# k-Nearest Neighbors

Adopted from Andrew Moore's IBL Tutorial & Tan et Al., 'Introduction to Data Mining'

# Instance-Based Classifiers

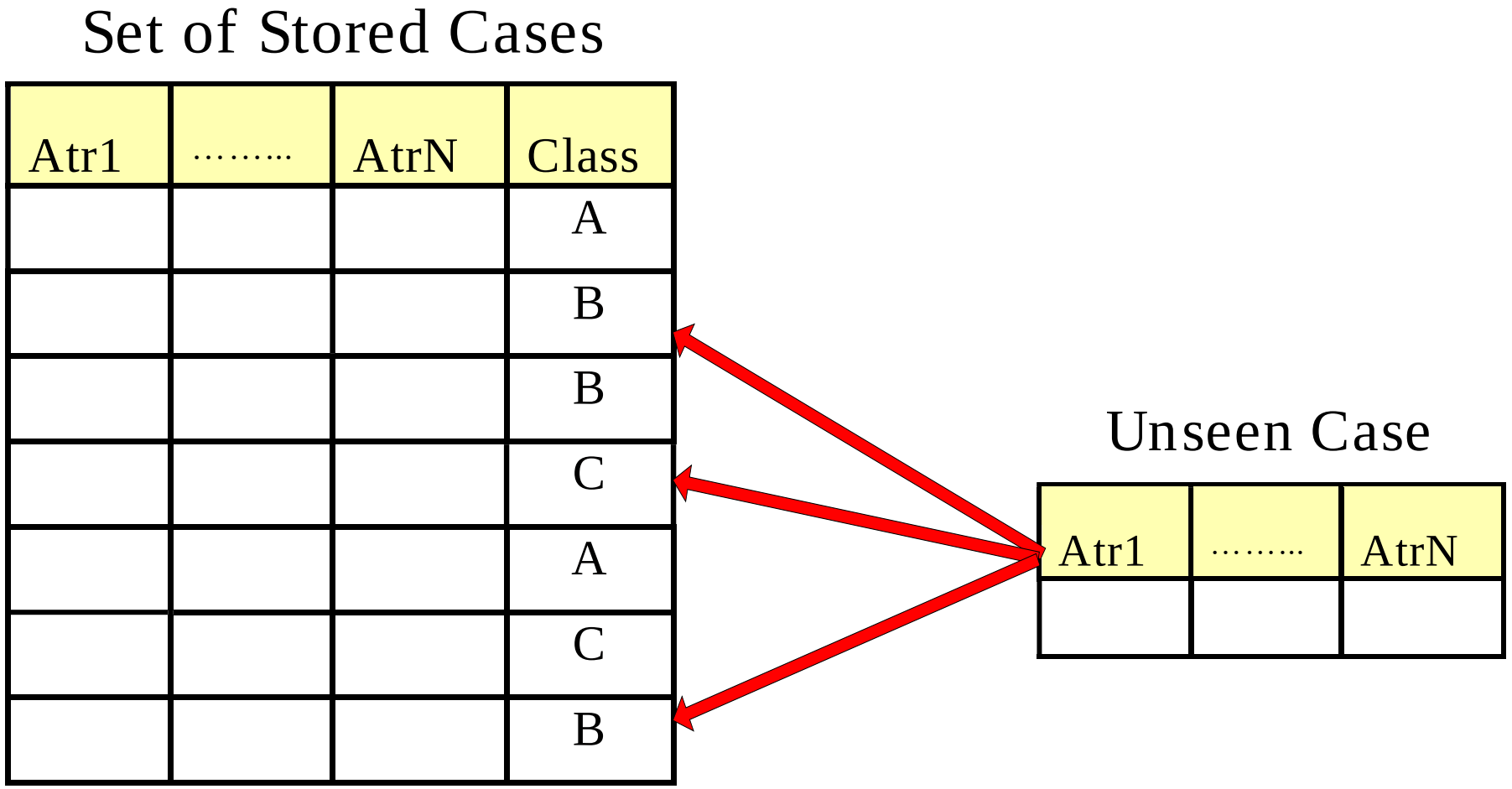
- Build a database of previous observations
- To make a prediction for a new item *x'*, find the most similar database item *x* and use its output *f(x)* for *f(x')*
- Provides a local approximation to target function or concept

You need:

1. A distance metric (to determine similarity)
2. Number of neighbors to consult
3. Method for combining neighbors' outputs

# Instance-Based Classifiers

- Store the training records
- Use training records to predict the class label of unseen cases



# Examples

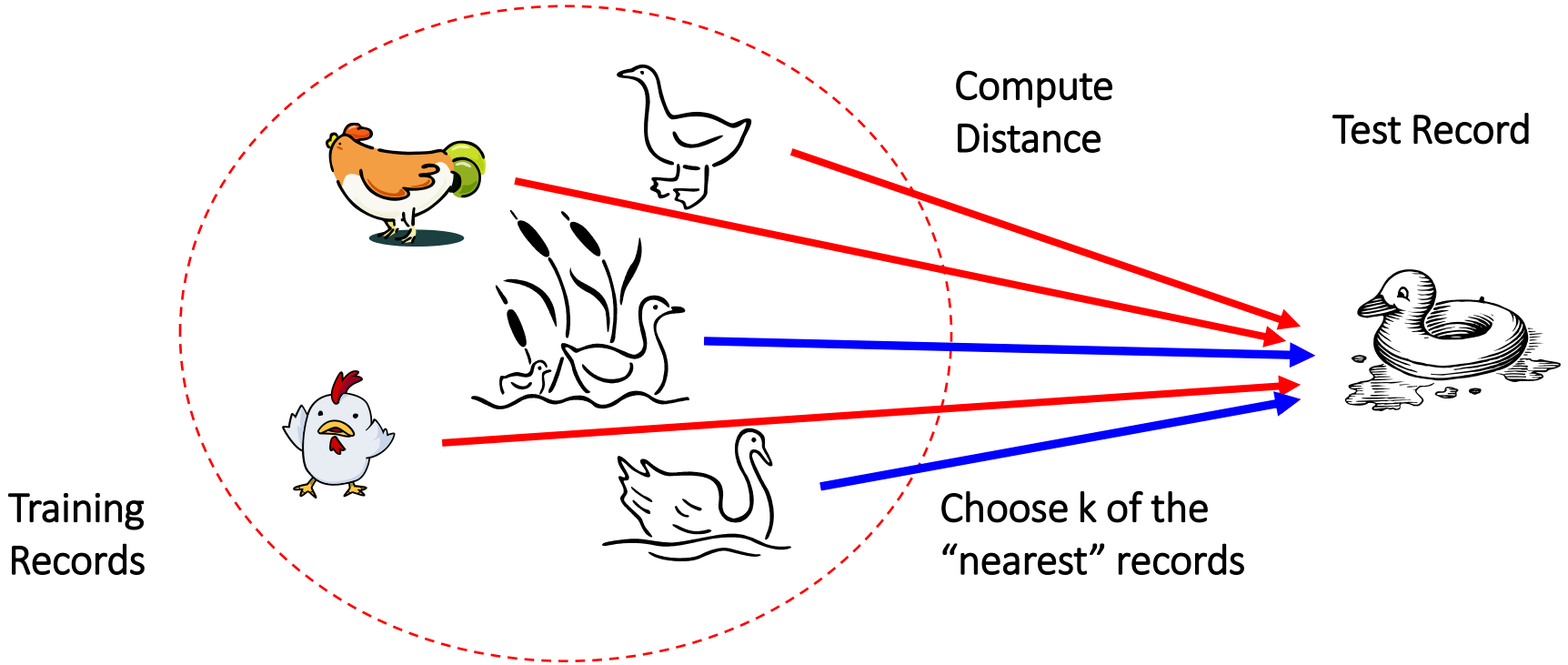
- Rote-learner
  - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

- Nearest neighbor
  - Uses k “closest” points (nearest neighbors) for performing classification

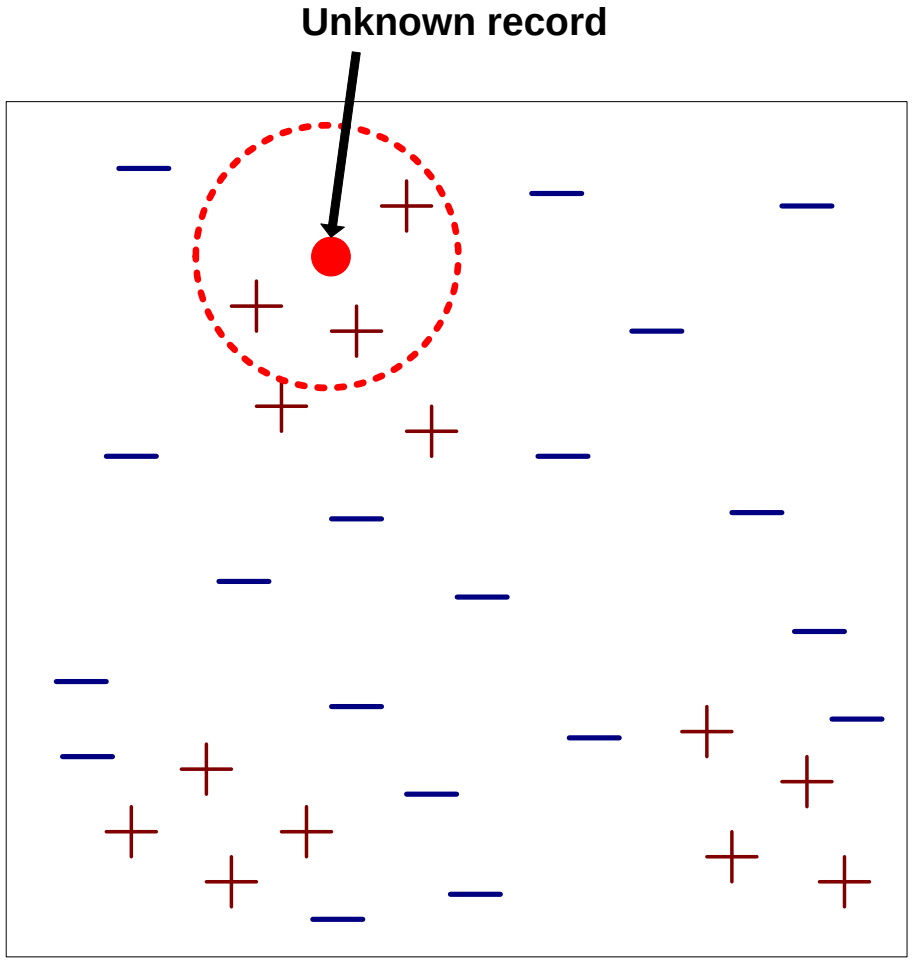
# Basic Idea

Basic idea:

