Unsupervised Learning

Lei Li and Yu-xiang Wang
UCSB

Some slides borrowed from Eric Xing, Arti Singh

Recap

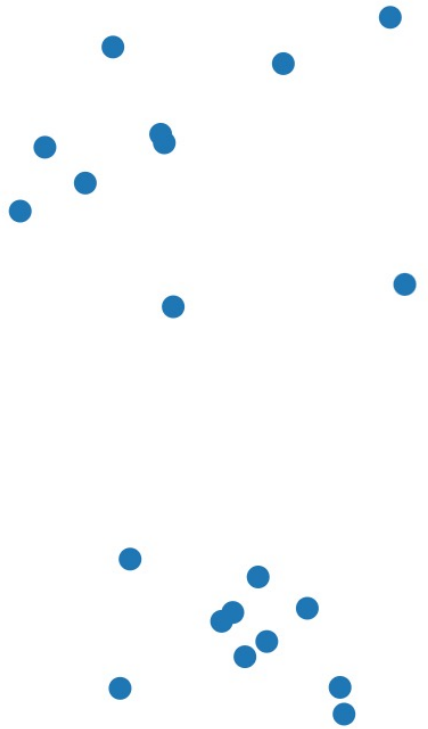
- Linear Regression
- Linear classifier
- Both need pairs of data-labels <x, y> to train
- What if we do not have labels y
  - Can we still train a model to predict data class?

Unsupervised Learning

- Basically finding “patterns” of data
- No golden labels to teach the model
  - Only raw data x, but not y
- Instances:
  - Clustering: Give data samples, find groupings
  - Dimensionality Reduction: Given high-dimensional data, compress them into low-dimensions

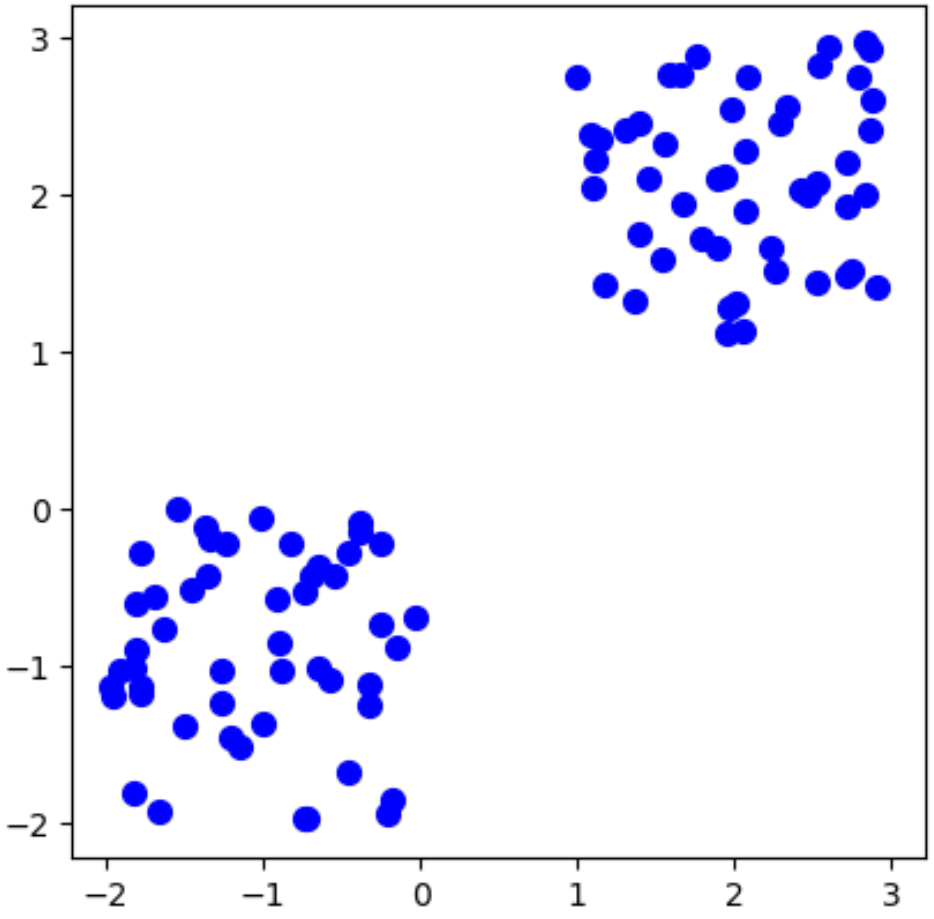
Why Unsupervised Learning

- No human annotated data (too expensive)
- Do not have clear target, but still want to find “meaningful” patterns
- Just fitting the data with most likely distribution



Clustering

- Informally: find natural groups of data
- Organizing data into clusters such that
  - High intra-cluster similarity
  - Low inter-cluster similarity



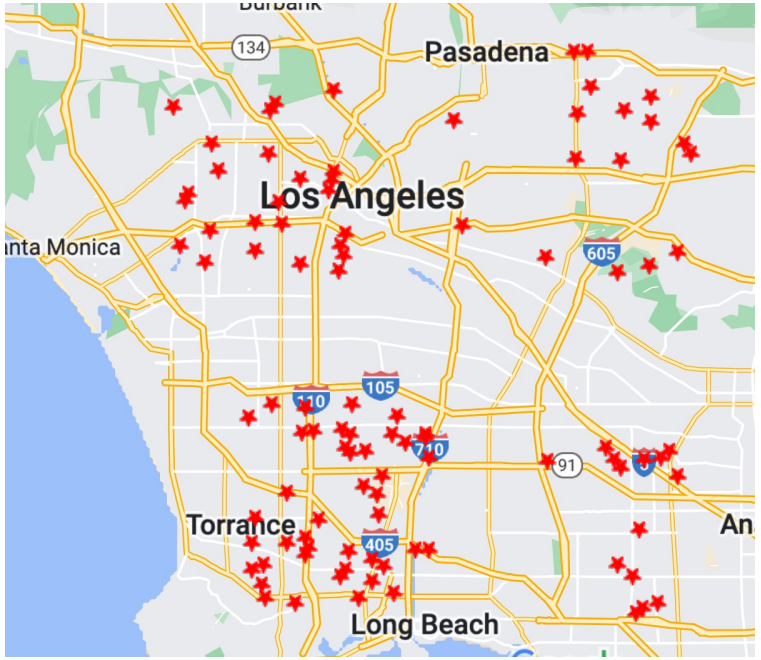
Grouping News Articles

- Benjamin Netanyahu Questioned in Israel Graft Inquiry
- Rockefeller Foundation Picks Rajiv J. Shah, a Trustee, as President
- Iare Hollingworth, Reporter Who Broke News of World War II, Dies at 105
- Danielle Brooks: The First Time I Saw Myself on a Billboard
- For Troubled Student, a Change of School and Direction
- 10 Key Moments and More From Trump's News Conference
- California Today: The Tale of the Laguna Beach Jumper

Choosing Pizza store sites

Pizza Hero wants to open a few stores at Los Angeles.

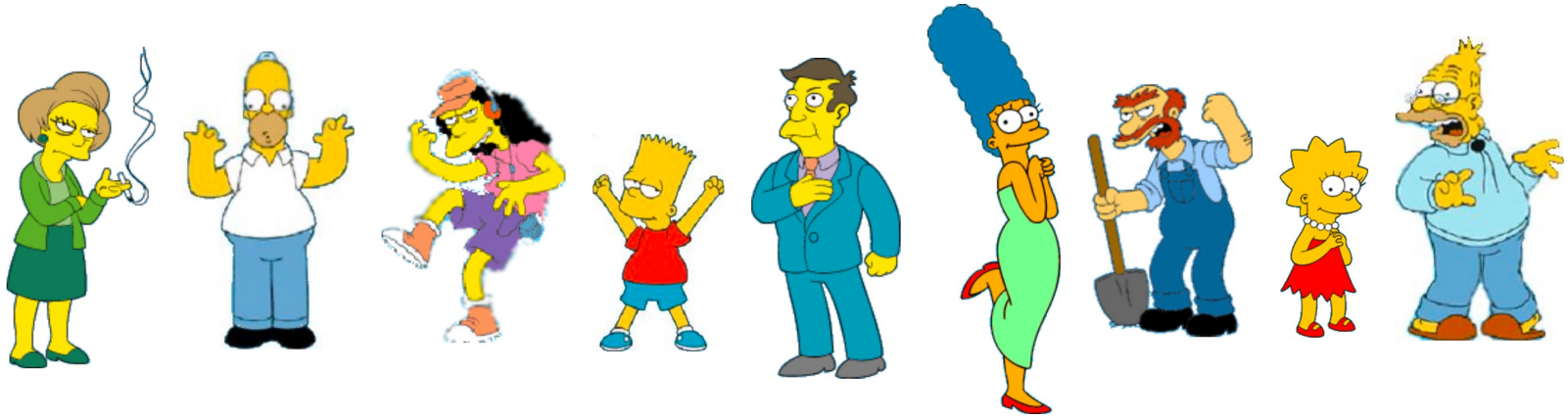
- Through survey they collected pizza ordering requests from locations across the city.
- How to decide the proper sites?



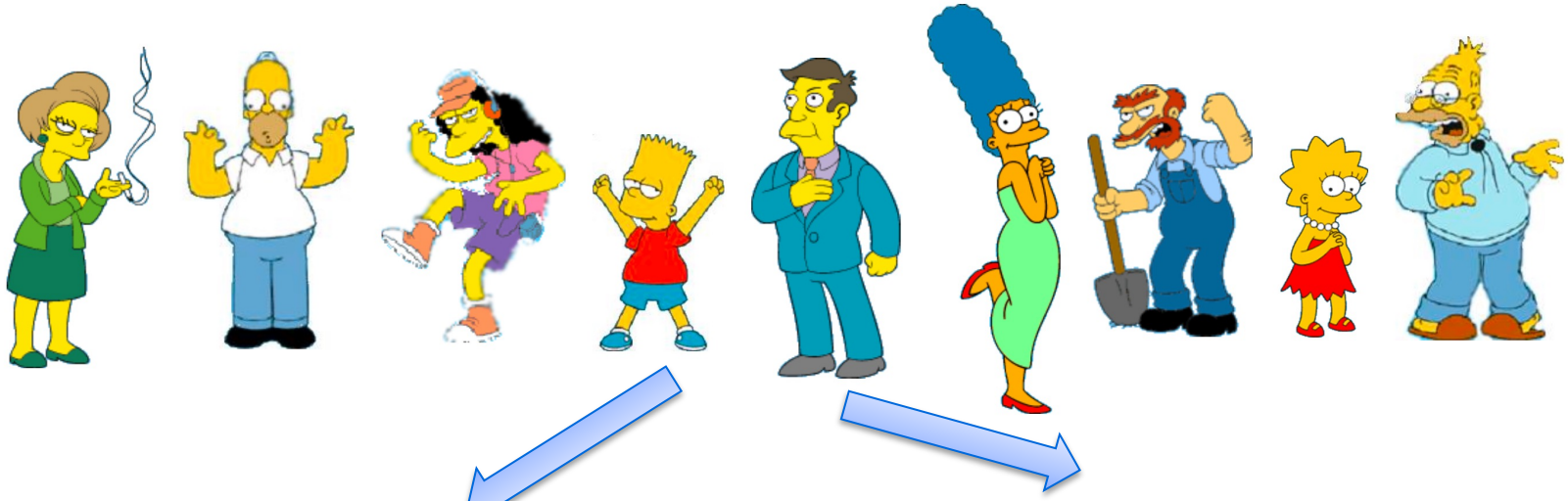
Why Clustering?

