ML for Business Analytics

Recap – Data Mining Algorithms

Logistics:

1. 3 weekly hours
2. Assignments (30%), Project (70%)
3. Communication through Moodle / Email
4. Weekly reception hour
5. Email - or.izhakp@afeka.ac.il

What You Are Getting

1. Deep knowledge of machine learning processes and pipelines
2. The ability to solve real-world problem using ML
3. Technical skills: improvement of scientific computing programming skills using Python
4. Build an end-to-end machine learning project

What You Are Giving

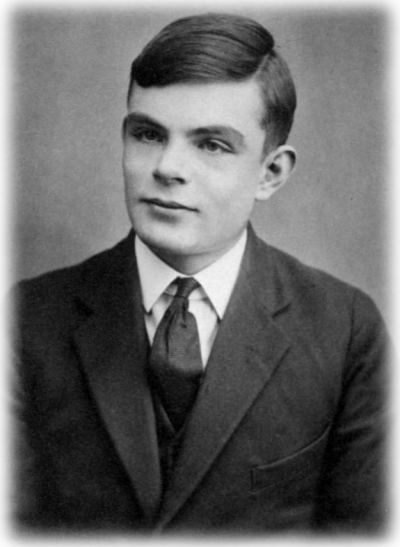
1. three assignments (groups of 2-3 members)
2. Final project
3. Class participation
4. **Smiles and fun ☺**

Introduction

Introduction

“What we want is a machine that can learn from experience“

Alan Turing, 1947



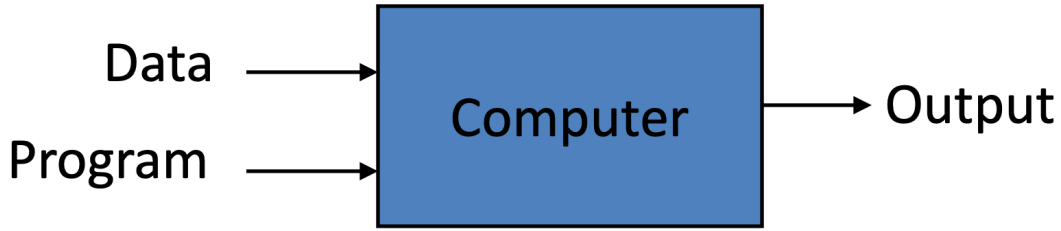
Introduction

“Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.”

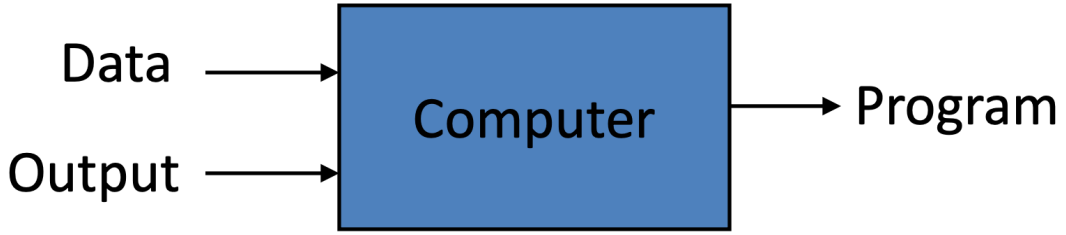
Arthur Samuel, 1959

Introduction

Traditional Programming



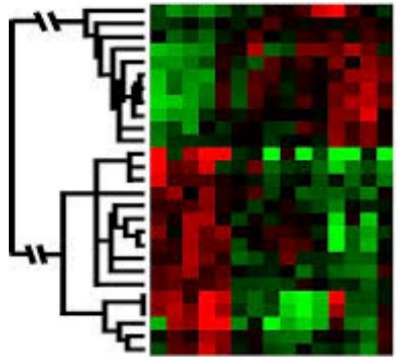
Machine Learning



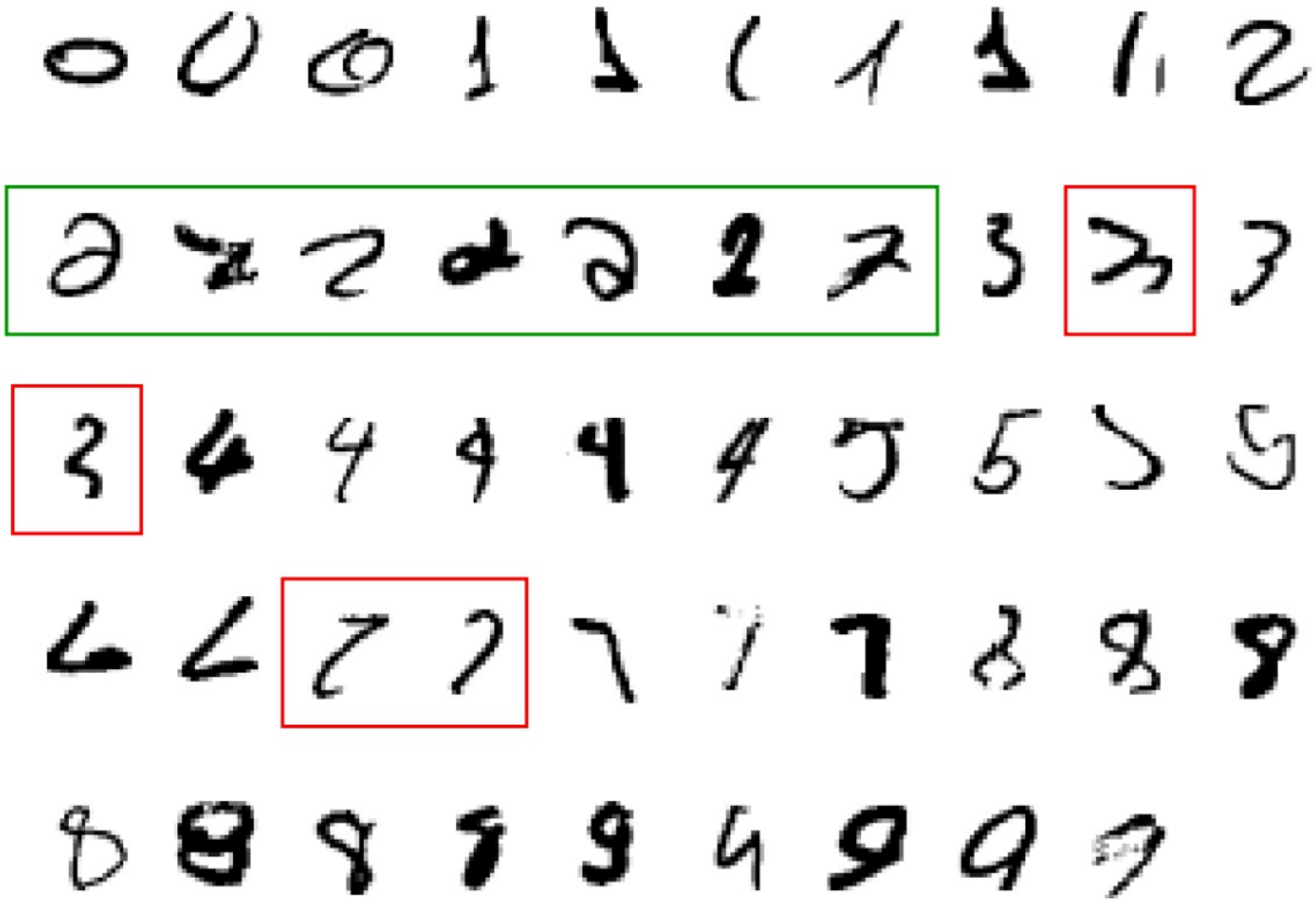
When Do We Use ML?

- Human expertise does not exist (navigating on Mars)
- Humans can’t explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



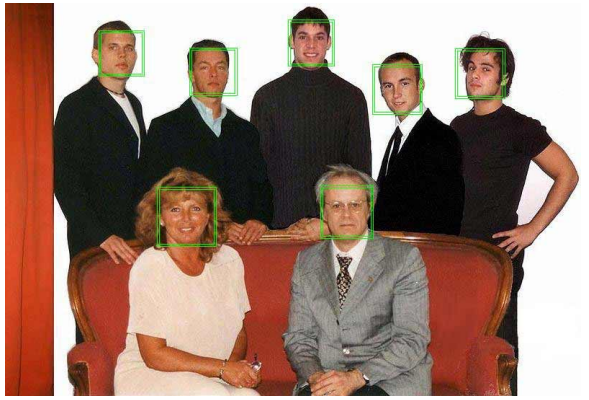
Where is the “2” Digit?



Tasks using ML

Recognizing patterns

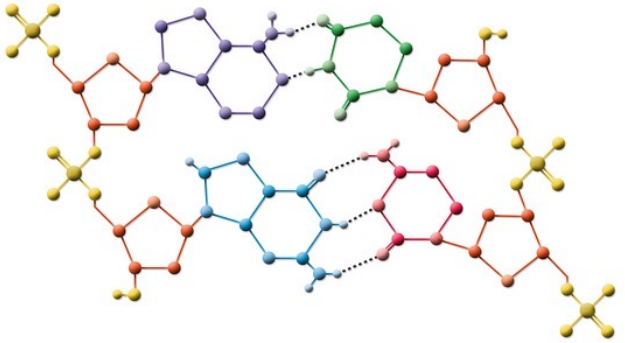
- Facial identities or facial expressions
- Handwritten or spoken words
- Medical images



Generating patterns: Generating images or motion sequences

Recognizing anomalies:

- Unusual credit card transactions
- Unusual patterns of sensor readings in a nuclear power plant