If it walks like a duck, quacks like a duck, then it's probably a duck



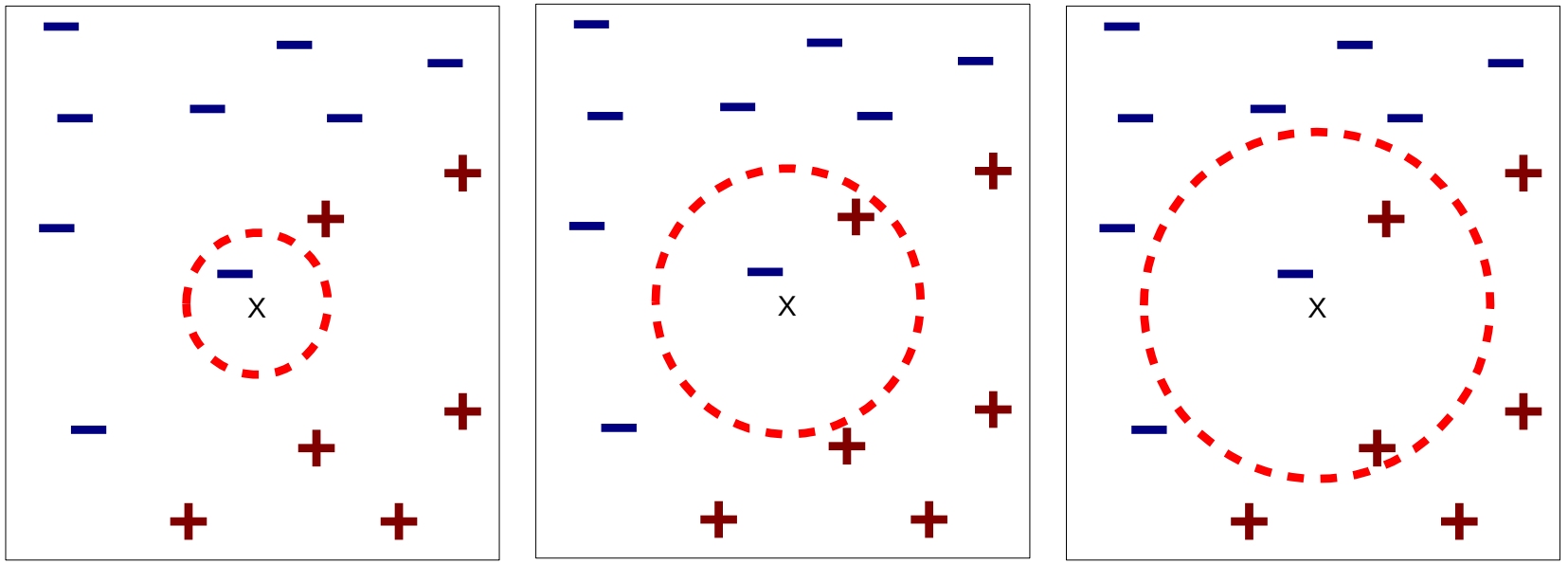
# Classifiers




- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking **majority vote**)

# Definition

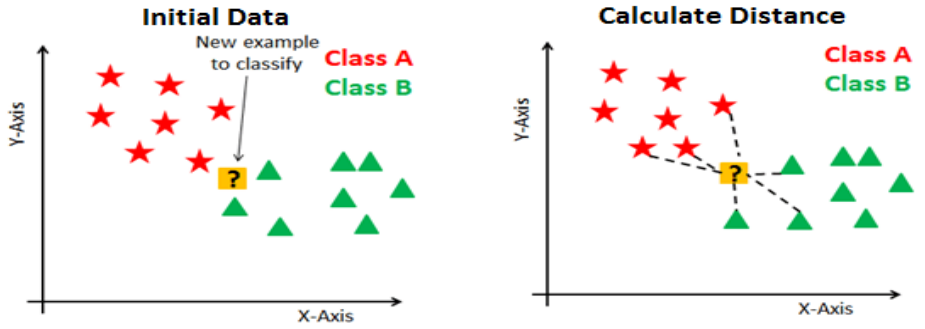



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

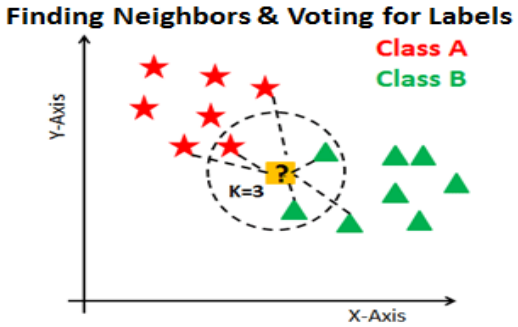
# Classification

- Compute distance between two points:
  - Euclidean distance
- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$









# Example

| Outlook | Temperature | Humidity | Windy | Play |
|---|---|---|---|---|
| sunny | hot | high | FALSE | no |
| sunny | hot | high | TRUE | no |
| overcast | hot | high | FALSE | yes |
| rainy | mild | high | FALSE | yes |
| rainy | cool | normal | FALSE | yes |
| rainy | cool | normal | TRUE | no |
| overcast | cool | normal | TRUE | ? |
| sunny | mild | high | FALSE | ? |
| sunny | cool | normal | FALSE | ? |
| rainy | mild | normal | FALSE | ? |
| sunny | mild | normal | TRUE | ? |
| overcast | mild | high | TRUE | ? |
| overcast | hot | normal | FALSE | ? |
| rainy | mild | high | TRUE | ? |

testing

# 1-Nearest Neighbor

1. A distance metric: Euclidean

$$D(x, x') = \sqrt{\sum_{i=1}^{f} (x_i - x'_i)^2}$$

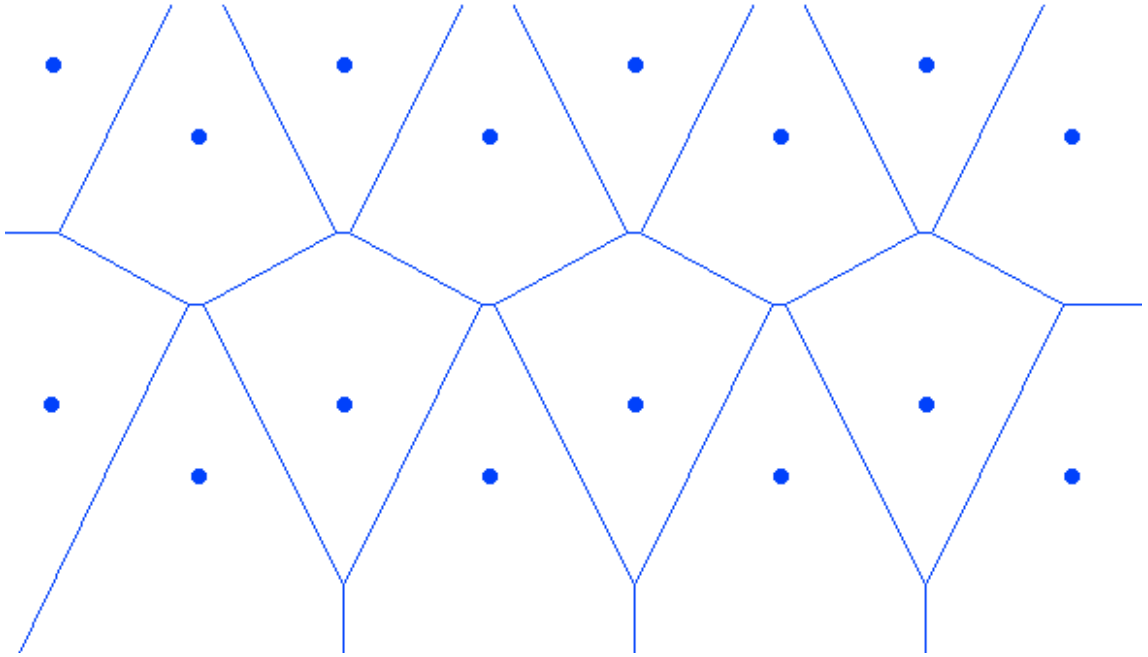
2. Number of neighbors to consult: 1

3. Combining neighbors' outputs: N/A

Equivalent to memorizing everything you've ever seen and reporting the most similar result

# 1-Nearest Neighbor

We can draw the 1-nearest-neighbor region for each item: a Voronoi diagram



# Algorithm

- Given training data $(x_1, y_1) \ldots (x_n, y_n)$, determine $y_{new}$ for $x_{new}$
- Find $x'$ most similar to $x_{new}$ using Euclidean distance
- Assign $y_{new} = y'$
- Works for classification or regression

# 1-Nearest Neighbor Drawbacks

- 1-NN fits the data exactly, including any noise
- May not generalize well to new data

# k-Nearest Neighbors

- Distance metric: Euclidean
- Number of neighbors to consult: k
- Combining neighbors' outputs:
  - Classification
    - Majority vote
    - Weighted majority vote: nearer have more influence
  - Regression
    - Average (real-valued)
    - Weighted average: nearer have more influence
- Result: *Smoother*, more generalizable result

$$y_{new} = \text{argmax}_c \sum (y_k = c)$$

$$y_{new} = \text{argmax}_c \sum_k w_k (y_k = c)$$

$$y_{new} = \sum_k \frac{1}{k} y_k$$

$$y_{new} = \sum_k w_k y_k$$

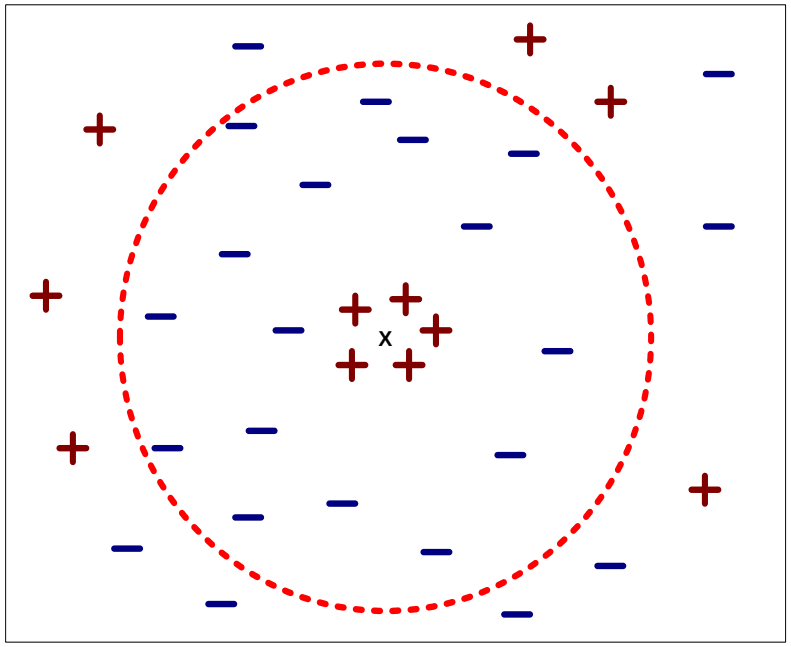
# k-Nearest Neighbors

- k is a parameter of the k-NN algorithm
  - This does not make it “parametric”.  Confusing!

- Recall: set parameters using validation data set
  - Not the training set (overfitting)

# k-Nearest Neighbors

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



# k-Nearest Neighbors

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes

  - Example of an attribute dominating distance computation:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

# k-Nearest Neighbors

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive
    - For each test, need to scan all training data

# k-Nearest Neighbors

- How expensive is it to perform k-NN on a new instance?
  - O(n) to find the nearest neighbor
  - The more you know, the longer it takes to make a decision!
  - Can be reduced to O(log n) using kd-trees

# k-Nearest Neighbors

- k neighbors in training data to the input data x: break ties arbitrarily
- All k neighbors will vote: majority wins
- Extension:
  - Weighted k-NN

- "k" is a variable:
  - Often we experiment with different values of k=1, 3, 5, to find out the optimal one
- Why is KNN important?
  - Often a baseline
  - Must beat this one to claim innovation
- Applications of k-NN
  - Document similarity

# Pros & Cons

- Pros
  - k-NN is simple! (to understand, implement)
  - Often used as a baseline for other algorithms
  - “Training” is fast: just add new item to database

- Cons
  - Most work done at query time: may be expensive
  - Must store O(n) data for later queries
  - Performance is sensitive to choice of distance metric
    - And normalization of feature values

