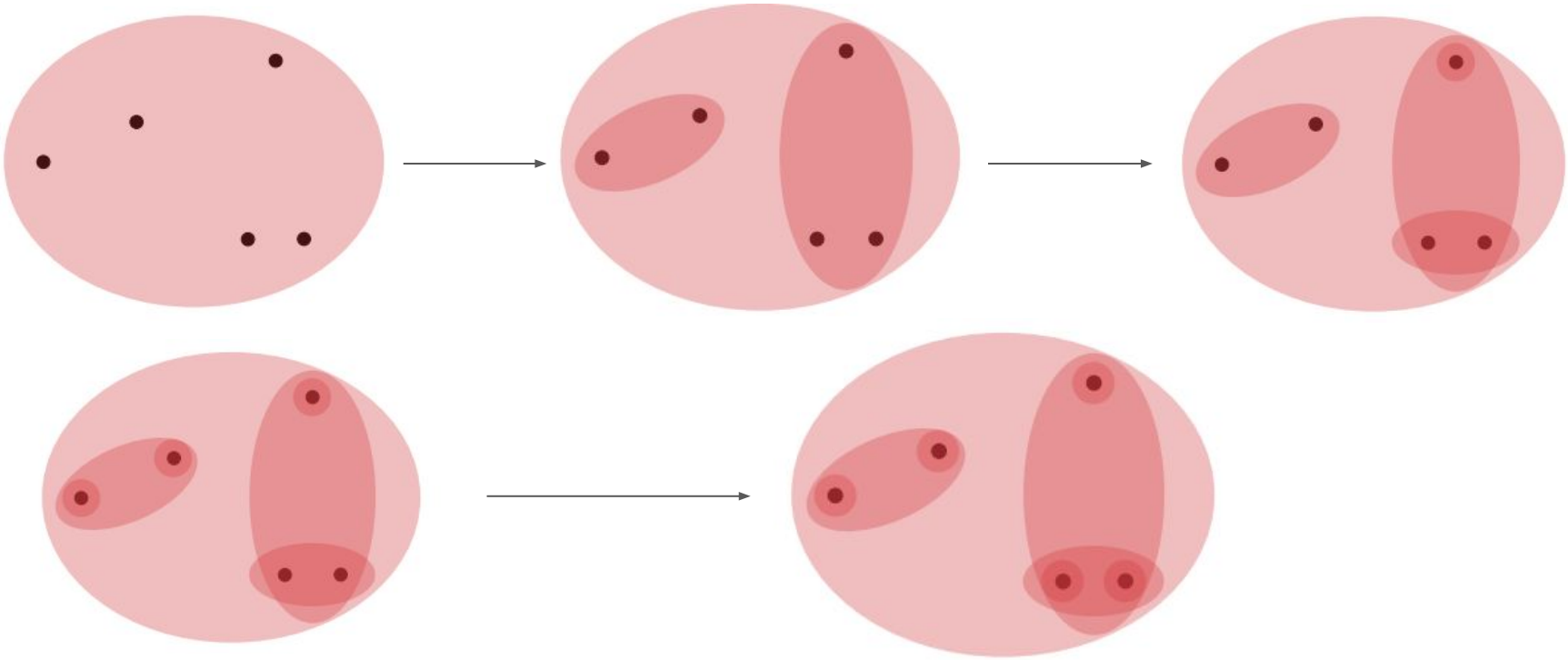
Slide Credit: Henry Chai

Bottom-up (agglomerative)



Slide Credit: Henry Chai

Top-down (divisive)



Slide Credit: Henry Chai

Bottom-up Hierarchical Clustering

- Bottom-up (agglomerative)
  - Start with each data point in its own cluster
  - Iteratively combine the ***most similar*** clusters
  - Stop when all data points are in a single cluster
- Key question: how do we define similarity between clusters?
  - Single-linkage: consider the closest data points

