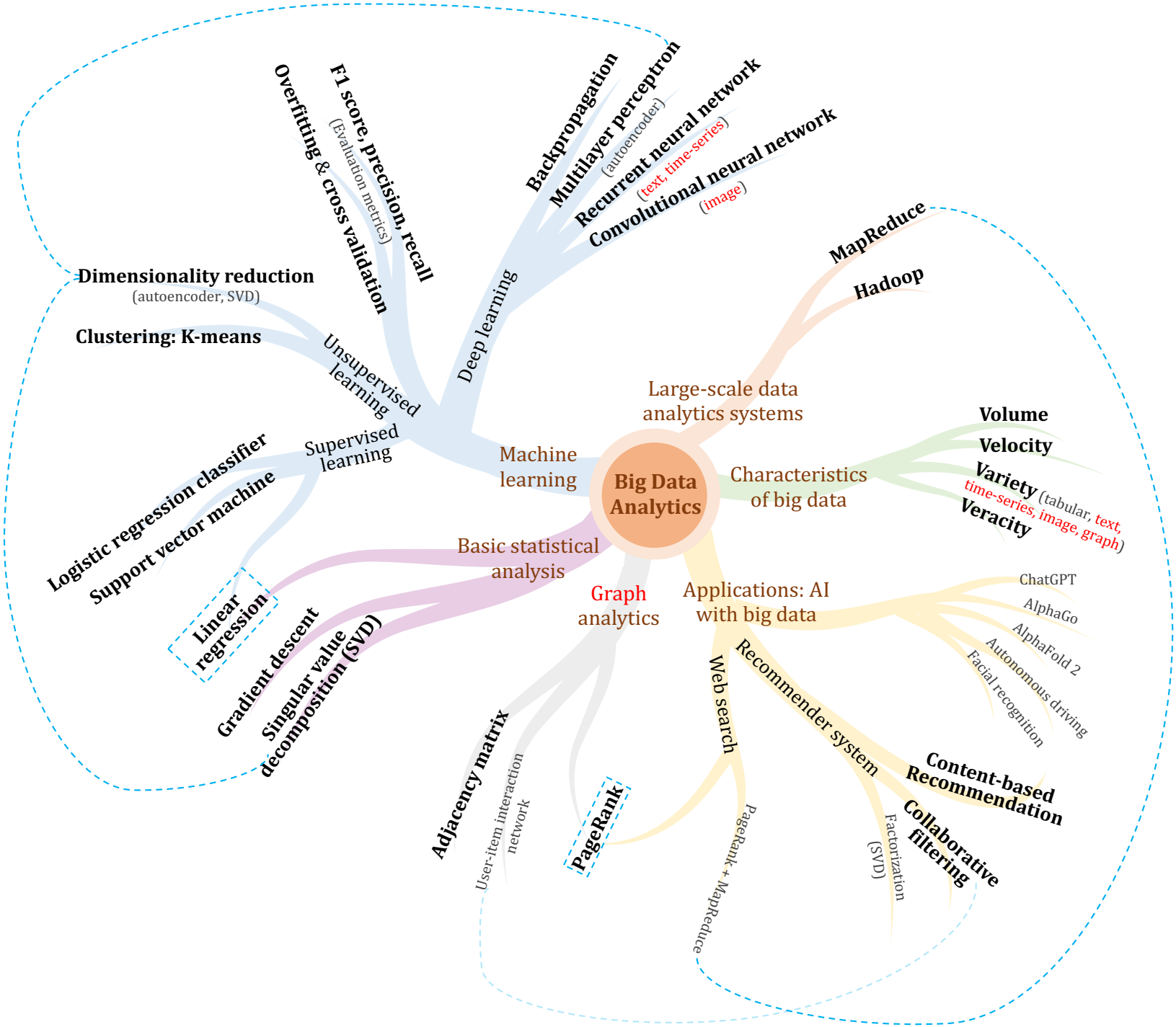
THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP4434 Big Data Analytics

Lecture 4 Support Vector Machines, & Clustering

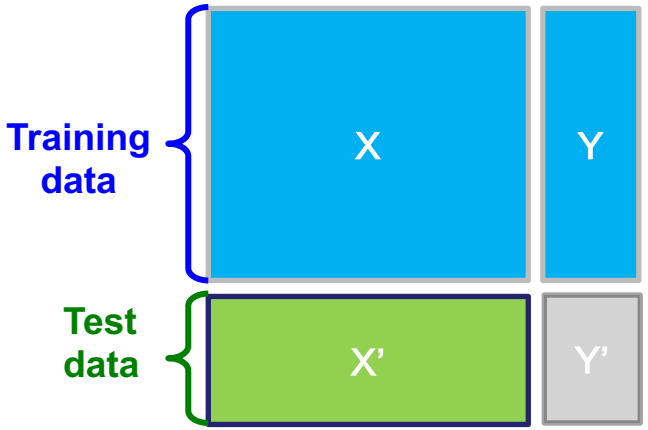
HUANG Xiao

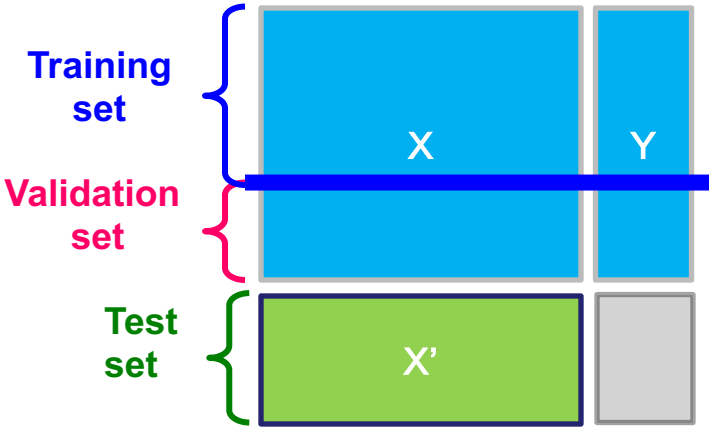
xiaohuang@comp.polyu.edu.hk


Big Data Analytics
Machine learning
Deep learning
Backpropagation
Multilayer perceptron
(autoencoder)
Recurrent neural network
(text, time-series)
Convolutional neural network
(image)
Unsupervised learning
Dimensionality reduction
(autoencoder, SVD)
Clustering: K-means
Supervised learning
F1 score, precision, recall
(Evaluation metrics)
Overfitting & cross validation
Logistic regression classifier
Support vector machine
Linear regression
Basic statistical analysis
Gradient descent
Singular value decomposition (SVD)
Large-scale data analytics systems
MapReduce
Hadoop
Characteristics of big data
Volume
Velocity
Variety (tabular, text, time-series, image, graph)
Veracity
Applications: AI with big data
ChatGPT
AlphaGo
AlphaFold 2
Autonomous driving
Facial recognition
Recommender system
Content-based Recommendation
Collaborative filtering
Factorization (SVD)
Web search
PageRank + MapReduce
Graph analytics
PageRank
Adjacency matrix
User-item interaction network

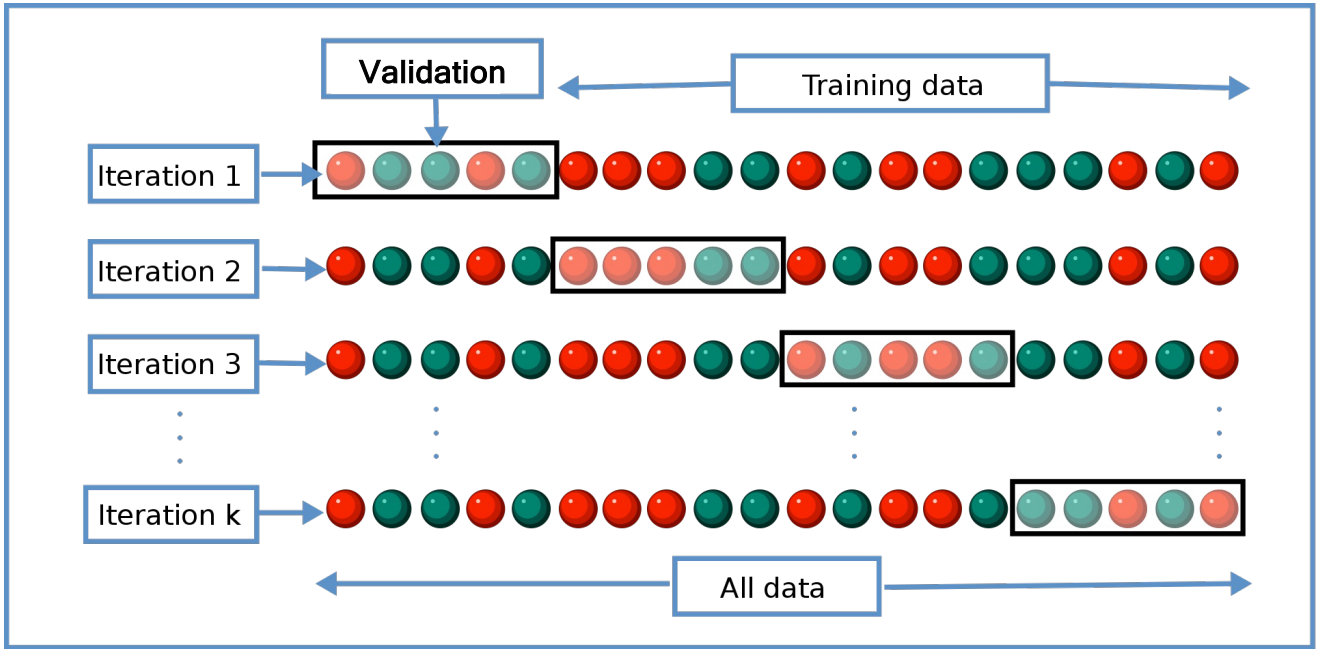
Validation set

- **Task:** Given data **(X,Y)** build a model **f()** to predict **Y'** based on **X'**
- **Strategy:**
  Estimate $y = f(x)$ on (X, Y)
  Hope that the same $f(x)$ also works to predict unknown Y'
  - The **"hope"** is called **generalization**
  - **Overfitting:** If f(x) predicts well Y but is unable to predict Y'
  - We want to build a model that generalizes well to unseen data
  - Solution: **k-fold Cross-validation**



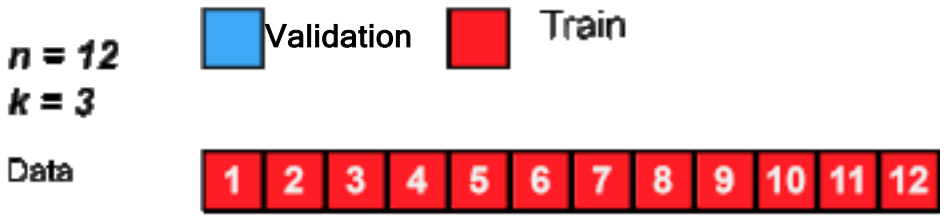


k-fold Cross-validation

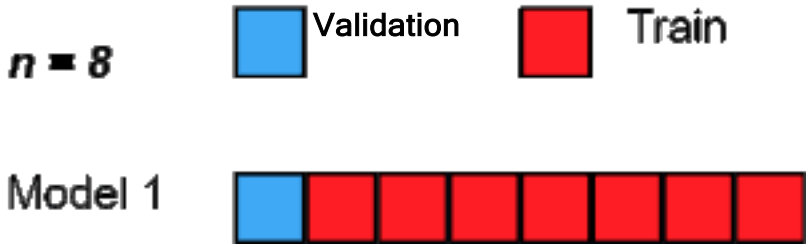


- The original sample is randomly partitioned into k equal sized subsamples
- Of the k subsamples, a single subsample is retained as the validation data for testing the model
- The remaining k − 1 subsamples are used as training data
- The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data
- The k results can then be averaged to produce a single estimation

Leave-one-out Cross-validation



k-fold cross-validation



leave-one-out cross-validation

- When k = n (the number of observations), k-fold cross-validation is equivalent to leave-one-out cross-validation

Boston Housing (has an ethical problem)

The Boston Housing Dataset consists of price of houses in various places in Boston. The Boston Housing Dataset has 506 cases. There are **13** Features in each case of the dataset. Alongside with price, the dataset also provide information such as Crime (CRIM), areas of non-retail business in the town (INDUS), the age of people who own the house (AGE), and there are many other attributes.