# Examples

https://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html

| | | | [3,3] | [3,7] | [3,3] | [3,7] |
|---|---|---|---|---|---|---|
| x1 | x2 | Class | Distance | Distance | Distance | Distance |
| 2 | 3 | Yes | 1,00 | 4,12 | 1,00 | 4,12 |
| 1 | 8 | No | 5,39 | 2,24 | 5,39 | 2,24 |
| 6 | 5 | No | 3,61 | 3,61 | 3,61 | 3,61 |
| 3 | 8 | Yes | 5,00 | 1,00 | 5,00 | 1,00 |
| 1 | 1 | No | 2,83 | 6,32 | 2,83 | 6,32 |
| 2 | 9 | Yes | 6,08 | 2,24 | 6,08 | 2,24 |
| 1 | 0 | Yes | 3,61 | 7,28 | 3,61 | 7,28 |
| 2 | 2 | Yes | 1,41 | 5,10 | 1,41 | 5,10 |
| 2 | 1 | No | 2,24 | 6,08 | 2,24 | 6,08 |
| 2 | 4 | No | 1,41 | 3,16 | 1,41 | 3,16 |
| | | | | | | |
| | Classify | | k=3 | | k=5 | |
| | 3 | 3 | Yes | | No | |
| | 3 | 7 | Yes | | No | |

# Support Vector Machines

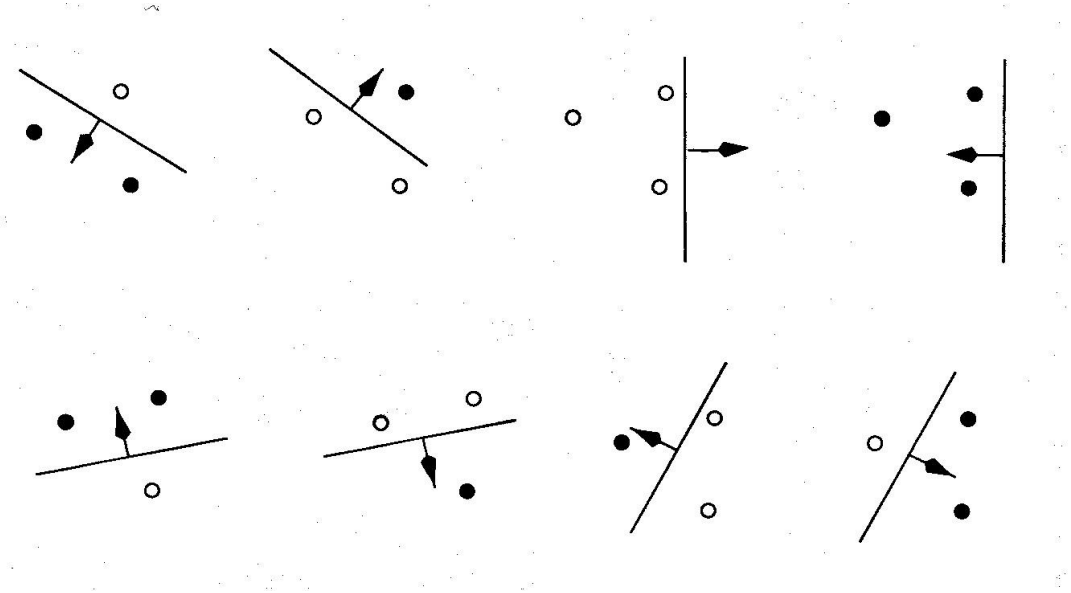
Adopted from Gomes, C. & Hinton, G. Slides

# Introduction

- Suppose that we pick n datapoints and assign labels of + or – to them at random
- If our model class (e.g. a neural net with a certain number of hidden units) is powerful enough to learn any association of labels with the data, its too powerful!
- Maybe we can characterize the power of a model class by asking how many datapoints it can "shatter" i.e. learn perfectly for all possible assignments of labels
  - This number of datapoints is called the Vapnik-Chervonenkis dimension
  - The model does not need to shatter all sets of datapoints of size h. One set is sufficient.
    - For planes in 3-D, h=4 even though 4 co-planar points cannot be shattered.

# Example

- Suppose our model class is a hyperplane
- In 2-D, we can find a plane (i.e. a line) to deal with any labeling of three points
- A 2-D hyperplane shatters 3 points



- But we cannot deal with some of the possible labelings of four points

# Support Vector Machines

- Supervised learning methods for classification and regression
  - relatively new class of successful learning methods

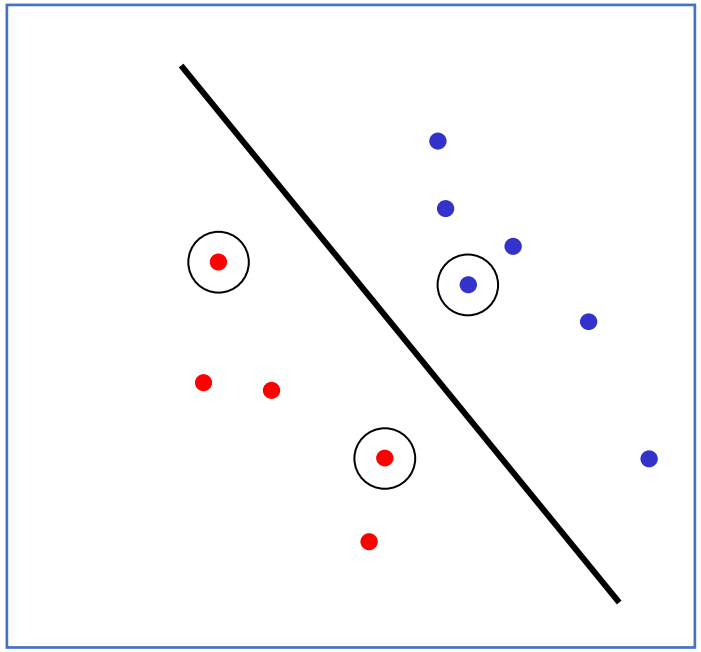
- They can represent non-linear functions and they have an efficient training algorithm

- SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition
  - 1.1% error rate which was comparable to a very carefully constructed (and complex) Artificial Neural Network

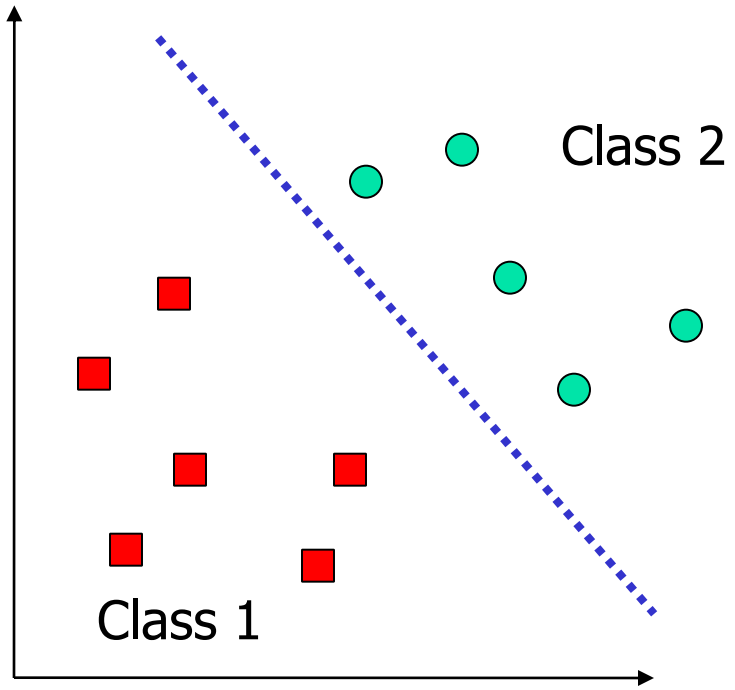
# Support Vector Machines

- The line that maximizes the minimum margin is a good bet
- This maximum-margin separator is determined by a subset of the datapoints.
  - Datapoints in this subset are called “support vectors”
  - It will be useful computationally if only a small fraction of the datapoints are support vectors, because we use the support vectors to decide which side of the separator a test case is on

The support vectors are indicated by the circles around them.



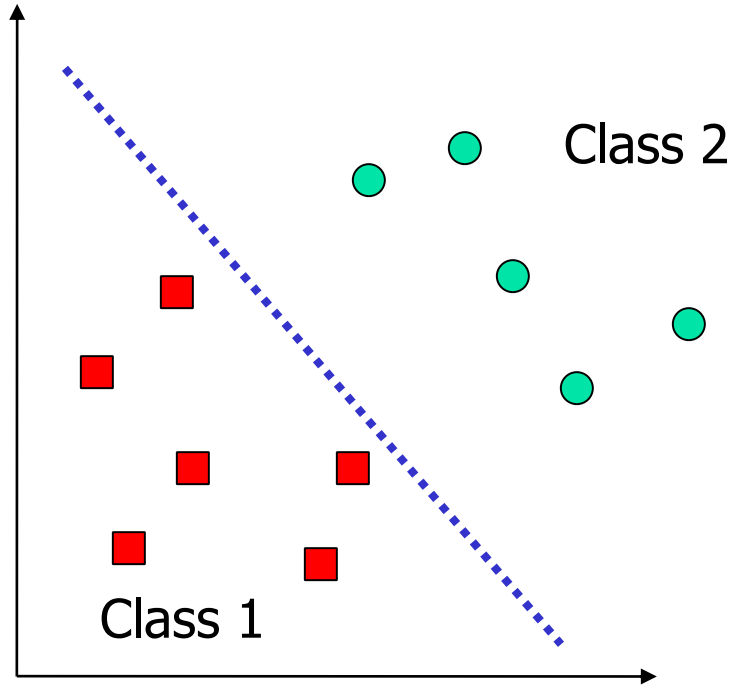
# Two Class Problem

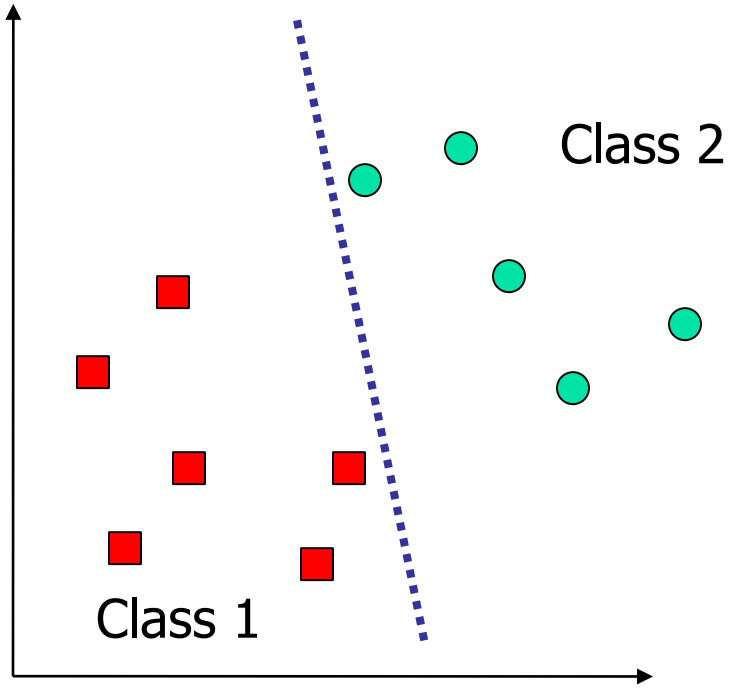



Many decision boundaries can separate these two classes

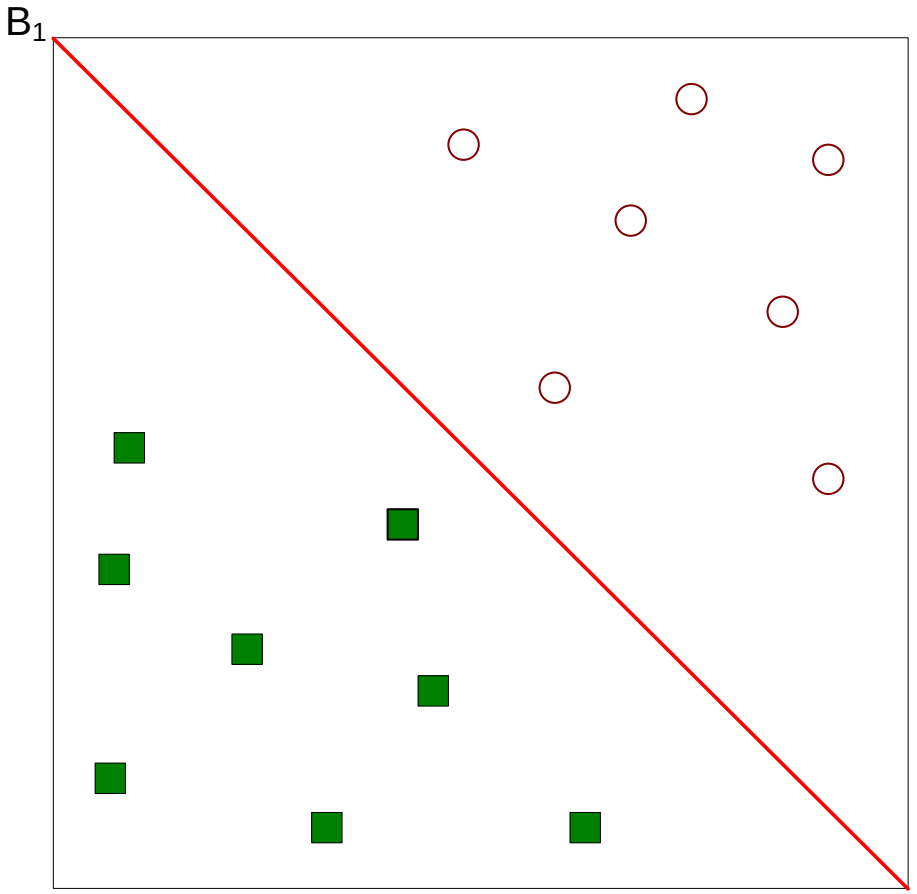
Which one should we choose?

