Recall from last time ...

Unsupervised Learning

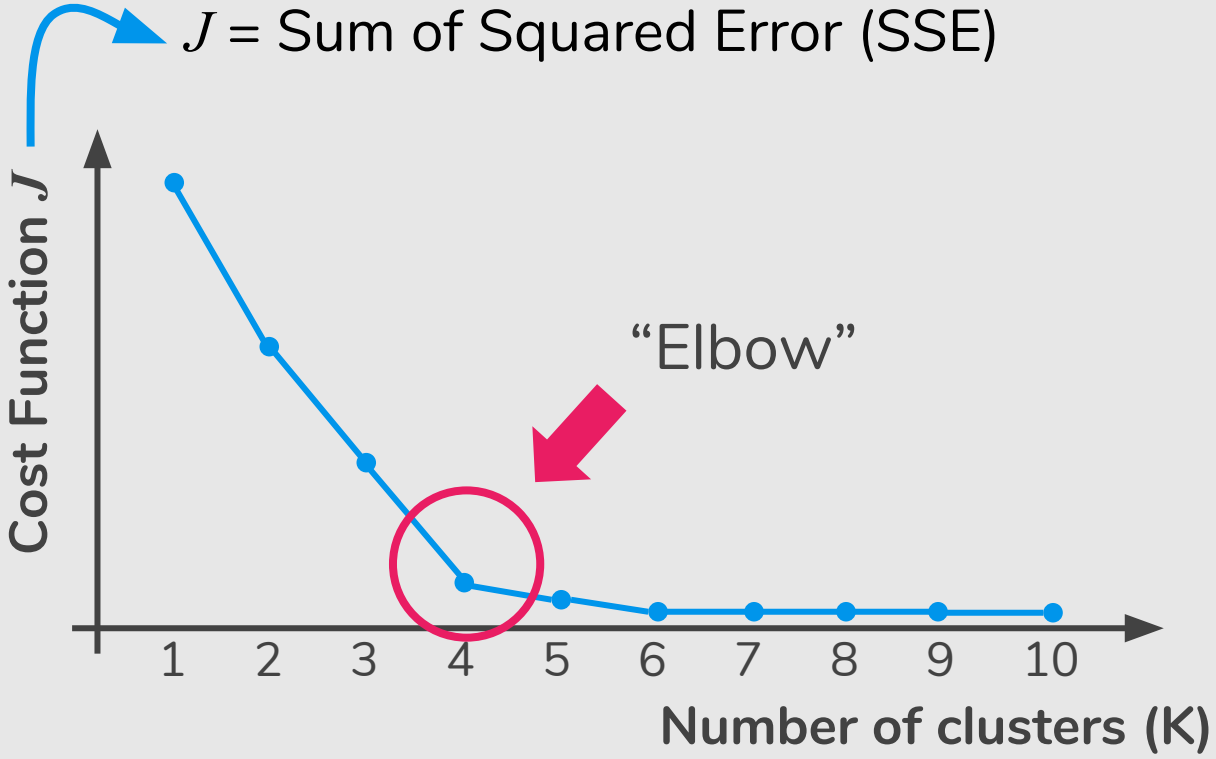
The goal of unsupervised learning is **to find patterns** in the data, and build new and useful representations of it.

Clustering
k-Means Algorithm

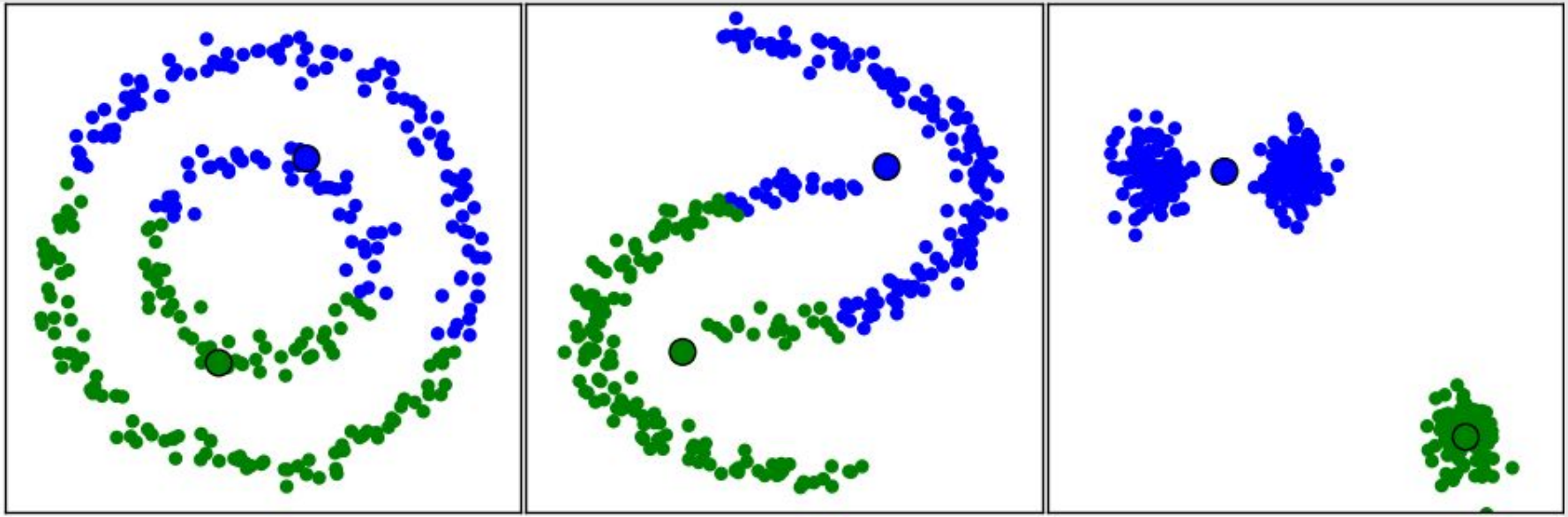
k-Means Algorithm

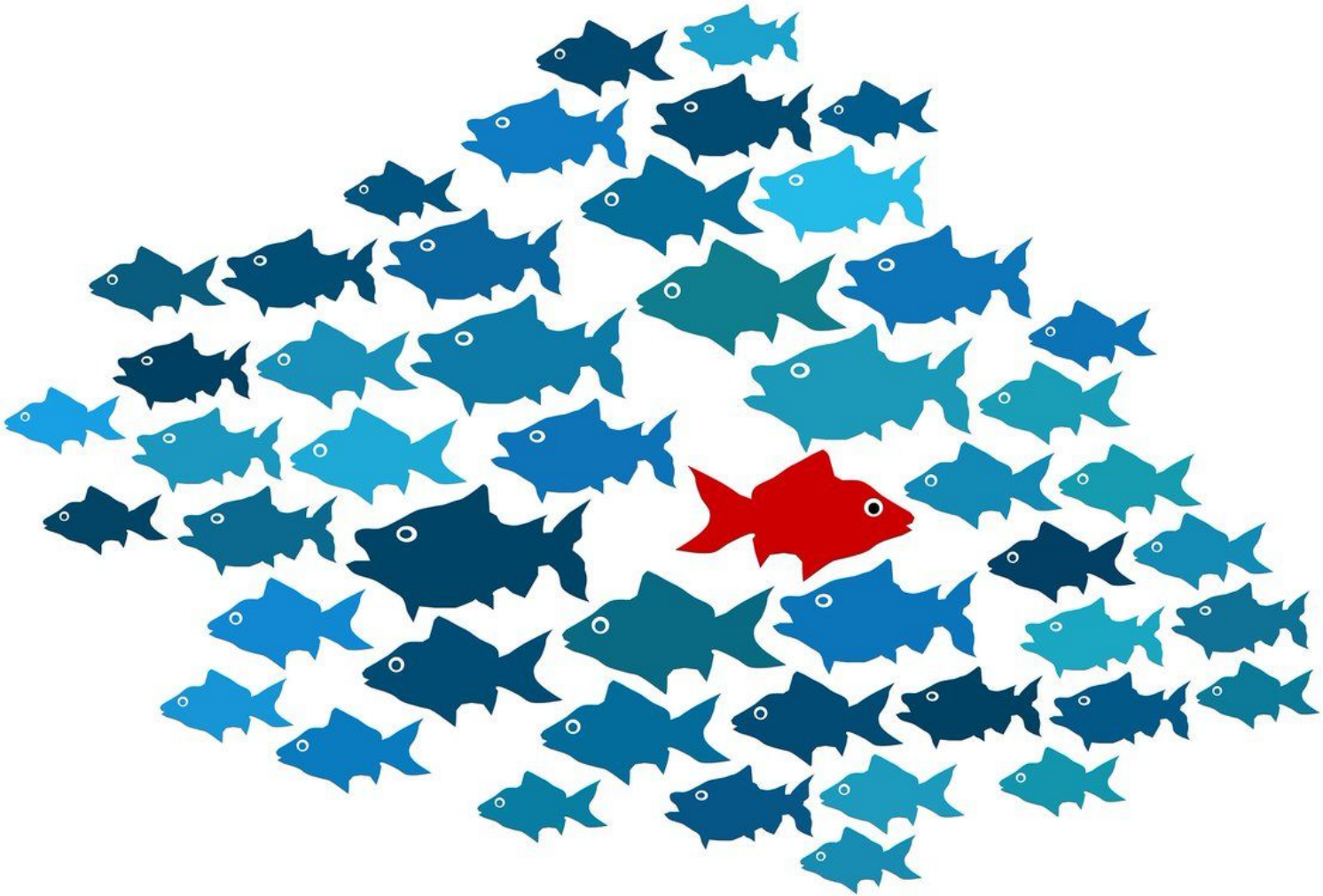
1. Define the *k* centroids.
2. Find the closest centroid & update cluster assignments.
3. Move the centroids to the center of their clusters.
4. Repeat steps 2 and 3 until the centroid stop moving a lot at each iteration (i.e., until the algorithm converges).