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTST	Price
0.006	18.0	2.31	0.0	0.538	6.575	65.2	4.090	1.0	296.0	15.3	396.9	4.98	24.0
0.027	0	7.07	0.0	0.469	6.421	78.9	4.967	2.0	242.0	17.8	396.9	9.14	21.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...

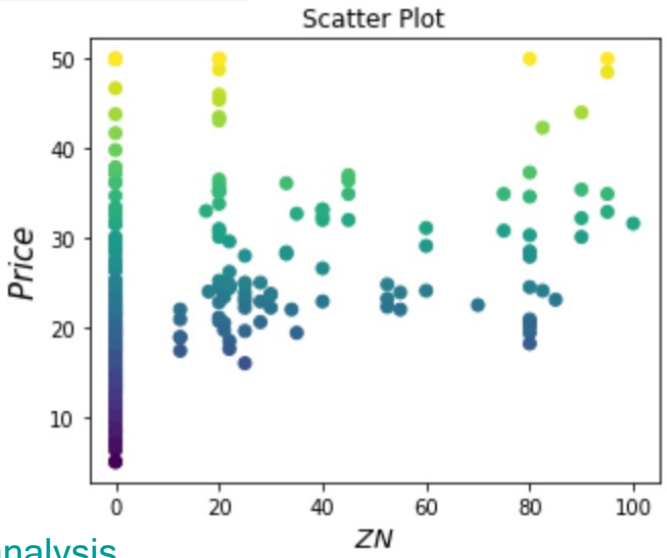
Generate Training Data

```
In [41]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.datasets import load_boston, load_diabetes
         from sklearn.model_selection import train_test_split

         np.random.seed(42)

         def load_data():
             dataset = load_boston()
             print(dataset.feature_names)
             return train_test_split(dataset.data, dataset.target, test_size=0.25, random_state=0)

         X_train, X_test, Y_train, Y_test = load_data()
         print(X_train.shape)

         plt.figure(figsize=(5,4))
         plt.scatter(X[:,1],y,c=y)
         plt.ylabel("$Price$", fontsize=15)
         plt.xlabel("$ZN$", rotation=0, fontsize=15)
         plt.title('Scatter Plot')
         plt.show()

         ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO
          'B' 'LSTAT']
         (379, 13)
```

Boston Housing Data

Split dataset

Plot figure



ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

https://www.kaggle.com/tolgahancepel/boston-housing-regression-analysis

Build Model

Train model

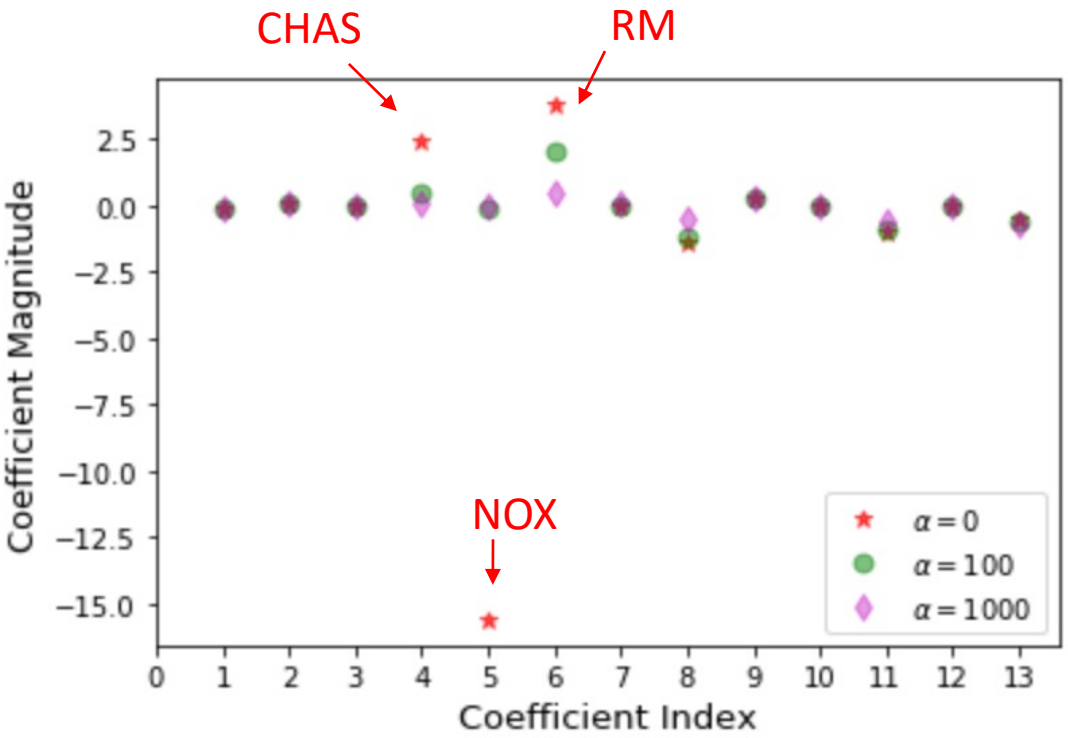
Cross validation

In [56]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score

alpha = 0
model = Ridge(alpha=alpha, solver='auto', random_state=42)

model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
cross_valid = cross_val_score(model, data, target, scoring='neg_mean_squared_error', cv = 5)
print('Cross Validation Errors:\n', -np.mean(cross_valid))
print('theta 0: \n', model.intercept_)
print('theta 1-13: \n', model.coef_)
```