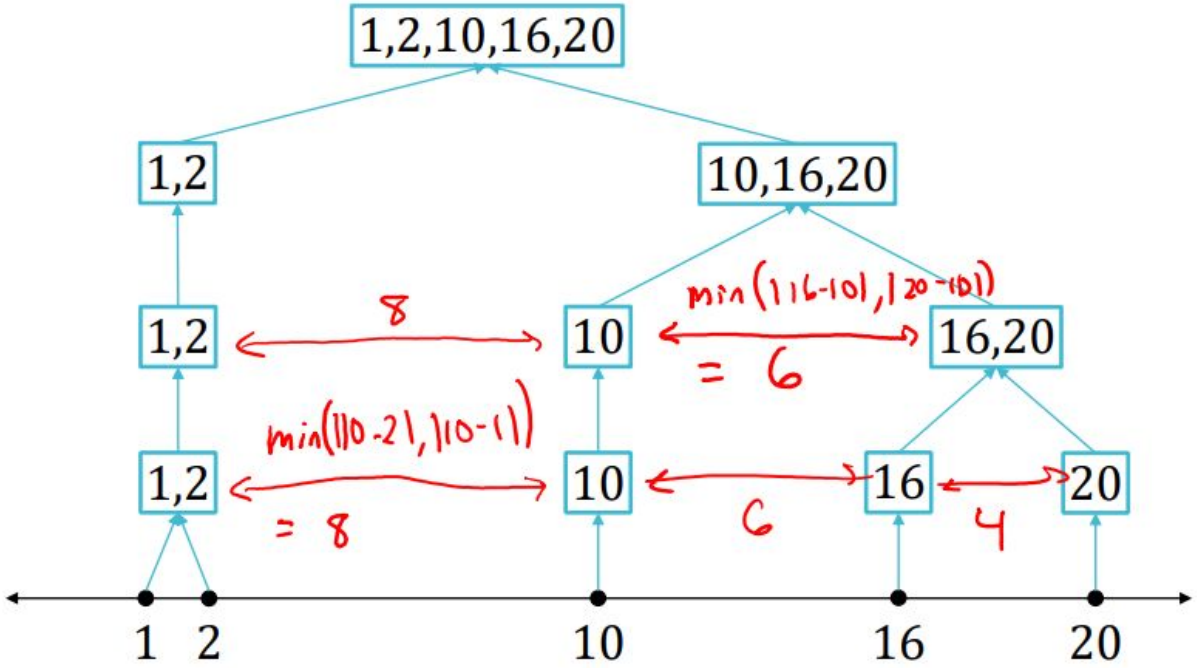
$$d_{SL}(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})$$