Elbow Method



k-Means: Additional Issues



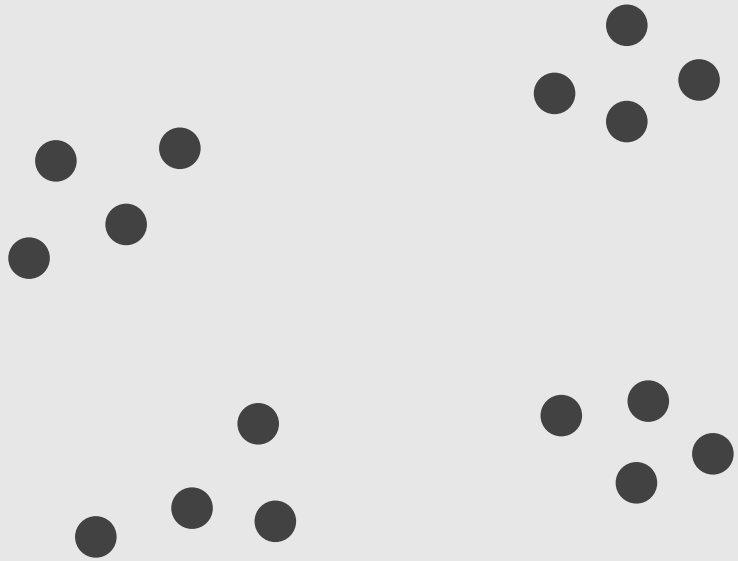


Reducing the SSE with Postprocessing

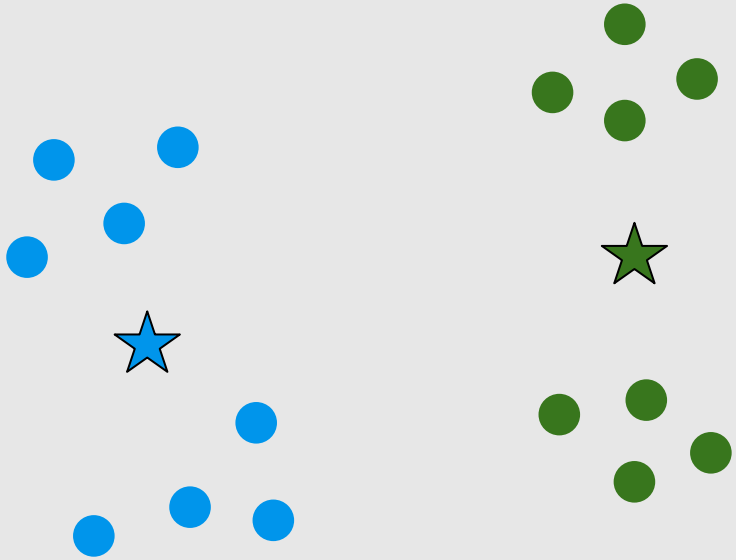
- **Split a cluster**: the cluster with the largest SSE is usually chosen.
- **Introduce a new cluster centroid**: often the point that is farthest from any cluster center is chosen.
- **Merge two clusters**: The clusters with the closest centroids are typically chosen.

k-Means Variations

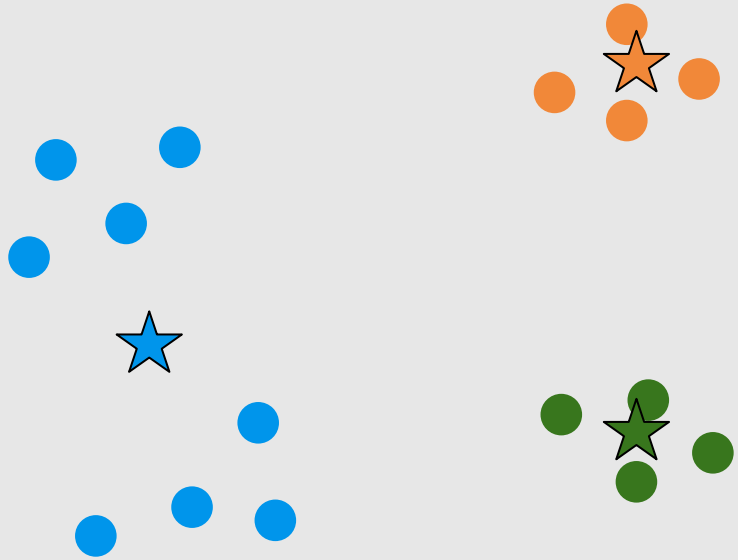
Bisecting k-Means



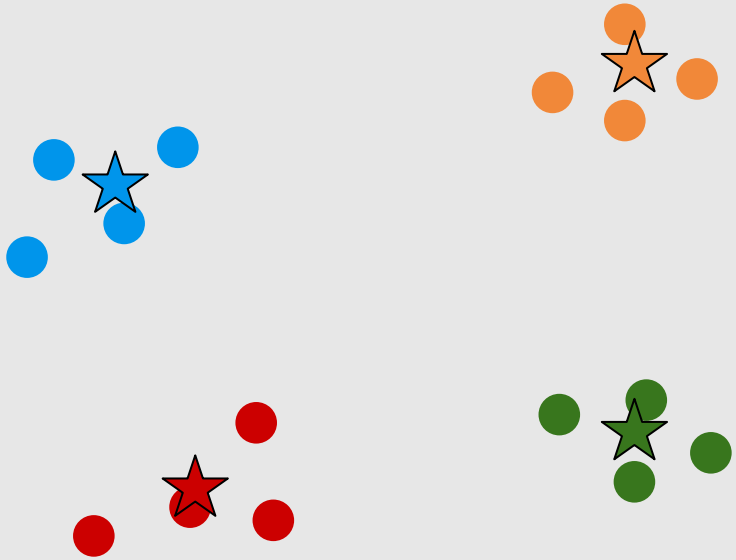
Bisecting k-Means



Bisecting k-Means



Bisecting k-Means

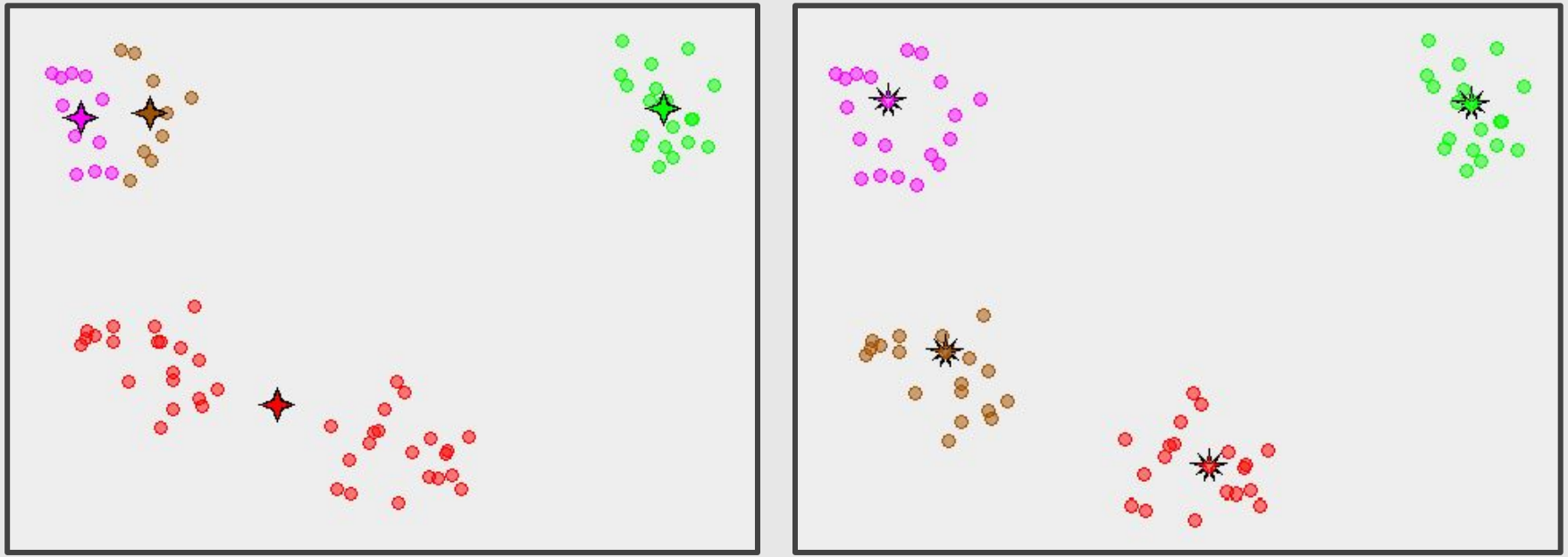


Mini-batch k-Means

- Uses mini-batches to reduce the computation time, while still attempting to optimize the same objective function.

- Converges faster than k-Means, but the quality of the results is reduced.

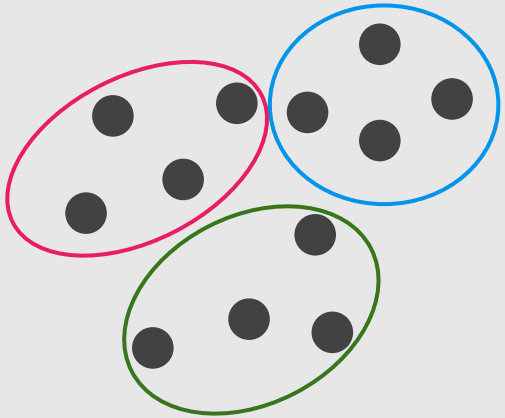
k-Means (left) vs k-Medoids (right)



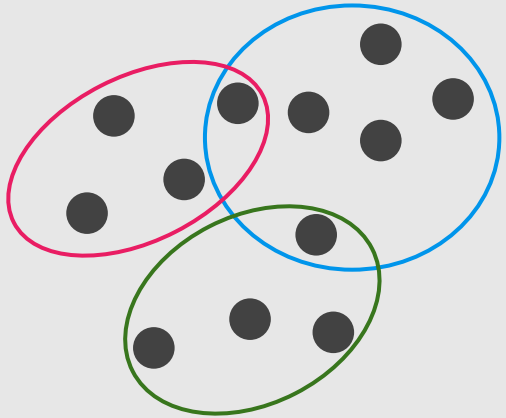
Credit: https://commons.wikimedia.org/wiki/File:K-means_versus_k-medoids.png

Fuzzy Clustering (Soft Clustering)

- Each data point can belong to more than one cluster.