# Example of Bad Decision Boundaries



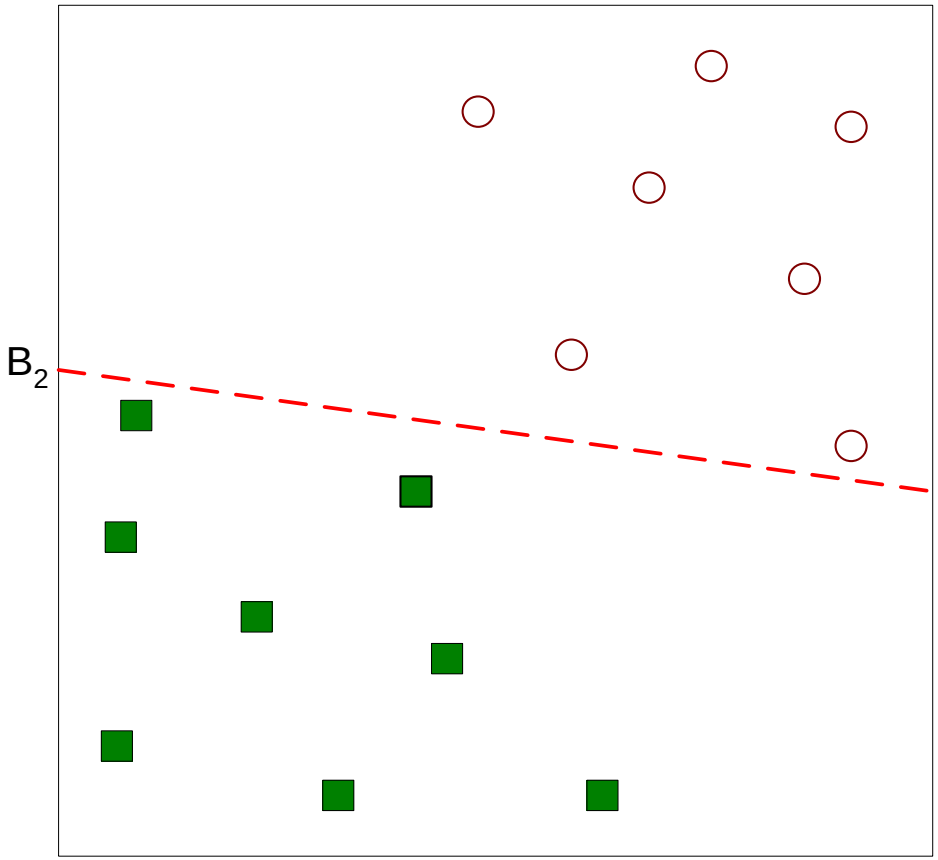



# Decision



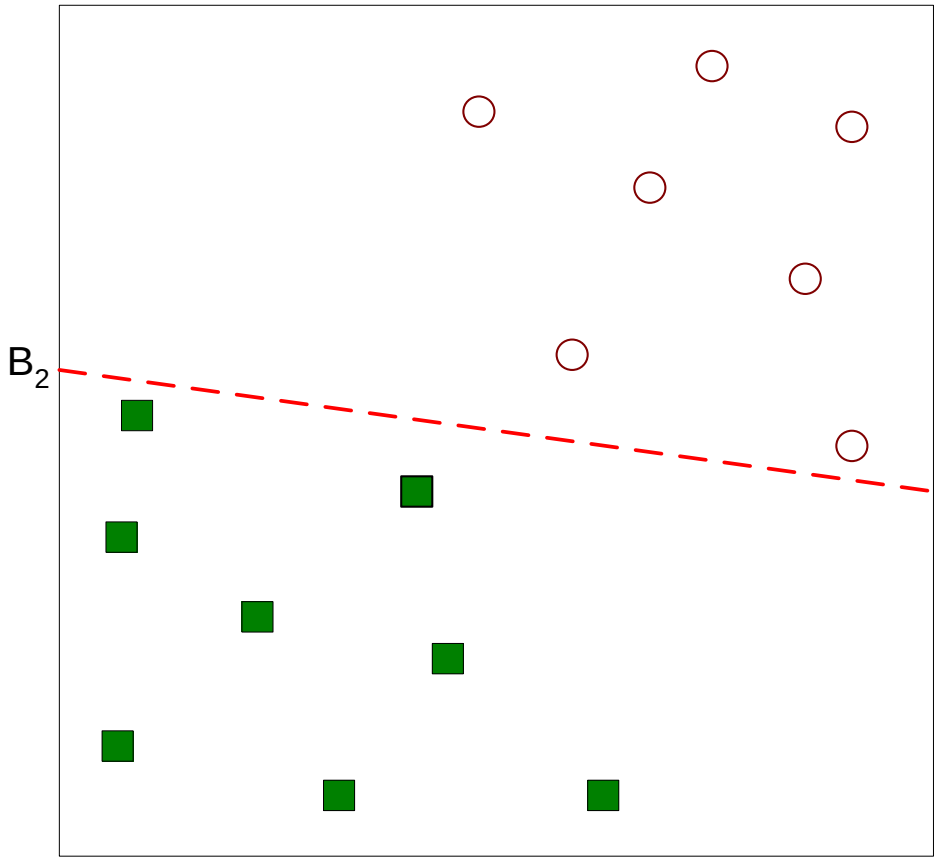

- One Possible Solution

# Decision



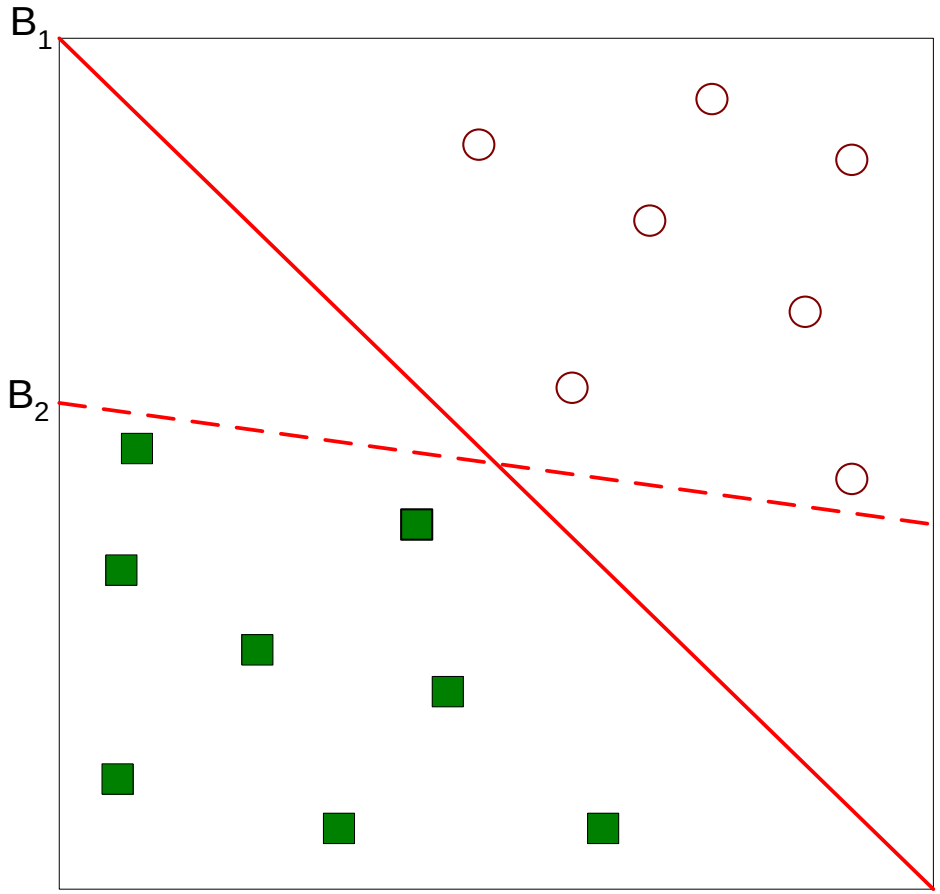

- Another possible solution

# Decision



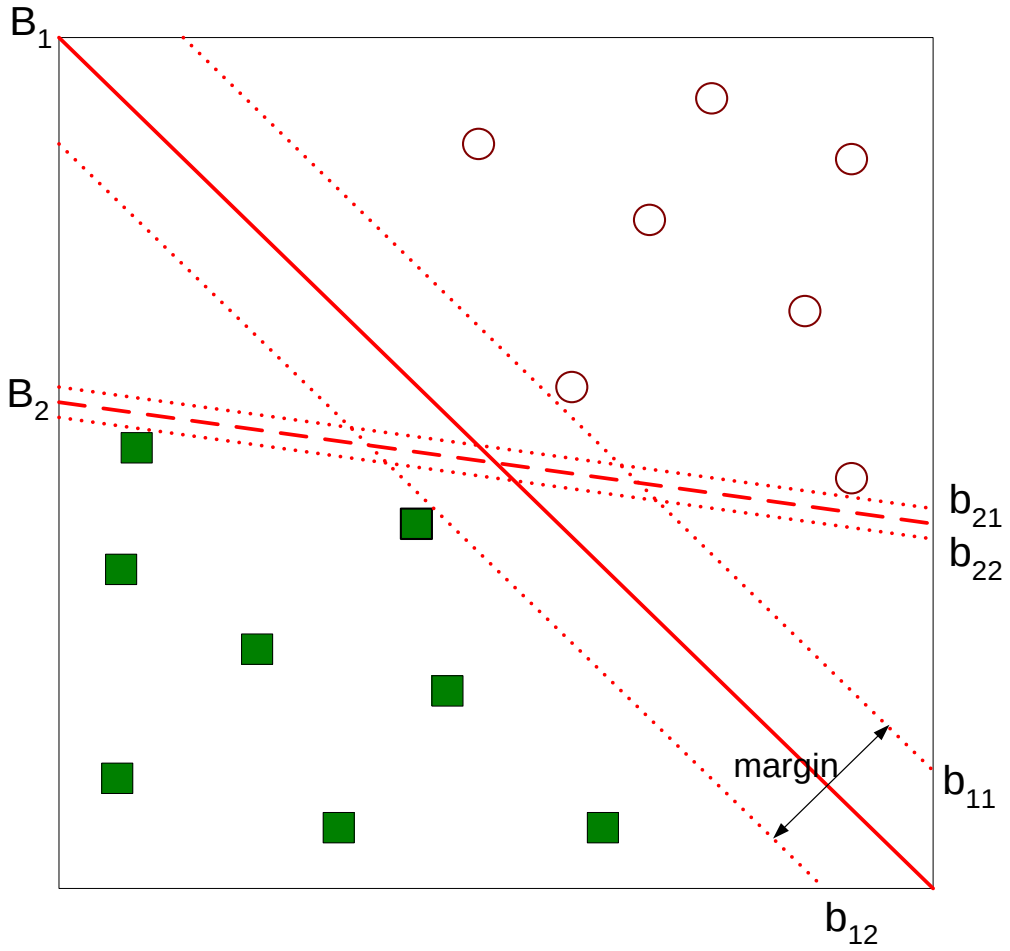

- Other possible solutions

# Decision




- Which one is better? B1 or B2?
- How do you define better?