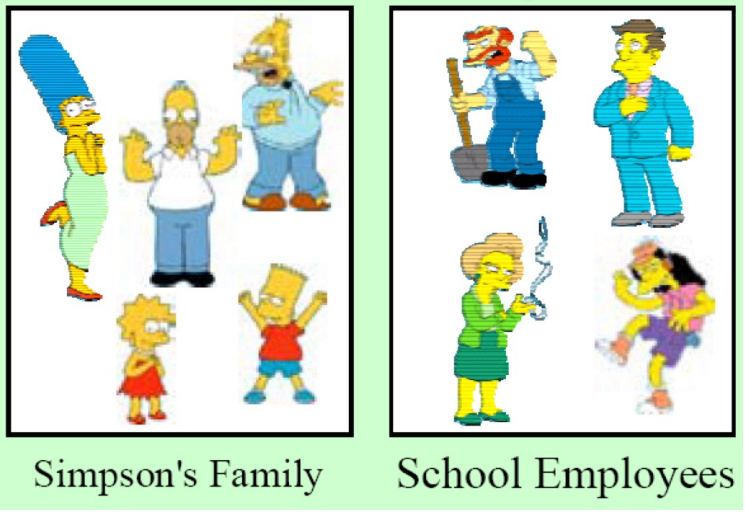
- Organizing data into clusters provides better interpretation of the data
  - Reveal internal structure of data samples
- Partition the data itself can be the goal
  - Image segmentation: separating objects from background
- Knowledge discovery in data
  - E.g. reoccurring patterns, topics, etc.

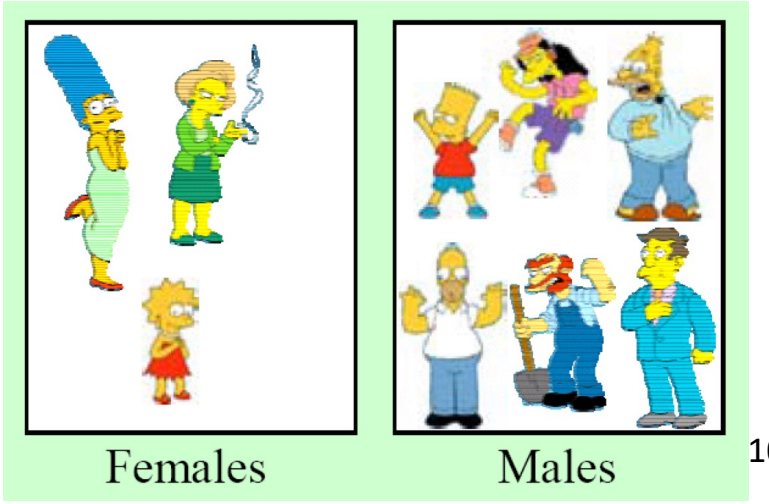
What is a natural grouping among these objects?



Clustering is subjective







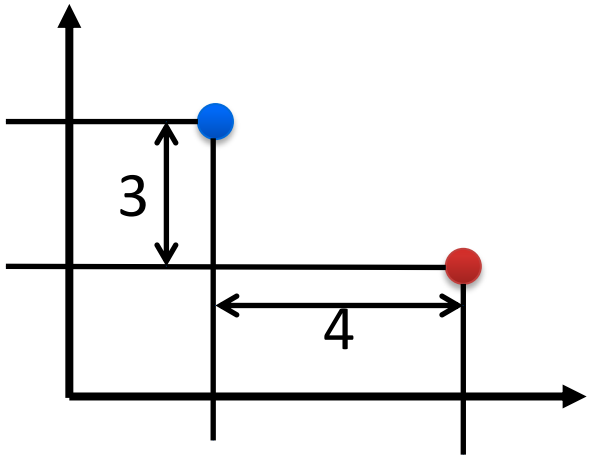
How to measure Similarity?

Similarity is hard to define. But we know it when we see it.

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach - think in terms of a distance (rather than similarity) between vectors or correlations between random variables.

Distance Metrics

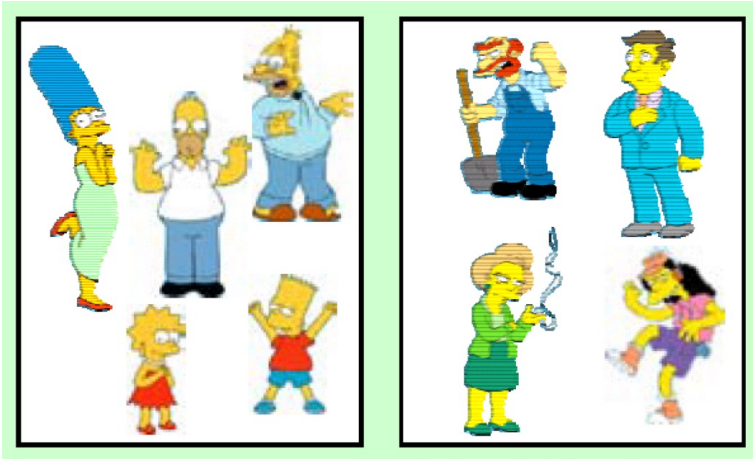
$x = (x_1, x_2, \ldots, x_m)$
$y = (y_1, y_2, \ldots, y_m)$

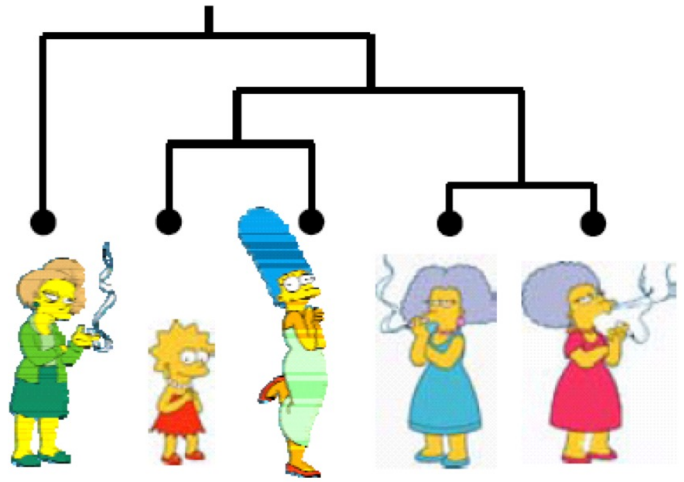


- Euclidean distance $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$ **5**
- Manhattan distance $d(x, y) = \sum_i |x_i - y_i|$ **7**
- Cosine similarity $d(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$

Types of Clustering

- Partition algorithms
  - K-means clustering
  - Spectral clustering
  - Mixture-model
- Hierarchical algorithms
  - Bottom-up
  - Top-down





Desirable Properties of Clustering Algorithm

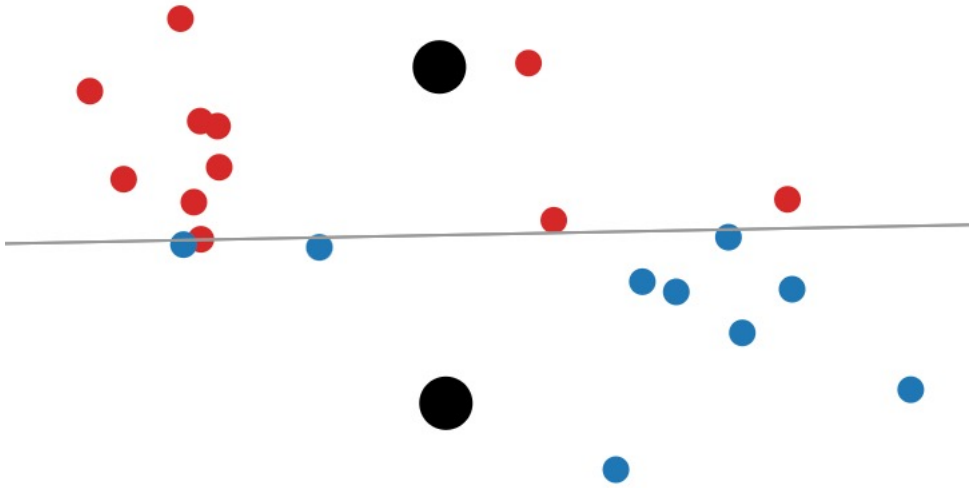
- Scalability
- General
- No requirement for domain knowledge
- Interpretability and Usability

K-Means Clustering

- Group N data samples (m-dimensional) into K non-overlapping groups
- The user has to specify K the number of clusters.