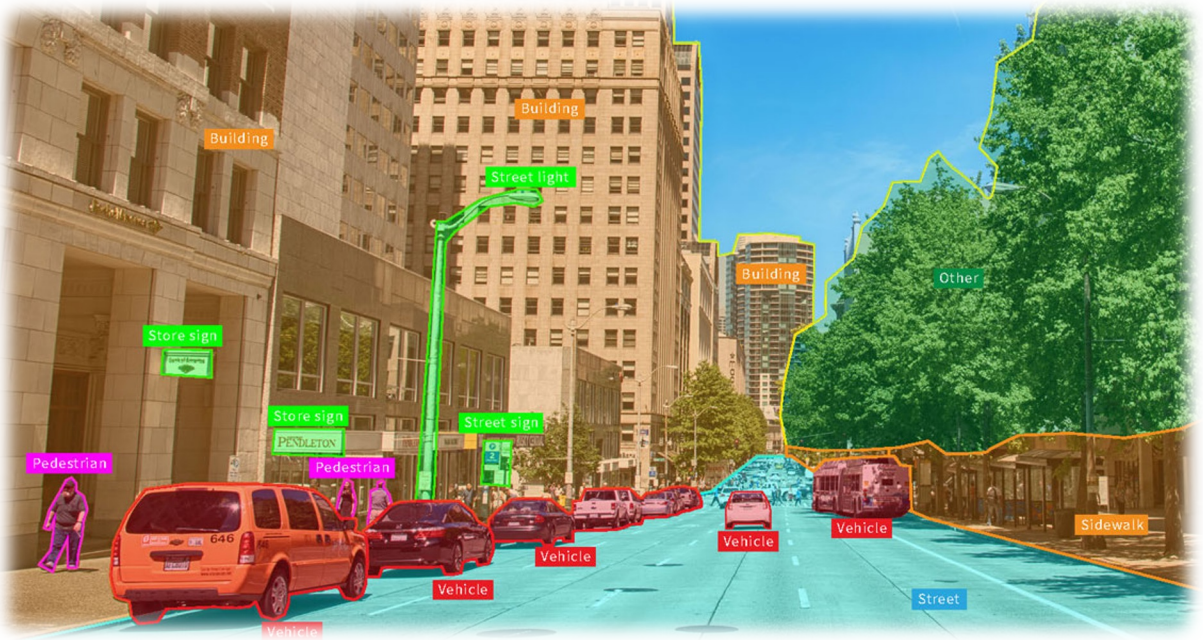


Prediction: Future stock prices or currency exchange rates

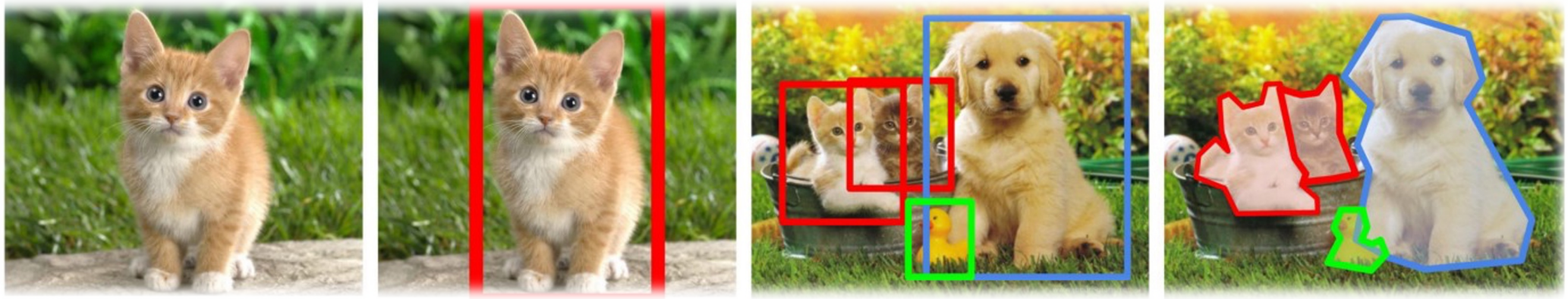
Sample Applications

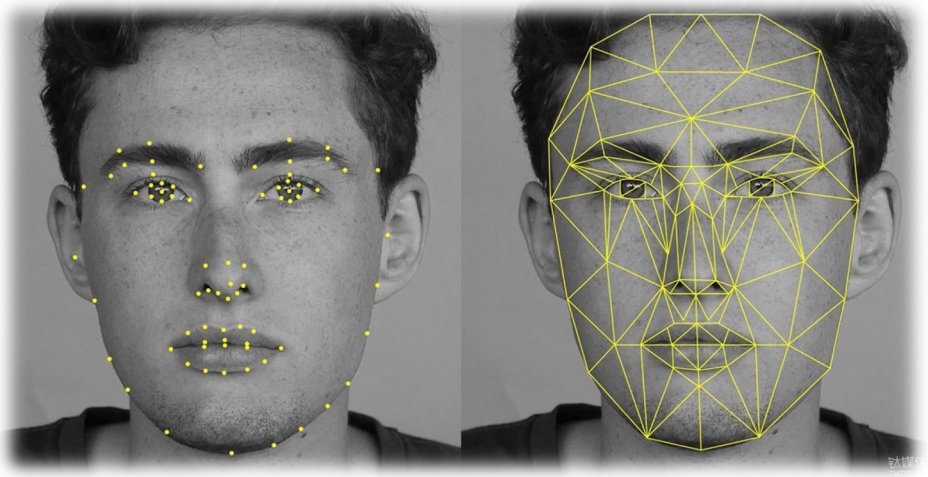
- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks

Sample Applications



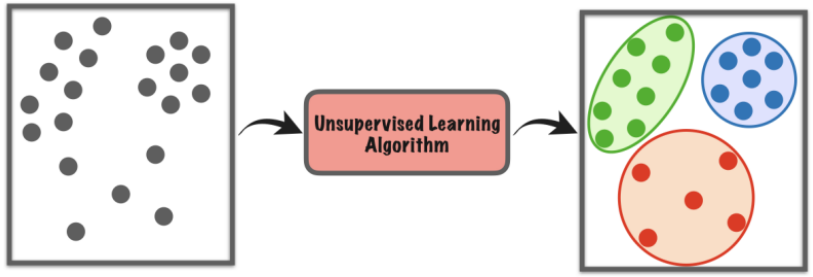
Sample Applications





Types of Learning

Unsupervised learning – Given training data (without desired outputs)



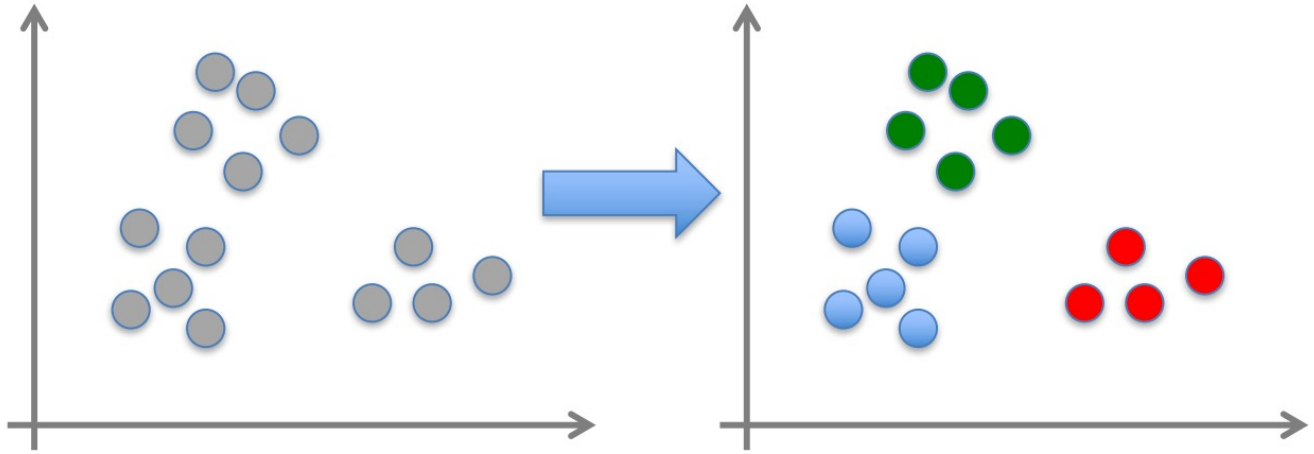
Reinforcement learning

Supervised learning – Given training data + desired outputs (labels/target)

Unsupervised Learning

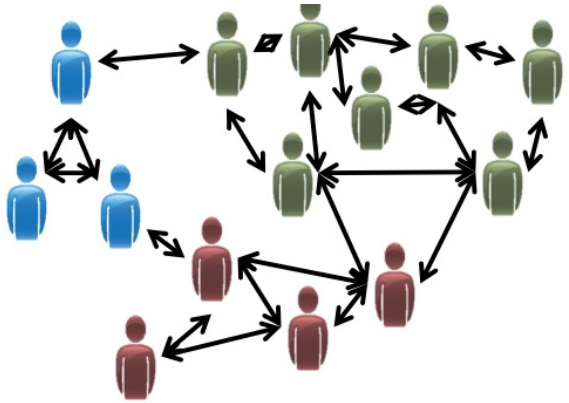
Input: $x_1, x_2, \ldots, x_n$

Process: Find hidden structure behind the $\{x_i\}$



Unsupervised Learning

Organize computing clusters



Social network analysis

Market segmentation

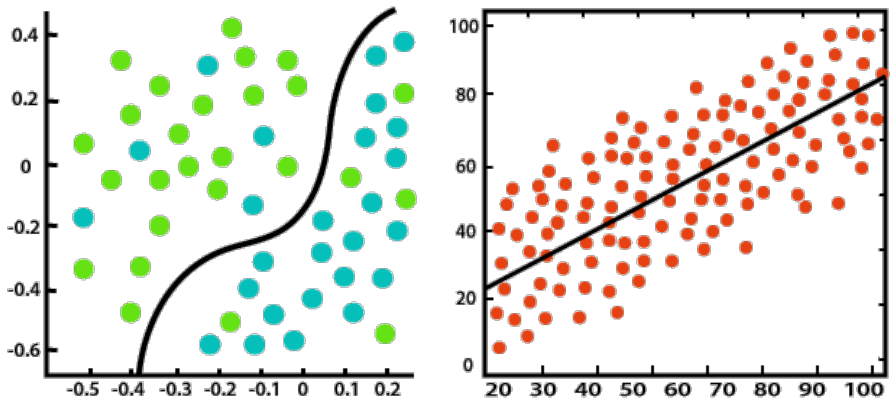
Astronomical data analysis

Types of Learning

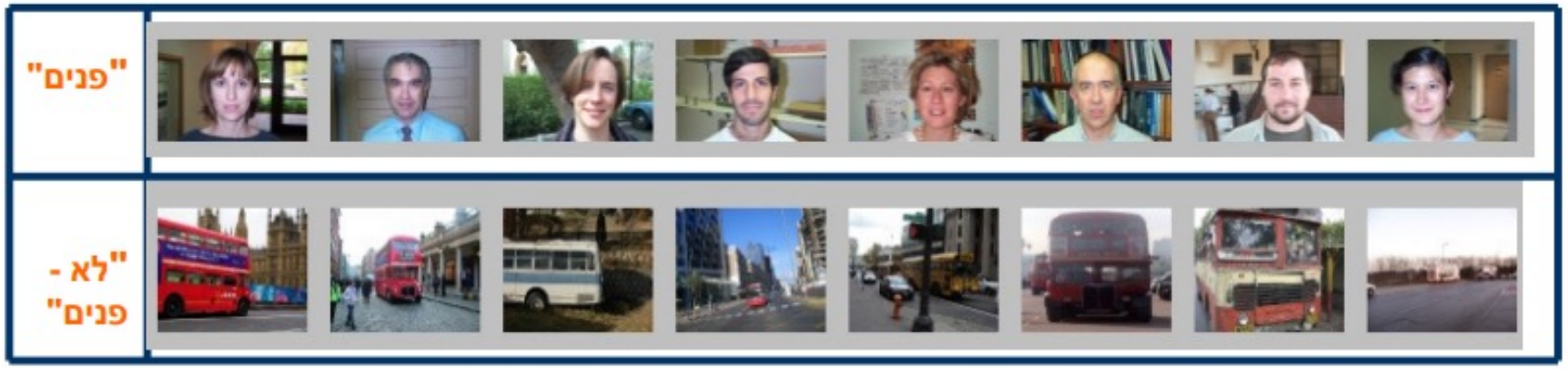
Supervised learning – Given training data + desired outputs (labels/target)

Classification - predicting a discrete-valued target

Regression - predicting a continuous-valued target



Supervised Learning



Given a new input, the computer need to decide: face or not?