```
Cross Validation Errors:
 37.13180746769889
theta 0:
 36.933255457119316
theta 1-13:
 [-1.17735289e-01  4.40174969e-02 -5.76814314e-03  2.39341594e+00
 -1.55894211e+01  3.76896770e+00 -7.03517828e-03 -1.43495641e+00
  2.40081086e-01 -1.12972810e-02 -9.85546732e-01  8.44443453e-03
 -4.99116797e-01]
```

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_{13} x_{13}$$

Regularization



α	MSE
0	37.1318 (overfitting)
100	29.9057
1000	32.8280 (underfitting)

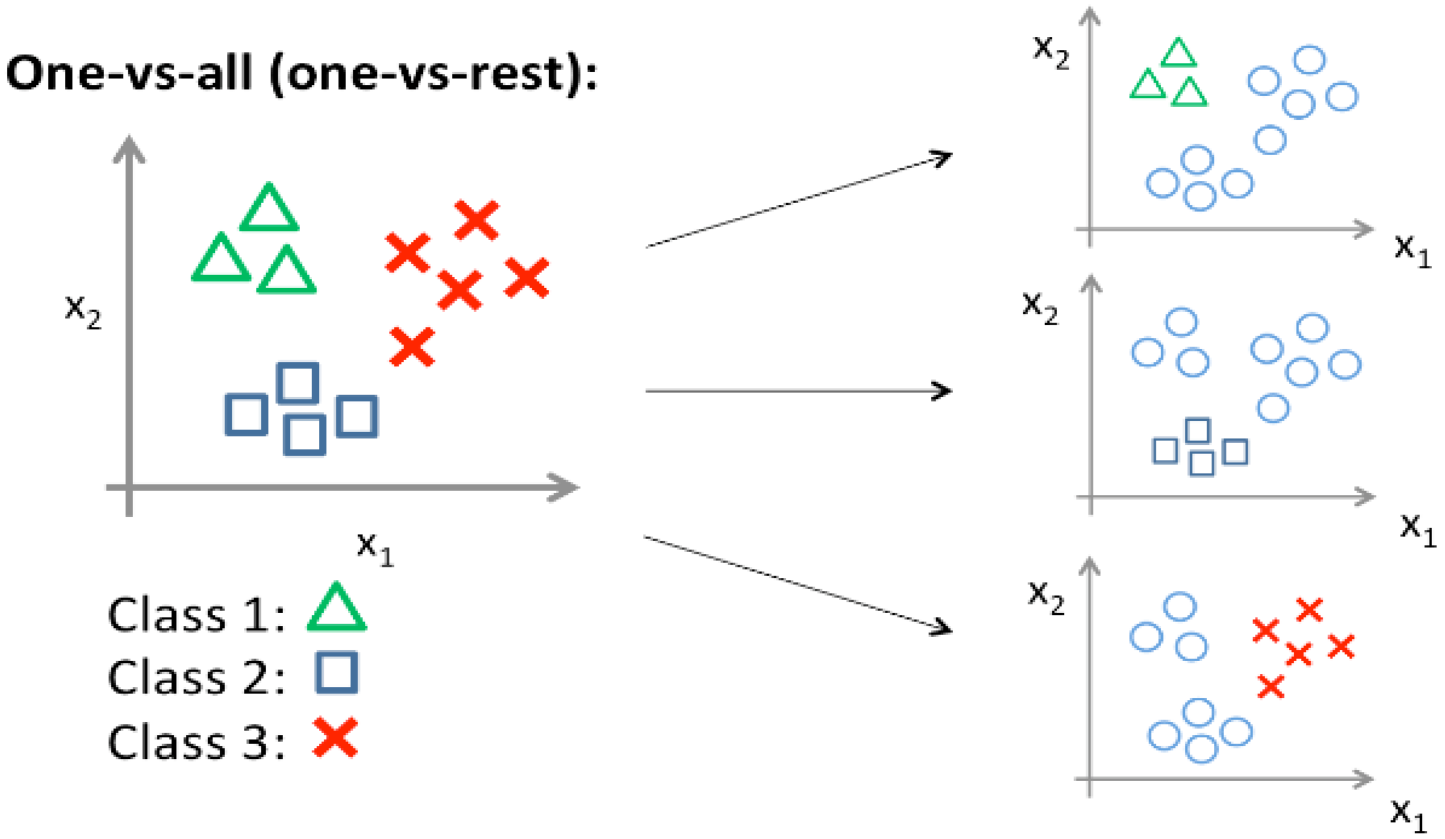
The magnitudes of coefficient indices 4,5,6 are considerably reduced after regularization with α = 100, resulting in lower mean square error

How about Multi-class Classification

- Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability of $y = 1$

- On a new input x, pick the class that maximizes $\max_i \left(h_\theta^{(i)}(x) \right)$

One-vs-All Approach



Exercise

- Assume that there is a classification problem with 4 classes. Each instance has 5 features. What is the total number of parameters, if you are solving it by using linear logistic regression and one-vs-all approach? Remember to include θ_0.

- We have 4 classes, so we need 4 binary classifiers. In each classifier, we have θ_0, θ_1, θ_2, ..., θ_5. Thus, in total, we have 4*6 = 24 parameters.

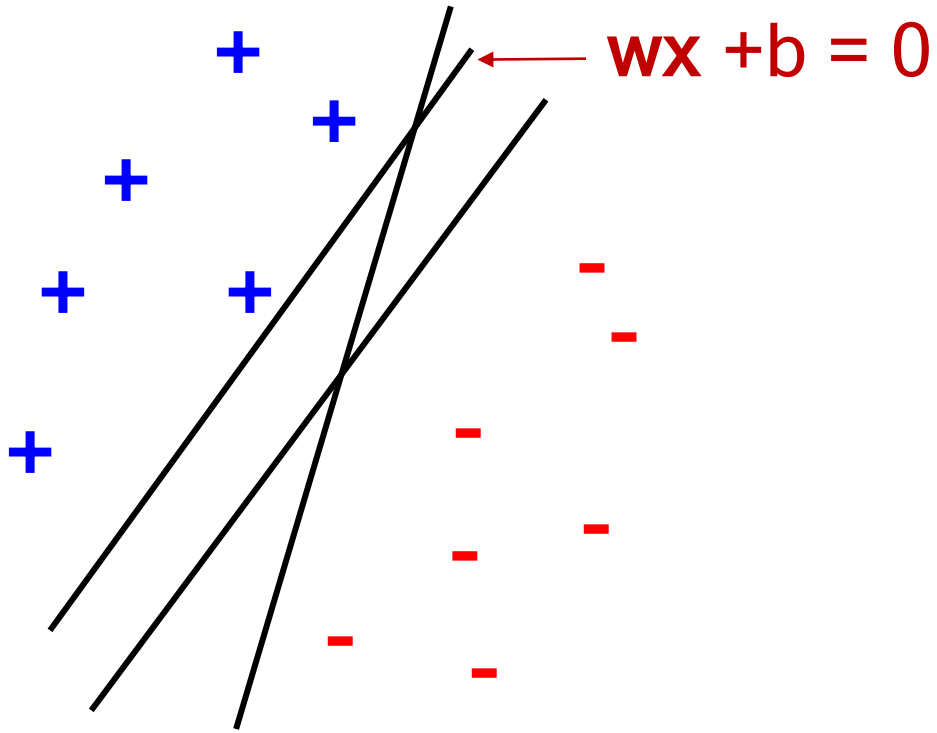
History of Support Vector Machines

- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

Support Vector Machines

- Want to separate "+" from "-" using a line

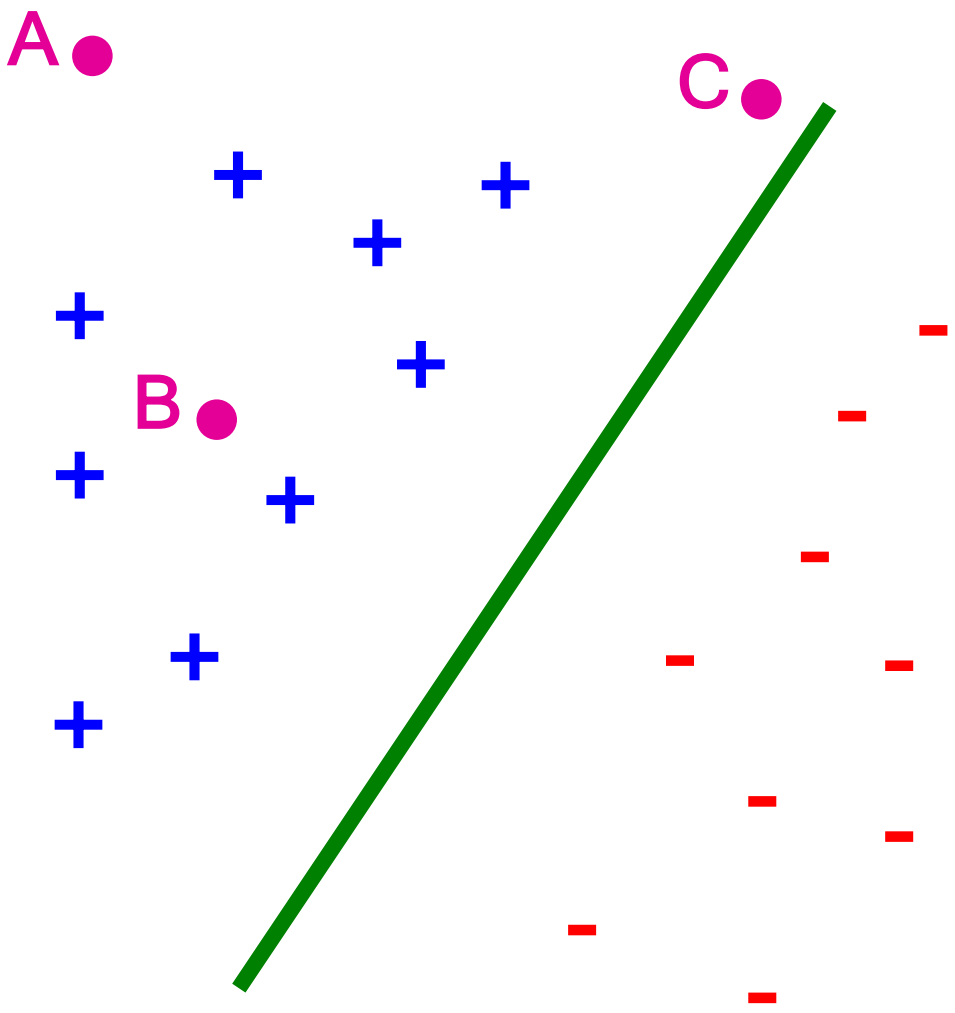


Data:

- **Training examples:**
  - $(x^{(1)}, y_1) \ldots (x^{(m)}, y_m)$
- **Each example *i*:**
  - $x^{(i)} = (x_1^{(i)}, \ldots , x_d^{(i)})$
    - $x_j^{(i)}$ is real valued
  - $y_i \in \{ -1, +1 \}$

Which is best linear separator (defined by *w*)?

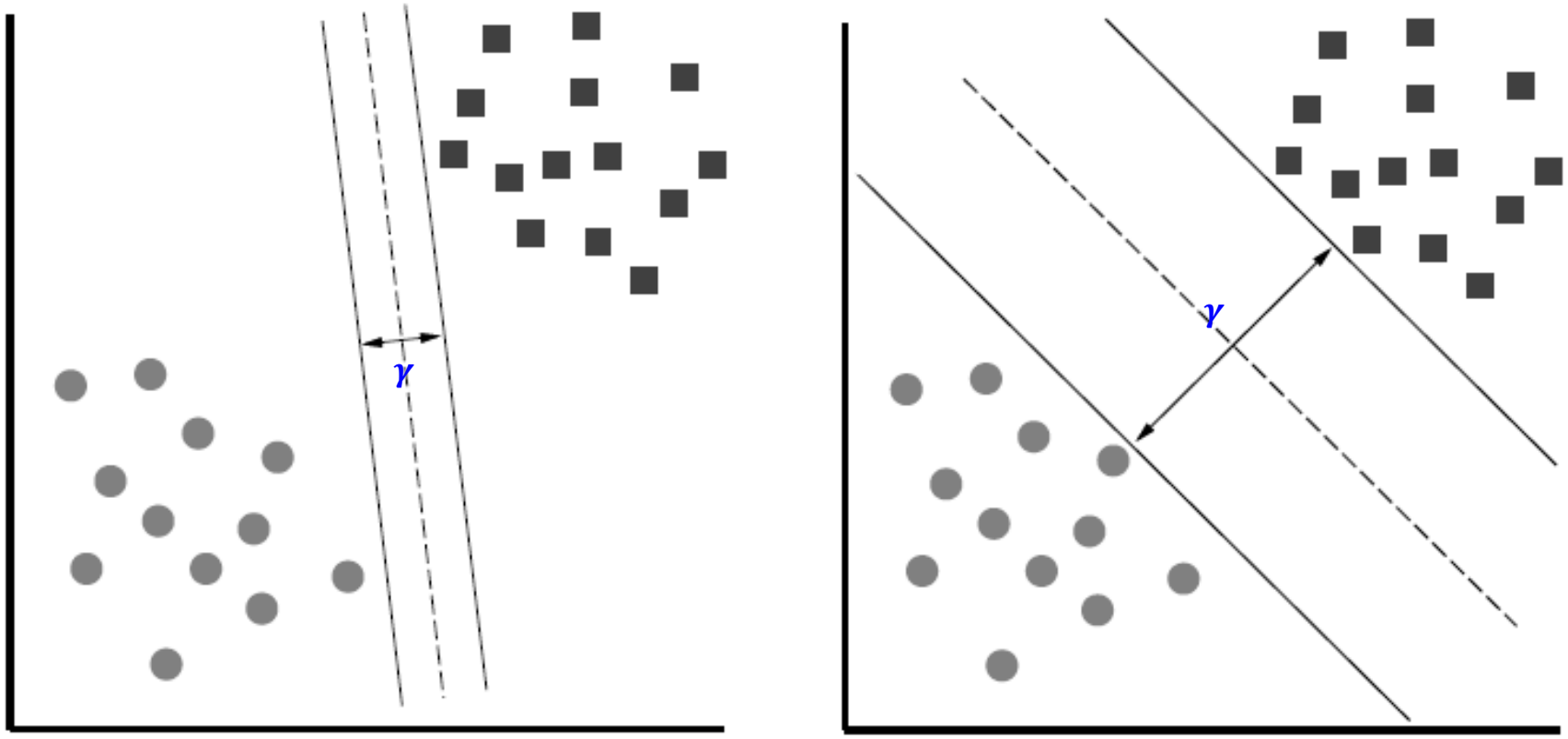
Largest Margin



- Distance from the separating hyperplane corresponds to the "confidence" of prediction
- **Example:**
  - We are more confident about the class of **A** and **B** than of **C**

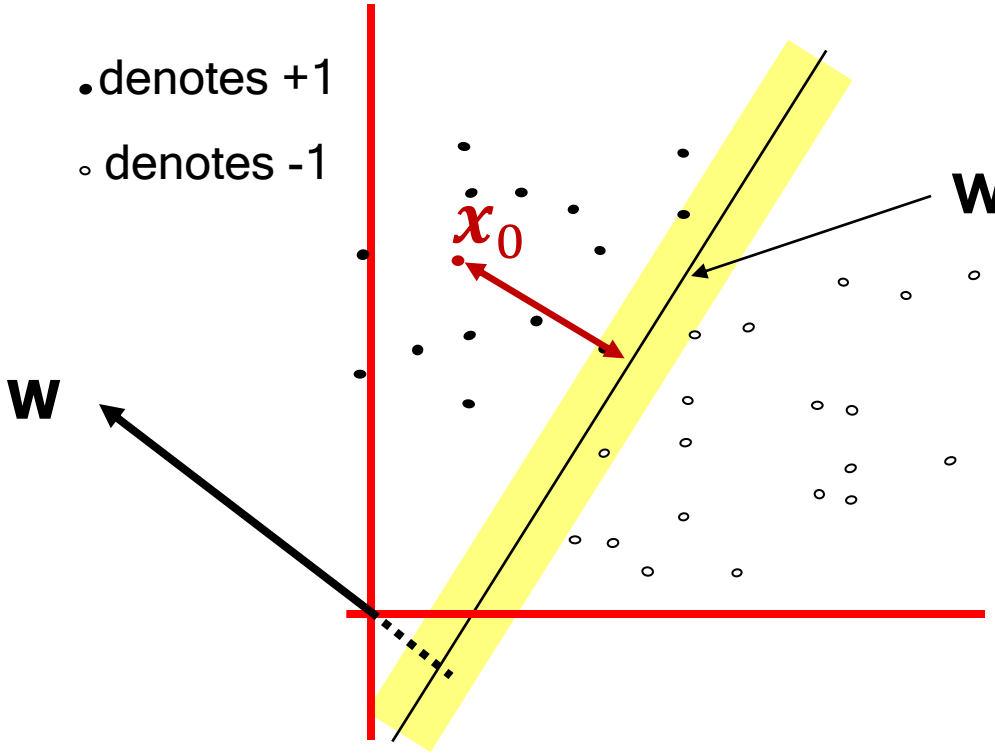
Largest Margin

- **Margin γ (gamma): Distance of closest example from the decision line/hyperplane**



The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.

Distance from a point to a line



x – Vector

w – Normal Vector