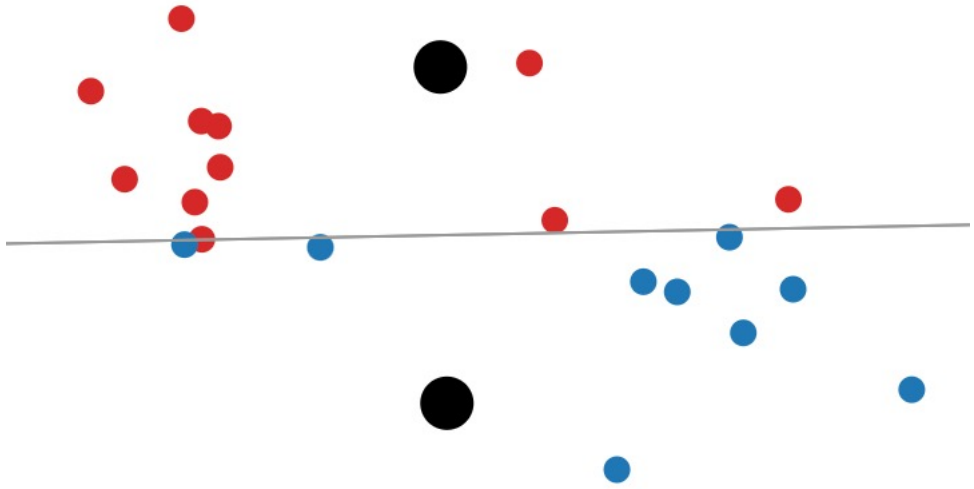
K-Means Clustering Algorithm

1. Start with initial K cluster centers randomly



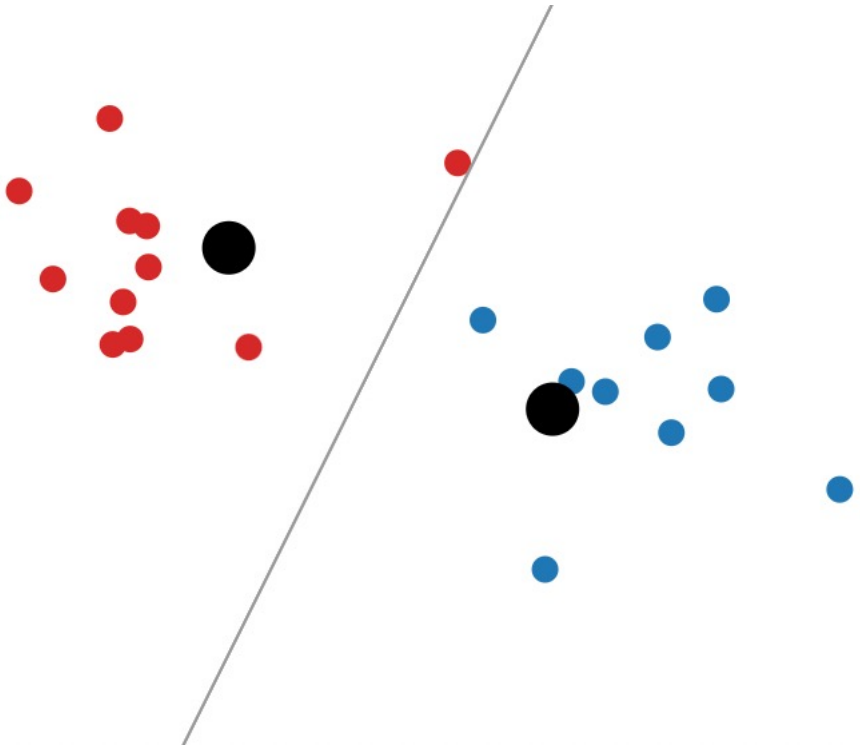
K-Means Clustering Algorithm

2. Assign each data point to its nearest center



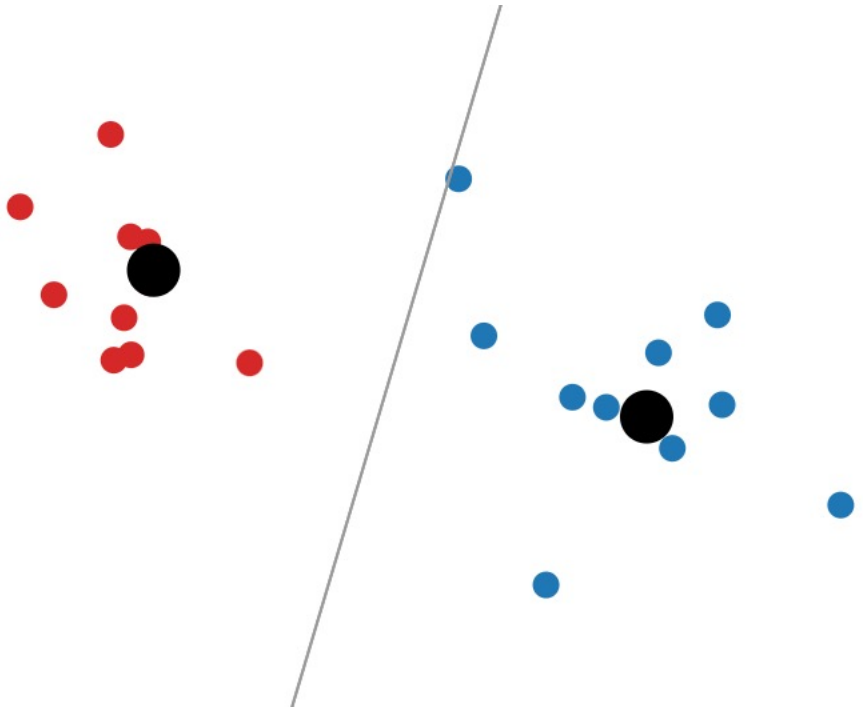
K-Means Clustering Algorithm

3. Update each group center with the mean of its members $\mu_i = \frac{1}{|C_i|} \sum_{j \in C_i} x_j$



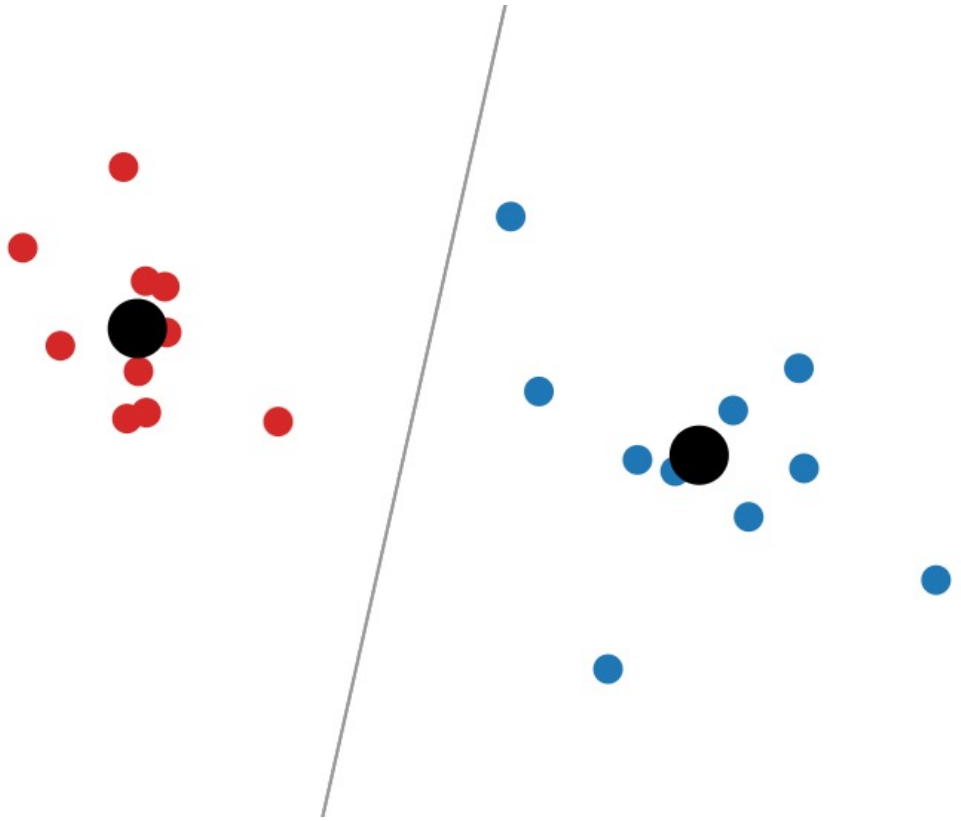
K-Means Clustering Algorithm

4. Repeat steps 2-3 with many iterations.



K-Means Clustering Algorithm

5. Finish when members in clusters do not change.



K-Means Algorithm

1. Start with initial K cluster centers randomly
2. Assign each data point to its nearest center using distance function
3. Update each group center with the mean of its members
4. Repeat steps 2-3 with many iterations until members in clusters do not change.

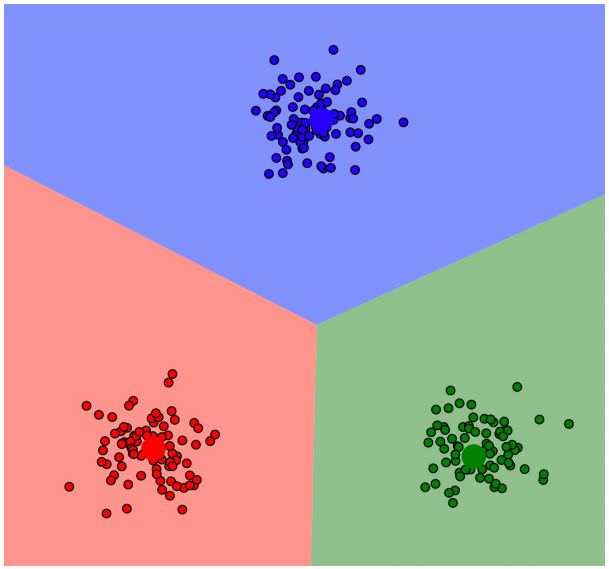
Demo:https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Variation – K-Medoids

1. Start with initial K cluster centers randomly
2. Assign each data point to its nearest center using distance function
3. Update the center with the point bearing the smallest total distance to all other points in the same cluster
4. Repeat steps 2-3 with many iterations until members in clusters do not change.

Why K-Means Works?

- What is a good partition?
  - High intra-cluster similarity
- K-Means optimizes
  - The total sum of distance from members to the cluster centers



$$\mathrm{f}(\mu_1 \ldots \mu_k, C) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

- Optimal solution

$$\min_{\mu} \min_{C} f(\mu, C)$$

K-Means algorithm

- Optimize the objective

$$\min_{\mu} \min_{C} f(\mu, C) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

- K-Means is a *coordinate descent* algorithm
  1. (cluster assignment) Fix μ, optimize f w.r.t. C
  2. (cluster centering) Fix C, optimize f w.r.t. μ
- We will revisit this style of algorithms later (e.g. EM alg. for Gaussian Mixture Model)

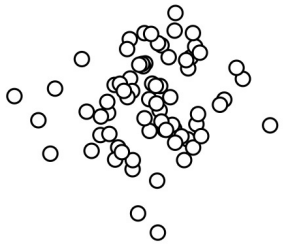
Quiz

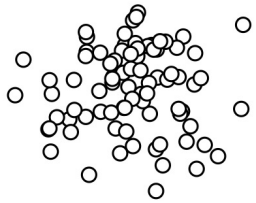
- On Edstem

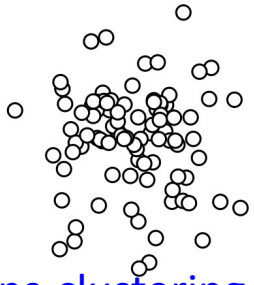
Seed Initialization

Results are quite sensitive to seed selection

Does k-means always succeed?





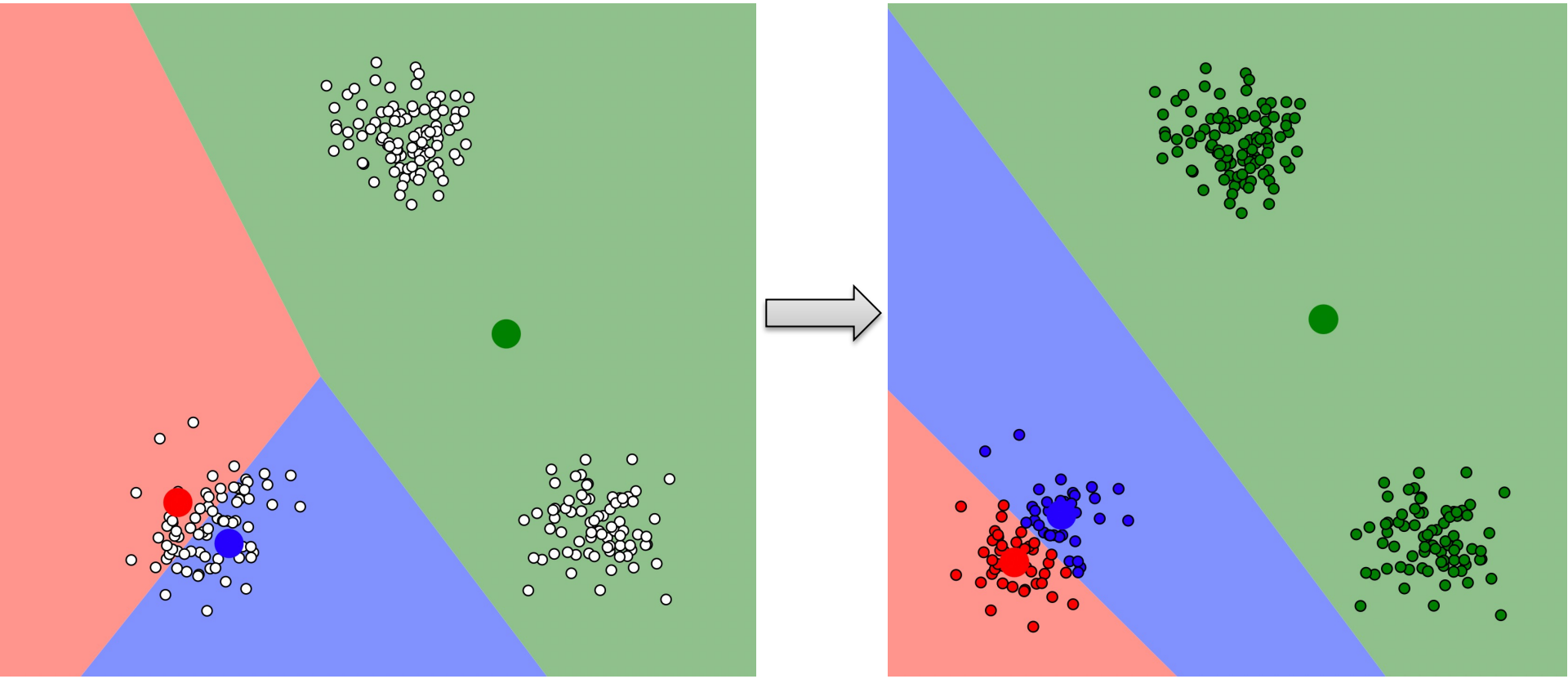


https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Try example for Gaussian Mixture

Seed Initialization

Results are quite sensitive to seed selection



https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Try example for Gaussian Mixture