  - Complete-linkage: consider the farthest data points

$$d_{CL}(C_i, C_j) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})$$

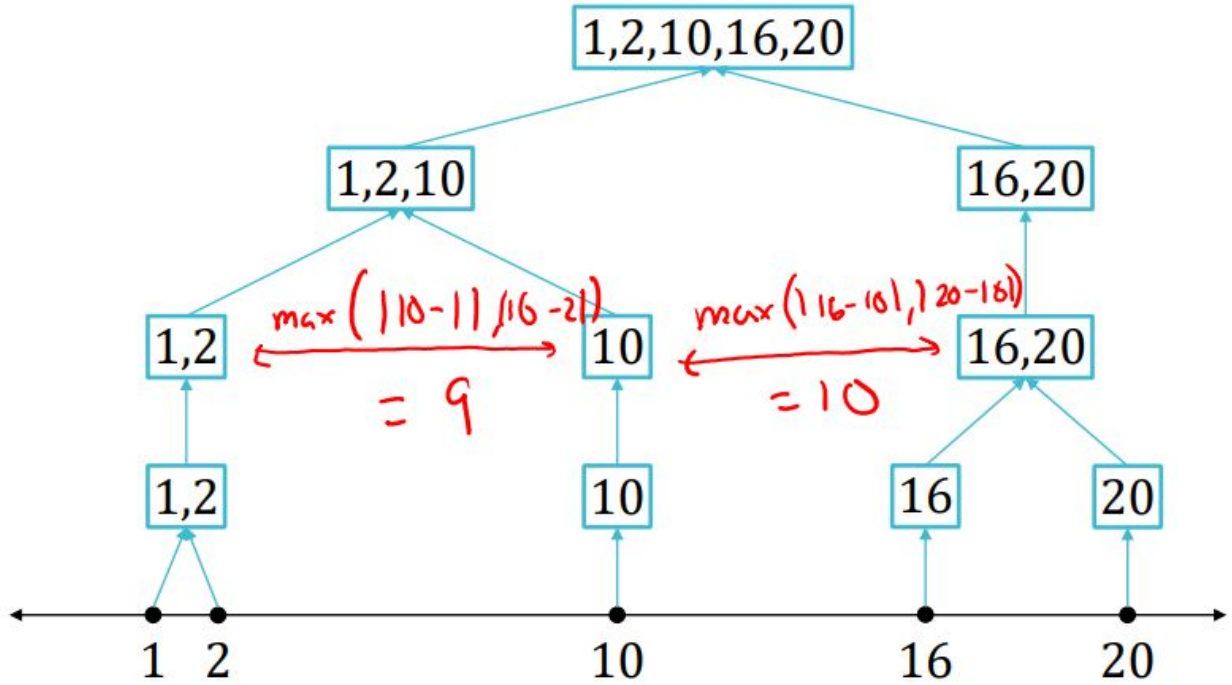
Slide Credit: Henry Chai

Single Linkage Dendrogram



Slide Credit: Henry Chai

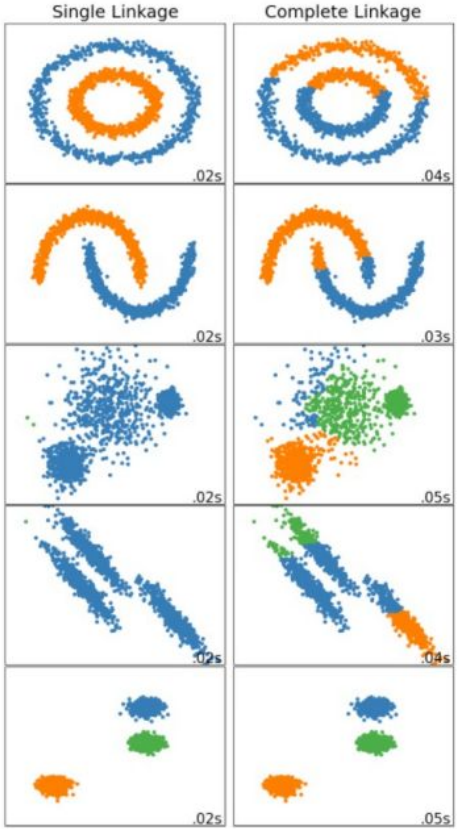
Complete Linkage Dendrogram



Slide Credit: Henry Cha

Single vs Complete Linkage

- Single-linkage prioritizes local behaviour, can lead to long, snakelike (i.e., nonconvex) clusters.
- Complete-linkage tries to keep the diameter of clusters small; clusters tend to be more spherical (i.e., convex)



Slide Credit: Henry Chai

Top-down Hierarcihcal Clustering

- Top-down (divisive)
  - Start with all data points in one cluster
  - Iteratively split the largest cluster into two clusters
  - Stop when all clusters are single data points

- **Key question**: how can we partition a cluster?
  - K-Means with K = 2

Slide Credit: Henry Chai

Partition-Based Clustering

- Given a desired number of clusters, K, return a partition of the data set into K groups or clusters, $\{C_1, \dots, C_K\}$, that optimize some objective function
  - Can be used as a subroutine for top-down hierarchical clustering when $K = 2$

1. What objective function should we optimize?

2. How can we perform optimization in this setting?

Slide Credit: Henry Chai

Which partition is better?