Supervised Learning

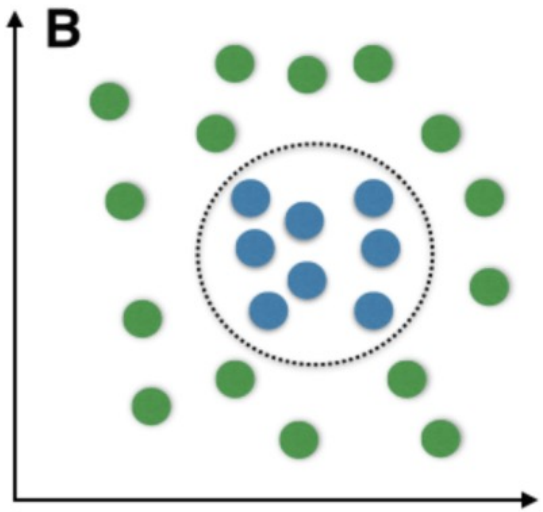
Input: Set of vectors $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Process: Learn a function $f(x)$ to predict y given x

Example,

Each x_i is an image, y_i is 1 if the image include face, otherwise 0.

Given an image (x_j) predict if it has face or not.



Learning Process

1. **Collect Data**

   Define and collect the data for the project, define the problem (supervised/unsupervised)

2. **Data Presentation**

   Transform the data into machine formats

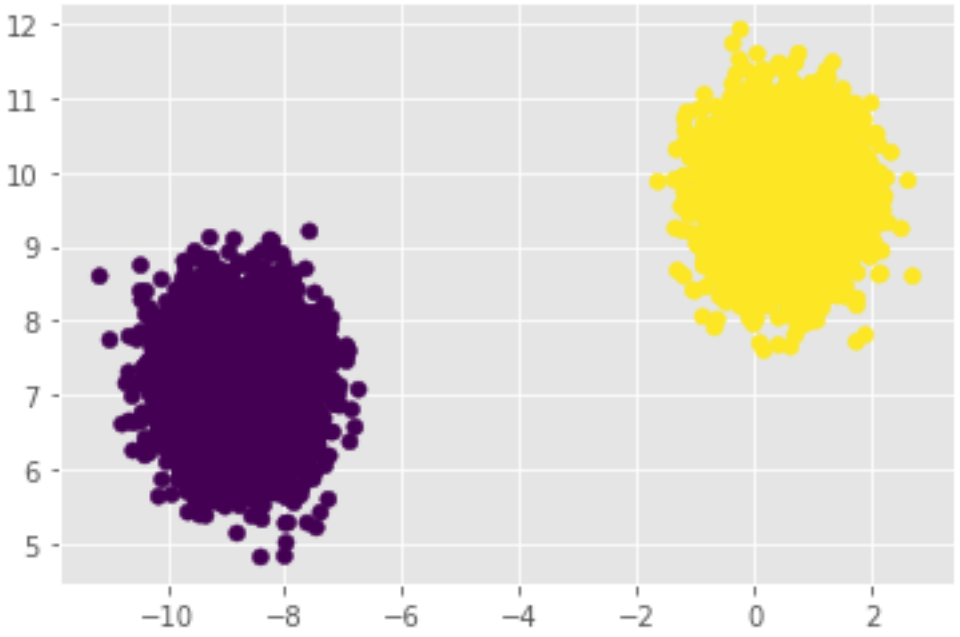
3. **Learning Algorithm**

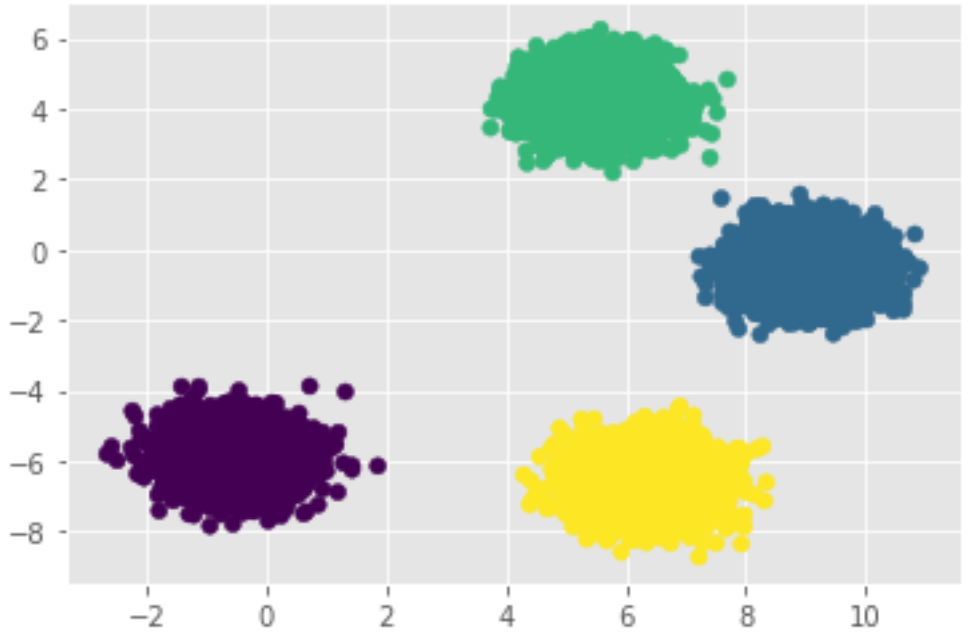
   Choose and evaluate a learning algorithm

4. **Performance Metrics**

   Calculate the performance metrics to answer the question “is my algorithm did well?”

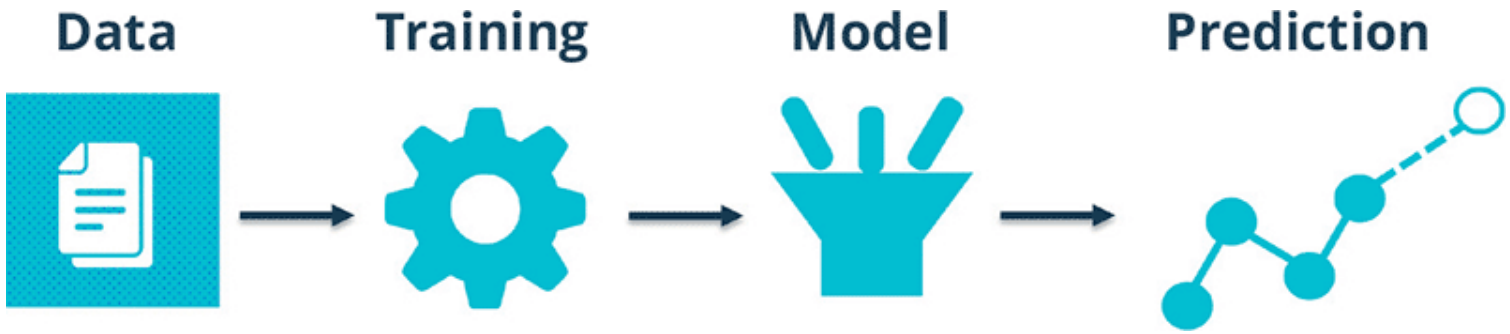
Binary vs. Multiclass Problem





The Training Process

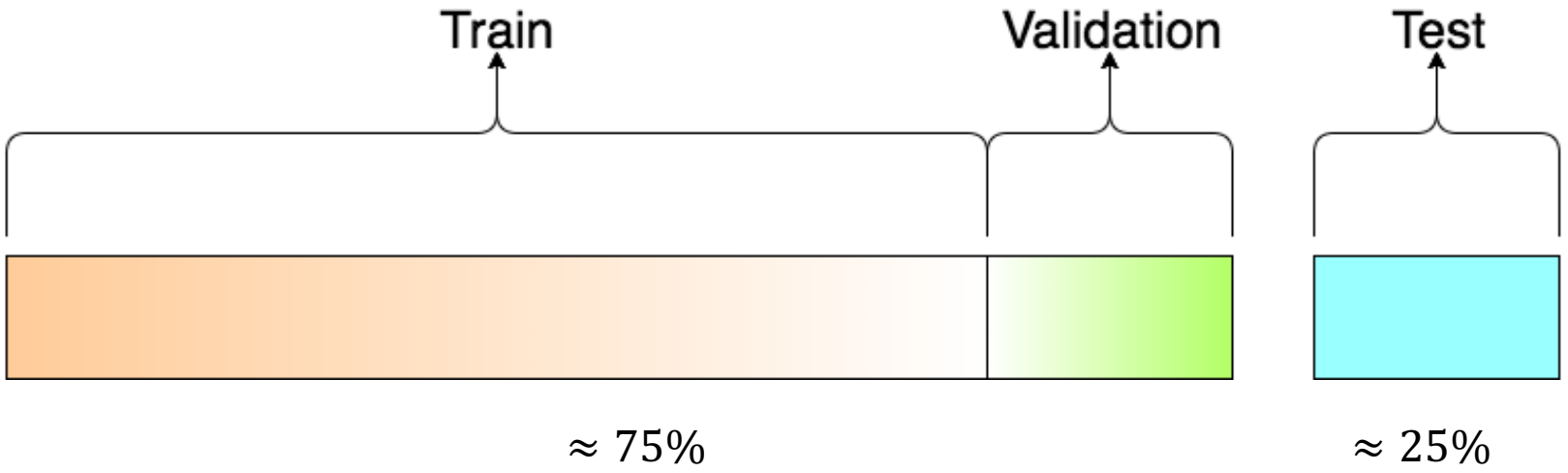
Consider you are a chocolate tester, and your job is to taste chocolates and identify their type. At your first day, you will taste every chocolate to be familiar with the taste. When you are ready, the company would examine your knowledge by a simple test: they will give you a chocolate and you will need to identify the type.