# Decision

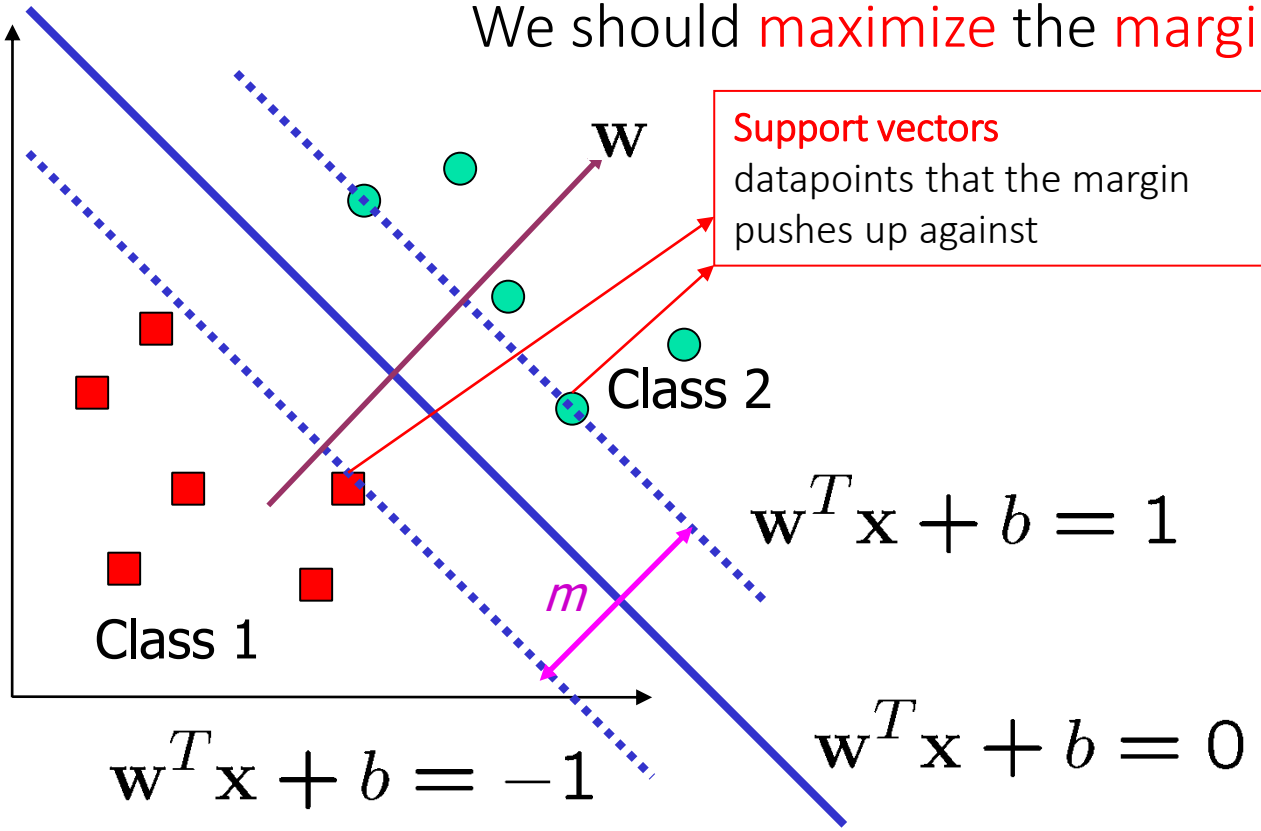



- Find hyperplane maximizes the margin => B1 is better than B2

# Good Decision Boundary

The decision boundary should be as far away from the data of both classes as possible

We should maximize the margin, *m*




$$m = \frac{2}{||\mathbf{w}||}$$

$\mathbf{w}$ → Weights for each feature to predict the output

$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

# The Optimization Problem

Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of $x_i$

The decision boundary should classify all points correctly $\Rightarrow$

$m = \frac{2}{||\mathbf{w}||}$ A constrained optimization problem

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

Minimize $\frac{1}{2}||\mathbf{w}||^2$

- $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$

# Training

- To find the maximum margin separator, we have to solve the following optimization problem:

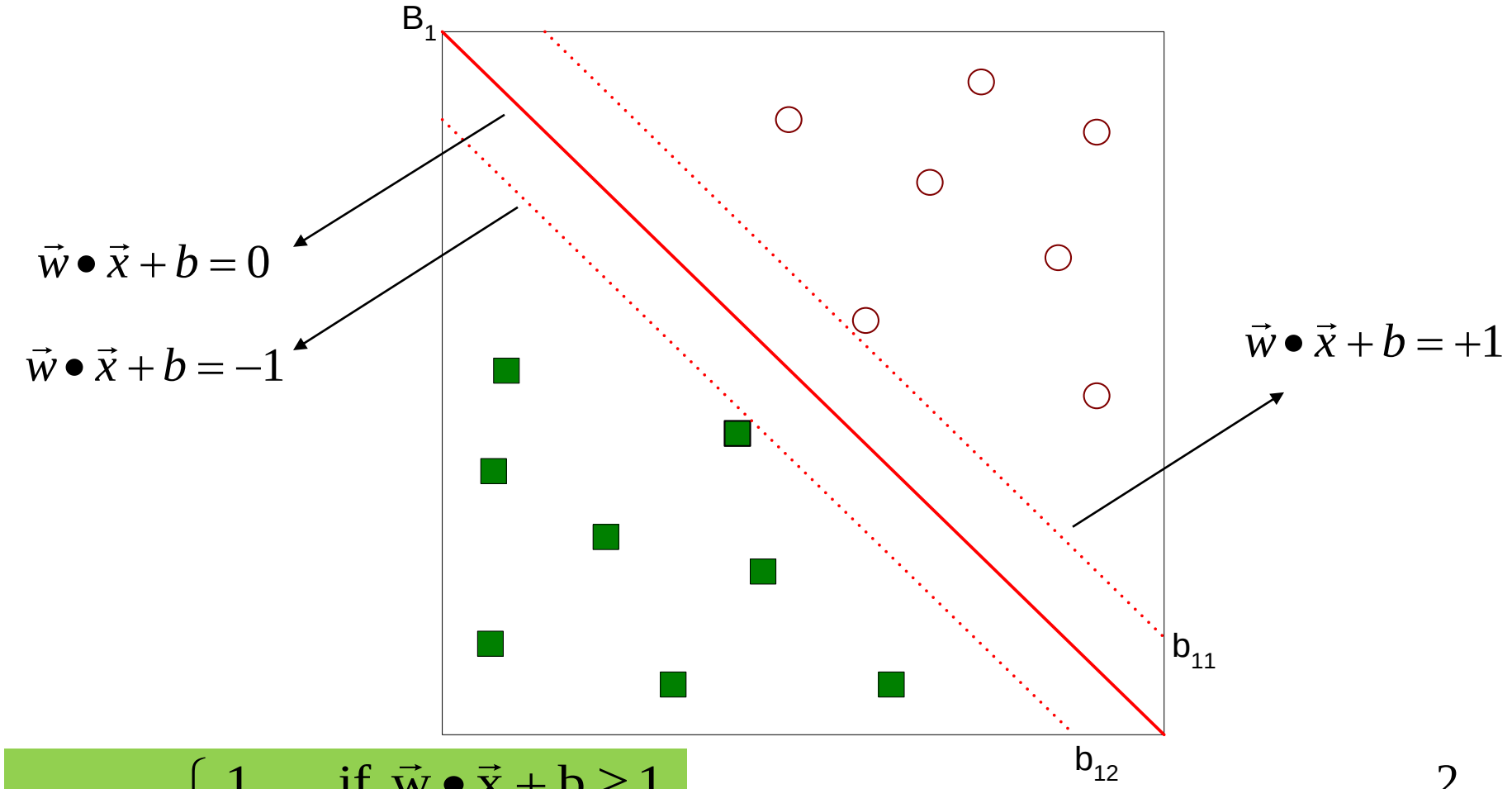
$$\mathbf{w}.\mathbf{x}^c + b > +1 \quad for\ positive\ cases$$

$$\mathbf{w}.\mathbf{x}^c + b < -1 \quad for\ negative\ cases$$

$$and \quad \| \mathbf{w} \|^2 \ is\ as\ small\ as\ possible$$

- This is tricky but it's a convex problem
- There is only one optimum and we can find it without fiddling with learning rates or weight decay or early stopping
  - Don't worry about the optimization problem. It has been solved. Its called quadratic programming.
  - It takes time proportional to $N^2$ which is really bad for very big datasets
    - so for big datasets we end up doing approximate optimization!

# Training




$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

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

# Training

The separator is defined as the set of points for which:

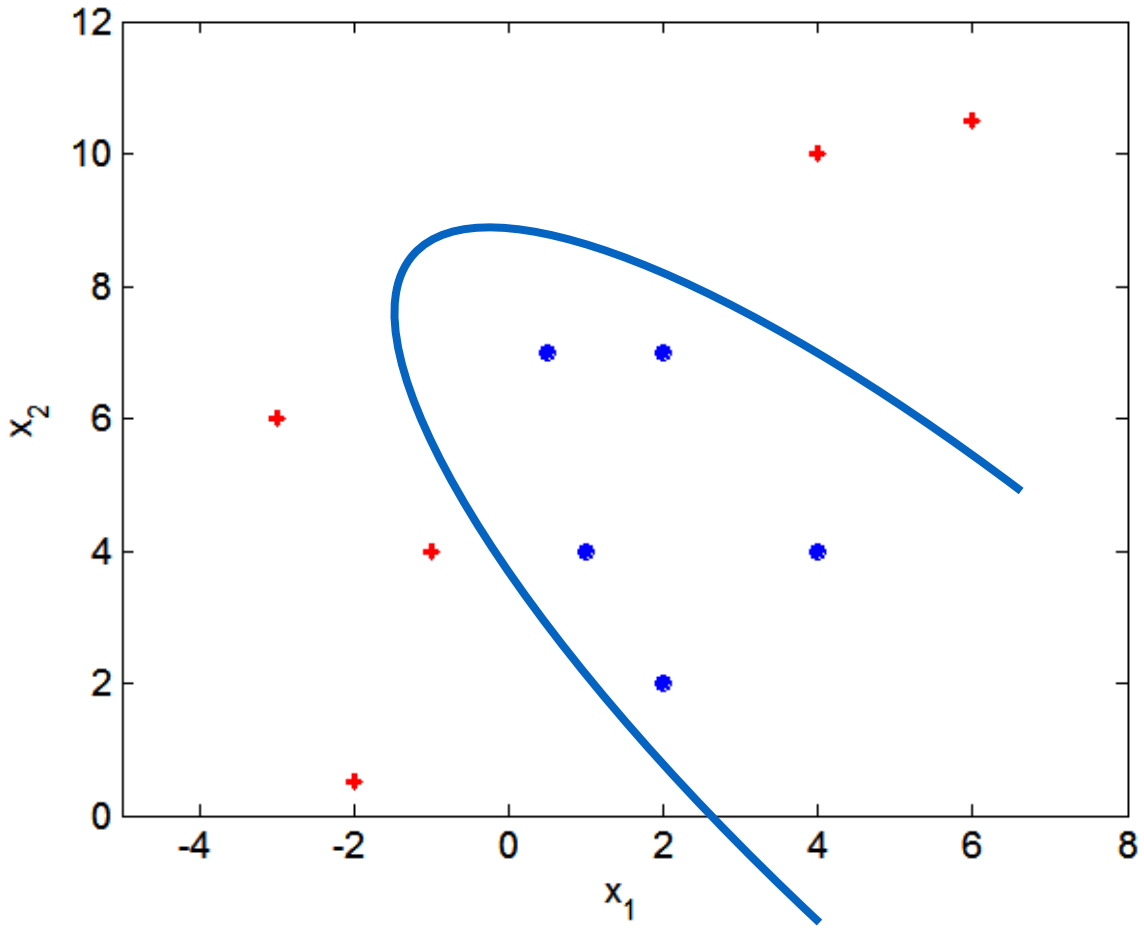
$$\mathbf{w}.\mathbf{x} + b = 0$$

$$so\ if\quad \mathbf{w}.\mathbf{x}^c + b > 0\quad say\ its\ a\ positive\ case$$

$$and\ if\quad \mathbf{w}.\mathbf{x}^c + b < 0\quad say\ its\ a\ negative\ case$$

# Nonlinear Cases

What if decision boundary is not linear?