Option A

Option B

Slide Credit: Henry Chai

Recipe for K-means

- Define a model and model parameters
  - Assume K clusters and use the Euclidean distance
  - Parameters: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and $z^{(1)}, \ldots, z^{(N)}$
- Write down an objective function

$$\sum_{n=1}^{N} \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu}_{z^{(n)}} \right\|_2$$

- Optimize the objective w.r.t. the model parameters
  - Use (block) coordinate descent

Slide Credit: Henry Cha

Optimizing the K-means objective

$$\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K, z^{(1)}, \ldots, z^{(\mathrm{N})} = \operatorname{argmin} \sum_{n=1}^{N} \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu}_{z^{(n)}} \right\|_2$$

- If $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ are fixed

$$\hat{z}^{(n)} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmin}} \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k \right\|_2$$

- If $z^{(1)}, \ldots, z^{(N)}$ are fixed

$$\hat{\boldsymbol{\mu}}_k = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{n : z^{(n)} = k} \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu} \right\|_2$$

$$= \frac{1}{N_k} \sum_{n : z^{(n)} = k} \boldsymbol{x}^{(n)}$$

Slide Credit: Henry Chai

K-means Algorithm

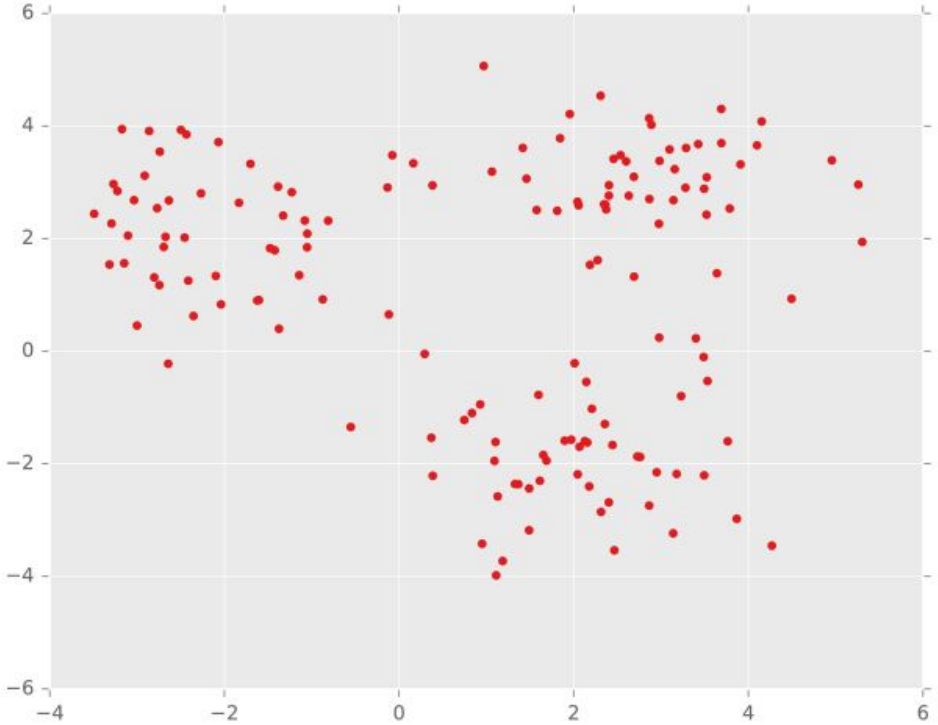
- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}\right)\right\}_{n=1}^{N}, K$

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
2. While NOT CONVERGED
   a. Assign each data point to the cluster with the nearest cluster center:
   $$z^{(n)} = \underset{k}{\operatorname{argmin}} \left\|\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k\right\|_2$$
   b. Recompute the cluster centers:
   $$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n\,:z^{(n)}=k} \boldsymbol{x}^{(n)}$$
   where N_k is the number of data points in cluster k

- Output: cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $z^{(1)}, \ldots, z^{(N)}$