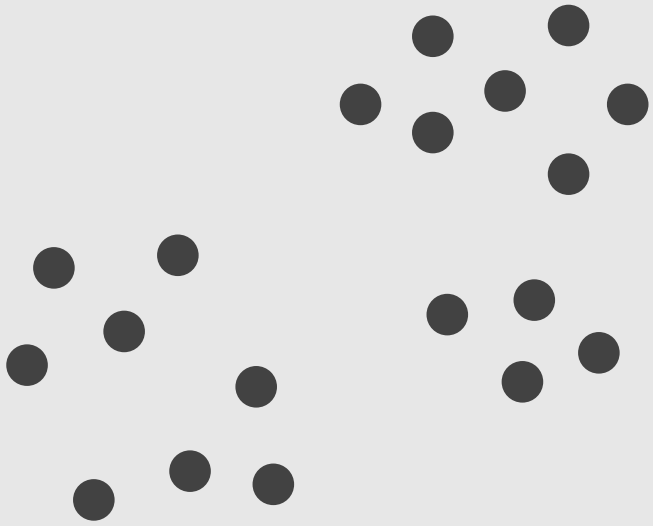
Hard clustering



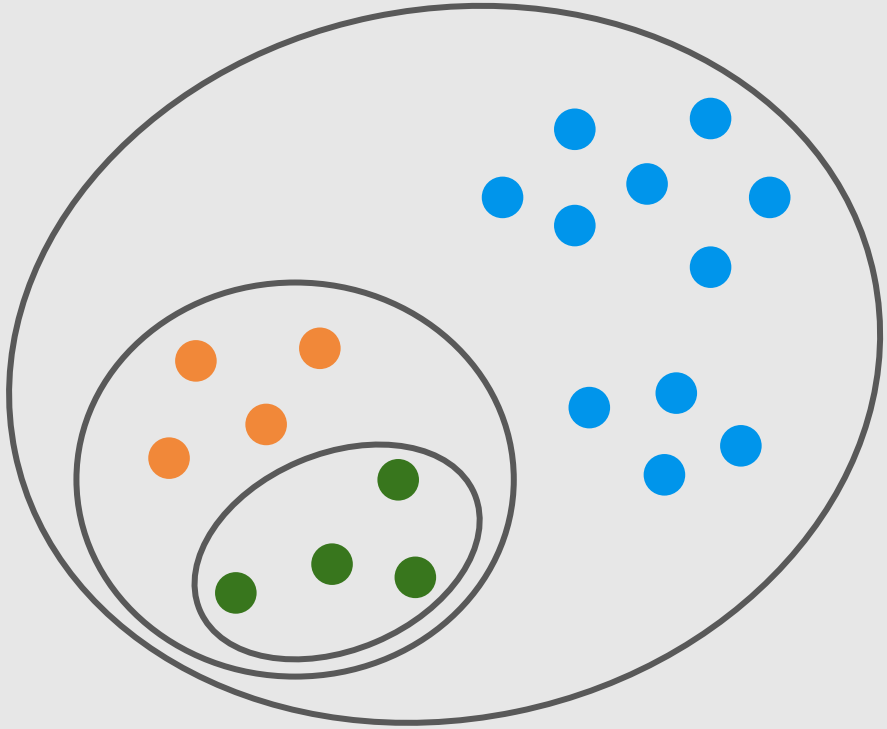
Soft clustering

Hierarchical Clustering

Hierarchical versus Partitional



Original data



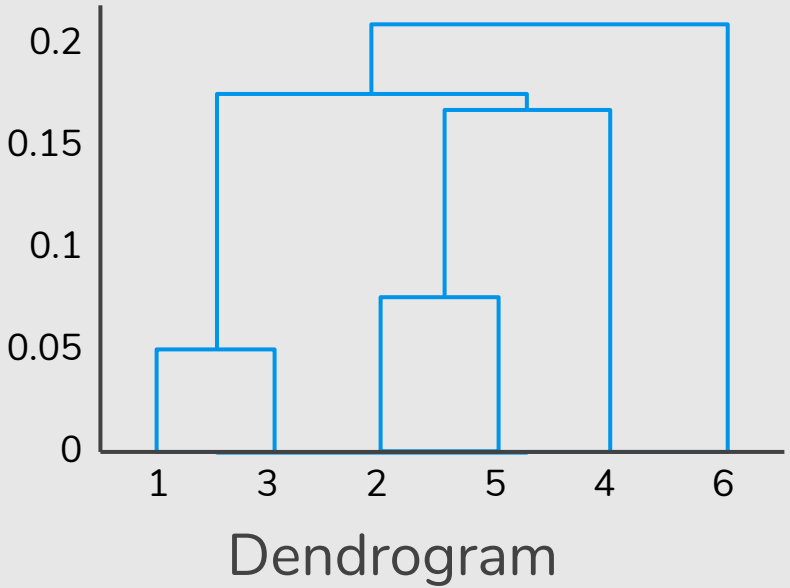
Hierarchical clustering

Hierarchical Clustering

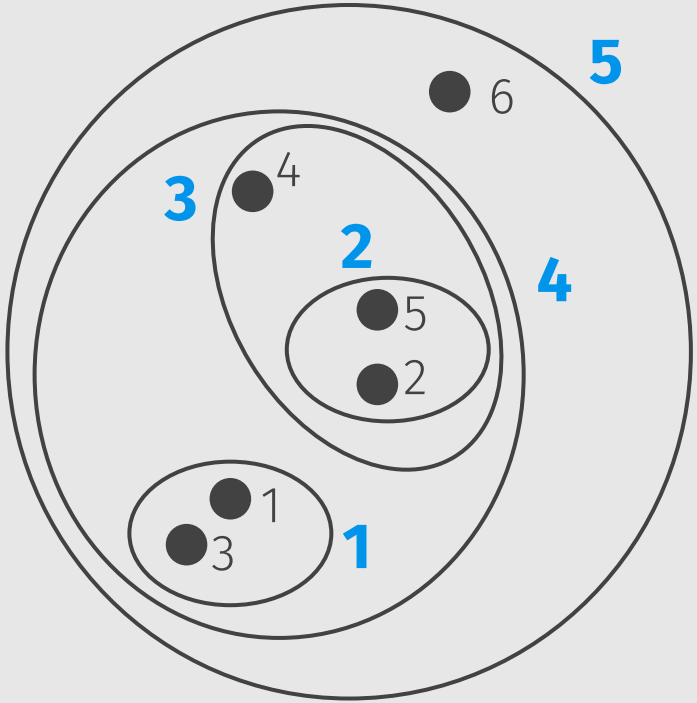
- **Agglomerative** (“bottom up”): each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

- **Divisive** (“top down”): all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Agglomerative Hierarchical Clustering



Dendrogram



Agglomerative Hierarchical Clustering

1: compute the **proximity matrix**, if necessary.

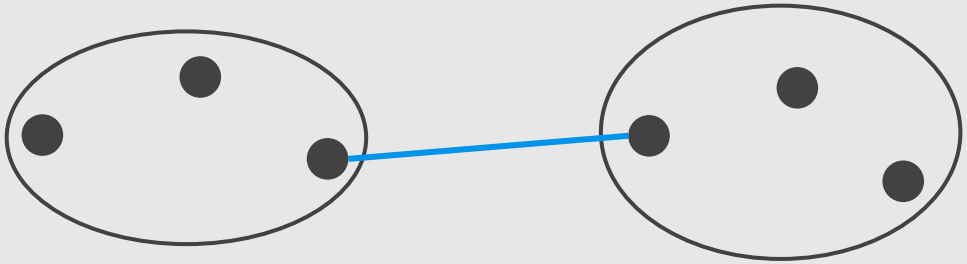
2: **repeat**

3: merge the closest two clusters.

4: update the proximity matrix to reflect the proximity between the new cluster and the original clusters.

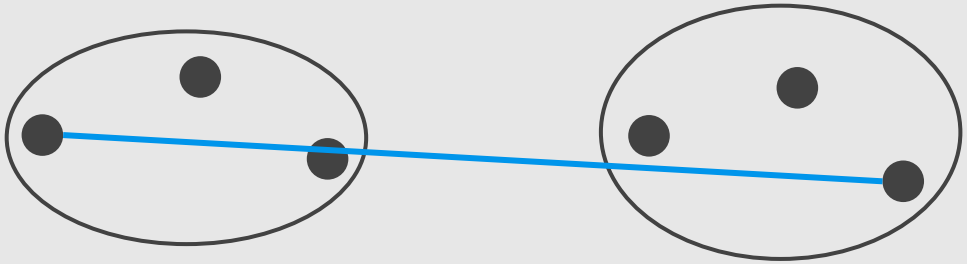
5: **until** only one cluster remains.

Defining Proximity between Clusters



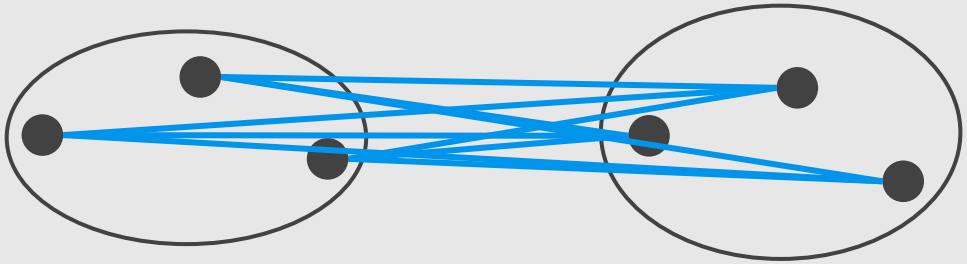
Single link or **MIN**: defines cluster proximity as the **proximity** between the closest two points that are in different clusters.

Defining Proximity between Clusters



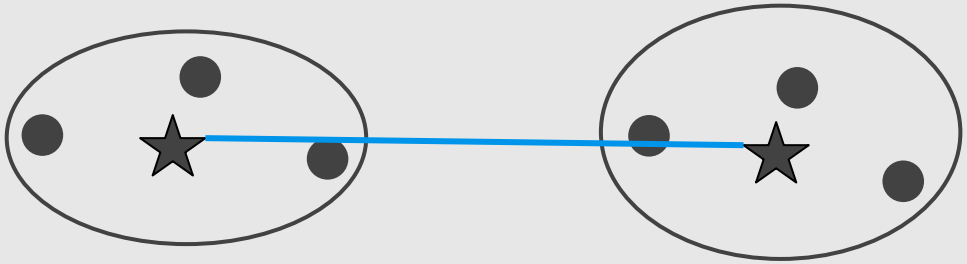
Complete link or **MAX**: takes the proximity between the **farthest** two points in different clusters to be the cluster proximity.

Defining Proximity between Clusters



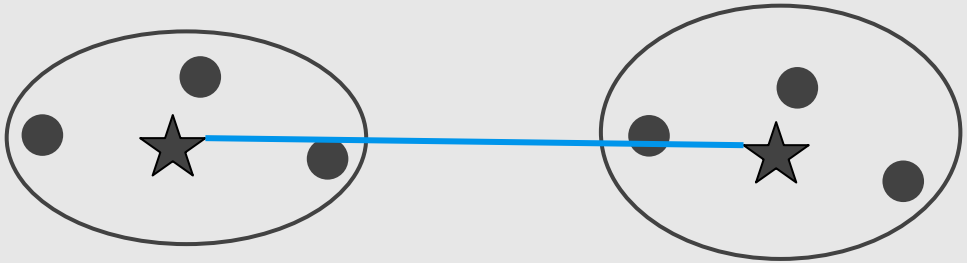
Average: defines cluster proximity to be the **average pairwise** proximities of all pairs of points from different clusters.

Defining Proximity between Clusters



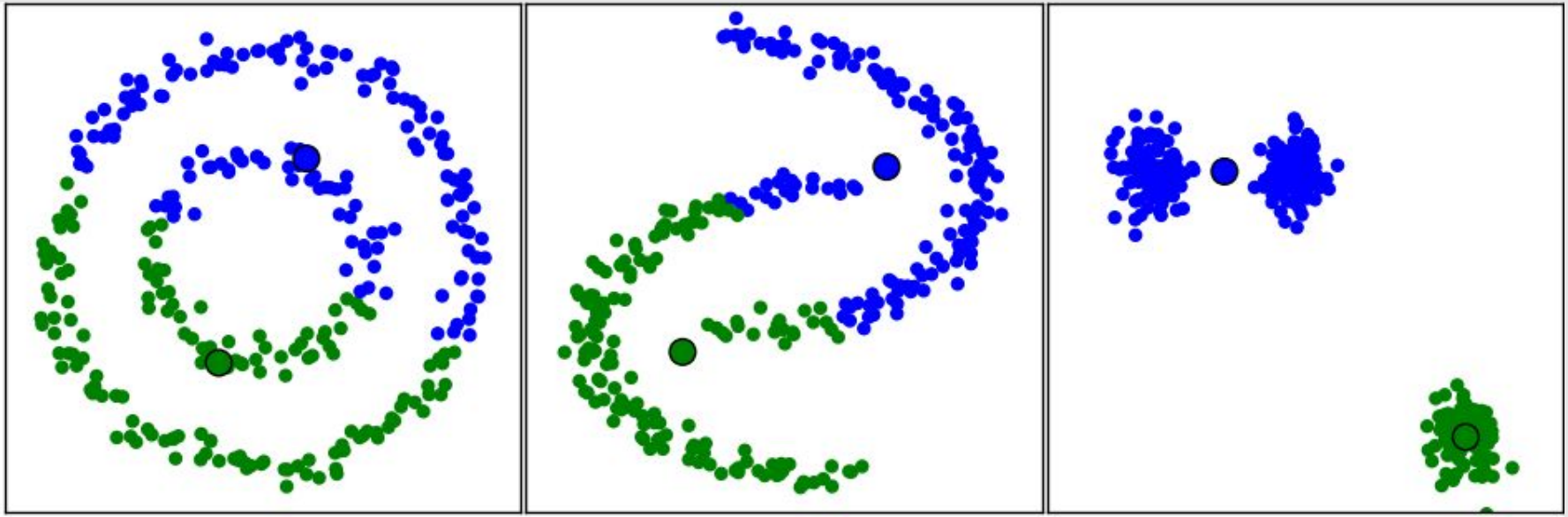
Centroids: the cluster proximity is commonly defined as the proximity between cluster centroids.

Defining Proximity between Clusters



Ward's: measures the proximity between two clusters in terms of the increase in the SSE that results from merging the two cluster.