# Nonlinear Cases

What if the problem is not linearly separable?

Introduce slack variables

Need to minimize:

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left( \sum_{i=1}^{N} \xi_i^k \right)$$

Subject to:

$$\vec{w} \cdot \vec{x_i} + b \geq 1 - \xi_i \quad \text{if } y_i = 1$$
$$\vec{w} \cdot \vec{x_i} + b \leq -1 + \xi_i \quad \text{if } y_i = -1$$

# Clustering

Adopted from ‘Data Mining Concepts and Techniques’ & Debasis, S.

# Introduction

- 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

# Introduction

- **Biology**: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval**: document clustering
- **Land use**: Identification of areas of similar land use in an earth observation database
- **Marketing**: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **City-planning**: Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies**: Observed earth quake epicenters should be clustered along continent faults
- **Climate**: understanding earth climate, find patterns of atmospheric and ocean
- **Economic Science**: market research

# Introduction

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- 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

# Quality of Clustering

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

# Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

# Important Points

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidean, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

# Important Points

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

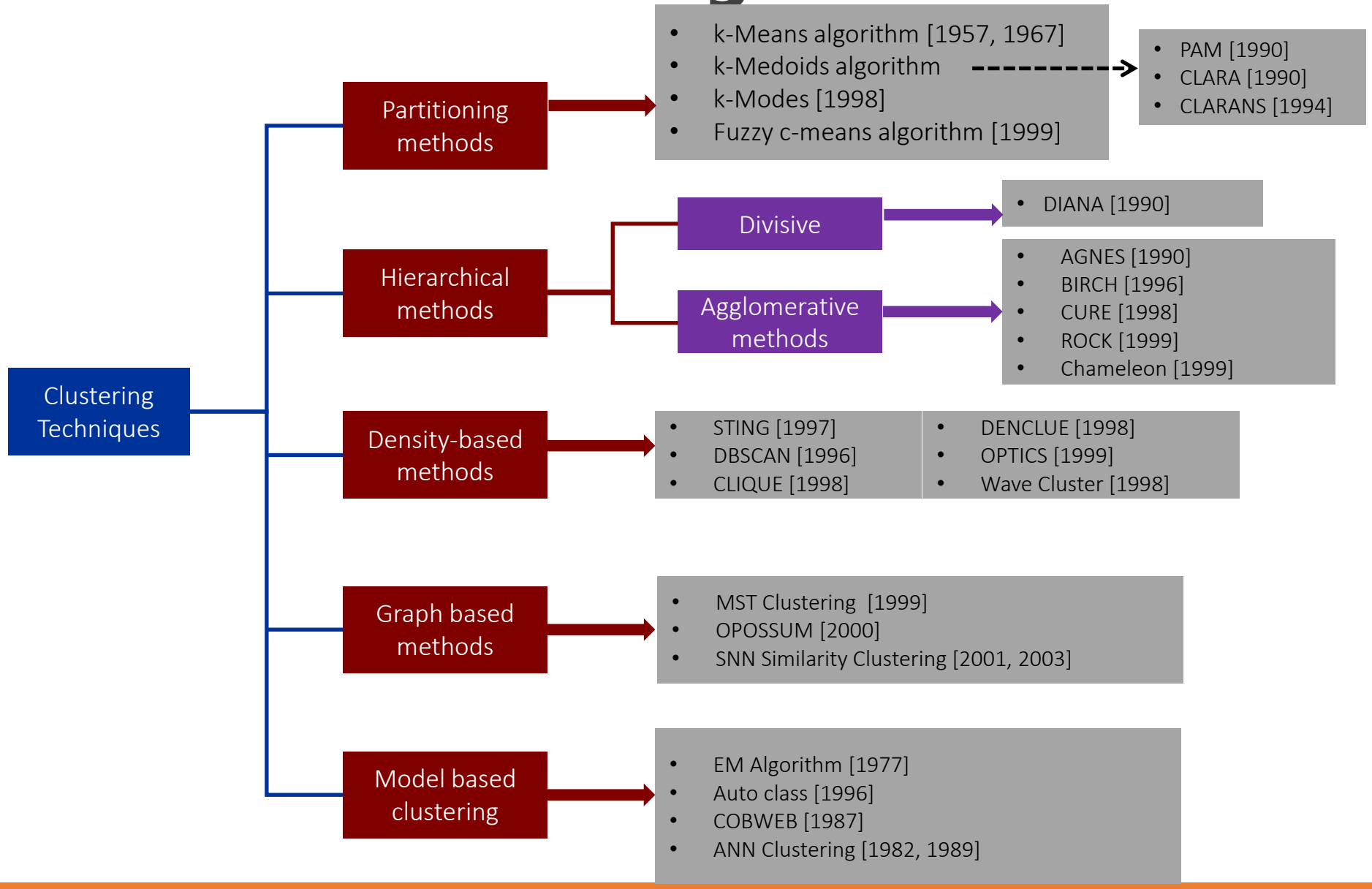
# Categories

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CHAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

# Categories

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

# All Together



# The Problem

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where $c_i$ is the centroid or medoid 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

- Given a set of n distinct objects, the k-Means clustering algorithm partitions the objects into k number of clusters such that intracluster similarity is high but the intercluster similarity is low

- User has to specify k, the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters

# k-Means

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

# k-Means

Algorithm k-Means clustering

Input: D is a dataset containing *n* objects, *k* is the number of cluster
Output: A set of *k* clusters
Steps:

1. Randomly choose *k* objects from D as the initial cluster centroids.

2. **For** each of the objects in D **do**
   - Compute distance between the current objects and *k* cluster centroids
   - Assign the current object to that cluster to which it is closest.

3. Compute the "cluster centers" of each cluster. These become the new cluster centroids.

4. Repeat step 2-3 until the convergence criterion is satisfied

5. Stop

# k-Means

**Note:**

1) Objects are defined in terms of set of attributes. $A = \{A_1, A_2, \ldots\ldots, A_m\}$ where each $A_i$ is continuous data type.

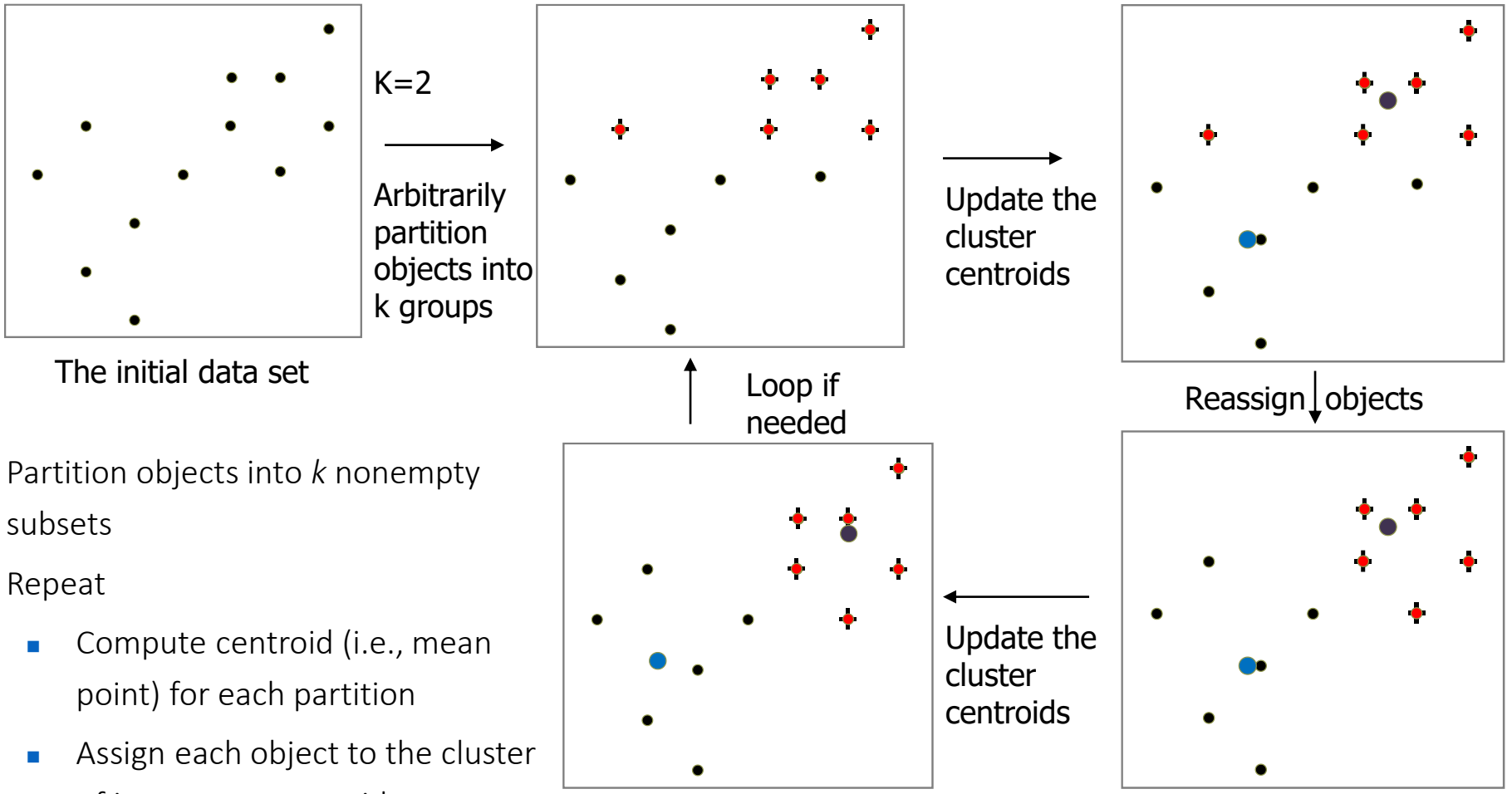
2) Distance computation: Any distance such as $L_1, L_2, L_3$ or cosine similarity.

3) Minimum distance is the measure of closeness between an object and centroid.

4) Mean Calculation: It is the mean value of each attribute values of all objects.

5) Convergence criteria: Any one of the following are termination condition of the algorithm.
   - Number of maximum iteration permissible.
   - No change of centroid values in any cluster.
   - Zero (or no significant) movement(s) of object from one cluster to another.
   - Cluster quality reaches to a certain level of acceptance.