Train, Validation and Test Sets

In practice we want to assess performance “out of sample”

how well will the classifier do on new unseen data?



Train, Validation and Test Sets

Train - The sample of data used to fit the model.

Validation - The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. (“semi-testing” your model)

Test - The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

How to Evaluate My Supervised Algorithm?

TP (True Positive) - Actually positive and predicted to be positive

FN (False Negative) - Actually positive but predicted to be negative

FP (False Positive) - Negative but predicted to be positive

TN (True Negative) - Negative and predicted to be negative

True **N**egative	**F**alse **P**ositive
False **N**egative	**T**rue **P**ositive

Confusion Matrix - Example

actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]

predicted = [1, 0, 0, 1, 0, 0, 1, 1, 1, 0]

TP = 3, FP = 2

TN = 4, FN = 1

True **N**egative	**F**alse **P**ositive
False **N**egative	**T**rue **P**ositive

4	**2**
1	**3**

Accuracy

Accuracy - the ratio of the number of correct predictions made to all the predictions

$$\frac{TP + TN}{TP + TN + FP + FN}$$

True **N**egative	**F**alse **P**ositive
False **N**egative	**T**rue **P**ositive

4	**2**
1	**3**

$$Acc = \frac{7}{10} = 0.7$$

Additional Measures

True Positive Rate (TPR) / Recall – the ratio of the number of correctly predicted positive class to the total number of positive classes.

$$\frac{TP}{TP + FN}$$

Precision – *"from all the points predicted positive how many of them are actually positive?"*

$$\frac{TP}{TP + FP}$$

Additional Measures

F-β Score -For ML models where both FN and FP have equal importance to be low, then we can combine the advantage of **Precision** and **Recall** in a new metric called F-β Score.

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

If $\beta < 1$ is used when FP have more impact than FN,

Else If $\beta > 1$ is used when FN have more impact than FP,

Else ($\beta = 1$) is used when FN and FP have equal importance

Unsupervised Learning

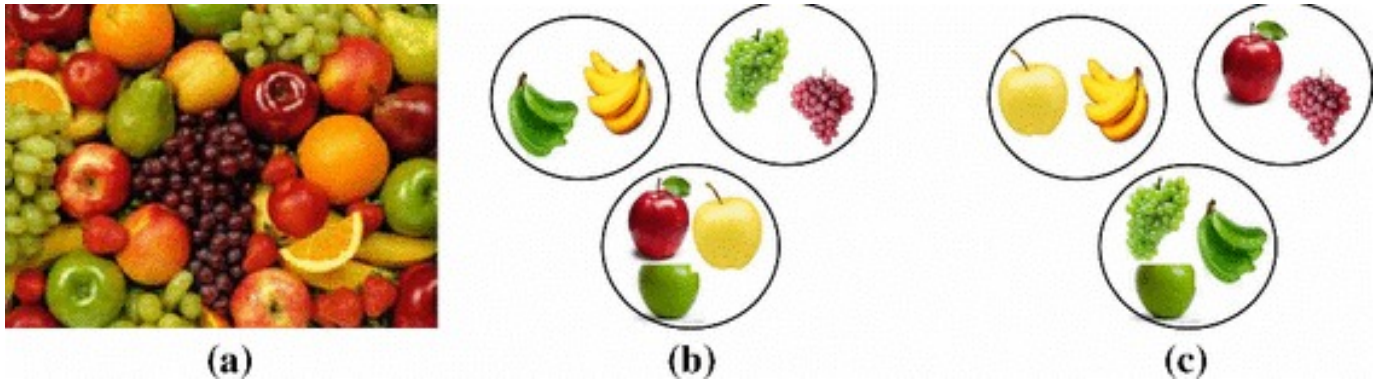
clustering

Definition of Clustering

Definition:

Given a database $D = \{t_1, t_2, \ldots\ldots, t_n\}$ of n tuples, the clustering problem is to define a mapping $f : \mathrm{D} \rightarrow C$, where each $t_i \in D$ is assigned to one cluster $c_i \in C$.

Here, $C = \{c_1, c_2, \ldots\ldots, c_k\}$ denotes a set of clusters.



Clustering - Problems

1 - The (best) number of cluster is not known – NP-hard.

- There is not correct answer to a clustering problem.
- In fact, many answers may be found.
- The exact number of cluster required is not easy to determine.

Clustering - Problems

2. There may not be any a priori knowledge concerning the clusters.

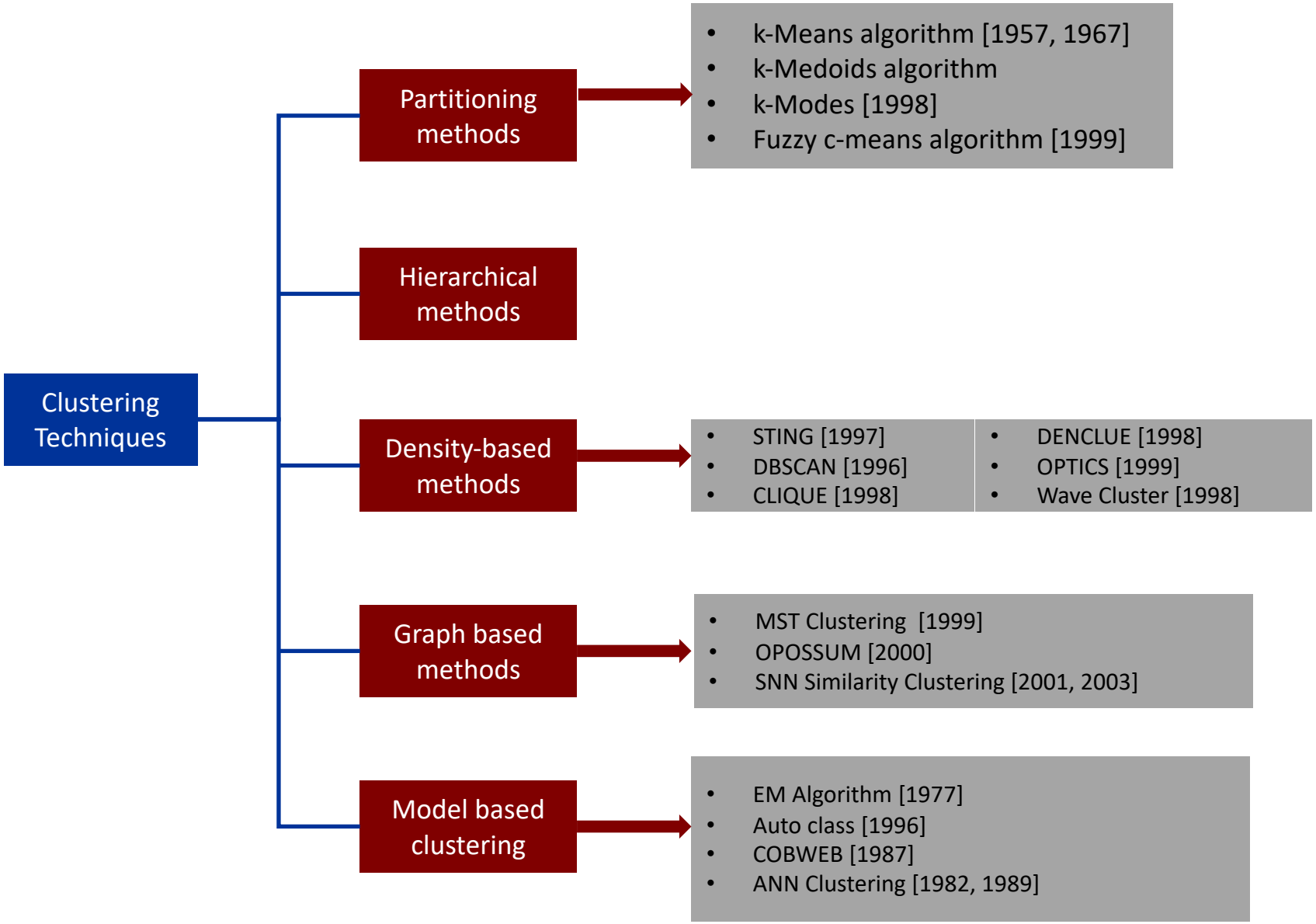
- This is an issue that what data should be used for clustering.
- We have not supervisory learning to aid the process.
- Clustering can be viewed as similar to unsupervised learning.

Clustering - Problems

3. Interpreting the semantic meaning of each cluster may be difficult.

- With classification, the labeling of classes is known ahead of time.
- In contrast, with clustering, this may not be the case.
- Thus, when the clustering process is finished yielding a set of clusters,

the exact meaning of each cluster may not be obvious.


Clustering Techniques
Partitioning methods
k-Means algorithm [1957, 1967]
k-Medoids algorithm
k-Modes [1998]
Fuzzy c-means algorithm [1999]
Hierarchical methods
Density-based methods
STING [1997]
DBSCAN [1996]
CLIQUE [1998]
DENCLUE [1998]
OPTICS [1999]
Wave Cluster [1998]
Graph based methods
MST Clustering [1999]
OPOSSUM [2000]
SNN Similarity Clustering [2001, 2003]
Model based clustering
EM Algorithm [1977]
Auto class [1996]
COBWEB [1987]
ANN Clustering [1982, 1989]

K-Means Algorithm

Notations:

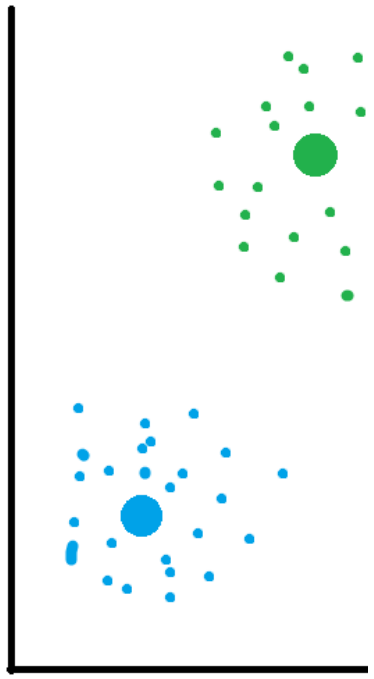
- x : an object under clustering
- n : number of objects under clustering
- $\boldsymbol{C}_i$: the *i-th* cluster
- c_i : the centroid of cluster $\boldsymbol{C}_i$
- n_i : number of objects in the cluster $\boldsymbol{C}_i$
- k : number of clusters

K-Means Algorithm

The goal is to create k clusters such that items in the same cluster are similar to each other. For that, we need to define our cost function.