DBSCAN

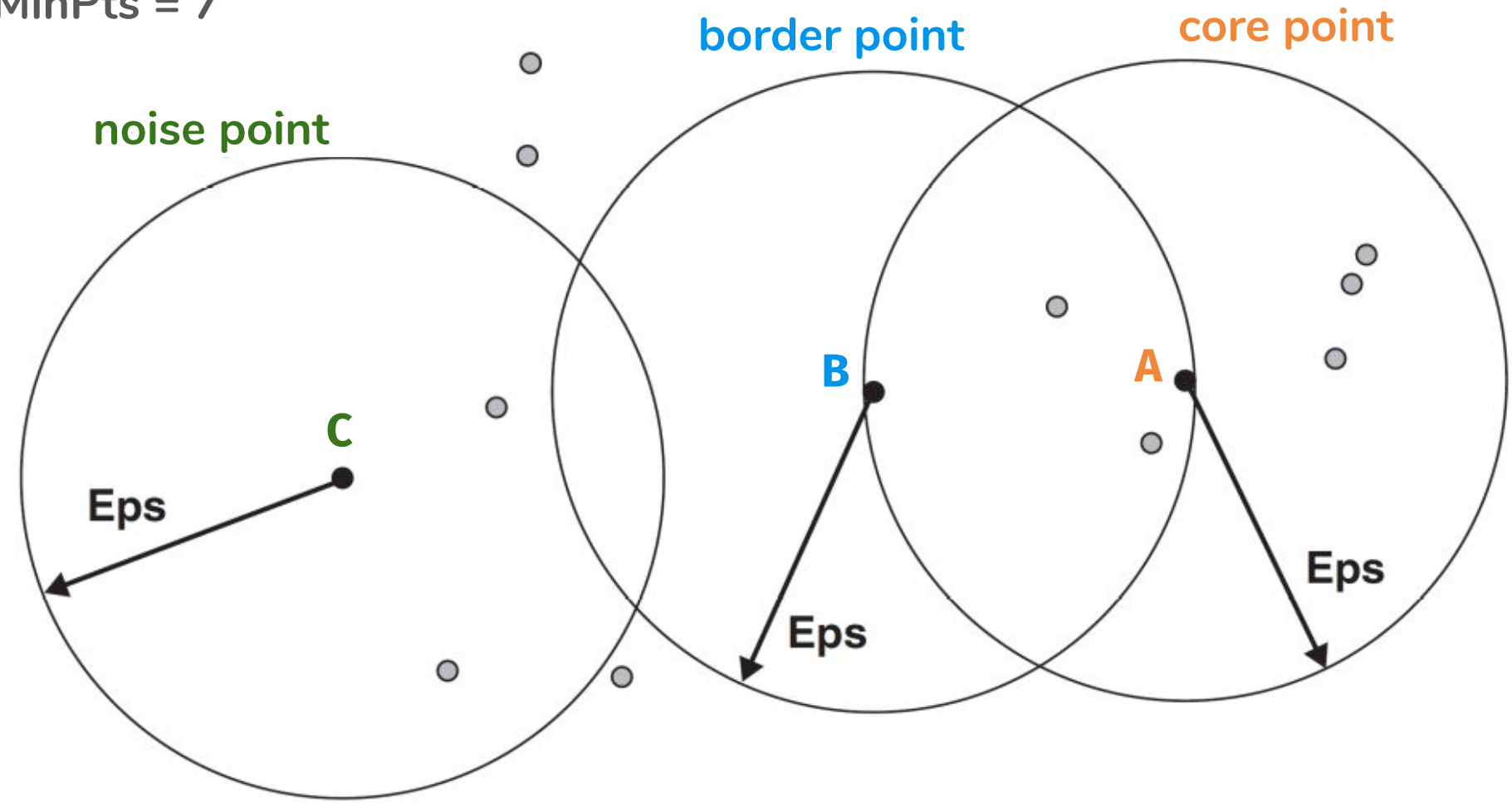


DBSCAN Clustering

- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

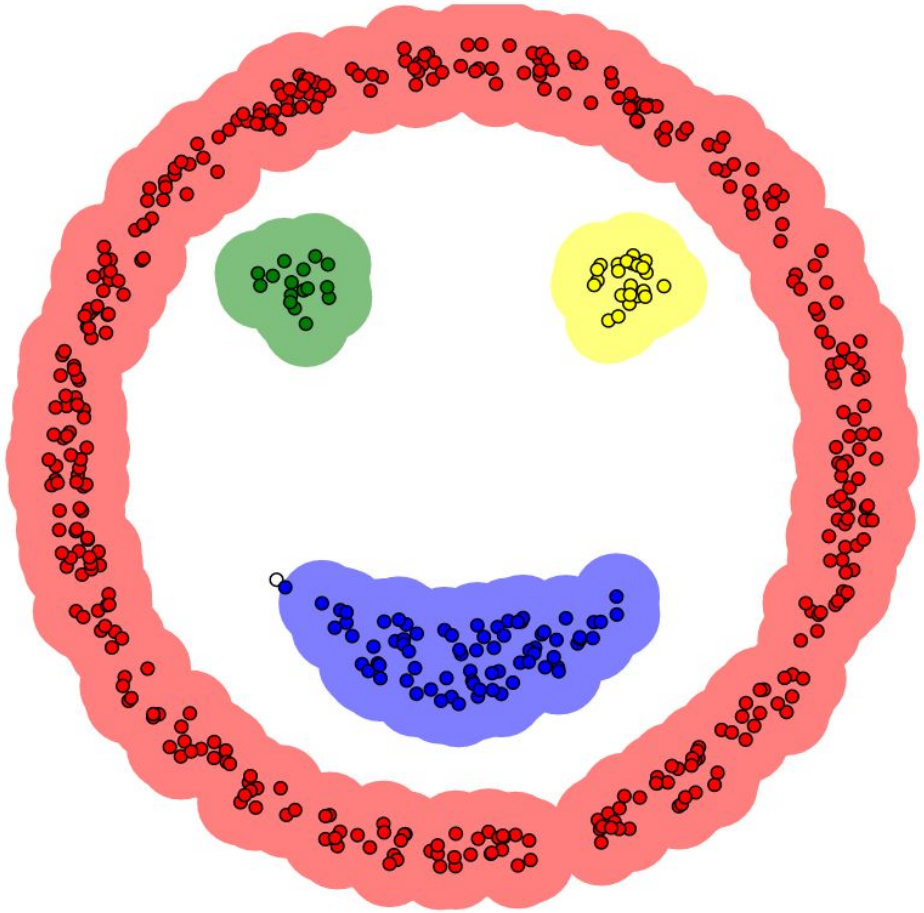
- Given a set of points in some space, **it groups together points that are closely packed together** (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions.

MinPts = 7



DBSCAN Clustering

- **Core points**: A point is a core point if there are at least *MinPts* within a distance of *Eps*, where *MinPts* and *Eps* are user-specified parameters.
- **Border points**: A border point is not a core point, but falls within the neighborhood of a core point.
- **Noise points**: A noise point is any point that is neither a core point nor a border point.



epsilon = 1.00
minPoints = 4

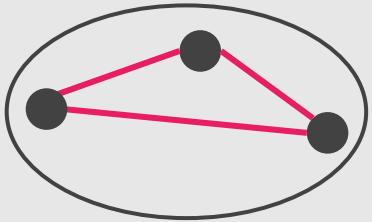
Restart

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering

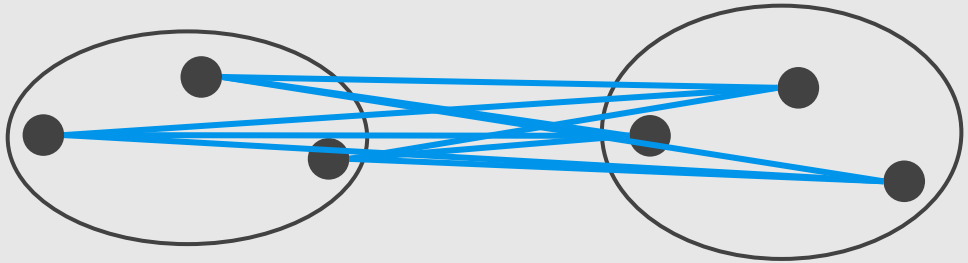
Clustering Performance Evaluation

Silhouette Coefficient

- The silhouette value is a measure of how similar a sample is to its own cluster (**cohesion**) compared to other clusters (**separation**).



Cohesion



Separation

Silhouette Coefficient

- The silhouette value is a measure of how similar a sample is to its own cluster (**cohesion**) compared to other clusters (**separation**).

- The silhouette ranges from −1 to +1.
  - High value = the clustering configuration is appropriate.
  - Low value = the clustering configuration may have too many or too few clusters.

Silhouette Coefficient

- The Silhouette Coefficient is defined **for each sample** and is composed of two scores:
  - a: The mean distance between a sample and all other points **in the same cluster**.
  - b: The mean distance between a sample and all other points **in the next nearest cluster**.

Silhouette Coefficient

- The Silhouette Coefficient s for **a single sample** is given as:

$$s = \frac{b - a}{max(a,b)}$$

- The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering ($a \ll b$). Scores around zero indicate overlapping clusters.

http://scikit-learn.org/stable/modules/clustering.html#clustering

Previous 2.2. Manifold... | Next 2.4. Biclustering | Up 2. Unsupervis...

scikit-learn v0.19.0
Other versions

Please cite us if you use the software.

2.3. Clustering
2.3.1. Overview of clustering methods
2.3.2. K-means
- 2.3.2.1. Mini Batch K-Means

2.3.3. Affinity Propagation
2.3.4. Mean Shift
2.3.5. Spectral clustering
- 2.3.5.1. Different label assignment strategies

2.3.6. Hierarchical clustering
- 2.3.6.1. Different linkage type: Ward, complete and average linkage
- 2.3.6.2. Adding connectivity constraints
- 2.3.6.3. Varying the metric

2.3.7. DBSCAN
2.3.8. Birch
2.3.9. Clustering performance

2.3. Clustering

Clustering of unlabeled data can be performed with the module `sklearn.cluster`.

Each clustering algorithm comes in two variants: a class, that implements the `fit` method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the `labels_` attribute.

Input data

One important thing to note is that the algorithms implemented in this module can take different kinds of matrix as input. All the methods accept standard data matrices of shape `[n_samples, n_features]`. These can be obtained from the classes in the `sklearn.feature_extraction` module. For `AffinityPropagation`, `SpectralClustering` and `DBSCAN` one can also input similarity matrices of shape `[n_samples, n_samples]`. These can be obtained from the functions in the `sklearn.metrics.pairwise` module.

2.3.1. Overview of clustering methods

RECOD
reasoning for complex data

Dimensionality Reduction

Machine Learning and Pattern Recognition

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886/MO444, September 25, 2018

Why is Dimensionality Reduction useful?

Why is Dimensionality Reduction useful?

- **Data Compression**
  - Reduce **time complexity**: less computation required
  - Reduce **space complexity**: less number of features
  - **More interpretable**: it removes noise

Why is Dimensionality Reduction useful?

- **Data Compression**
  - Reduce **time complexity**: less computation required
  - Reduce **space complexity**: less number of features
  - **More interpretable**: it removes noise
- **Data Visualization**

Why is Dimensionality Reduction useful?