Seed Initialization

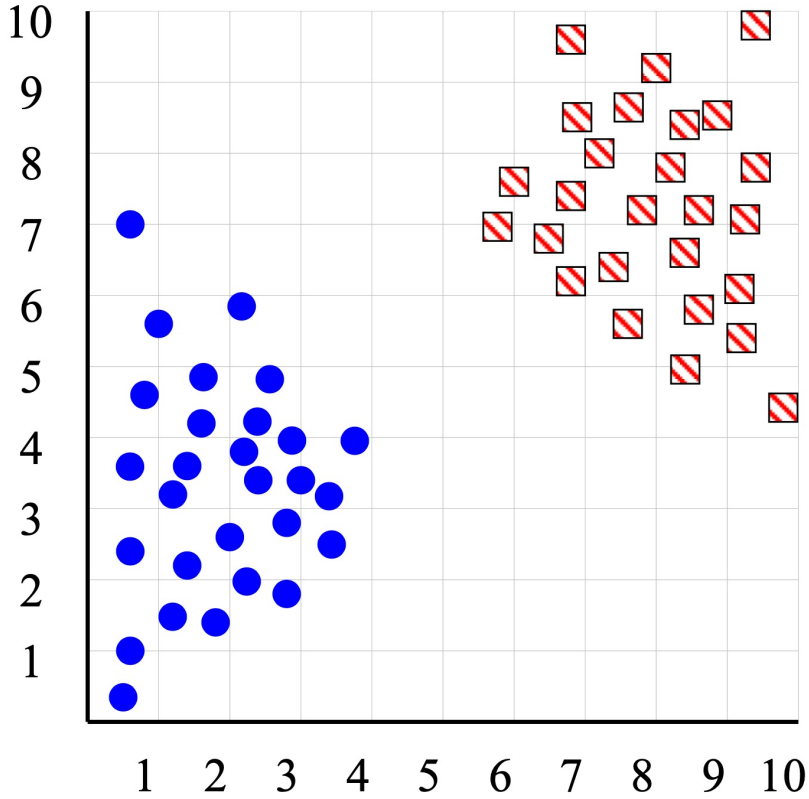
- Results can vary based on initial cluster assignments
- Some initializations can result in poor convergence rate, or converge to a sub-optimal result
  - Try multiple starting points (very important!!!)
  - K-means++ algorithm
    - Key idea: initialize centers that are far apart

David Arthur and Sergei Vassilvitskii, K-means++: the advantages of careful seeding. SODA 2007

How to decide the number of clusters?

- What is the objective for different k?

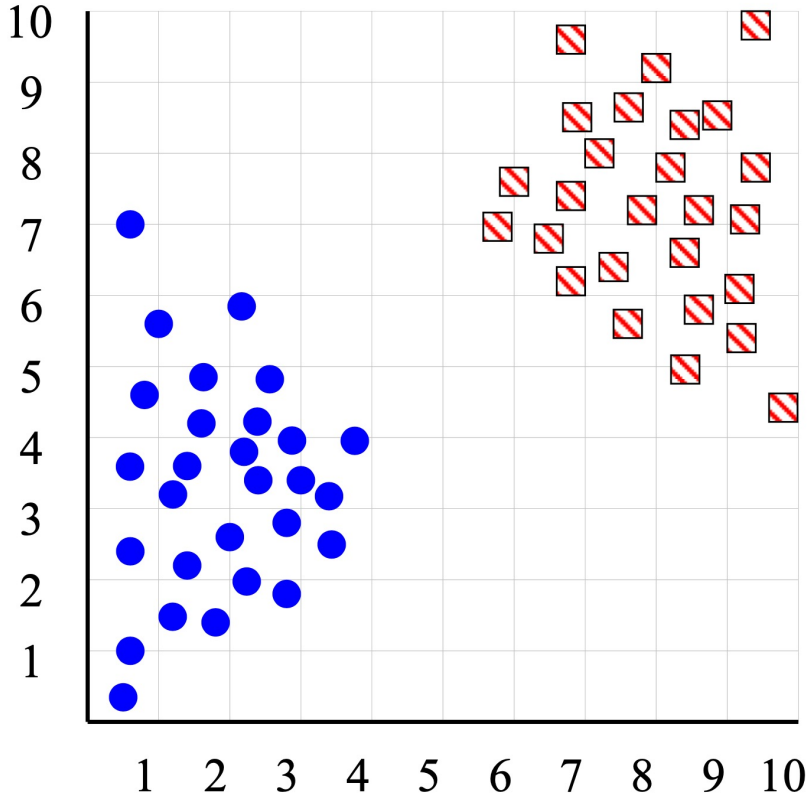
$$s = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$$



How to decide the number of clusters?

$$s = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$$

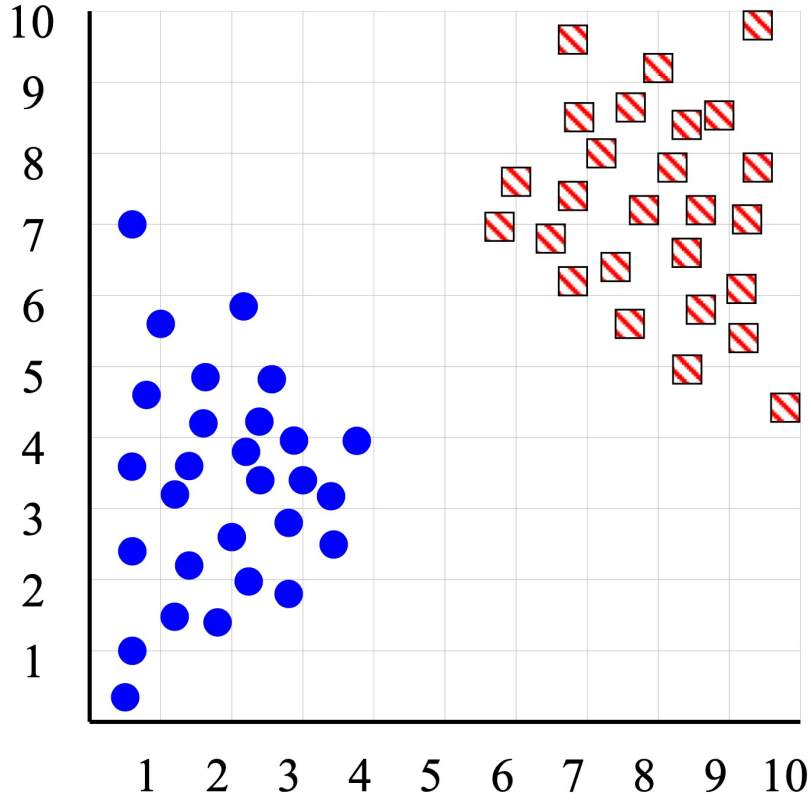
When $k = 1$, the objective s is 873.0



How to decide the number of clusters?

$$s = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$$

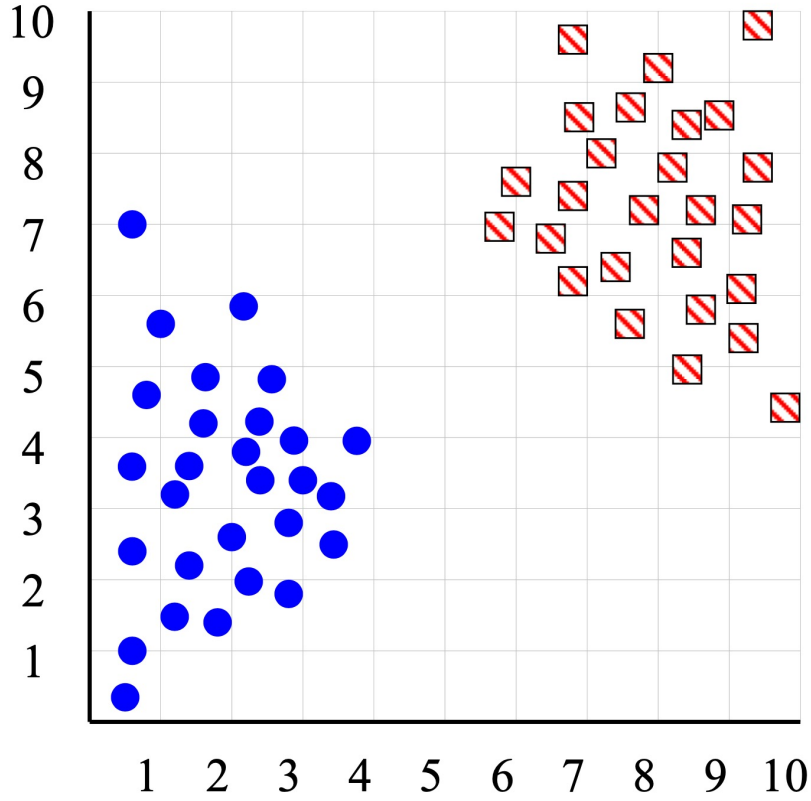
When $k = 2$, the objective s is 173.1



How to decide the number of clusters?

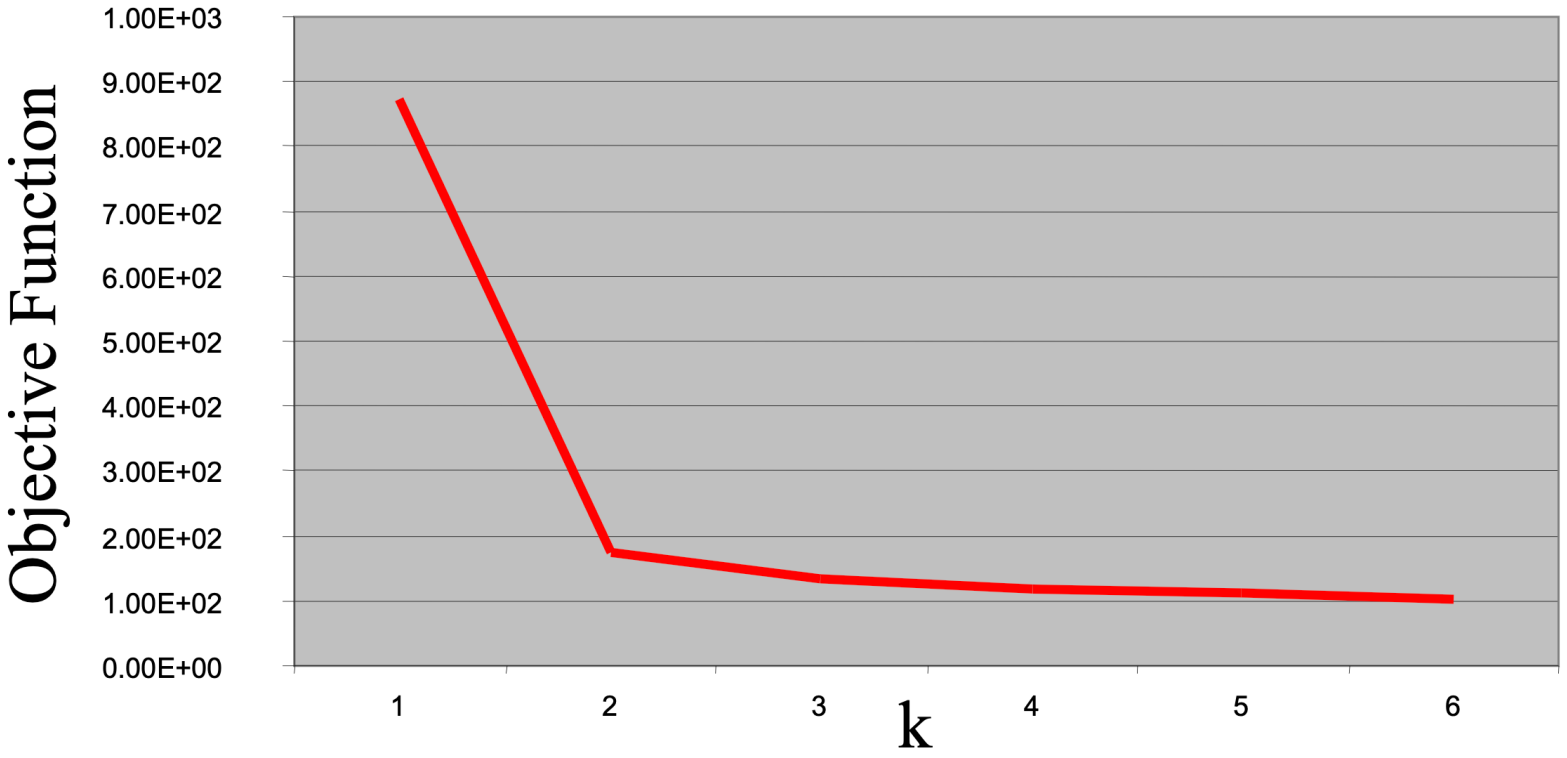
$$s = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$$