For $x \in C_i$ the "error" is $(C_i - x)$. Error/distance can not be negative, then $(C_i - x)^2$. We want it for each x:

$$\sum_{x \in C_i} (c_i - x)^2$$

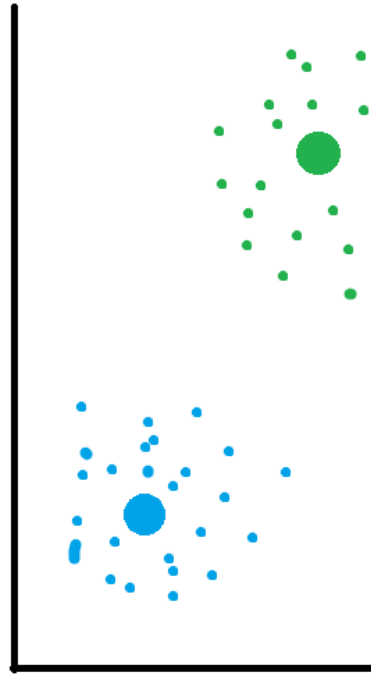


K-Means Algorithm

$$\sum_{x \in C_i} (c_i - x)^2$$

We have total of k clusters, so:

$$\sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2$$



is the cost function ☺ called **Sum of Squared Error (SSE).**

K-Means Algorithm

The goal is to create k clusters such that items in the same cluster are similar to each other. The sum-of-squared error function of a clustering defined as (L_2 norm):

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

If we want to minimize the SSE , what should we do ??

K-Means Algorithm

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

To minimize SSE: $\frac{\partial (SSE)}{\partial c_i} = 0$

$$\frac{\partial}{\partial c_i} \left(\sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2 \right) = 0$$

Or,

$$\sum_{i=1}^{k} \sum_{x \in \boldsymbol{C}_i} \frac{\partial}{\partial c_i} (c_i - x)^2 = 0$$

K-Means Algorithm

$$\sum_{x \in \boldsymbol{C}_i} 2(c_i - x) = 0 \Leftrightarrow 2 \sum_{x \in \boldsymbol{C}_i} (c_i - x) = 0$$

$$\sum_{x \in \boldsymbol{C}_i} c_i - \sum_{x \in \boldsymbol{C}_i} x = 0$$

$$\sum_{x \in \boldsymbol{C}_i} c_i = \sum_{x \in \boldsymbol{C}_i} x$$

$$n_i \cdot c_i = \sum_{x \in \boldsymbol{C}_i} x$$

$$c_i = \frac{1}{n_i} \sum_{x \in \boldsymbol{C}_i} x$$

The best centroid for minimizing SSE is the mean of the objects in the cluster.

K-Means Algorithm

The algorithm procedure:

- Selects *k* number of objects at random from the set of n objects.
- These *k* objects are treated as the centroids of *k* clusters.
- For each of the remaining objects, it is assigned to one of the closest centroid.
- The centroid of each cluster is then updated by calculating the mean values of attributes of each object.
- Repeat until convergence

K-Means Algorithm – Visualization

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

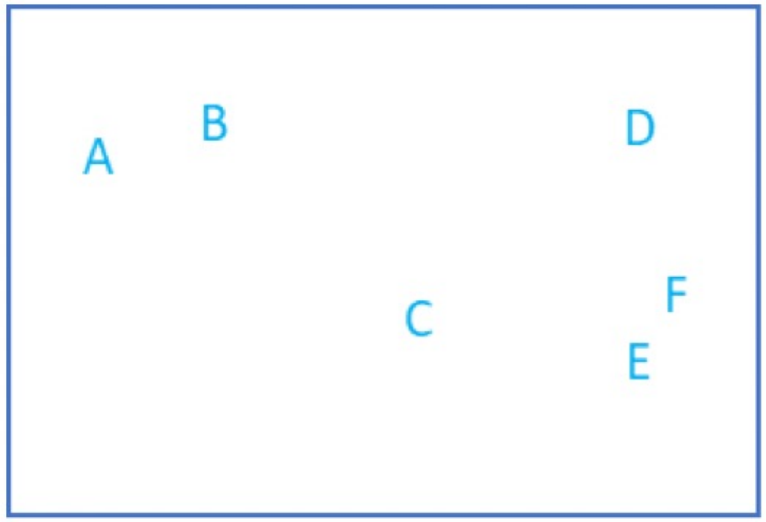
The value of “K”

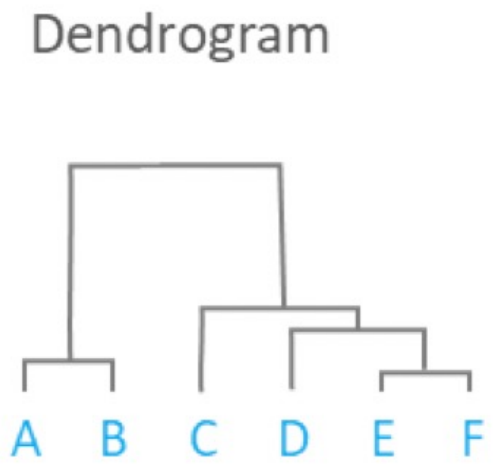
- The k-means algorithm produces only one set of clusters, for which, user must specify the desired number, *k* of clusters.
- In fact, *k* should be the best guess on the number of clusters present in the given data.
- Choosing the best value of *k* for a given dataset is NP-Hard problem.
- There is no principled way to know what the value of *k* ought to be. We may try with successive value of k starting with 2.

Hierarchical Clustering

clustering approach via partitioning data set sequentially

- Construct nested partitions layer by layer via grouping objects into a tree of clusters (without the need to know the number of clusters in advance)
- Use (generalised) distance matrix as clustering criteria





Agglomerative vs. Divisive

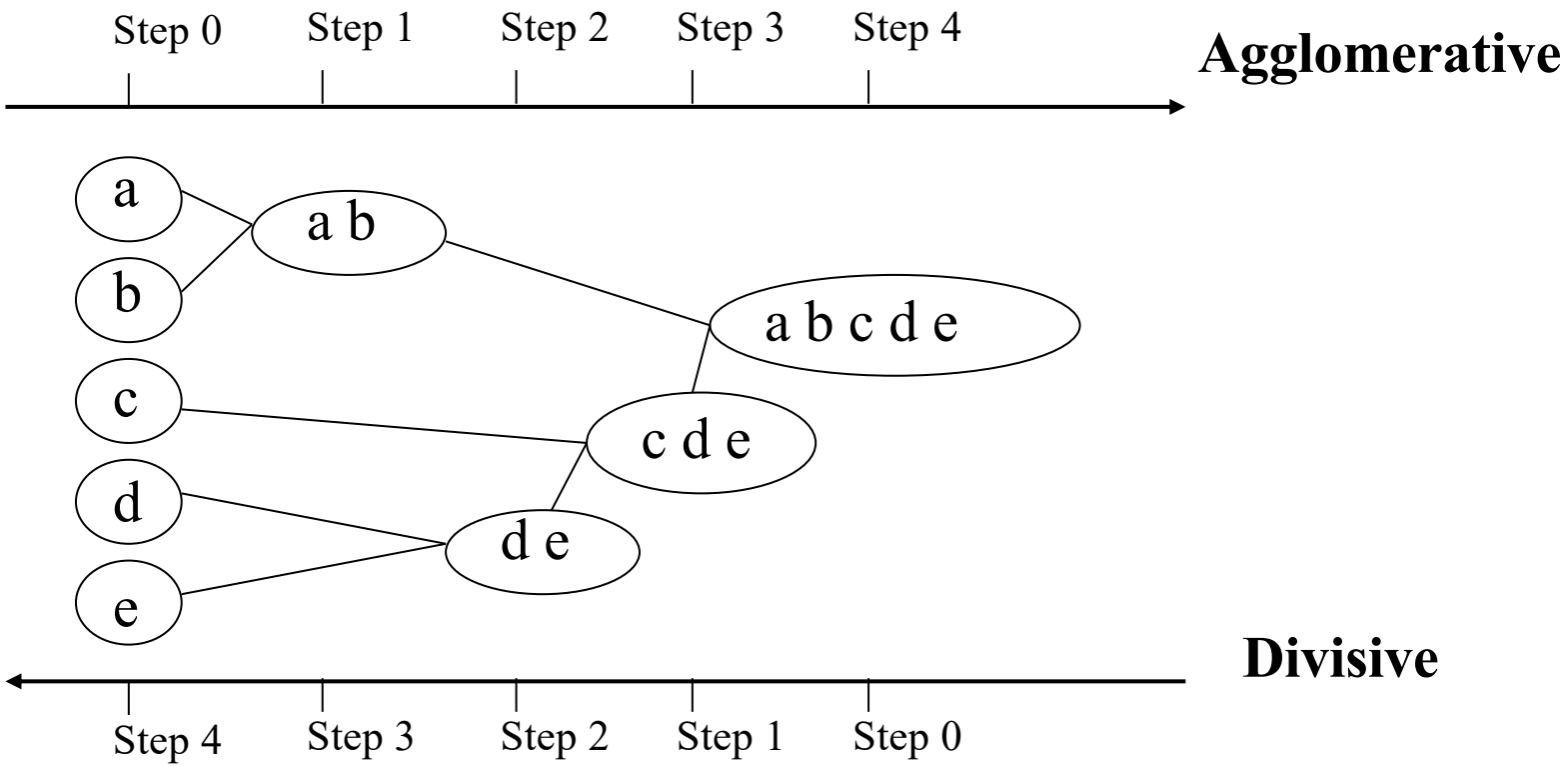
Agglomerative: a bottom-up strategy

- Initially each data object is in its own (atomic) cluster
- Then merge these atomic clusters into larger and larger clusters