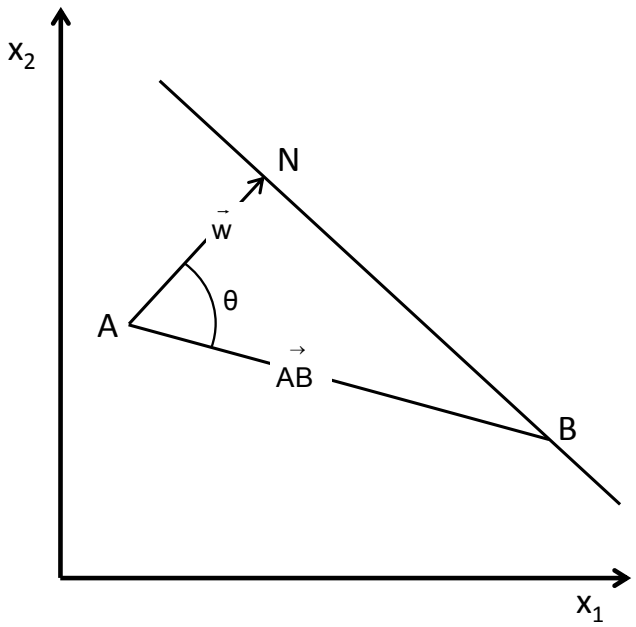
b – Scale Value

- What is the distance expression for a point x_0 to a line **wx**+b= 0?

$$d(\mathbf{x}_0) = \frac{|\mathbf{x}_0 \cdot \mathbf{w} + b|}{\sqrt{||\mathbf{w}||_2^2}} = \frac{|\mathbf{x}_0 \cdot \mathbf{w} + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

- http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

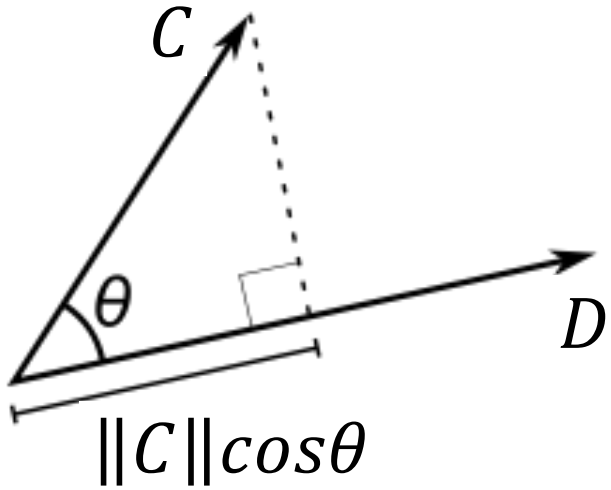
Distance from a point to a line (method 2)



$$\left\|\overrightarrow{AN}\right\| = \left\|\overrightarrow{AB}\right\| \cos\theta = \left\|\overrightarrow{AB}\right\| \frac{\overrightarrow{AB}\,\vec{w}}{\left\|\overrightarrow{AB}\right\| \left\|w\right\|} = \frac{\overrightarrow{AB}\,\vec{w}}{\left\|w\right\|}$$

$$= \frac{(x_{B1} - x_{A1}, x_{B2} - x_{A2})^\top (-w_1, -w_2)}{\|w\|}$$

$$= \frac{w^T x_A - \overbrace{w^T x_B}^{=-b}}{\left\|w\right\|} = \frac{w^T x_A + b}{\left\|w\right\|}$$

- 

$$C \cdot D = \|C\| \cdot \|D\| \cdot \cos\theta$$

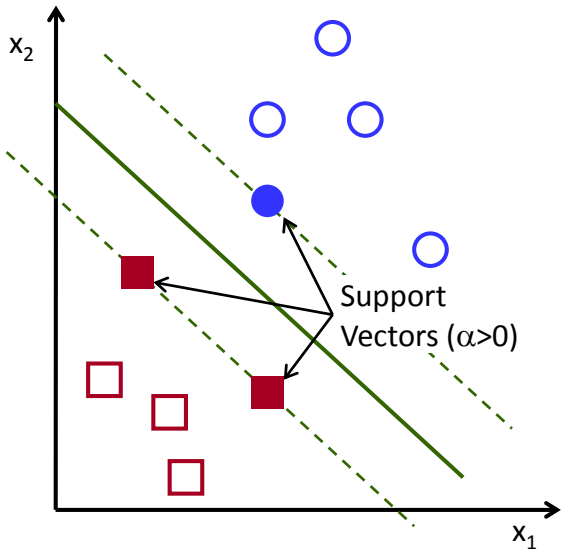
Linear SVM Mathematically

- Let training set $\{(\mathbf{x}^{(i)}, y_i)\}_{i=1..n}$, $\mathbf{x}^{(i)} \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin γ. Then for each training example $(\mathbf{x}^{(i)}, y_i)$:

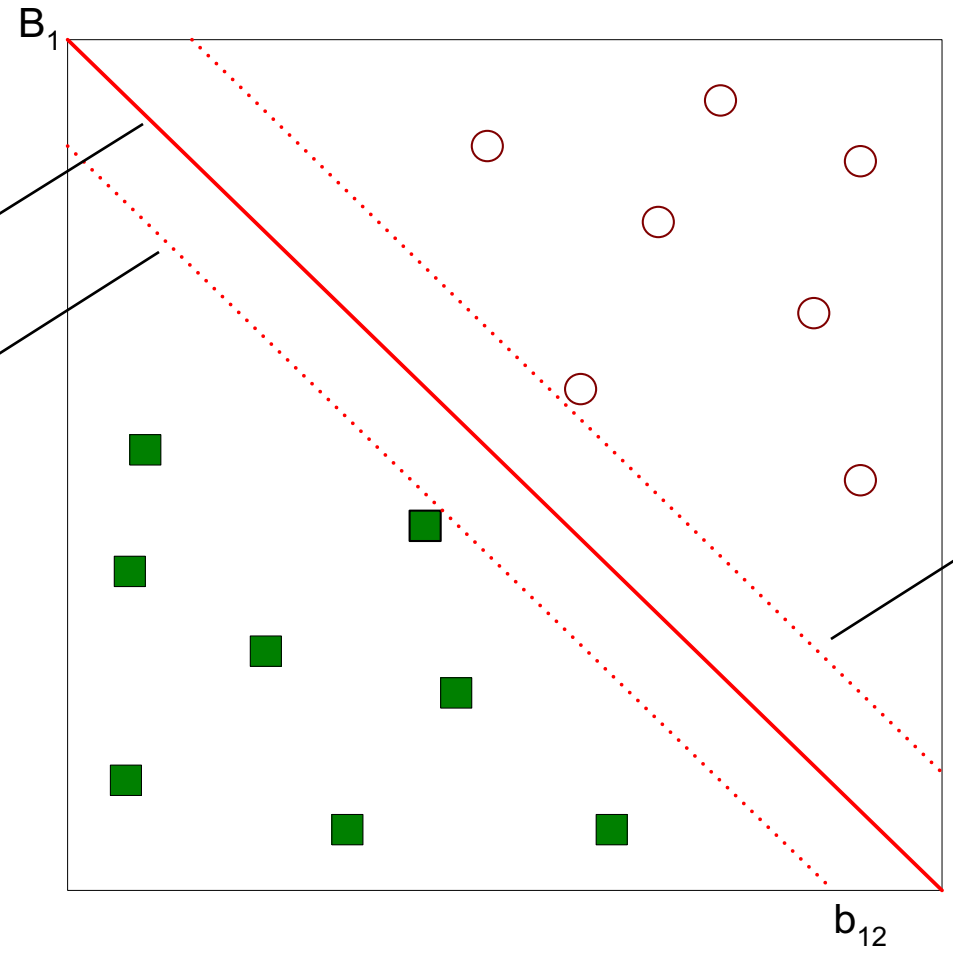
$$\mathbf{w}^T\mathbf{x}^{(i)} + b \leq -\gamma/2 \quad \text{if } y_i = -1$$
$$\mathbf{w}^T\mathbf{x}^{(i)} + b \geq \gamma/2 \quad \text{if } y_i = 1$$
$$\Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq \gamma/2$$

- For every support vector $\mathbf{x}^{(s)}$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\gamma/2$ in the equality, we obtain that distance between each $\mathbf{x}^{(s)}$ and the hyperplane is

$$\frac{y_s(\mathbf{w}^\top \mathbf{x}^{(s)} + b)}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||}$$



Linear Support Vector Machine (SVM)



$\mathbf{w}^T\mathbf{x} + b = 0$

$\mathbf{w}^T\mathbf{x} + b = -1$

$\mathbf{w}^T\mathbf{x} + b = 1$

$$y_i = \begin{cases} -1, & \text{if } \mathbf{w}^T\mathbf{x}^{(i)} + b \leq -1 \\ 1, & \text{if } \mathbf{w}^T\mathbf{x}^{(i)} + b \geq 1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Linear SVM Mathematically (cont.)

- Then the margin can be expressed through (rescaled) **w** and b as:

$$\text{New margin } \gamma' = \frac{2}{\|\mathbf{w}\|}$$