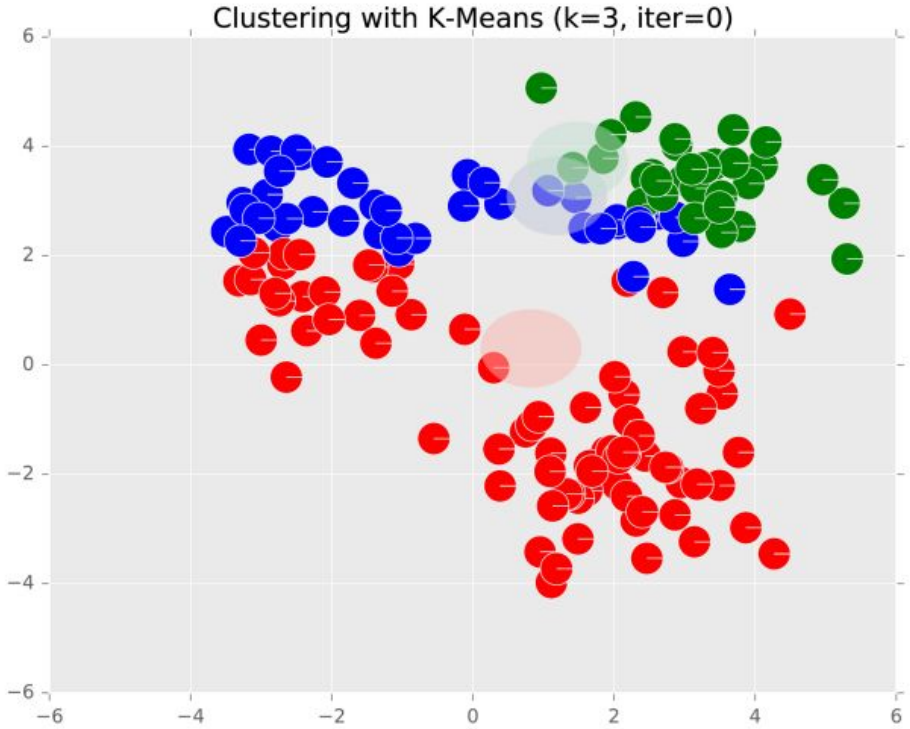
Slide Credit: Henry Chai

K-means: Example (K = 3)



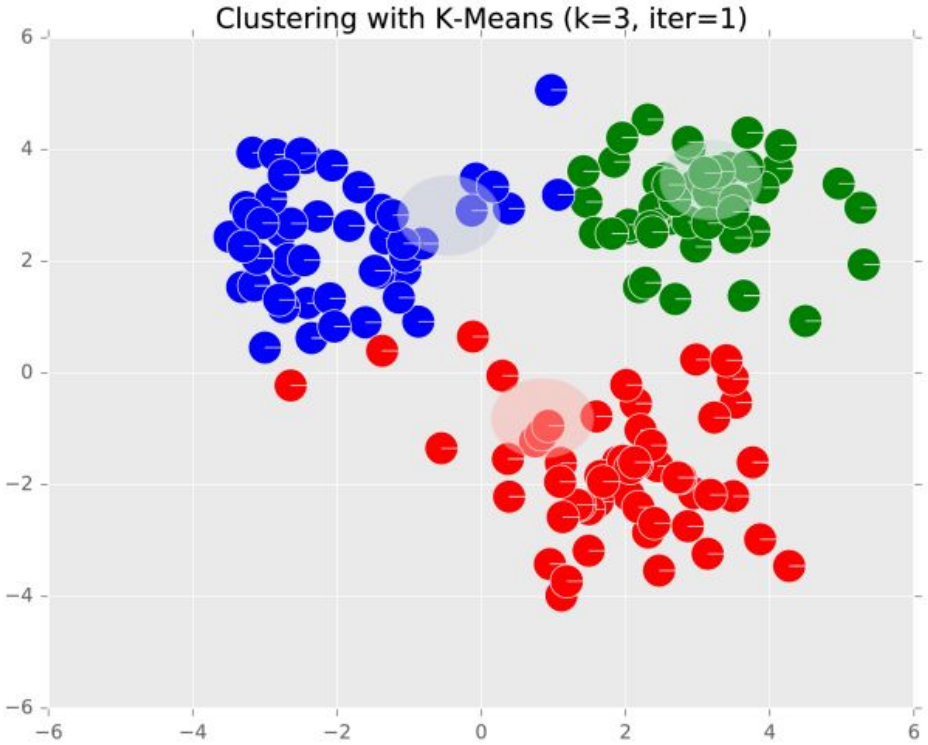
Slide Credit: Henry Chai

K-means: Example (K = 3)