When $k = 3$, the objective s is 133.6

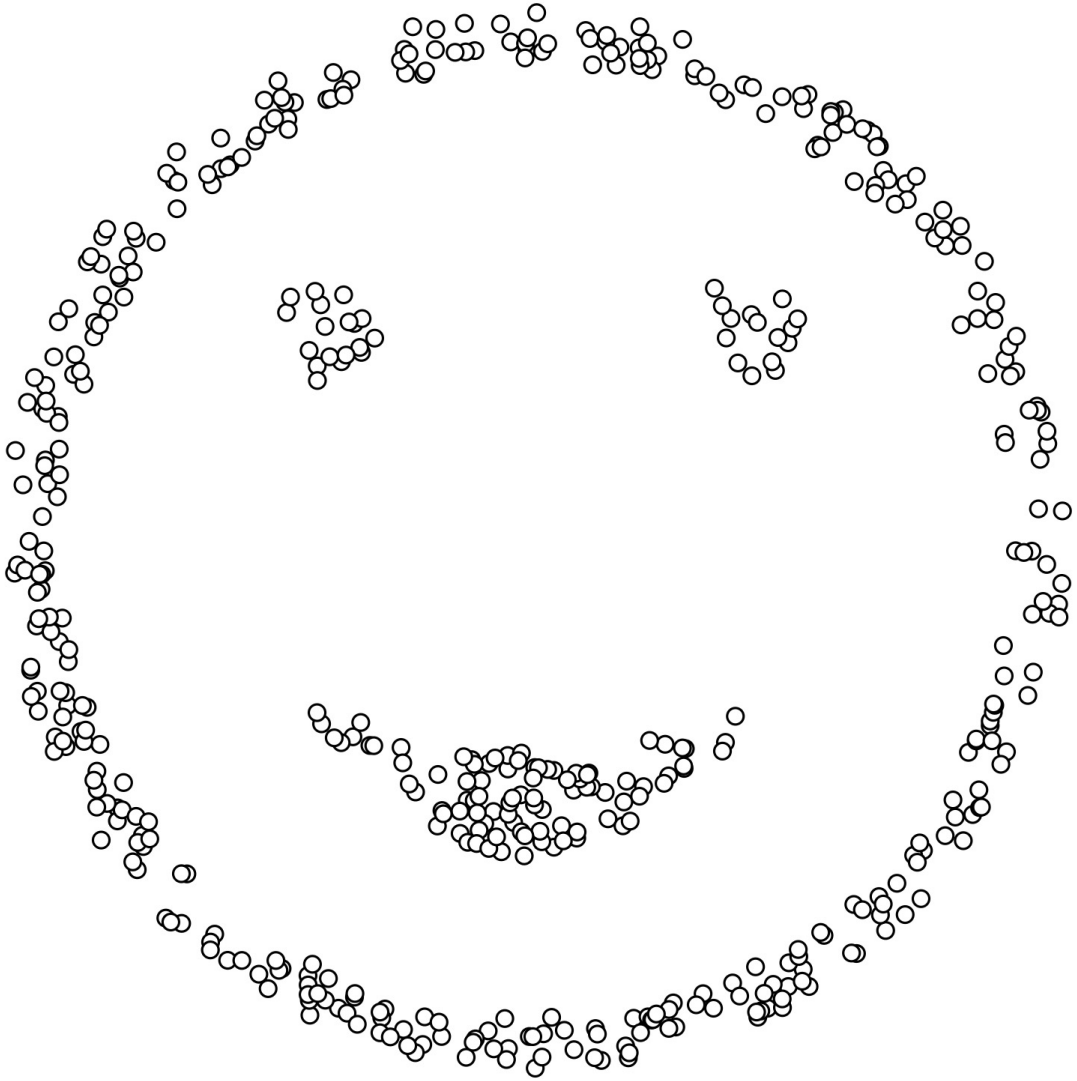


Is larger K always better?

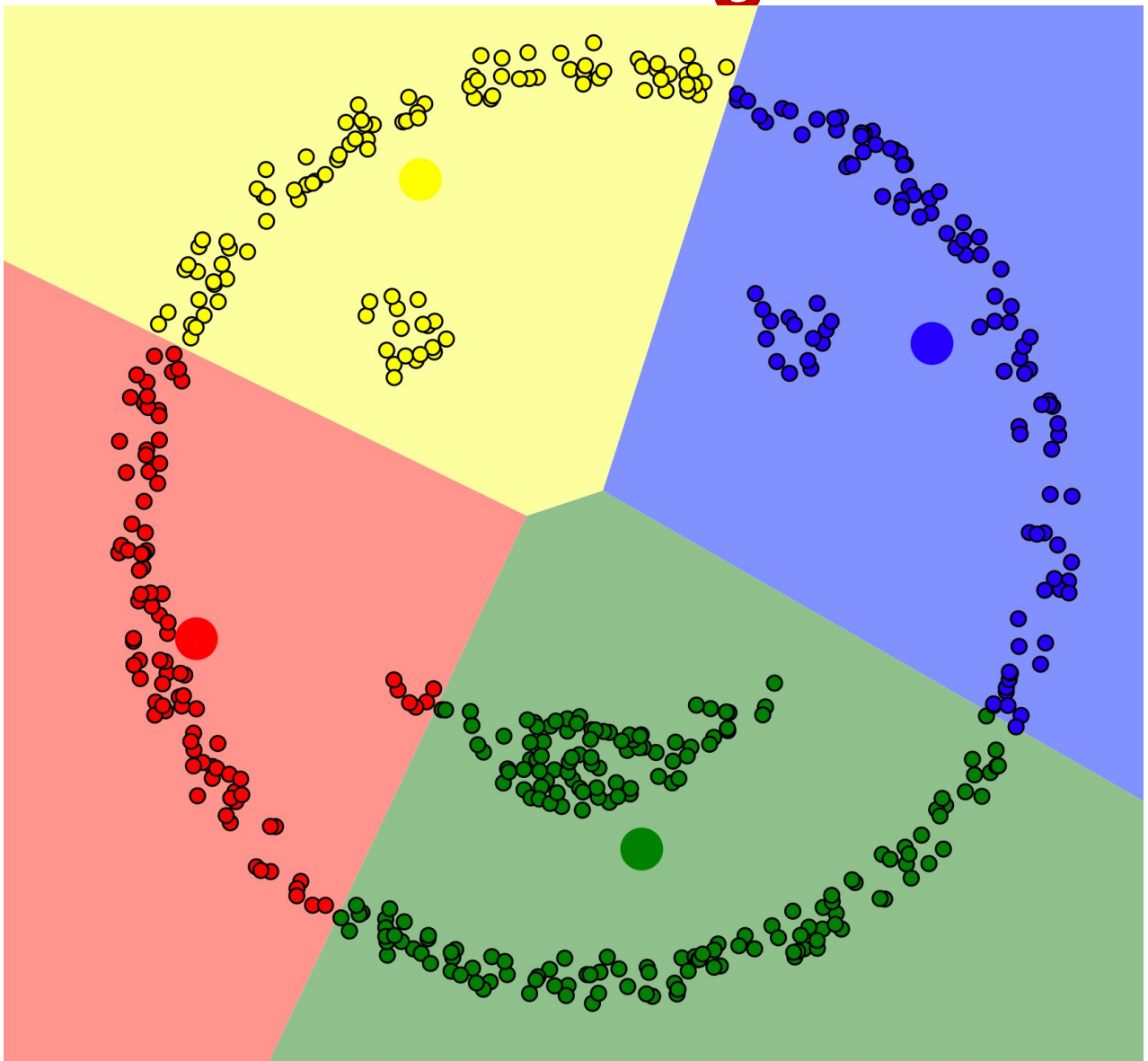
- It is not always obvious to choose the right K in high-dimensional data.
- “Knee finding”- abrupt change of objective at k=2



When K-Means fails?



Even setting the right K, K-Means could not find the right clusters!



Summary: K-Means Clustering

- Strength
  - Simple, easy to implement and debug
  - Intuitive objective function: optimizing intra-cluster similarity
  - Efficient: complexity ?
- Weakness
  - Applicable only when ***mean*** can be calculated
    - What about Categorical data?
  - Terminates at a local optimum.
  - Initialization is important
  - Sensitive to noisy data and outliers
  - Not able to find clusters with *non-convex* shape

Dimensionality Reduction

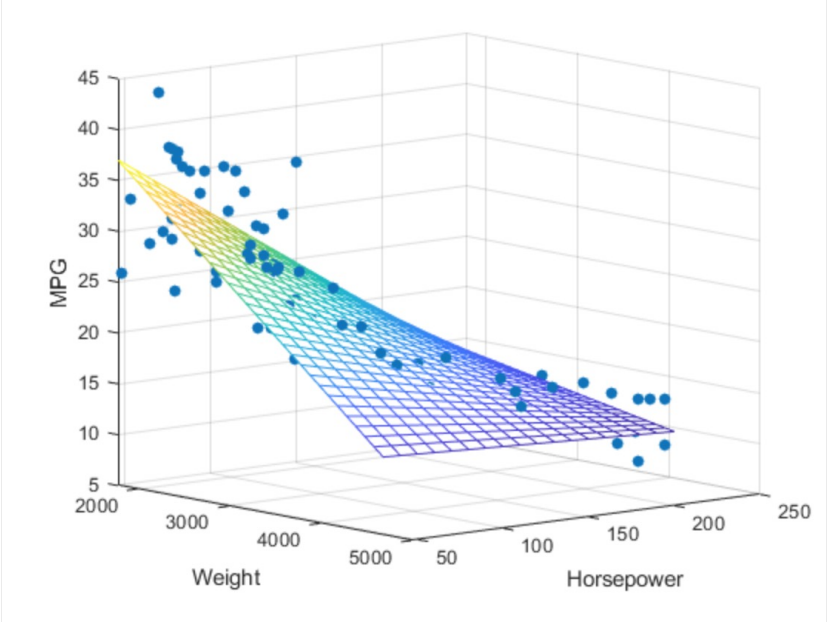
- High-dimensional data
- Document classification/clustering: How many features to represent a document?
  - Words (unigram), bigrams, n-gram
- Features for representing a user on Youtube?
  - Each visited video id can be features

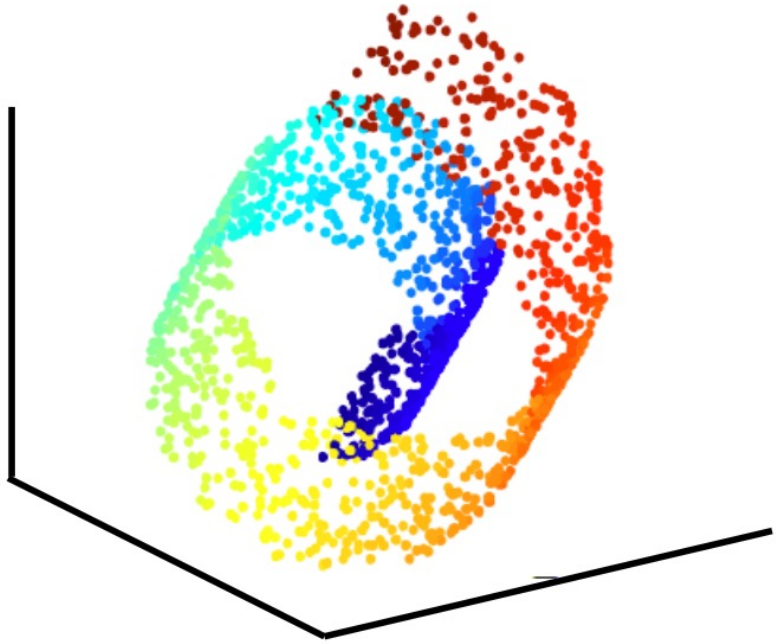
Curse of Dimensionality

- Why are more features bad?
  - Redundant/noisy/useless features
  - Need more storage space and computation
  - More model parameters (even if using decision trees or linear models)
  - Impossible to visualize (> 3 dims)
  - As dimensionality increases, the distances between data points are indifferent.