- Then we can formulate the *quadratic optimization problem:*

Find **w** and b such that

$\gamma' = \frac{2}{\|\mathbf{w}\|}$ is maximized

and for all $(\mathbf{x}^{(i)}, y_i)$, $i = 1 \ldots m$: $y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$

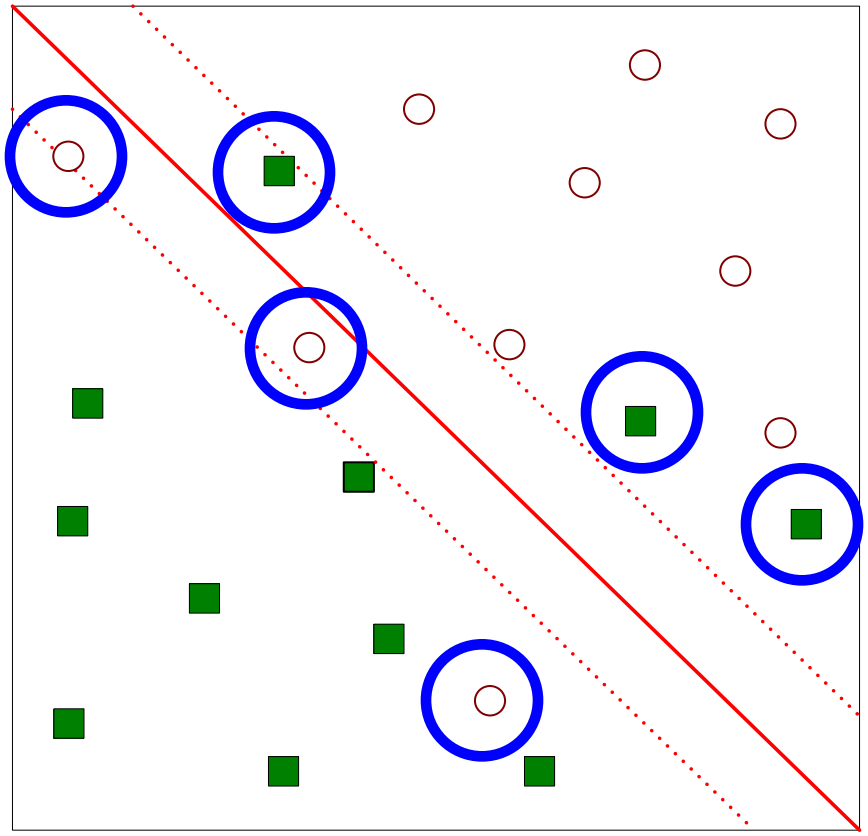
- Which can be reformulated as:

Find **w** and b such that

$\mathbf{\Phi(w)} = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}^{(i)}, y_i)$, $i = 1 \ldots m$: $y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$

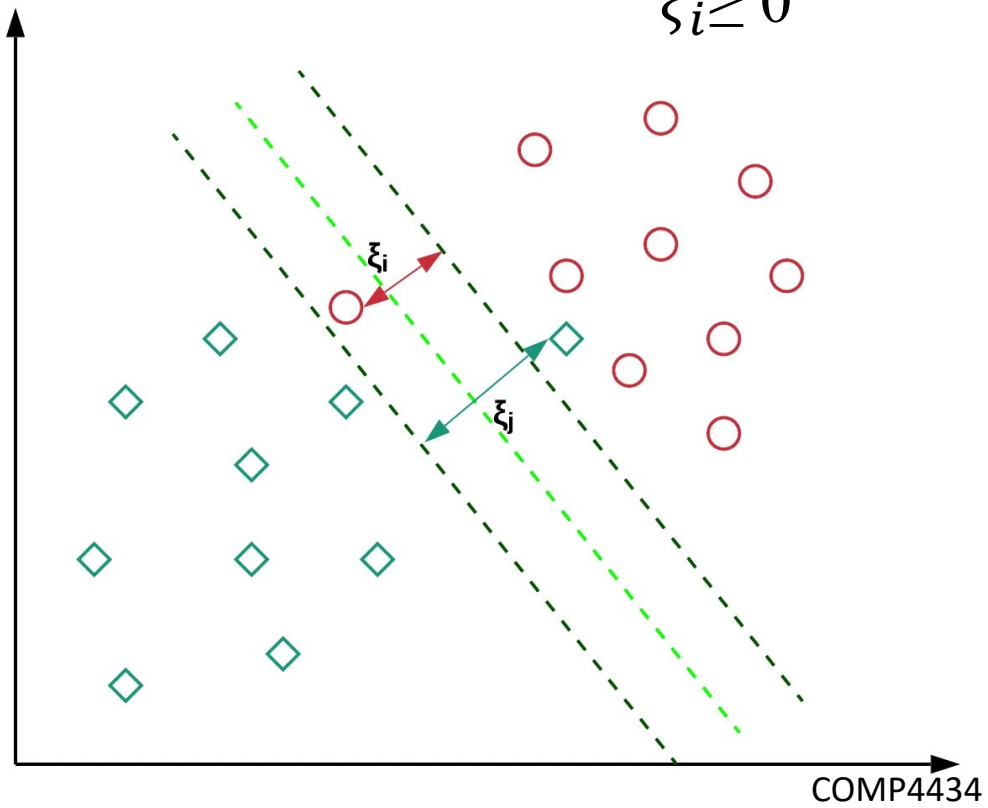
What if the problem is not linearly separable



SVM with soft margin

- Need to minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{i=1}^{m}\xi_i^2\right)$

- subject to:
$$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \leq -1 + \xi_i \quad \text{if } y_i = -1$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \geq 1 - \xi_i \quad \text{if } y_i = 1$$
$$\xi_i \geq 0$$

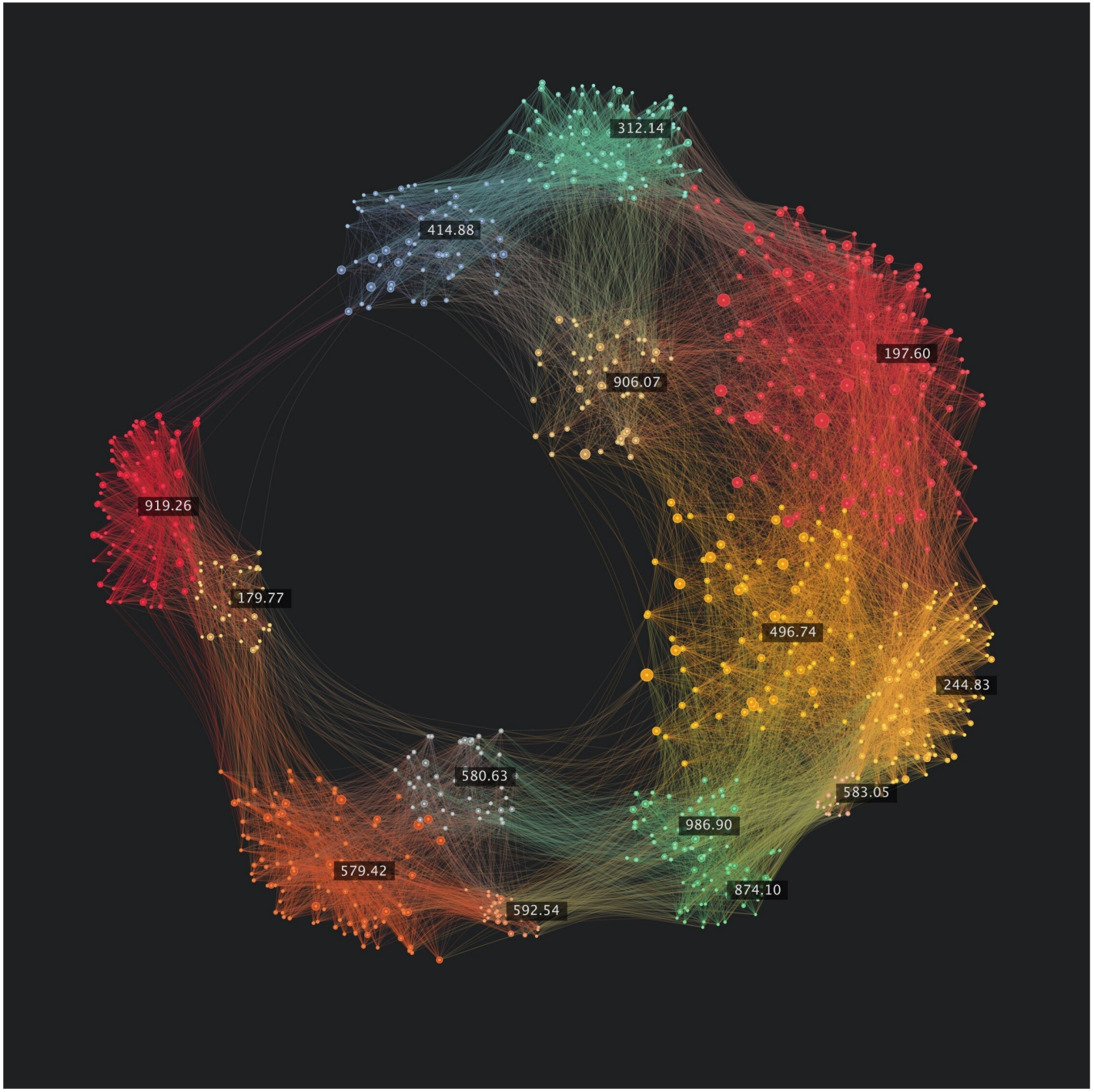


Characteristics of SVM

- The learning problem is formulated as a convex optimization problem
- Efficient algorithms are available to find the global minima
- High computational complexity for building the model
- Robust to noise
- Overfitting is handled by maximizing the margin of the decision boundary
- SVM can handle irrelevant and redundant attributes better than many other techniques
- The user needs to provide the type of kernel function & cost function (for nonlinear SVM)
- Difficult to handle missing values

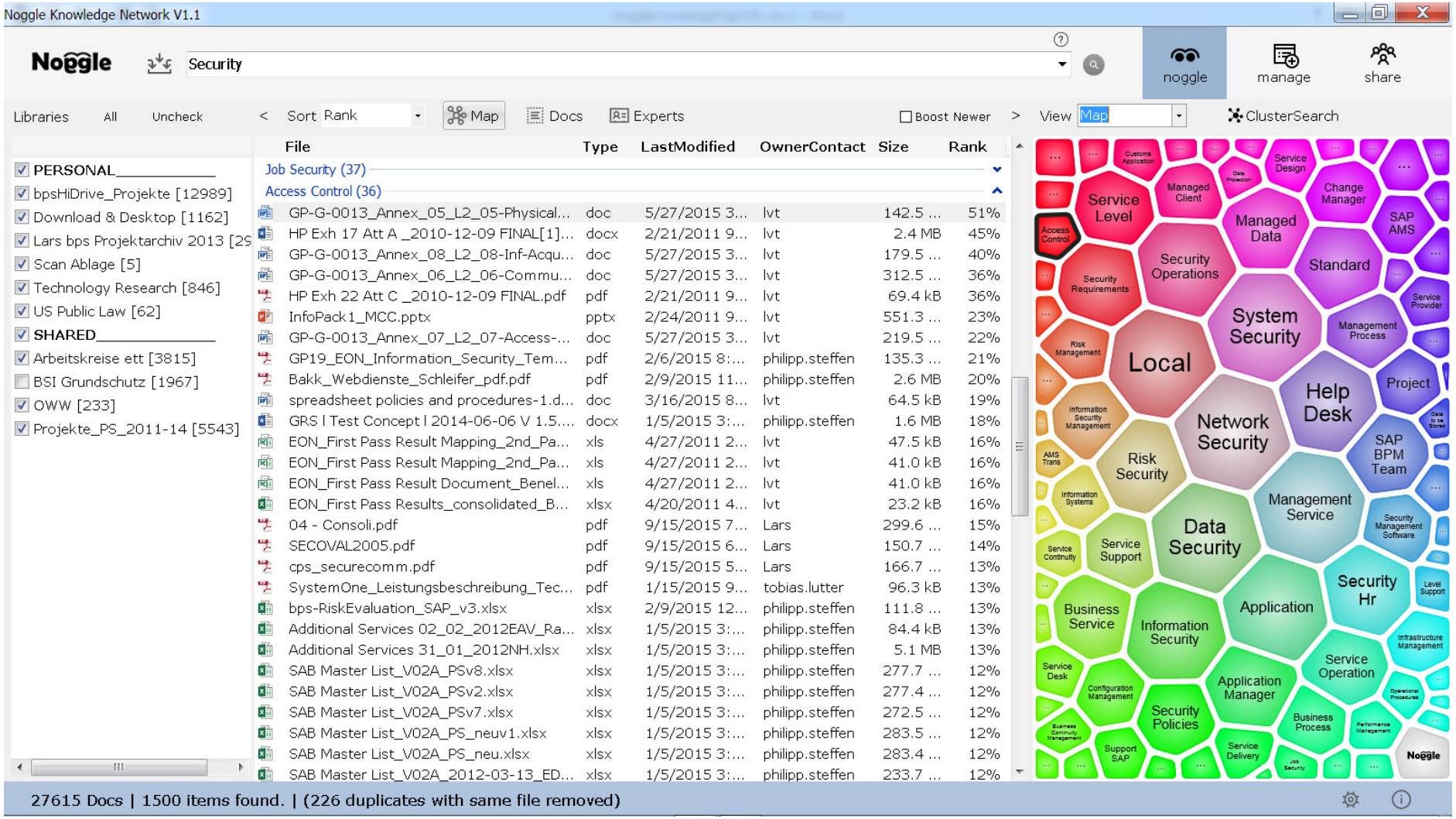
What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms



Given a cloud of data points we want to understand its structure.

Document Clustering



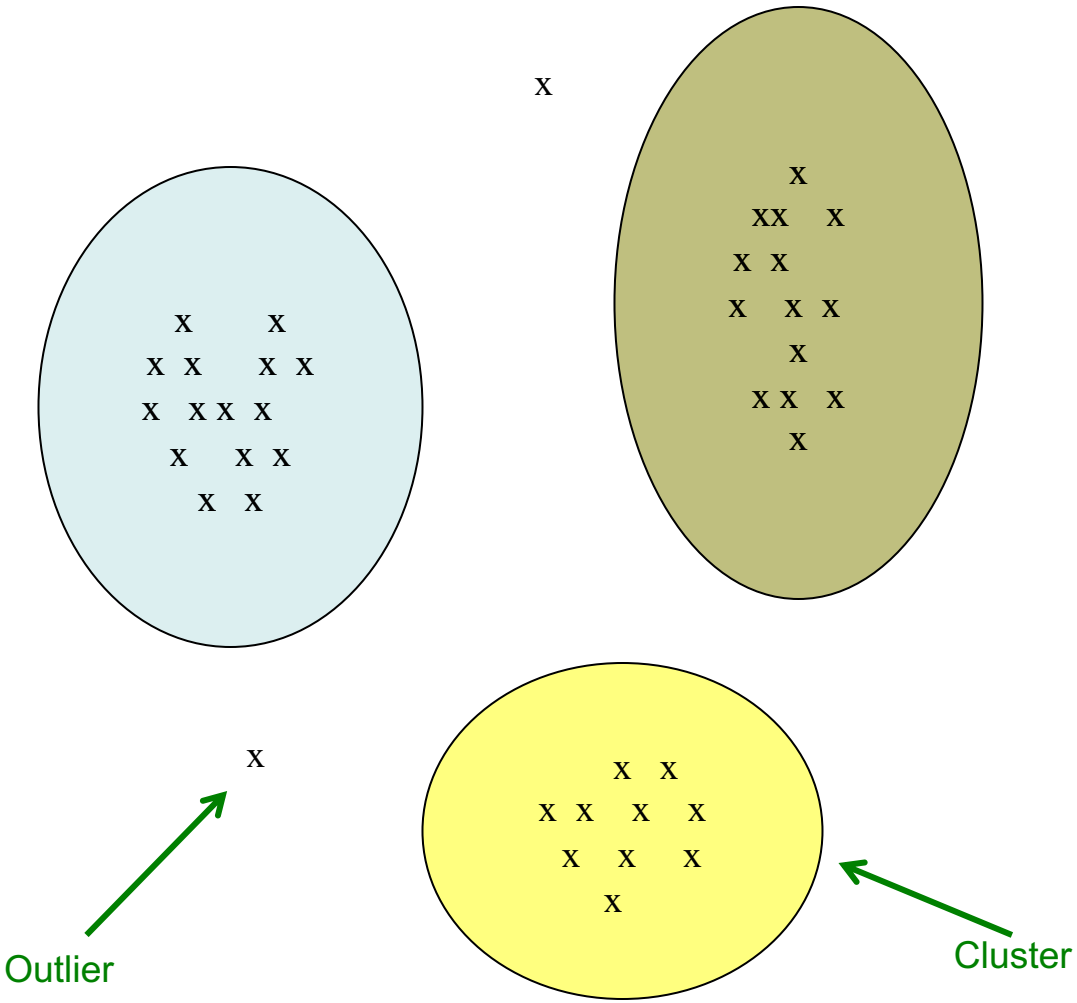
Clustering for Data Understanding & Applications

- **Customer Segmentation**: Businesses use clustering to group customers with similar purchasing behavior. This helps in targeted marketing, personalized recommendations, and product/service customization
- **Image Segmentation**: In computer vision, clustering is used to segment images into regions with similar features. This is useful in object detection, image recognition
- **Anomaly Detection**: Clustering can help identify outliers or anomalies in datasets. This is crucial in fraud detection, network security, and quality control