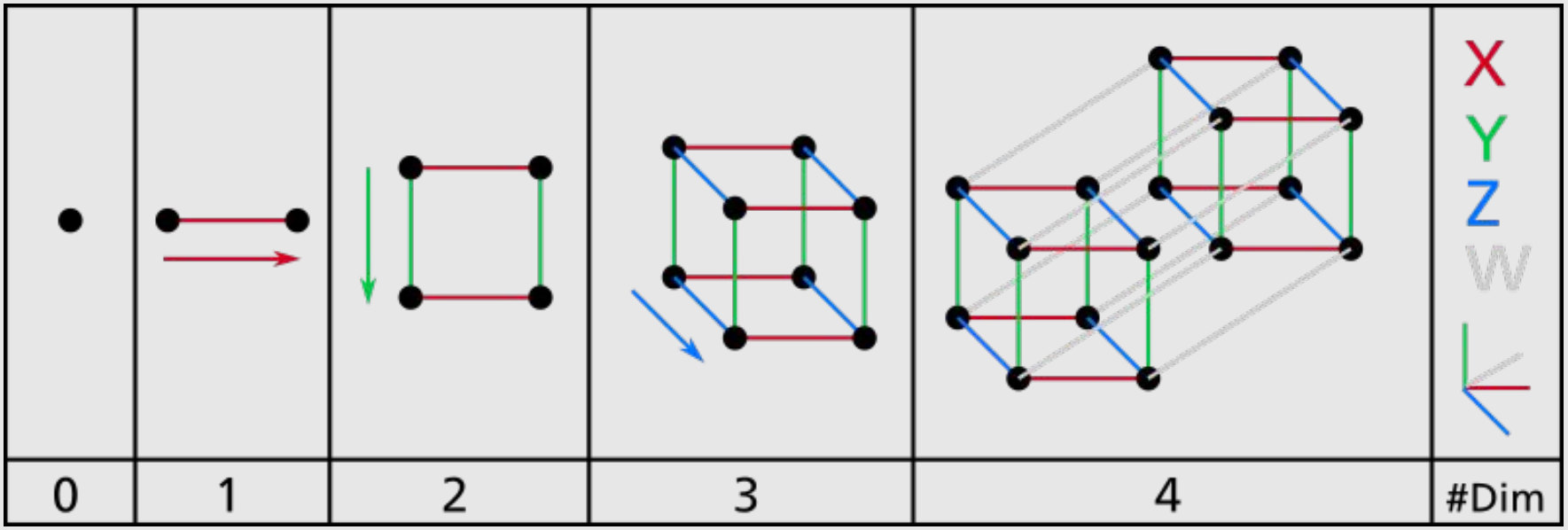
- **Data Compression**
  - Reduce **time complexity**: less computation required
  - Reduce **space complexity**: less number of features
  - **More interpretable**: it removes noise
- **Data Visualization**
- To mitigate **“the curse of dimensionality”**

Today's Agenda

- The Curse of Dimensionality
- PCA (Principal Component Analysis)
  - PCA Formulation
  - PCA Algorithm
  - Choosing *k*

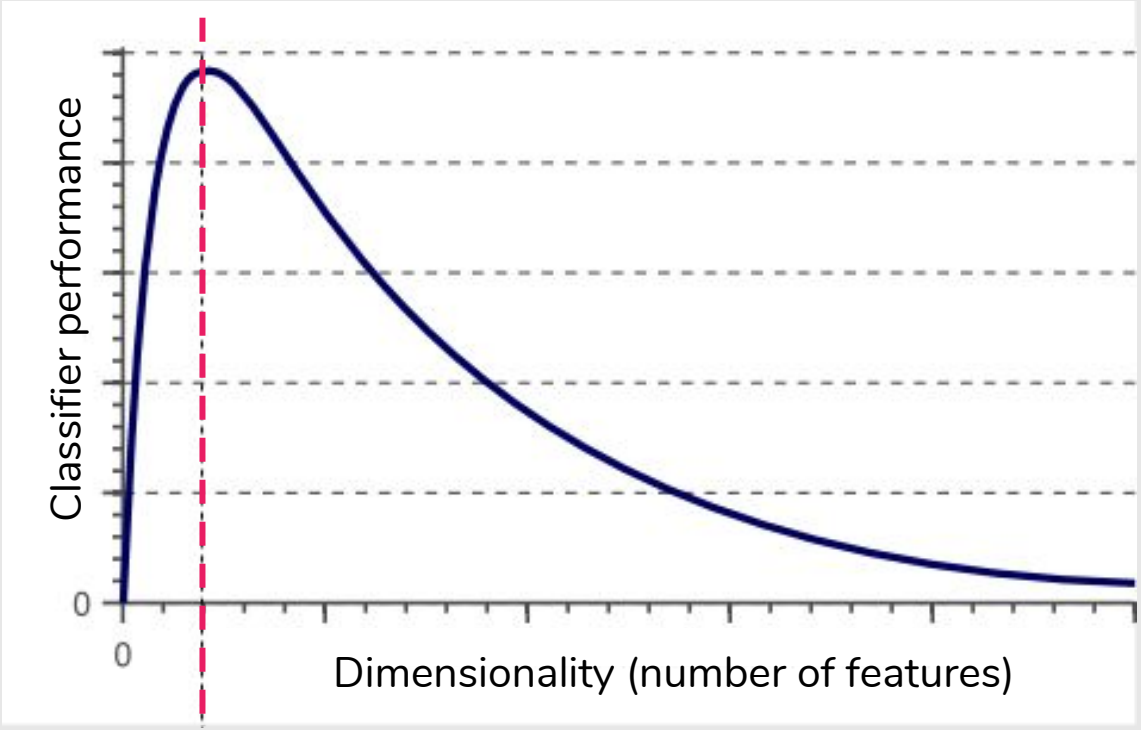
The Curse of Dimensionality

The Curse of Dimensionality



Even a basic 4D hypercube is incredibly hard to picture in our mind.

The Curse of Dimensionality



Optimal number of features

The Curse of Dimensionality

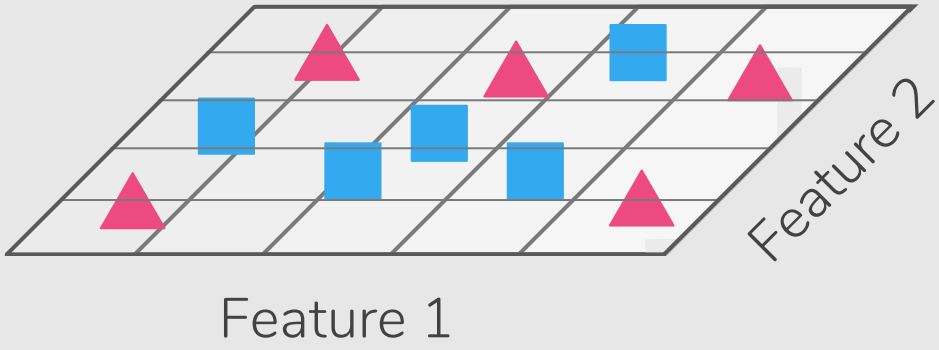
As the dimensionality of data grows, the density of observations becomes lower and lower and lower.

Feature 1

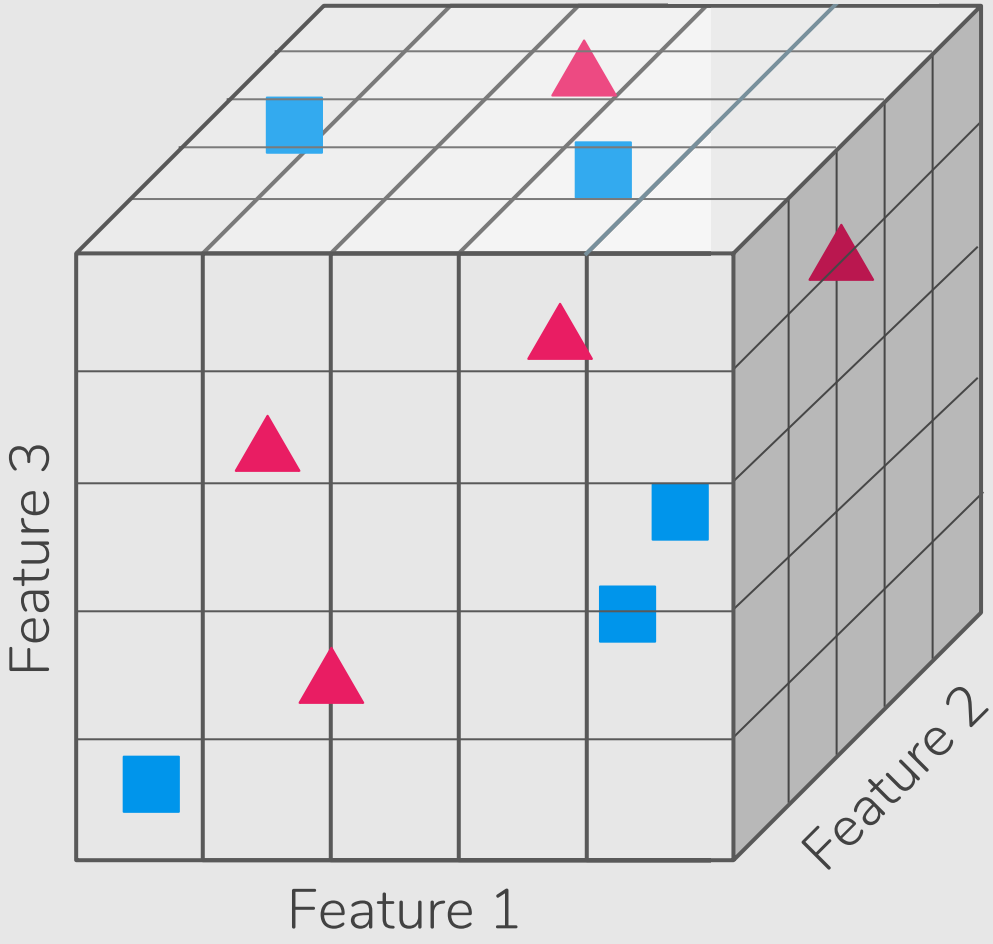
10 samples
1 dimension: 5 regions

The Curse of Dimensionality

As the dimensionality of data grows, the density of observations becomes lower and lower and lower.

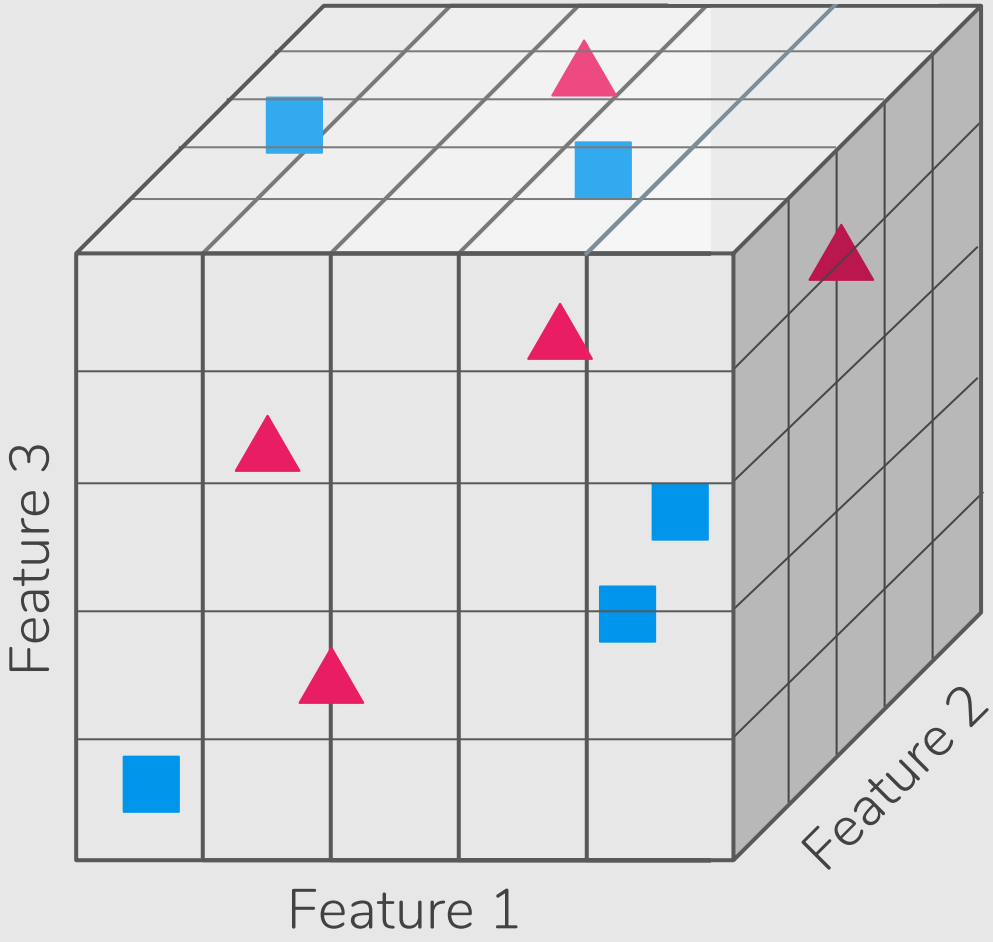


10 samples
2 dimensions: 25 regions



As the dimensionality of data grows, the density of observations becomes lower and lower and lower.