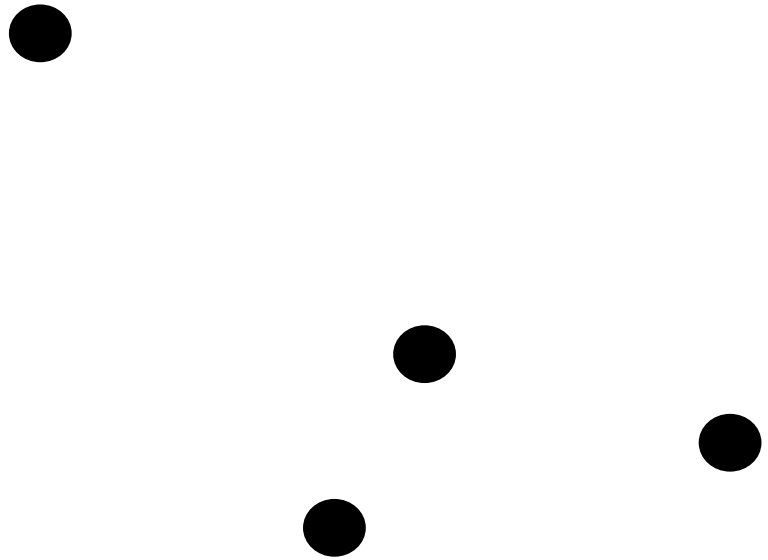
Divisive: a top-down strategy

- Initially all objects are in one single cluster
- Then the cluster is subdivided into smaller and smaller clusters

Agglomerative vs. Divisive

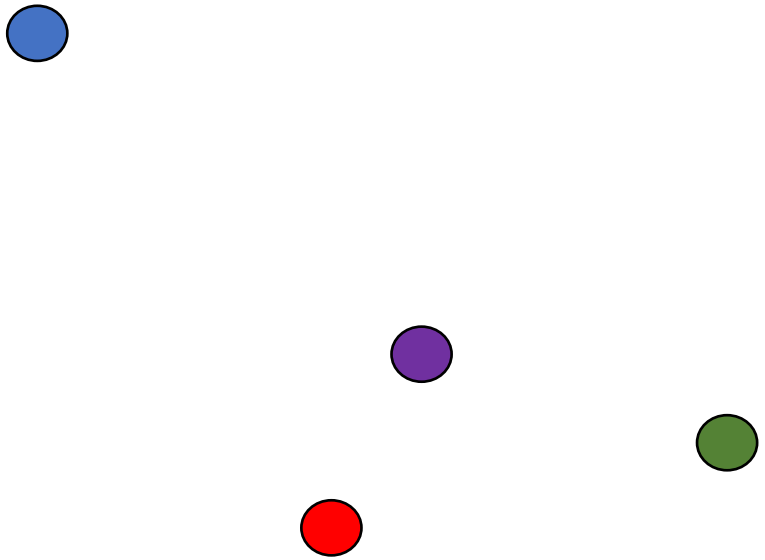


Hierarchical Clustering - Example



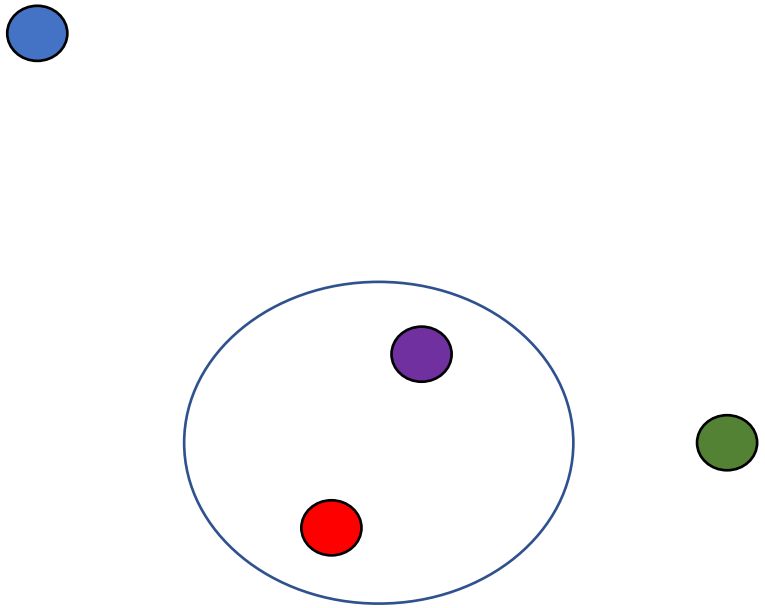
Hierarchical Clustering - Example

Step 1: assign each point to a separate cluster



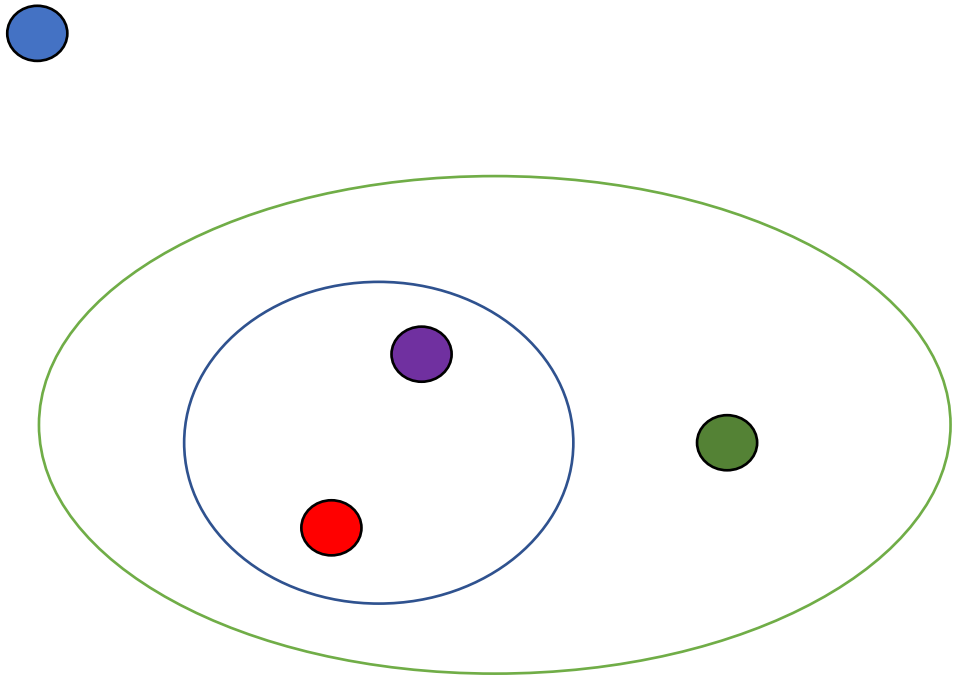
Hierarchical Clustering - Example

Then, at each iteration, we merge the closest pair of clusters and repeat this step until only a single cluster is left:



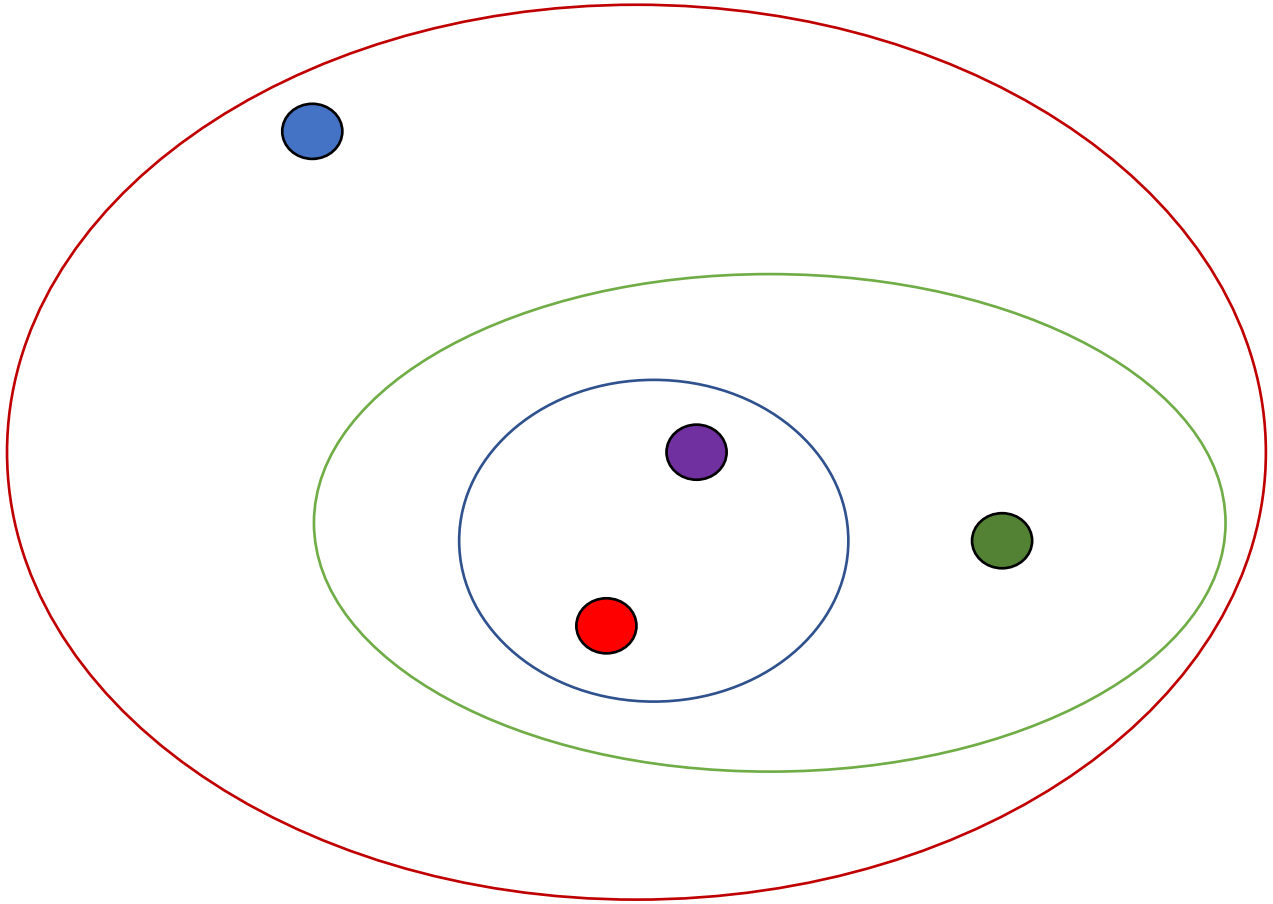
Hierarchical Clustering - Example

Then, at each iteration, we merge the closest pair of clusters and repeat this step until only a single cluster is left:



Hierarchical Clustering - Example

Then, at each iteration, we merge the closest pair of clusters and repeat this step until only a single cluster is left:

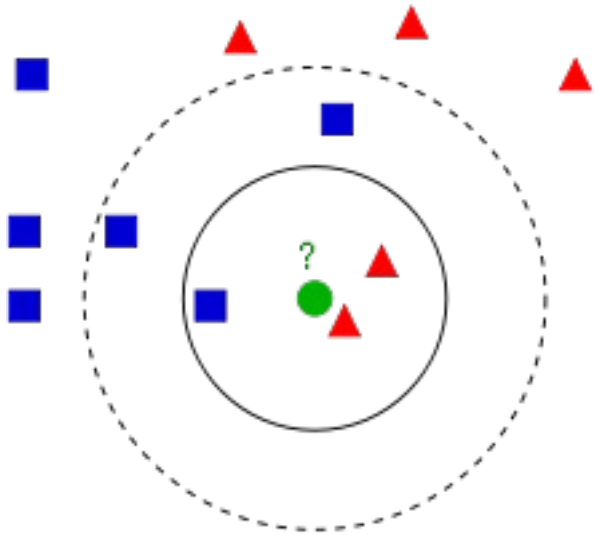


Supervised Learning

KNN

Introduction

K-Nearest Neighbors is one of the simplest ML algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

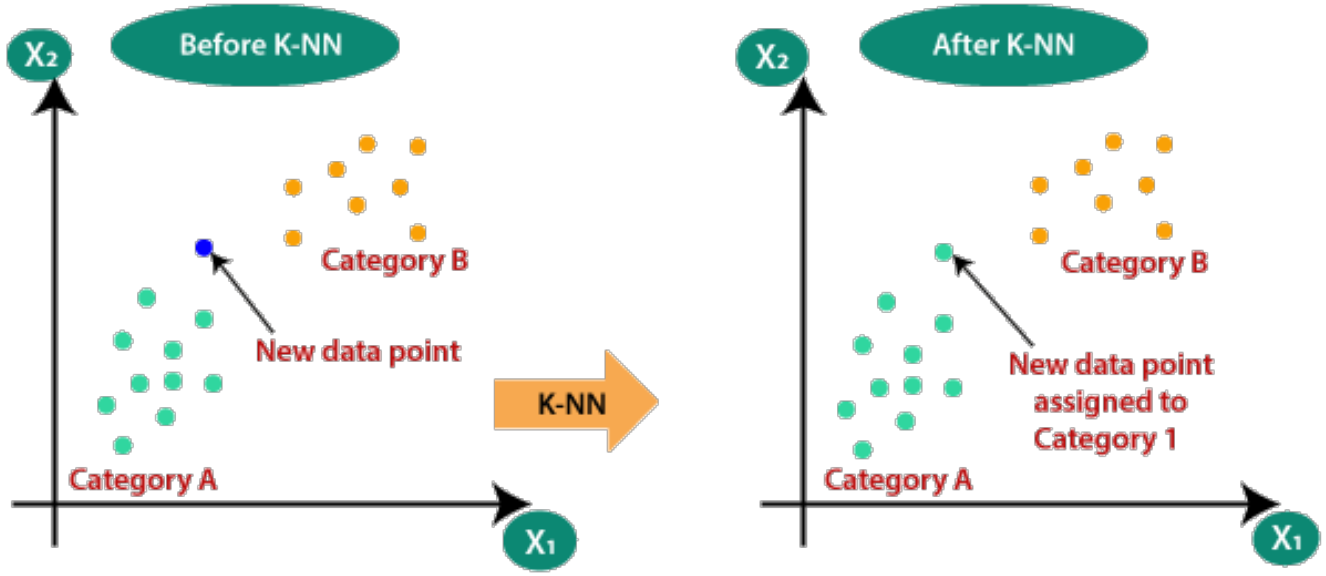


Introduction

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

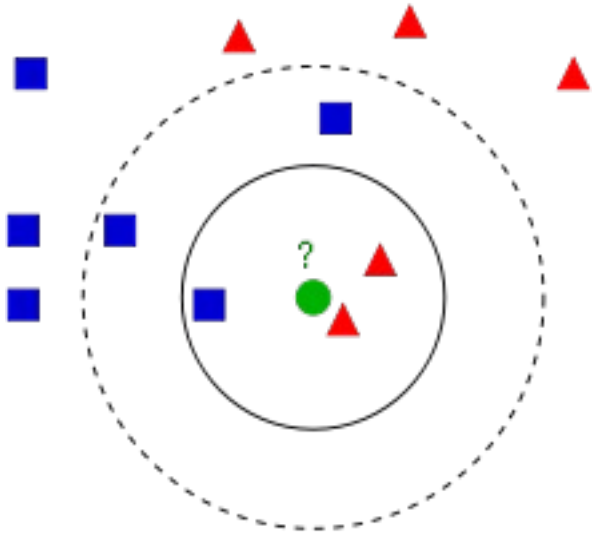
Why to use KNN?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.



How Does it Work?

1. Given a dataset D and a new observation x, choose a distance metric
2. Find the distance between x and all $x_j \in D$
3. Find the top k nearest neighbors
4. Use **voting** and classify the new observation



Recommended: use odd value of K to avoid equality in voting

How to Select the value of K?

1. There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
2. A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
3. It is more than recommended to use odd value of K to avoid equality in voting.

Weighted K-Nearest Neighbors (KNN)

Weighted K-NN: Matrix Approach

Generally, let X, Y be vectors in $\mathbb{R}^n$, and let $D(X, Y)$ be the weighted distance between X, Y.

Define M a diagonal matrix of size $n\ x\ n$ where M_{ii} is the weight of each feature

$$D(X, Y) = \sqrt{(X - Y) \cdot M \cdot (X - Y)^T}$$

$$M = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & w_n \end{bmatrix}$$

Weighted K-NN - Example

Given $X = (6,1,5,5)$ and $Y = (3,7,1,2)$ vectors in $\mathbb{R}^4$.

The weights are $w_1 = 3, w_2 = 2, w_3 = 1, w_4 = 7$ then:

$$X - Y = [3, -6,4,3], \qquad M = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

$$D(X,Y) = \sqrt{(X-Y) \cdot M \cdot (X-Y)^T}$$

Weighted K-NN - Example

$$D(X,Y) = \sqrt{(X-Y)\cdot M\cdot (X-Y)^T} =$$

$$= \sqrt{(3,-6,4,3)\cdot \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -6 \\ 4 \\ 3 \end{bmatrix}} =$$

$$= \sqrt{(9,-12,4,21)\cdot \begin{bmatrix} 3 \\ -6 \\ 4 \\ 3 \end{bmatrix}} = \sqrt{178}$$

Weighted KNN – Example

Customer	Credit Level	Bank Insurance	Y
C_1	7	7	BAD
C_2	7	4	BAD
C_3	3	4	GOOD
C_4	1	4	GOOD
C_5	3	7	?

Given the weights $3,1$ for Credit and Bank respectively, use $K = 2$ to determine the classification of C_5

Weighted KNN – Example

Calculate weighted distances –

$$M = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C_5 - C_1 = (-4,0) \Rightarrow \sqrt{(-4,0) \cdot \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -4 \\ 0 \end{bmatrix}} = \sqrt{48}$$

$$C_5 - C_2 = (-4,3) \Rightarrow \sqrt{(-4,3) \cdot \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -4 \\ 3 \end{bmatrix}} = \sqrt{57}$$

$$C_5 - C_3 = (0,3) \Rightarrow \sqrt{(0,3) \cdot \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 3 \end{bmatrix}} = \sqrt{9}$$

$$C_5 - C_4 = (2,3) \Rightarrow \sqrt{(2,3) \cdot \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 3 \end{bmatrix}} = \sqrt{21}$$

Weighted KNN – Example

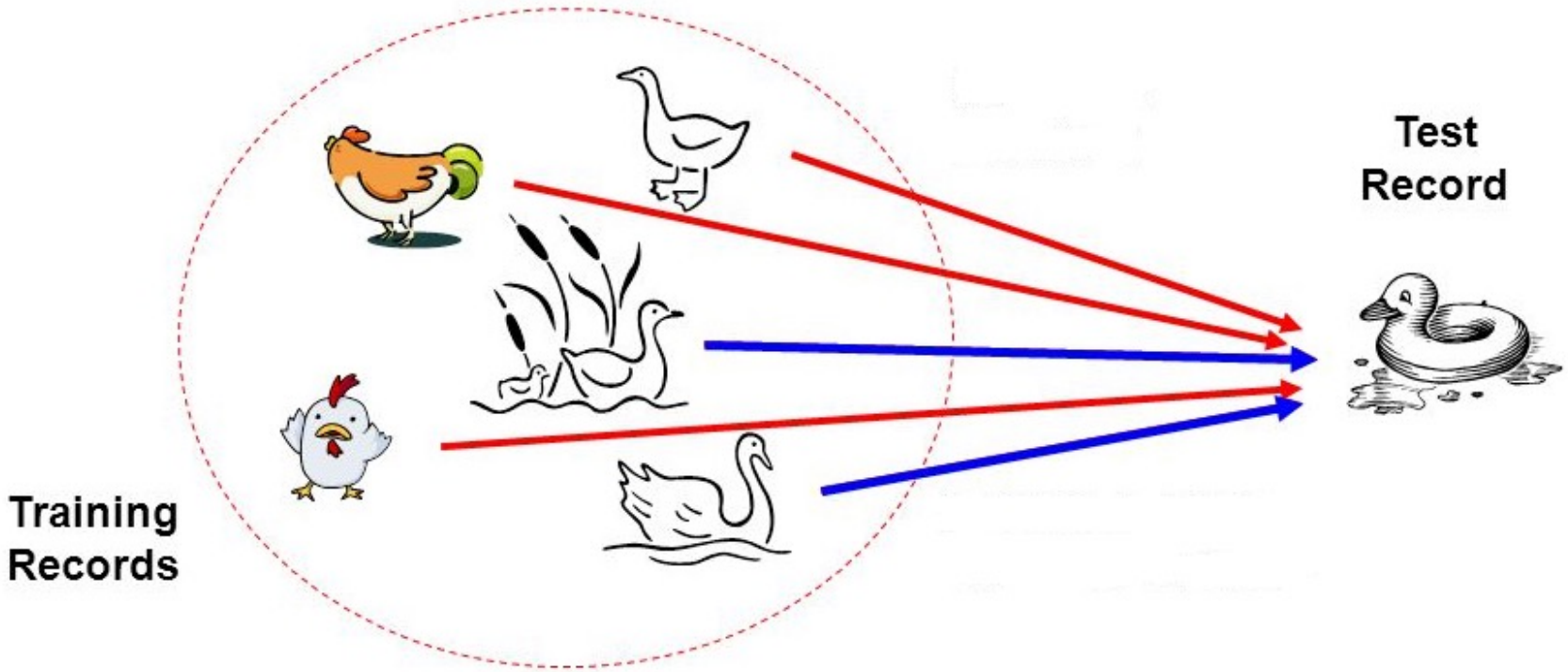
Customer	Credit Level	Bank Insurance	Y
C_1	7	7	BAD
C_2	7	4	BAD
C_3	3	4	GOOD
C_4	1	4	GOOD
C_5	3	7	GOOD

Supervised Learning

Naïve Bayes

Bayesian Classifier

Principle: If it walks like a duck, quacks like a duck, then it is probably a duck



Bayesian Classifier

- A statistical classifier - Performs *probabilistic prediction*, *i.e.*, predicts class membership probabilities
- Based on Bayes' Theorem.
- Assumption- the attributes are independent given the class.