- **Social Network Analysis**: Clustering can group users with similar connections or behavior in social networks. This is used for community detection and influence analysis

Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, classification
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors:
  - Localizing search to one or a small number of clusters
- Outlier detection:
  - Outliers are often viewed as those “far away” from any cluster

Example: Clusters & Outliers



Problem definition of clustering

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- **Usually:**
  - Points are in a high-dimensional space
  - Similarity is defined using a distance measure
    - Euclidean, Cosine, Jaccard, edit distance, …

Clustering Problem: Galaxies

- A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands)
- **Problem:** Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Digital Sky Survey

Clustering is a hard problem!

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy

- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different:** Almost all pairs of points are at about the same distance

Clustering Problem: Music

- **Intuitively: Music divides into categories, and customers prefer a few categories**
  - But what are categories really?

- Represent a song by a set of customers who like it
- Similar songs have similar sets of customers, and vice-versa

Clustering Problem: Music

Space of all songs:

- Think of a space with one dimension for each customer
  - Values in a dimension may be 0 or 1 only
  - A song is a point in this space (x_1, x_2,…, x_d), where x_i = 1 iff the i^{th} customer bought the CD
- For Spotify:
  - Spotify lets you discover, organize, and share over 100 million songs, over 5 million podcast titles and 350,000+ audiobooks
  - In 2023, Spotify has 551 million users and 220 million premium subscribers across 184 regions
- **Task:** Find clusters of similar songs

Clustering Problem: Documents

Finding topics:

- Represent a document by a vector $(x_1, x_2, \ldots, x_d)$, where $x_i = 1$ iff the i^{th} word (in some order) appears in the document
  - It actually doesn't matter if d is infinite; i.e., we don't limit the set of words

- **Documents with similar sets of words may be about the same topic**