10 samples
3 dimensions: 125 regions



- 1 dimension: the sample density is 10/5 = 2 samples/interval
- 2 dimensions: the sample density is 10/25 = 0.4 samples/interval
- 3 dimensions: the sample density is 10/125 = 0.08 samples/interval

The Curse of Dimensionality: Solution?

The Curse of Dimensionality: Solution?

- Increase the size of the training set to reach a sufficient density of training instances.

The Curse of Dimensionality: Solution?

- Increase the size of the training set to reach a sufficient density of training instances.
- Unfortunately, the number of training instances required to reach a given density grows exponentially with the number of dimensions.

How to reduce dimensionality?

How to reduce dimensionality?

- Feature Selection

- Feature Extraction

How to reduce dimensionality?

- **Feature Selection:** choosing a subset of all the features (the ones more informative).
  - $\mathbf{x_1}, x_2, \mathbf{x_3}, x_4, \mathbf{x_5}$
- **Feature Extraction**

How to reduce dimensionality?

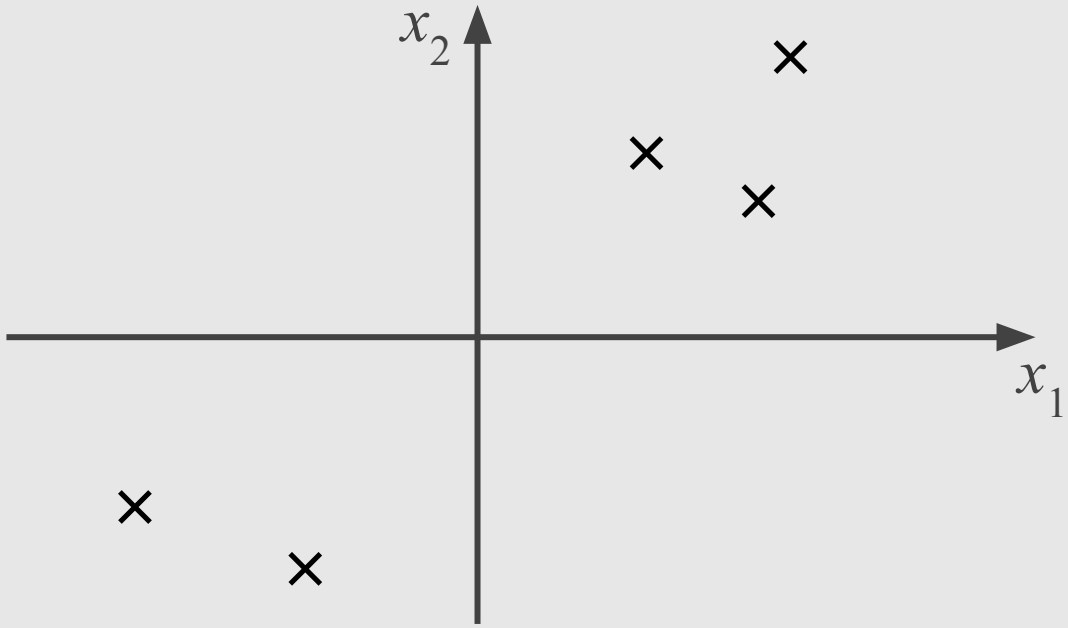
- **Feature Selection:** choosing a subset of all the features (the ones more informative).
  - $\mathbf{x_1}, x_2, \mathbf{x_3}, x_4, \mathbf{x_5}$
- **Feature Extraction:** create a subset of new features by combining the existing ones.
  - $z = f(x_1, x_2, x_3, x_4, x_5)$

PCA: Principal Component Analysis

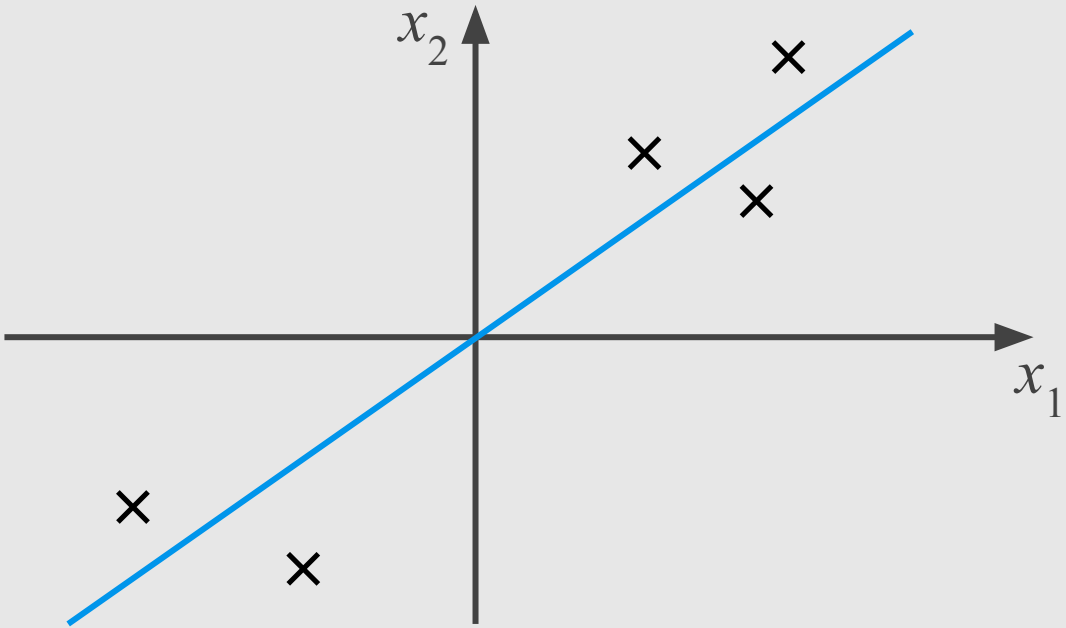
Principal Component Analysis (PCA)

- The most popular dimensionality reduction algorithm.
- PCA have two steps:
  - It **identifies the hyperplane** that lies closest to the data.
  - It **projects** the data onto it.

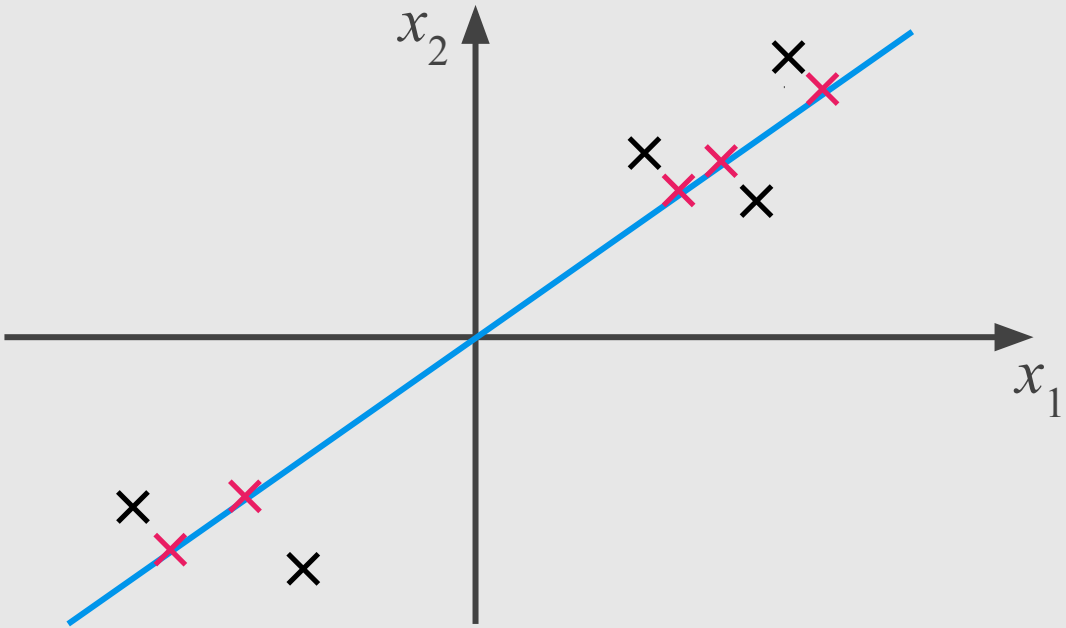
Problem Formulation (PCA)



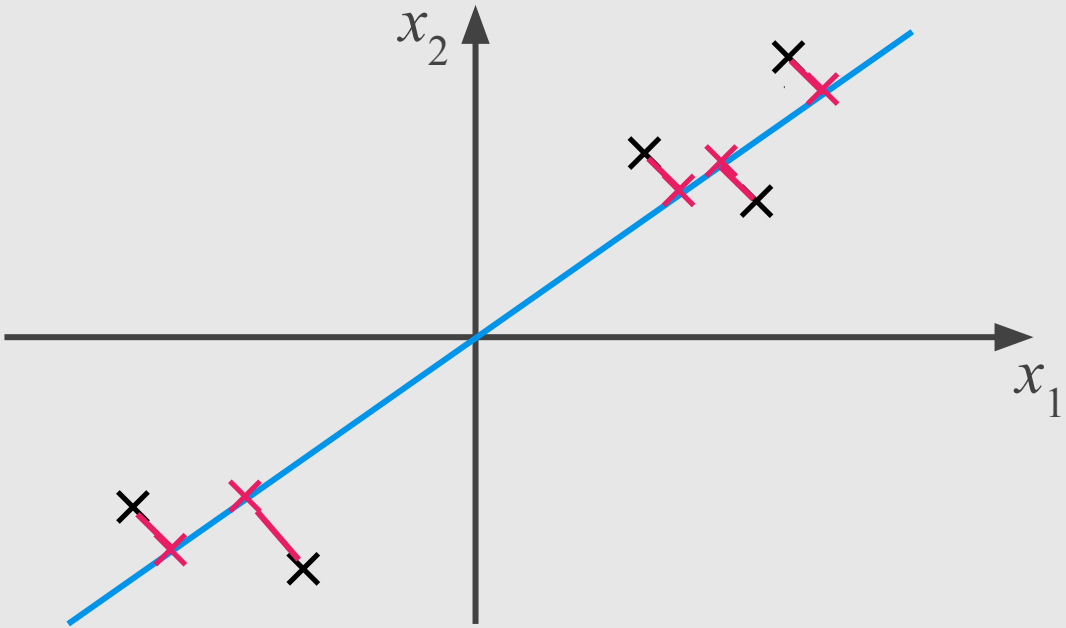
Problem Formulation (PCA)