Naïve Bayes Classifier – Algorithm Outline

- **Step 1**: Calculate the prior probability for given class labels.
- **Step 2**: Find Likelihood probability with each attribute for each class.
- **Step 3**: Put these value in Bayes Formula and calculate posterior probability

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- **Step 4**: See which class has a higher probability, given the input belongs to the higher probability class.

Naïve Bayes Classifier – Algorithm

Consider the following dataset:

Weather	Temp	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

Question: What is the probability of playing when the weather is overcast?

Naïve Bayes Classifier – Algorithm

Question: What is the probability of playing when the weather is overcast?

$$P(Yes|Overcast) = \frac{P(Overcast|Yes) \cdot P(Yes)}{P(Overcast)}$$

1. $P(Overcast) = 0.29,\ P(Yes) = 0.64$
2. $P(Overcast|Yes) = 0.44$
3. $P(Yes|Overcast) = \frac{0.44 \cdot 0.64}{0.29} = 0.98$
4. Similarly, you can calculate the probability of not playing

Pros and Cons

The Naïve Bayes' approach is a very popular one, which often works well.

However, it has a number of potential problems

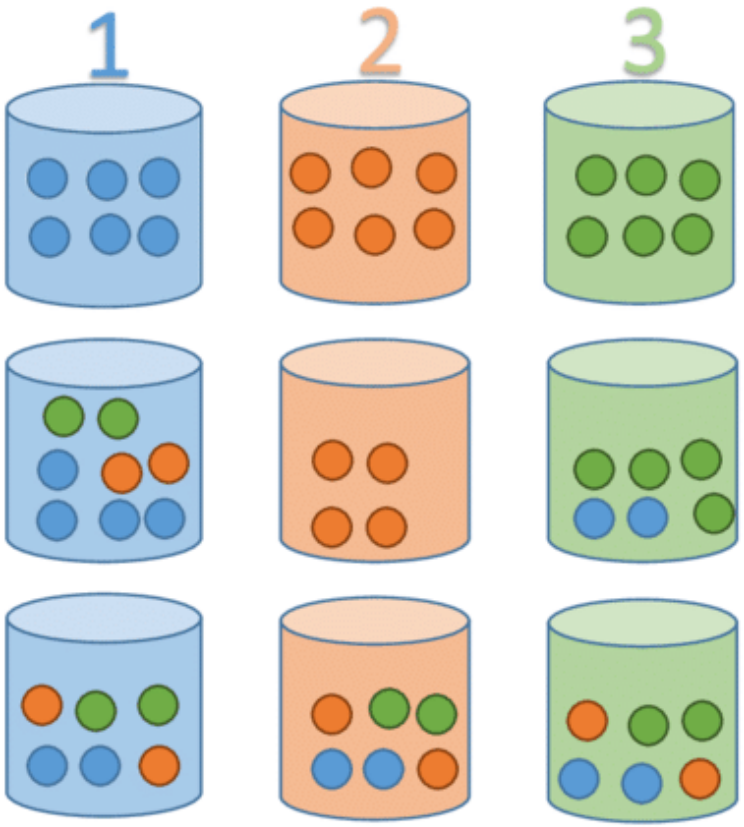
- It relies on all attributes being categorical.
  - Why is it bad for continuous attributes?
- If the data is less, then it estimates poorly.

Supervised Learning

Decision Tree

How do we define information?

The Information within a source is the uncertainty of the source.



Zero uncertainty

Medium uncertainty

High uncertainty

General Case

Suppose X can have one of m values… $V_1, V_2, \ \ldots \ V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution?

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m = -\sum_{j=1}^{m} p_j \log_2 p_j$$

H(X) = The **entropy** of X

Entropy and its Meaning

Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.

With the growth of Information Technology,

entropy becomes an important concept in Information Theory

Entropy is the level of uncertainty.

“High Entropy” means X is from a uniform (boring) distribution

“Low Entropy” means X is from varied (peaks and valleys) distribution

Measure of Information Content

- Does sun rises in the East? (answer is with 0 uncertainty)
- Will mother give birth to male baby? (answer is with ½ uncertainty)
- Is there a planet like earth in the galaxy? (answer is with an extreme uncertainty)

Entropy - Example

Consider a dataset with 1 blue ball, 2 greens and 3 reds.

$$P_{blue} = \frac{1}{6}$$

$$P_{green} = \frac{2}{6} = \frac{1}{3}$$

$$P_{red} = \frac{3}{6} = \frac{1}{2}$$

$$H(D) = -\left[\frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{2}\log\frac{1}{2}\right] = 1.46$$

Algorithm ID3

- In ID3, entropy is used to measure how informative a node is.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
- The attribute with the largest value of information gain is chosen as the splitting attribute and
- It partitions into a number of smaller training sets based on the distinct values of attribute under split.

Defining Information Gain

Definitions:

- D - the training set
- $H(D)$ - the entropy of the training set D

$$H = -\sum_{j} p_j \cdot \log_2(p_j)$$

- The training set D has $c_1, c_2, \dots, c_k$ - distinct classes
- $p_i = \frac{|C_{i,D}|}{|D|}$ where $C_{i,D}$ is the set of tuples of class c_i in D.

Weighted Entropy

$$H_A(D)=\sum_{j=1}^{m}\frac{|D_j|}{|D|}H(D_j)$$

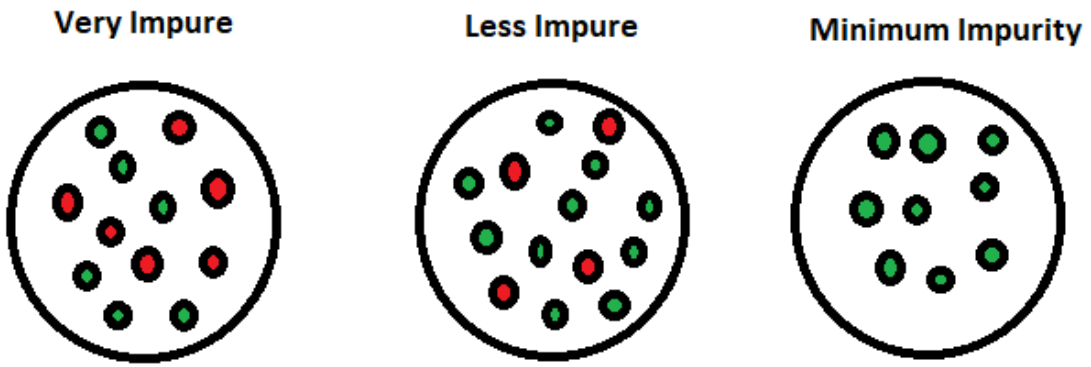
- The term $\frac{|D_j|}{|D|}$ denotes the weight of the *j*-th training set.
- More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from *D* based on the splitting of *A*.

Defining Information Gain

Information gain, $\alpha(A, D)$ of the training set *D* splitting on the attribute *A* is given by

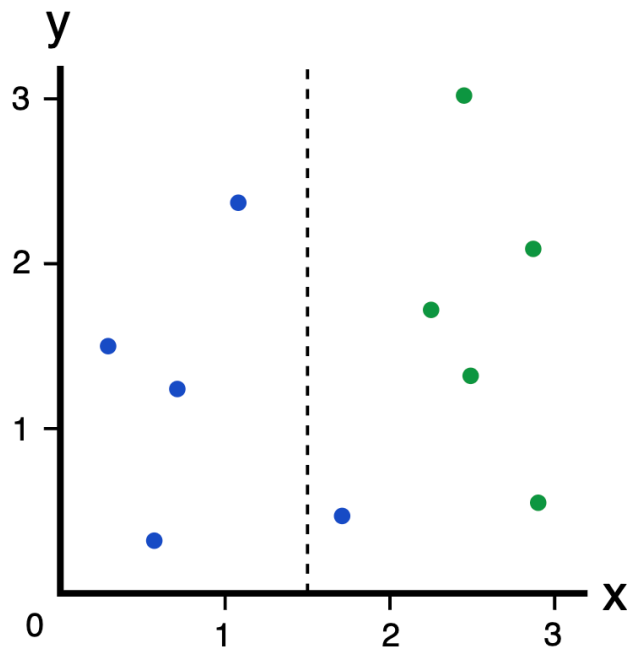
$$\alpha(A, D) = H(D) - H_A(D)$$

In other words, $\alpha(A, D)$ gives us an estimation how much would be gained by splitting on *A*. The attribute *A* with the highest value of α should be chosen as the splitting attribute for *D*.



Information Gain - Example

Consider the following graph.



Before the split, we had 5 blues and 5 greens, total of:

$$H = -[0.5 \log 0.5 + 0.5 \log 0.5] = 1$$

After the split, we get:

Left – all dataset is blue, means $H_{left} = 0$

Right - $H_{right} = -\left[\frac{1}{6}\log\frac{1}{6} + \frac{5}{6}\log\frac{5}{6}\right] = 0.65$

Information Gain - Example

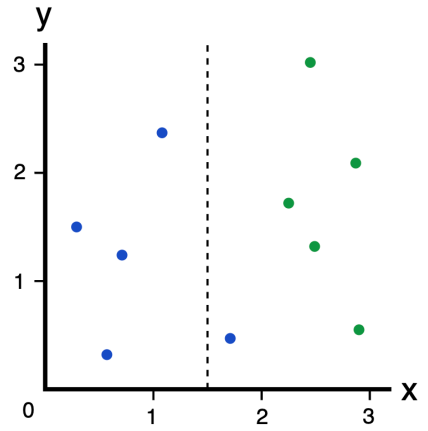
Now that we have the entropies for both branches, we can determine the quality of the split by **weighting the entropy of each branch by how many elements it has**. Since Left Branch has 4 elements and Right Branch has 6:

$$H_{split} = 0.4 \cdot 0 + 0.6 \cdot 0.65 = 0.39$$

Information Gain = how much Entropy we removed

$$\alpha = 1 - 0.39 = 0.61$$

This makes sense: **higher Information Gain = more Entropy removed**



Information Gain - Example

Information Gain = how much Entropy we removed

$$\alpha = 1 - 0.39 = 0.61$$

This makes sense: **higher Information Gain = more Entropy removed**, which is what we want. In the perfect case, each branch would contain only one color after the split, which would be zero entropy!