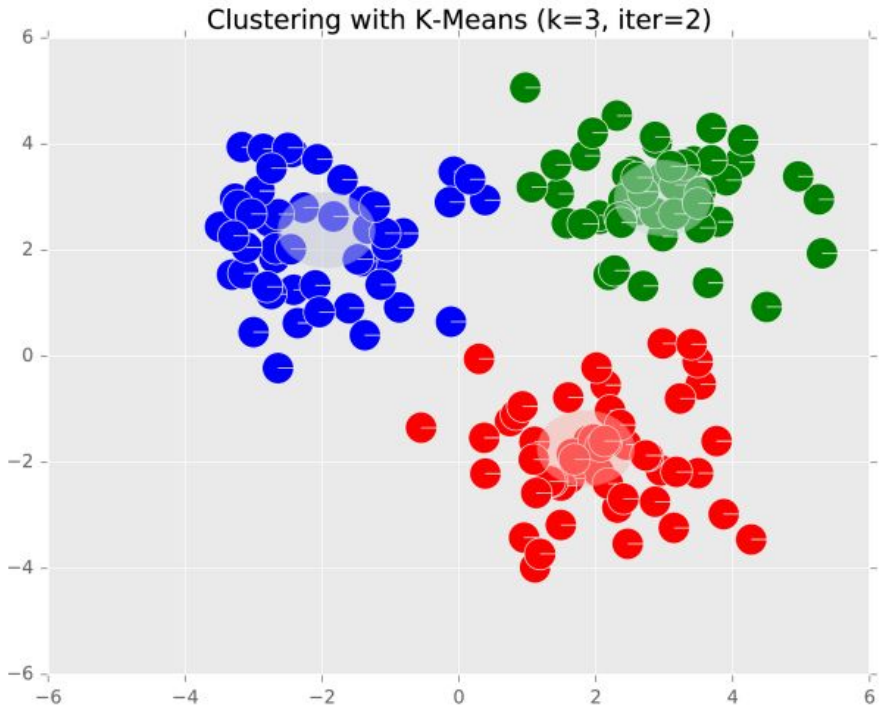
Slide Credit: Henry Chai

K-means: Example (K = 3)



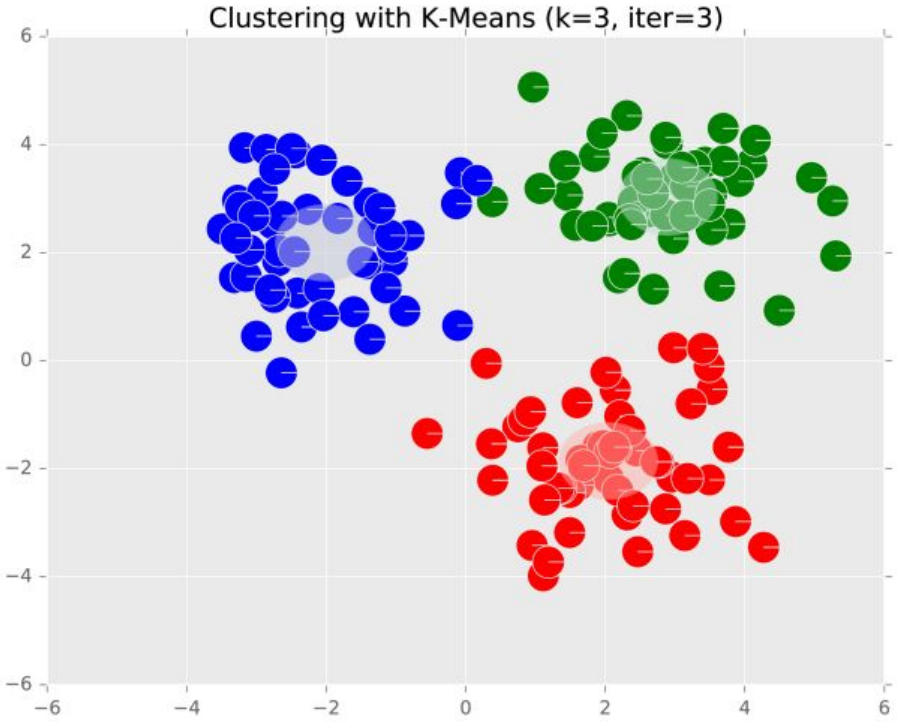
Slide Credit: Henry Chai

K-means: Example (K = 3)