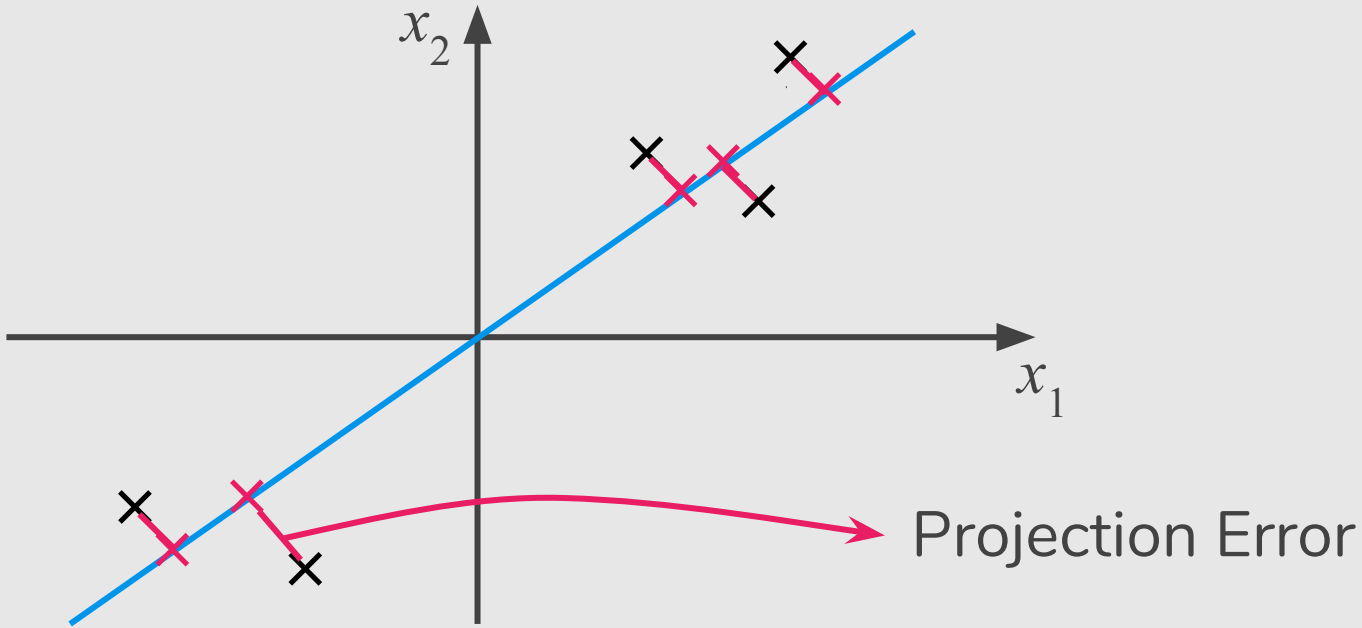
Problem Formulation (PCA)



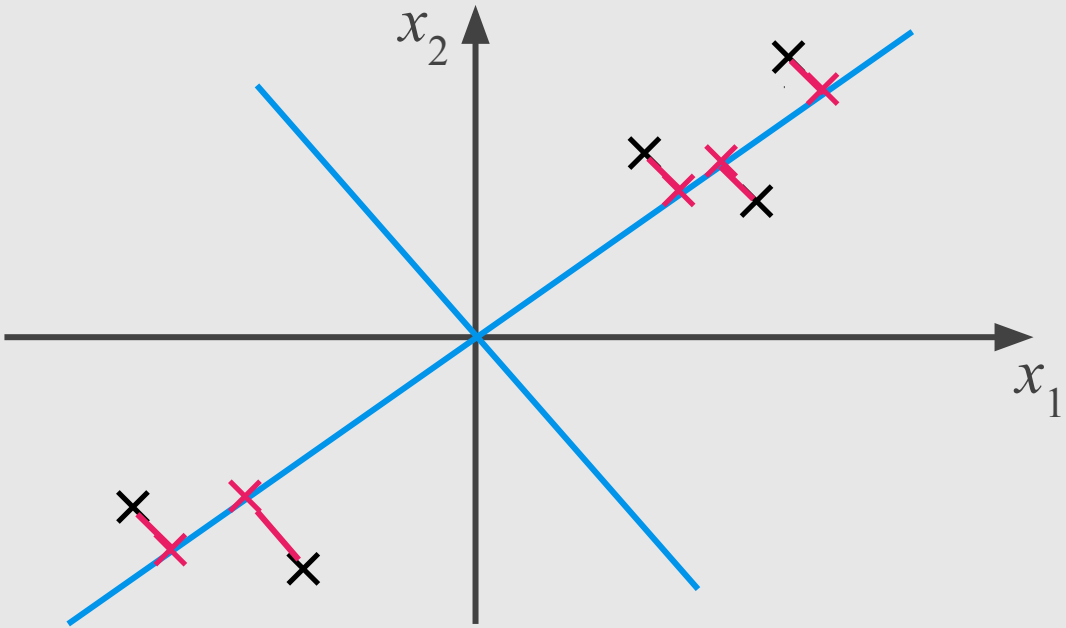
Problem Formulation (PCA)



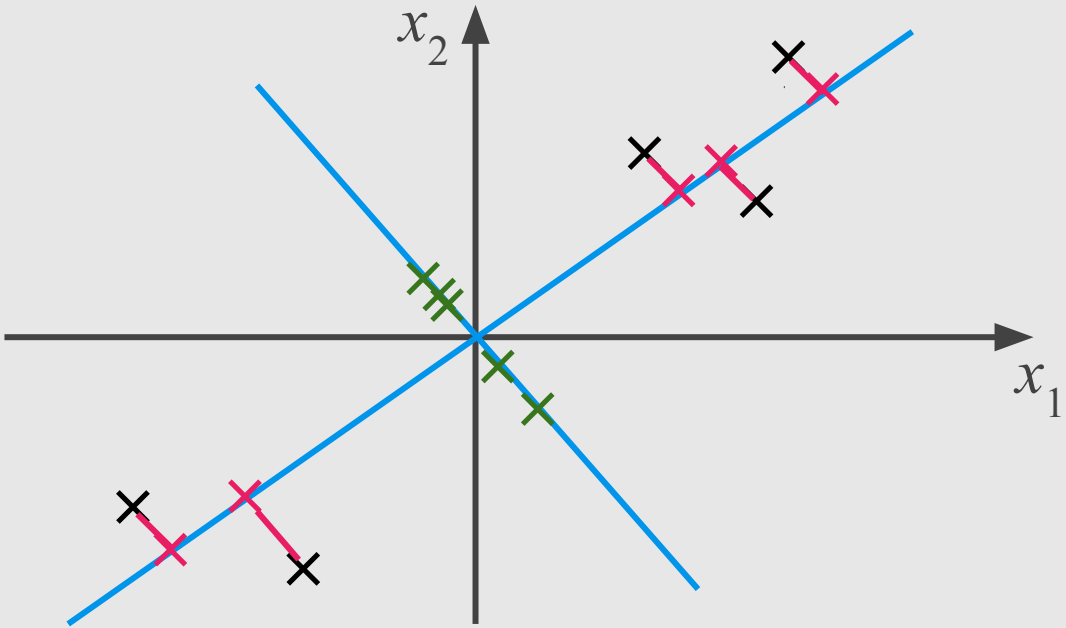
Problem Formulation (PCA)



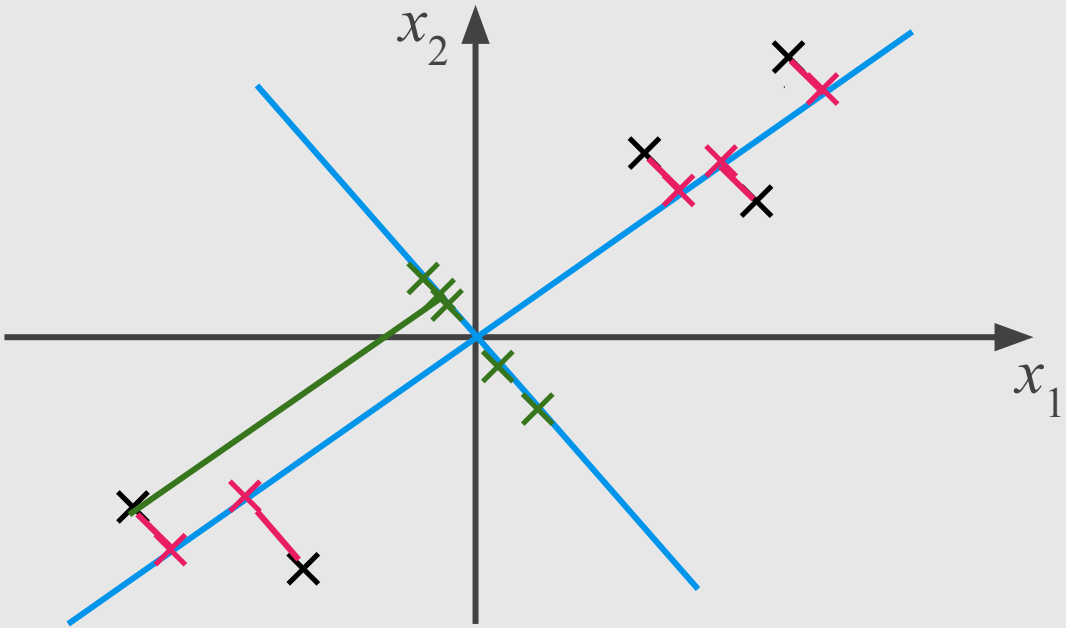
Problem Formulation (PCA)



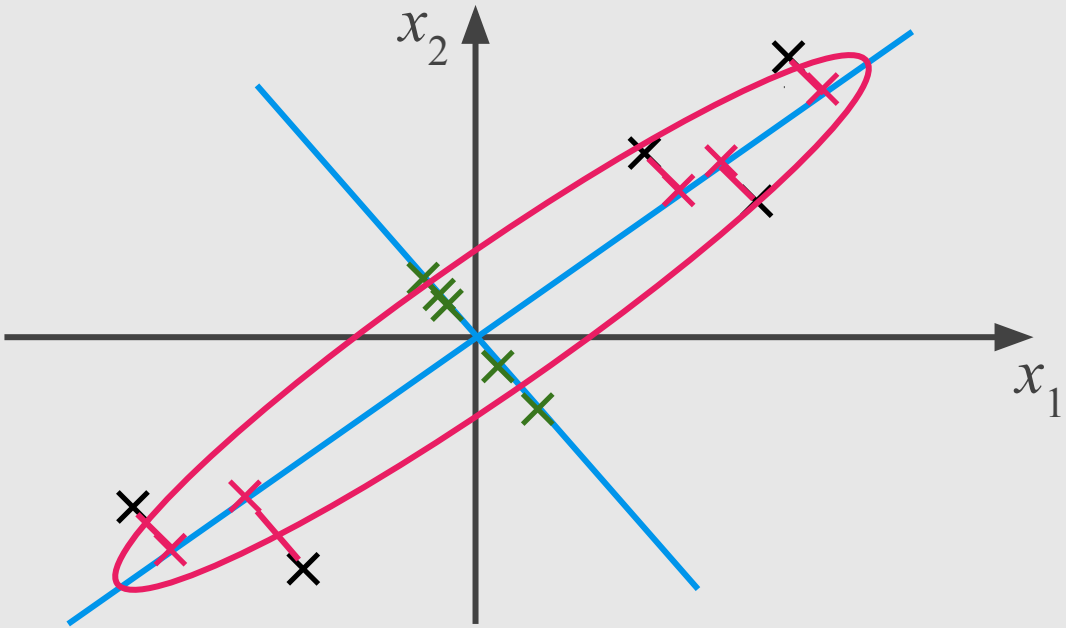
Problem Formulation (PCA)