Dimensionality Reduction

- Reveal latent features
  - Assuming data points stay in a low-dimensional manifold

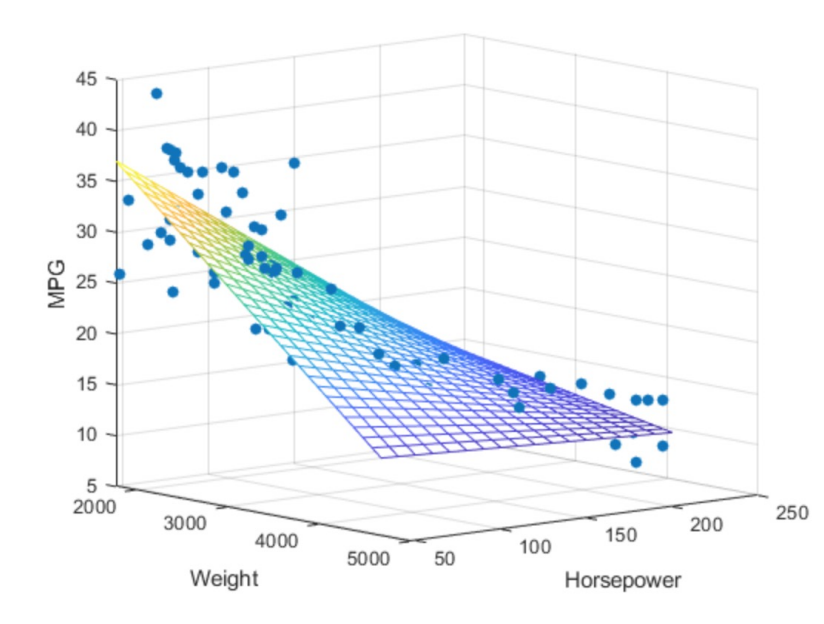




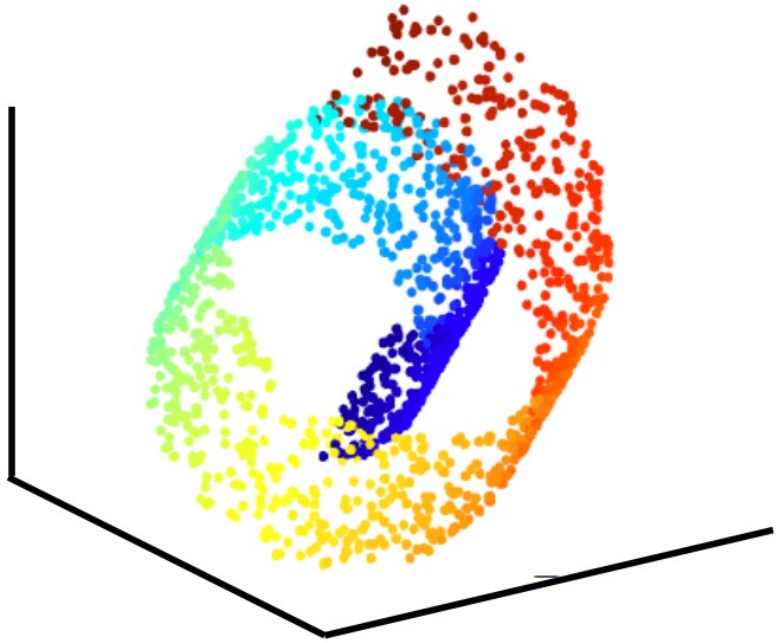
Latent Space

- A linear/nonlinear combination of features that capture essential factors of data
- Do not necessarily have physical meaning

Linear



Nonlinear

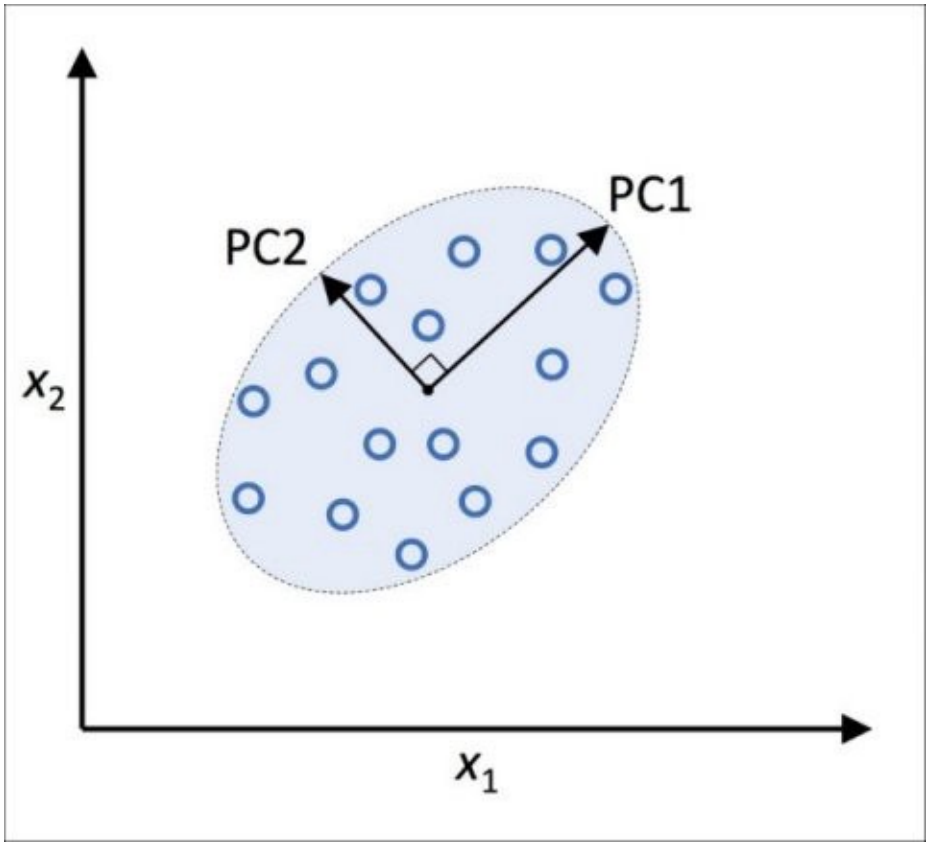


Dimensionality Reduction

- Linear:
  - Principal Component Analysis (PCA)
  - Independent Component Analysis (ICA)
  - Non-negative matrix factorization (NMF)
- Nonlinear:
  - Kernel PCA
  - Local Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (T-SNE)
  - UMAP

Principal Component Analysis

- Assuming data lie in a linear low dimensional subspace
- Axes of this subspace are known as principal components



Principal Component Analysis

Data X: $n \times d$ matrix

1. Compute the mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

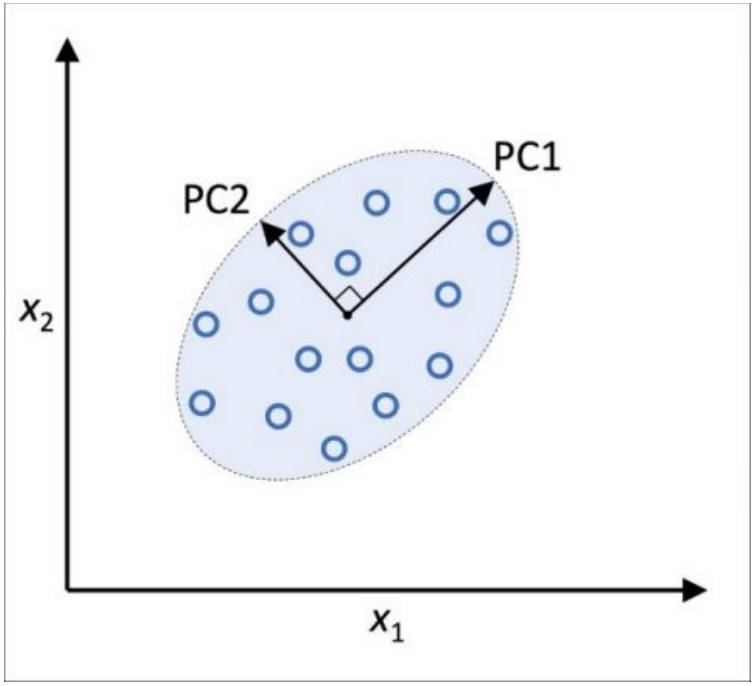
2. Subtract the mean:

$$\tilde{x}_i = x_i - \bar{x}$$

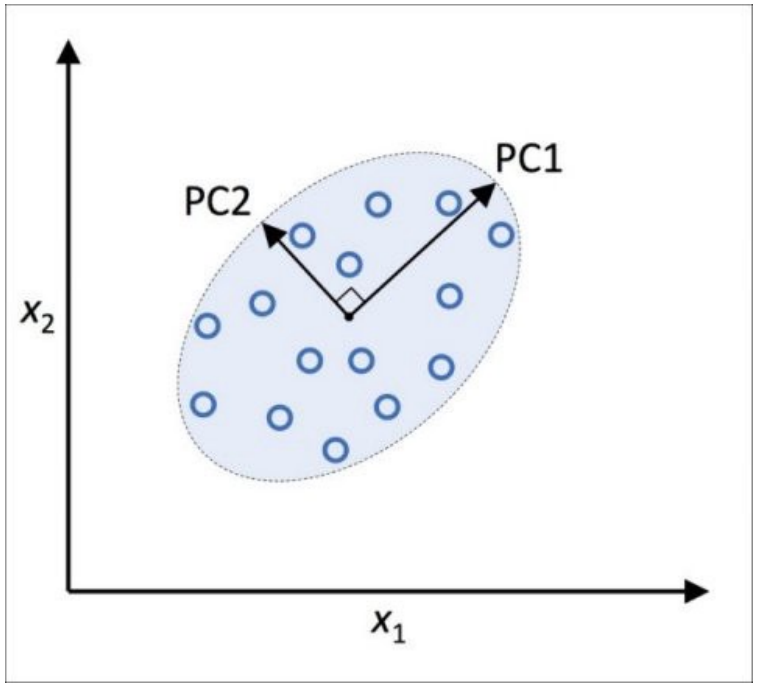
3. Compute the covariance

$$S = \frac{1}{n}\sum_{i=1}^{n} \tilde{x}_i \cdot \tilde{x}_i^T = \frac{1}{n}\tilde{X}^T\tilde{X}$$

4. 1st Principal Component – the max variance direction, which can be computed by eigenvalue decomposition



Principal Component Analysis



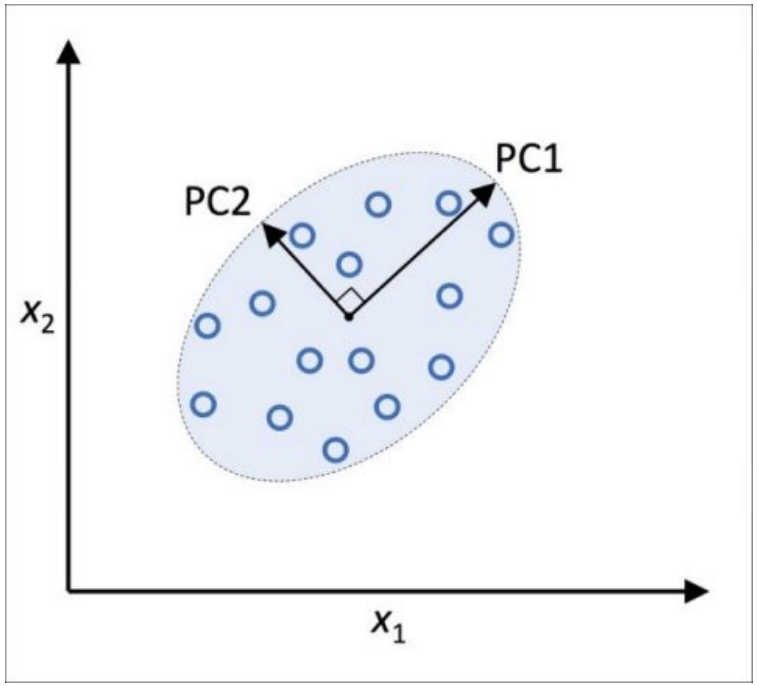
v is 1st Principal Component (unit length vector)

How is $\tilde{x}$ represented along 1st principal component direction?