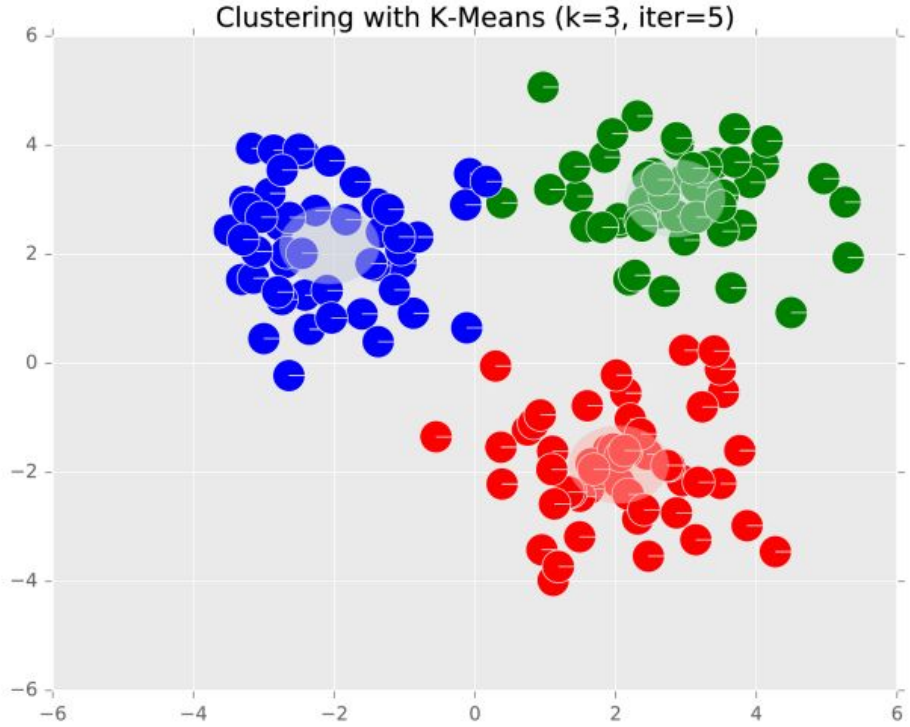
Slide Credit: Henry Chai

K-means: Example (K = 3)



Slide Credit: Henry Chai

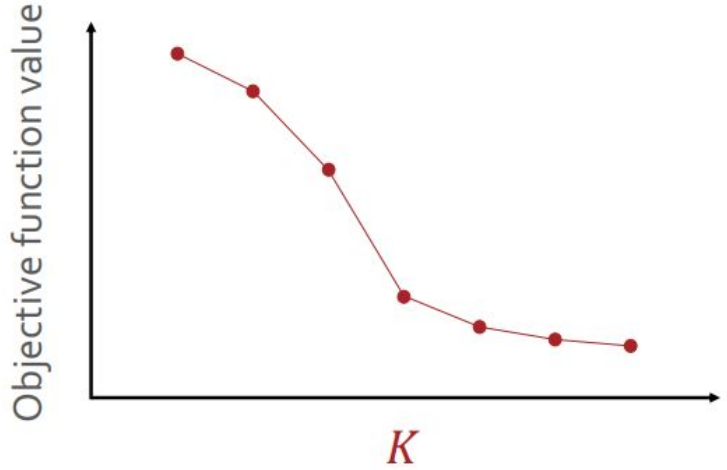
K-means: Example (K = 3)



Slide Credit: Henry Chai

Setting K

- Idea: choose the value of K that minimizes the objective function



- Look for the characteristic "elbow" or largest decrease when going from $K-1$ to K

Slide Credit: Henry Chai

Next Class:

Machine Learning System Design