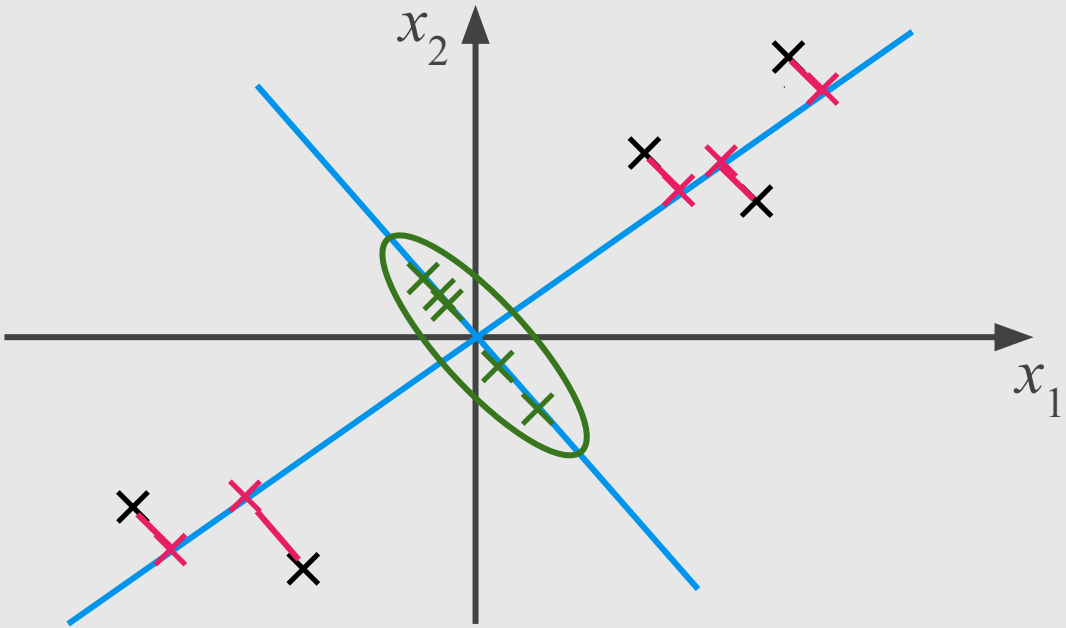
Problem Formulation (PCA)



Problem Formulation (PCA)

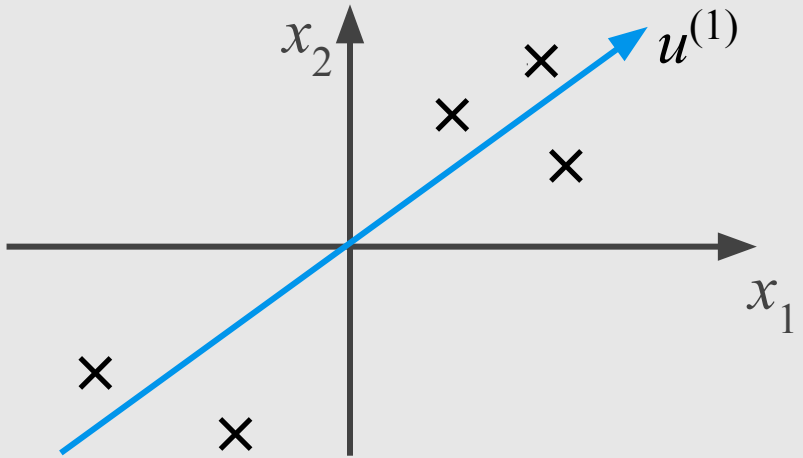


Problem Formulation (PCA)



Problem Formulation (PCA)

- Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.



Problem Formulation (PCA)

- Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA Algorithm
By Singular Value Decomposition

Data Preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

Center the data

Data Preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

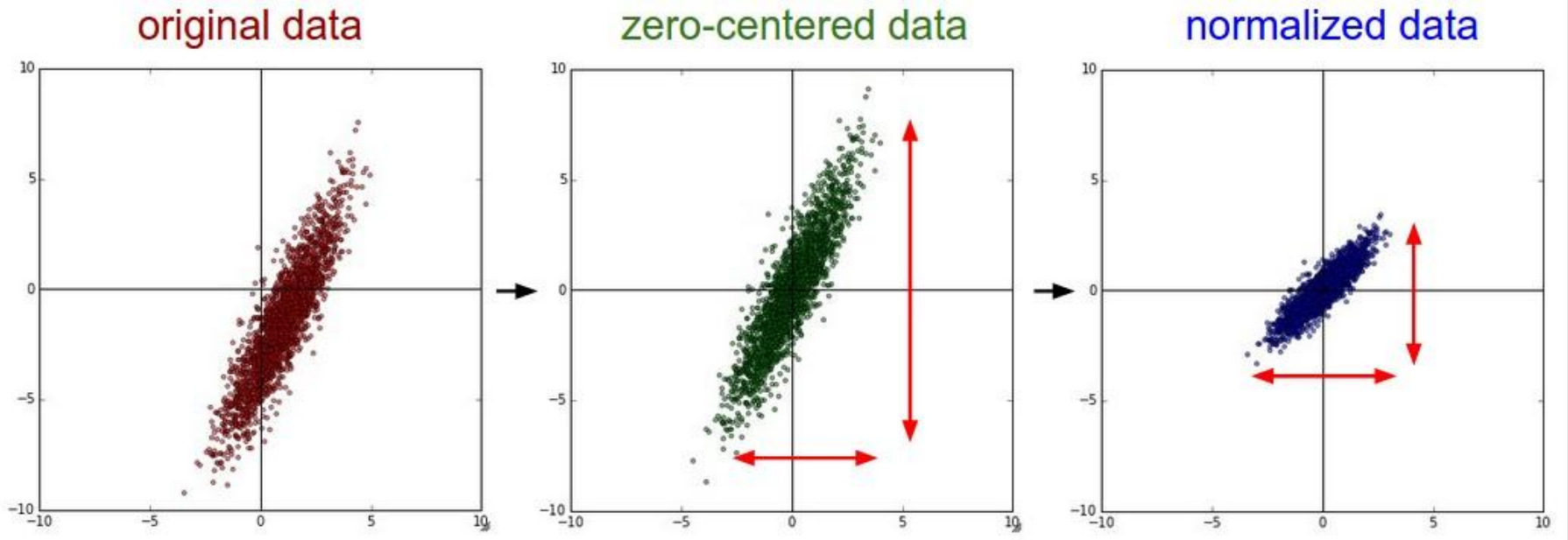
$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

Center the data

If different features on different scales, scale features to have comparable range of values.

Data Preprocessing



Credit: http://cs231n.github.io/neural-networks-2/

PCA Algorithm

Reduce data from n-dimensions to k-dimensions

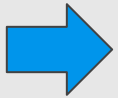
Compute “covariance matrix”:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}}$$

PCA Algorithm

Reduce data from n-dimensions to k-dimensions

Compute “covariance matrix”:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}}$$  $n \times n$ matrix

PCA Algorithm

Reduce data from n-dimensions to k-dimensions

Compute “covariance matrix”:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}}$$ ⇒ $n \times n$ matrix

Compute “eigenvectors” of matrix $\boldsymbol{\Sigma}$:

[U, S, V] = svd(sigma) ⇒ Singular Value Decomposition

PCA Algorithm

Reduce data from n-dimensions to k-dimensions

Compute “covariance matrix”:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}}$$ ⇒ $n \times n$ matrix

Compute “eigenvectors” of matrix $\mathbf{\Sigma}$:

[U, S, V] = svd(sigma) ⇒ Singular Value Decomposition

PCA Algorithm

From [U, S, V] = svd(sigma), we get:

$$U = \begin{bmatrix} | & | & | \\ u^{(1)} & \cdots & u^{(n)} \\ | & | & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

PCA Algorithm

From [U, S, V] = svd(sigma), we get:

$$U = \underbrace{\begin{bmatrix} | & | & | \\ u^{(1)} & \cdots & u^{(n)} \\ | & | & | \end{bmatrix}}_{k} \in \mathbb{R}^{n \times n} \qquad x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

PCA Algorithm

From [U, S, V] = svd(sigma), we get:

$$U = \underbrace{\begin{bmatrix} | & | & | \\ u^{(1)} & \cdots & u^{(n)} \\ | & | & | \end{bmatrix}}_{k} \in \mathbb{R}^{n \times n}$$

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

$$z = \underbrace{\begin{bmatrix} | & | & | \\ u^{(1)} & \cdots & u^{(k)} \\ | & | & | \end{bmatrix}^{\mathrm{T}}}_{k \times n} \underbrace{x}_{n \times 1}$$

PCA Algorithm

After mean normalization and optionally feature scaling:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}}$$

[U, S, V] = svd(sigma)

$z = (\mathrm{U}_{\mathrm{reduce}})^{\mathrm{T}} \times x$

Choosing the Number of Principal Components

Choosing k (#Principal Components)

Typically, choose k to be smallest value so that:

$$\frac{1/m \sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{1/m \sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01$$

“99% of variance is retained”

Choosing k (#Principal Components)

Typically, choose k to be smallest value so that:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01$$

Average squared projection error

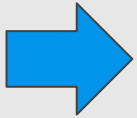
Total variation in the data

"99% of variance is retained"

Choosing k (#Principal Components)

[U, S, V] = svd(sigma)

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}$$

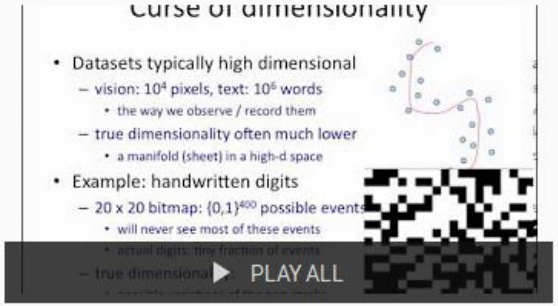


$$1-\frac{\sum_{i=1}^{k}s_{ii}}{\sum_{i=1}^{n}s_{ii}}$$

Principal Component Analysis (12 videos, 3-15min)

https://www.youtube.com/playlist?list=PLBv09BD7ez_5_yapAg86Od6JeeypkS4YM

Search



Principal Component Analysis

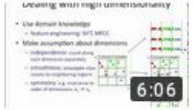
12 videos • 119,895 views • Last updated on May 21, 2014

Victor Lavrenko SUBSCRIBE 19K

Lectures 18 and 19 in the Introductory Applied Machine Learning (IAML) course by Victor Lavrenko at the

1 

PCA 1: curse of dimensionality
Victor Lavrenko

2 

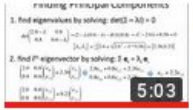
PCA 2: dimensionality reduction
Victor Lavrenko

3 

PCA 3: direction of greatest variance
Victor Lavrenko

4 

PCA 4: principal components = eigenvectors
Victor Lavrenko

5 

PCA 5: finding eigenvalues and eigenvectors
Victor Lavrenko

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 8 “Dimensionality Reduction”
- Pattern Recognition and Machine Learning, Chap. 12 “Continuous Latent Variables”
- Pattern Classification, Chap. 10 “Unsupervised Learning and Clustering”

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 8