Similarity is defined using a distance measure

- **Sets as vectors:**
  - Measure similarity by the **cosine distance**

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

cosine distance = 1 - cosine similarity

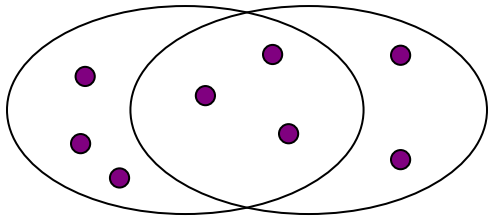
  - Measure similarity by **Euclidean distance**

$$d(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{i=1}^{n}(B_i - A_i)^2}$$

- **Sets as sets:**
  - Measure similarity by the **Jaccard distance**

Jaccard similarity

- The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
  $sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
- **Jaccard distance:** $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$



3 in intersection
8 in union
Jaccard similarity= 3/8
Jaccard distance = 5/8

- Document D_1 is a set of its b words
- Equivalently, each document is a 0/1 vector in the space of *k words*
  - Each unique word is a dimension
  - Vectors are very sparse

k-means Clustering Algorithm

- Partitioning method: Partitioning ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or clustroid of cluster C_i)

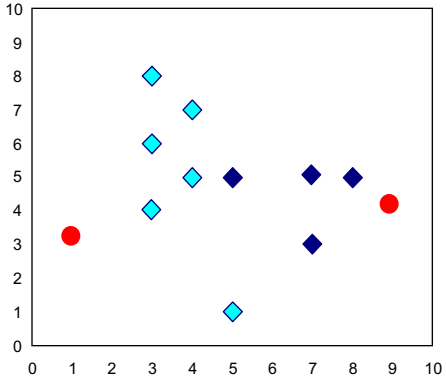
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

k-means Clustering Algorithm

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

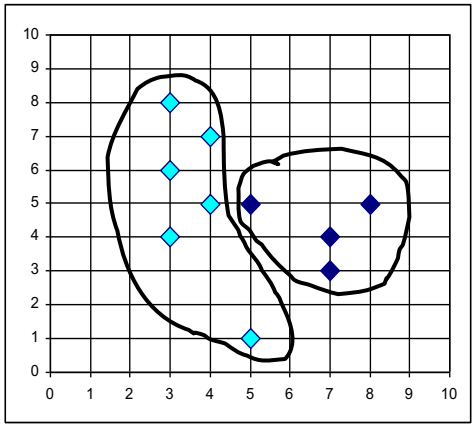
Demo



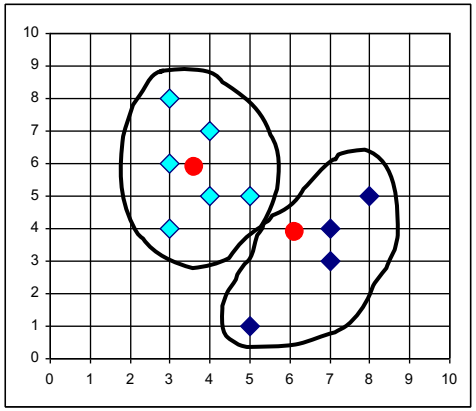
k=2

Arbitrarily choose k means

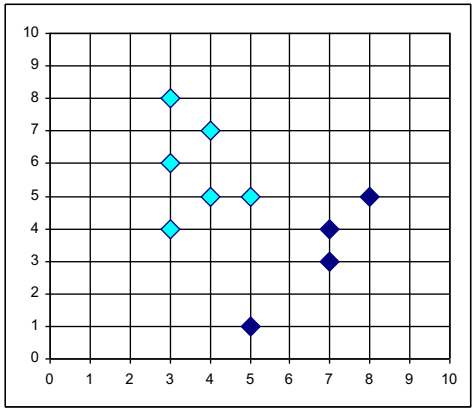
Assign each objects to most similar center



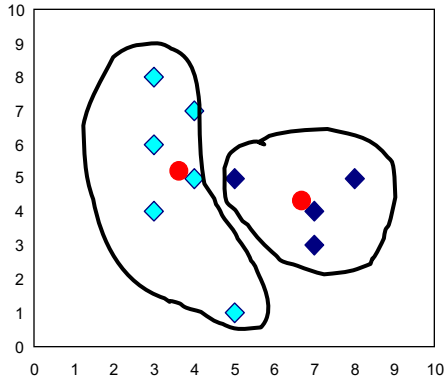
reassign



Update the cluster means



reassign

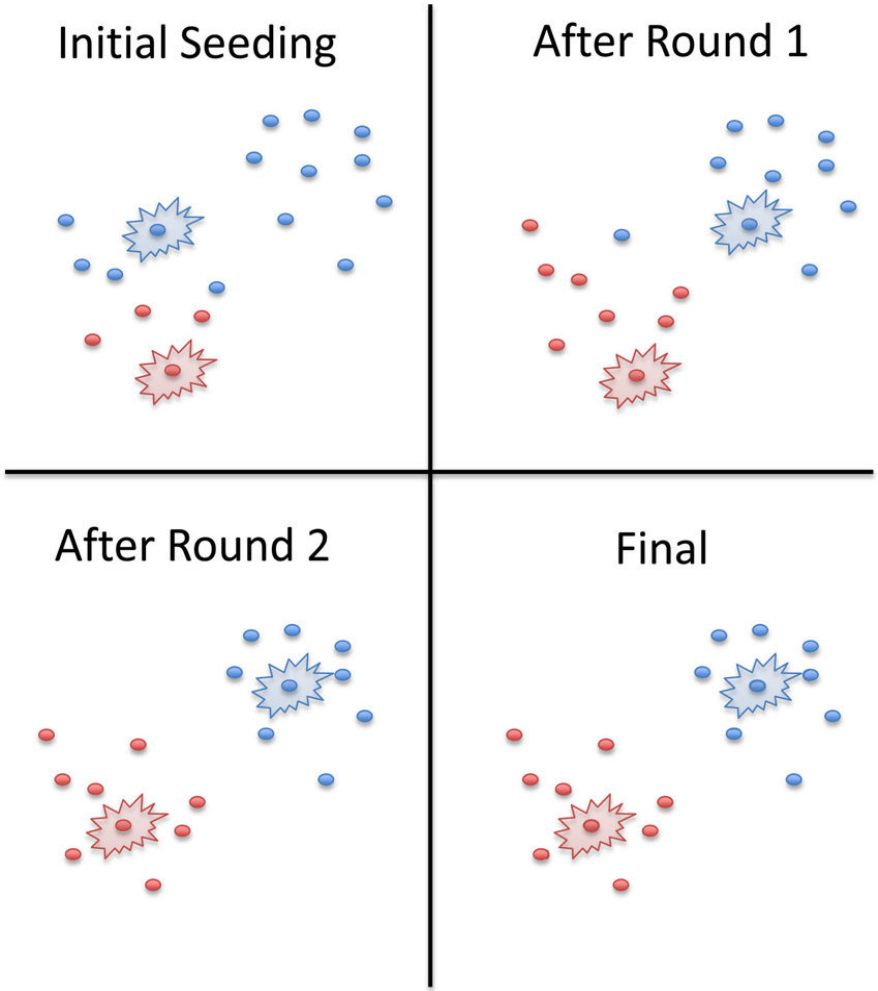


Update the cluster means

Populating Clusters

- **1)** For each point, place it in the cluster whose current centroid it is nearest
- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters
- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

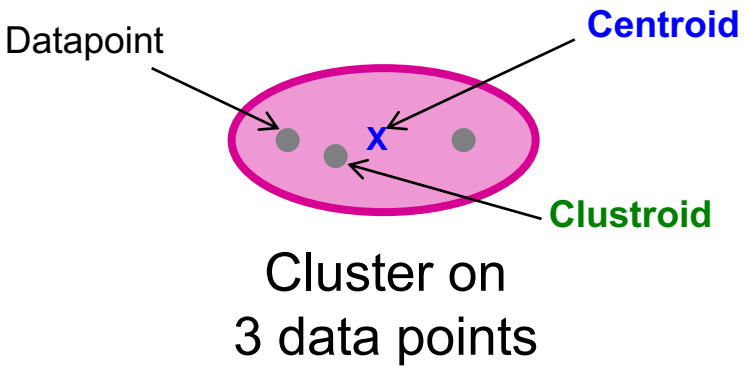
Initialization of k-means



- The way to initialize the centroids was not specified. One popular way to start is to randomly choose k of the examples
- The results produced depend on the initial values for the centroids, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points

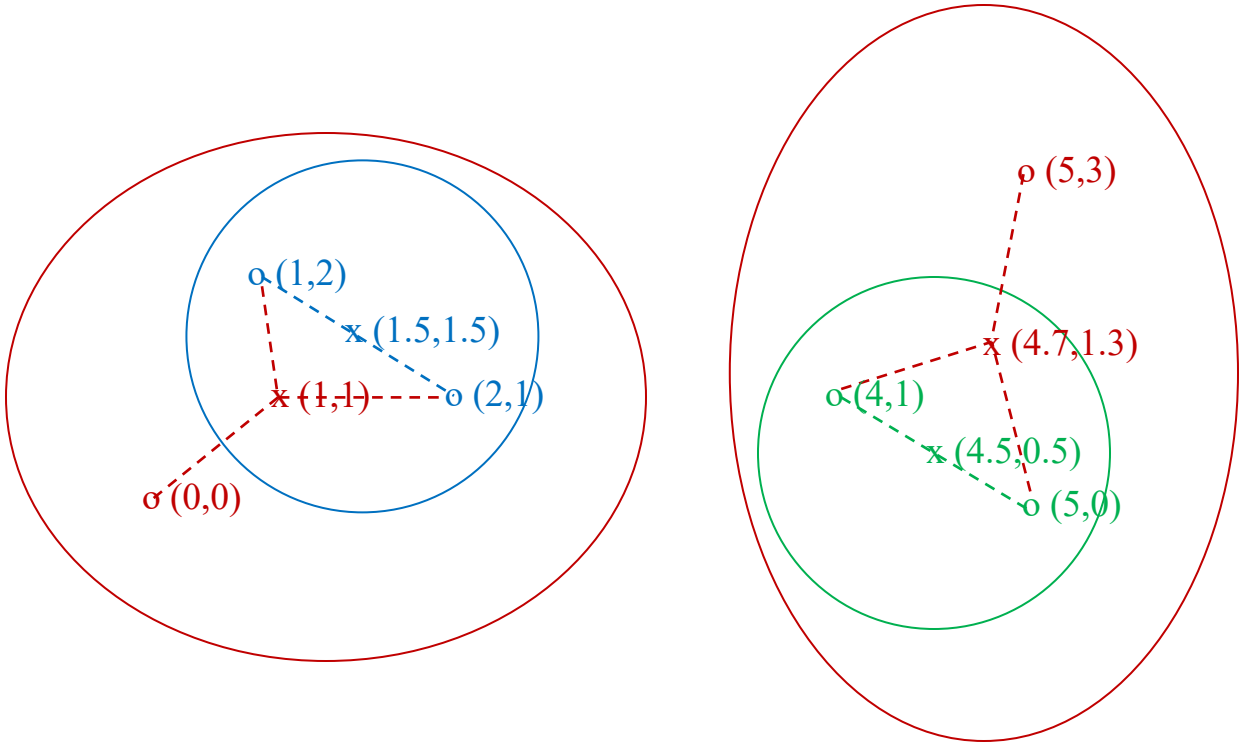
Centroid & Clustroid