Sample variance along 1st principal component direction?

1st Principal Component:

Principal Component Analysis



1st Principal Component:

$$\max_{\mathbf{v}} v^T \tilde{X}^T \tilde{X} v \quad \text{s.t.} \quad v^T v = 1$$

Lagrangian method:

$$\max_{v} v^T \tilde{X}^T \tilde{X} v - \lambda v^T v$$

$\frac{\partial}{\partial v} = 0$ ➔

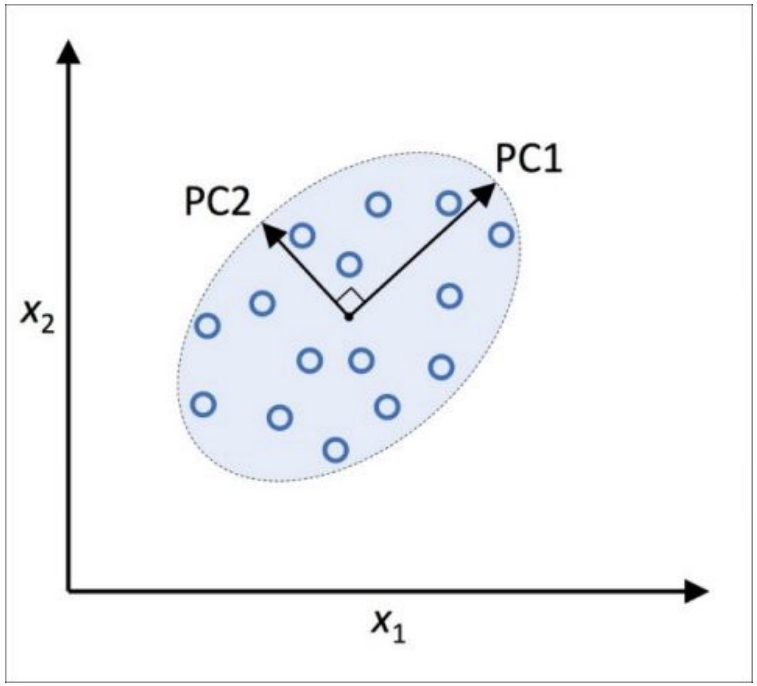
$$\tilde{X}^T \tilde{X} v - \lambda v = 0$$

➔

$$(\tilde{X}^T \tilde{X}) v = \lambda v$$

➔ v is the top-1 eigenvector of covariance matrix

Principal Component Analysis



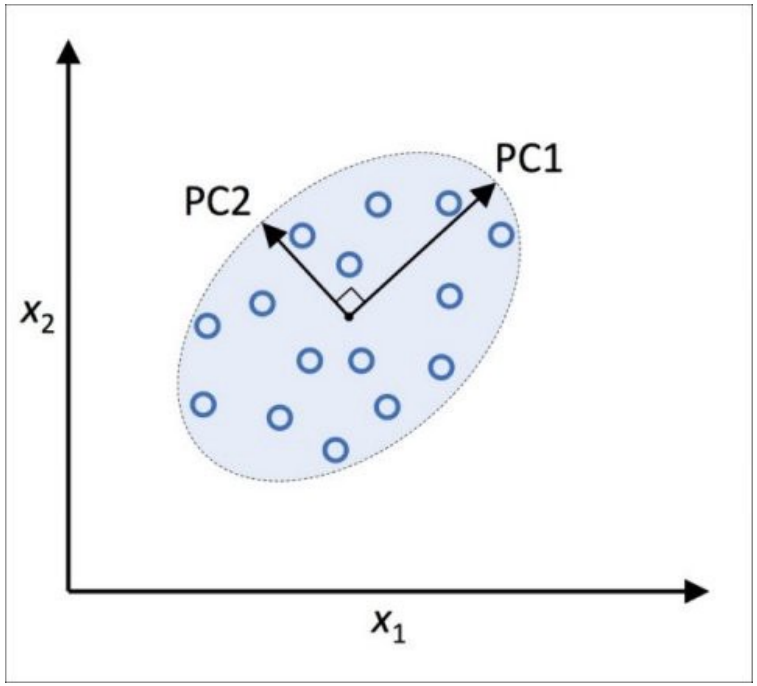
1st Principal Component v_1 is the top-1 eigenvector of covariance matrix, with the largest eigenvalue λ_1.

How about the 2nd Principal Component, 3rd Principal Component, …?

v_2 should be a unit vector orthogonal to v_1, maximal variance direction after removing PC1

i.e. 2nd Principal Component is the eigenvector of covariance matrix associated with 2nd largest eigenvalue

Alternative Interpretation



Minimum reconstruction error: PCA finds vectors v such that projection on the vectors yields minimum MSE

$$\min_{v} \frac{1}{n} \sum_{i=1}^{n} \|\tilde{x}_i - (v^T \tilde{x}_i) v\|^2$$

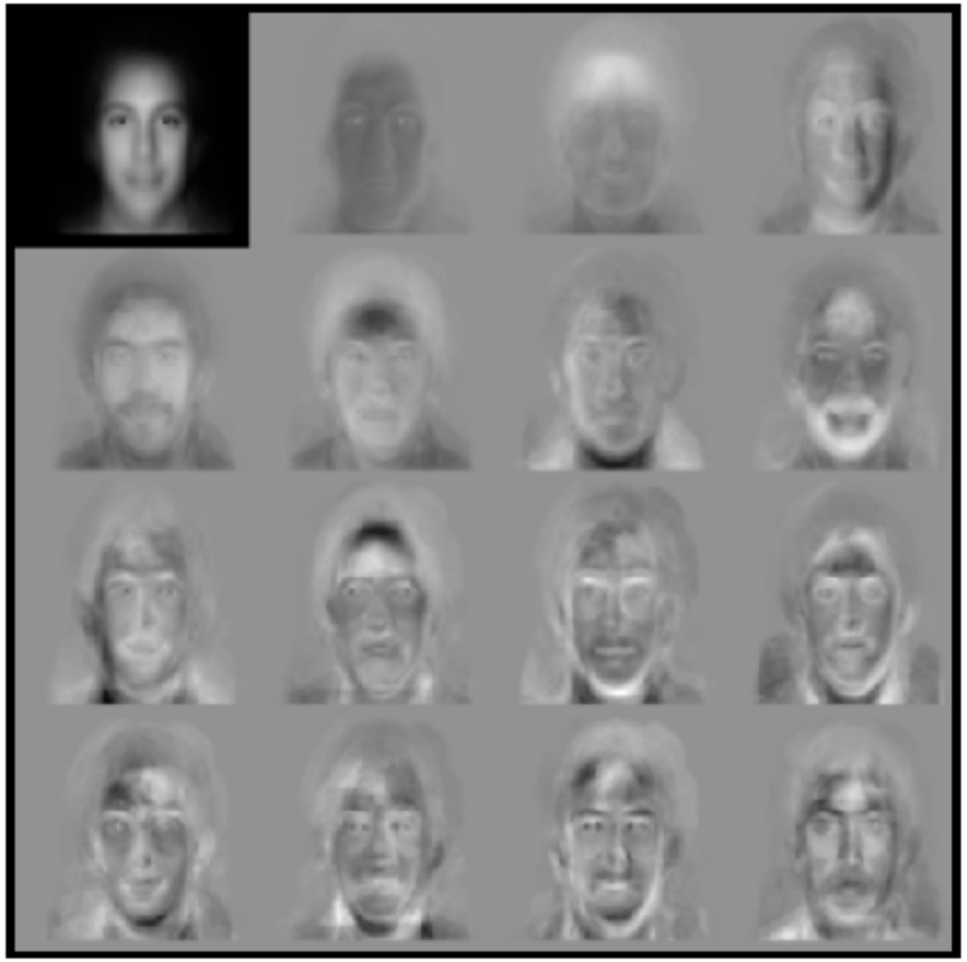
Dimensionality Reduction using PCA

- Eigenvalue λ indicates the amount of variability along the principal direction
- Small eigenvalues mean tiny variability, therefore those directions can be removed
- Projecting data to the top-k principal components $V = (v_1, v_2, \ldots, v_k)^T$

$$\widehat{x_i} = V\widetilde{x_i} = V(x_i - \frac{1}{n}\sum_{j=1} x_j)$$

Or in matrix form: $\hat{X} = \tilde{X}V^T$

Example: Eigenface



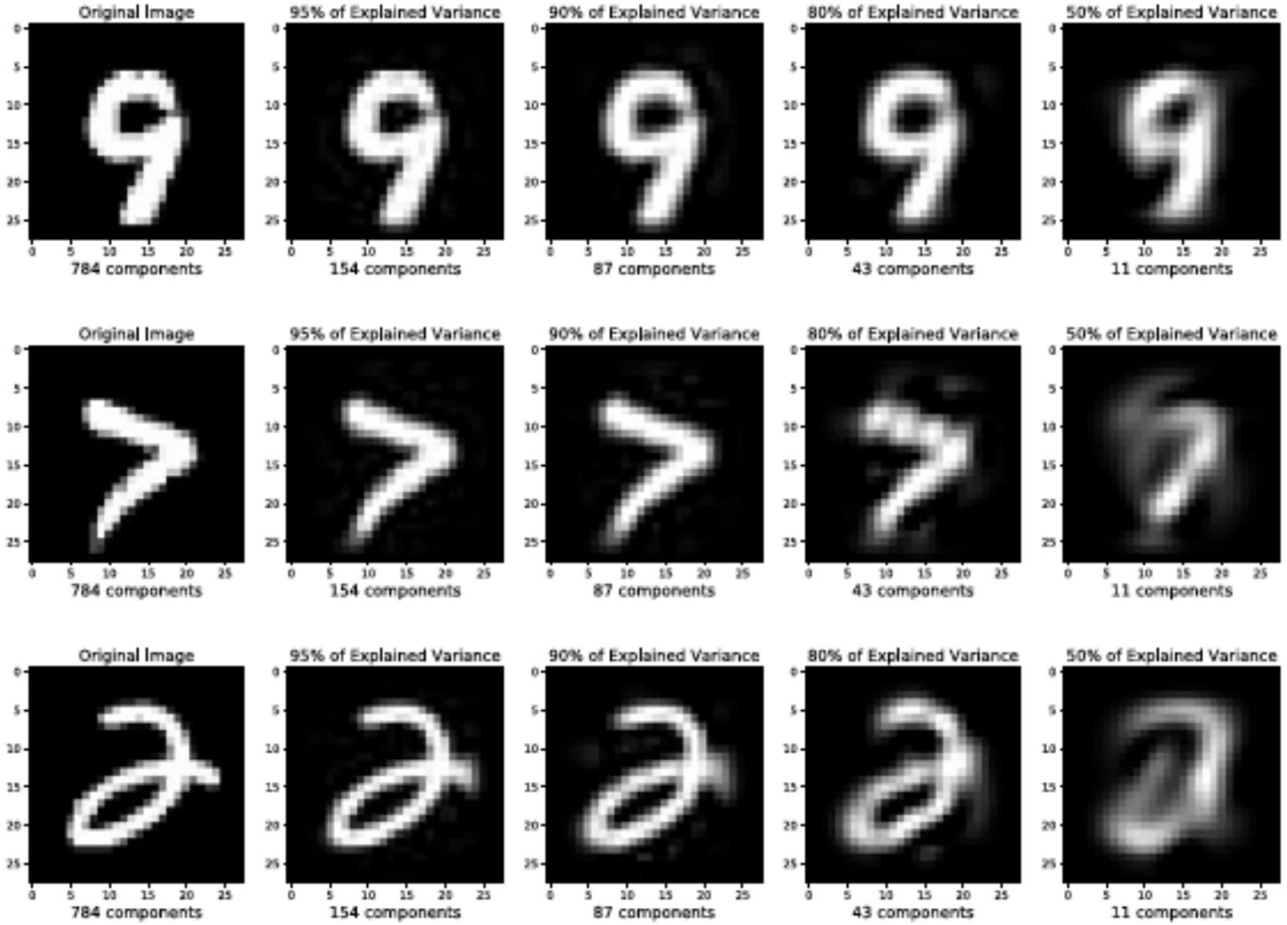
Data: 7562 images of human faces

Top left image is linear combination of the rest.