# k-Means



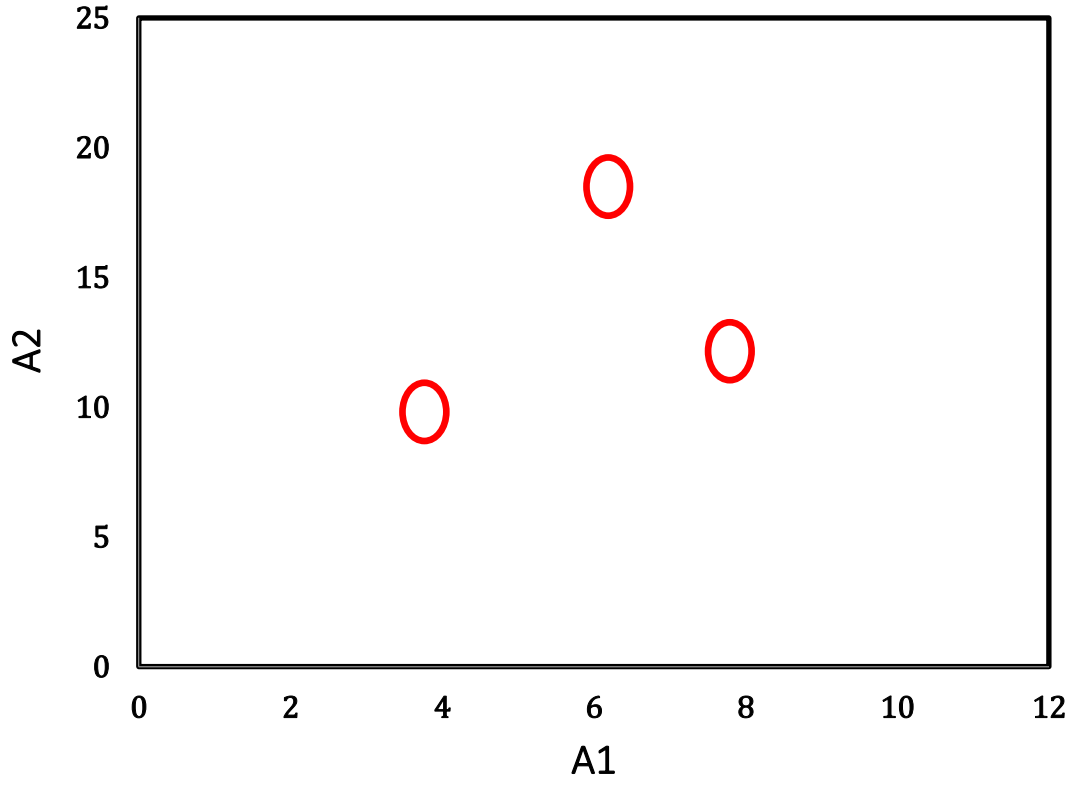

- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

# k-Means

Table : objects with two attributes $A_1$ and $A_2$.

| $A_1$ | $A_2$ |
|---|---|
| 6.8 | 12.6 |
| 0.8 | 9.8 |
| 1.2 | 11.6 |
| 2.8 | 9.6 |
| 3.8 | 9.9 |
| 4.4 | 6.5 |
| 4.8 | 1.1 |
| 6.0 | 19.9 |
| 6.2 | 18.5 |
| 7.6 | 17.4 |
| 7.8 | 12.2 |
| 6.6 | 7.7 |
| 8.2 | 4.5 |
| 8.4 | 6.9 |
| 9.0 | 3.4 |
| 9.6 | 11.1 |

Fig: Plotting data of Table



# k-Means

- Suppose, k=3. Three objects are chosen at random shown as circled (see previous Fig). These three centroids are shown below.

Initial Centroids chosen randomly

| Centroid | Objects | |
|---|---|---|
| | A1 | A2 |
| $c_1$ | 3.8 | 9.9 |
| $c_2$ | 7.8 | 12.2 |
| $c_3$ | 6.2 | 18.5 |

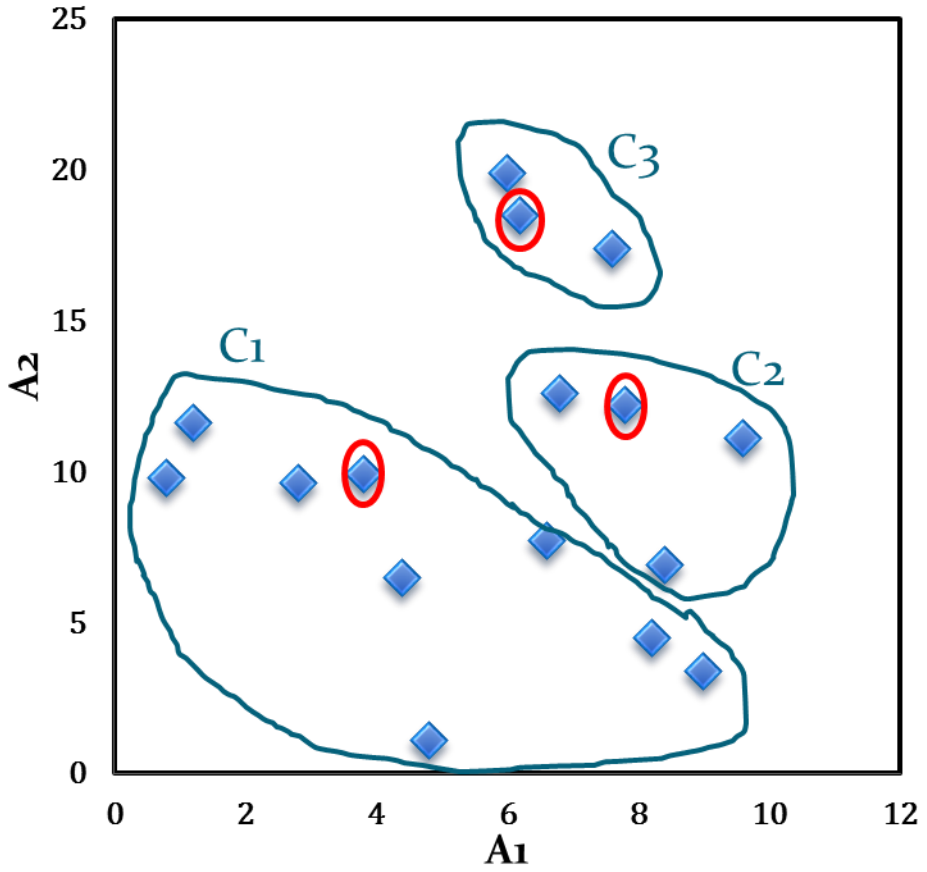
- Let us consider the Euclidean distance measure ($L_2$ Norm) as the distance measurement in our illustration.
- Let $d_1$, $d_2$ and $d_3$ denote the distance from an object to $c_1$, $c_2$ and $c_3$ respectively. The distance calculations are shown in the Table.
- Assignment of each object to the respective centroid is shown in the right-most column and the clustering so obtained is shown in the upcoming Fig.

# k-Means

Table: Distance calculation

| $A_1$ | $A_2$ | $d_1$ | $d_2$ | $d_3$ | cluster |
|---|---|---|---|---|---|
| 6.8 | 12.6 | 4.0 | 1.1 | 5.9 | 2 |
| 0.8 | 9.8 | 3.0 | 7.4 | 10.2 | 1 |
| 1.2 | 11.6 | 3.1 | 6.6 | 8.5 | 1 |
| 2.8 | 9.6 | 1.0 | 5.6 | 9.5 | 1 |
| 3.8 | 9.9 | 0.0 | 4.6 | 8.9 | 1 |
| 4.4 | 6.5 | 3.5 | 6.6 | 12.1 | 1 |
| 4.8 | 1.1 | 8.9 | 11.5 | 17.5 | 1 |
| 6.0 | 19.9 | 10.2 | 7.9 | 1.4 | 3 |
| 6.2 | 18.5 | 8.9 | 6.5 | 0.0 | 3 |
| 7.6 | 17.4 | 8.4 | 5.2 | 1.8 | 3 |
| 7.8 | 12.2 | 4.6 | 0.0 | 6.5 | 2 |
| 6.6 | 7.7 | 3.6 | 4.7 | 10.8 | 1 |
| 8.2 | 4.5 | 7.0 | 7.7 | 14.1 | 1 |
| 8.4 | 6.9 | 5.5 | 5.3 | 11.8 | 2 |
| 9.0 | 3.4 | 8.3 | 8.9 | 15.4 | 1 |
| 9.6 | 11.1 | 5.9 | 2.1 | 8.1 | 2 |

Fig: Initial cluster with respect to Table



# k-Means

The calculation new centroids of the three cluster using the mean of attribute values of $A_1$ and $A_2$ is shown in the Table below. The cluster with new centroids are shown in Fig.

Calculation of new centroids

| New Centroid | Objects | |
|---|---|---|
| | A1 | A2 |
| $c_1$ | 4.6 | 7.1 |
| $c_2$ | 8.2 | 10.7 |
| $c_3$ | 6.6 | 18.6 |

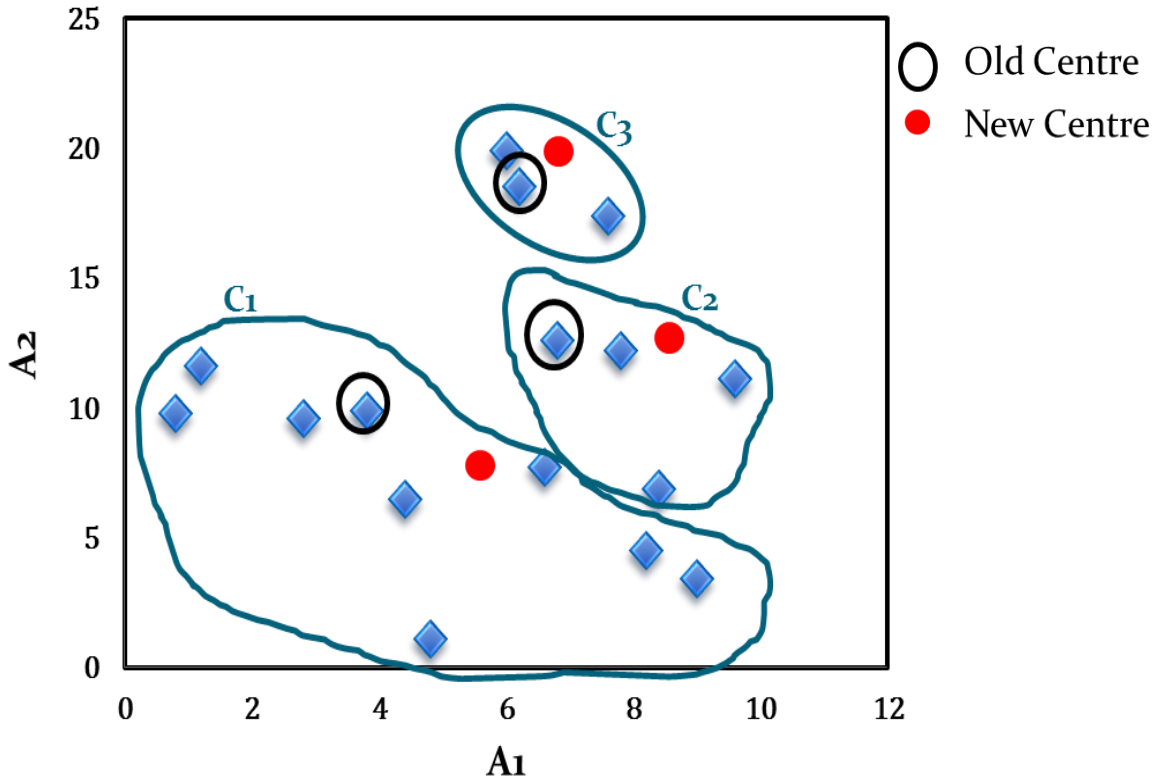



Fig: Initial cluster with new centroids

# k-Means

We next reassign the 16 objects to three clusters by determining which centroid is closest to each one. This gives the revised set of clusters shown in Fig.