- **Centroid** is the avg. of all (data)points in the cluster. This means centroid is an “artificial” point
- **Clustroid** is an existing (data)point that is “closest” to all other points in the cluster



Clustroid

- **Euclidean case:** each cluster has a centroid
  - ***centroid*** = average of its (data) points
  - use the node that is "closest" to the centroid as a clustroid



Data:
o … data point
x … centroid

- **What about the non-Euclidean case?**

Clustroid (non-Euclidean Case)

- Non-Euclidean: The only "locations" we can talk about are the points themselves, i.e., there is no "average" of two points
- ***clustroid*** = point "***closest***" to other points
- **Possible meanings of "closest":**
  - Smallest average distance to other points
  - Smallest sum of squares of distances to other points, e.g., for distance metric ***d*** clustroid ***c*** of cluster ***C*** is:

$$\min_{c} \sum_{x \in C} d(x,c)^2$$

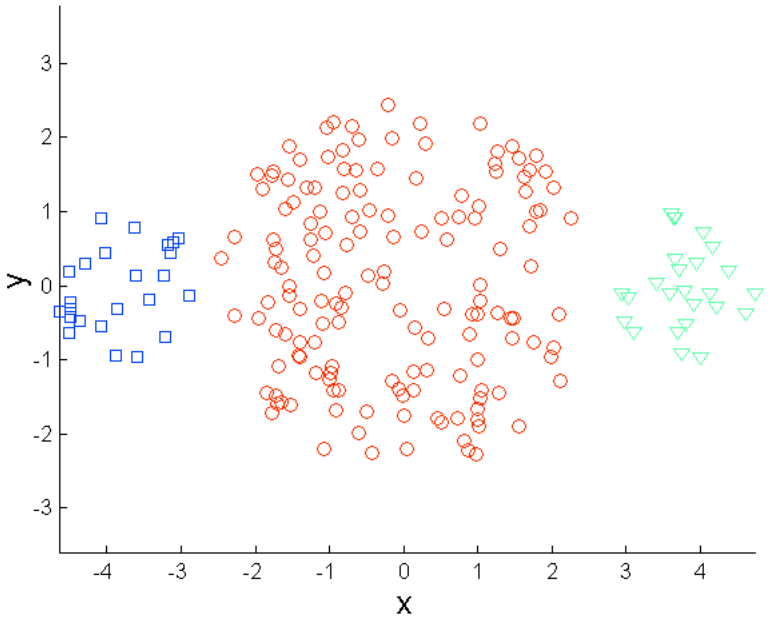
  - Smallest maximum distance to other points

Pros & Cons

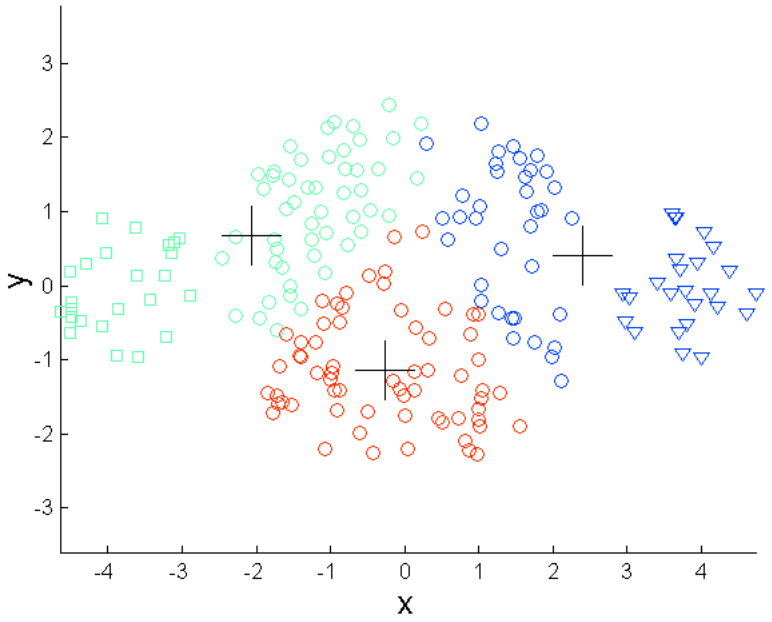
- Simple iterative method
- User provides “K”

- Often too simple ----> bad results
- Difficult to guess the correct “K”
  - We may not know the number of clusters before we want to find clusters
- No guarantee of optimal solution
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

Limitations: when clusters are of differing sizes

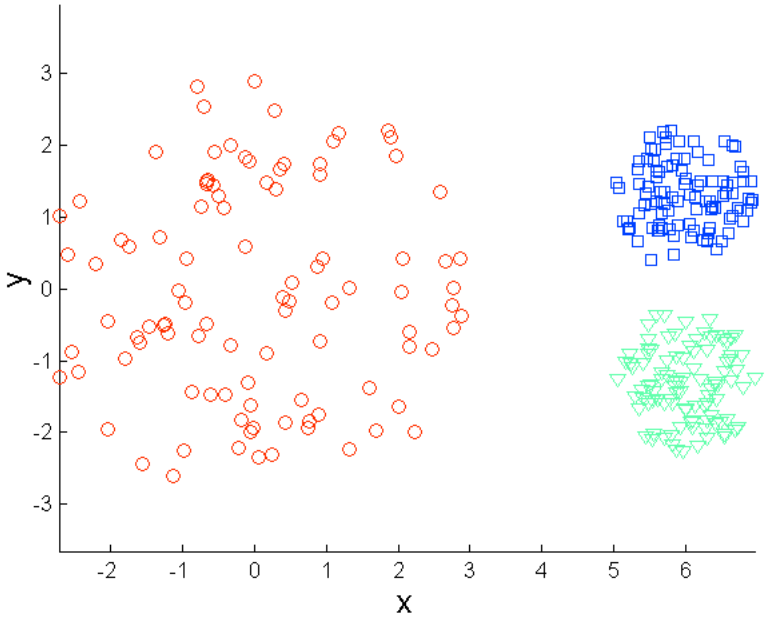


Original Points

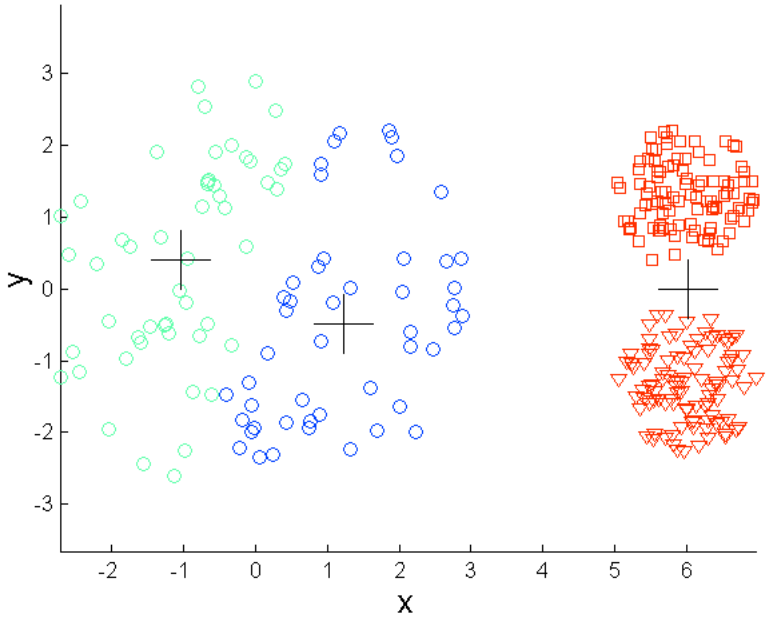


K-means (3 Clusters)

Limitations: when clusters are of differing densities

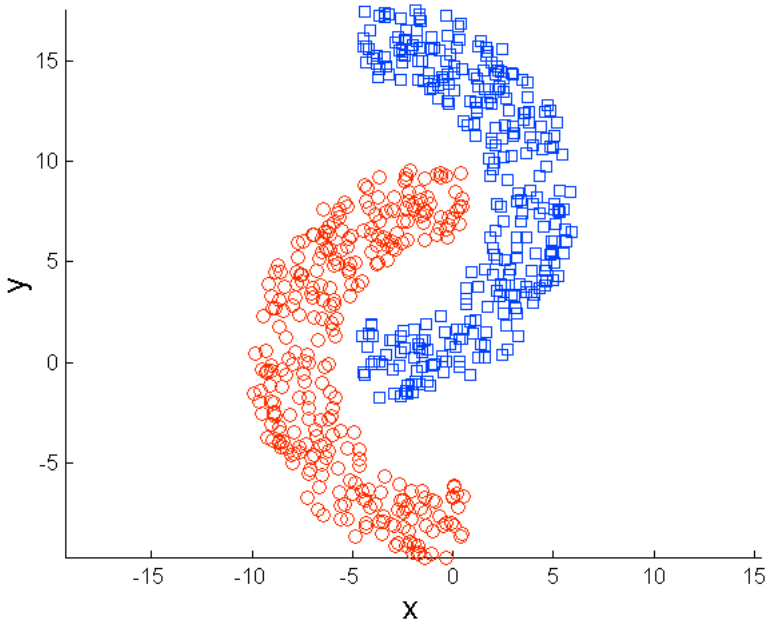


Original Points

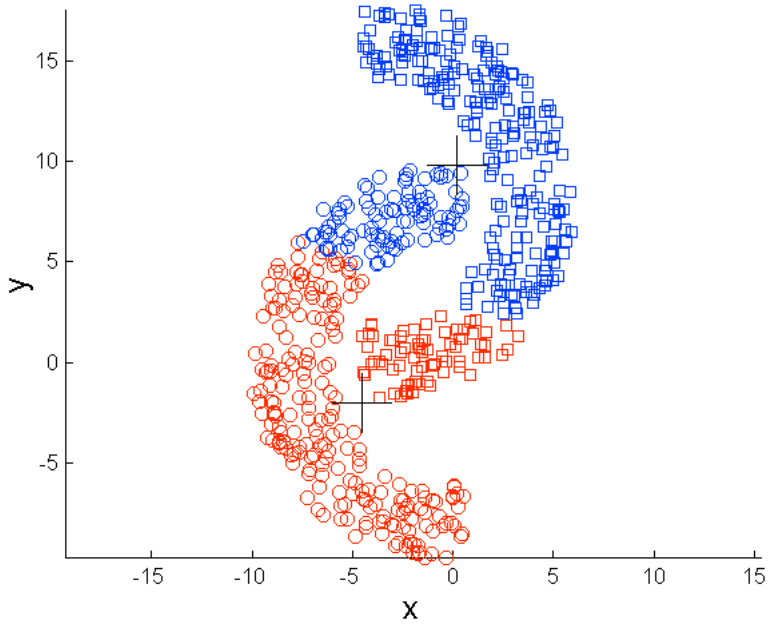


K-means (3 Clusters)

Limitations: when non-globular shapes



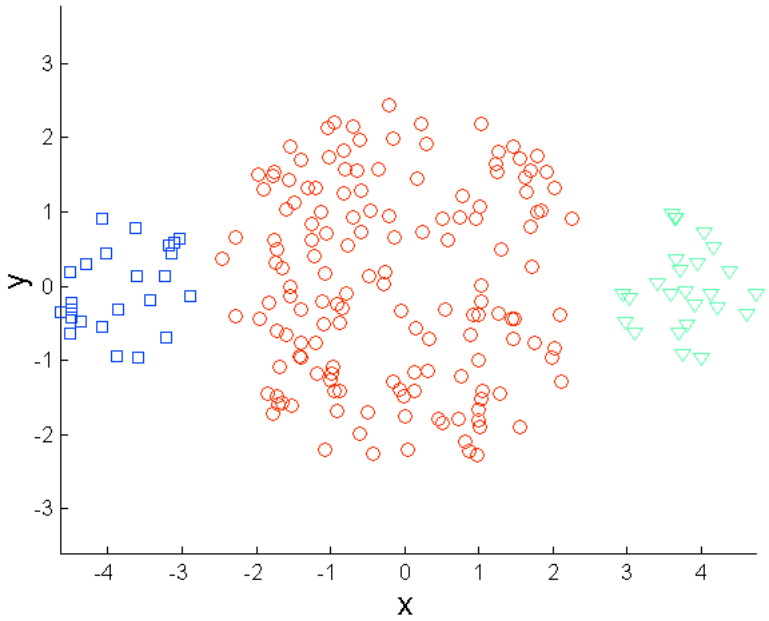
Original Points



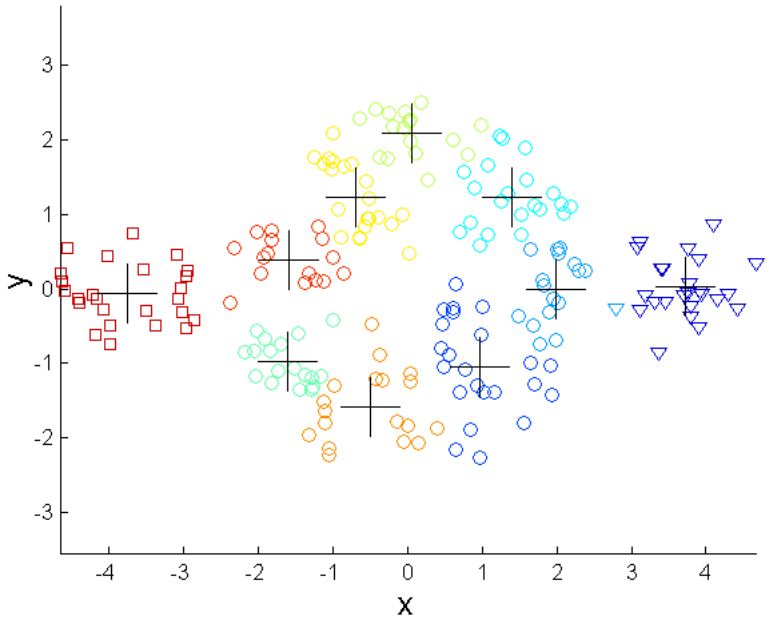
K-means (2 Clusters)

Overcoming K-means Limitations

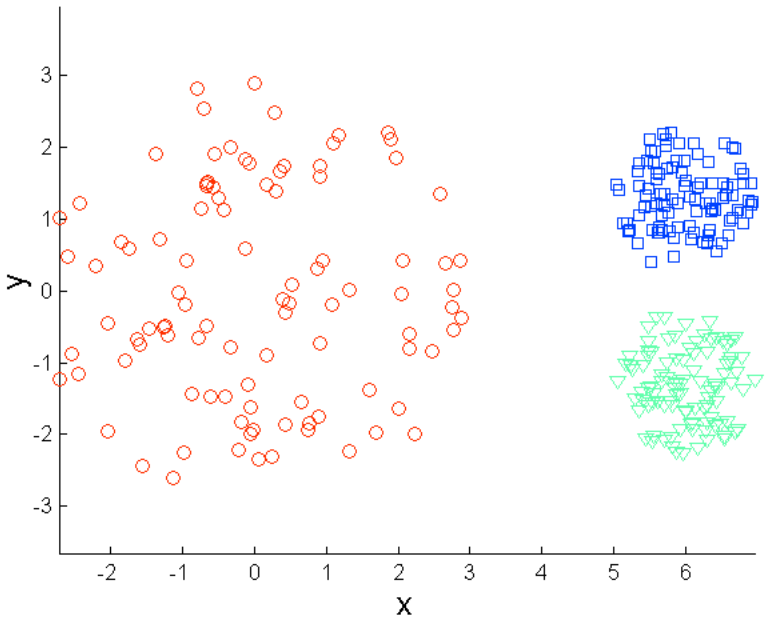
- One solution is to find many clusters
  - each of them represents a part of a natural cluster
  - small clusters need to be put together in a post-processing step

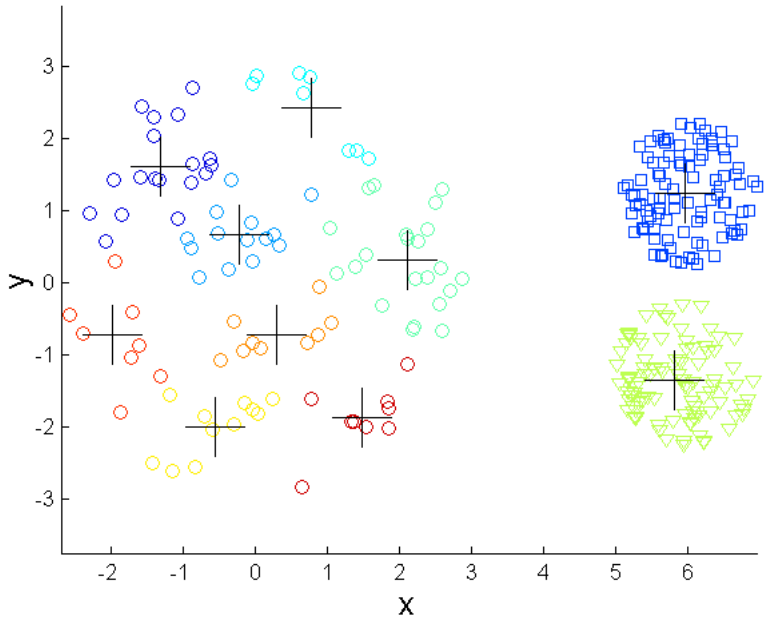


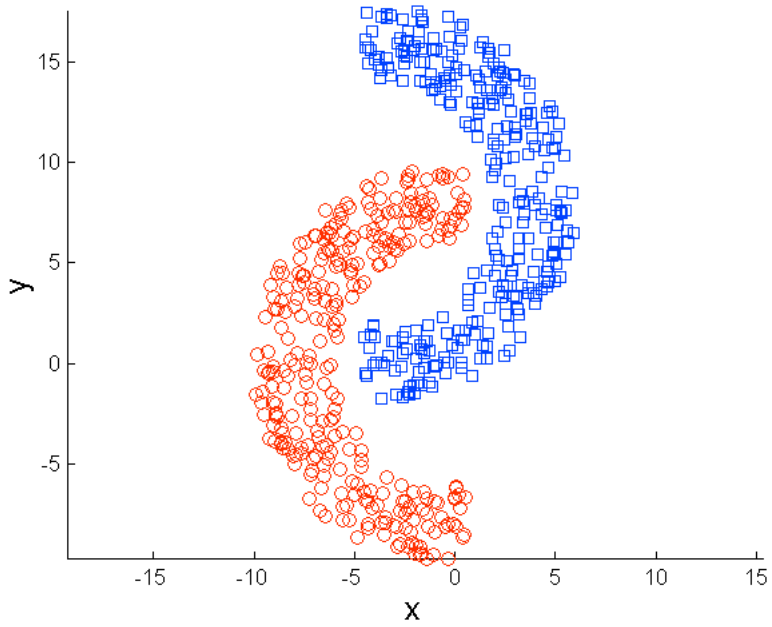
Original Points

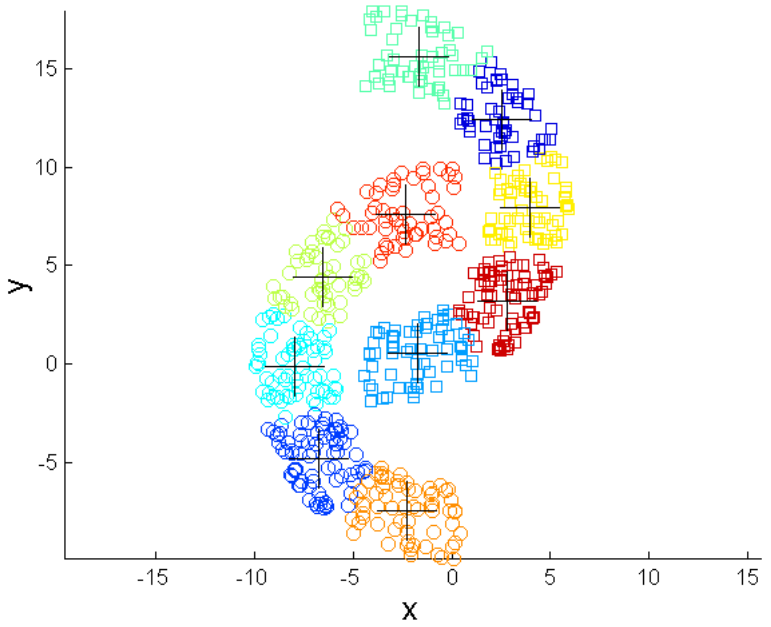


K-means Clusters









Original Points

K-means Clusters