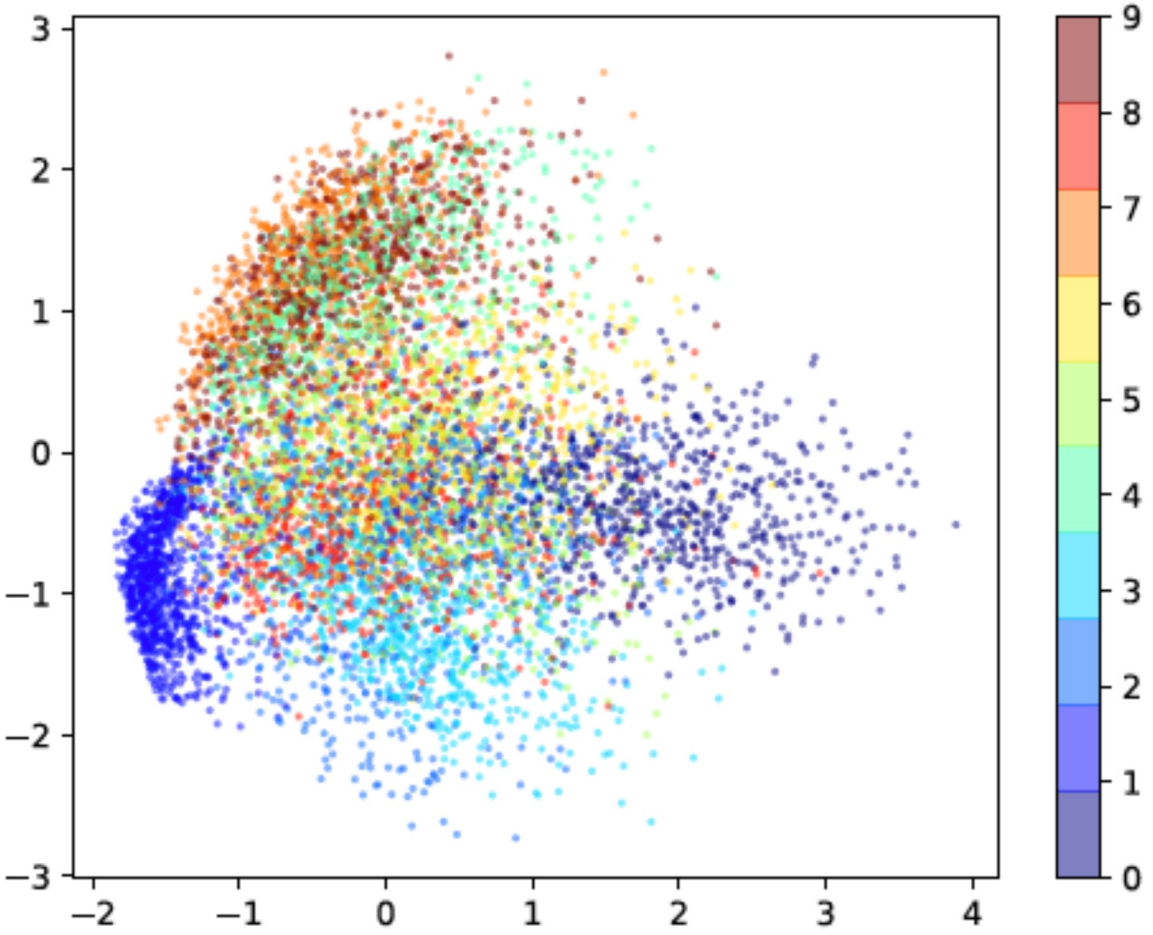
Independently developed by
Sirovich and Kirby, 1987
Turk and Pentland, 1991

Example: handwritten digits

- MNIST dataset: 28 x 28 images of digits
- Project to k dimensional principal components and then reconstruct to original space



Visualizing MNIST in 2D



Interactive Illustration

https://setosa.io/ev/principal-component-analysis/

Properties of PCA

- Advantage:
  - Eigen decomposition, $O(n^3)$
  - No need to tune parameters
- Weaknesses
  - Linear projection
  - Second-order statistics (covariance)

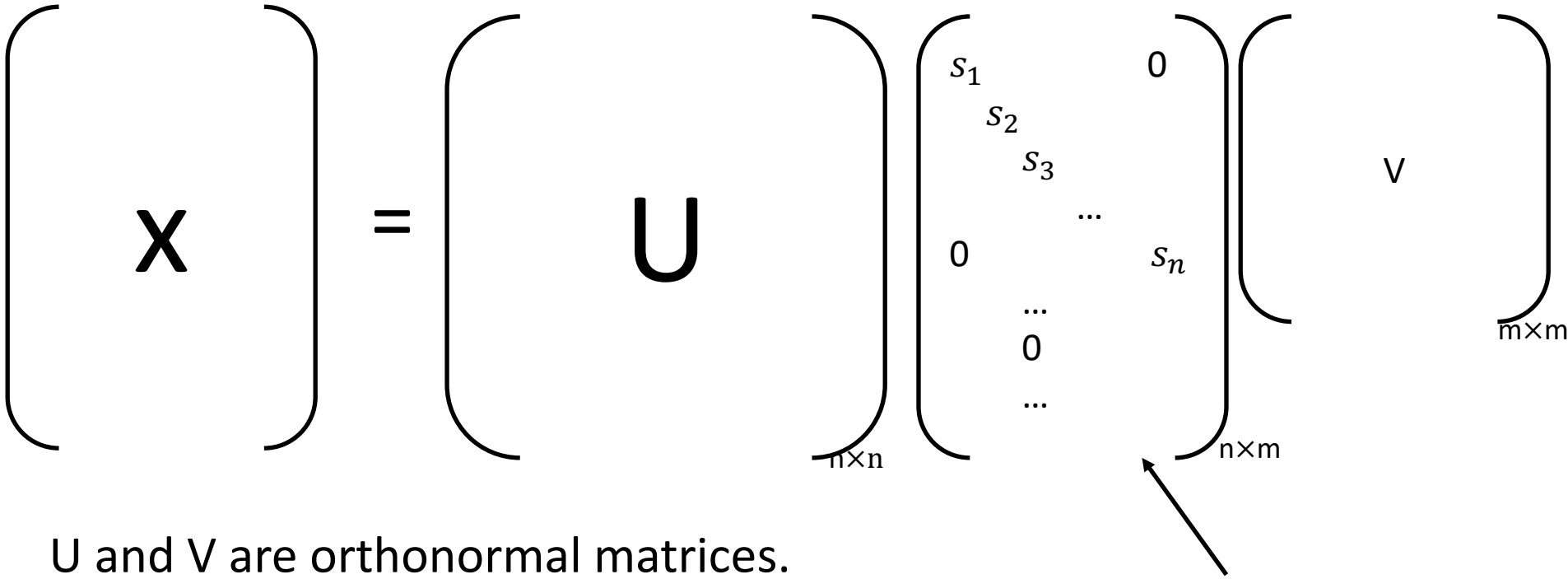
Computing PCA with Python

```
def pca(X):
  # Data matrix X, assumes 0-centered
  n, m = X.shape

  # Compute covariance matrix
  S = numpy.dot(X.T, X) / n
  # Eigen decomposition
  eigen_vals, eigen_vecs = numpy.linalg.eig(S)
  # Project X onto PC space
  X_pca = numpy.dot(X, eigen_vecs)
  return X_pca
```

Singular Value Decomposition

$$X = USV^T$$



U and V are orthonormal matrices.

$$U^T U = I$$
$$V^T V = I$$

Diagonal matrix with singular values (ordered)

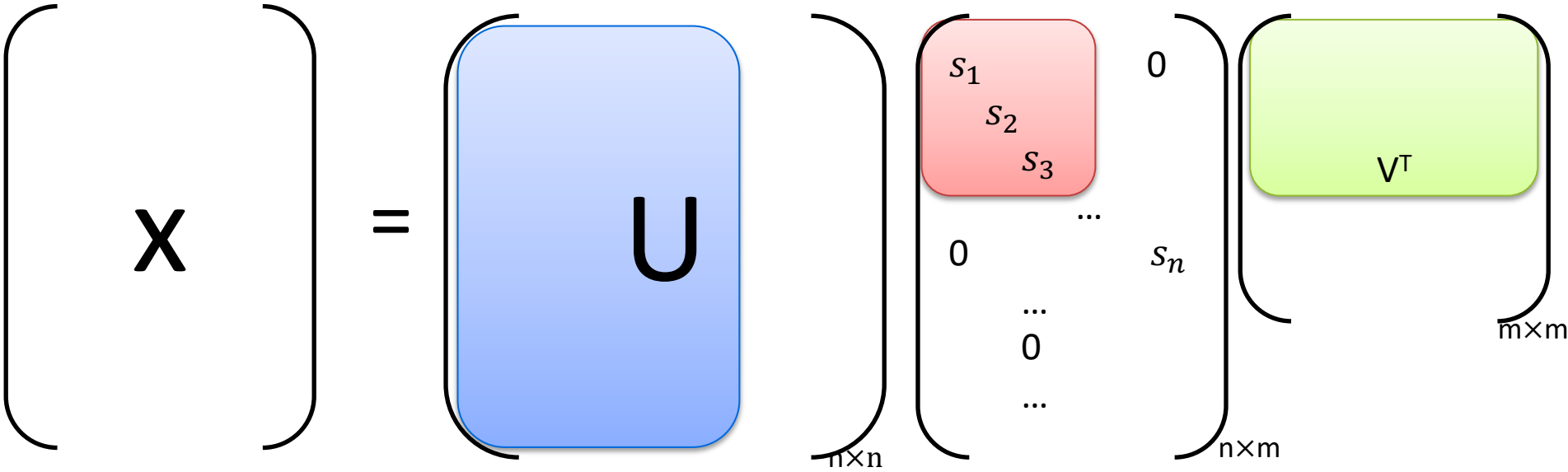
Relation between SVD and PCA

- Right singular vectors = Principal components
- Why?
- So we can obtain dimensionality reduction by truncating singular values.

$$\hat{X} = U_{n\times k} S_{k\times k} V^T_{k\times m}$$

Dimensionality Reduction by SVD

$$\hat{X} = U_{n\times k} S_{k\times k} V^T_{k\times m}$$



SVD and PCA

- PCA can be computed using SVD
- Computing SVD with top-k singular values is faster than PCA with eigen decomposition.
- SVD can be used to compute the pseudo-inverse of a rectangular matrix

Summary: Dimensionality Reduction

- Principal Component Analysis
  1. Center the data by subtracting the mean
  2. Compute covariance matrix $S = \frac{1}{n}\tilde{X}^T\tilde{X}$
  3. Eigen decomposition on covariance matrix
  4. Eigenvectors are principal components, eigenvalues are energy
- Singular Value Decomposition
  - Can be used to compute PCA, and faster
  - Reduce the dimension by truncating at top-k singular values. $\hat{X} = U_{n\times k}S_{k\times k}V^T_{k\times m}$