Note that point p moves from cluster $C_2$ to cluster $C_1$.

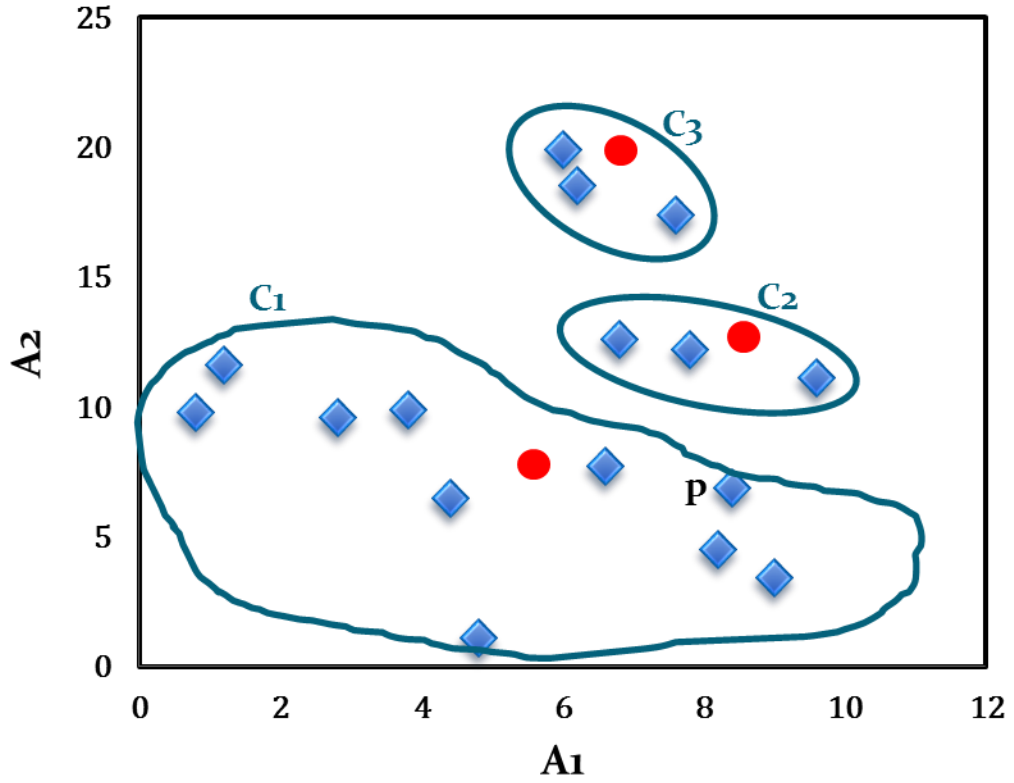



Fig : Cluster after first iteration

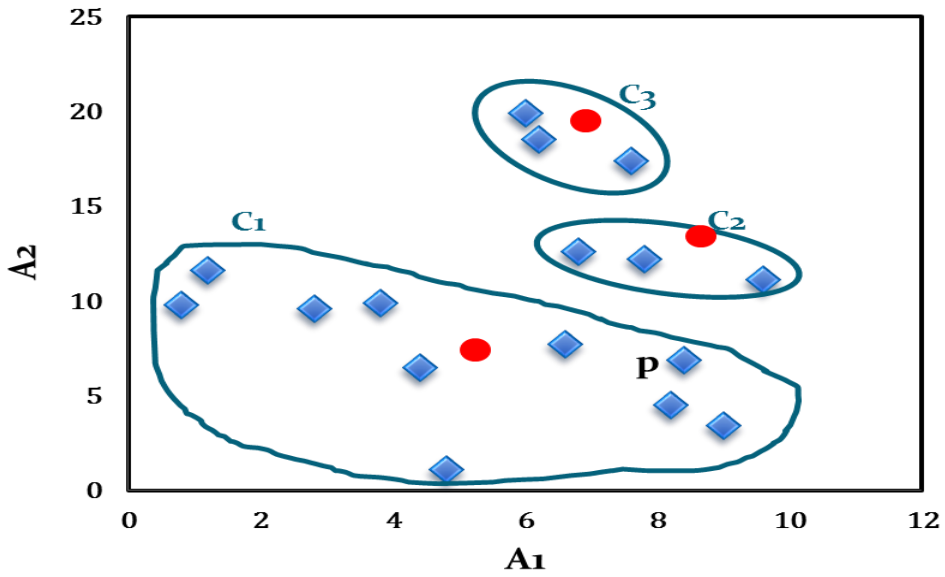
# k-Means

- The newly obtained centroids after second iteration are given in the table below. Note that the centroid $c_3$ remains unchanged, where $c_2$ and $c_1$ changed a little.
- With respect to newly obtained cluster centres, 16 points are reassigned again. These are the same clusters as before. Hence, their centroids also remain unchanged.
- Considering this as the termination criteria, the k-means algorithm stops here. Hence, the final cluster in the current Fig is same as in the previous Fig.

Cluster centres after second iteration

| Centroid | Revised Centroids | |
|---|---|---|
| | A1 | A2 |
| $c_1$ | 5.0 | 7.1 |
| $c_2$ | 8.1 | 12.0 |
| $c_3$ | 6.6 | 18.6 |

Fig : Cluster after Second iteration



# k-Means

- Visualization

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

# k-Means

- Let us analyse the k-Means algorithm and discuss the pros and cons of the algorithm.
- We shall refer to the following notations in our discussion.
- Notations:
  - $x$ : an object under clustering
  - $n$ : number of objects under clustering
  - $\boldsymbol{C}_i$ : the *i-th* cluster
  - $c_i$ : the centroid of cluster $\boldsymbol{C}_i$
  - $n_i$ : number of objects in the cluster $\boldsymbol{C}_i$
  - $c$ : denotes the centroid of all objects
  - $k$ : number of clusters

# k-Means

1. Value of k:
   - The k-means algorithm produces only one set of clusters, for which, user must specify the desired number, *k* of clusters.
   - In fact, *k* should be the best guess on the number of clusters present in the given data. Choosing the best value of *k* for a given dataset is, therefore, an issue.
   - We may not have an idea about the possible number of clusters for high dimensional data, and for data that are not scatter-plotted.
   - Further, possible number of clusters is hidden or ambiguous in image, audio, video and multimedia clustering applications etc.
   - There is no principled way to know what the value of *k* ought to be. We may try with successive value of k starting with 2.
   - The process is stopped when two consecutive *k* values produce more-or-less identical results (with respect to some cluster quality estimation).
   - Normally $k \ll n$ and there is heuristic to follow $k \approx \sqrt{n}$.
   - Elbow method

# k-Means

## Example: k versus cluster quality

- Usually, there is some objective function to be met as a goal of clustering.
- One such objective function is sum-square-error denoted by SSE and defined as

$$SSE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (x - c_i)^2$$

- Here, $x - c_i$ denotes the error, if $x$ is in cluster $\boldsymbol{C}_i$ with cluster centroid $c_i$.
- Usually, this error is measured as distance norms like $L_1$, $L_2$, $L_3$ or Cosine similarity, etc.

# k-Means

## Example k versus cluster quality

- With reference to an arbitrary experiment, suppose the following results are obtained.

| k | SSE |
|---|---|
| 1 | 62.8 |
| 2 | 12.3 |
| 3 | 9.4 |
| 4 | 9.3 |
| 5 | 9.2 |
| 6 | 9.1 |
| 7 | 9.05 |
| 8 | 9.0 |

- With respect to this observation, we can choose the value of $k \approx 3$, as with this smallest value of k it gives reasonably good result.
- Note: If $k = n$, then SSE=0; However, the cluster is useless! This is another example of overfitting.

# k-Means

2. Choosing initial centroids:
   - Another requirement in the k-Means algorithm to choose initial cluster centroid for each *k* would be clusters.
   - It is observed that the k-Means algorithm terminate whatever be the initial choice of the cluster centroids.
   - It is also observed that initial choice influences the ultimate cluster quality. In other words, the result may be trapped into local optima, if initial centroids are chosen properly.
   - One technique that is usually followed to avoid the above problem is to choose initial centroids in multiple runs, each with a different set of randomly chosen initial centroids, and then select the best cluster (with respect to some quality measurement criterion, e.g. SSE).
   - However, this strategy suffers from the combinational explosion problem due to the number of all possible solutions.

# k-Means

## 2. Choosing initial centroids:

- A detail calculation reveals that there are $c(n, k)$ possible combinations to examine the search of global optima.

$$c(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}(i)^n$$

- For example, there are $o(10^{10})$ different ways to cluster 20 items into 4 clusters!

- Thus, the strategy having its own limitation is practical only if
  1) The sample is negatively small (~100-1000), and
  2) *k* is relatively small compared to *n* (i.e.. $k \ll n$).

# k-Means

## Distance Measurement:

- To assign a point to the closest centroid, we need a proximity measure that should quantify the notion of "closest" for the objects under clustering.
- Usually Euclidean distance ($L_2$ norm) is the best measure when object points are defined in n-dimensional Euclidean space.
- Other measure namely cosine similarity is more appropriate when objects are of document type.
- Further, there may be other type of proximity measures that appropriate in the context of applications.
- For example, Manhattan distance ($L_1$ norm), Jaccard measure, etc.

# k-Means

## Distance Measurement:

- Thus, in the context of different measures, the sum-of-squared error (i.e., objective function/convergence criteria) of a clustering can be stated as under.
- Data in Euclidean space ($L_2$ norm):

$$SSE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} (c_i - x)^2$$

- Data in Euclidean space ($L_1$ norm):
- The Manhattan distance ($L_1$ norm) is used as a proximity measure, where the objective is to minimize the sum-of-absolute error denoted as SAE and defined as

$$SAE = \sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} |c_i - x|$$

# k-Means

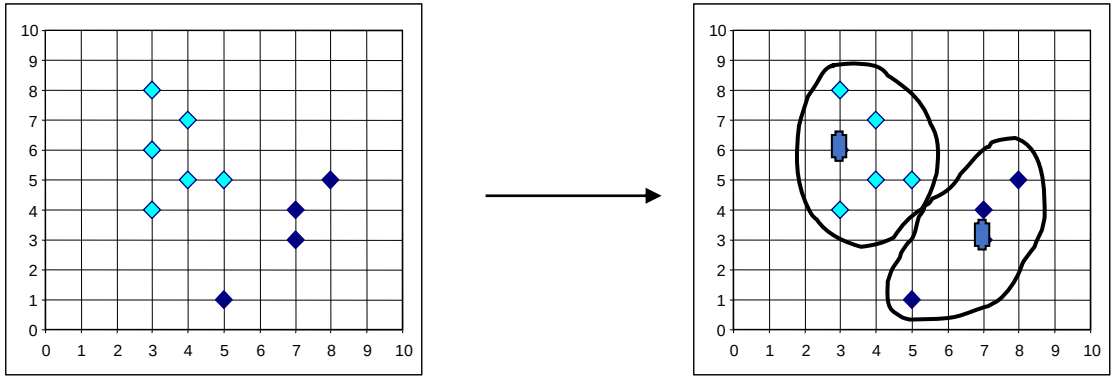
- Strength: *Efficient*: $O(tkn)$, where $n$ is # objects, $k$ is # clusters, and $t$ is # iterations. Normally, $k$, $t << n$.
    - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
    - Applicable only to objects in a continuous n-dimensional space
        - Using the k-modes method for categorical data
        - In comparison, k-medoids can be applied to a wide range of data
    - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
    - Sensitive to noisy data and *outliers*
    - Not suitable to discover clusters with *non-convex shapes*

# k-Means

- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

# k-Means

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster



# Other Approaches

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling