DATA MINING LECTURE 2

Data Preprocessing

Data Preprocessing

- Data Preprocessing: An Overview
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction
- Summary

2

Why Preprocess Data?

- Welcome to the Real World! *Data is Dirty!*
- But, No quality data, no quality mining results!
- Preprocessing is one of the most critical steps in a data mining process

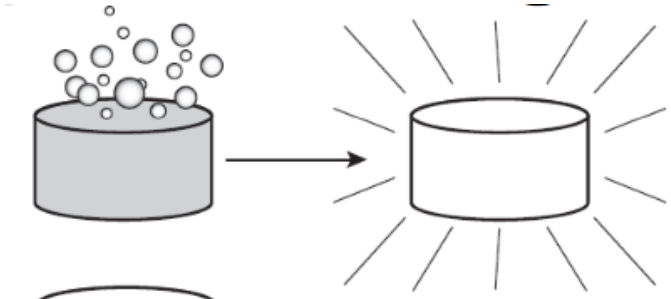
Why Preprocess Data? Contd.

- Measures for data quality: A multidimensional view
  - Accuracy: correct or wrong, accurate or not
  - Completeness: not recorded, unavailable, …
  - Consistency: some modified but some not, dangling, …
  - Timeliness: timely update?
  - Believability: how trustable the data are correct?
  - Interpretability: how easily the data can be understood?

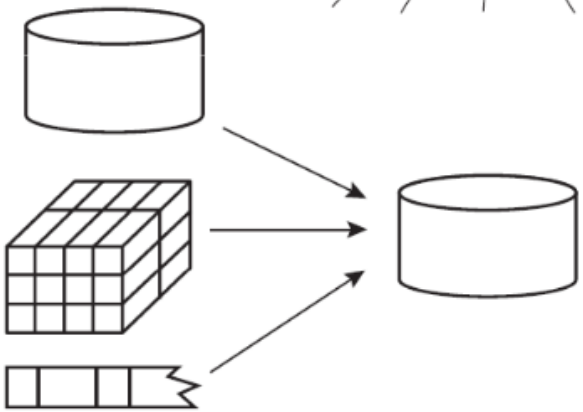
Data Preprocessing: Major Tasks

5

Data cleaning



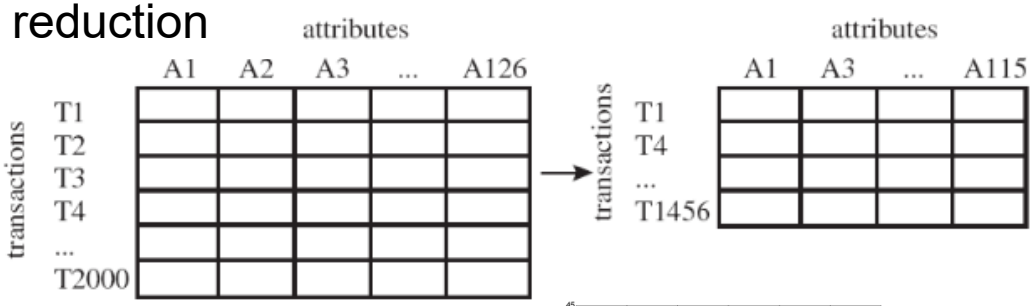
Data Integration



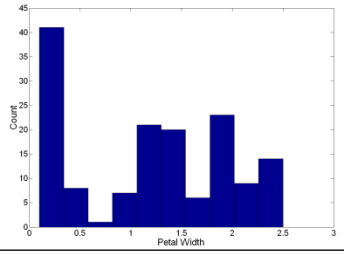
Data transformation

$-2, 32, 100, 59, 48 \longrightarrow -0.02, 0.32, 1.00, 0.59, 0.48$

Data reduction



Data Exploratory Analysis

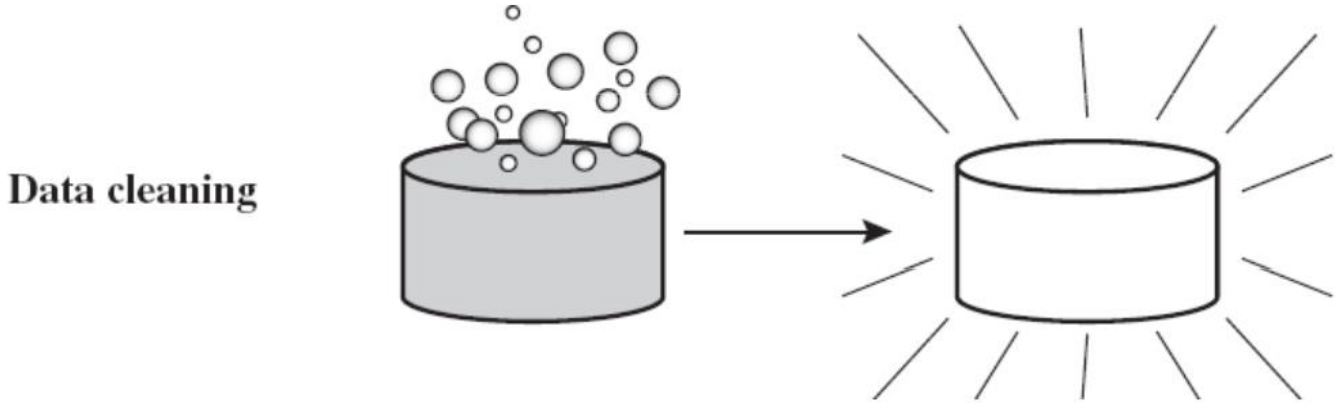


Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction
- Summary

7

Data Cleaning



Why Data Cleaning?

- **Data in the real world is dirty. 'Dirty' denotes-**
  - Incomplete/missing: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=" "
  - noisy: containing errors or outliers
    - e.g., Salary="-10"
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age="42" Birthday="03/07/1997"
    - e.g., Was rating "1,2,3", now rating "A, B, C"
    - e.g., discrepancy between **duplicate** records
  - Redundant: including everything, some of which are irrelevant to our task
  - Duplicate: Data-entry is duplicated.

Dirty Data

- Examples of data quality problems:
  - Noise and outliers
  - Missing values
  - Duplicate data

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	10000K	Yes
6	No	NULL	60K	No
7	Yes	Divorced	220K	NULL
8	No	Single	85K	Yes
9	No	Married	90K	No
9	No	Single	90K	No

A mistake or a millionaire?

Missing values

Inconsistent duplicate entries

Data Cleaning

- Data cleaning tasks
  - Fill in missing values
  - Identify outliers/noise and smooth out noisy data
  - Correct inconsistent data
  - Remove duplicate/redundant data

Missing Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- <u>Missing data may need to be inferred</u>

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the task is classification—not effective in certain cases)
- Fill in the missing value manually: tedious + infeasible?
- Use a global constant to fill in the missing value: e.g., "unknown", a new class?!
- Use the attribute mean to fill in the missing value
- Use the attribute mean for all samples of the same class to fill in the missing value: smarter
- Use the most probable value to fill in the missing value: inference-based such as regression, Bayesian formula, decision tree

Noisy Data

- What is noise?

  - Random error in a measured variable.

- Incorrect attribute values may be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention

How to Handle Noisy Data?

- Binning/bucketing
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Simple Discretization Methods: Binning

- Equal-width (distance) partitioning:
  - It divides the range into N intervals of equal size: uniform grid
  - if A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B-A)/N$.

  - Advantage/disadvantage
    - The most straightforward
    - But outliers may dominate presentation
    - Skewed data is not handled well.

- Equal-depth (frequency) partitioning:
  - It divides the range into N intervals, each containing approximately same number of samples

  - Advantage/disadvantage
    - Good data scaling
    - Managing categorical attributes can be tricky.

Example

Equal frequency:

```
Input:[5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

Output:
[5, 10, 11, 13]
[15, 35, 50, 55]
[72, 92, 204, 215]
```

Equal Width:

width of intervals will be: $W = (B-A)/N$

```
Input: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

Output:
[5, 10, 11, 13, 15, 35, 50, 55, 72]
[92]
[204, 215]
```

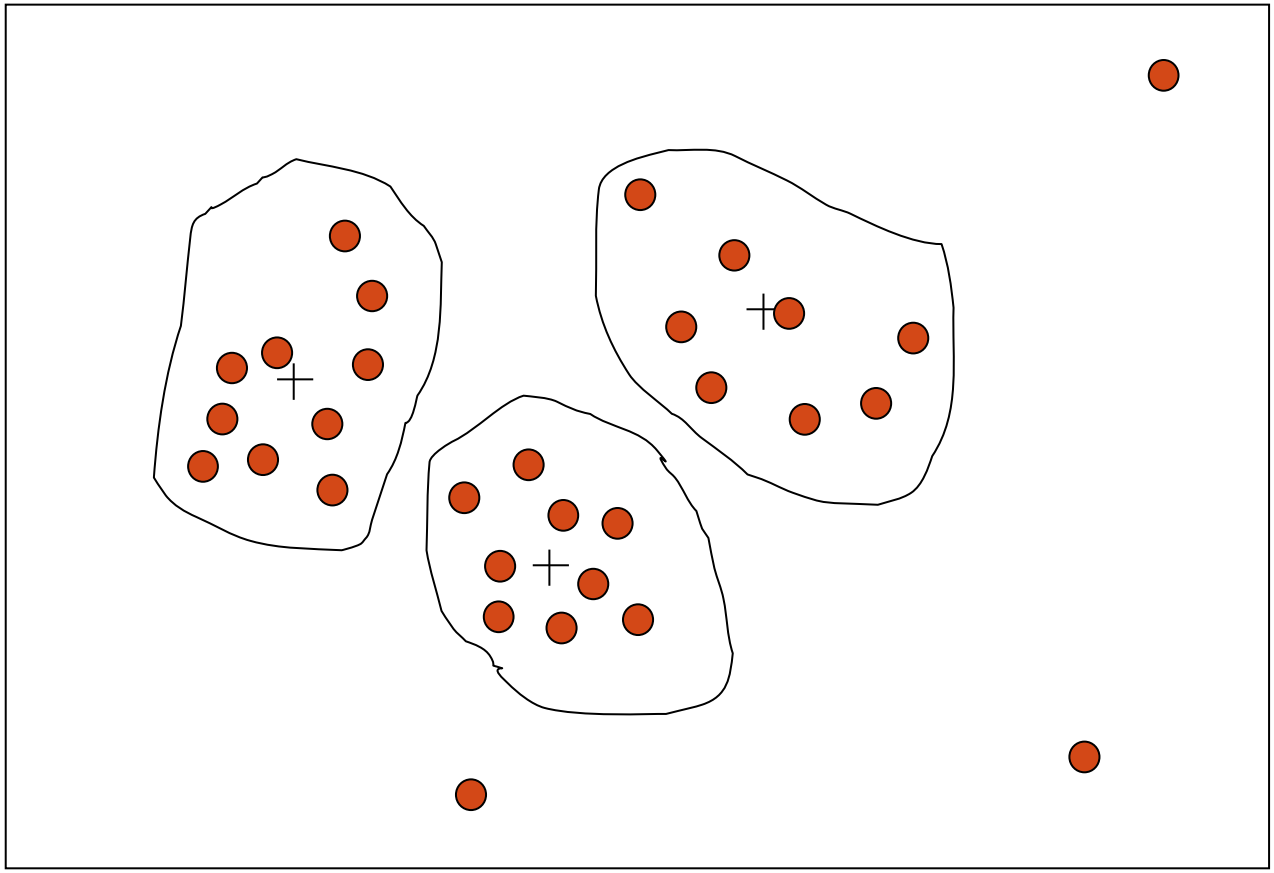
N=3
W= (215-5)/3
= 75

[5~75]
[76~150]
[151~215]

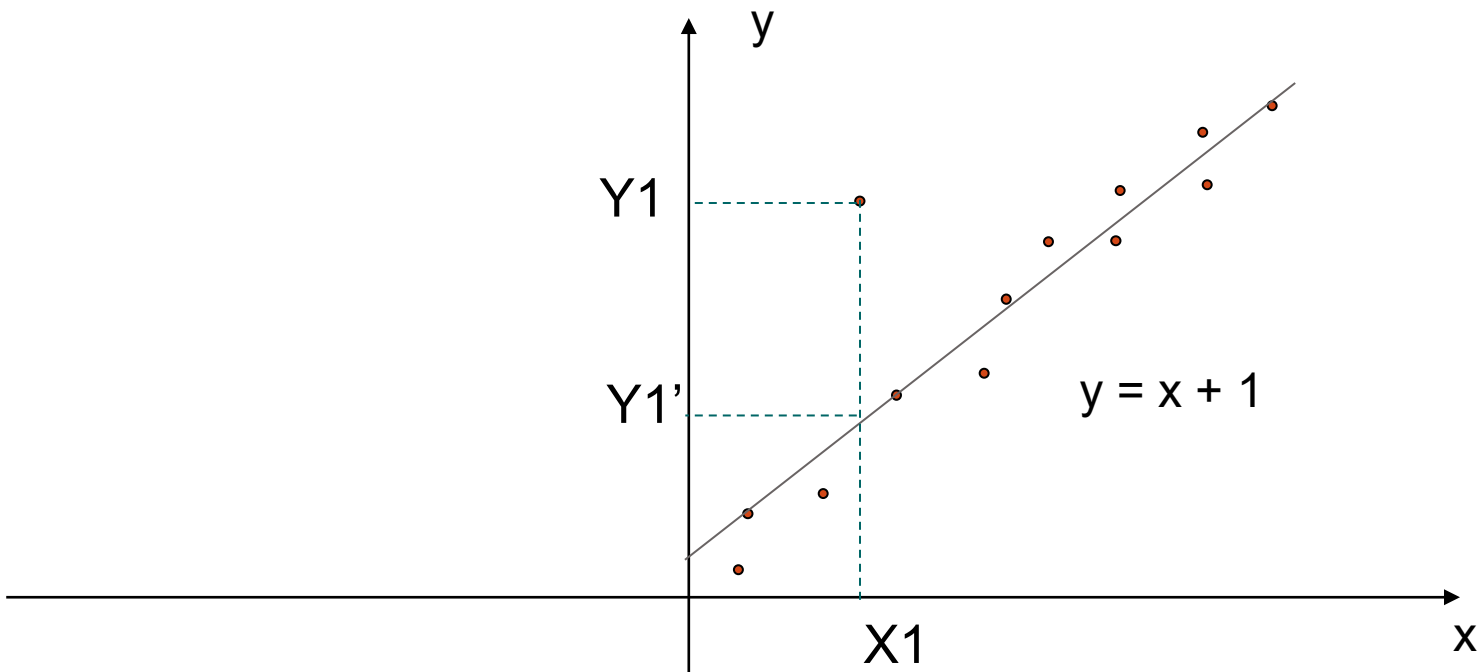
Equal depth/frequency binning: An example

* Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
* Partition into (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Cluster Analysis



Regression



Note

- Linear regression (best line to fit two variables)
- Multiple linear regression (more than two variables, fit to a multidimensional surface

How to Handle Inconsistent Data?

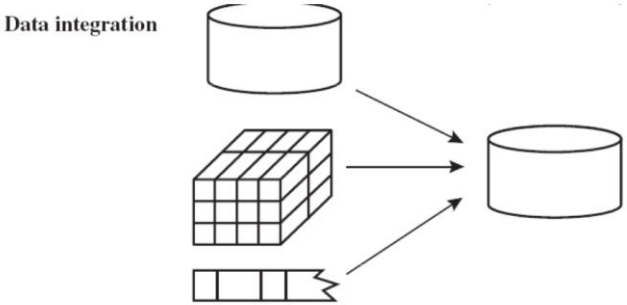
- Inconsistent data may come from
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)

- Handling Inconsistent Data-
  - Manual correction using external references
  - Semi-automatic using various tools
    - To detect violation of known functional dependencies and data constraints
    - To correct redundant data

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction
- Summary

Data Integration



Data Integration

- **Data integration**:
  - Combines data from multiple sources into a coherent store
- Schema integration and object matching
  - Entity identification problem:
    - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
    - Value conflict, e.g., good height = tall; poor height = short
- Why such problem arises?
  - For the same real world entity, attribute values from different sources are different
  - Possible reasons: different representations, different scales, e.g., metric vs. British units

Schema integration and object matching

- *custom_id* and *cust_number*
  - Schema conflict

- "Yes" and "No", and '1' and '0' for an attribute
  - Value conflict

- Solutions
  - meta data (data about data)

Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

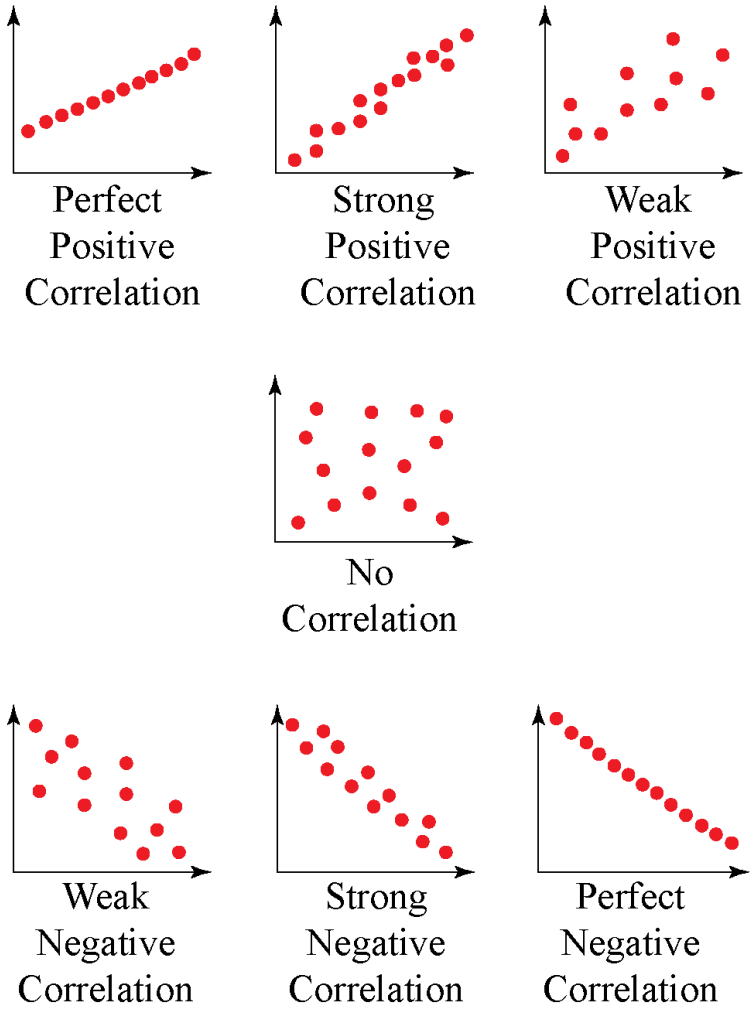
Detecting Redundancy

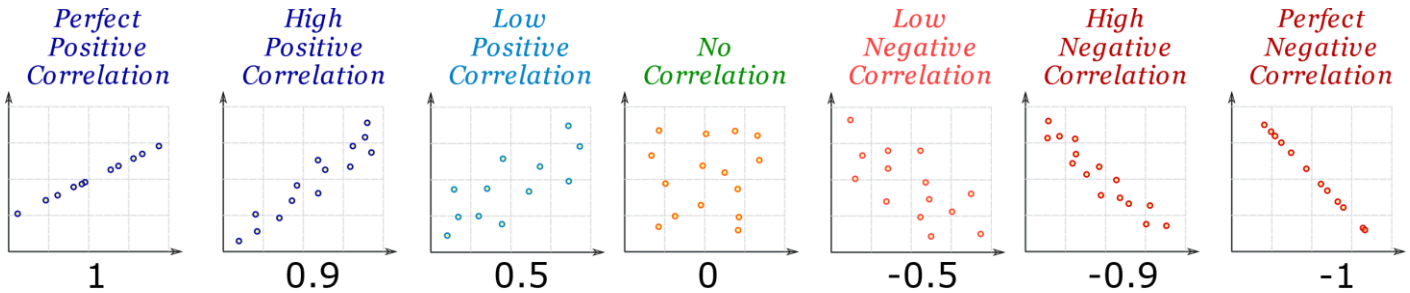
- If an attributed can be “derived” from another attribute or a set of attributes, it may be redundant.

- Some redundancies can be detected by correlation analysis, e.g.,
  – **Correlation analysis**
  – Chi-square test

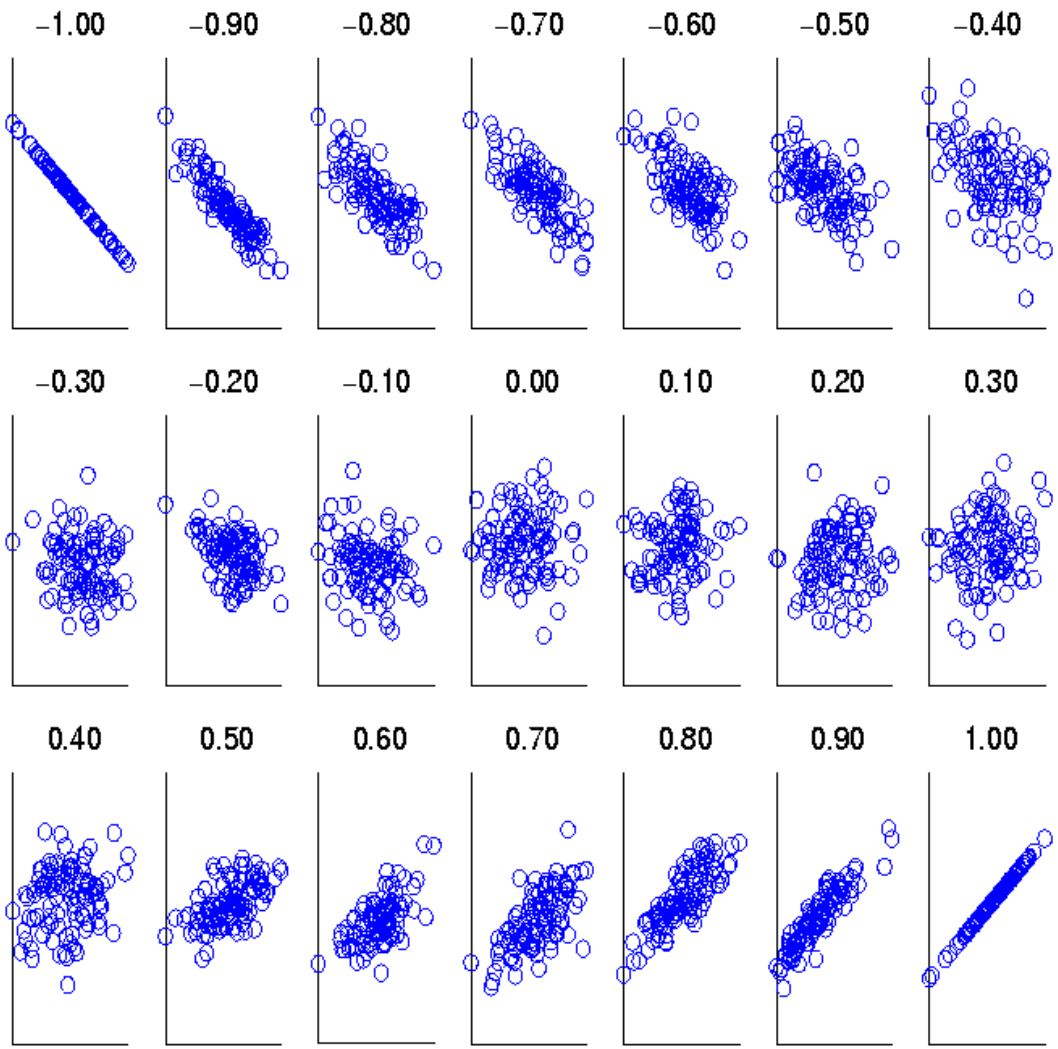
Correlation Analysis

- The correlation coefficient is a value that indicates the strength of the relationship between variables.
- The coefficient can take any values from -1 to 1. The interpretations of the values are:
  - **-1:** Perfect negative correlation. The variables tend to move in opposite directions (i.e., when one variable increases, the other variable decreases).
  - **0:** No correlation. The variables do not have a relationship with each other.
  - **1:** Perfect positive correlation. The variables tend to move in the same direction (i.e., when one variable increases, the other variable also increases).


Perfect Positive Correlation
Strong Positive Correlation
Weak Positive Correlation
No Correlation
Weak Negative Correlation
Strong Negative Correlation
Perfect Negative Correlation


Perfect Positive Correlation
1
High Positive Correlation
0.9
Low Positive Correlation
0.5
No Correlation
0
Low Negative Correlation
-0.5
High Negative Correlation
-0.9
Perfect Negative Correlation
-1

Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

Correlation Coefficient, *r*

	X	Y
	S&P 500	Apple
2013	1691.75	68.96
2014	1977.80	100.11
2015	1884.09	109.06
2016	2151.13	112.18
2017	2519.36	154.12

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where:

•$\mathbf{r_{xy}}$ – the **correlation coefficient** of the linear relationship between the variables x and y. It is also known as ***Pearson's correlation coefficient*** when applied to sample data.

•$\mathbf{x_i}$ – the values of the x-variable in a sample

•$\bar{\mathbf{x}}$ – the mean of the values of the x-variable

•$\mathbf{y_i}$ – the values of the y-variable in a sample

•$\bar{\mathbf{y}}$ – the mean of the values of the y-variable

Example Data:

	S&P 500	Apple
2013	1691.75	68.96
2014	1977.80	100.11
2015	1884.09	109.06
2016	2151.13	112.18
2017	2519.36	154.12
	X	Y

Task:

Find correlation between two attributes [X and Y]

We will do it by finding the **correlation-coefficient** between them

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

	S&P 500	Apple	Step 3: a $(x-\bar{x})$	Step 3: b $(y-\bar{y})$	Step 4: a x b	Step 5: a^2	Step 5: b^2
2013	1691.75	68.96	- 353.08	- 39.93	14,096.91	124,662.66	1,594.09
2014	1977.80	100.11	- 67.03	- 8.78	588.22	4,492.48	77.02
2015	1884.09	109.06	- 160.74	0.17	- 27.97	25,836.07	0.03
2016	2151.13	112.18	106.30	3.29	350.16	11,300.52	10.85
2017	2519.36	154.12	474.53	45.23	21,465.08	225,182.62	2,046.11
Mean	**2044.83**	**108.89**	**Sums**		**36,472.40**	**391,474.35**	**3,728.10**

$$r_{xy} = \frac{36{,}472.40}{\sqrt{391{,}474.35 \times 3{,}728.10}} = 0.95$$

The coefficient indicates that the given variables have a high positive correlation.

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction
- Summary

31

Data Transformation

Data transformation $-2, 32, 100, 59, 48 \longrightarrow -0.02, 0.32, 1.00, 0.59, 0.48$

Data Transformation

- √ Smoothing: remove noise from data (binning, clustering, regression)
- √ Aggregation: summarization, data-cube generation
- Generalization: concept hierarchy climbing
- √ Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- √ Attribute/feature construction
  - New attributes constructed from the given ones

Data Summarization

- Summary Statistics
  - Summary statistics are numbers that summarize properties of the data
    - Summarized properties include frequency, location and spread
      - Examples: location - mean
        spread - standard deviation
    - Most summary statistics can be calculated in a single pass through the data

Central Tendency: Mean and Median

- The mean/average is the most common measure of the location of a set of points.
- However, the mean is very sensitive to outliers.
- Thus, the median or a trimmed mean is also commonly used.

$$\text{mean}(x) = \overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\text{median}(x) = \begin{cases} x_{(r+1)} & \text{if } m \text{ is odd, i.e., } m = 2r+1 \\ \frac{1}{2}(x_{(r)} + x_{(r+1)}) & \text{if } m \text{ is even, i.e., } m = 2r \end{cases}$$

Frequency and Mode

- The frequency of an attribute value is the percentage of time the value occurs in the data set
  - For example, given the attribute ‘gender’ and a representative population of people, the gender ‘female’ occurs about 50% of the time.
- The mode of an attribute is the most frequent attribute value

Mean

7, 3, 4, 1, 7, 6

Sum of numbers divided by the total numbers

Mean = (7+3+4+1+7+6)/6
= 28/6 = 4.66

Median

7, 3, 4, 1, 7, 6

Arrange in order and pick the middle value

1, 3, 4, 6, 7, 7

Median = (4+6)/2 = 5

Mode

7, 3, 4, 1, 7, 6

Most common number

7, 3, 4, 1, 7, 6

Mode = 7

Range

7, 3, 4, 1, 7, 6

Difference between highest and lowest

Range = 7 – 1 = 6

Data Transformation: Normalization

Data Transformation: Normalization

Min-max normalization

$$v' = \frac{v - \min(A)}{\max(A) - \min(A)}$$

Z-score normalization

$$v' = \frac{v - \mu}{\sigma}$$

Decimal Scaling

$$v' = \frac{v}{10^j}$$

min-max normalization

Given Dataset, [A: 8, 10, 15, 20]

Task: Perform min-max normalization on the given data, and calculate normalized values.

Equation

$$v' = \frac{v - \min(A)}{\max(A) - \min(A)}$$

$\min(A) = 8$
$\max(A) = 20$

For v=8,

Normalized value, $v' = \frac{8-8}{20-8}$
$= 0$

For v=10,

Normalized value, $v' = \frac{10-8}{20-8}$
$= 0.16$

For v=15,

Normalized value, $v' = \frac{15-8}{20-8}$
$= 0.58$

For v=20,

Normalized value, $v' = \frac{20-8}{20-8}$
$= 1.0$

min-max normalization (contd.)

Given Dataset, [A: 8, 10, 15, 20]

Task: Perform min-max normalization on the given data **within a range [40, 60]**, , and calculat normalized values.

Equation

$$v' = \frac{v - \min(A)}{\max(A) - \min(A)} [newmax(A) - newmin(A)] + newmin(A)$$

$\min(A) = 8$
$\max(A) = 20$
$newmax(A) = 40$
$newmax(A) = 60$

For v=8,

Normalized value, $v' = \frac{8-8}{20-8}$ [60-40]+40
$= 40$

For v=10,

Normalized value, $v' = \frac{10-8}{20-8}$ [60-40]+40
$= 43.33$

For v=15,

Normalized value, $v' = \frac{15-8}{20-8}$ [60-40]+40
$= 51.67$

For v=20,

Normalized value, $v' = \frac{20-8}{20-8}$ [60-40]+40
$= 60$

Z-Score Normalization

Given Dataset, [A: 8, 10, 15, 20]

Task: Perform z-score normalization on the given data, and calculate normalized values.

Equation $$v' = \frac{v - \mu}{\sigma}$$

Mean, $\mu = \frac{\Sigma x_i}{N} = \frac{8+10+15+20}{4} = 13.25$

Standard Deviation, $\sigma = \sqrt{\frac{\Sigma\,(x_i - \mu)^2}{N}}$

$$= \sqrt{\frac{(8-13.25)^2+(10-13.25)^2+(15-13.25)^2+(20-13.25)^2}{4}} = 4.65$$

For v=8,

Normalized value, $v' = \frac{8 - 13.25}{4.65}$

$= -1.129$

For v=10,

Normalized value, $v' = \frac{10 - 13.25}{4.65}$

$= -0.698$

For v=15,

Normalized value, $v' = \frac{15 - 13.25}{4.65}$

$= 0.376$

For v=20,

Normalized value, $v' = \frac{20 - 13.25}{4.65}$

$= 1.45$

Example of Decimal scaling :

CGPA	Formula	CGPA Normalized after Decimal scaling
2	2/10	0.2
3 max	3/10	0.3

0ne 0

We will check the maximum value among our attribute CGPA. Here maximum value is 3 so we can convert it to a decimal by dividing by 10. Why 10?

we will count total numbers in our maximum value and then put 1 and after 1 we can put zeros equal to the length of the maximum value.

Here 3 is the maximum value and the total numbers in this value are only 1. so we will put one zero after one.

Example 2:

Salary bonus	Formula	CGPA Normalized after Decimal scaling
400 max Three 0	400 / 1000	0.4
310	310 / 1000	0.31

We will check the maximum value of our attribute "**salary bonus**". Here maximum value is 400 so we can convert it into a decimal by dividing it by 1000. Why 1000?

400 contains three digits and we so we can put three zeros after 1. So, it looks like 1000.

Attribute/Feature Construction

- New attributes are constructed from given attributes and added in order to help improve accuracy and understanding of structure in high-dimension data
- Example
  - Add the attribute *area* based on the attributes *height* and *width*

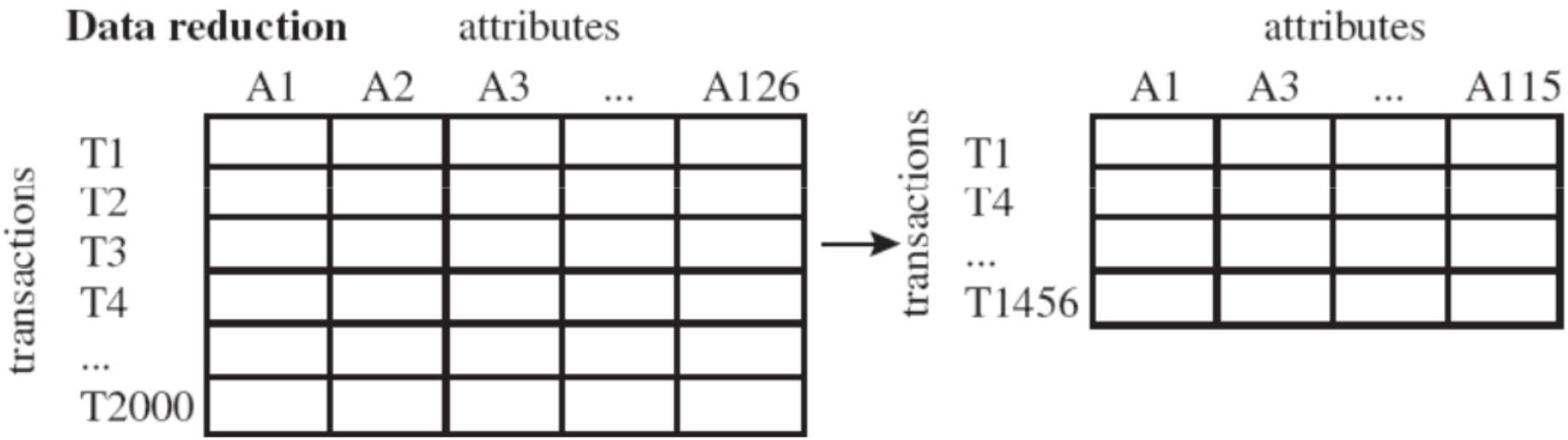
Area = height X width

Data Preprocessing

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction
- Summary

45

Data Reduction



- Reduction in number of sample
- Reduction in number of attribute (dimension)

Data Reduction

- Why data reduction?
  - A database/data warehouse may store terabytes of data
  - Complex data analysis/mining may take a very long time to run on the complete data set

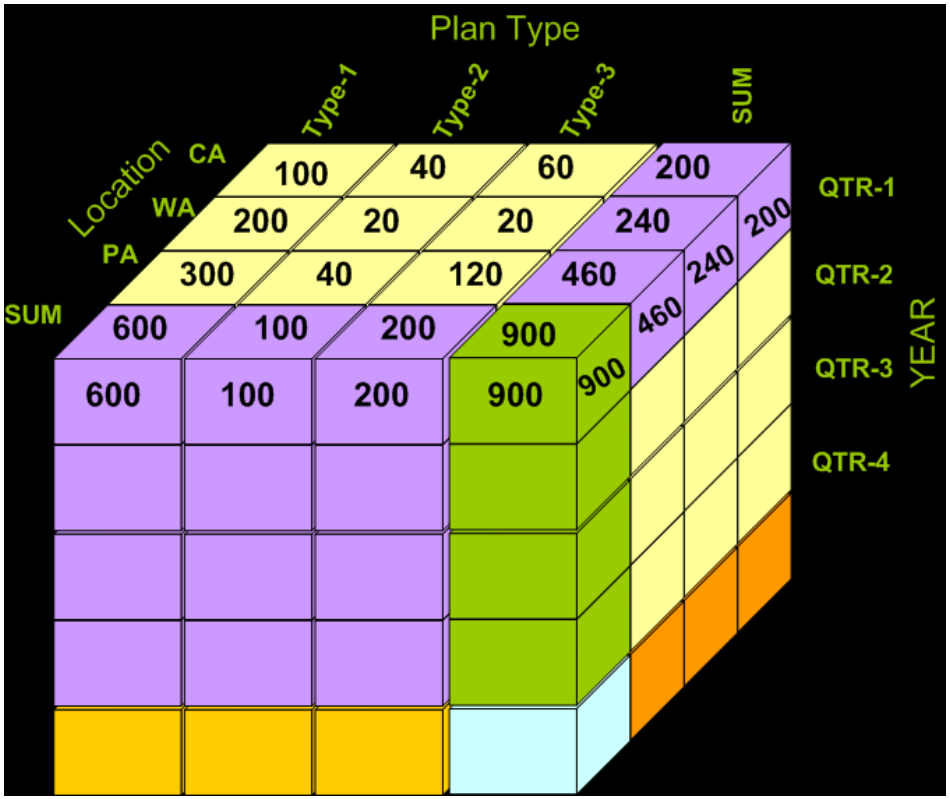
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results

Data reduction strategies

- Data cube aggregation-
- Dimensionality reduction — e.g., remove unimportant attributes, Principal Component analysis (PCA), Wavelet transforms, feature-selection techniques
- Numerosity reduction — alternative-brief form of data representation; e.g., Regression and Log-Linear Models, Histograms, clustering, sampling
- Data Discretization – e.g., Binning, entropy-based discretization, cluster analysis, interval merging by chi-square analysis etc.
- Concept hierarchy generation-

Data Reduction 1: Data Cube Aggregation

- Multiple levels of aggregation in data cubes
- Queries regarding aggregated information should be answered using data cube, when possible

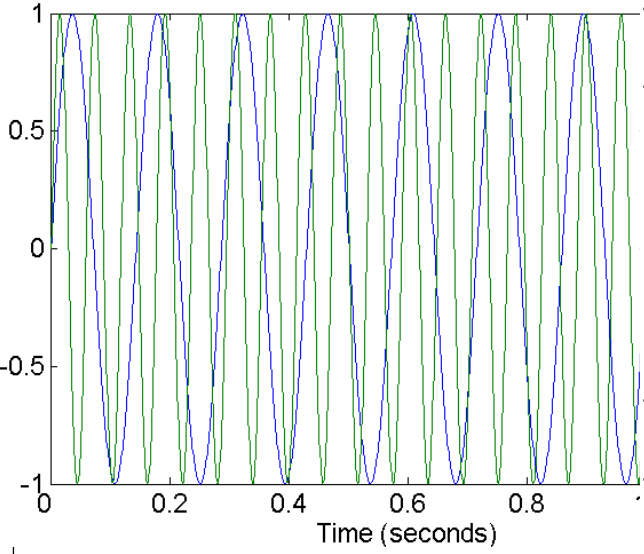


Data Reduction 2: Dimensionality Reduction

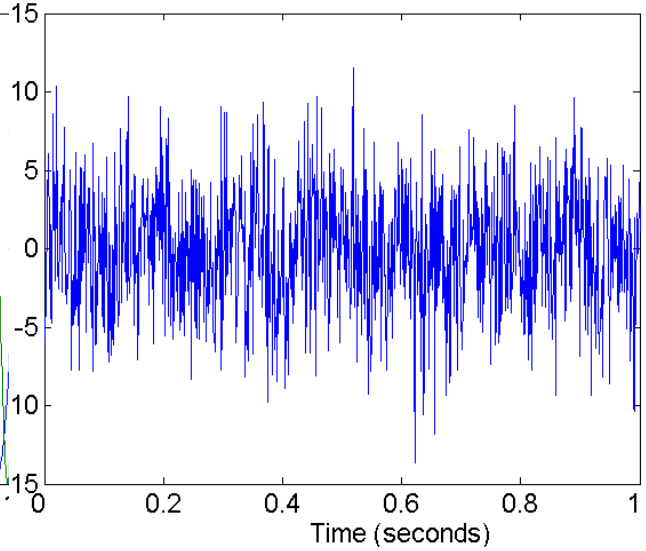
- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

Mapping Data to a New Space

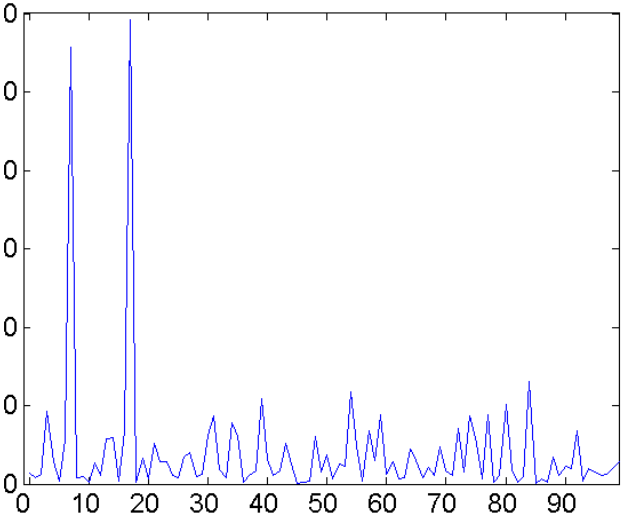
- **Fourier transform**
- **Wavelet transform**



Two Sine Waves



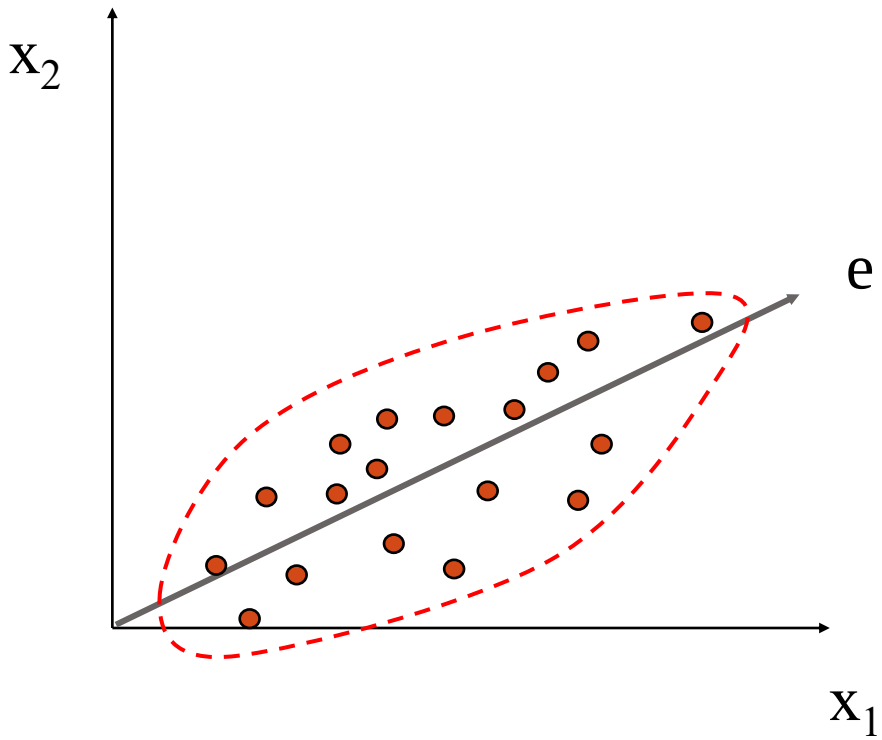
Two Sine Waves + Noise



Frequency

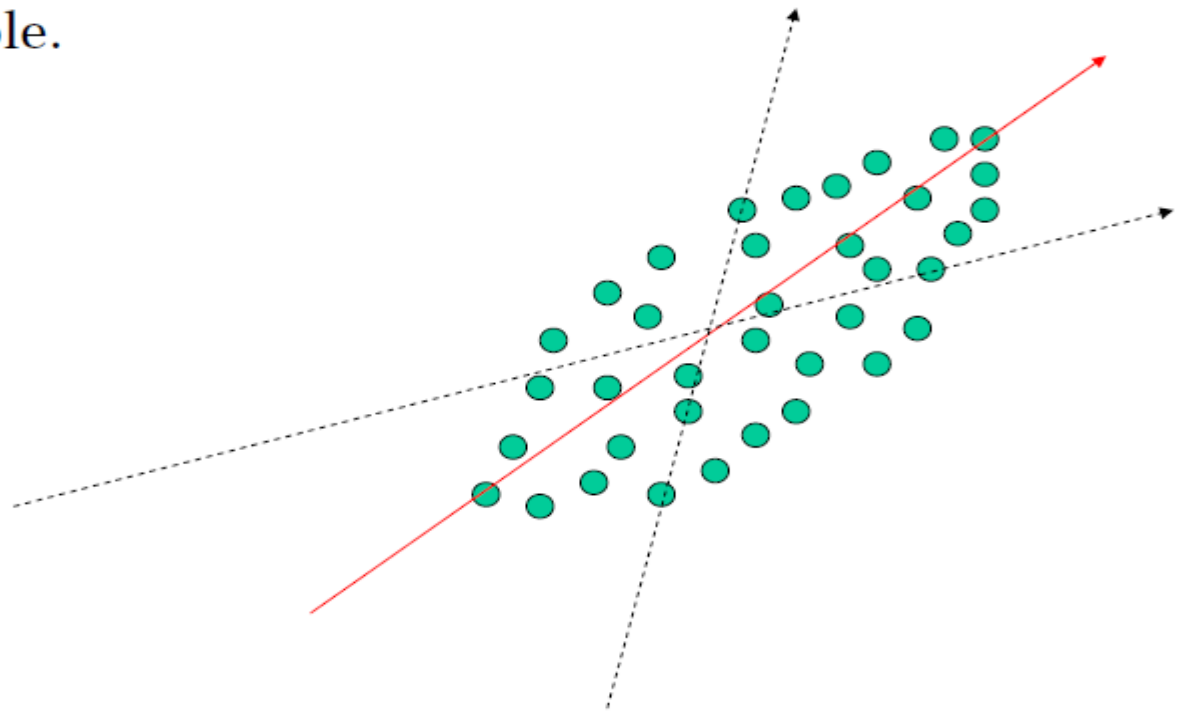
Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space.



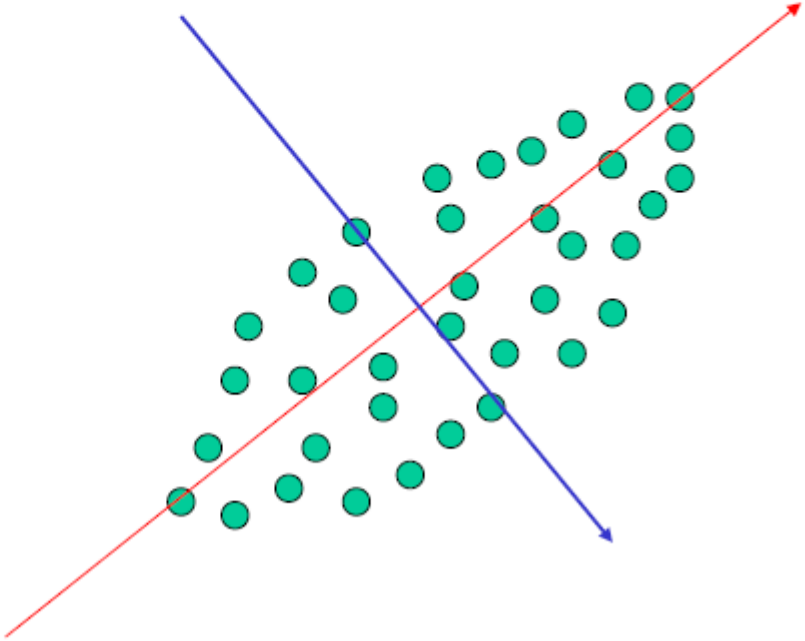
Basic Idea of PCA

Goal: Map data points into a few dimension while trying to preserve the variance of data as much as possible.



Basic Idea of PCA

Goal: Map data points into a few dimension while trying to preserve the variance of data as much as possible.



Attribute Subset Selection [Feature Selection]

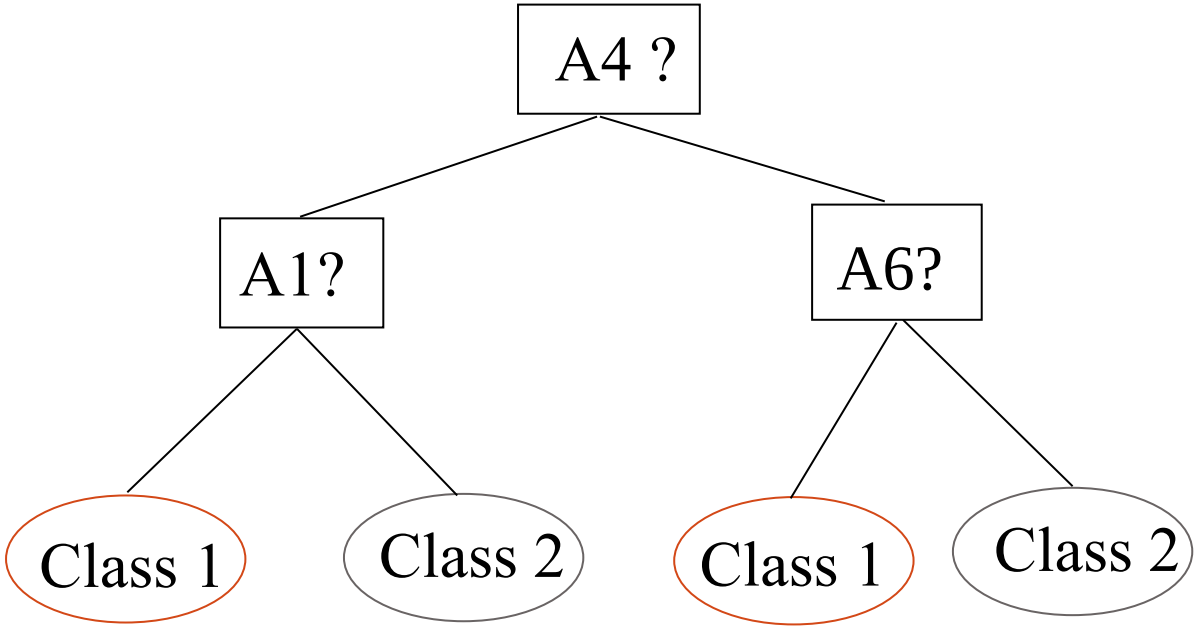
- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Heuristic Search in Attribute Selection

- Typical heuristic attribute selection methods:
  - Best single attribute under the attribute independence assumption: choose by significance tests (statistical approaches)
  - Best step-wise feature selection:
    - The best single-attribute is picked first
    - Then next best attribute condition to the first, ...
  - Step-wise attribute elimination:
    - Repeatedly eliminate the worst attribute
  - Best combined attribute selection and elimination
  - Optimal branch and bound:
    - Use attribute elimination and backtracking

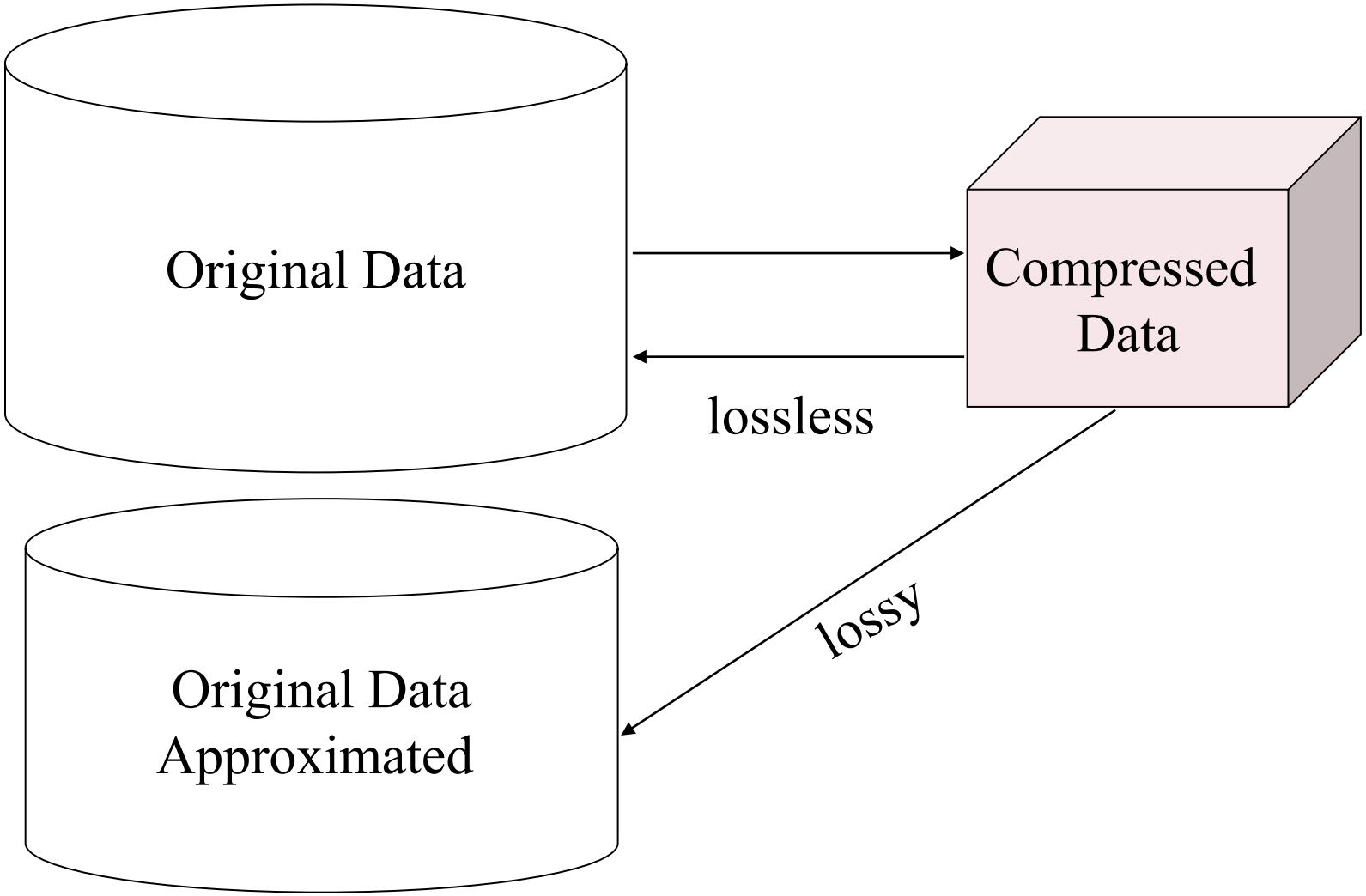
Example of Decision Tree Induction

Initial attribute set:
{A1, A2, A3, A4, A5, A6}



------> Reduced attribute set: {A1, A4, A6}

Data Compression



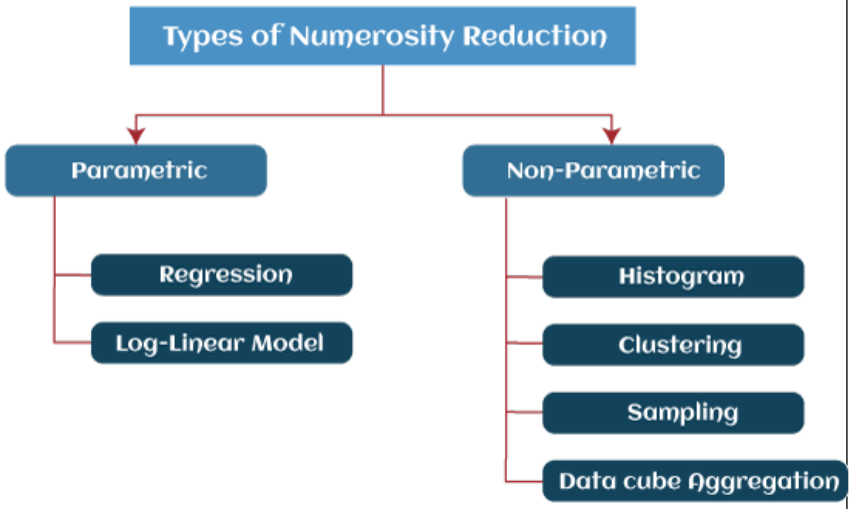
Data Reduction 3: Numerosity Reduction

- In the Numerosity reduction, the data volume is reduced by choosing an alternative, smaller form of data representation.

- These techniques may be ***parametric*** or ***nonparametric***.

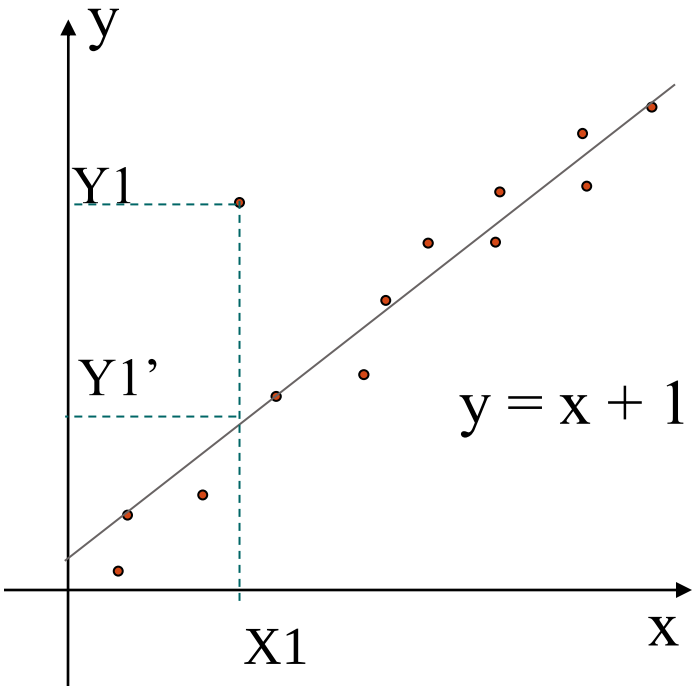
- ***For parametric methods***, a model is used to estimate the data, so that only <u>the data parameters need to be stored</u>, instead of the actual data, for example, Log-linear models.

- ***Non-parametric methods*** are used for <u>storing a reduced representation</u> of the data which include histograms, clustering, and sampling.



Parametric Method: Regression Analysis

- Regression analysis: A collective name for techniques for the modeling and analysis of numerical data consisting of values of a ***dependent variable*** (also called ***response variable*** or *measurement*) and of one or more *independent variables* (aka. ***explanatory variables*** or ***predictors***)
- The parameters are estimated so as to give a "**best fit**" of the data
- Most commonly the best fit is evaluated by using the ***least squares method***, but other criteria have also been used



- Used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships

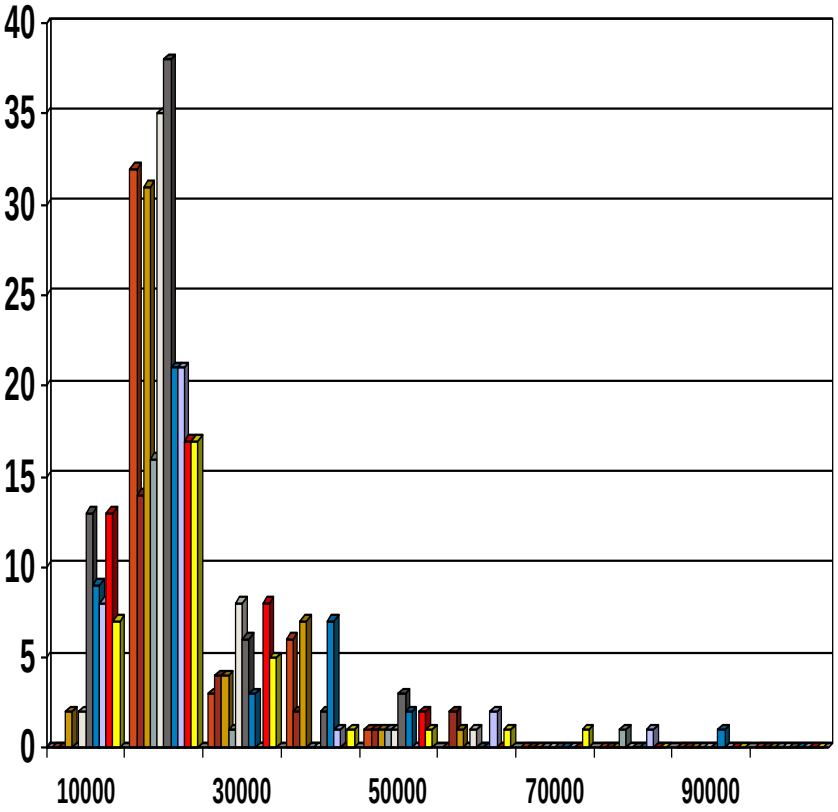
Regression Analysis and Log-Linear Models

- Linear regression: $Y = w X + b$
  - Two regression coefficients, w and b, specify the line and are to be estimated by using the data at hand
  - Using the least squares criterion to the known values of $Y_1, Y_2, \ldots, X_1, X_2, \ldots$.
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$
  - Many nonlinear functions can be transformed into the above
- Log-linear models:
  - Approximate discrete multidimensional probability distributions
  - Estimate the probability of each point (tuple) in a multi-dimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations
  - Useful for dimensionality reduct

Model	Equation	Interpretation
Level-Level Regression	$Y = \alpha + \beta X$	One unit change in X leads to β unit change in Y
Log-Linear Regression	$log(Y) = \alpha + \beta X$	One unit change in X leads to $100 * \beta$ percent change in Y
Linear-Log Regression	$Y = \alpha + \beta\, log(X)$	One percent change in X leads to $\beta/100$ unit change in Y
Log-Log Regression	$log(Y) = \alpha + \beta\, log(X)$	One percent change in X leads to β percent change in Y

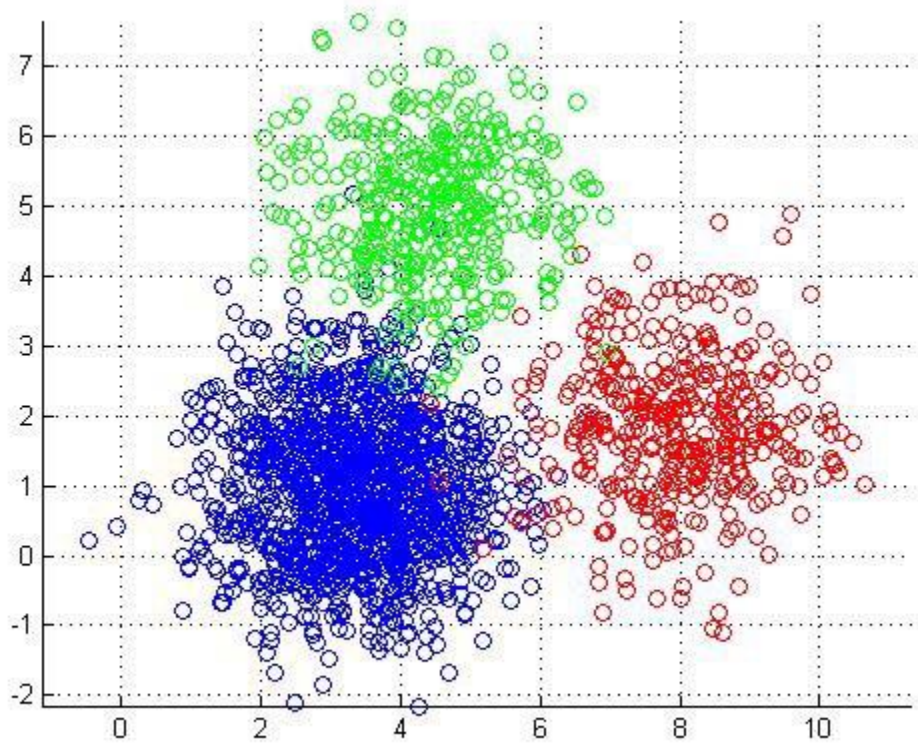
Non-parametric Method: Histogram Analysis

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)



Non-parametric Method: Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only



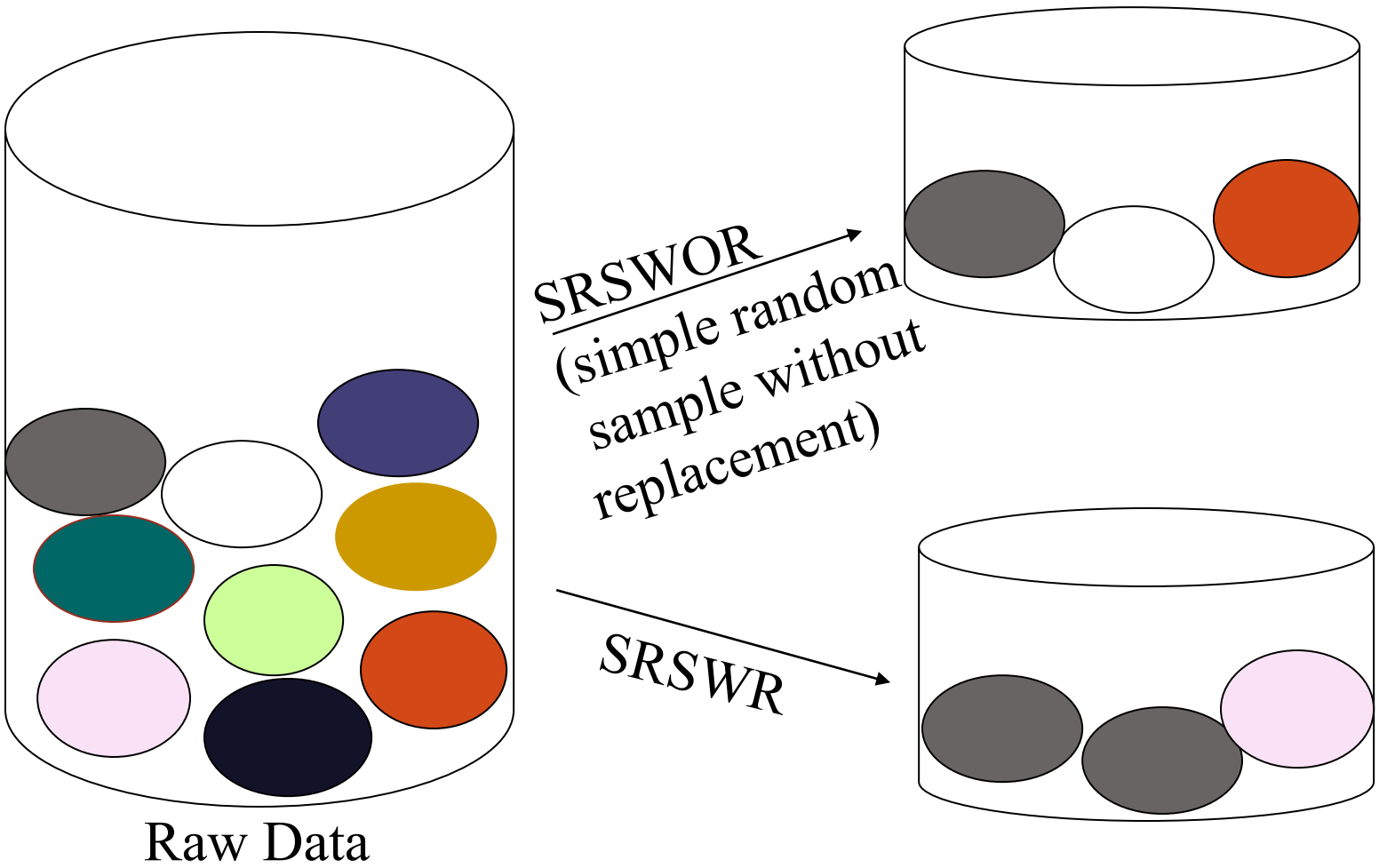
Non-parametric Method: Sampling

- Sampling: obtaining a small sample s to represent the whole data set N
- Key principle: Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling:

Types of Sampling

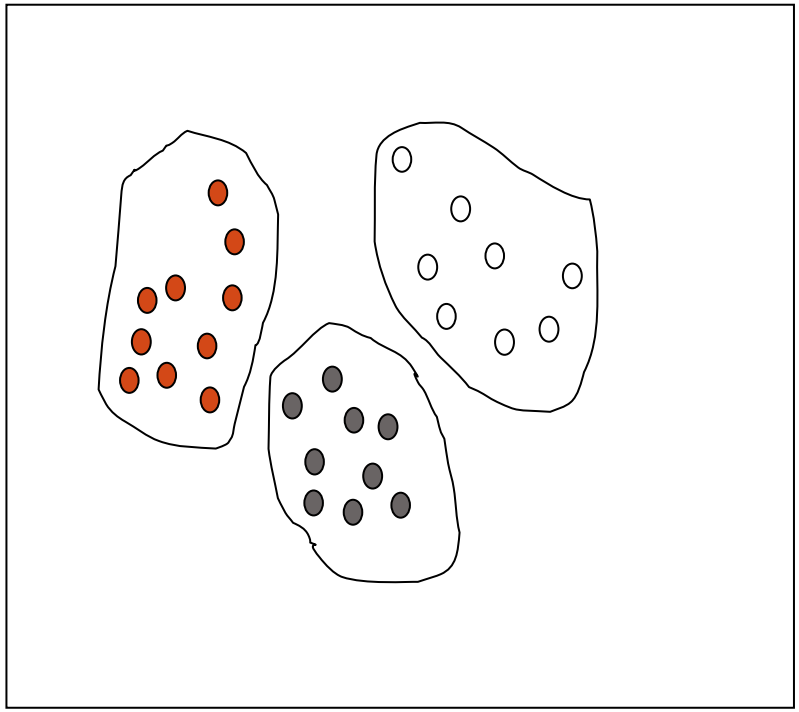
- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling:**
  - Partition the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)

Sampling: With or without Replacement

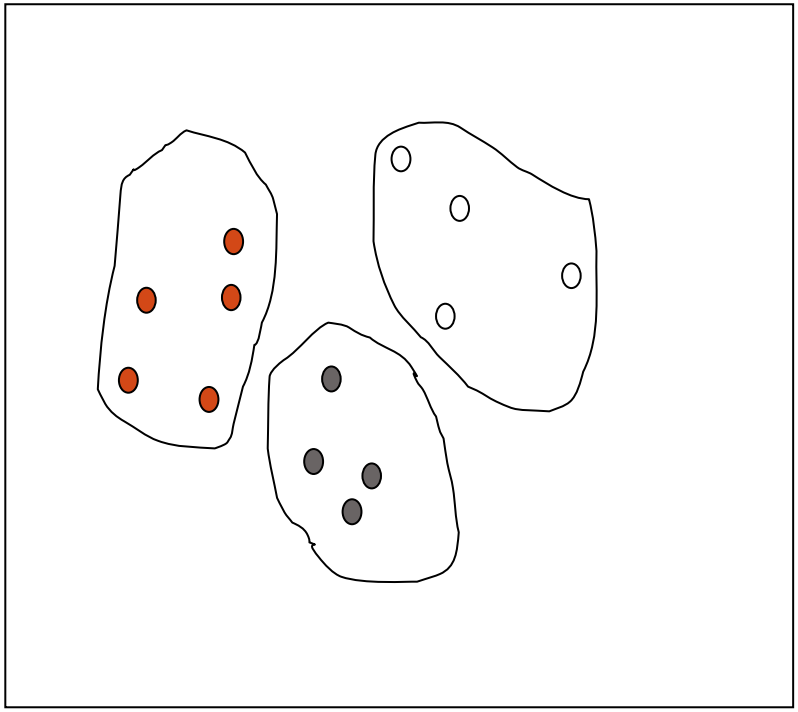


Sampling: Cluster or Stratified Sampling

Raw Data

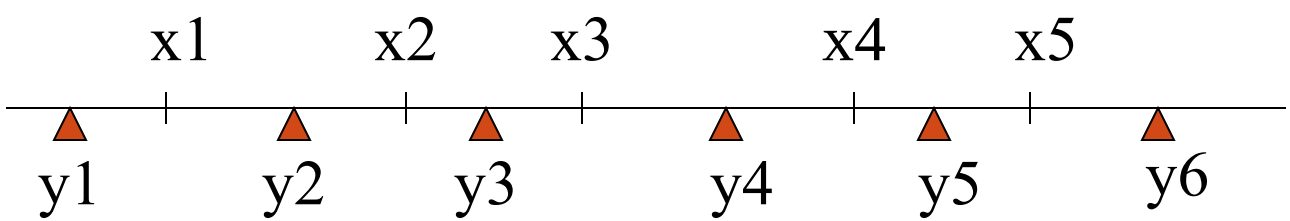


Cluster/Stratified Sample



Data Reduction 4: Discretization/Quantization

- Three types of attributes:
  - Nominal — values from an unordered set
  - Ordinal — values from an ordered set
  - Continuous — real numbers
- Discretization/Quantization:
  - divide the range of a continuous attribute into intervals



  - Some classification algorithms only accept categorical attributes.
  - Reduce data size by discretization
  - Prepare for further analysis

Discretization and Concept Hierarchy

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values.
- Concept Hierarchies
  - reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior).

Discretization and concept hierarchy generation for numeric data

- Hierarchical and recursive decomposition using:
  - Binning (data smoothing)
  - Histogram analysis (numerosity reduction)
  - Clustering analysis (numerosity reduction)
- Entropy-based discretization
- Segmentation by natural partitioning

Concept Hierarchy Generation

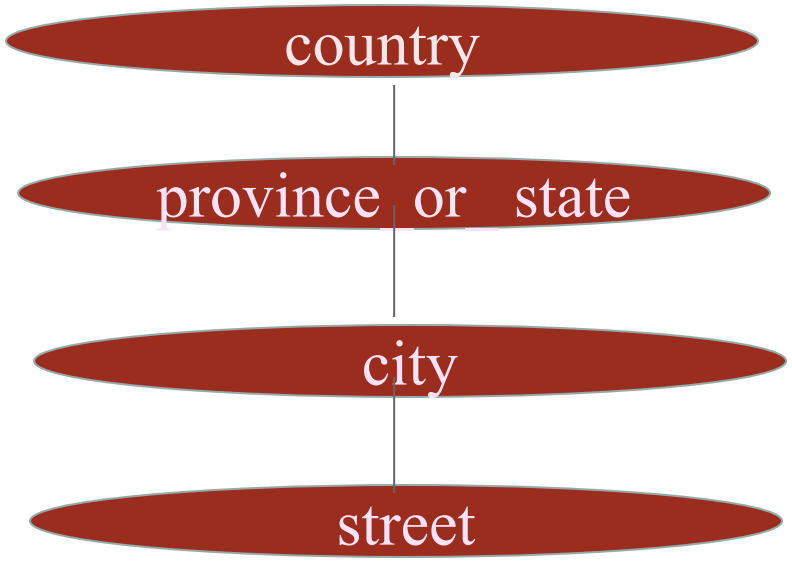
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth, adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - **{Urbana, Champaign, Chicago} < Illinois**
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street, city, state, country*}

Concept hierarchy generation w/o data semantics - Specification of a set of attributes

Concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set. The attribute with the most distinct values is placed at the lowest level of the hierarchy (limitations?)



15 distinct values

65 distinct values

3567 distinct values

674,339 distinct values

SUMMARY

- **Data preprocessing**
  - **Data cleaning**
    - **Missing values**
      - Use the most probable value to fill in the missing value (and five other methods)
    - **Noisy data**
      - Binning; Regression; Clusttering
  - **Data integration**
    - **Entity ID problem**
      - Metadata
    - **Redundancy**
      - Correlation analysis (Correlation coefficient, chi-square test)
  - **Data trasnformation**
    - **Smoothing**
      - **Data cleaning**
    - **Aggregation**
      - **Data reduction**
    - **Generailization**
      - **Data reduction**
    - **Normalization**
      - Min-max; z-score; decimal scaling
    - **Attribute Construction**
  - **Data reduction**
    - **Data cube aggregation**
      - Data cube store multidimensional aggregated information
    - **Attribute subset selection**
      - Stepwise forward selection; stepwise backward selection; combination; decision tree induction
    - **Dimensionality reduction**
      - Discrete wavelet trasnforms (DWT); Principle components analysis (PCA);
    - **Numerosity Reduction**
      - Regression and log-linear models; histograms; clustering; sampling
    - **Data discretization**
      - Binning; historgram analysis; entropy-based discretization;
      - Interval merging by chi-square analysis; cluster analysis; intuitive partitioning
    - **Concept hierarchy**
      - Concept hierarchy generation

Mining Association Rules

Road map

- Basic concepts
- Apriori algorithm
- FP-Growth Algorithm
- Summary

Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are *categorical*.
- No good algorithm for numerical data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

The model: data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*.
- Transaction t :
  - t a set of items, and $t \subseteq I$.
- Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$.

Transaction data: supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}

  t2: {apple, eggs, salt, yogurt}

  … …

  tn: {biscuit, eggs, milk}

- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data: a set of documents

- **A text document data set. Each document is treated as a "bag" of keywords**

doc1:	Student, Teach, School
doc2:	Student, School
doc3:	Teach, School, City, Game
doc4:	Baseball, Basketball
doc5:	Basketball, Player, Spectator
doc6:	Baseball, Coach, Game, Team
doc7:	Basketball, Team, City, Game

The model: rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$.
- An association rule is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \textit{ and } X \cap Y = \varnothing$$

- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset

Rule strength measures

- Support: The rule holds with support *sup* in *T* (the transaction data set) if sup% of transactions contain $X \cup Y$. [$X \cup Y$ happens sup% times.]
  - $sup = \Pr(X \cup Y)$.
- Confidence: The rule holds in *T* with confidence *conf* if *conf*% of transactions that contain *X* also contain *Y.*
  - $conf = \Pr(Y \mid X)$
- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability.

Support and Confidence

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *T* has *n* transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - Mining with data on hard disk (not in memory)

An example

t1:	Beef, Chicken, Milk
t2:	Beef, Cheese
t3:	Cheese, Boots
t4:	Beef, Chicken, Cheese
t5:	Beef, Chicken, Clothes, Cheese, Milk
t6:	Chicken, Clothes, Milk
t7:	Chicken, Milk, Clothes

- Transaction data
- Assume:

  minsup = 30%
  minconf = 80%
- An example frequent *itemset*:

{Chicken, Clothes, Milk} [sup = 3/7]

- Association rules from the itemset:

Clothes → Milk, Chicken [sup = 3/7, conf = 3/3]

… …

Clothes, Chicken → Milk, [sup = 3/7, conf = 3/3]

Many mining algorithms

- There are a large number of them!!
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
  - Given a transaction data set *T*, and a minimum support and a minimum confident, the set of association rules existing in *T* is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study here: the Apriori Algorithm

Road map

- Basic concepts
- Apriori algorithm
- FP-Growth Algorithm
- Summary

The Apriori algorithm

- **Probably the best known algorithm**
- **Two steps**:
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
  - Use frequent itemsets to generate rules.

- E.g., a frequent itemset

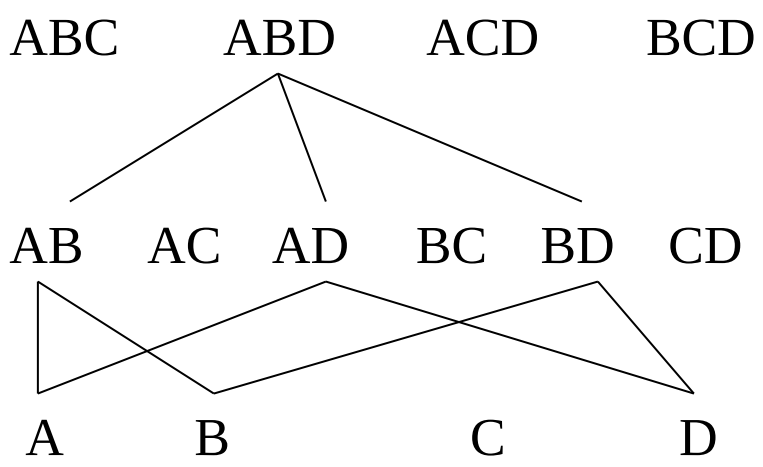
  {Chicken, Clothes, Milk} [sup = 3/7]

  and one rule from the frequent itemset

  Clothes $\rightarrow$ Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
  - In each iteration k, only consider itemsets that contain some k-1 frequent itemset.

- Find frequent itemsets of size 1: F_1
- From $k = 2$
  - C_k = candidates of size k: those itemsets of size k that could be frequent, given F_{k-1}
  - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).

Details: the algorithm

Algorithm Apriori(*T*)

$C_1 \leftarrow$ init-pass(T);

$F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq minsup\}$; // n: no. of transactions in T

for (k = 2; $F_{k-1} \neq \emptyset$; k++) **do**

  $C_k \leftarrow$ candidate-gen(F_{k-1});

  for each transaction $t \in T$ **do**

    for each candidate $c \in C_k$ **do**

      if c is contained in t **then**

        $c.count$++;

    end

  end

  $F_k \leftarrow \{c \in C_k \mid c.count/n \geq minsup\}$

end

return $F \leftarrow \bigcup_k F_k$;

Apriori candidate generation

- The candidate-gen function takes F_{k-1} and returns a superset (called the candidates) of the set of all frequent *k*-itemsets. It has two steps
  - *join* step: Generate all possible candidate itemsets C_k of length k
  - *prune* step: Remove those candidates in C_k that cannot be frequent.

Candidate-gen function

Function candidate-gen($F_{k\text{-}1}$)

$C_k \leftarrow \emptyset$;

forall $f_1, f_2 \in F_{k\text{-}1}$

with $f_1 = \{i_1, \ldots, i_{k\text{-}2}, i_{k\text{-}1}\}$

and $f_2 = \{i_1, \ldots, i_{k\text{-}2}, i'_{k\text{-}1}\}$

and $i_{k\text{-}1} < i'_{k\text{-}1}$ **do**

$c \leftarrow \{i_1, \ldots, i_{k\text{-}1}, i'_{k\text{-}1}\}$; // join f_1 and f_2

$C_k \leftarrow C_k \cup \{c\}$;

for each (k-1)-subset s of c **do**

if ($s \notin F_{k\text{-}1}$) **then**

delete c from C_k; // prune

end

end

return C_k;

Apriori Candidate Generation — Example

Given, minimum support = 50% -> (50/100) *(n=4) => 2
minimum confidence =70%

n=number of transactions

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D →

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→ F_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D →

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

→ F_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

:2

{1, 2,3}:0
{1, 3,5}:1

Scan D →

F_3

itemset	sup
{2 3 5}	2

Step 2: Generating rules from frequent itemsets

- Frequent itemsets ≠ association rules
- One more step is needed to generate association rules
- For each frequent itemset *X*,

  For each proper nonempty subset *A* of *X*,
  - Let *B* = X - *A*
  - ***A* → *B* is an association rule if**
    - Confidence(A → B) ≥ minconf,

      support(A → B) = support(A∪B)

      confidence(A → B) = support(A ∪ B) / support(A)

Apriori Generating Rule- Example

Candidate Generation part

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D →

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→

F_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D ←

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

←

F_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D →

F_3

itemset	sup
{2 3 5}	2

In C3, other itemsets are not shown due to their supp<minsup.

Apriori Generating Rule- Example

Given,
Minimum support = 50%
Minimum confidence = 70%

itemset	sup
{2 3 5}	2

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Rule	Support	Confidence	Confidence(%)
{2,3} -> {5}	2	2/2 = 1	100% >=70%
{2,5} -> {3}	2	2/3 = 0.67	67%
{3,5} -> {2}	2	2/2 = 1	100% >=70%
{5} -> {2,3}	2	2/3 = 0.67	67%
{3} -> {2,5}	2	2/3 = 0.67	67%
{2} -> {3,5}	2	2/3 = 0.67	67%

We found the following rules:

{2,3} -> {5}
{3,5} -> {2}

Interpretation of rule: {2,3} -> {5}:

If we sell 2 and 3 together, probability of selling 5 is 100%

Generating rules: summary

- To recap, in order to obtain $A \rightarrow B$, we need to have support($A \cup B$) and support(A)
- All the required information for confidence computation has already been recorded in itemset generation. No need to see the data *T* any more.
- This step is not as time-consuming as frequent itemsets generation.

On Apriori Algorithm

Seems to be very expensive

- Level-wise search
- K = the size of the largest itemset
- It makes at most K passes over data
- In practice, K is bounded (10).
- The algorithm is very fast. Under some conditions, all rules can be found in linear time.

Apriori Limitations

- Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge.

- Apriori needs multiple scans of the database to check the support of each itemset generated and this leads to high costs.

Road map

- Basic concepts
- Apriori algorithm
- FP-Growth Algorithm
- Summary

FP-Growth Algorithm Introduction

- This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.
- For so much, it uses a divide-and-conquer strategy.
- The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

FP-Growth Algorithm Introduction (Contd.)

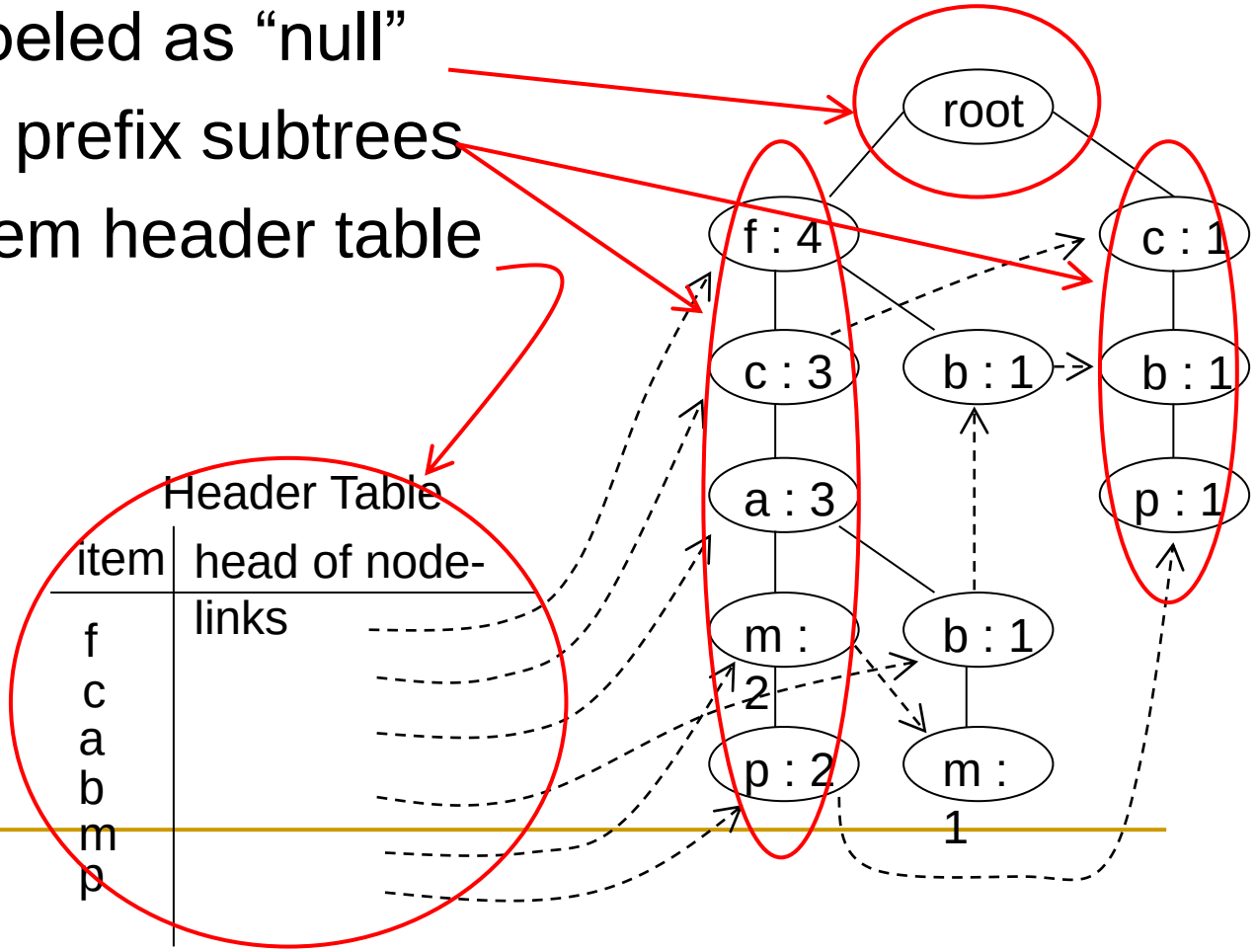
This algorithm works as follows:

- First, it compresses the input database creating an FP-tree instance to represent frequent items.
- After this first step, it divides the compressed database into a set of conditional databases, each associated with one frequent pattern.
- Finally, each such database is mined separately.

Using this strategy, the FP-Growth reduces the search costs by recursively looking for short patterns and then concatenating them into the long frequent patterns.

FP-Tree

- Three components:
  - One root: labeled as “null”
  - A set of item prefix subtrees
  - A frequent-item header table



An Example

The given data is a hypothetical dataset of transactions with each letter representing an item. **The minimum support given is 3.**

TID	Items Bought
100	*f, a, c, d, g, i, m, p*
200	*a, b, c, f, l, m, o*
300	*b, f, h, j, o*
400	*b, c, k, s, p*
500	*a, f, c, e, l, p, m, n*

- In the frequent pattern growth algorithm, first, we find the frequency of each item. The following table gives the frequency of each item in the given data.

TID	Items Bought
100	*f, a, c, d, g, i, m, p*
200	*a, b, c, f, l, m, o*
300	*b, f, h, j, o*
400	*b, c, k, s, p*
500	*a, f, c, e, l, p, m, n*

Item	Frequency	Item	Frequency
a	3	j	1
b	3	k	1
c	4	l	2
d	1	m	3
e	1	n	1
f	4	o	2
g	1	p	3
h	1	s	1
i	1		

A **Frequent Pattern set (L)** is built, which will contain all the elements whose frequency is greater than or equal to the minimum support.

These elements are stored in descending order of their respective frequencies.

As minimum support is 3,after insertion of the relevant items, the set L looks like this:-

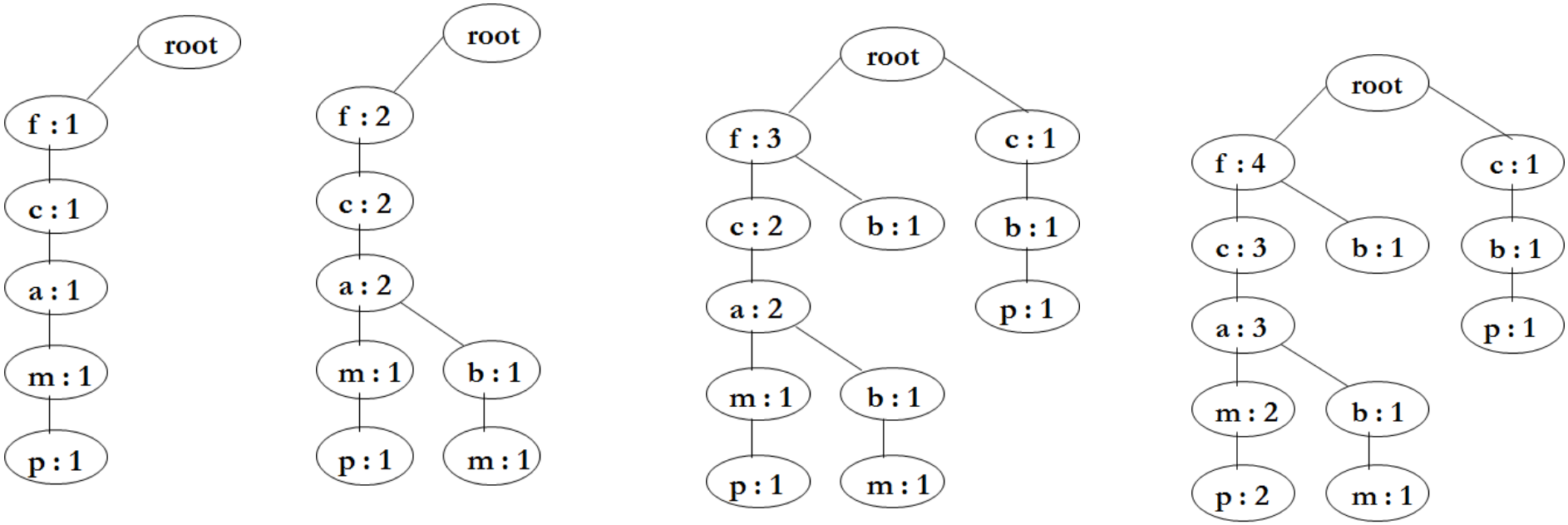
L = { (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) }

Now, for each transaction, the respective **Ordered-Item set** is built.

Frequent Pattern set L = { (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) }

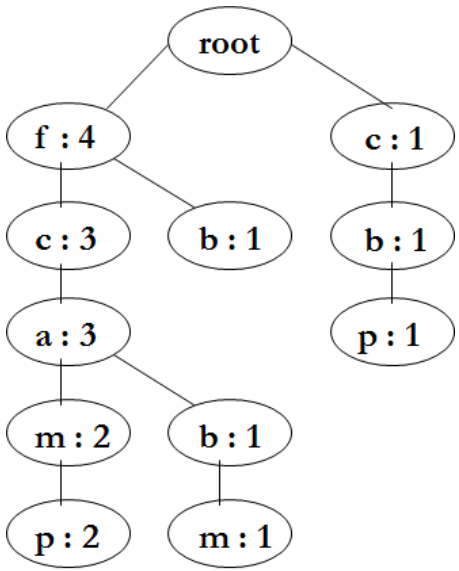
TID	Items Bought	(Ordered) Frequent Items
100	*f, a, c, d, g, i, m, p*	*f, c, a, m, p*
200	*a, b, c, f, l, m, o*	*f, c, a, b, m*
300	*b, f, h, j, o*	*f, b*
400	*b, c, k, s, p*	*c, b, p*
500	*a, f, c, e, l, p, m, n*	*f, c, a, m, p*

Now, all the Ordered-Item sets are inserted into a Tree Data Structure (**frequent pattern tree**).



TID	(Ordered) Frequent Items
100	*f, c, a, m, p*
200	*f, c, a, b, m*
300	*f, b*
400	*c, b, p*
500	*f, c, a, m, p*

- Now, for each item, the **Conditional Pattern Base** is computed which is the path labels of all the paths which lead to any node of the given item in the frequent-pattern tree.



Item	Conditional Pattern Base
p	{{f, c, a, m : 2}, {c, b : 1}}
m	{{f, c, a : 2}, {f, c, a, b : 1}}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}
a	{{f, c : 3}}
c	{{f : 3}}
f	Φ

- Now for each item, the **Conditional Frequent Pattern Tree** is built. It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base.

Item	Conditional Pattern Base	Conditional FP-Tree
p	{{f, c, a, m : 2}, {c, b : 1}}	{c : 3}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a :3}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	Φ
a	{{f, c : 3}}	{f, c : 3}
c	{{f : 3}}	{f : 3}
f	Φ	Φ

- From the Conditional Frequent Pattern tree, the **Frequent Pattern rules** are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding item.

Item	Conditional Pattern Base	Conditional FP-Tree	Frequent Patterns Generated
p	{{f, c, a, m : 2}, {c, b : 1}}	{c : 3}	{<c, p : 3>}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a :3}	{ <f, m : 3>, <c, m : 3> <a, m : 3>, <f, c, m : 3> <f, a, m : 3>, <c, a, m :3>}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	Φ	{}
a	{{f, c : 3}}	{f, c : 3}	{<f, a : 3>, <c, a : 3>, <f, c, a:3>}
c	{{f : 3}}	{f : 3}	{ <f, c : 3>}
f	Φ	Φ	{}

Generating Association Rules

- From the frequent itemsets, we will follow the same procedure of “Rule Generation” as shown in Apriori Algorithm.
  - Here, for each pattern we will generate all possible rules.
  - For all rules, we select the valid rules those pass the minconf threshold.

Differences between Apriori and FP-Growth

Apriori	FP Growth
Apriori generates frequent patterns by making the itemsets using pairings such as single item set, double itemset, and triple itemset.	FP Growth generates an FP-Tree for making frequent patterns.
Apriori uses candidate generation where frequent subsets are extended one item at a time.	FP-growth generates a conditional FP-Tree for every item in the data.
Since apriori scans the database in each step, it becomes time-consuming for data where the number of items is larger.	FP-tree requires only one database scan in its beginning steps, so it consumes less time.
A converted version of the database is saved in the memory	A set of conditional FP-tree for every item is saved in the memory
It uses a breadth-first search	It uses a depth-first search.

Decision Tree Classifier

Topics

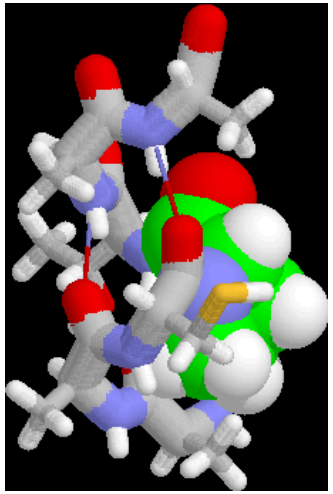
- Classification Basics
- Introduction to Decision Tree Classifier
- Basic Algorithms
  - CART
  - ID3
  - C4.5
- Issues of Decision Trees

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: <u>previously unseen</u> records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc



Classification Techniques

- Rule-based Methods
- Decision Tree based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

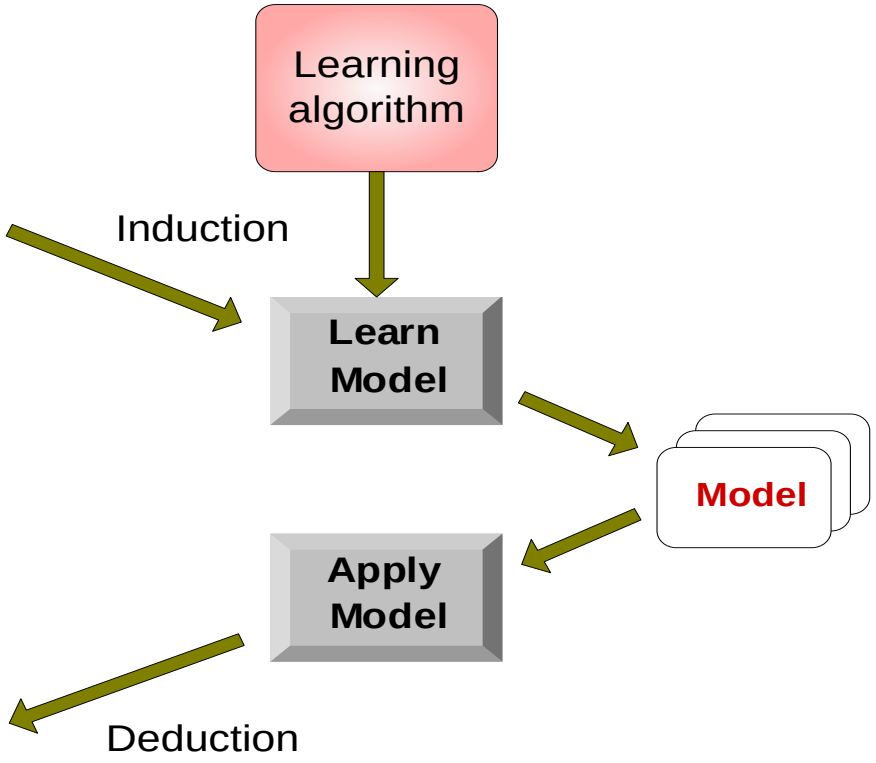
Illustrating a basic Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

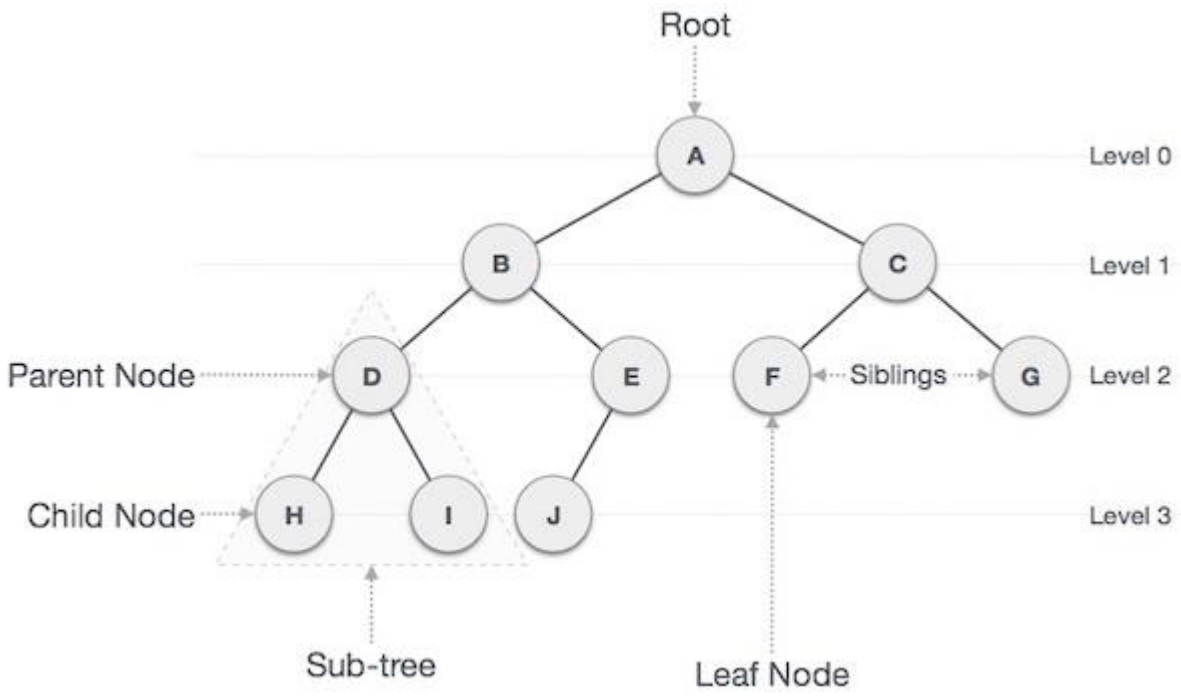
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision Tree

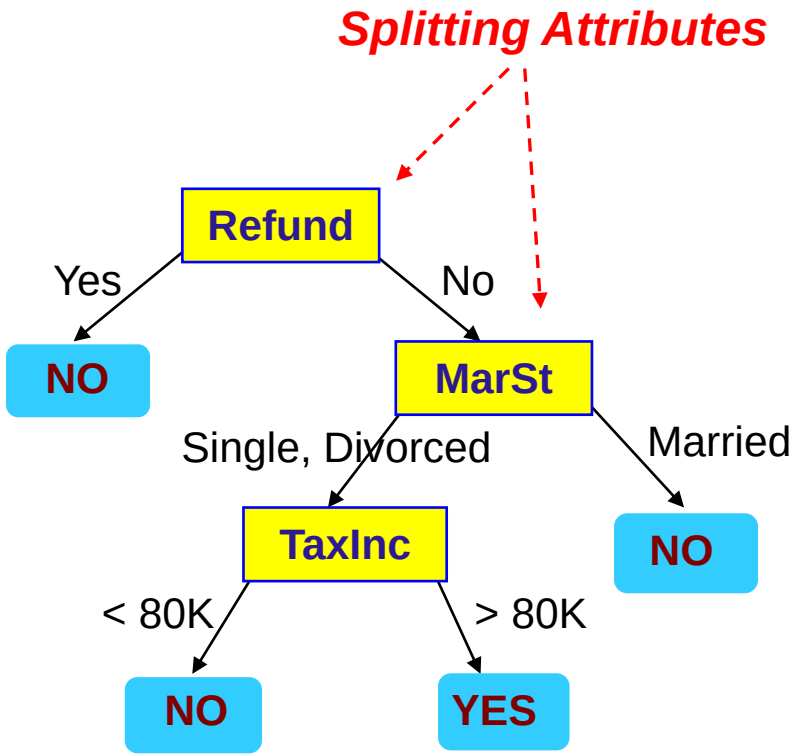
Tree Introduction



Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

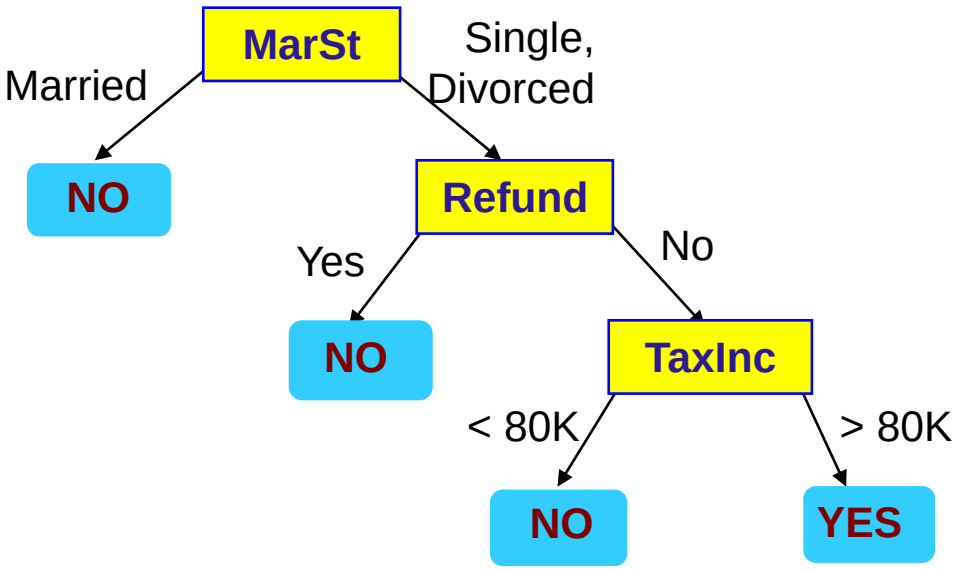
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

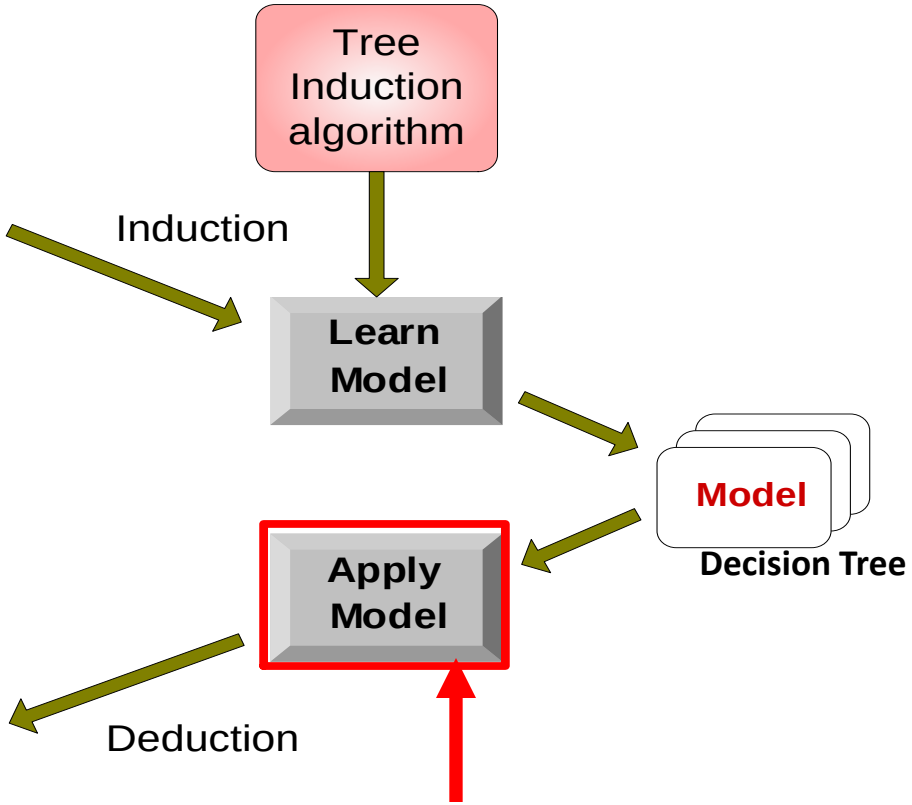
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

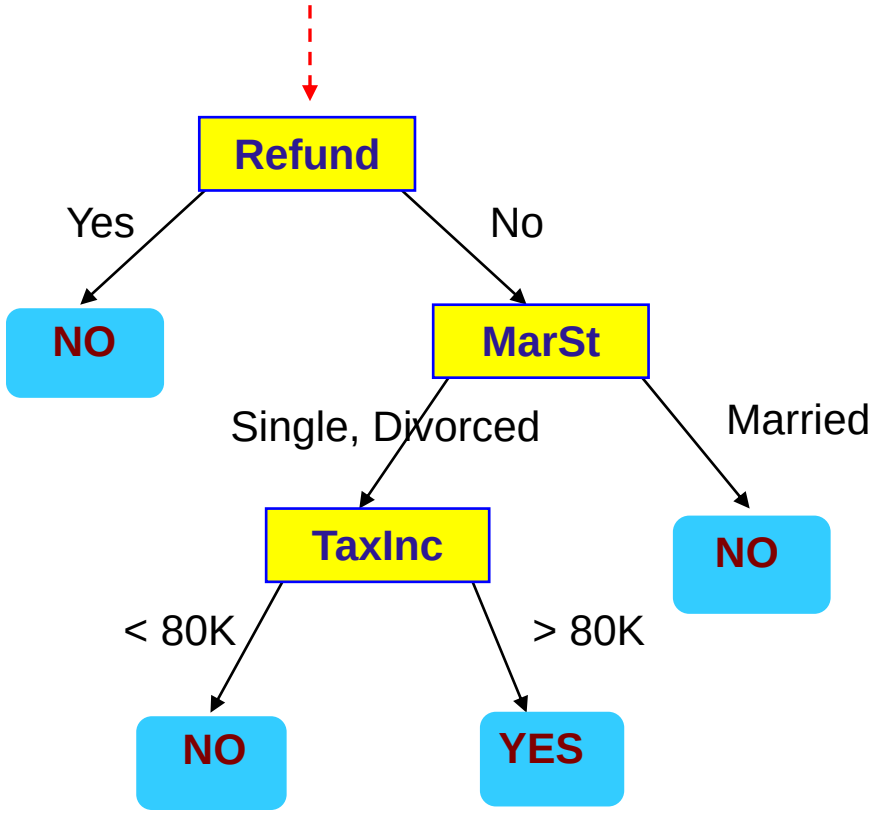
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

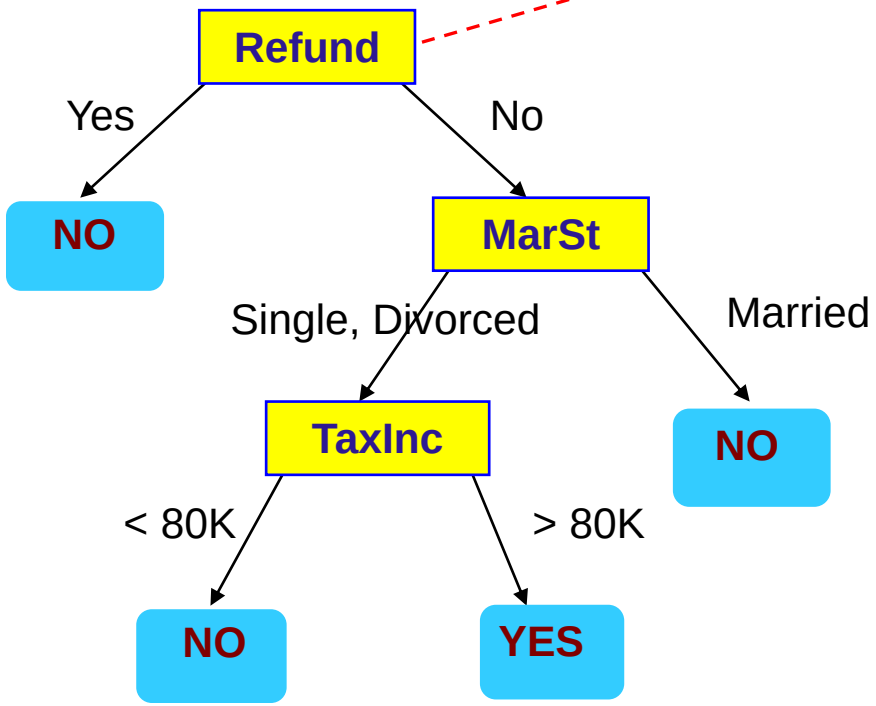
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

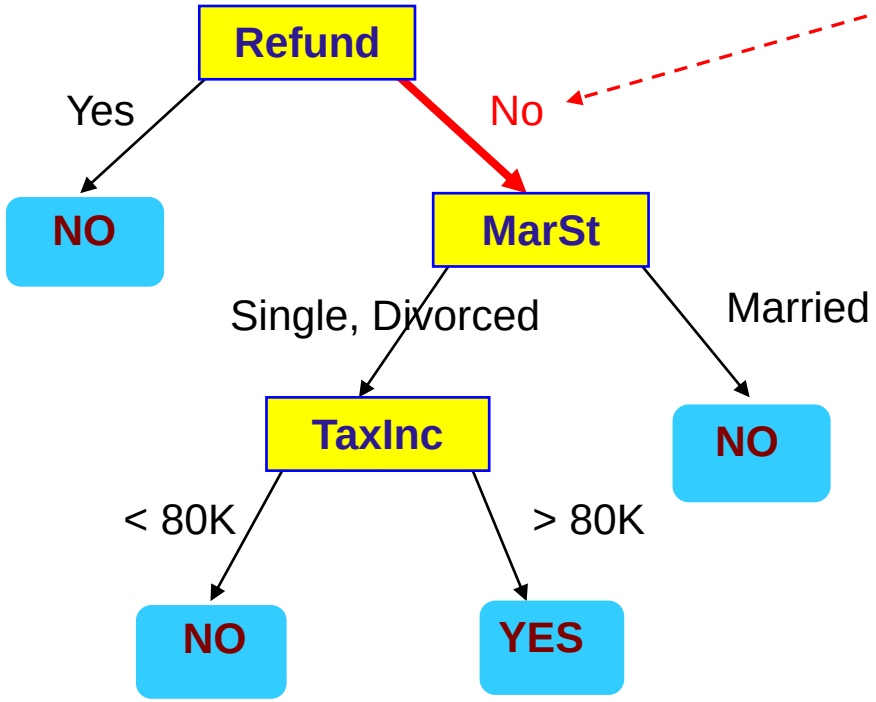
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

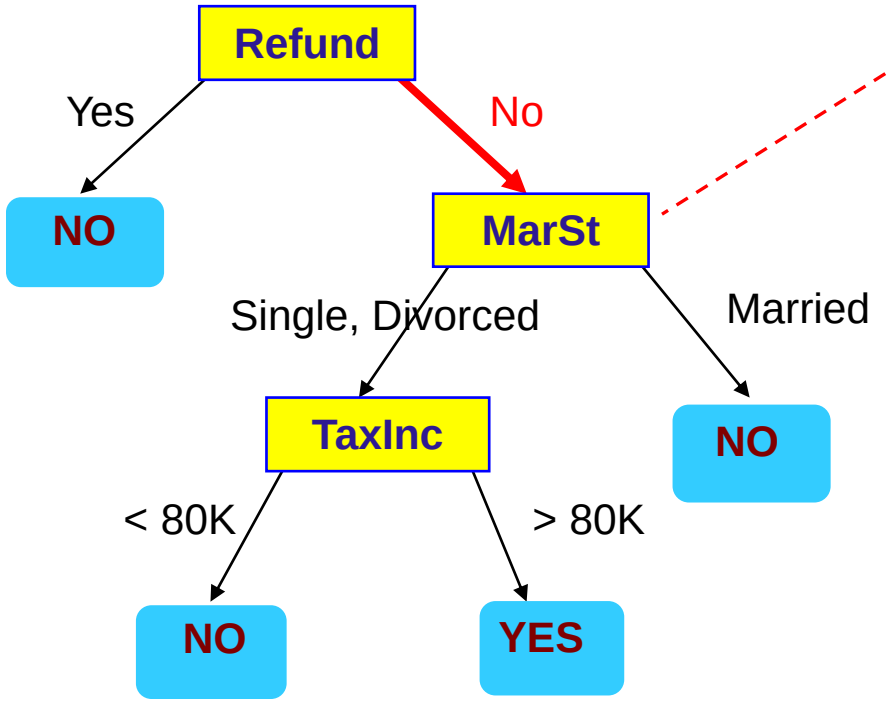
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

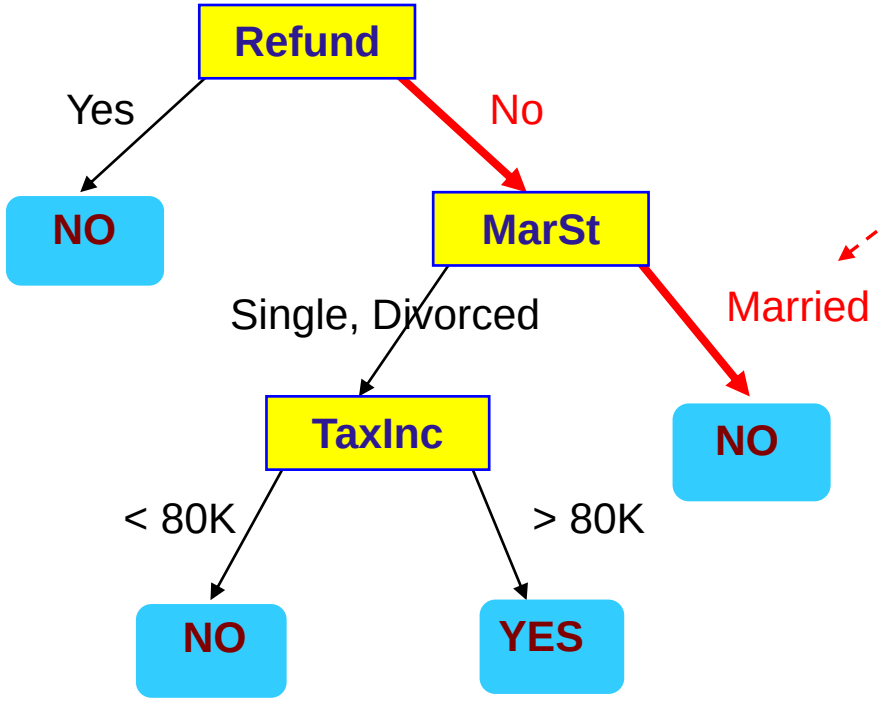
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

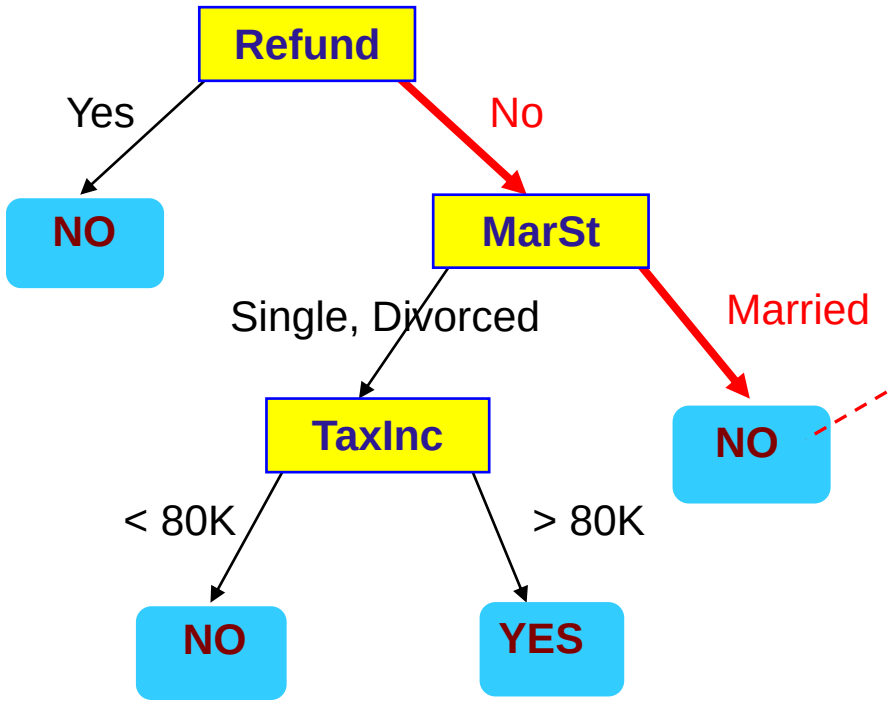
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

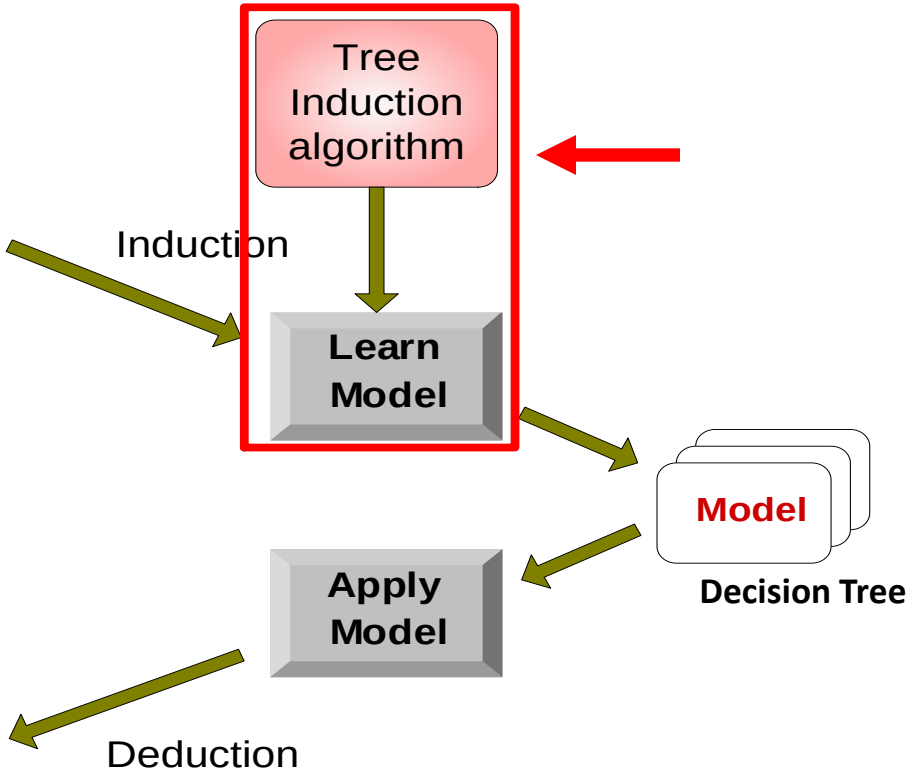
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

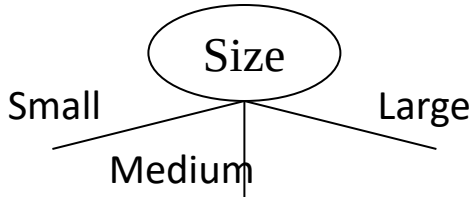
How to Specify Test Condition?

- Depends on attribute types
  - Nominal: Nominal scale is a naming scale, where variables are simply “named” or labeled, with no specific order.
  - Ordinal: Ordinal scale has all its variables in a specific order.
  - Continuous

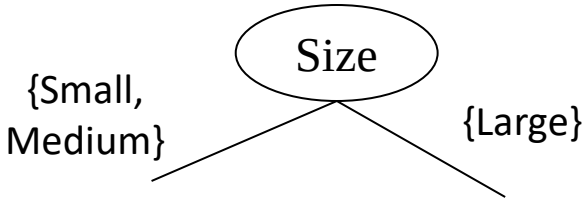
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Ordinal Attributes

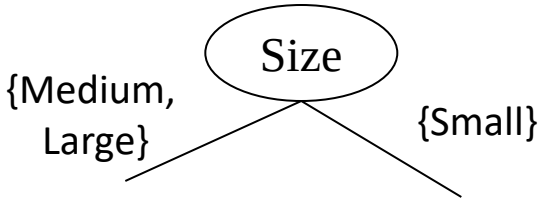
- Multi-way split: Use as many partitions as distinct values.



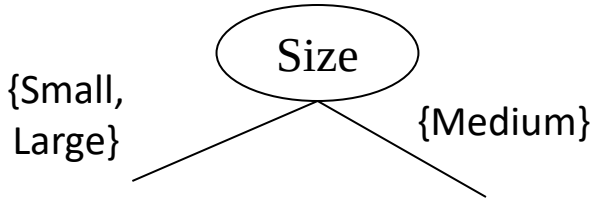
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



OR

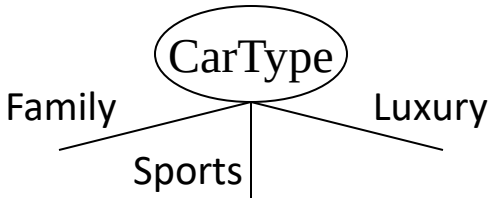


- What about this split?

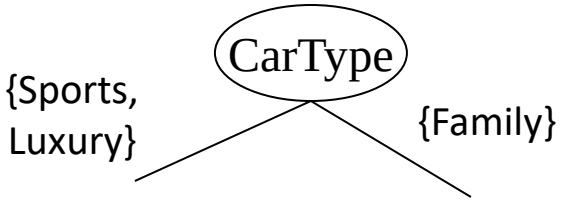


Splitting Based on Nominal Attributes

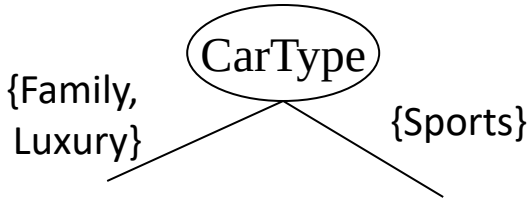
- Multi-way split: Use as many partitions as distinct values.



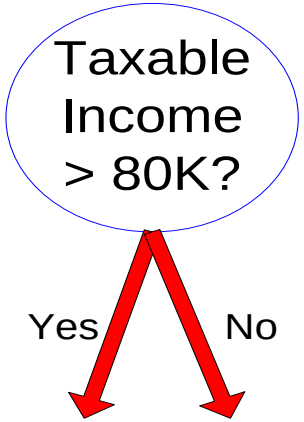
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



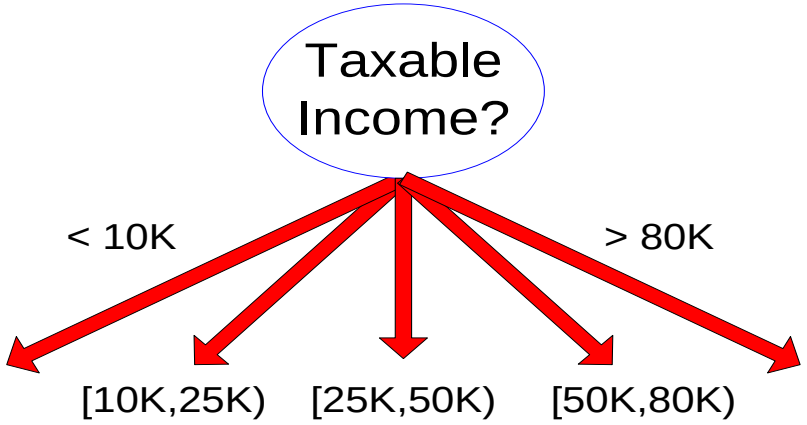
OR



Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

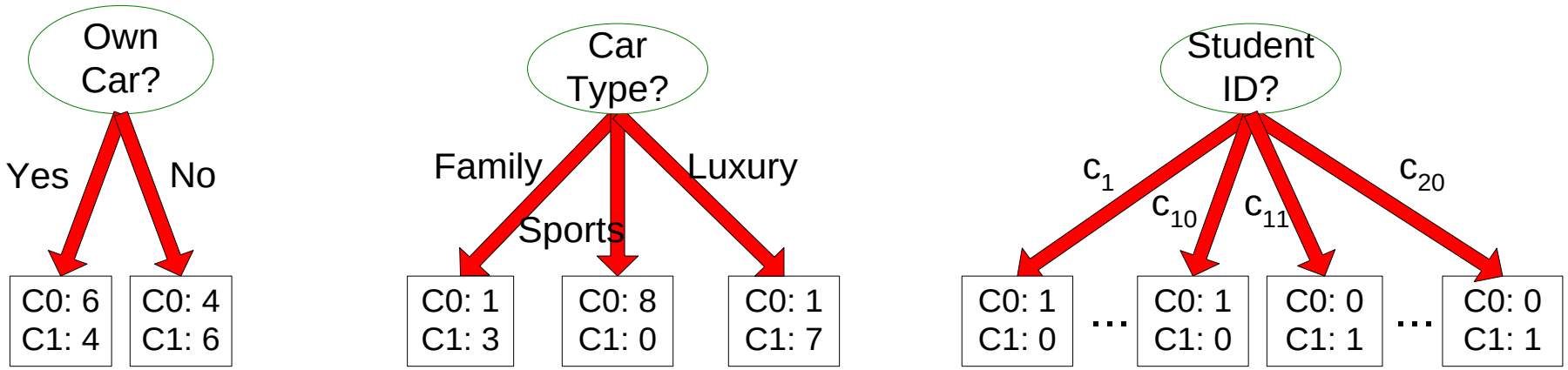
Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

Which one should be the root node?

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

  If all the elements belong to a single class, then it can be called **pure**.

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous,	**Homogeneous,**
High degree of impurity	**Low degree of impurity**

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

The degree of Gini index varies between 0 and 0.5, where,
- 0 denotes that all elements belong to a certain class or if there exists only one class
- 0.5 denotes equally distributed elements into some classes.

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

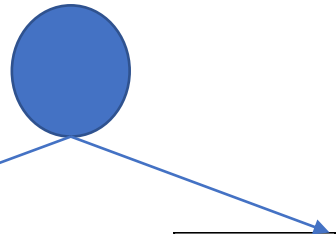
Splitting Based on GINI

- When a node p is split into k partitions (children), the **quality of split** is computed by taking the weighted average of the individual Gini indexes,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.



C1	**1**
C2	**5**

C1	**0**
C2	**6**

CART-algorithm

- In the decision tree, the nodes are split into sub-nodes on the basis of a threshold value of an attribute.
- The CART algorithm does that by searching for the best homogeneity for the sub-nodes, with the help of the Gini Index criterion.
  - The root node is taken as the training set and is split into two by considering the best attribute and threshold value.
  - Further, the subsets are also split using the same logic.
  - This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree. This is also known as Tree Pruning.

CART-greedy approach

- The input variables and the split points are selected through a greedy algorithm. Constructing a binary decision tree is a technique of splitting up the input space. A predetermined ending condition, such as a minimum number of training examples given to each leaf node of the tree, is used to halt tree building (**Stopping Criteria**).

- The input space is divided using the Greedy approach. This is known as recursive binary splitting. This is a numerical method in which all of the values are aligned and several split points are tried and assessed using a cost function, with the split with the lowest cost being chosen.

CART- A Step by Step Example

Given,
Planning Tennis
Dataset

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Task
Construct Decision
Tree using CART.

Q: Which node (attribute) will be the root node?

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Gini(Outlook=Sunny) = 1 – $(2/5)^2$ – $(3/5)^2$ = 1 – 0.16 – 0.36 = 0.48

Gini(Outlook=Overcast) = 1 – $(4/4)^2$ – $(0/4)^2$ = 0

Gini(Outlook=Rain) = 1 – $(3/5)^2$ – $(2/5)^2$ = 1 – 0.36 – 0.16 = 0.48

Then, we will calculate weighted sum of gini indexes for outlook feature.

Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342

Temperature

Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

Gini(Temp=Hot) = 1 – $(2/4)^2$ – $(2/4)^2$ = 0.5

Gini(Temp=Cool) = 1 – $(3/4)^2$ – $(1/4)^2$ = 1 – 0.5625 – 0.0625 = 0.375

Gini(Temp=Mild) = 1 – $(4/6)^2$ – $(2/6)^2$ = 1 – 0.444 – 0.111 = 0.445

We'll calculate weighted sum of gini index for temperature feature

Gini(Temp) = (4/14) x 0.5 + (4/14) x 0.375 + (6/14) x 0.445 = 0.142 + 0.107 + 0.190 = 0.439

Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Gini(Humidity=High) = 1 – $(3/7)^2$ – $(4/7)^2$ = 1 – 0.183 – 0.326 = 0.489

Gini(Humidity=Normal) = 1 – $(6/7)^2$ – $(1/7)^2$ = 1 – 0.734 – 0.02 = 0.244

Weighted sum for humidity feature will be calculated next

Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

Gini(Wind=Weak) = 1 – $(6/8)^2$ – $(2/8)^2$ = 1 – 0.5625 – 0.062 = 0.375

Gini(Wind=Strong) = 1 – $(3/6)^2$ – $(3/6)^2$ = 1 – 0.25 – 0.25 = 0.5

Gini(Wind) = (8/14) x 0.375 + (6/14) x 0.5 = 0.428

Time to decide

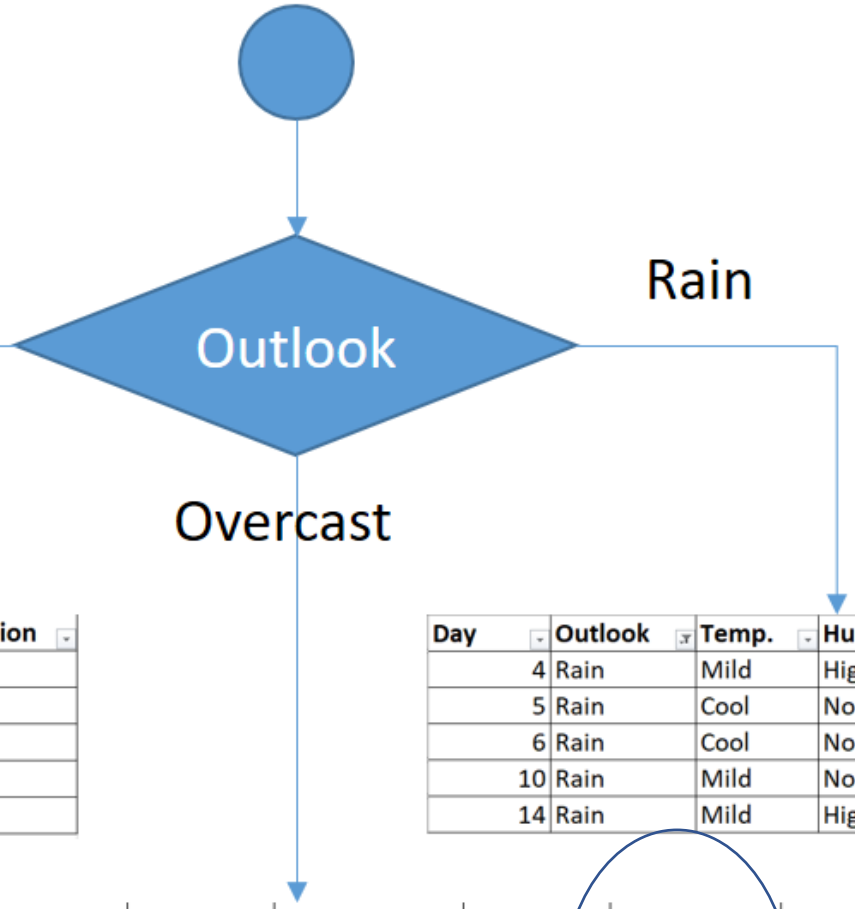
We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

We'll put outlook decision at the top of the tree.

We'll put outlook decision at the top of the tree.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



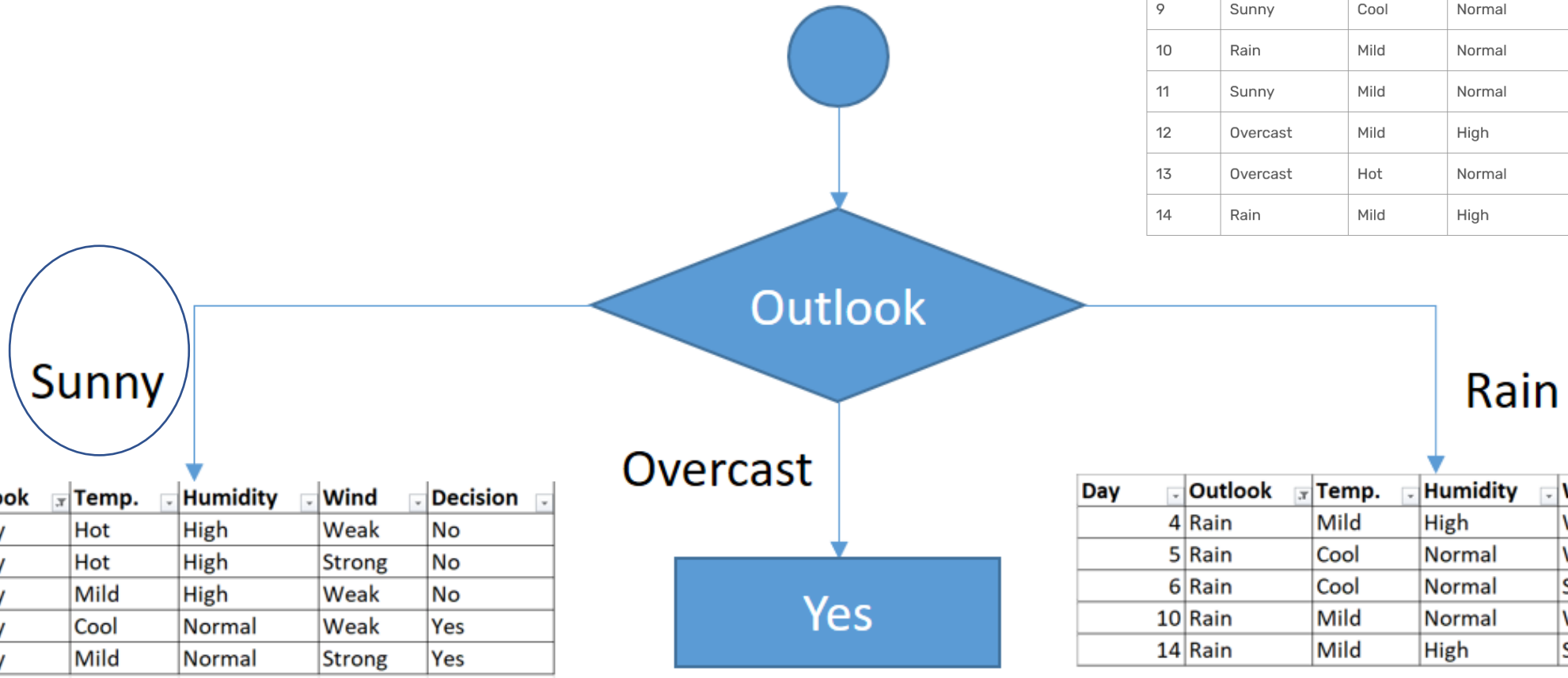
Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

You might realize that sub dataset in the overcast leaf has only yes decisions. This means that overcast leaf is over.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We will apply same principles to those sub datasets in the following steps.

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$

Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$

Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.) = $(2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2$

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$

Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$

Gini(Outlook=Sunny and Humidity) = $(3/5)x0 + (2/5)x0 = 0$

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak) = $1 - (1/3)^2 - (2/3)^2 = 0.266$

Gini(Outlook=Sunny and Wind=Strong) = $1- (1/2)^2 - (1/2)^2 = 0.2$

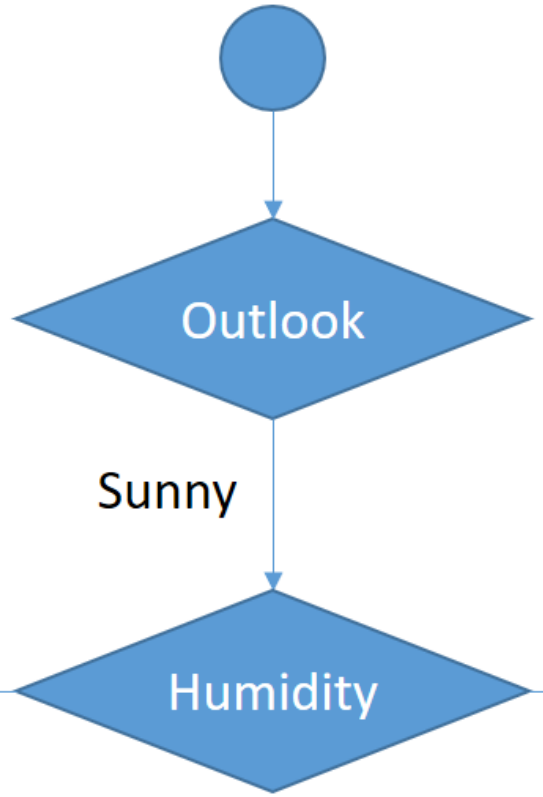
Gini(Outlook=Sunny and Wind) = $(3/5)x0.266 + (2/5)x0.2 = 0.466$

Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

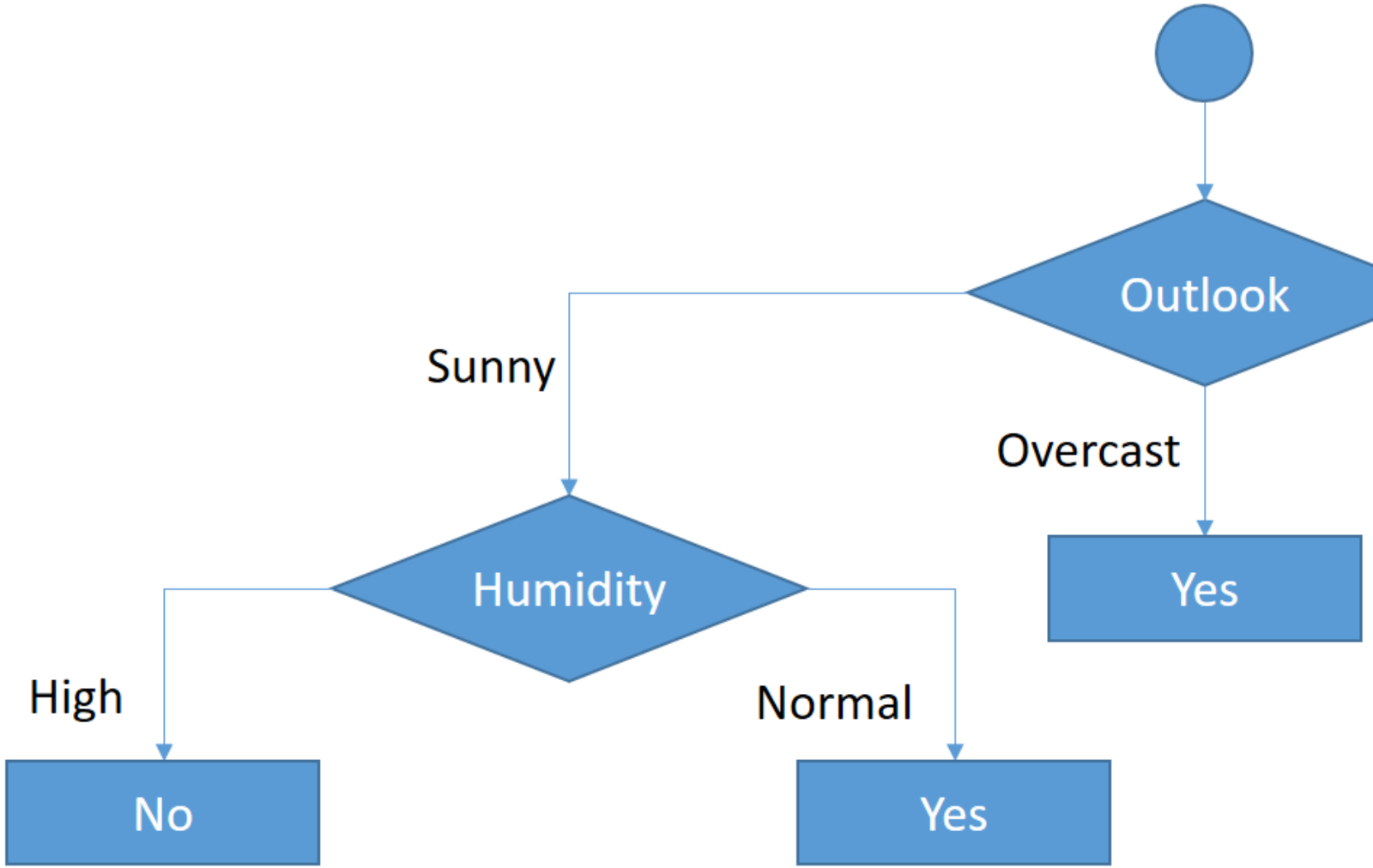
Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

As seen, decision is always no for high humidity and sunny outlook.
On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Now, we need to focus on rain outlook.

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Temp.=Mild) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Temp.) = $(2/5)x0.5 + (3/5)x0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Humidity=Normal) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Humidity) = $(2/5)x0.5 + (3/5)x0.444 = 0.466$

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$

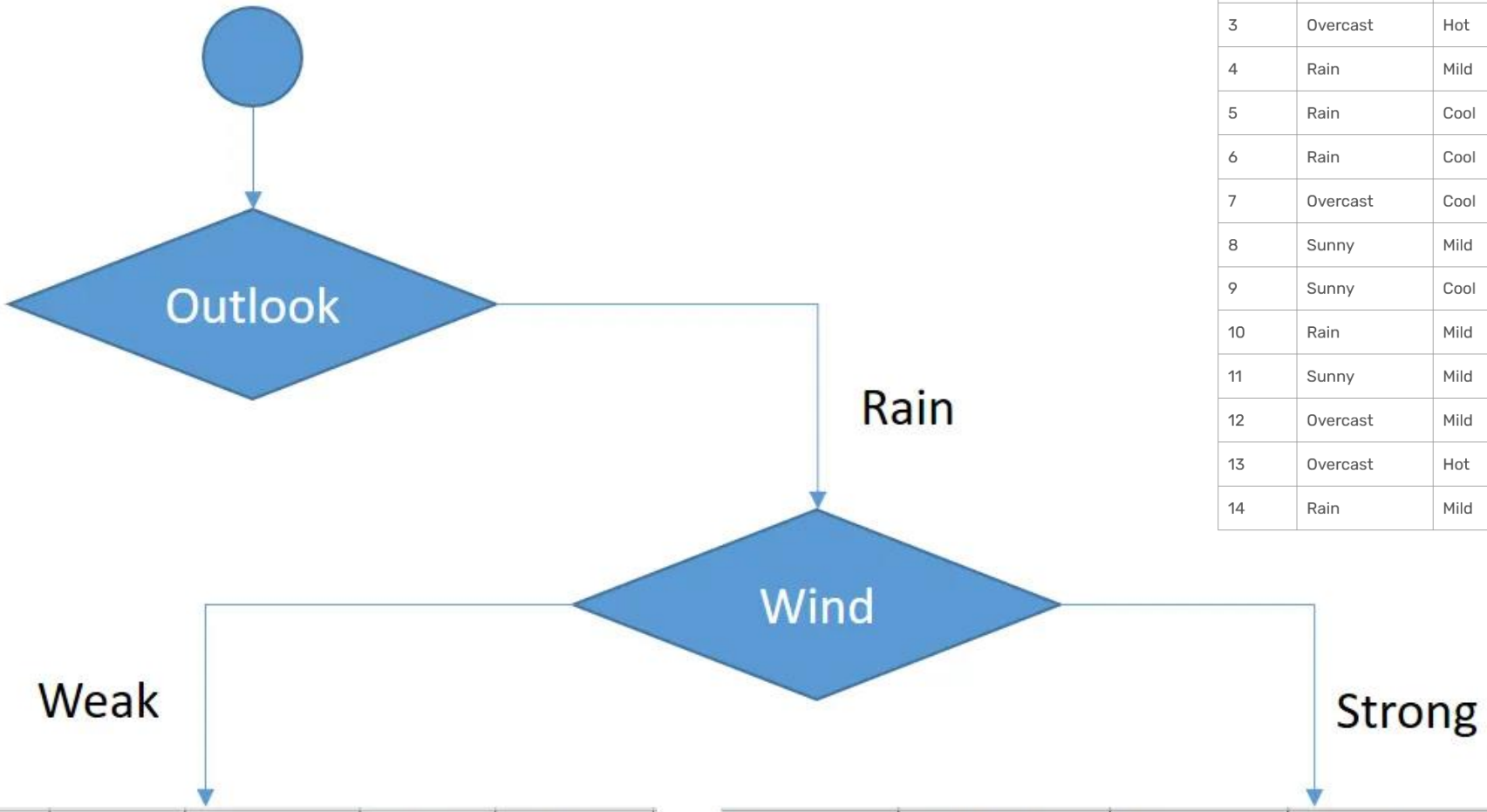
Gini(Outlook=Rain and Wind) = $(3/5)x0 + (2/5)x0 = 0$

Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

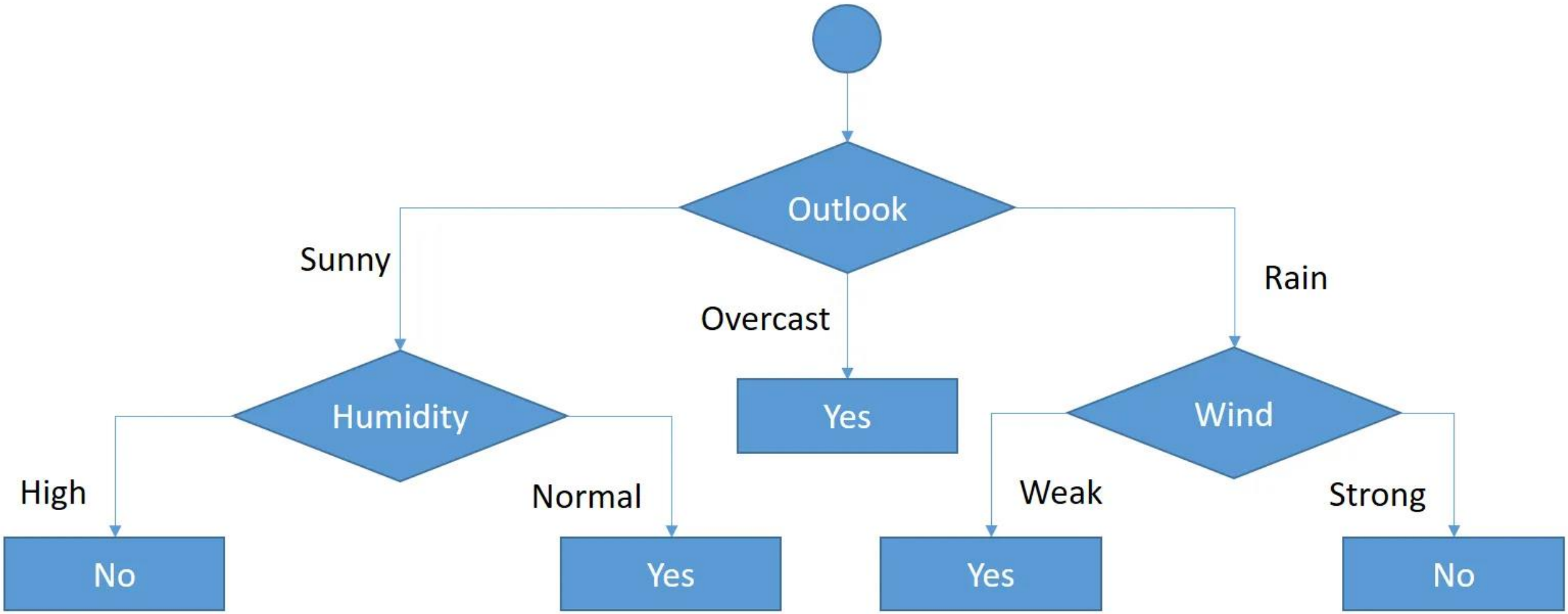
Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

As seen, decision is always yes when wind is weak. On the other hand, decision is always no if wind is strong. This means that this branch is over.



How to define a node for Numeric Data?

	Weight	Heart Disease
Lowest	155	No
	180	Yes
	190	No
	220	Yes
Highest	225	Yes

Step 1) Sort the patients by weight, lowest to highest.

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

Step 2) Calculate the average weight for all adjacent patients.

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

Step 3) Calculate the impurity values for each average weight.

167.5 → Gini impurity = ?

185 → Gini impurity = ?

205 → Gini impurity = ?

222.5 → Gini impurity = ?

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

Weight < 167.5

Heart Disease	
Yes	No
0	1

Gini impurity = 0

Heart Disease	
Yes	No
3	1

0.336

Gini impurity for Weight < 167.5 is the weighted average of the impurities for the two leaves.

$$= \left(\frac{1}{1+4}\right) 0 + \left(\frac{4}{1+4}\right) 0.336 = 0.3$$

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

Gini impurity = 0.3

Gini impurity = 0.47

Gini impurity = 0.27

Gini impurity = 0.4

The lowest impurity occurs when we separate using **weight < 205**…

…so this is the cutoff and impurity value we will use when we compare weight to chest pain or blocked arteries.

Decision Tree

ID3 Algorithm

C4.5 Algorithm

ID3: Introduction

- Iterative Dichotomizer: A mathematical algorithm for building the decision tree.
- Invented by J. Ross Quinlan in 1979.
- Uses Information Theory invented by Shannon in 1948.
- Builds the tree from the top down, with no backtracking.
- Information Gain is used to select the most useful attribute for classification.

Entropy

- Entropy measures the randomness/uncertainty in the data
- Let's consider a set S of examples with C many classes. Entropy of this set:

$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

$$\boxed{Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)}$$

- p_c is the probability that an element of S belongs to class c
  - .. basically, the fraction of elements of S belonging to class c
- Intuition: Entropy is a measure of the "degree of surprise"
  - Some dominant classes $\Longrightarrow$ small entropy (less uncertainty)
  - Equiprobable classes $\Longrightarrow$ high entropy (more uncertainty)
- Entropy denotes the average number of bits needed to encode S

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Useful Note: Calculating Entropy with Code

- Most programming languages and calculators do not have a $\log_2$ function.
- Use a conversion factor
- Take log function of 2, and divide by it.
- Example: $\log_{10}(2) \ = \ .301$
- Then divide to get $\log_2(n)$: [n=3/5]
- $\log_2\left(\frac{3}{5}\right) = \log_{10}(3/5)\ /\ .301$

- Taking $\log_{10}$(0) produces an error.
  - Substitute 0 for (0/3) $\log_{10}$(0/3)
  - Do not try to calculate $\log_{10}$(0/3)

Splitting Based on Information-GAIN [Quality of Split]

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)

- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

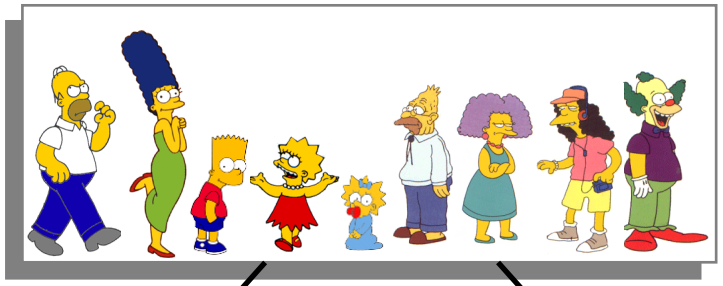
Information Gain (IG)

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Which attribute creates the most homogeneous branches?
- First the entropy of the total dataset is calculated.
- The dataset is then split on the different attributes.
- The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split.
- The resulting entropy is subtracted from the entropy before the split.
- The result is the Information Gain, or decrease in entropy.
- The attribute that yields the largest IG is chosen for the decision node.

Information Gain (cont'd)

- A branch set with entropy of 0 is a leaf node.
- Otherwise, the branch needs further splitting to classify its dataset.
- The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

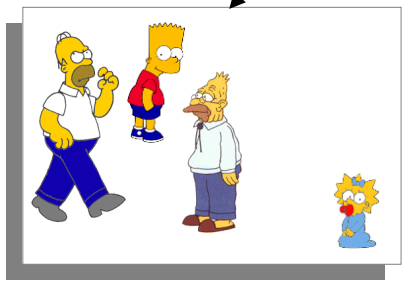
Another Example



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$
$= \mathbf{0.9911}$

yes — Hair Length <= 5? — no



$Entropy(1\mathbf{F},3\mathbf{M}) = -(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$
$= \mathbf{0.8113}$

$Entropy(3\mathbf{F},2\mathbf{M}) = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5)$
$= \mathbf{0.9710}$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$Gain(\text{Hair Length <= 5}) = \mathbf{0.9911} - (4/9 * \mathbf{0.8113} + 5/9 * \mathbf{0.9710}) = \mathbf{0.0911}$

An Example of Calculating Entropy and Information-Gain

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Information Gain(T,X) = Entropy(T) - Entropy(T, X)

IG(PlayGolf, Outlook) = E(PlayGolf) - E(PlayGolf, Outlook)
= 0.940 - 0.693
= 0.247

Lets complete a full example for ID3 algorithm

- First Select the root based on highest Gain.
- Iteratively select the feature with the highest I.G. for each child of previous node.

$$\text{Entropy}(S) = \sum - p(I) \,.\, \log_2 p(I)$$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum [\, p(S|A) \,.\, \text{Entropy}(S|A) \,]$$

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Entropy

We need to calculate the entropy first. Decision column consists of 14 instances and includes two labels: yes and no. There are 9 decisions labeled yes, and 5 decisions labeled no.

$$\text{Entropy(Decision)} = -\,p(\text{Yes}) \,.\, \log_2 p(\text{Yes}) - p(\text{No}) \,.\, \log_2 p(\text{No})$$

$$\text{Entropy(Decision)} = -\,(9/14) \,.\, \log_2(9/14) - (5/14) \,.\, \log_2(5/14) = 0.940$$

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Now, we need to find the most dominant factor for decisioning.
We start with **Wind.**

Strong wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

Here, there are 6 instances for strong wind. Decision is divided into two equal parts.

1- Entropy(Decision|Wind=Strong) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes)

2- Entropy(Decision|Wind=Strong) = – (3/6) . $\log_2$(3/6) – (3/6) . $\log_2$(3/6) = 1

Weak wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

There are 8 instances for weak wind. Decision of 2 items are no and 6 items are yes as illustrated below.

1- Entropy(Decision|Wind=Weak) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes)

2- Entropy(Decision|Wind=Weak) = – (2/8) . $\log_2$(2/8) – (6/8) . $\log_2$(6/8) = 0.811

Now, we can turn back to Gain(Decision, Wind) equation.

Gain(Decision, Wind) = Entropy(Decision) – [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] – [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)] = 0.940 – [(8/14) . 0.811] – [(6/14). 1] = 0.048

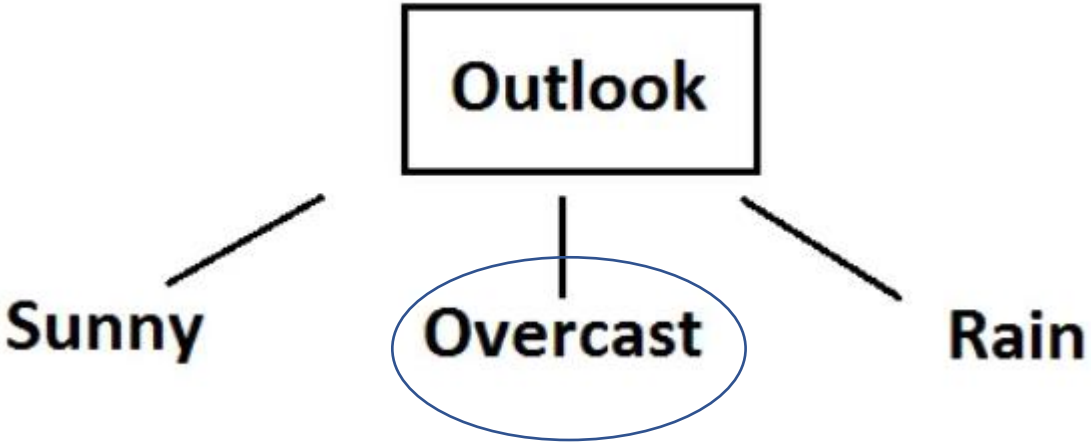
Other factors on decision

We have applied similar calculation on the other columns.

1- Gain(Decision, Outlook) = 0.246 Highest

2- Gain(Decision, Temperature) = 0.029

3- Gain(Decision, Humidity) = 0.151



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Overcast outlook on decision

Basically, decision will always be yes if outlook were overcast.

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Sunny outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Here, there are 5 instances for sunny outlook. Decision would be probably 3/5 percent no, 2/5 percent yes.

1- Gain(Outlook=Sunny|Temperature) = 0.570

2- Gain(Outlook=Sunny|Humidity) = 0.970

3- Gain(Outlook=Sunny|Wind) = 0.019

Now, humidity is the decision because it produces the highest score if outlook were sunny.

At this point, decision will always be no if humidity were high.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

On the other hand, decision will always be yes if humidity were normal

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Finally, it means that we need to check the humidity and decide if outlook were sunny.

Rain outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

1- Gain(Outlook=Rain | Temperature) = 0.01997309402197489

2- Gain(Outlook=Rain | Humidity) = 0.01997309402197489

3- Gain(Outlook=Rain | Wind) = 0.9709505944546686

Here, wind produces the highest score if outlook were rain. That's why, we need to check wind attribute in 2nd level if outlook were rain.

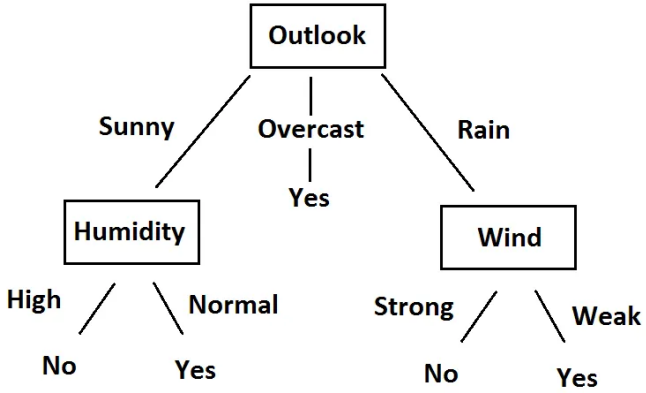
So, it is revealed that decision will always be yes if wind were weak and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

What's more, decision will be always no if wind were strong and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

So, decision tree construction is over. We can use the following rules for decisioning.



Final version of decision tree

Advantages of using ID3

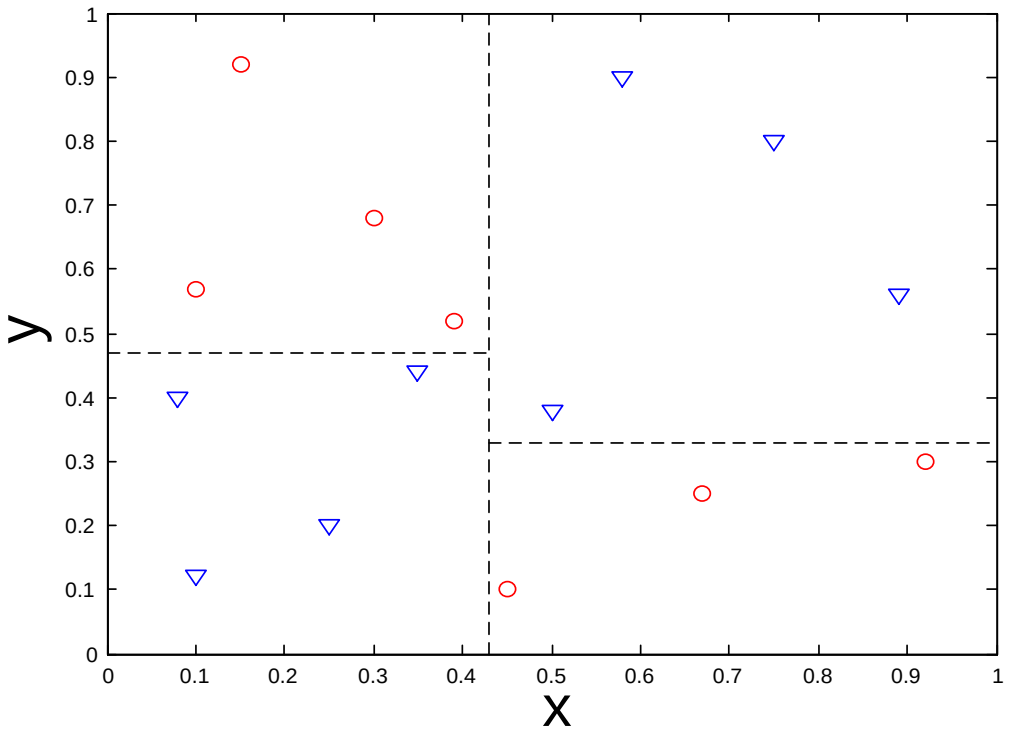
- Understandable prediction rules are created from the training data.
- Builds the fastest tree.
- Builds a short tree.
- Only need to test enough attributes until all data is classified.
- Finding leaf nodes enables test data to be pruned, reducing number of tests.
- Whole dataset is searched to create tree.

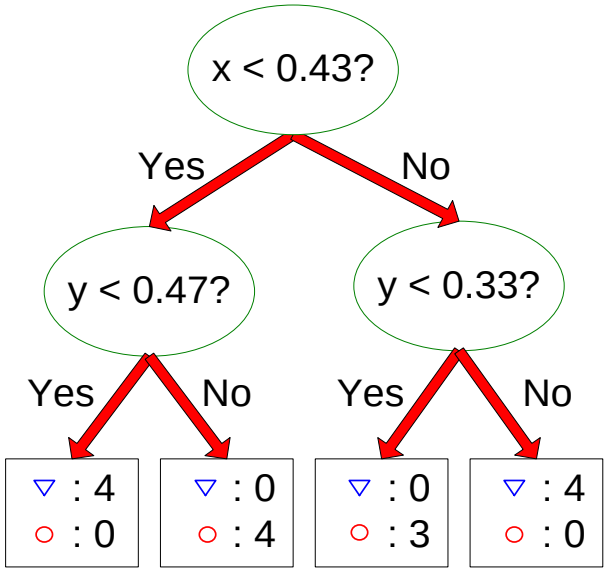
Disadvantages of using ID3

- Data may be over-fitted or over-classified, if a small sample is tested.
- Only one attribute at a time is tested for making a decision.
- Classifying continuous data may be computationally expensive, as many trees must be generated to see where to break the continuum.

C4.5 Algorithm: A successor of ID3

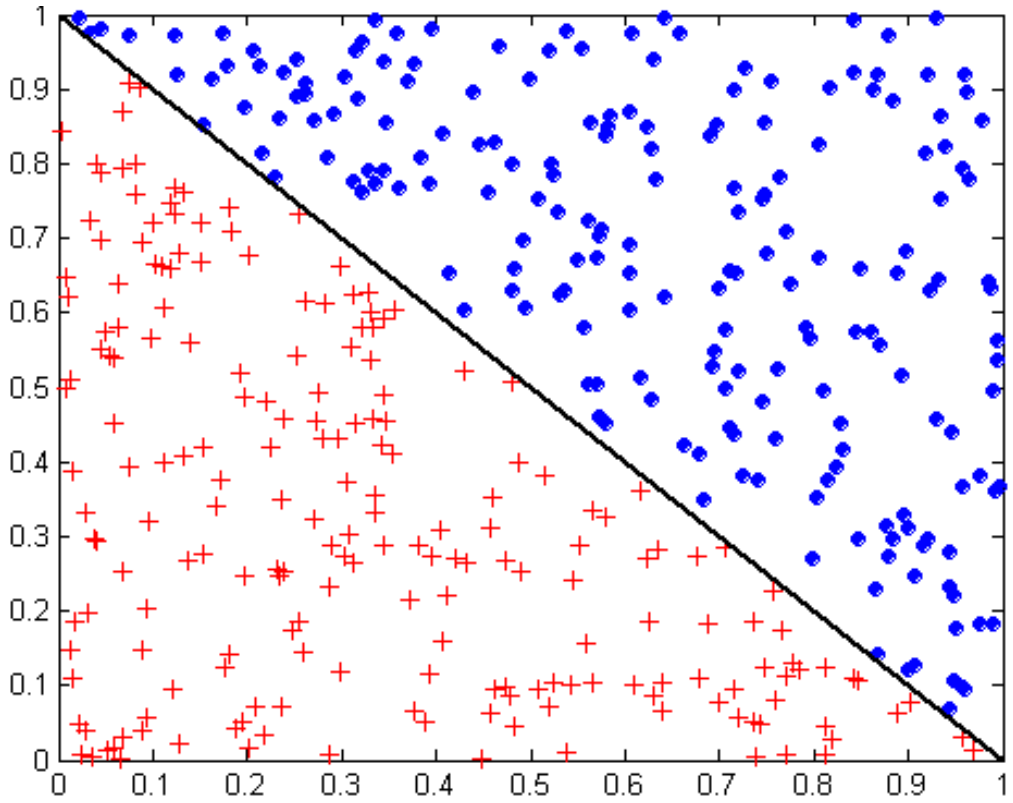
A bit about Decision Boundary

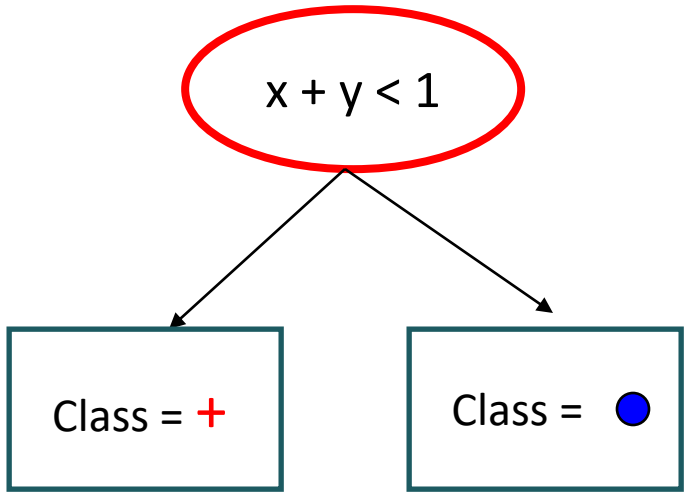




- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

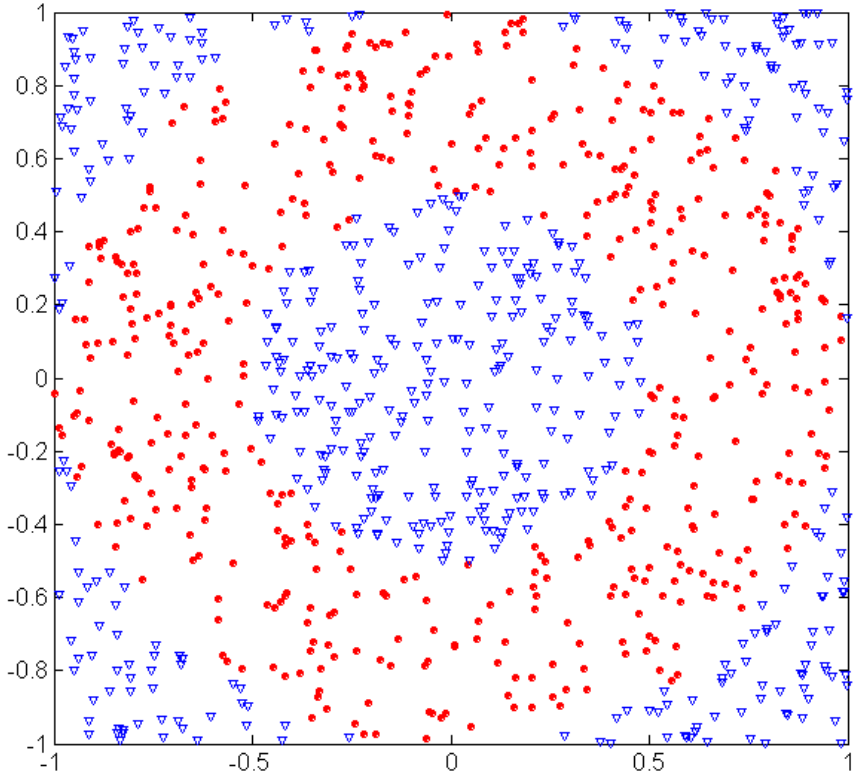
Linear Boundary





- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Vertical/Horizontal Boundaries



500 circular and 500 triangular data points.

Circular points:

$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

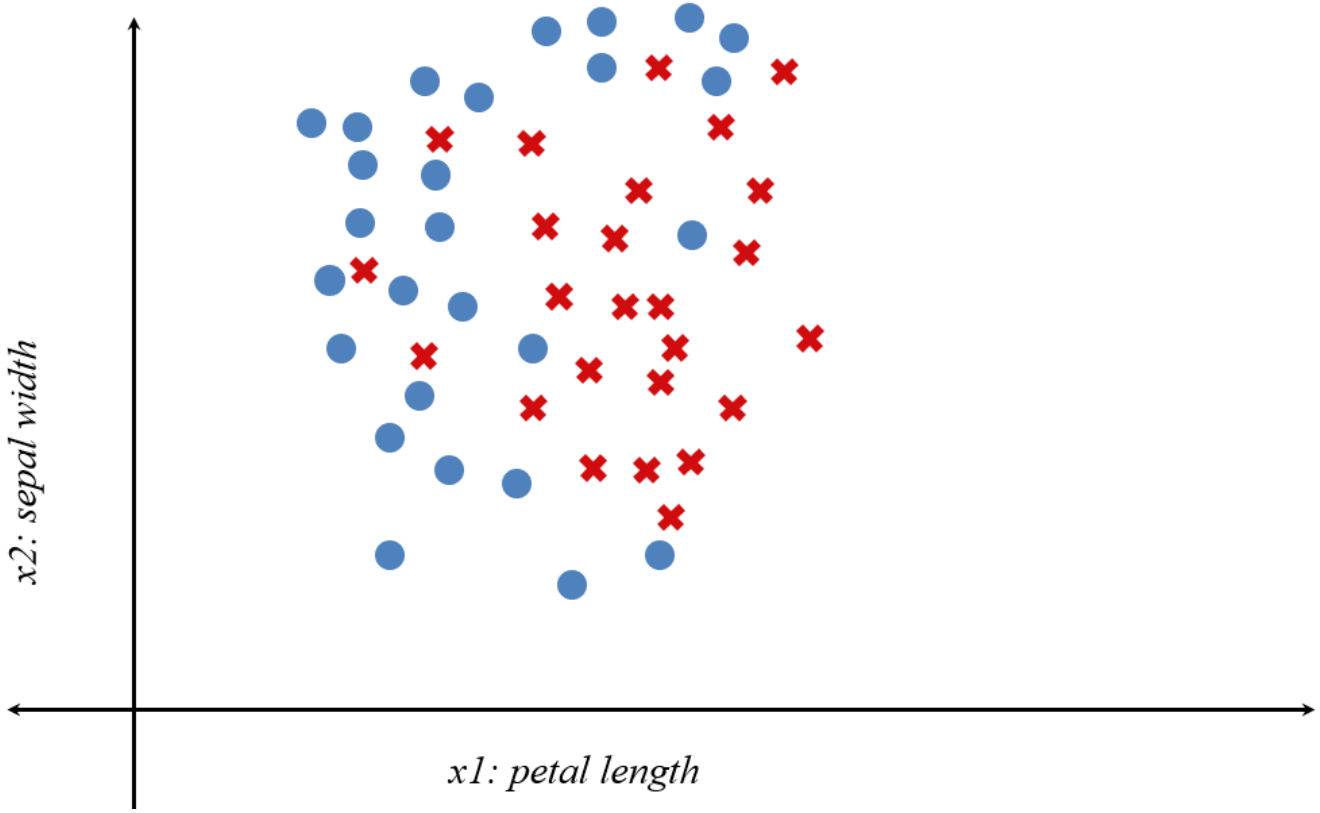
Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

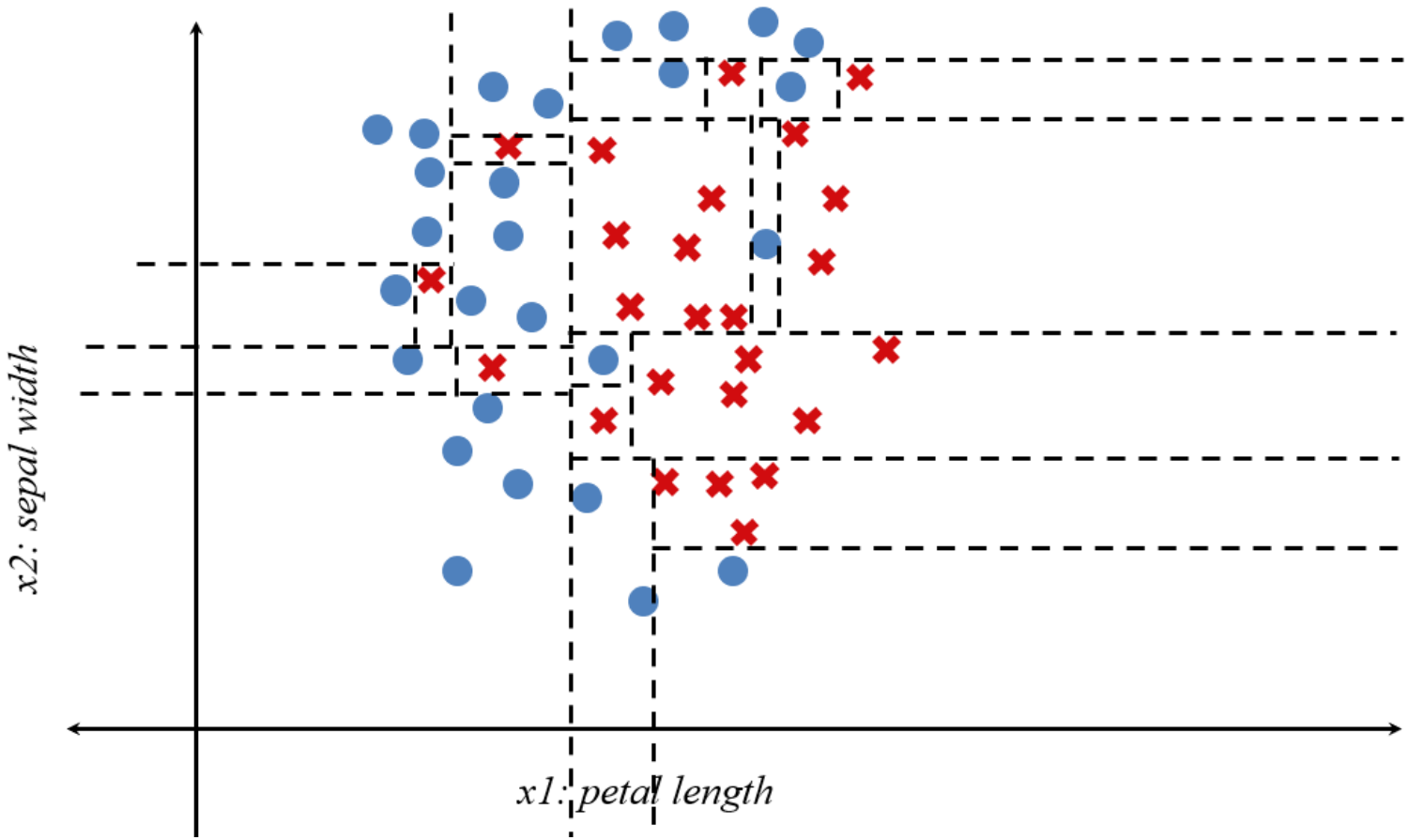
$\text{sqrt}(x_1^2+x_2^2) < 1$

DTs in practice…

Growing to purity is bad (*overfitting*)

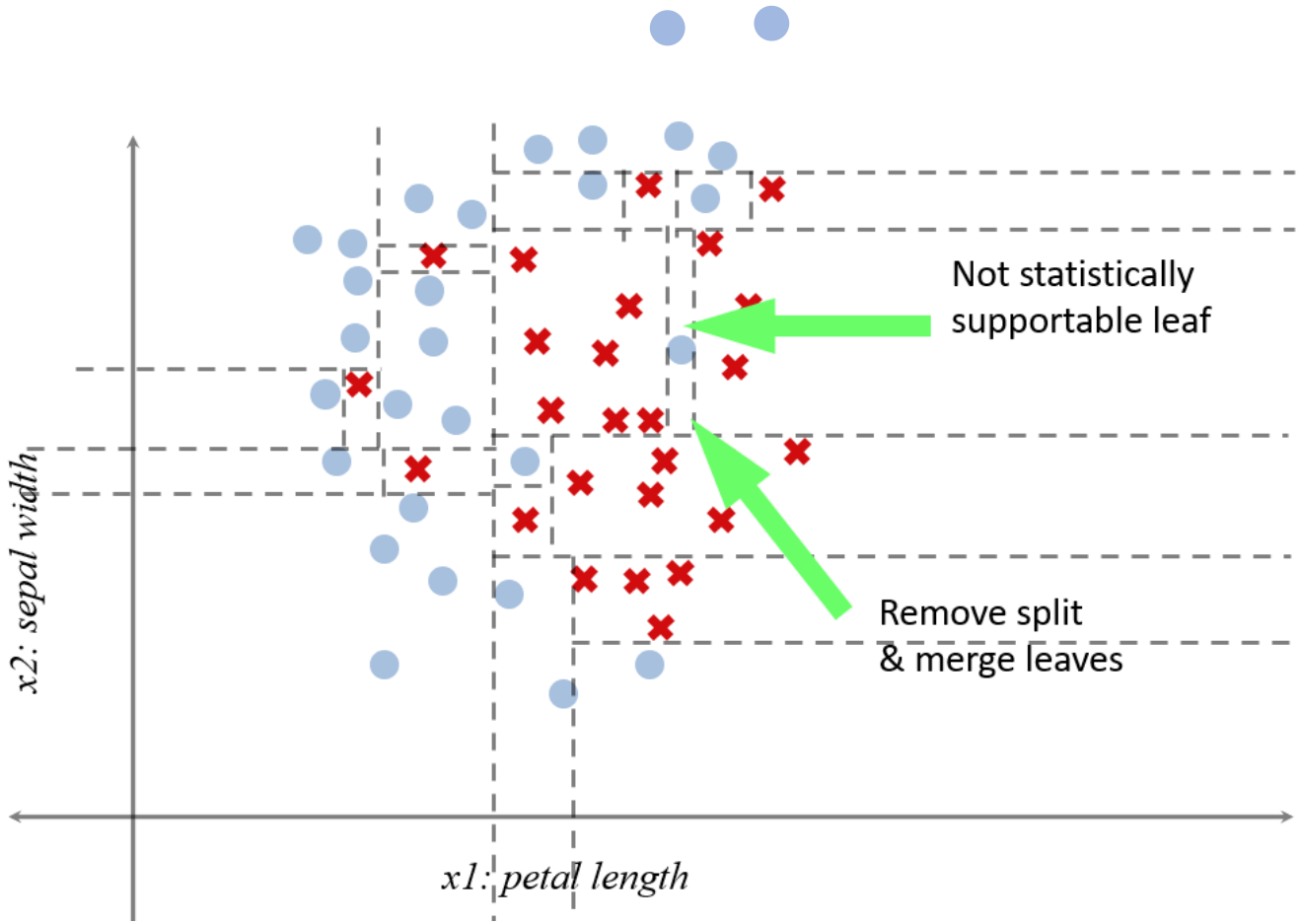


DTs in practice…



DTs in practice…

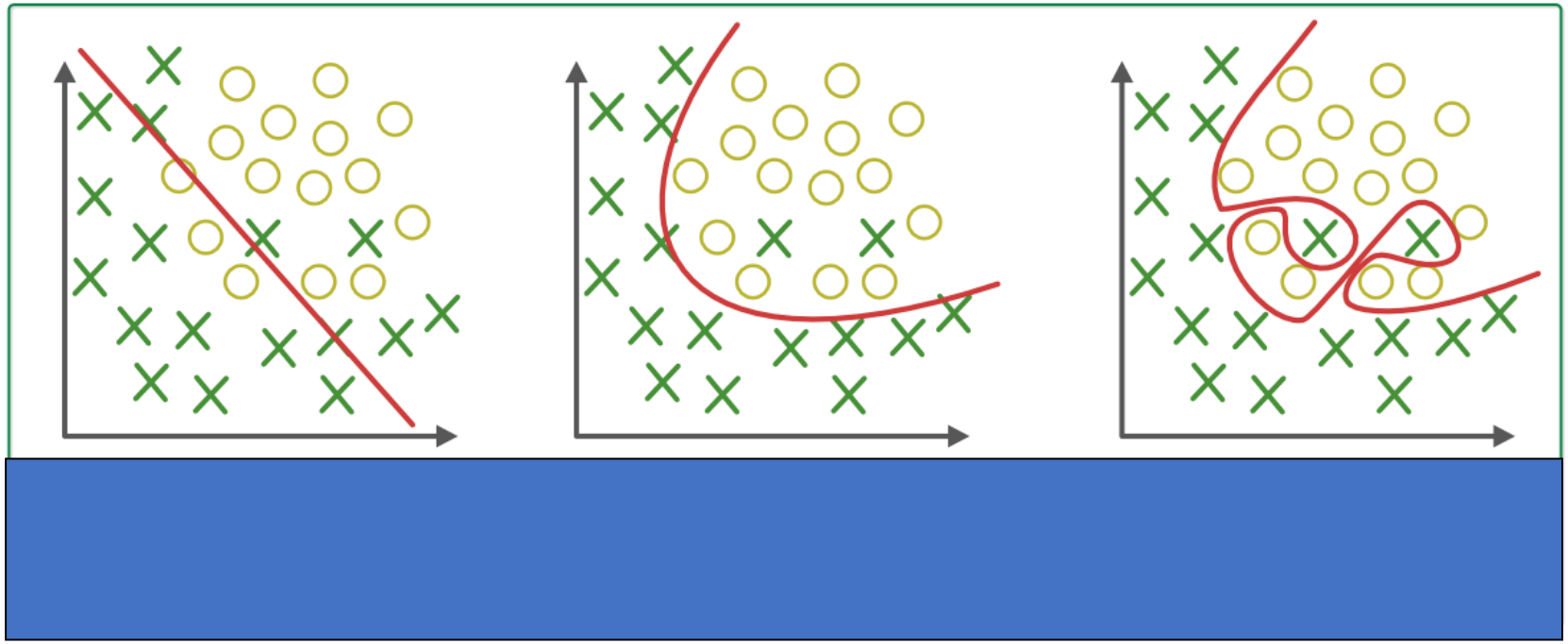
- Growing to purity is bad (*overfitting*)

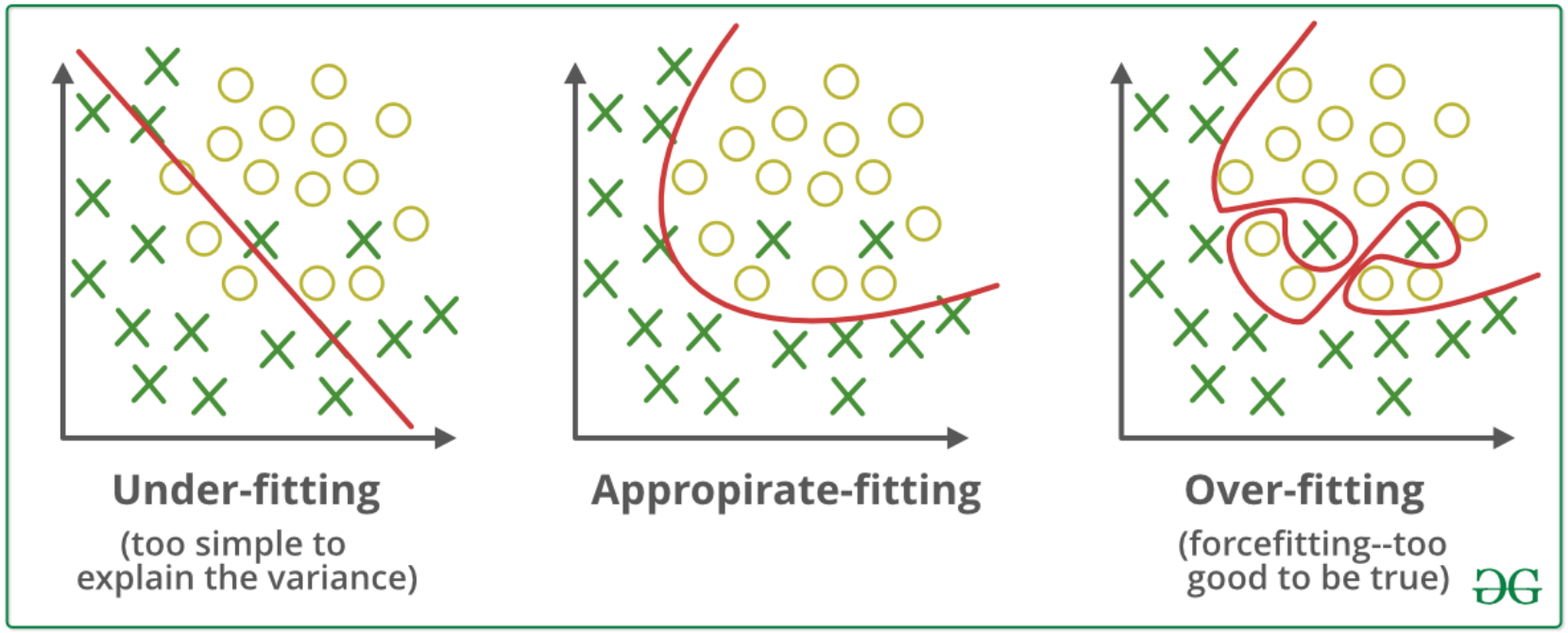


DTs in practice…

- Growing to purity is bad (*overfitting*)
  - Terminate growth early
  - Grow to purity, then *prune* back

Question: Which one of the following is good-fit/underfit/overfit?




Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

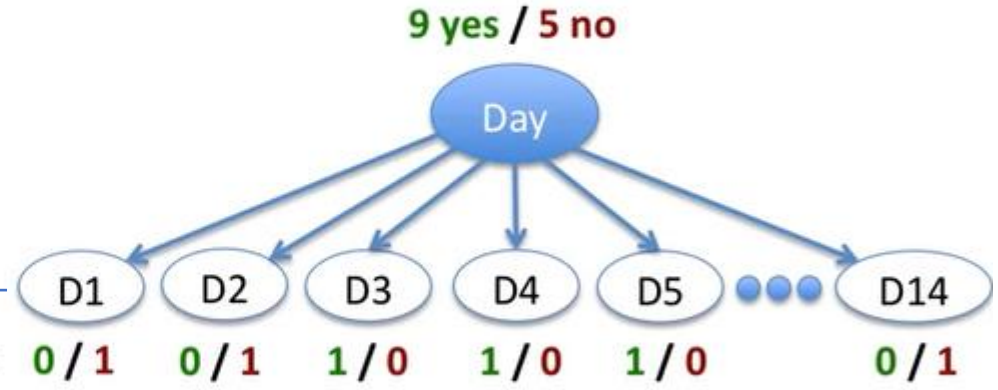
Avoid Overfitting in Decission-Tree

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - **Prepruning**: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning**: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Algorithm C 4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotometer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact biased towards splitting attribute having a large number of outcomes.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
  - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is $E_A(D) = 0$

Problems with Information Gain



$Entropy = 0$

all subsets perfectly pure => optimal split

Information Gain (I.G.)
will be maximum in this case.

ID3 is biased towards attributes with many values.

Question 1: How I.G. will treat this node?
Question 2: Will the tree size increase/decrease if choose this as the root node?
Question 3: Can we normalize the I.G. to tackle such cases?

Limitations of ID3

- Although, the previous situation is an extreme case, intuitively, we can infer that ID3 favours splitting attributes having a large number of values
  - compared to other attributes, which have a less variations in their values.
- Such a partition appears to be useless for classification.
- This type of problem is called **overfitting problem**.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as β.
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C 4.5 : Gain Ratio

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

$$w_j = \frac{number\ of\ values\ only\ for\ subset\ 'j'}{total\ numbers\ of\ values}$$

- *SplitInfo* is also referred as ***SplitEntropy/SplitInformation*** in the literature.
- It can be interpreted like – does this attribute contains lots of tiny subsets, or not~
- It will have low value if we have few subsets, and vice-verse.

Physical Interpretation of $E_A^*(\mathrm{D})$ [SplitInfo]

Example 9.18 : Split information $E_A^*(\mathrm{D})$

- To illustrate $E_A^*(\mathrm{D})$, let us examine the case where there are 32 instances and splitting an attribute A which has a_1, a_2, a_3 and a_4 sets of distinct values.
  - Distribution 1 : Highly non-uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	32	0	0	0

$$E_A^*(\mathrm{D}) = -\frac{32}{32} log_2(\frac{32}{32}) = -log_2 1 = 0$$

  - Distribution 2

	a_1	a_2	a_3	a_4
Frequency	16	16	0	0

$$E_A^*(\mathrm{D}) = -\frac{16}{32} log_2(\frac{16}{32}) - \frac{16}{32} log_2(\frac{16}{32}) = log_2 2 = 1$$

Physical Interpretation of $E_A^*(\mathrm{D})$ [SplitInfo]

- Distribution 3

	a_1	a_2	a_3	a_4
Frequency	16	8	8	0

$$E_A^*(\mathrm{D}) = -\frac{16}{32}log_2(\frac{16}{32}) - \frac{8}{32}log_2(\frac{8}{32}) - \frac{8}{32}log_2(\frac{8}{32}) = 1.5$$

- Distribution 4

	a_1	a_2	a_3	a_4
Frequency	16	8	4	4

$$E_A^*(\mathrm{D}) = 1.75$$

- Distribution 5: Uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	8	8	8	8

$$E_A^*(\mathrm{D}) = (-\frac{8}{32}log_2(\frac{8}{32}))*4 = -log_2(\frac{1}{4}) = 2.0$$

Physical Interpretation of $E_A^*(\mathrm{D})$ [SplitInfo]

- In general, if there are m attribute values, each occurring equally frequently, then the split information is $log_2 m$.
- Based on the Example 9.18, we can summarize our observation on split information as under:
  - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.
  - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.
  - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $log_2 m$ (both inclusive)

Physical Interpretation of $\beta(A, B)$ [Gain-Ratio]

- Information gain signifies how much information will be gained on partitioning the values of attribute A
  - Higher information gain means splitting of A is more desirable.
- On the other hand, split information forms the denominator in the gain ratio formula.
  - This implies that higher the value of split information is, lower the gain ratio.
  - In turns, it decreases the information gain.
- <u>Further, information gain is large when there are many distinct attribute values.</u>
  - <u>When many distinct values, split information is also a large value.</u>
  - <u>This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.</u>
- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.
- C4.5 follows the exact same algorithm as ID3, except using Gain-ratio instead of Information-Gain.

An Example of calculating Gain-ratio for Wind attribute

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

$$\text{Entropy(Decision)} = \sum - p(I) \,.\, \log_2 p(I) = - p(\text{Yes}) \,.\, \log_2 p(\text{Yes}) - p(\text{No}) \,.\, \log_2 p(\text{No}) = - (9/14) \,.\, \log_2(9/14) - (5/14) \,.\, \log_2(5/14) = 0.940$$

$$\text{GainRatio}(A) = \text{Gain}(A) \,/\, \text{SplitInfo}(A)$$

$$\text{SplitInfo}(A) = -\sum |Dj|/|D| \times \log_2 |Dj|/|D|$$

Wind Attribute

Wind is a nominal attribute. Its possible values are weak and strong.

Gain(Decision, Wind) = Entropy(Decision) − ∑ (p(Decision|Wind) . Entropy(Decision|Wind))

Gain(Decision, Wind) = Entropy(Decision) − [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] + [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)]

There are 8 weak wind instances. 2 of them are concluded as no, 6 of them are concluded as yes.

Entropy(Decision|Wind=Weak) = − p(No) . $\log_2$p(No) − p(Yes) . $\log_2$p(Yes) = − (2/8) . $\log_2$(2/8) − (6/8) . $\log_2$(6/8) = 0.811

Entropy(Decision|Wind=Strong) = − (3/6) . $\log_2$(3/6) − (3/6) . $\log_2$(3/6) = 1

Gain(Decision, Wind) = 0.940 − (8/14).(0.811) − (6/14).(1) = 0.940 − 0.463 − 0.428 = 0.049

There are 8 decisions for weak wind, and 6 decisions for strong wind.

SplitInfo(Decision, Wind) = -(8/14).$\log_2$(8/14) − (6/14).$\log_2$(6/14) = 0.461 + 0.524 = 0.985

GainRatio(Decision, Wind) = Gain(Decision, Wind) / SplitInfo(Decision, Wind) = 0.049 / 0.985 = 0.049

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.

2. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.

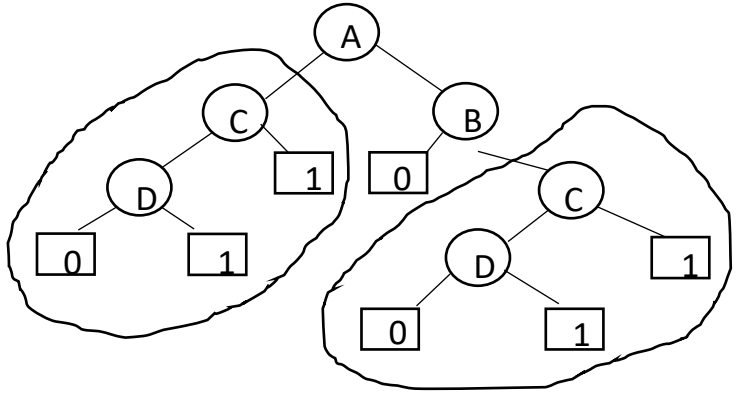
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

4. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of O(d), where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

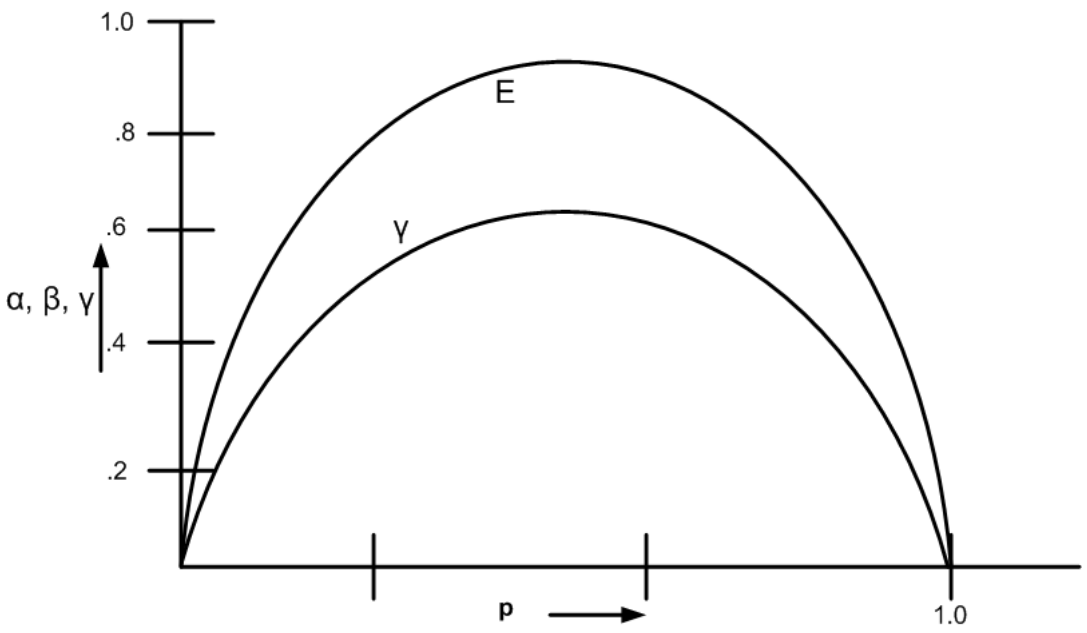
5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Notes on Decision Tree Induction algorithms

7. **Decision tree equivalence:** The different splitting criteria followed in different decision tree induction algorithms have little effect on the performance of the algorithms. This is because the different heuristic measures (such as information gain (α), Gini index (γ) and Gain ratio (β) are quite consistent with each other); also see the figure below.



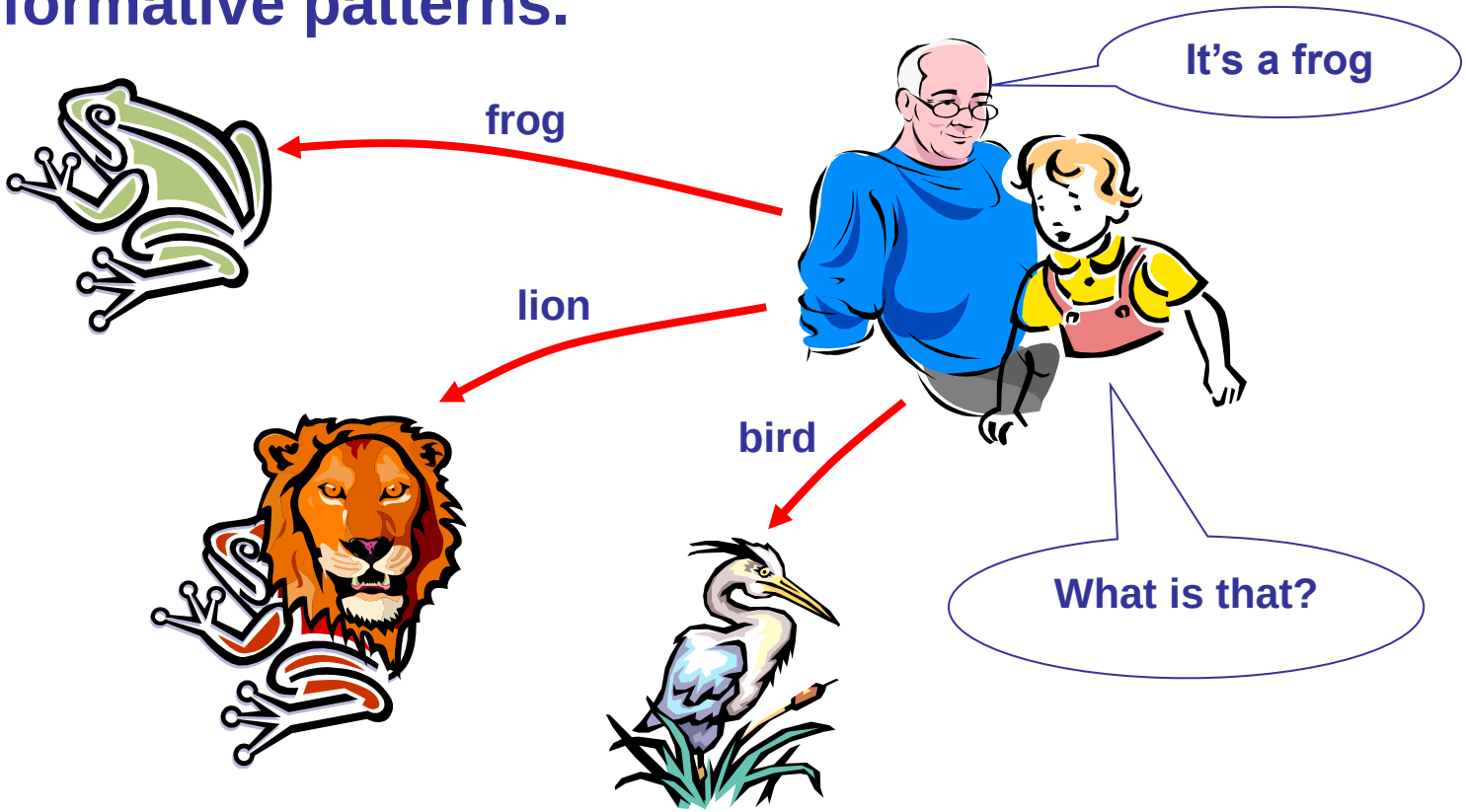
INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS (ANN)

Outline

- Definition, why and how are neural networks being used in solving problems
- Human biological neuron
- Artificial Neuron
- Applications of ANN
- Comparison of ANN vs conventional AI methods

The idea of ANNs..?

- **NNs learn relationship between cause and effect or organize large volumes of data into orderly and informative patterns.**



Neural networks to the rescue…

- **Neural network:** *information processing paradigm inspired by biological nervous systems, such as our brain*
- Structure: large number of highly interconnected processing elements (*neurons*) working together
- Like people, they learn *from experience* (by example)

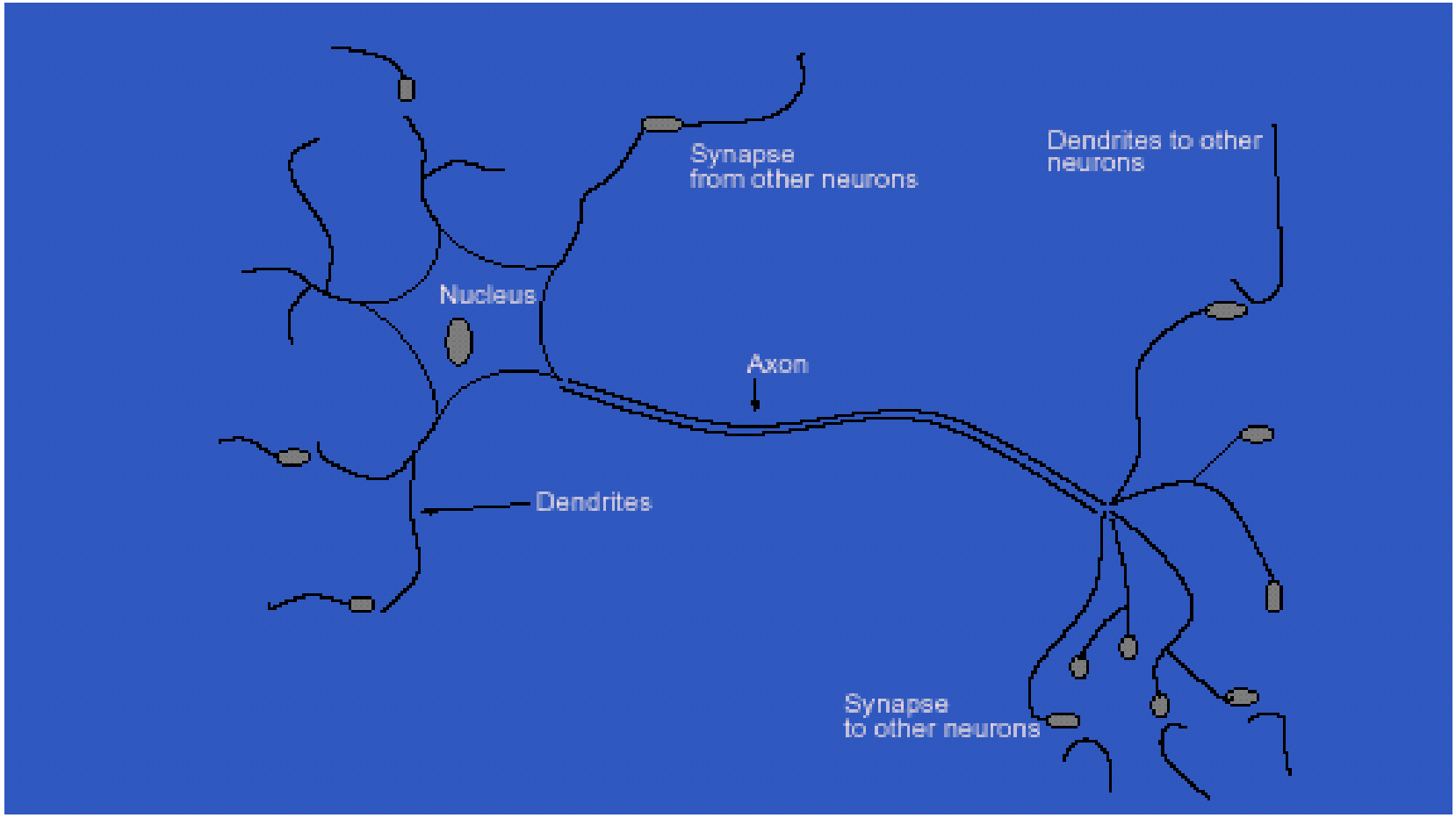
Definition of ANN

"Data processing system consisting of a large number of simple, highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain"

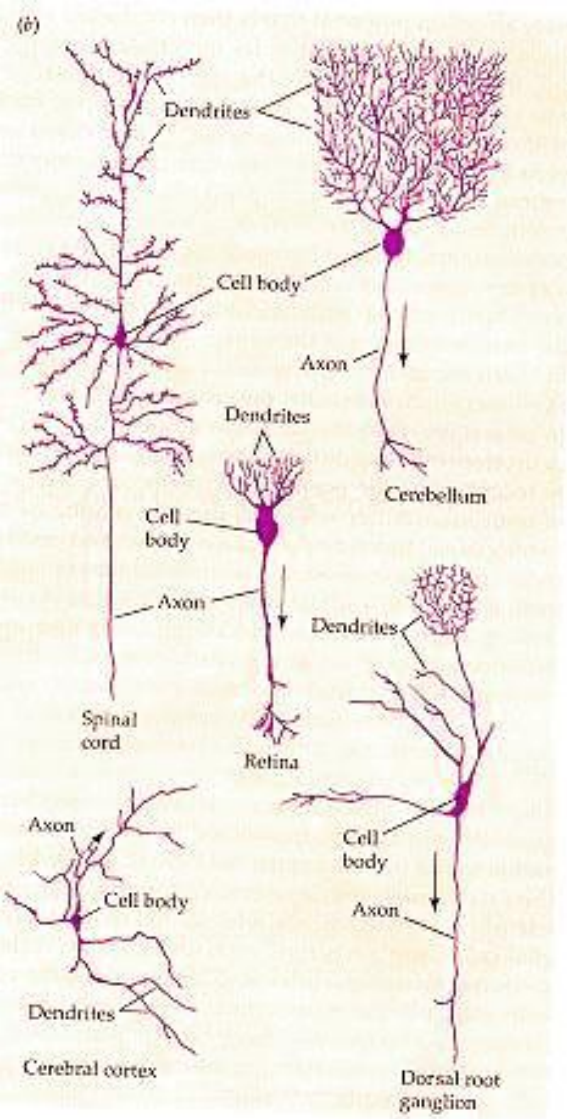
(Tsoukalas & Uhrig, 1997).

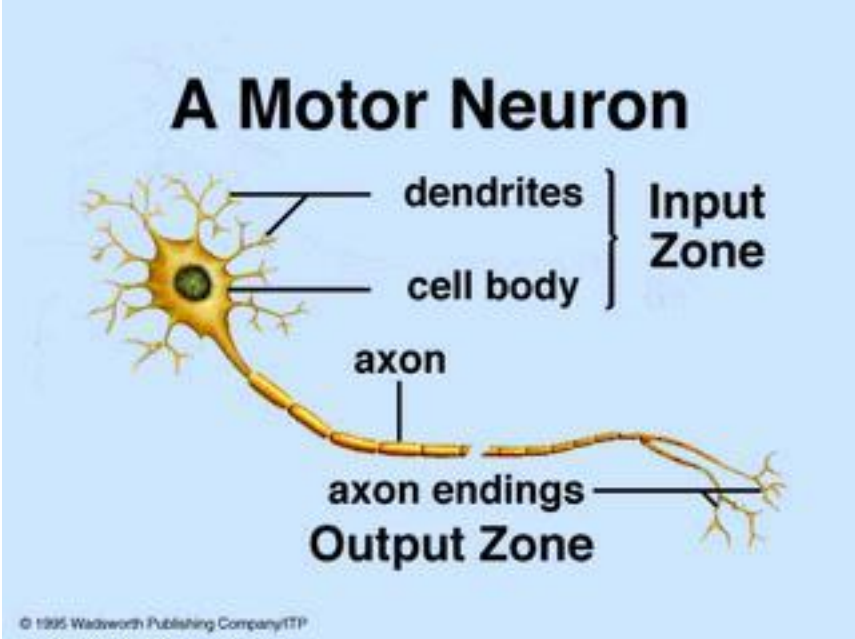
Inspiration from Neurobiology

Human Biological Neuron



Biological Neural Networks

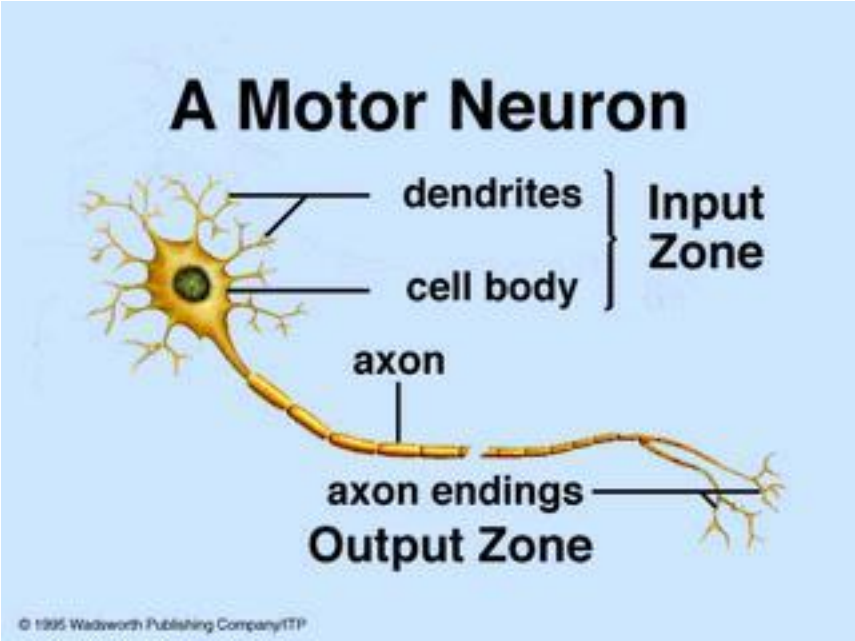




Biological neuron

Biological Neural Networks

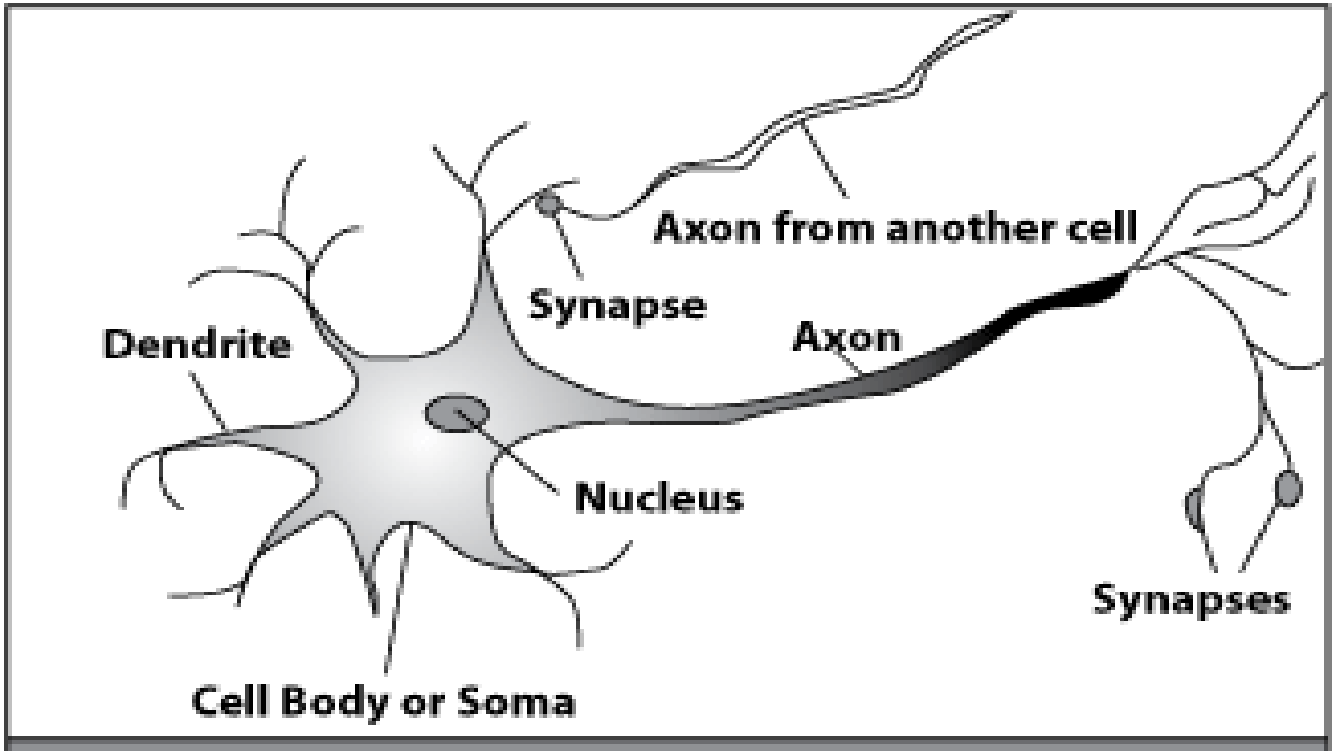
- A biological neuron has three types of main components; dendrites, soma (or cell body) and axon.
- Dendrites receives signals from other neurons.



- The soma, sums the incoming signals. When sufficient input is received, the cell fires; that is it transmit a signal over its axon to other cells.

How do our brains work?

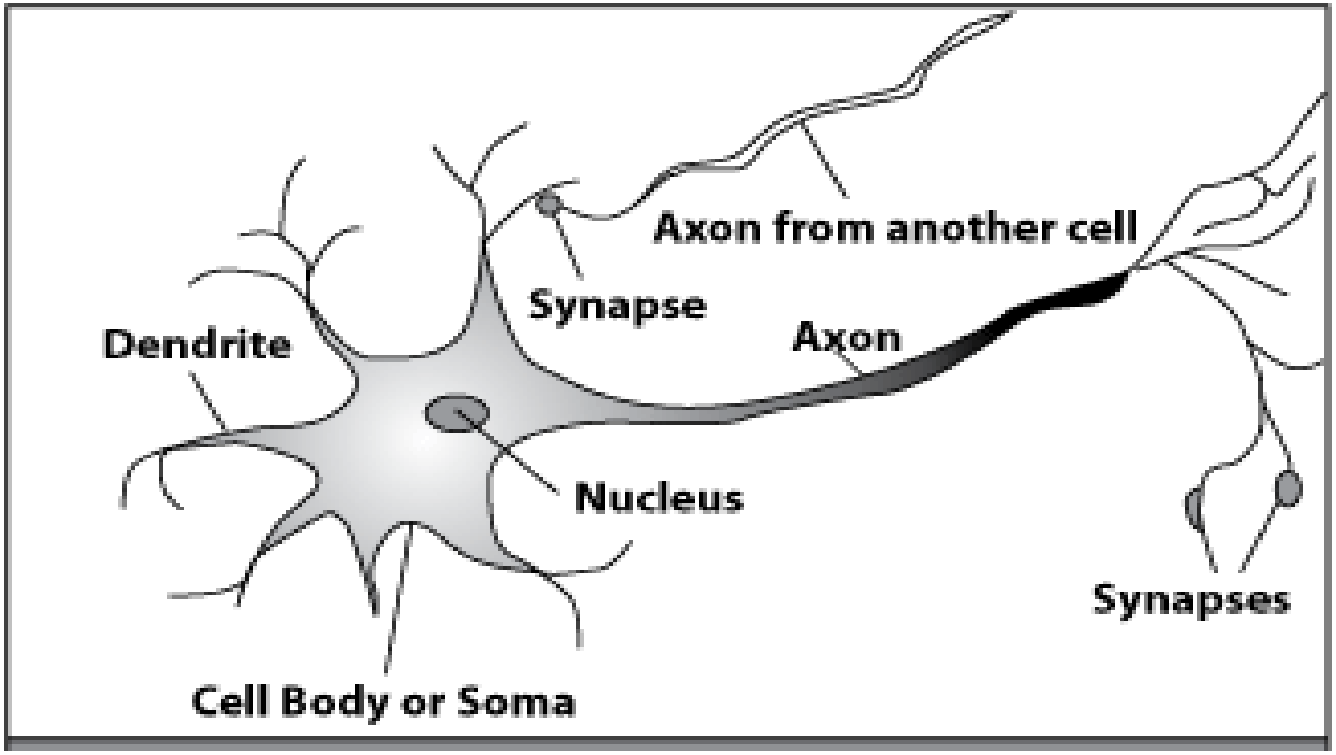
- A processing element



Dendrites: Input
Cell body: Processor
Synaptic: Link
Axon: Output

How do our brains work?

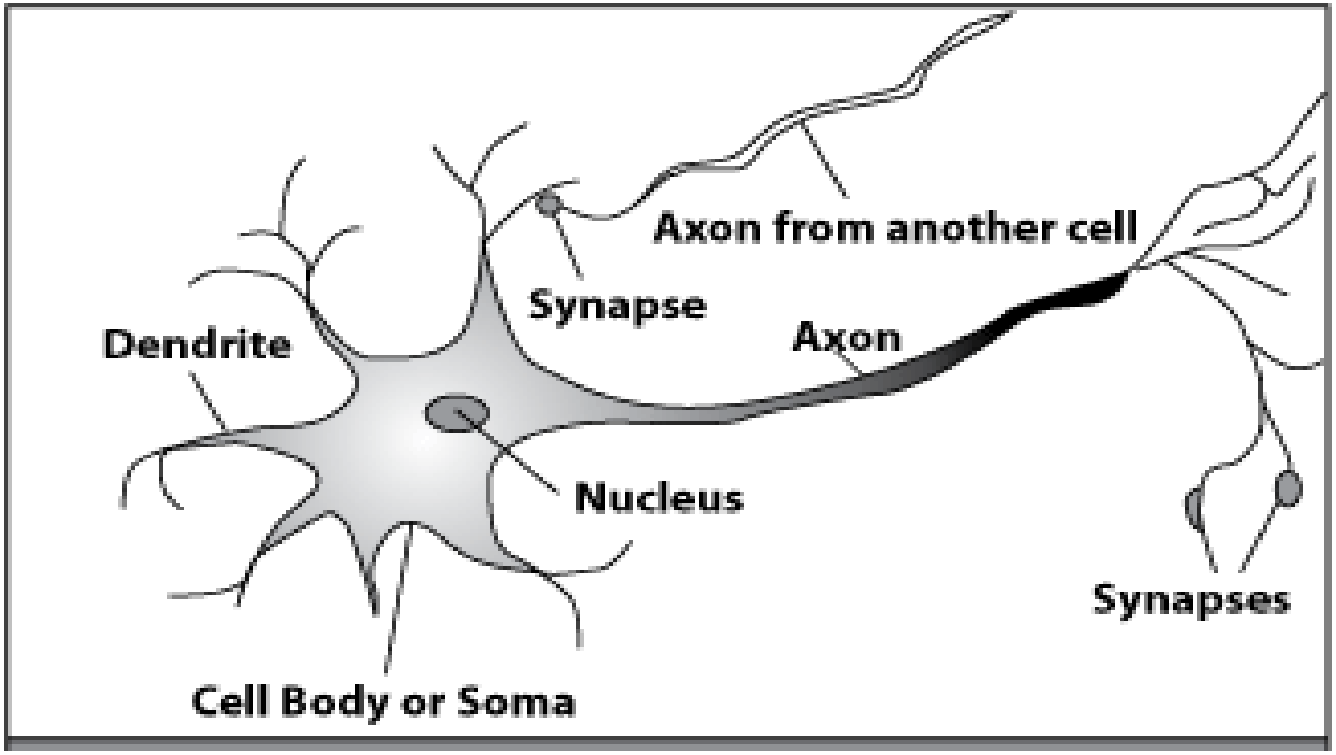
- A processing element



A neuron is connected to other neurons through about *10,000 synapses*

How do our brains work?

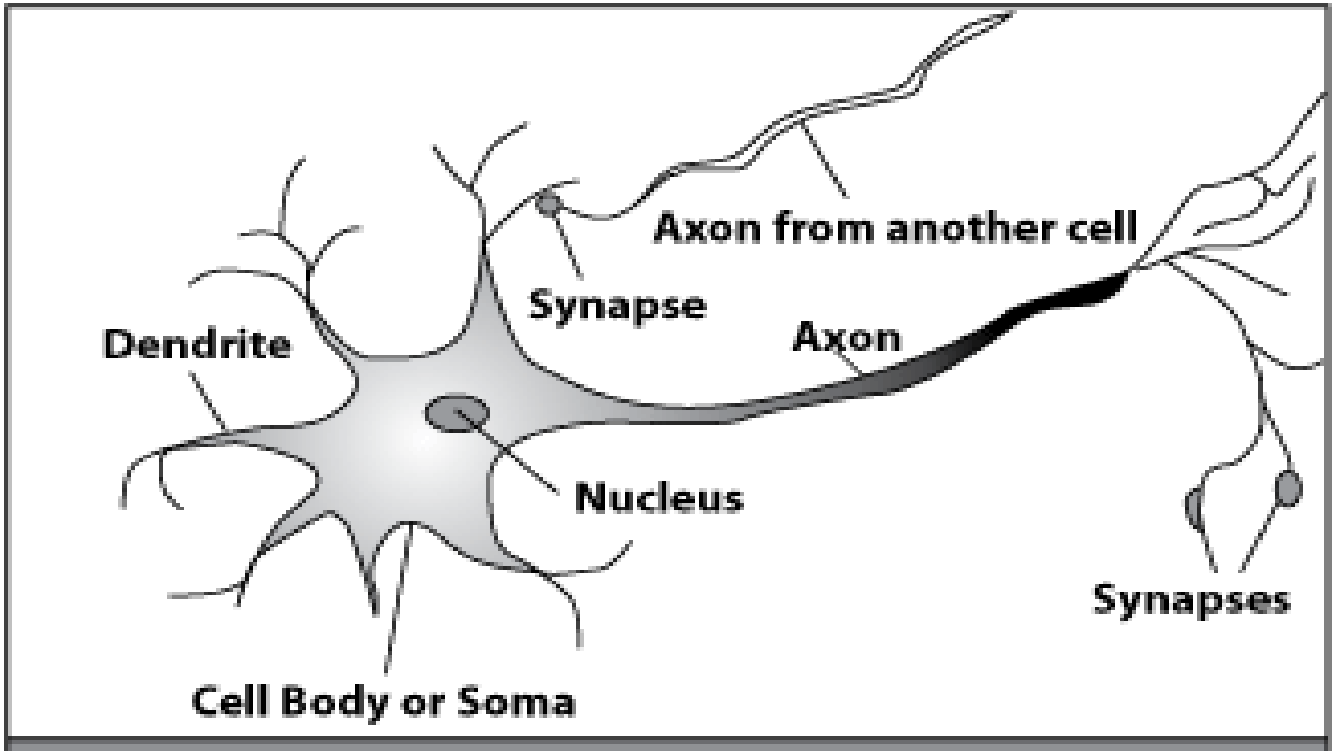
- A processing element



A neuron receives input from other neurons. Inputs are combined.

How do our brains work?

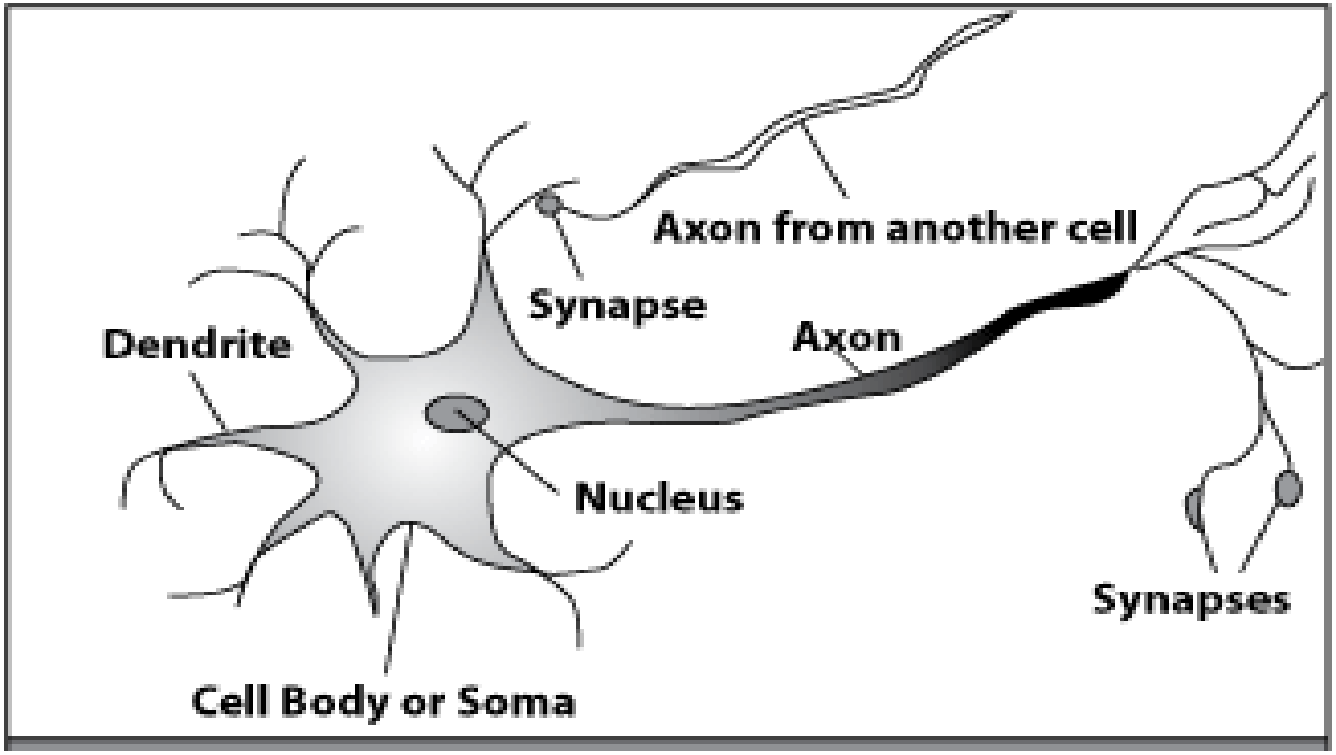
- A processing element



Once input exceeds a critical level, the neuron discharges a spike - an electrical pulse that travels from the body, down the axon, to the next neuron(s)

How do our brains work?

- A processing element



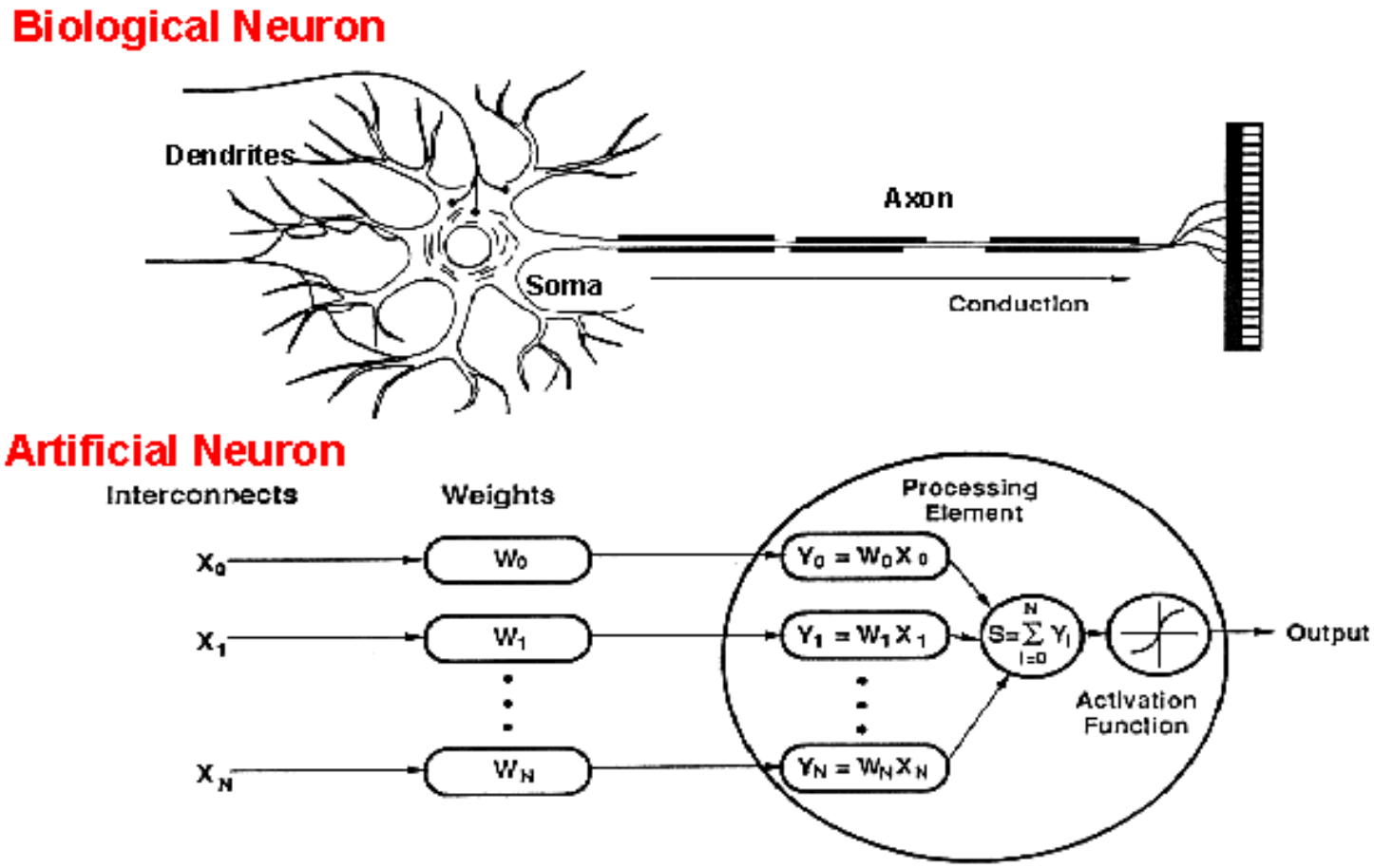
The axon-endings almost touch the dendrites or cell body of the next neuron.

Artificial Neurons

- **ANN is an information processing system that has certain performance characteristics in common with biological nets.**
- **Several key features of the processing elements of ANN are suggested by the properties of biological neurons:**

1. **The processing element receives many signals.**
2. **Signals may be modified by a weight at the receiving synapse.**
3. **The processing element sums the weighted inputs.**
4. **Under appropriate circumstances (sufficient input), the neuron transmits a single output.**
5. **The output from a particular neuron may go to many other neurons.**

How do ANNs work?

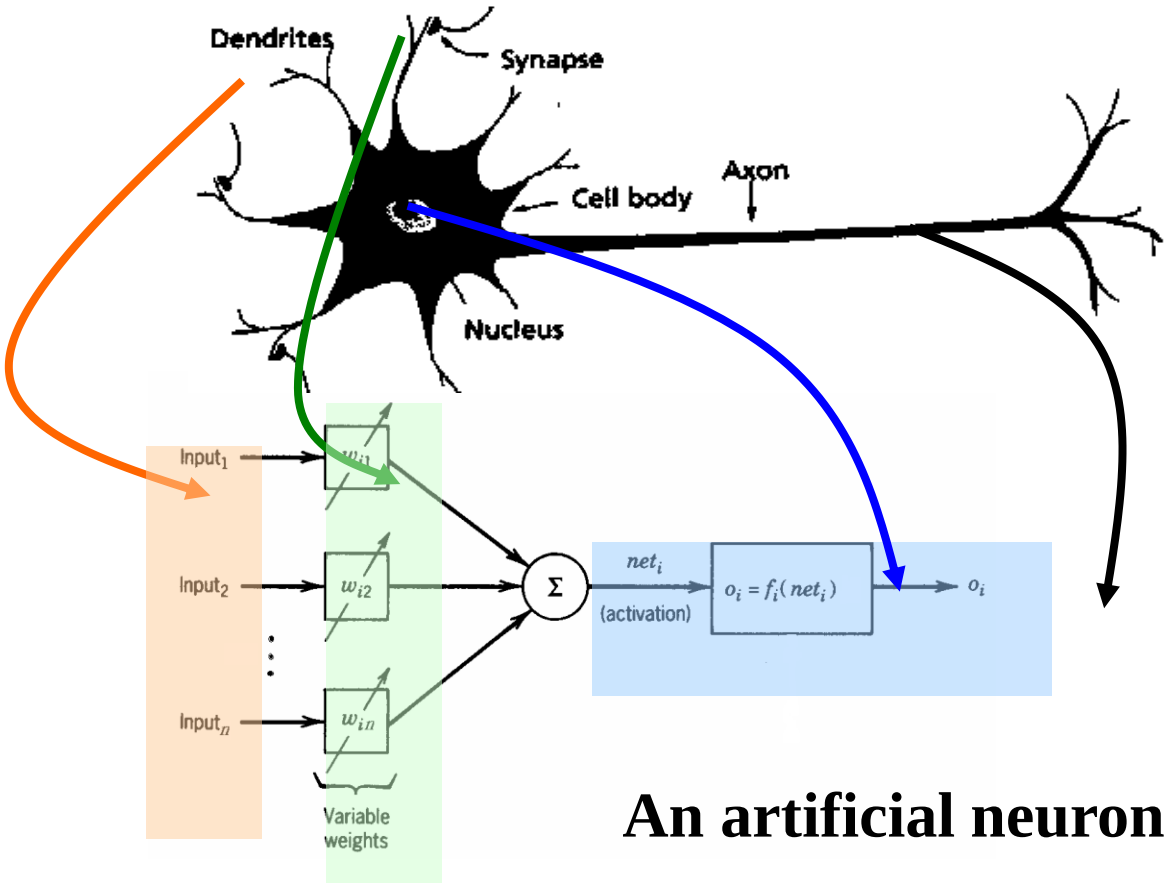


An artificial neuron is an imitation of a human neuron

Artificial Neurons

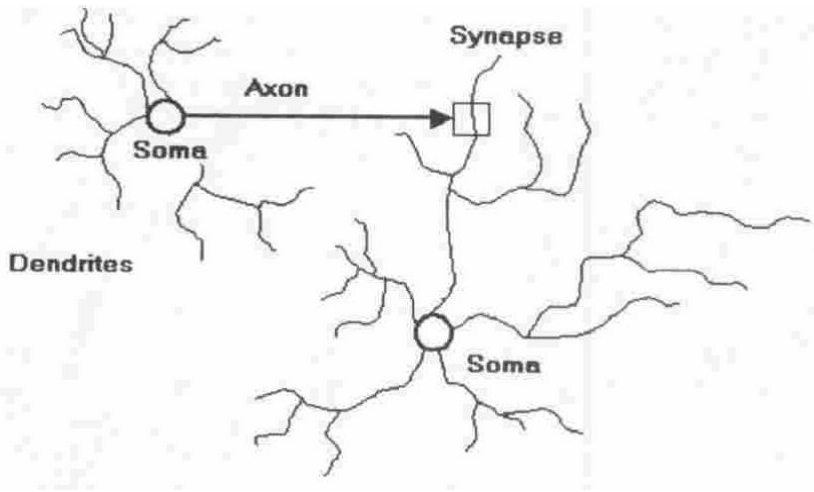
- From experience: examples / training data
- Strength of connection between the neurons is stored as a **weight-value** for the specific connection.
- Learning the solution to a problem = changing the connection weights

A physical neuron

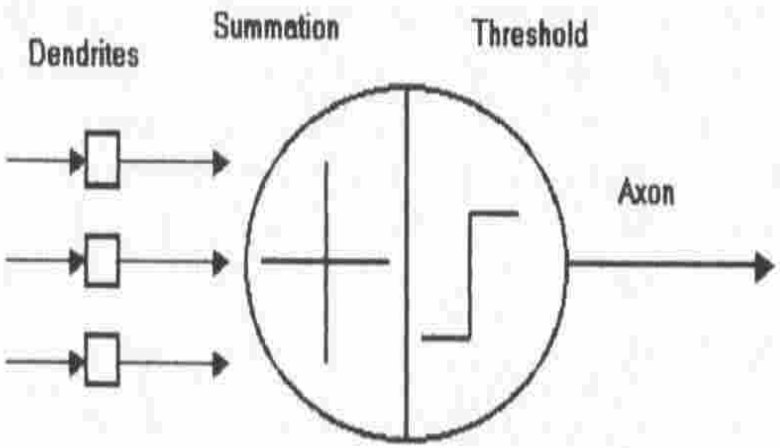


An artificial neuron

Artificial Neuron [Zoom-in view]

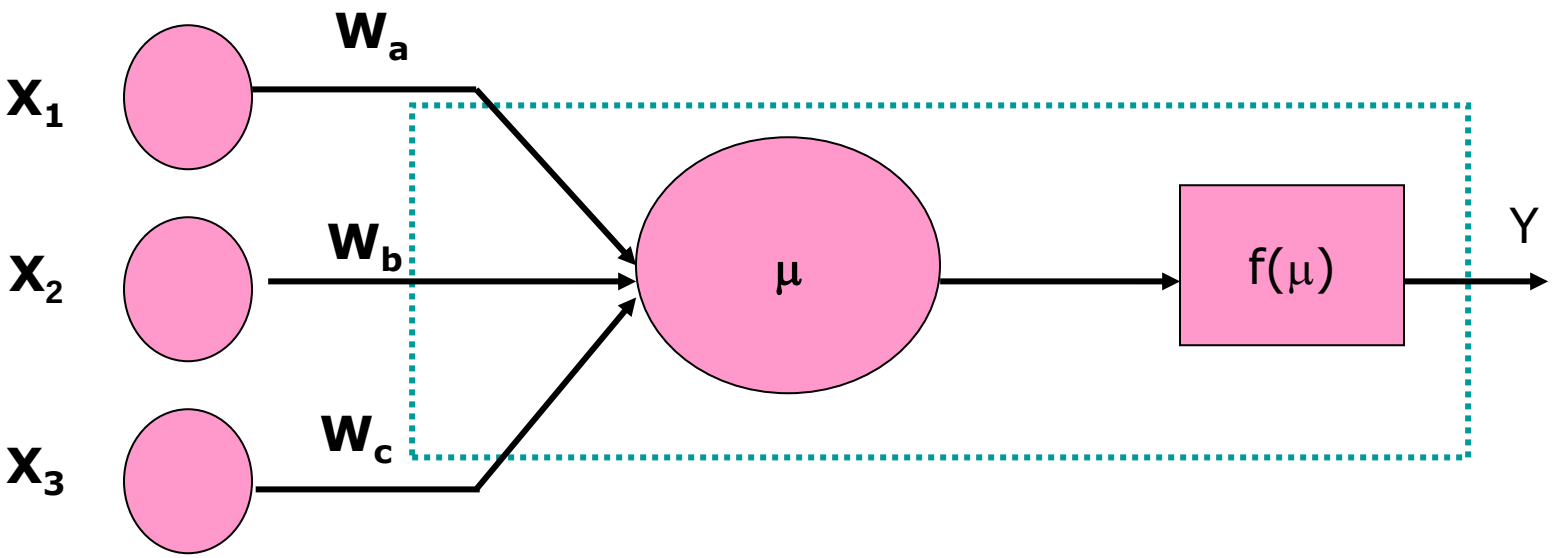


Four basic components of a human biological neuron



The components of a basic artificial neuron

Model Of A Neuron [Zoom-in view]



Input units

Connection weights

Summing function

computation

(dendrite)

(synapse)

(soma)

(axon)

- A neural net consists of a large number of simple processing elements called neurons, units, cells or nodes.

- Each neuron is connected to other neurons by means of directed communication links, each with associated weight.

- The weight represent information being used by the net to solve a problem.

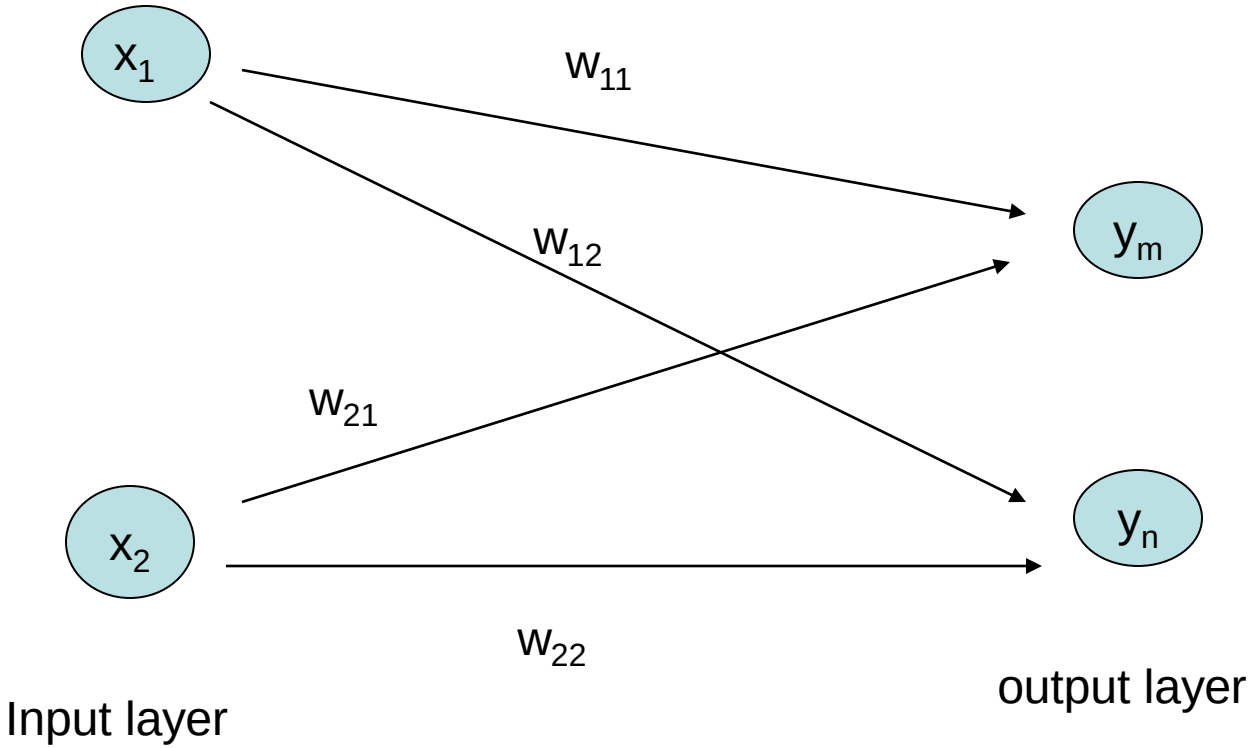
- Each neuron has an internal state, called its activation or activity level, which is a function of the inputs it has received. Typically, a neuron sends its activation as a signal to several other neurons.

- It is important to note that a neuron can send only one signal at a time, although that signal is broadcast to several other neurons.

Characterization

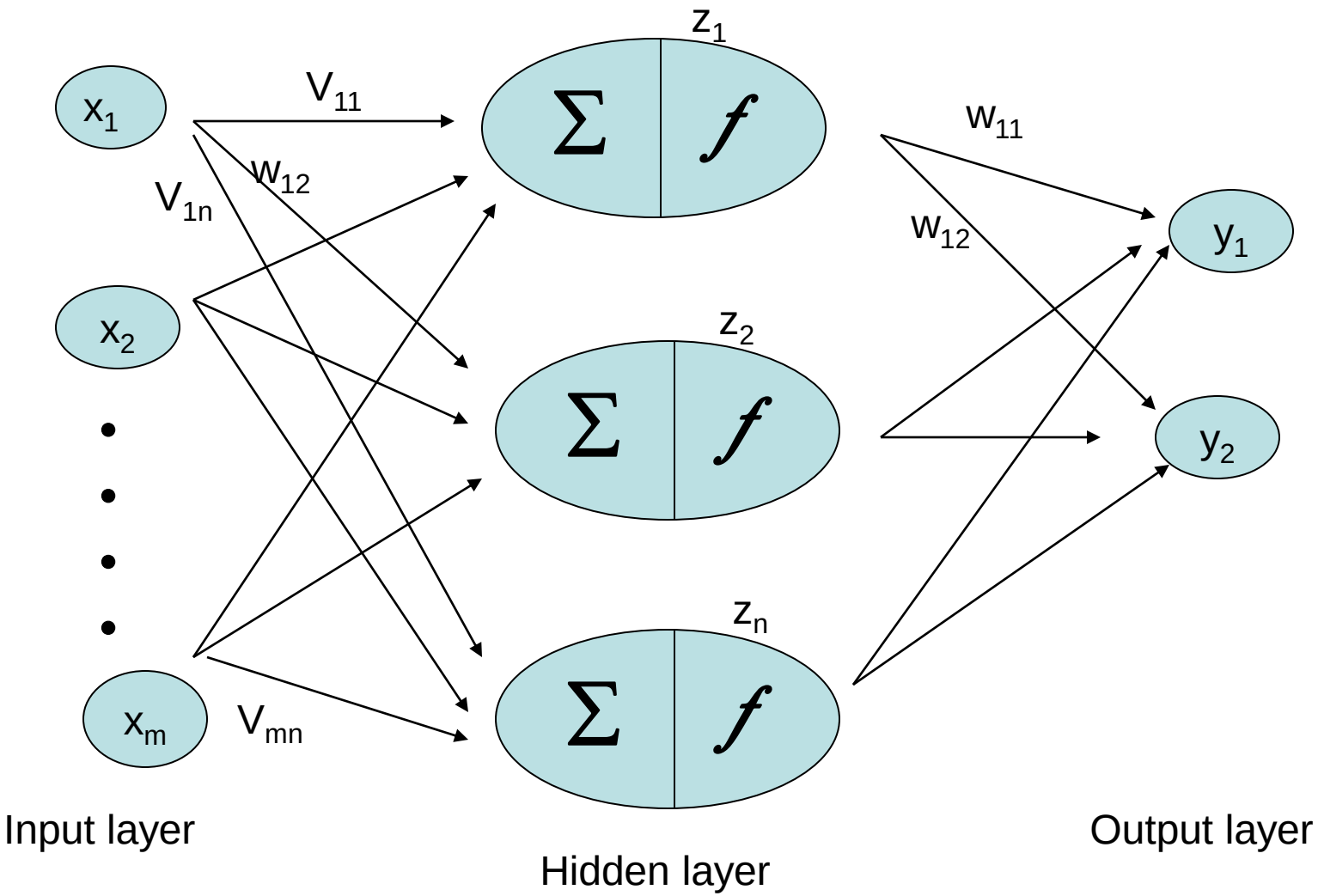
- Architecture
  - a pattern of connections between neurons
    - Single Layer Feedforward
    - Multilayer Feedforward
    - Recurrent
- Strategy / Learning Algorithm
  - a method of determining the connection weights
    - Supervised
    - Unsupervised
    - Reinforcement
- Activation Function
  - Function to compute output signal from input signal

Single Layer Feedforward NN



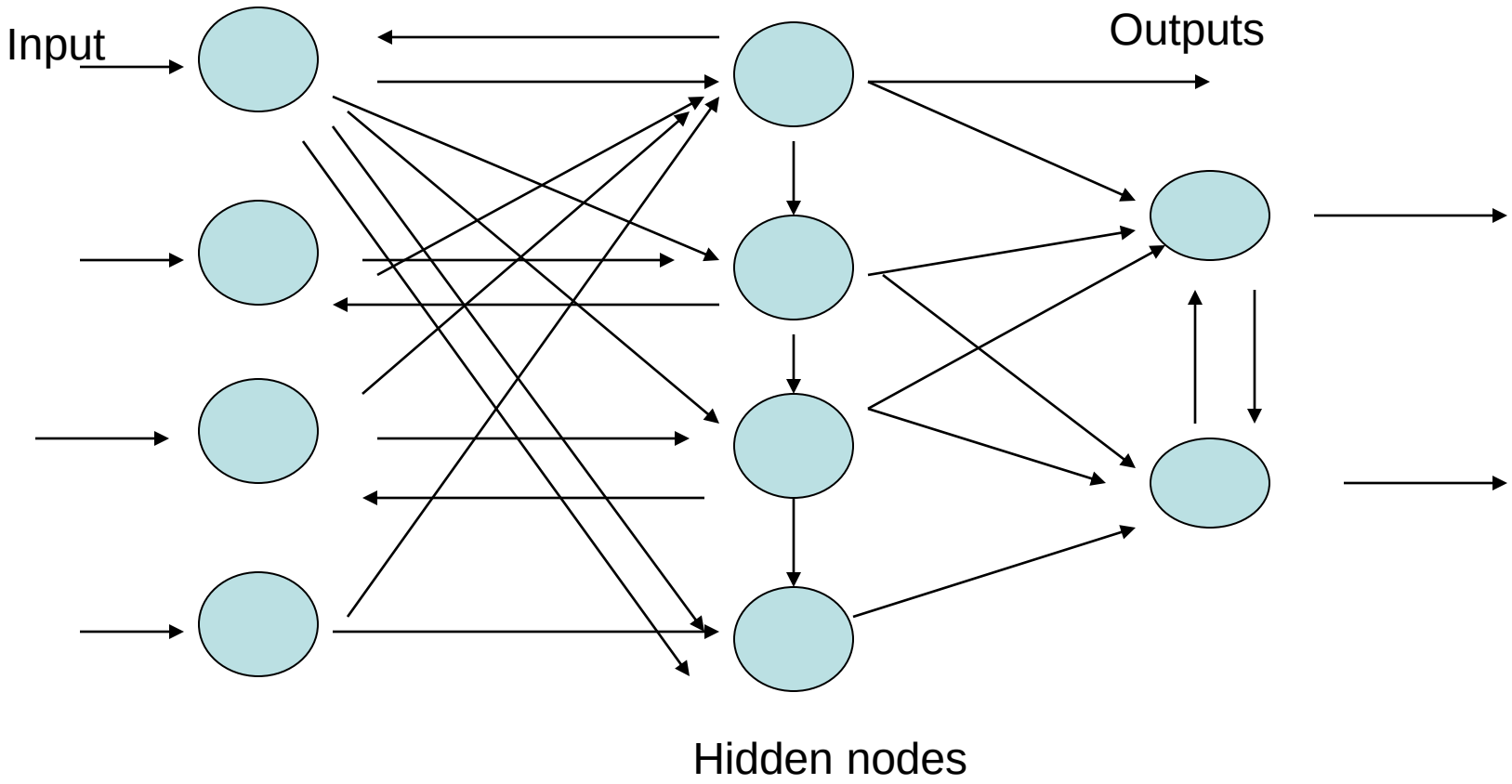
Contoh: **ADALINE, AM, Hopfield, LVQ, Perceptron, SOFM**

Multilayer Neural Network



Contoh: **CCN, GRNN, MADALINE, MLFF with BP, Neocognitron, RBF, RCE**

Recurrent NN



Contoh: **ART, BAM, BSB, Boltzman Machine, Cauchy Machine, Hopfield, RNN**

Activation Function

- An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. It also defines the non-linearity of the NN.

- Sometimes the activation function is called a "*transfer function*." If the output range of the activation function is limited, then it may be called a "*squashing function*." Many activation functions are nonlinear and may be referred to as the "*nonlinearity*" in the layer or the network design.

Activation Functions

- Identity

  $f(x) = x$

- Binary step

  $f(x) = 1$ if $x >= \theta$

  $f(x) = 0$ otherwise

- Binary sigmoid

  $f(x) = 1 / (1 + e^{-\sigma x})$

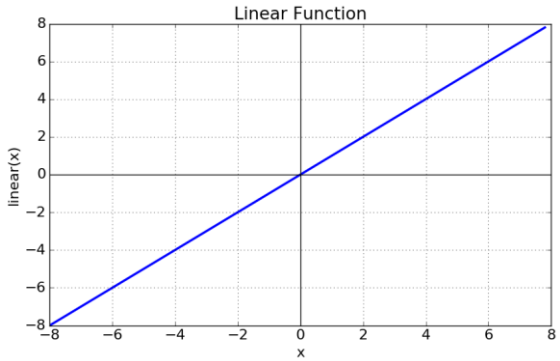
- Bipolar sigmoid

  $f(x) = -1 + 2 / (1 + e^{-\sigma x})$

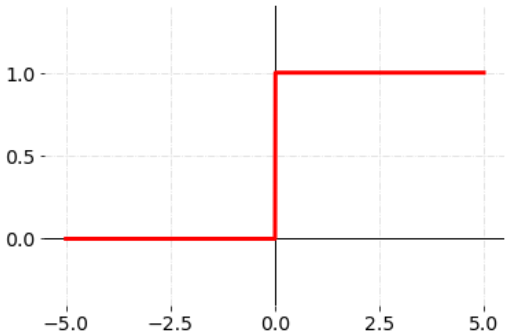
- Hyperbolic tangent

  $f(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$

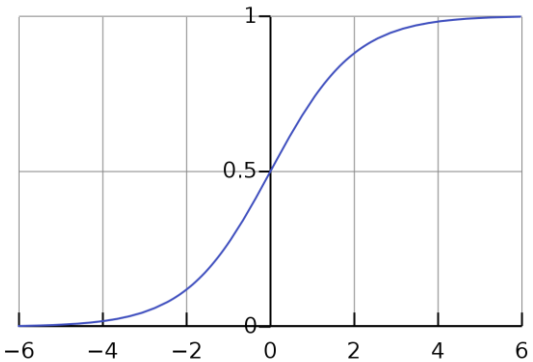
Activation Functions



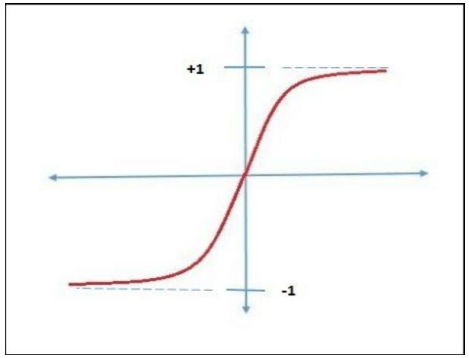
Identity: f(x) = x



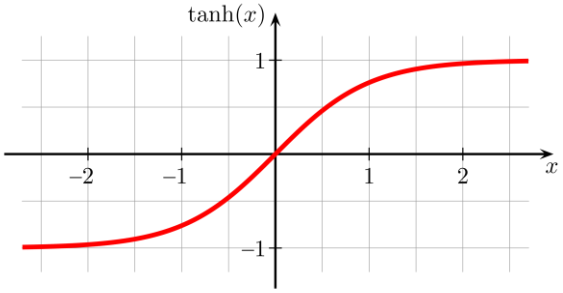
Binary Step: f(x) = 1 if x >= θ
f(x) = 0 otherwise



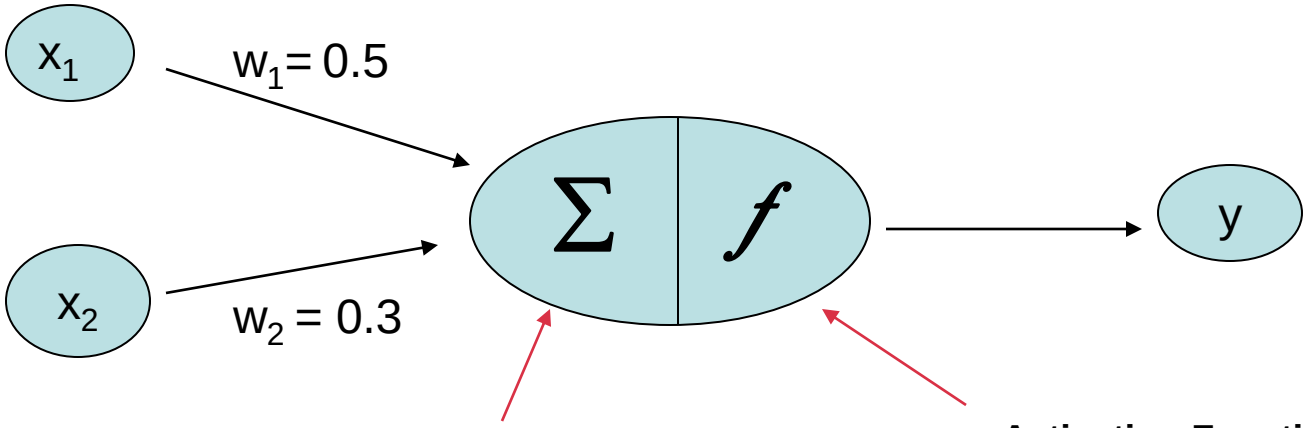
Binary Sigmoid:
$f(x) = 1 / (1 + e^{-\sigma x})$



Bipolar Sigmoid:
$f(x) = -1 + 2 / (1 + e^{-\sigma x})$



Hyperbolic tangent:
$f(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$

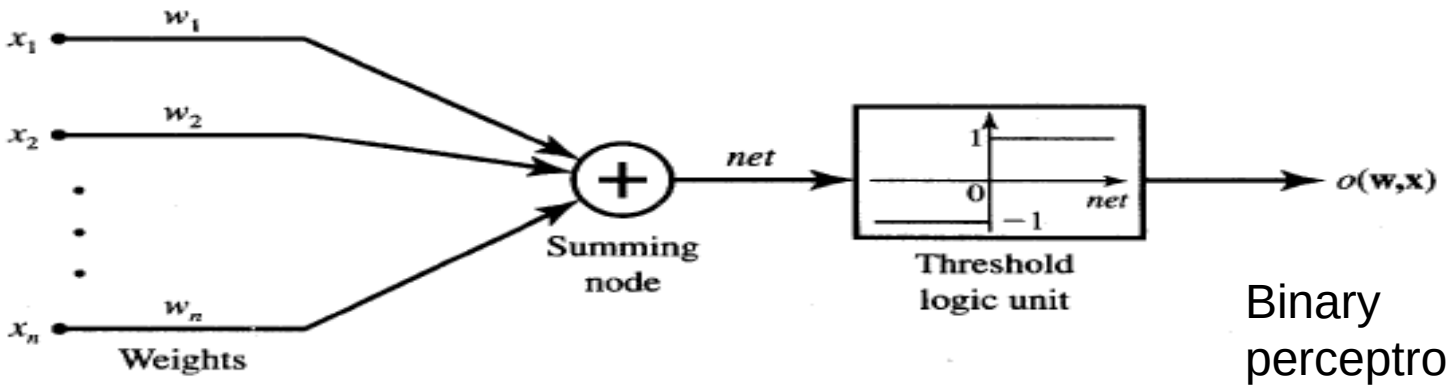


$y_{in} = x_1w_1 + x_2w_2$

<u>Activation Function:</u>
Binary Step Function
$\theta = 0.5$

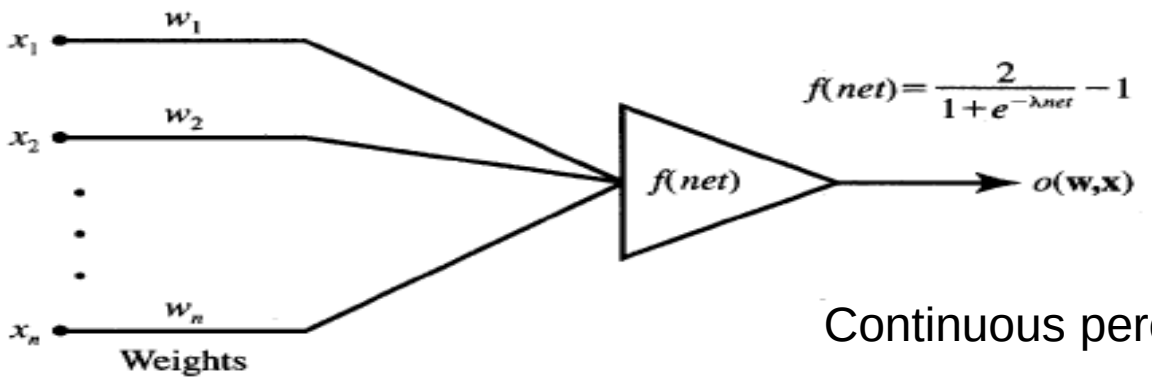
if y-in >= θ
f(y-in) = 1
else,
f(y-in) = 0

Common models of neurons



Binary perceptrons

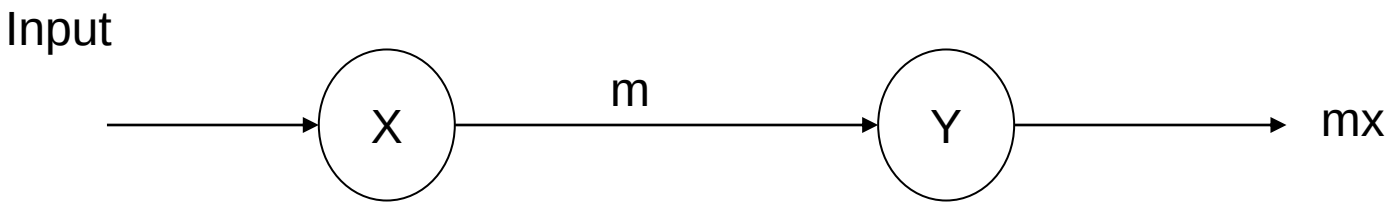
(a)



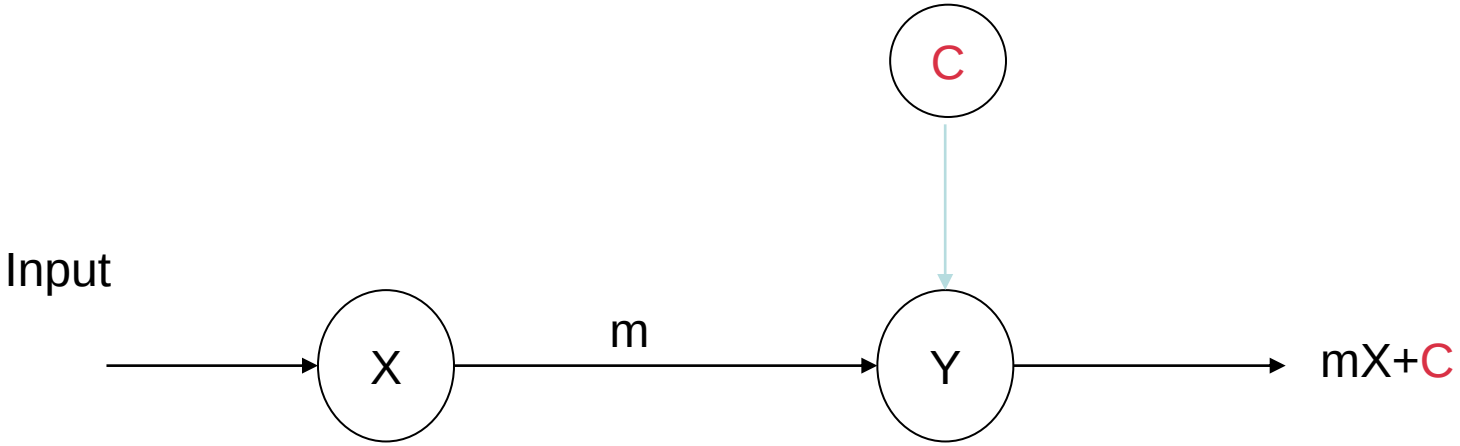
Continuous perceptrons

(b)

Neural net of pure linear eqn.



But, we know: Y=mX+C

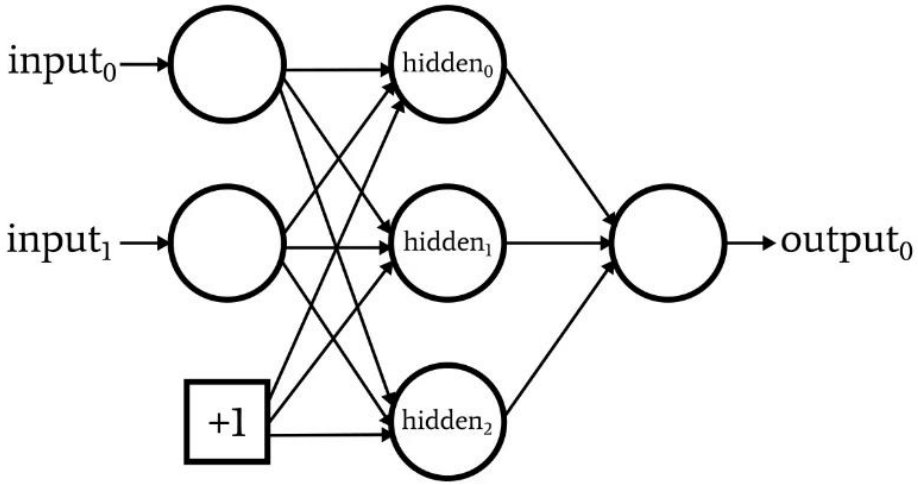


Bias in Neural Network

- The activation function in Neural Networks takes an input 'x' multiplied by a weight 'w'.

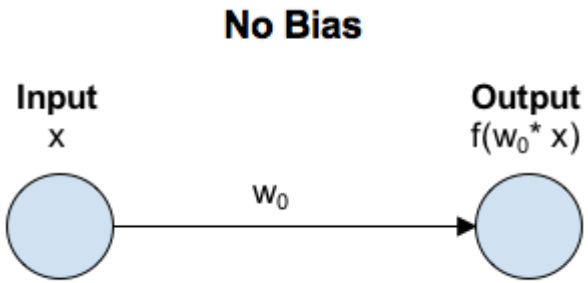
- Bias allows you to shift the activation function by adding a constant (i.e. the given bias) to the input.

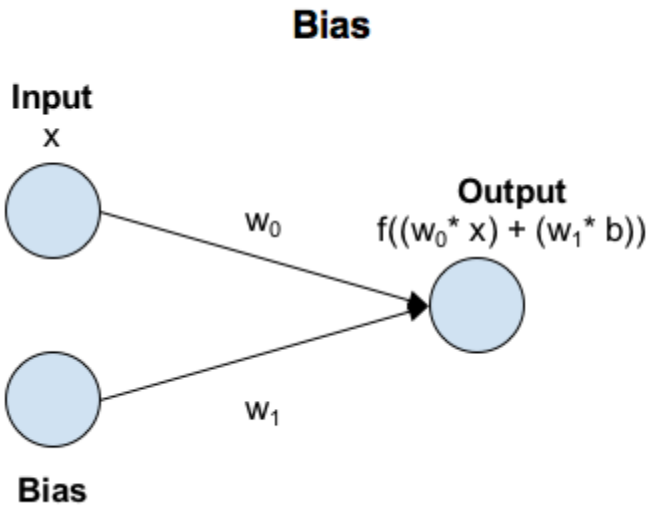
- Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value.



Bias

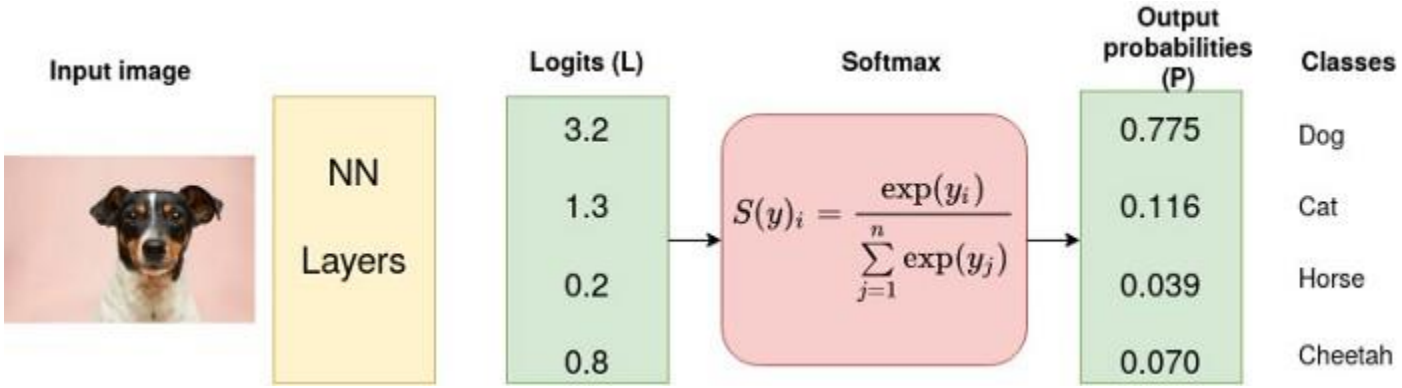
Bias In Neural Network





Logits and Softmax Layer

- Consider a CNN model which aims at classifying an image as either a dog, cat, horse or cheetah (4 possible outcomes/classes).
- The last (fully-connected) layer of the CNN outputs a vector of logits, L, that is passed through a Softmax layer that transforms the logits into probabilities, P.
- These probabilities are the model-predictions for each of the 4 classes.



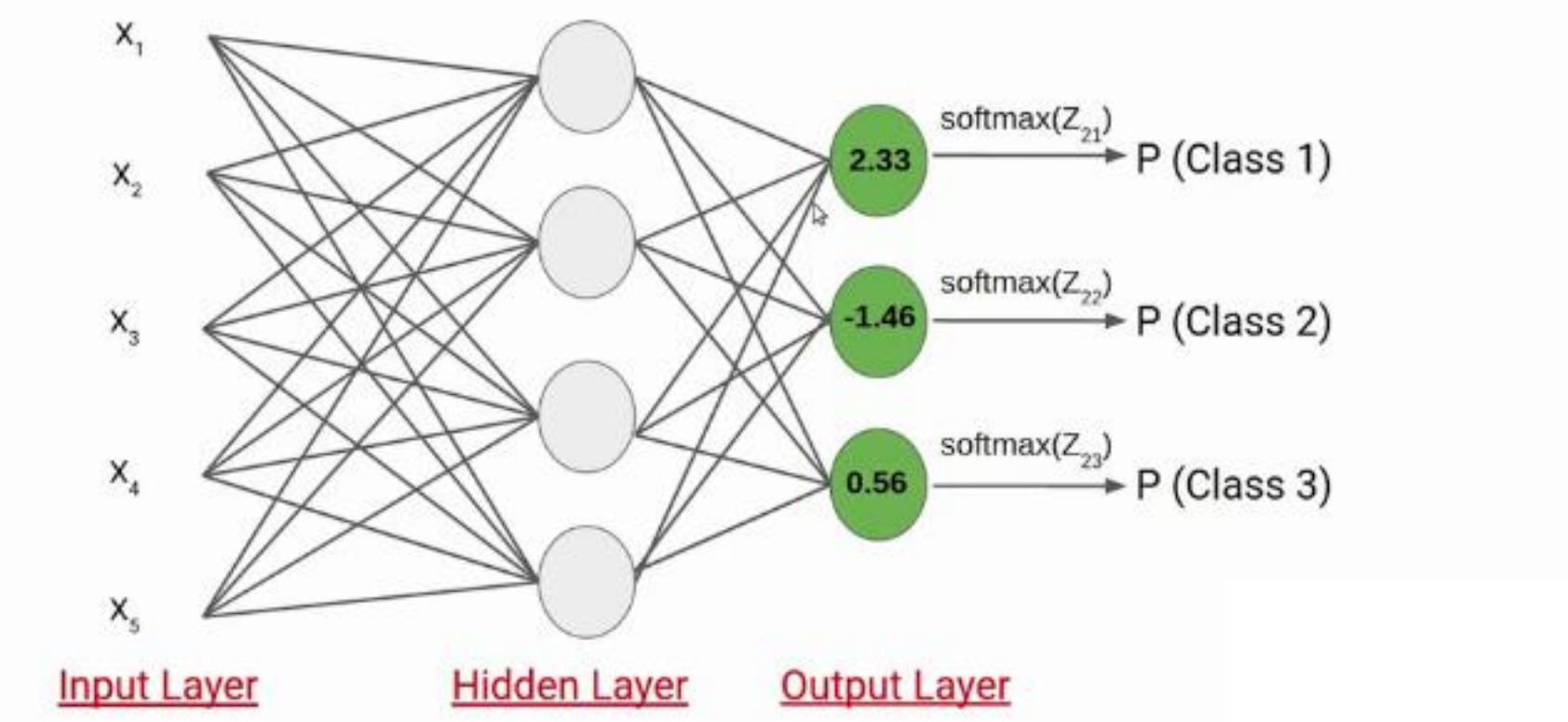
Softmax Activation

Softmax is an activation function that scales numbers/logits into probabilities.
The output of a Softmax is a vector (say *V)* with probabilities of each possible outcome. The probabilities in vector *V* sums to one for all possible outcomes or classes.

Mathematically, Softmax is defined as,

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

Softmax Layer



(Logits)

Softmax Layer

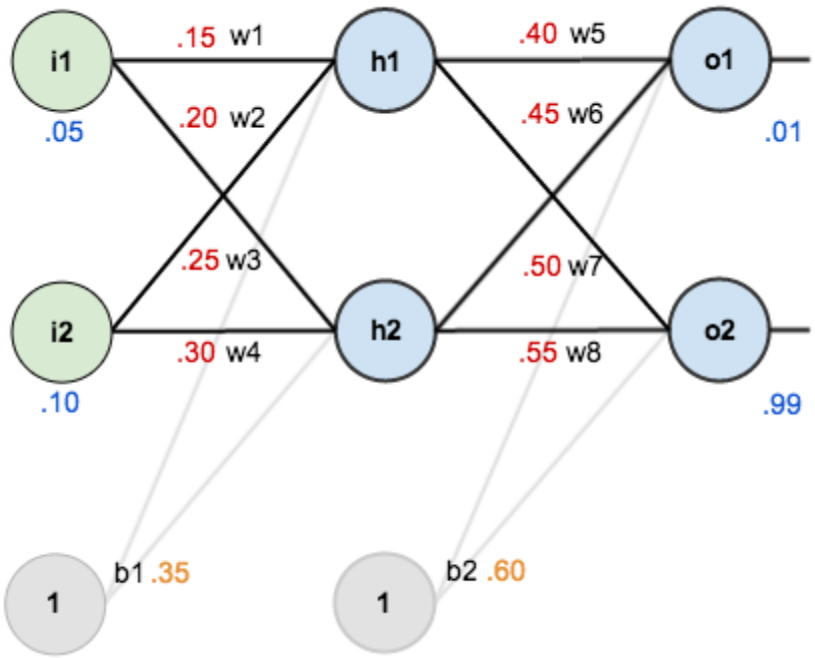
Example :

$$2.33 \rightarrow P(\text{Class } 1) = \frac{\exp(2.33)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.83827314$$

$$-1.46 \rightarrow P(\text{Class } 2) = \frac{\exp(-1.46)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.01894129$$

$$0.56 \rightarrow P(\text{Class } 3) = \frac{\exp(0.56)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.14278557$$

Test



Find the values of h1, h2, o1, o2
use tanh activation function.

Strategy / Learning Algorithm

Supervised Learning

- Learning is performed by presenting pattern with target
- During learning, produced output is compared with the desired output
  - The difference between both output is used to modify learning weights according to the learning algorithm
- Recognizing hand-written digits, pattern recognition and etc.
- Neural Network models: perceptron, feed-forward, radial basis function, support vector machine.

Unsupervised Learning

- Targets are not provided
- Appropriate for clustering task
  - Find similar groups of documents in the web, content addressable memory, clustering.
- Neural Network models: Kohonen, self organizing maps, Hopfield networks.

Reinforcement Learning

- Target is provided, but the desired output is absent.
- The net is only provided with guidance (appreciate/criticize) to determine the produced output is correct or vise versa.
- Weights are modified in the units that have errors

Self Study (ANN Misc.)

Where can neural network systems help…

- when we can't formulate an algorithmic solution.
- when we ***can*** get lots of examples of the behavior we require.

  ‘learning from experience’

- when we need to pick out the structure from existing data.

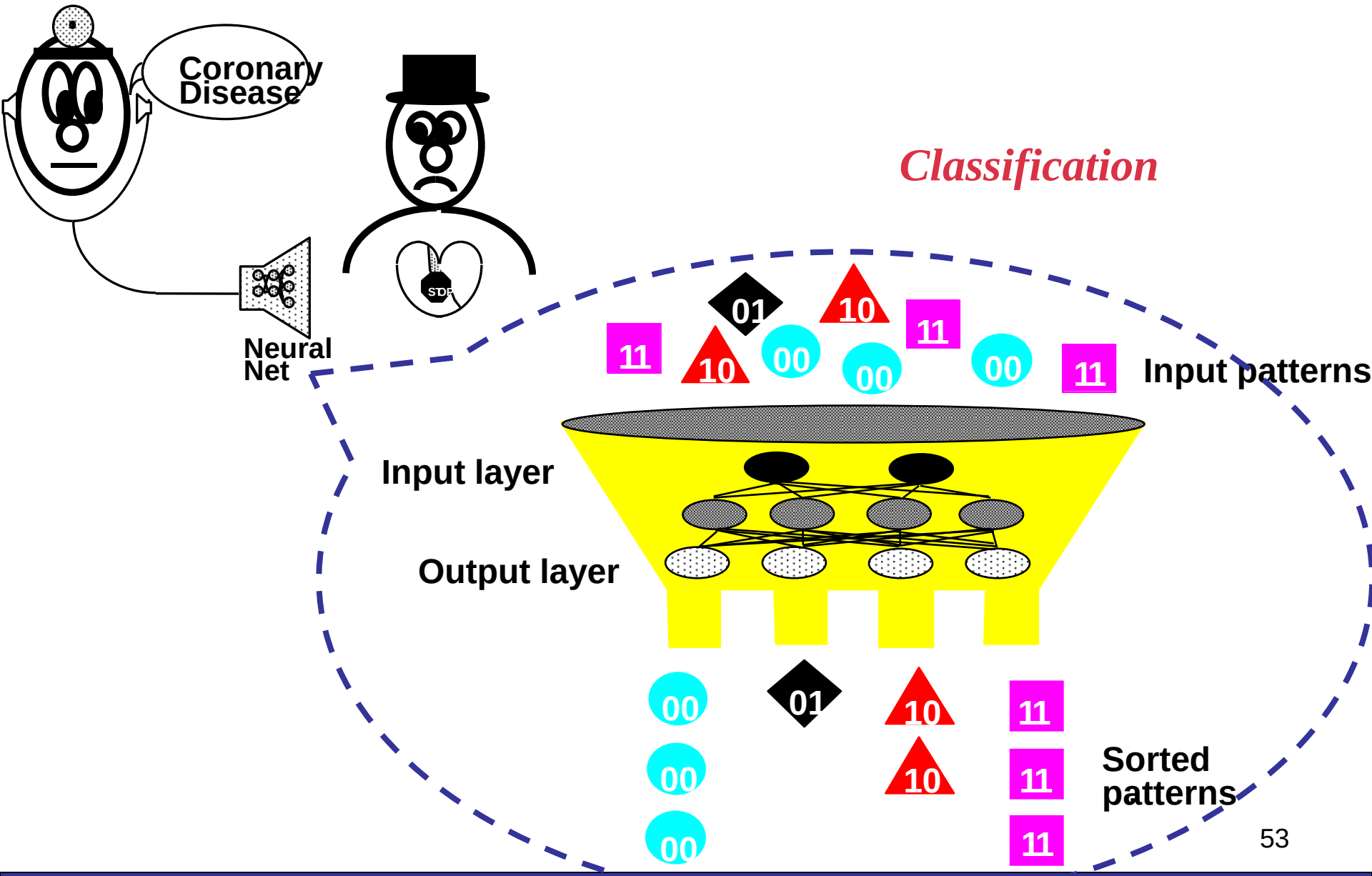
Who is interested?...

- Electrical Engineers – signal processing, control theory
- Computer Engineers – robotics
- Computer Scientists – artificial intelligence, pattern recognition
- Mathematicians – modelling tool when explicit relationships are unknown

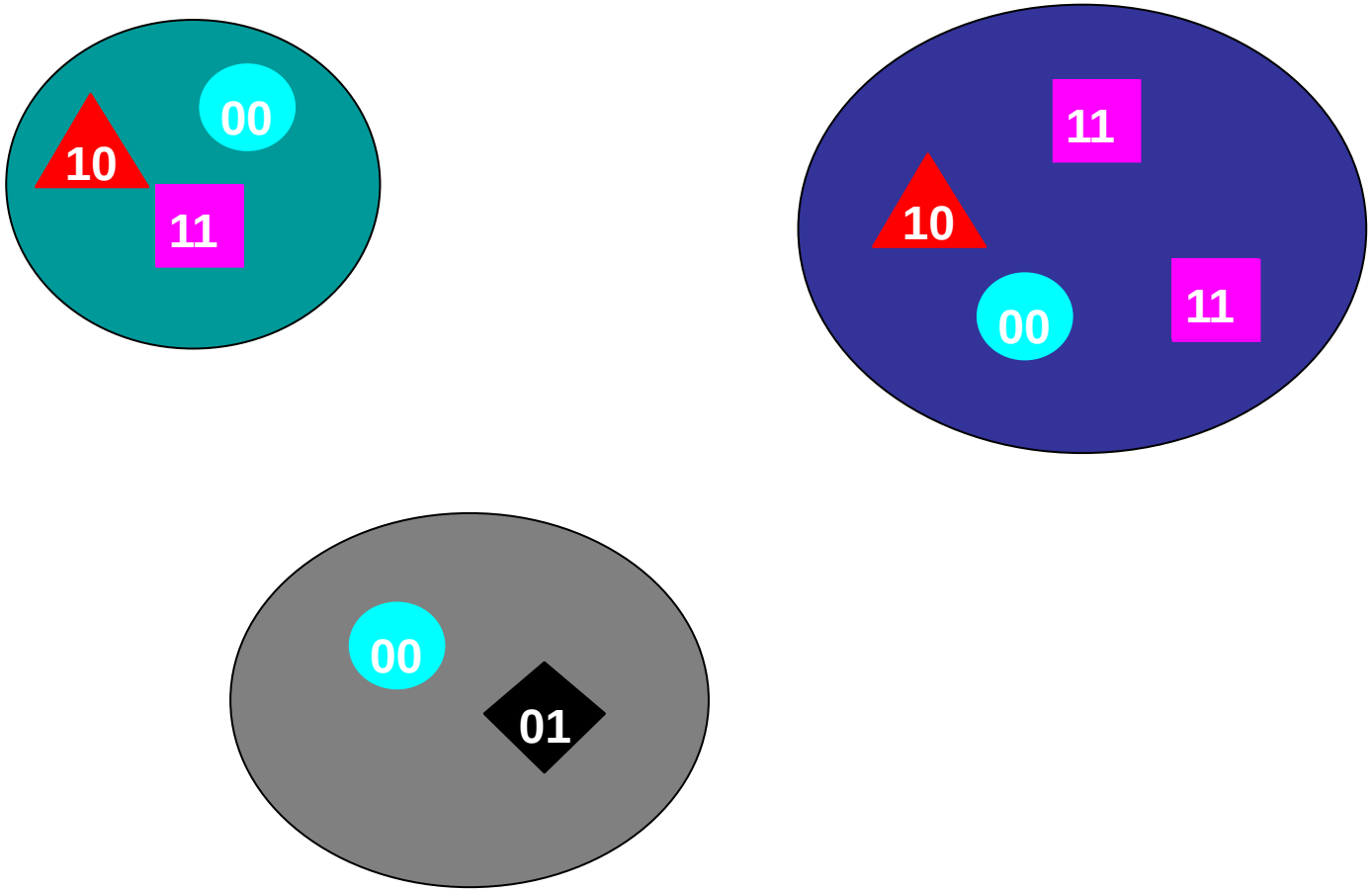
Problem Domains

- Storing and recalling patterns
- Classifying patterns
- Mapping inputs onto outputs
- Grouping similar patterns
- Finding solutions to constrained optimization problems

Classification

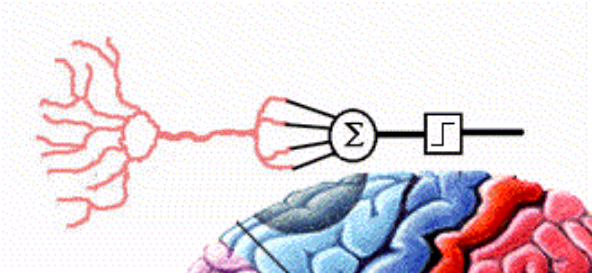


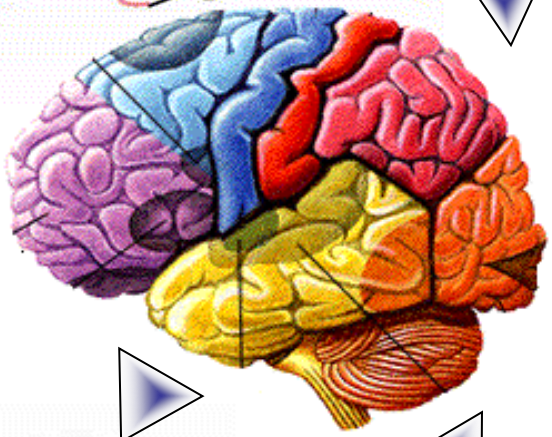
Clustering



ANN Applications

Medical Applications





Chemistry

Information
Searching & retrieval

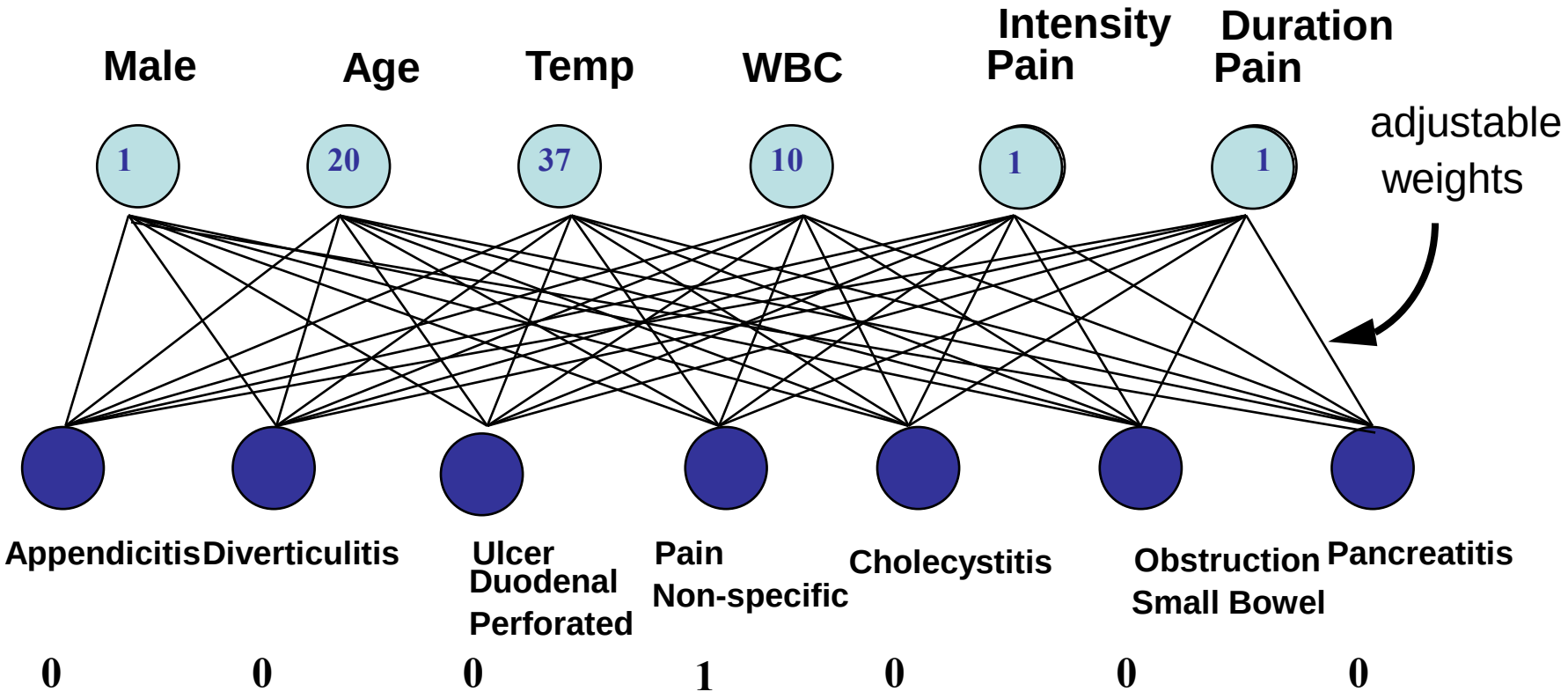
Education

Business & Management

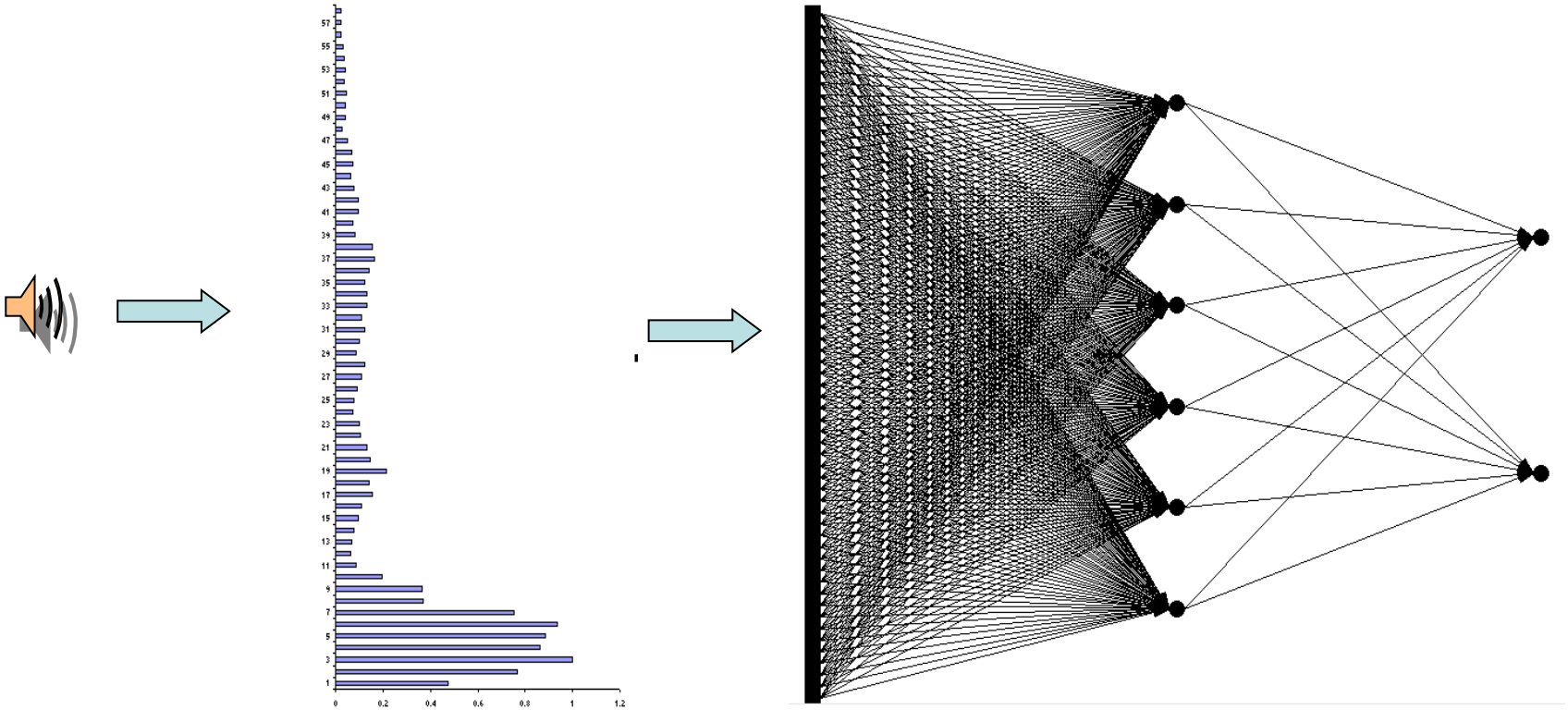
Applications of ANNs

- Signal processing
- Pattern recognition, e.g. handwritten characters or face identification.
- Diagnosis or mapping symptoms to a medical case.
- Speech recognition
- Human Emotion Detection
- Educational Loan Forecasting

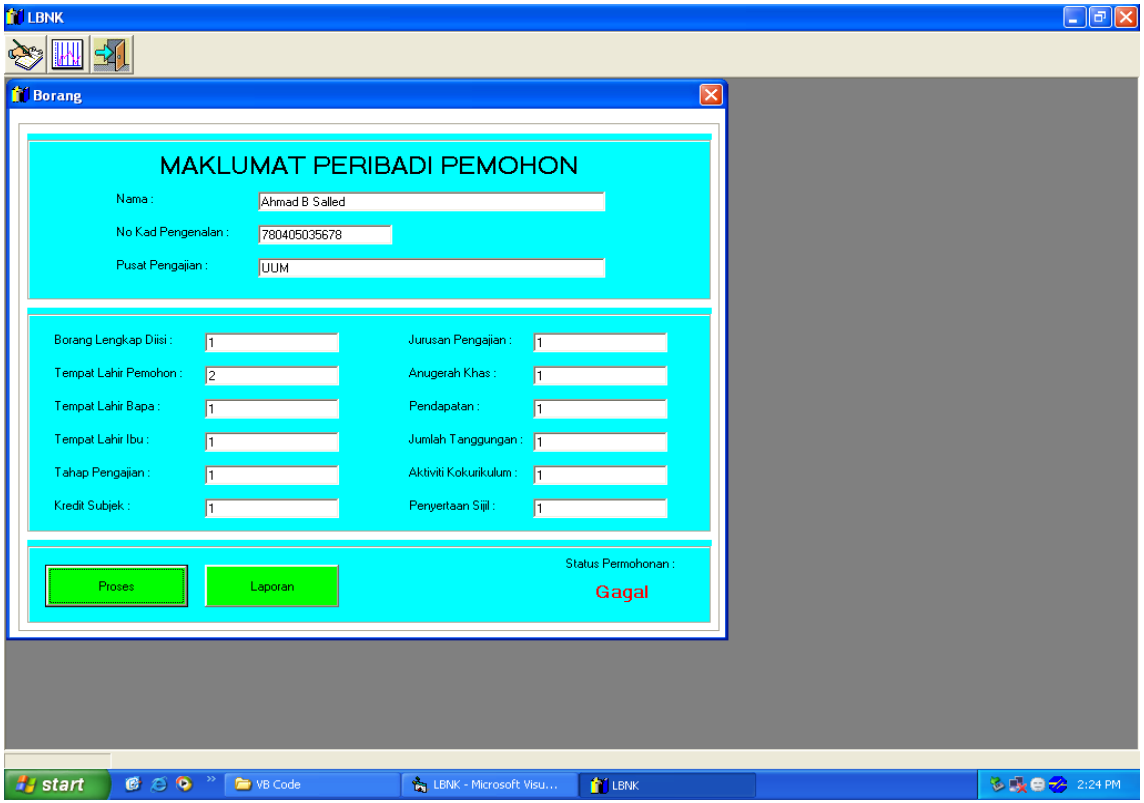
Abdominal Pain Prediction



Voice Recognition



Educational Loan Forecasting System



Advantages Of NN

NON-LINEARITY

It can model non-linear systems

INPUT-OUTPUT MAPPING

It can derive a relationship between a set of input & output responses

ADAPTIVITY

The ability to learn allows the network to adapt to changes in the surrounding environment

EVIDENTIAL RESPONSE

It can provide a confidence level to a given solution

Advantages Of NN

CONTEXTUAL INFORMATION

Knowledge is presented by the structure of the network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally in the network.

FAULT TOLERANCE

Distributed nature of the NN gives it fault tolerant capabilities

NEUROBIOLOGY ANALOGY

Models the architecture of the brain

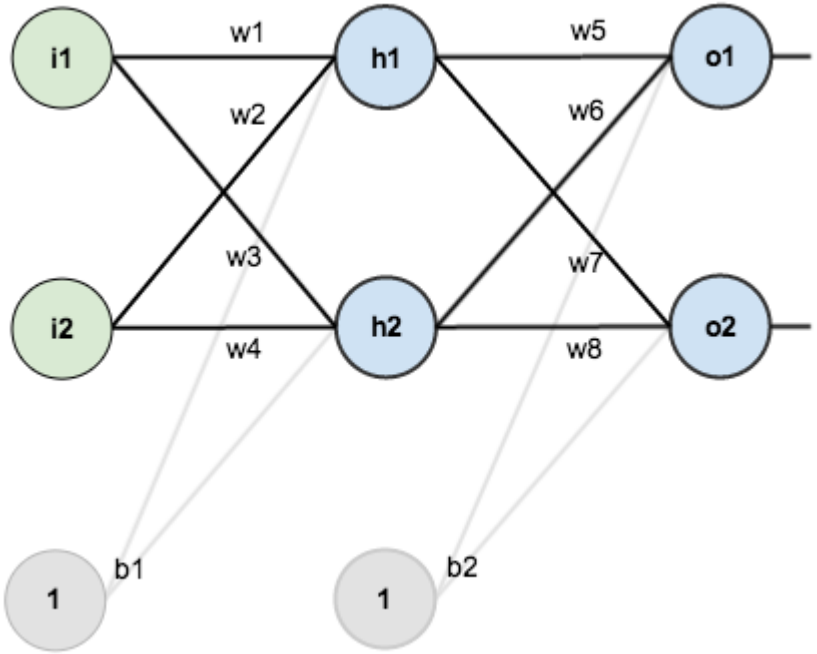
Comparison of ANN with conventional AI methods

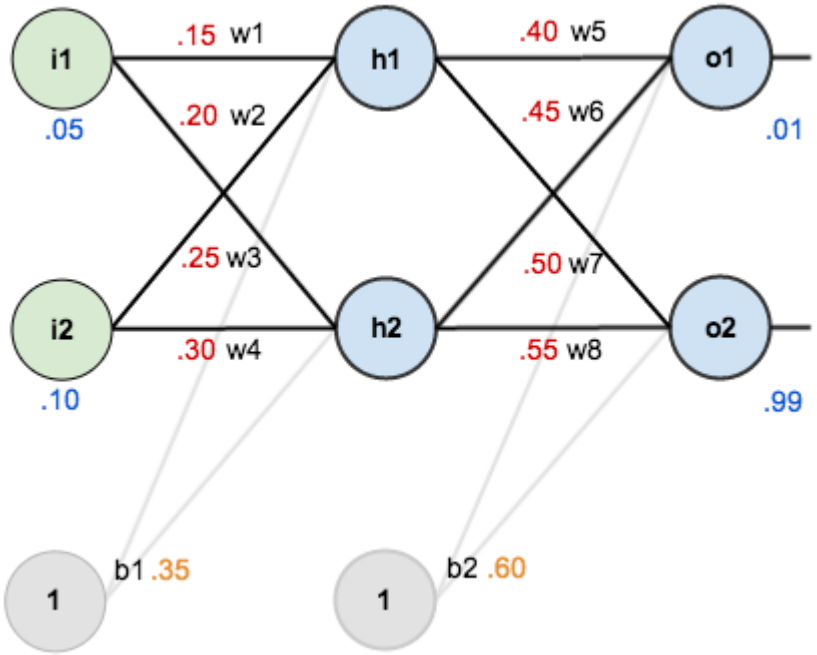
CHARACTERISTICS	TRADITIONAL COMPUTING (including Expert Systems)	ARTIFICIAL NEURAL NETWORKS
Processing style	Sequential	Parallel
Functions	Logically (left brained) via Rules Concepts Calculations	Gestault (right brained) via Images Pictures Controls
Learning Method	by rules (didactically)	by example (Socratically)
Applications	Accounting, word processing, math, inventory, digital communications	Sensor processing, speech recognition, pattern recognition, text recognition

Backpropagation

Suppose a Network is given

Ground-truth/true-values:
O_1=0.01
O_2 = 0.99.



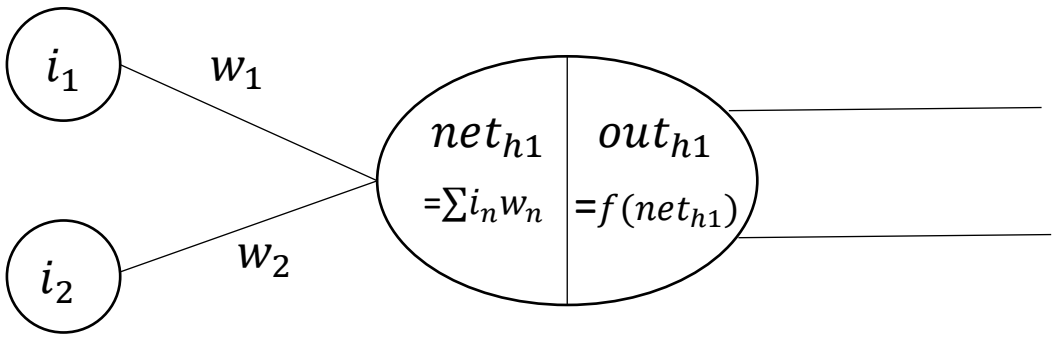


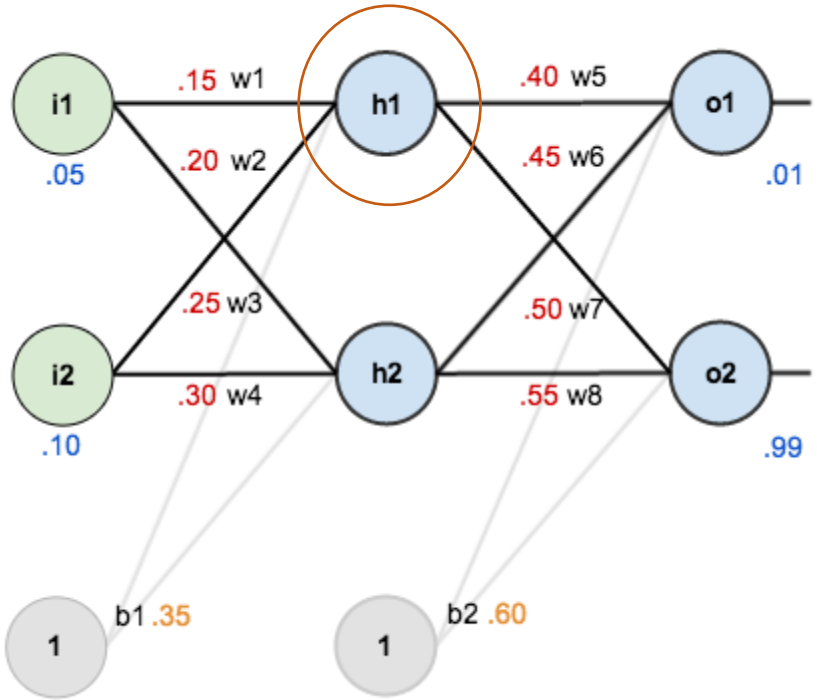
Random weight initialization

Goal

The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

The Forward Pass





Here's how we calculate the total net input for h_1:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

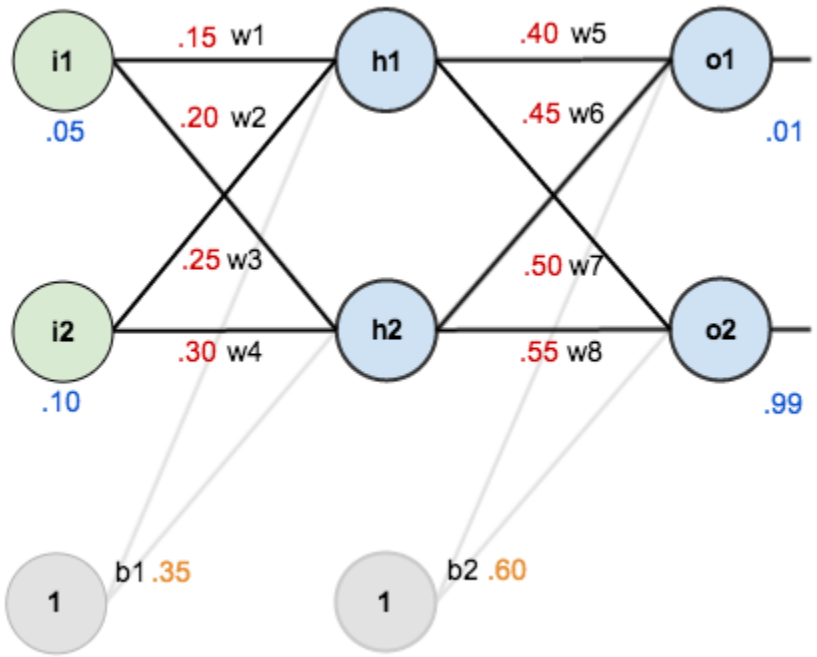
We then squash it using the logistic function to get the output of h_1:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$

The Forward Pass (contd.)



We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Here's the output for o_1:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$

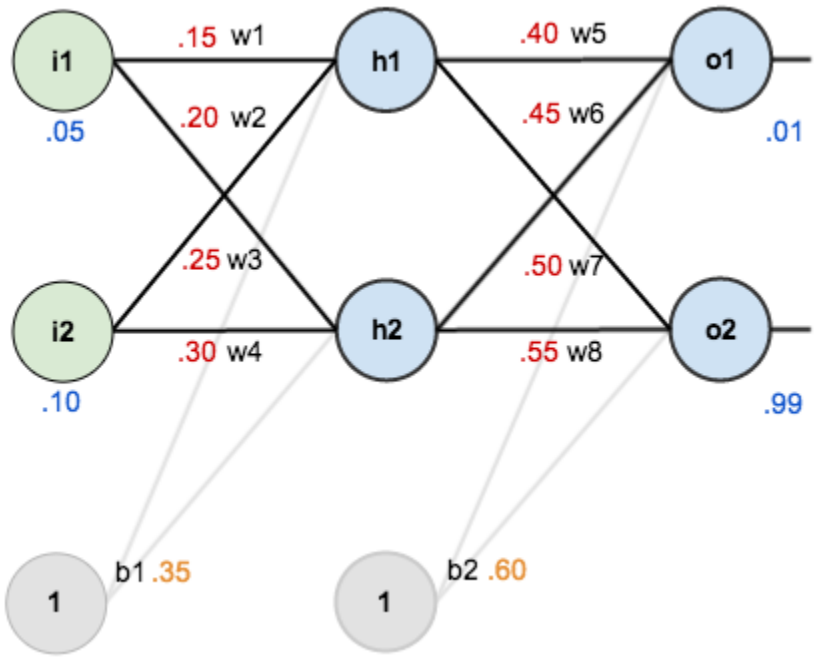
Calculating the Total Error

- We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Calculating the Total Error (contd.)

Ground-truth/true-values:
O1=0.01
O2= 0.99



For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

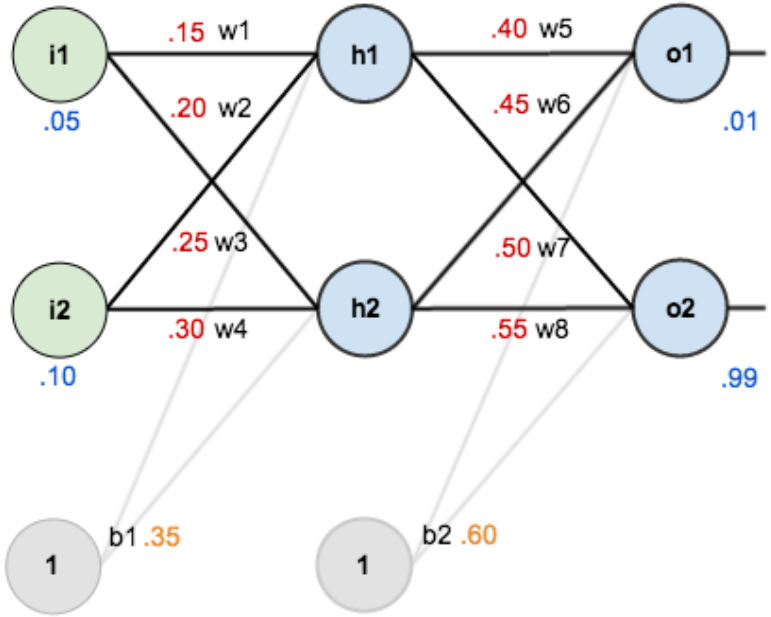
$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

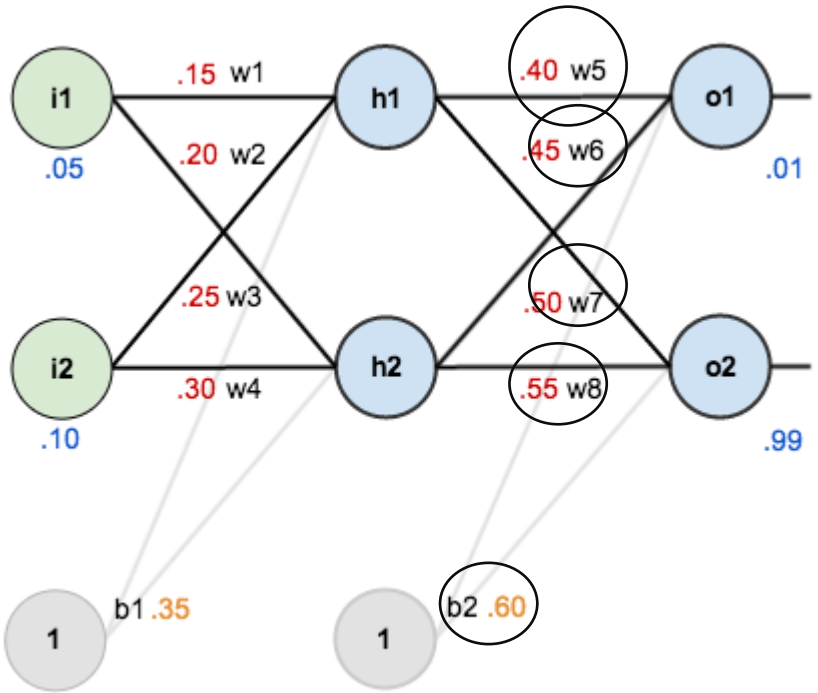
The Backwards Pass

- Our goal with backpropagation is **to update each of the weights** in the network so that they cause the **actual output to be closer the target output**, thereby **minimizing the error** for each output neuron and the network as a whole.



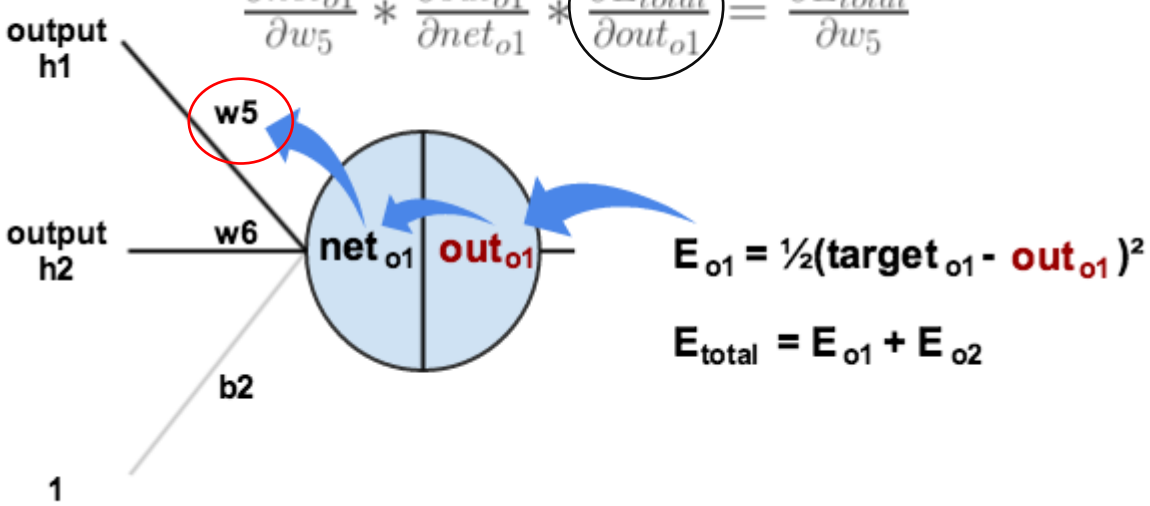
The weight-update scheme for the nodes of <u>Output Layer</u> and <u>hidden layer</u> are slightly different. We will see each of them.

Weight Update at Output Layer



Weight Update at Output Layer

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



Consider w_5. We want to know how much a change in w_5 affects the total error, aka $\frac{\partial E_{total}}{\partial w_5}$.

By applying the chain rule we know that:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

We need to figure out each piece in this equation.

First, how much does the total error change with respect to the output?

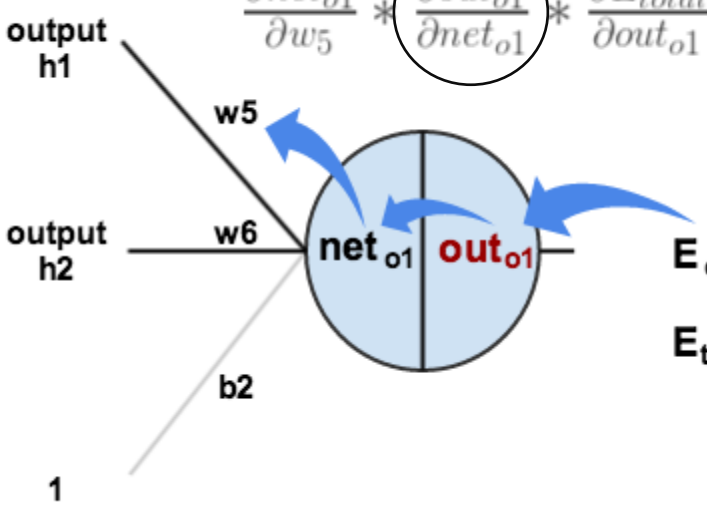
$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Weight Update at Output Layer

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

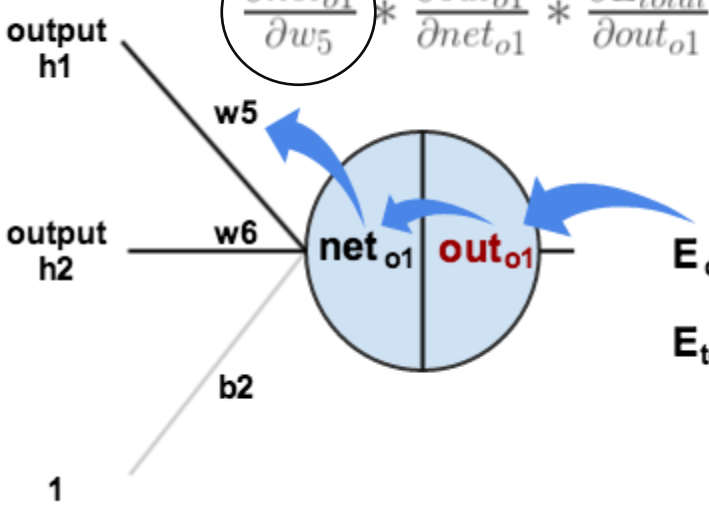
Next, how much does the output of o_1 change with respect to its total net input?

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Weight Update at Output Layer

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

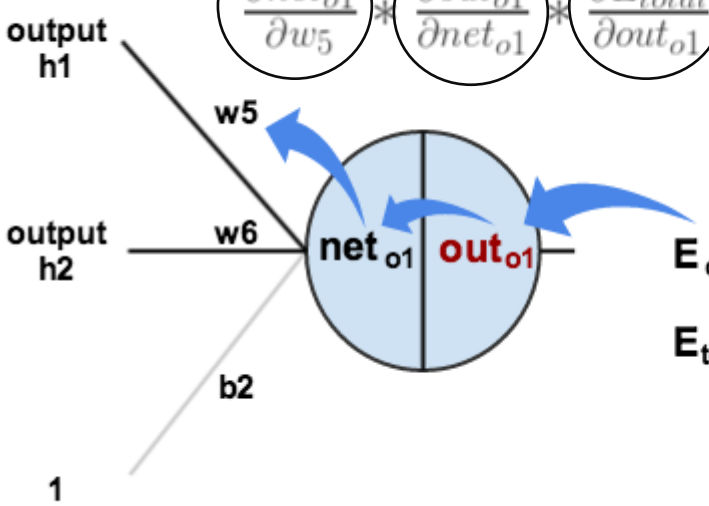
Finally, how much does the total net input of $o1$ change with respect to w_5?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

Weight Update at Output Layer

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

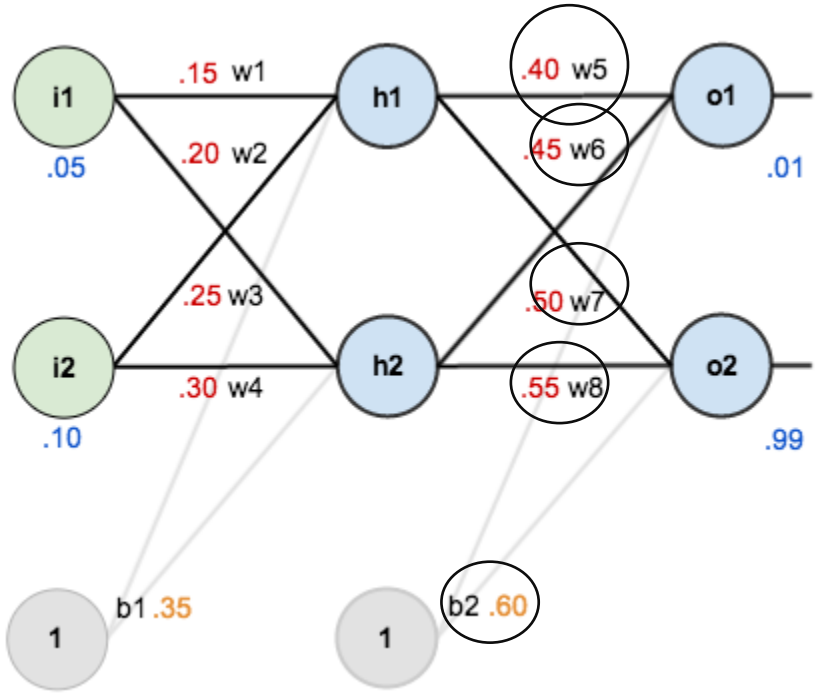
$E_{total} = E_{o1} + E_{o2}$

Putting it all together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

Weight Update at Output Layer



To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

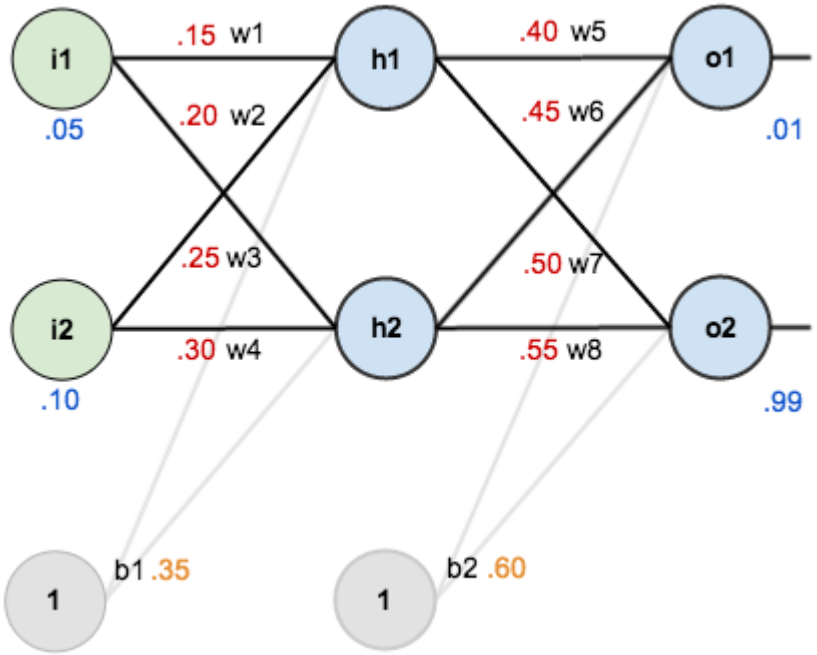
We can repeat this process to get the new weights w_6, w_7, and w_8:

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

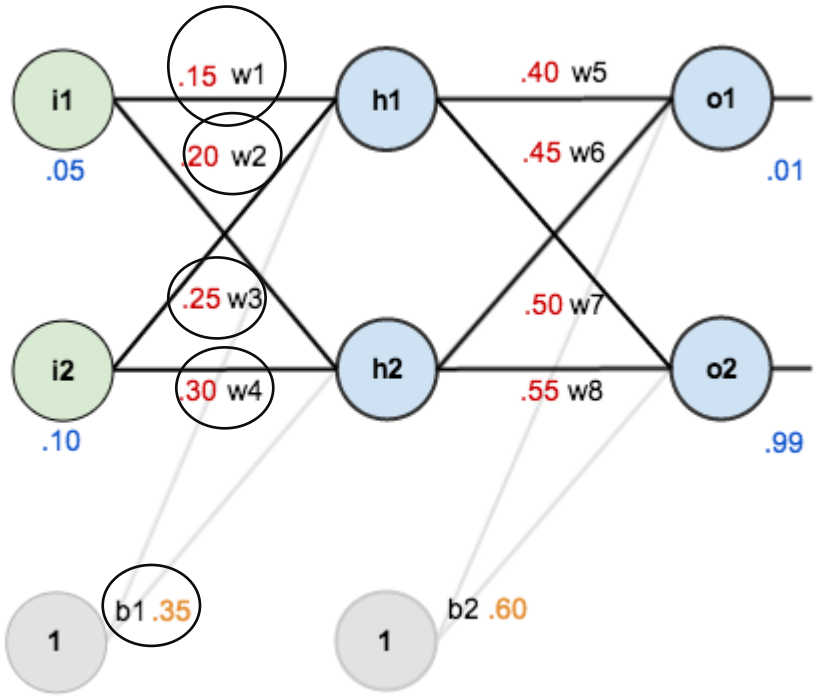
$$w_8^+ = 0.561370121$$

Home Task



- Calculate updated weight of w8 after the first forward pass.

Weight Update at Hidden Layer (Self-Study)

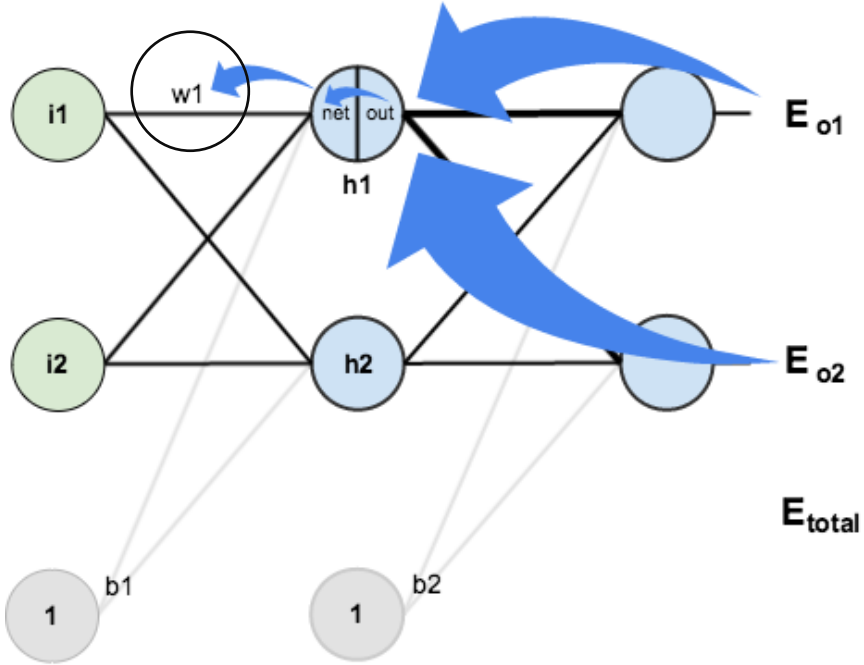


Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\downarrow$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

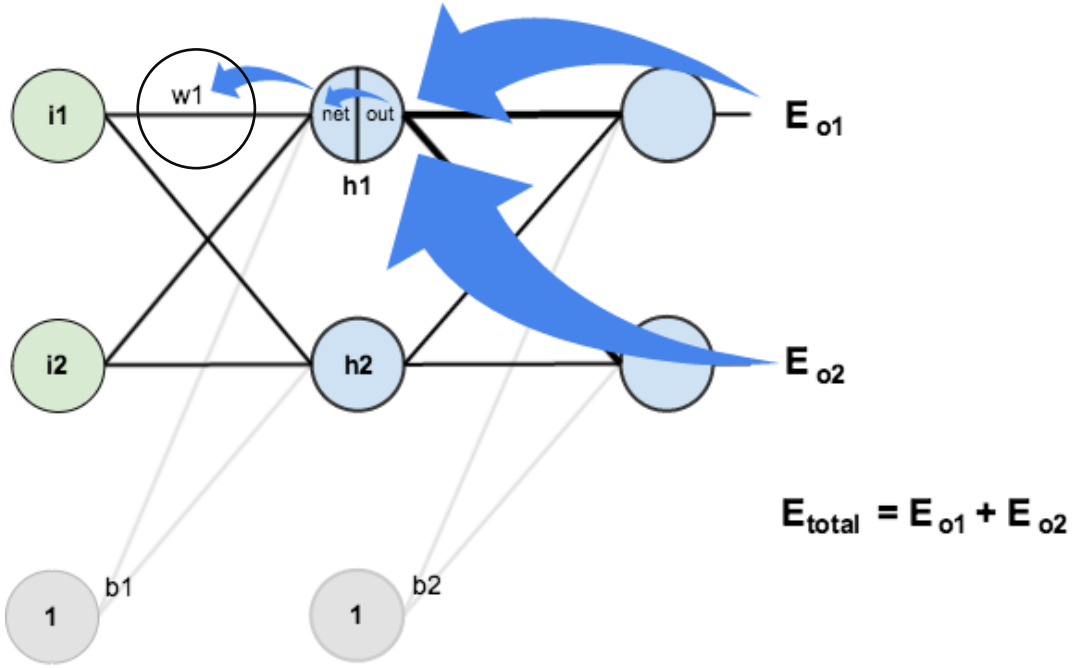
We're going to use a similar process as we did for the output layer, but slightly different to account for the fact that the output of each hidden layer neuron contributes to the output (and therefore error) of multiple output neurons. We know that out_{h1} affects both out_{o1} and out_{o2} therefore the $\frac{\partial E_{total}}{\partial out_{h1}}$ needs to take into consideration its effect on the both output neurons:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

Starting with $\frac{\partial E_{o1}}{\partial out_{h1}}$:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

We can calculate $\frac{\partial E_{o1}}{\partial net_{o1}}$ using values we calculated earlier:

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

And $\frac{\partial net_{o1}}{\partial out_{h1}}$ is equal to w_5:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

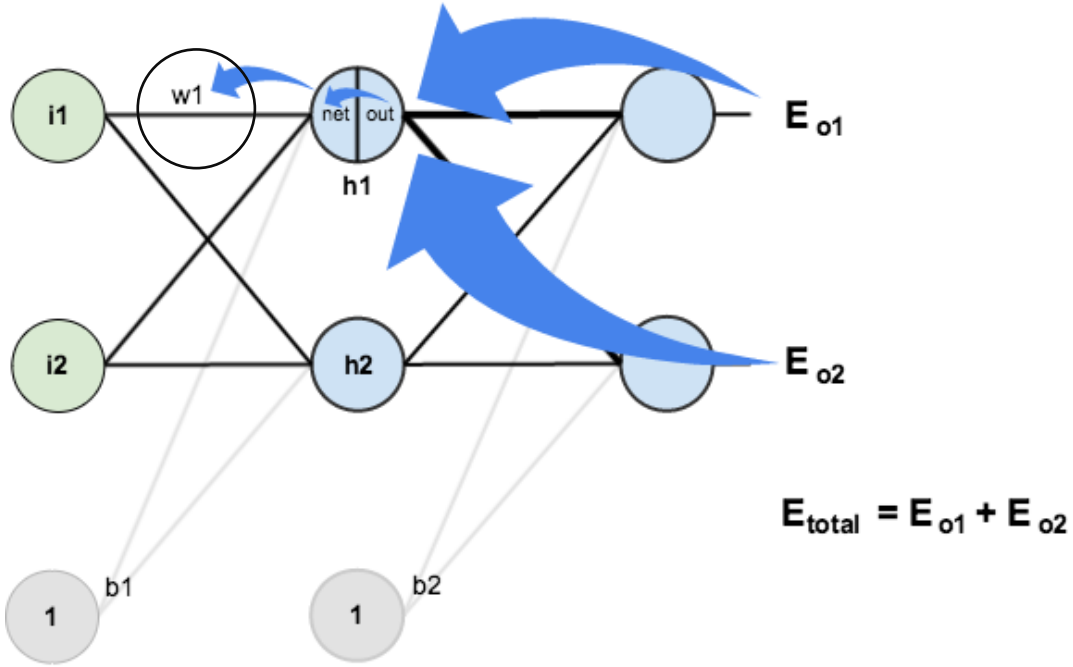
Plugging them in:

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Following the same process for $\frac{\partial E_{o2}}{\partial out_{h1}}$, we get:

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

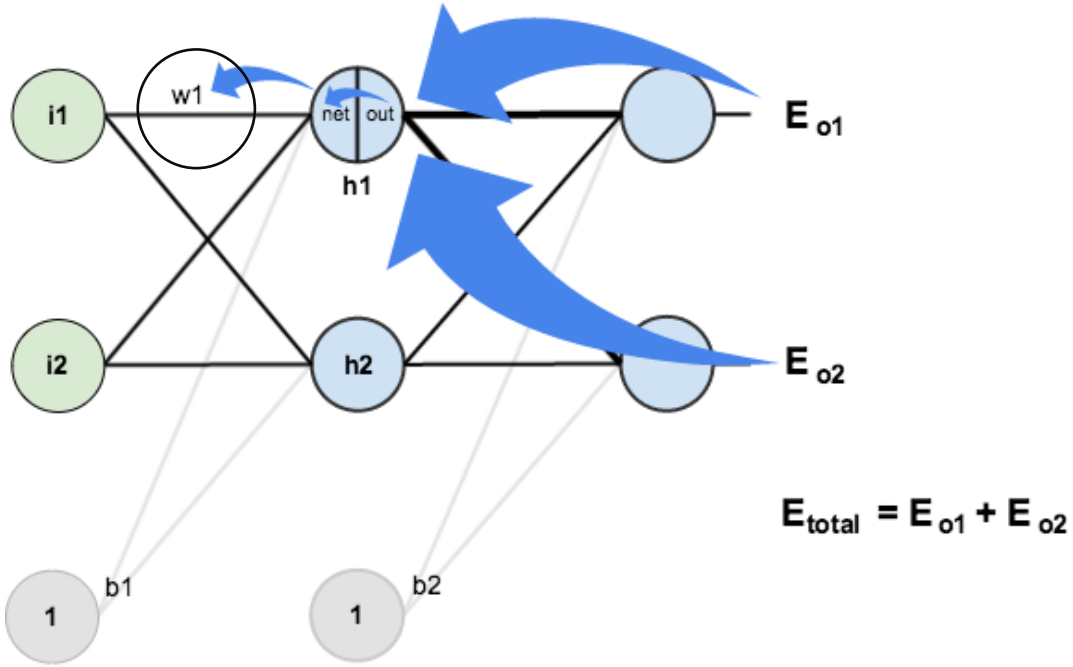
Therefore:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

Now that we have $\frac{\partial E_{total}}{\partial out_{h1}}$, we need to figure out $\frac{\partial out_{h1}}{\partial net_{h1}}$ and then $\frac{\partial net_{h1}}{\partial w}$ for each weight:

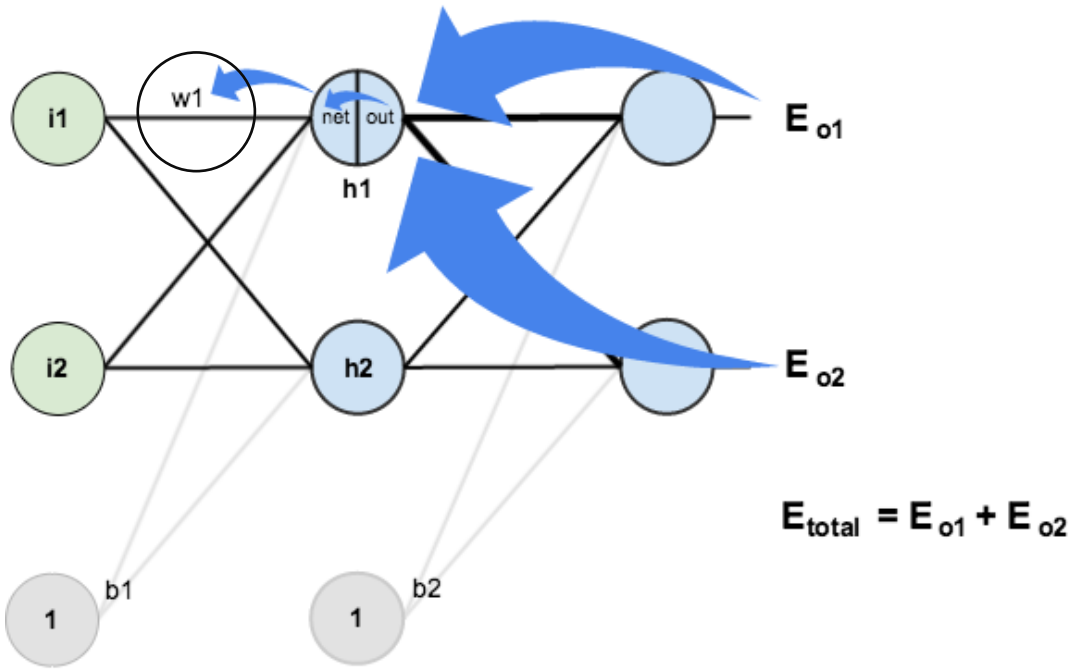
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

We calculate the partial derivative of the total net input to h_1 with respect to w_1 the same as we did for the output neuron:

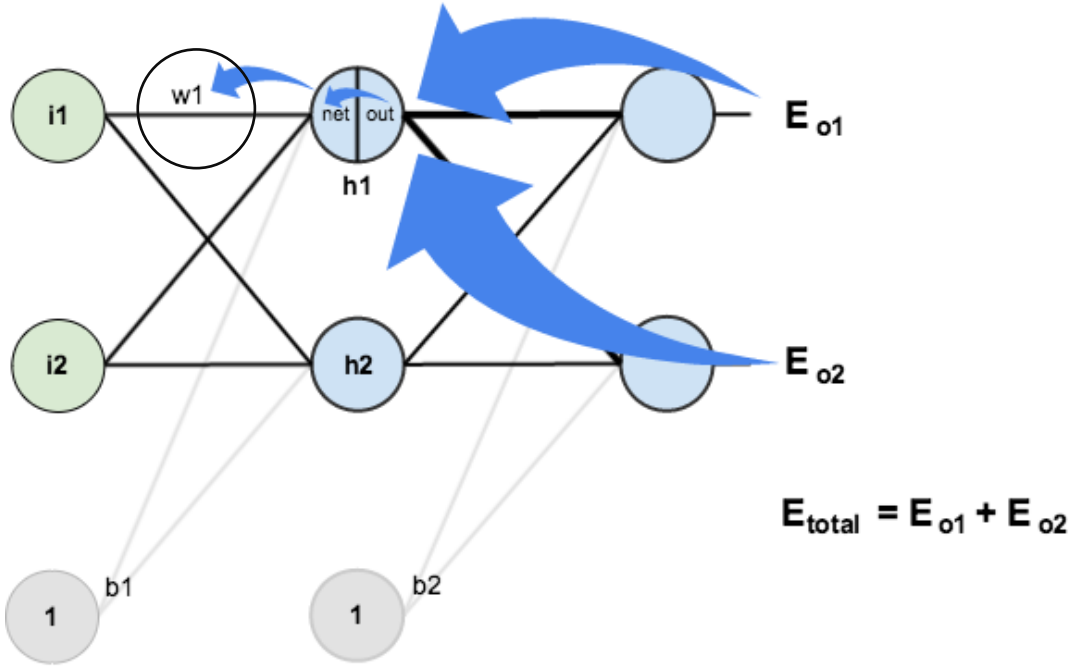
$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

Putting it all together:

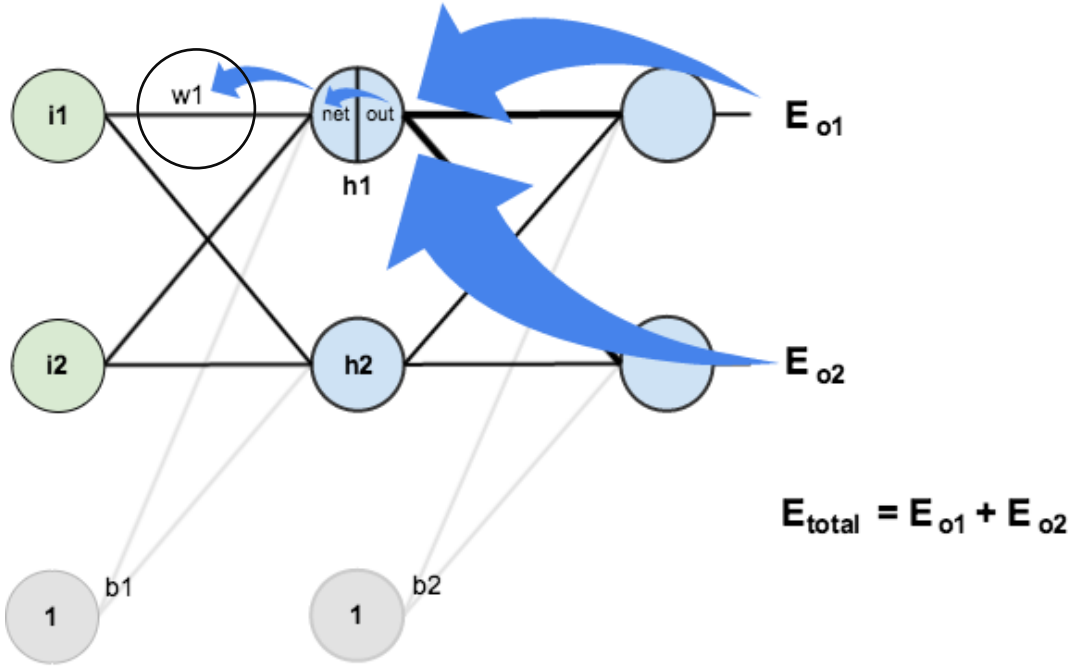
$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

Weight Update at Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



We can now update w_1:

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Repeating this for w_2, w_3, and w_4

$$w_2^+ = 0.19956143$$

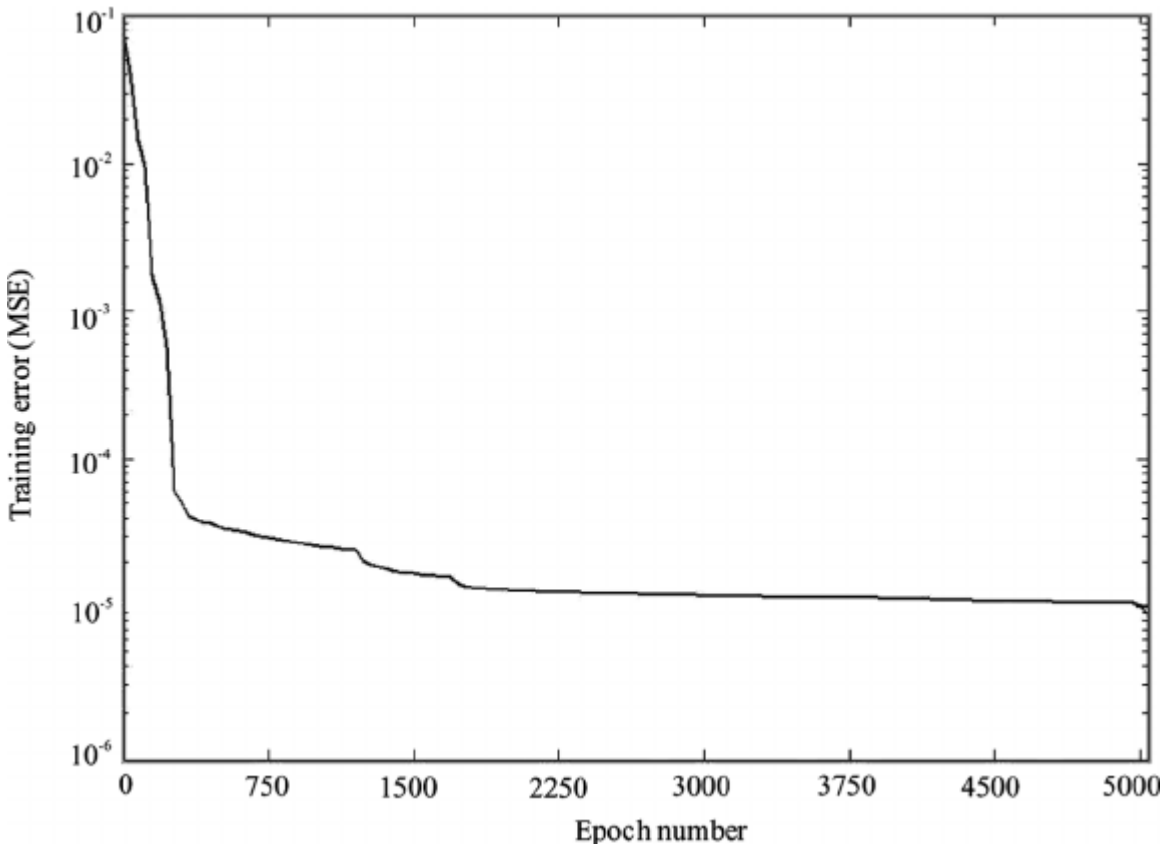
$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

Remarks!

- Finally, we've updated all of our weights!
- When we fed forward the 0.05 and 0.1 inputs originally, the error on the network was **0.298371109**.
- After this first round of backpropagation, the total error is now down to **0.291027924**.
- It might not seem like much, but after repeating this process **10,000 times**, for example, the error drops to **0.0000351085**.
- At this point, when we feed forward 0.05 and 0.1, the two outputs neurons generate **0.015912196 (vs 0.01 target)** and **0.984065734 (vs 0.99 target).**

A typical Error vs Epoch graph



Notes

Epoch :

An Epoch represent one iteration over the entire dataset.

Batch :

We cannot pass the entire dataset into the Neural Network at once. So, we divide the dataset into number of batches.

Iteration :

If we have 1000 images as Data ane a batch size of 20, then an Epoch should run 1000/20 = 50 iteration.

Backpropagation Algorithm

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - For each training example $<(x_1,\ldots x_n),t>$ Do
    - Input the instance $(x_1,\ldots,x_n)$ to the network and compute the network outputs o_k
    - For each output unit k
      - $\delta_k=o_k(1-o_k)(t_k-o_k)$
    - For each hidden unit h
      - $\delta_h=o_h(1-o_h)\sum_k w_{h,k}\,\delta_k$
    - For each network weight $w_{,j}$ Do
    - $w_{i,j}=w_{i,j}+\Delta w_{i,j}$ where

      $\Delta w_{i,j}=\eta\,\delta_j\,x_{i,j}$

Backpropagation

- Gradient descent over entire *network* weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum

  -in practice often works well (can be invoked multiple times with different initial weights)
- Often include weight *momentum* term

$$\Delta w_{i,j}(t) = \eta\, \delta_j\, x_{i,j} + \alpha\, \Delta w_{i,j}(t-1)$$

- Minimizes error training examples
  - Will it generalize well to unseen instances (over-fitting)?
- Training can be slow typical 1000-10000 iterations

  (use Levenberg-Marquardt instead of gradient descent)
- Using network after training is fast

Throwback to gradient Descent

Gradient descent

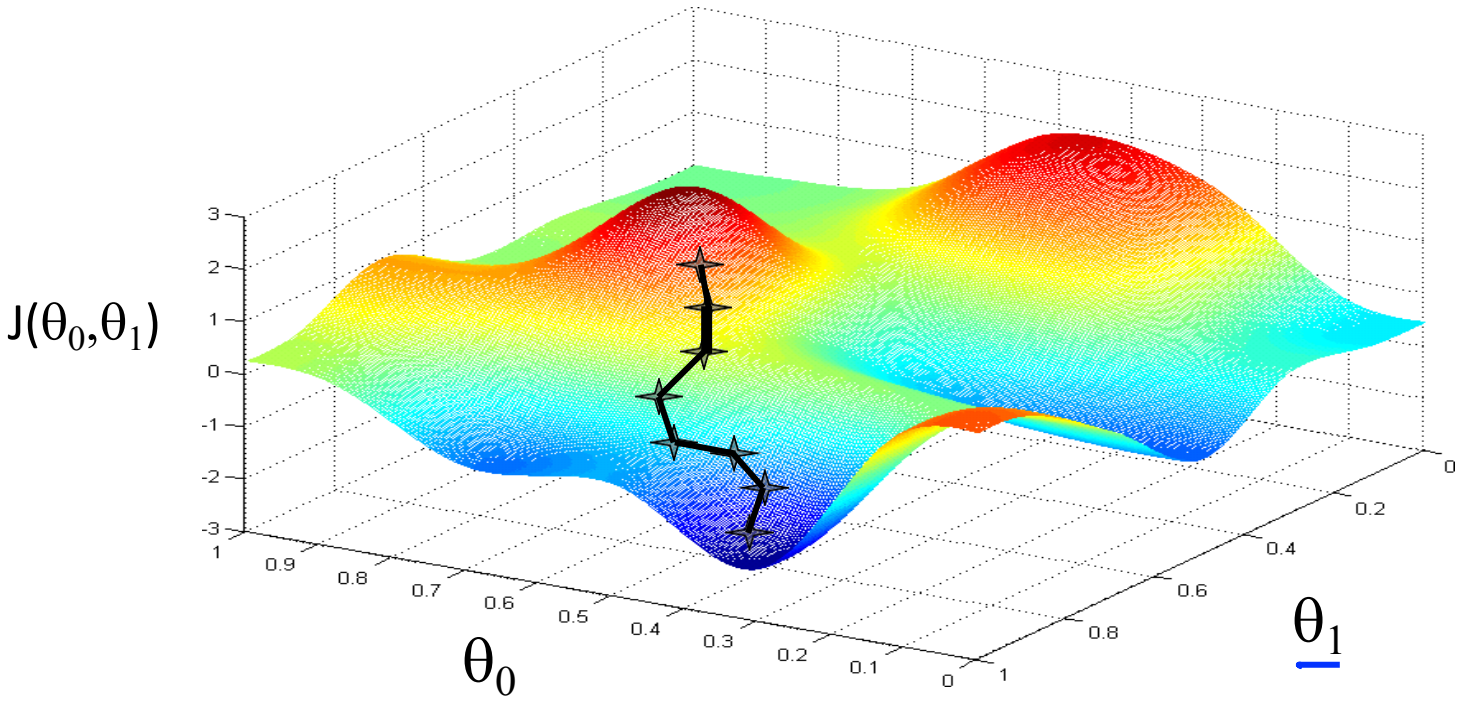
Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0 \ldots \theta_n} J(\theta_0, \ldots, \theta_n)$

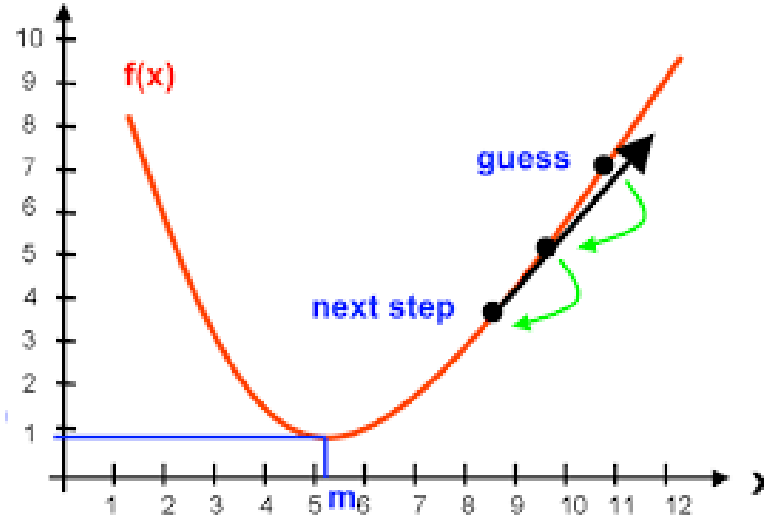
Outline:

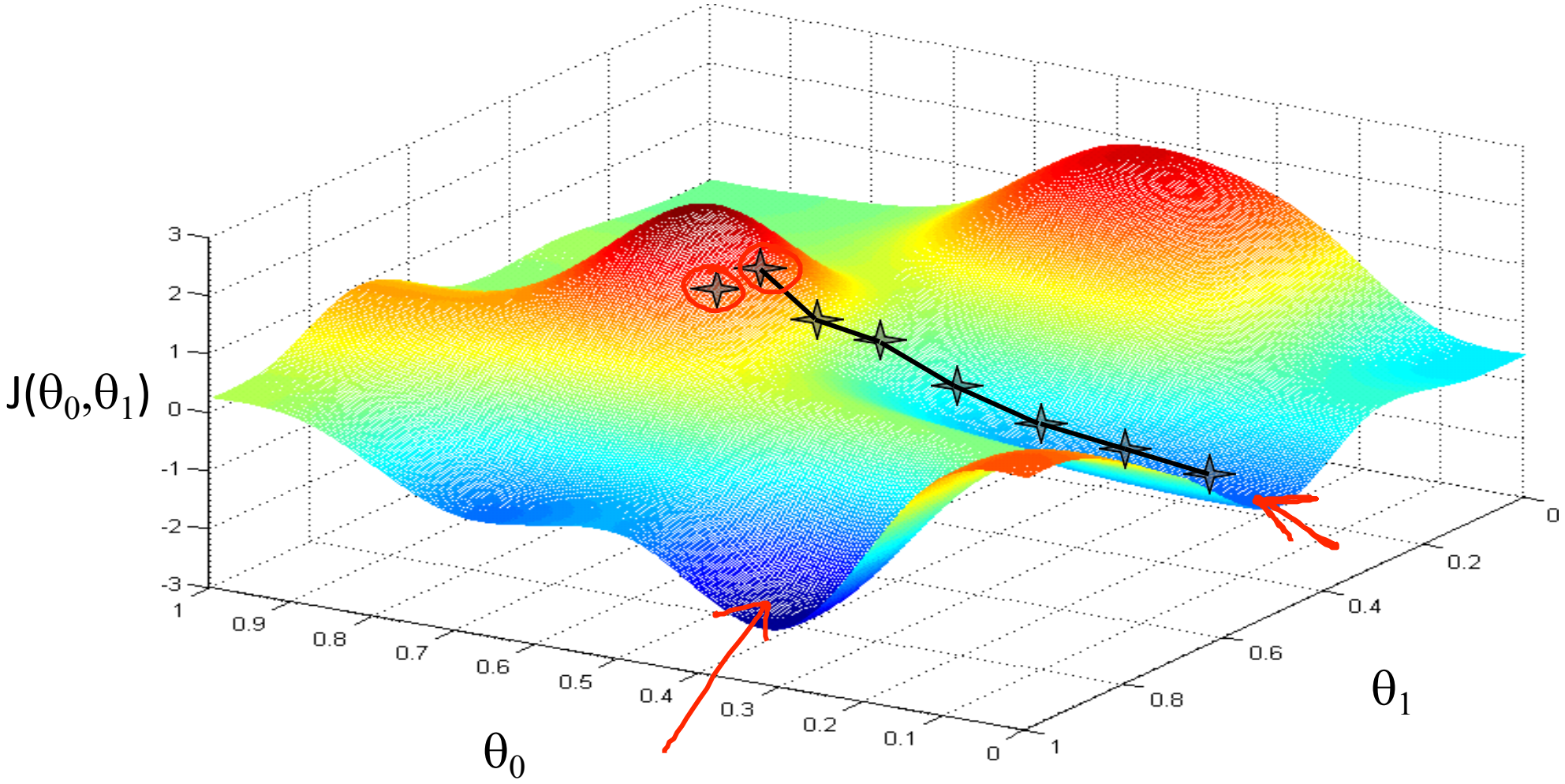
- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

3D-View



2D-View




$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Gradient descent intuition

Gradient descent algorithm

repeat until convergence {

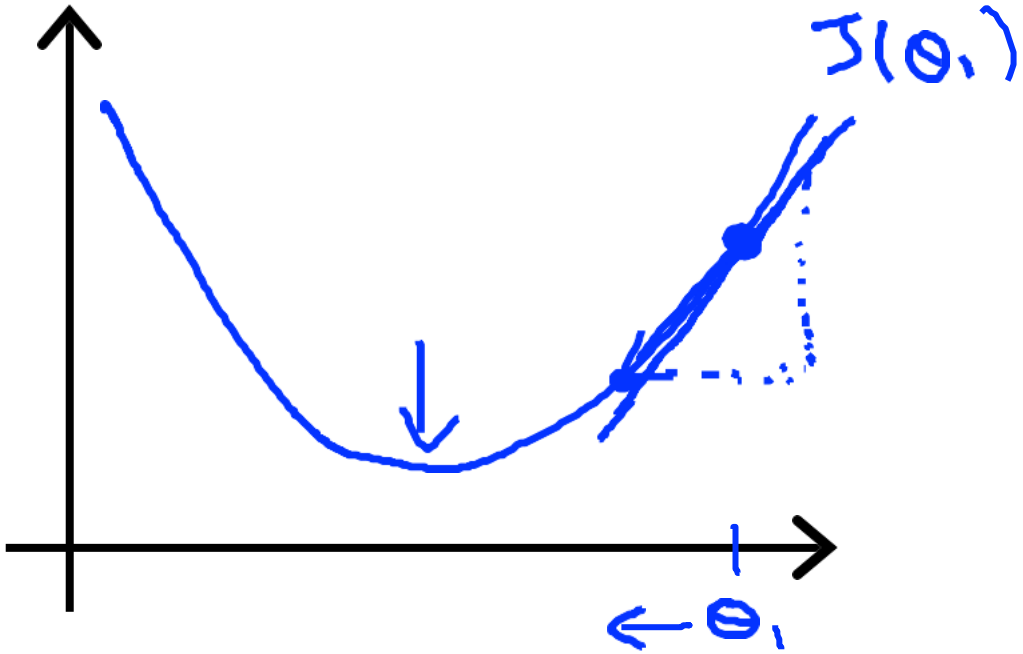
$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)} \qquad \text{(simultaneously update } j = 0 \text{ and } j = 1)$$

}

learning rate

derivative

$$\min_{\theta_1} J(\theta_1) \qquad \theta_1 \in \mathbb{R}.$$

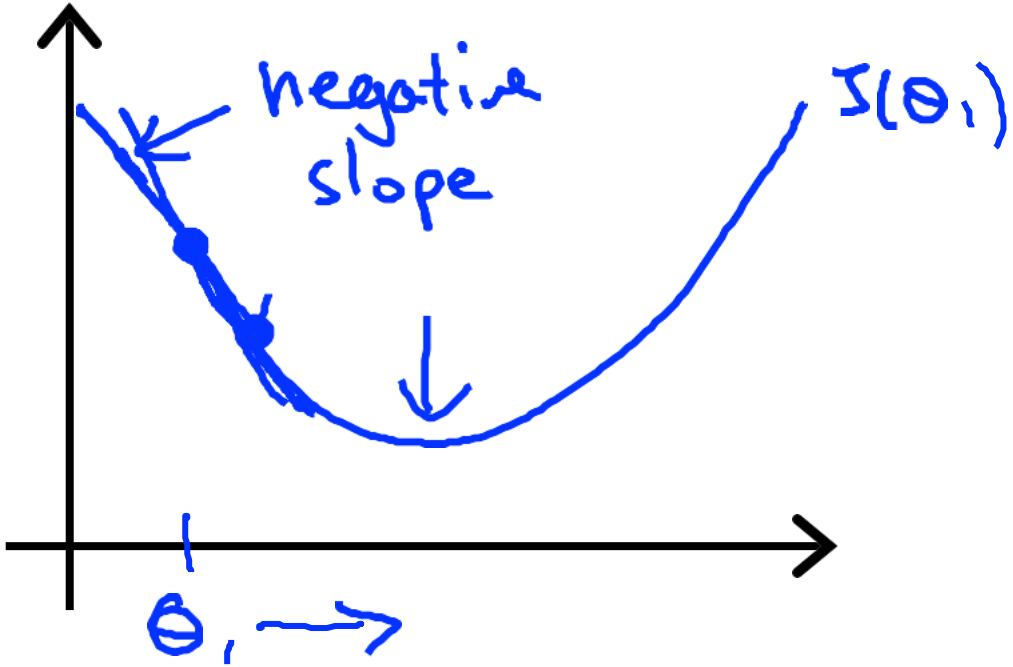


$(\theta_1 \in \mathbb{R})$

$$\theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{\partial}{\partial \theta_1} J(\theta_1)}}_{\geq 0}$$

$\frac{\partial}{\partial \theta_1}$

$$\theta_1 := \theta_1 - \underline{\alpha} \cdot (\text{positive number})$$

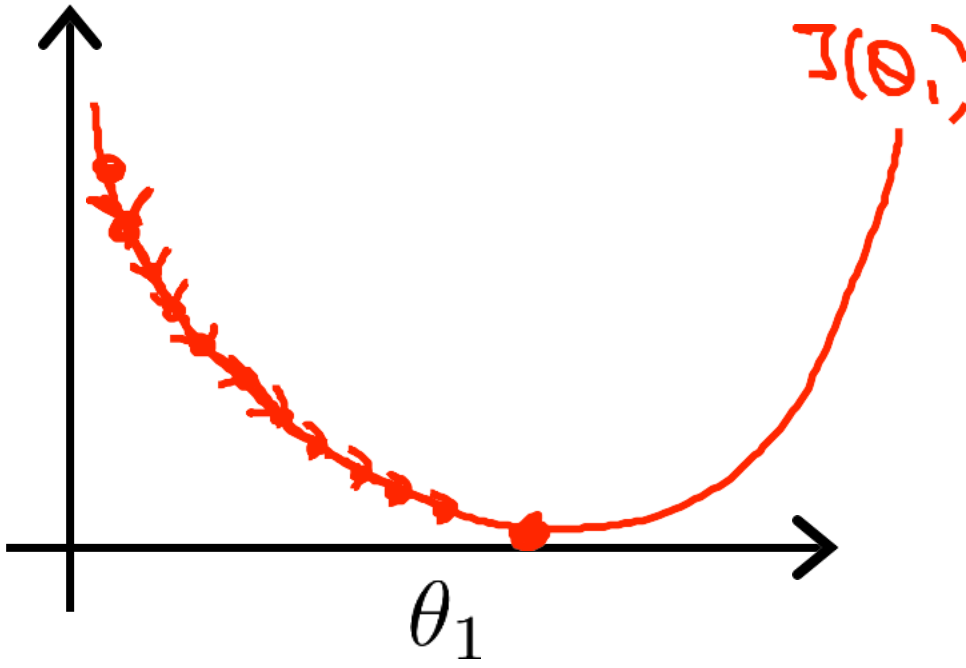


$$\underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\leq 0}$$

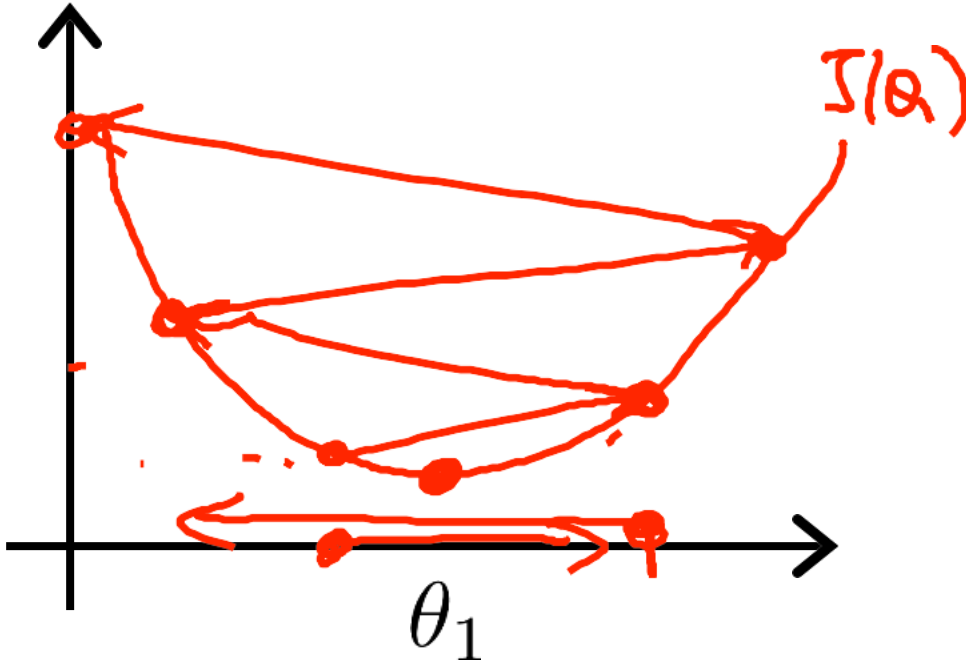
$$\theta_1 := \underline{\theta_1} - \alpha \, (\text{negative number})$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



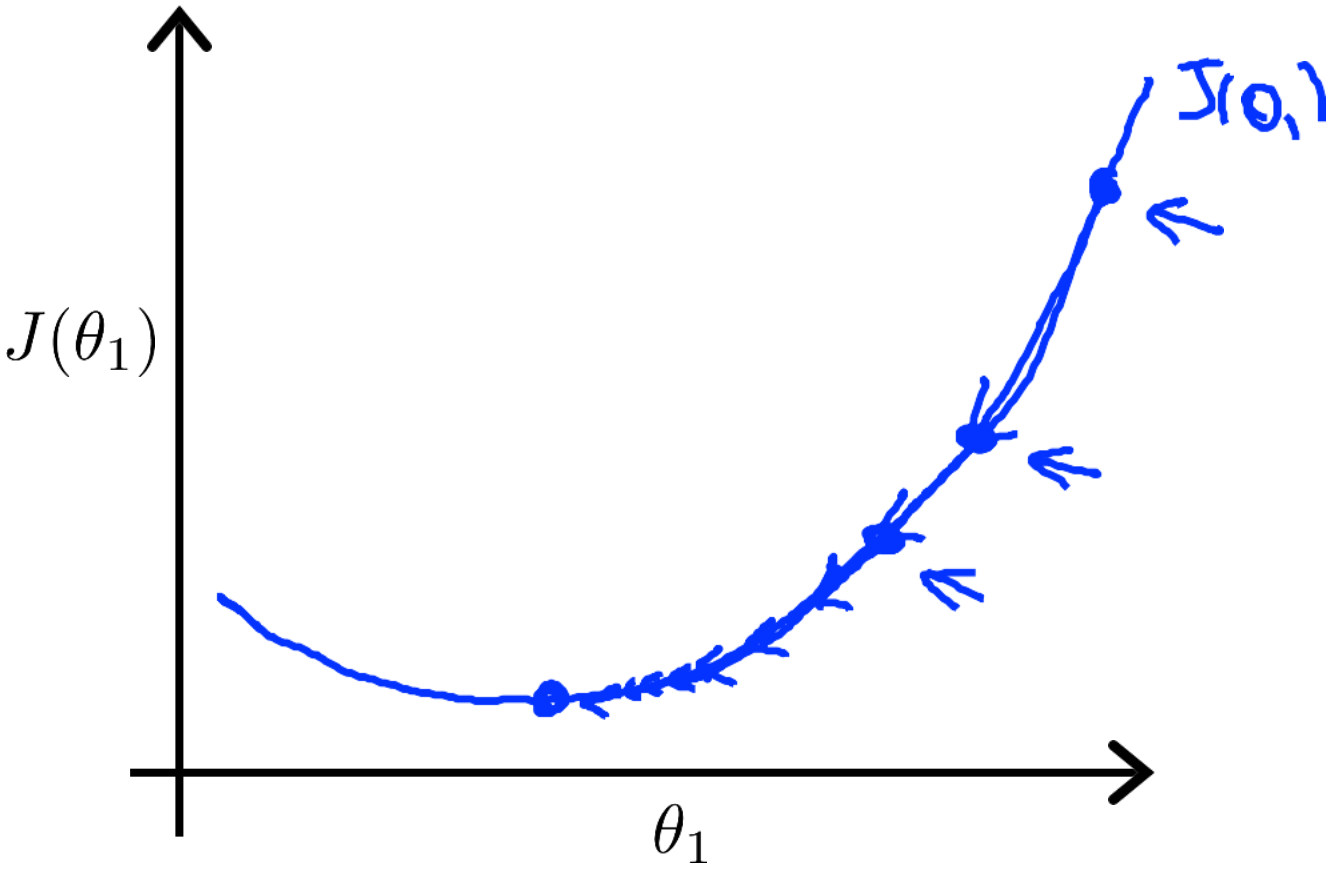
Learning Rate

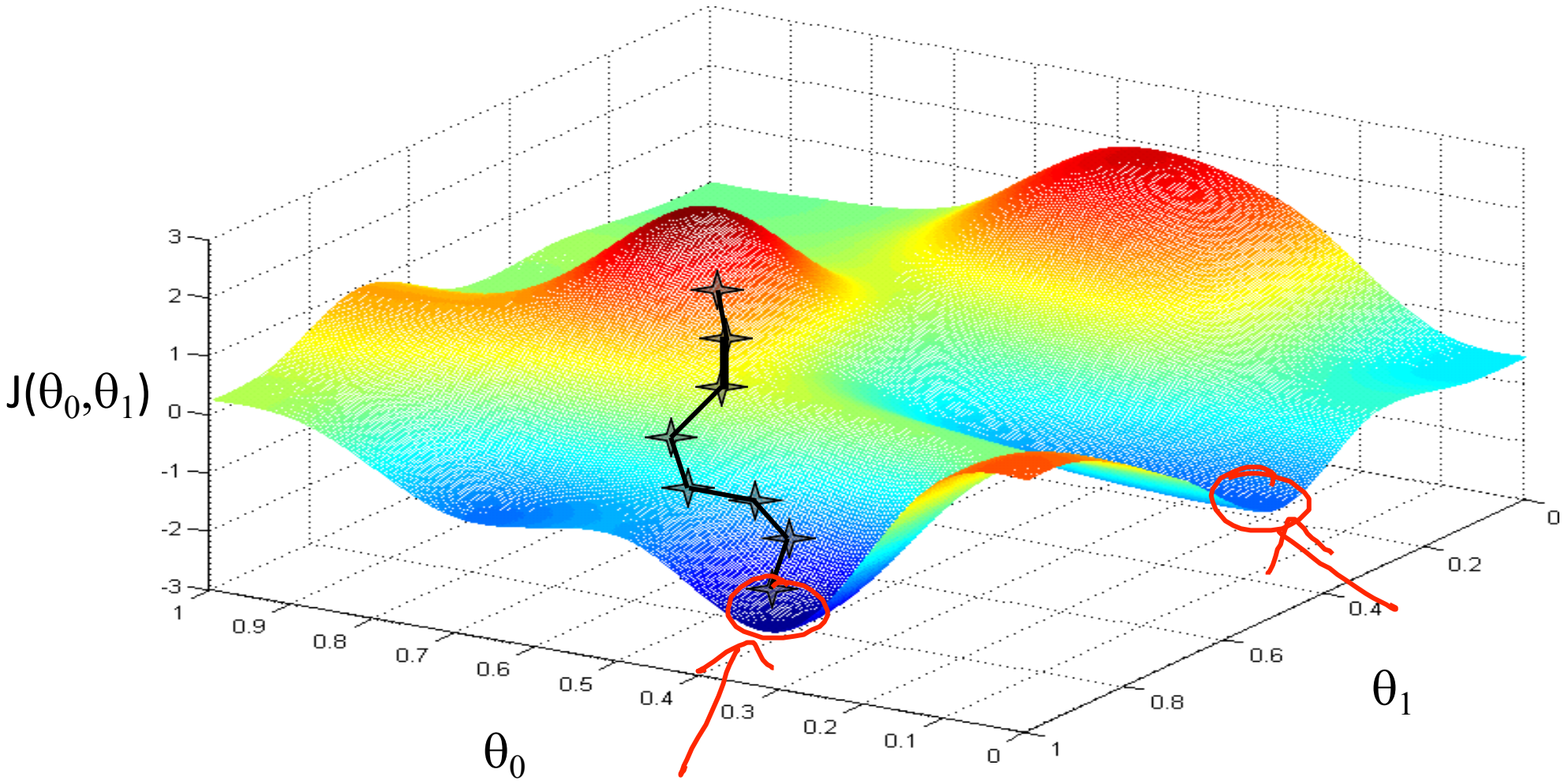
- In machine learning and statistics, the learning rate is a tuning parameter in an optimization algorithm **that determines the step size at each iteration while moving toward a minimum of a loss function**.
- Since it influences to what extent newly acquired information overrides old information, it metaphorically **represents the speed** at which a machine learning model "learns".
- In setting a learning rate, there is a **trade-off between the rate of convergence and overshooting**.
- While the descent direction is usually determined from the gradient of the loss function, the learning rate **determines how big a step is taken in that direction**.
- **A too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge** or get stuck in an undesirable local minimum.

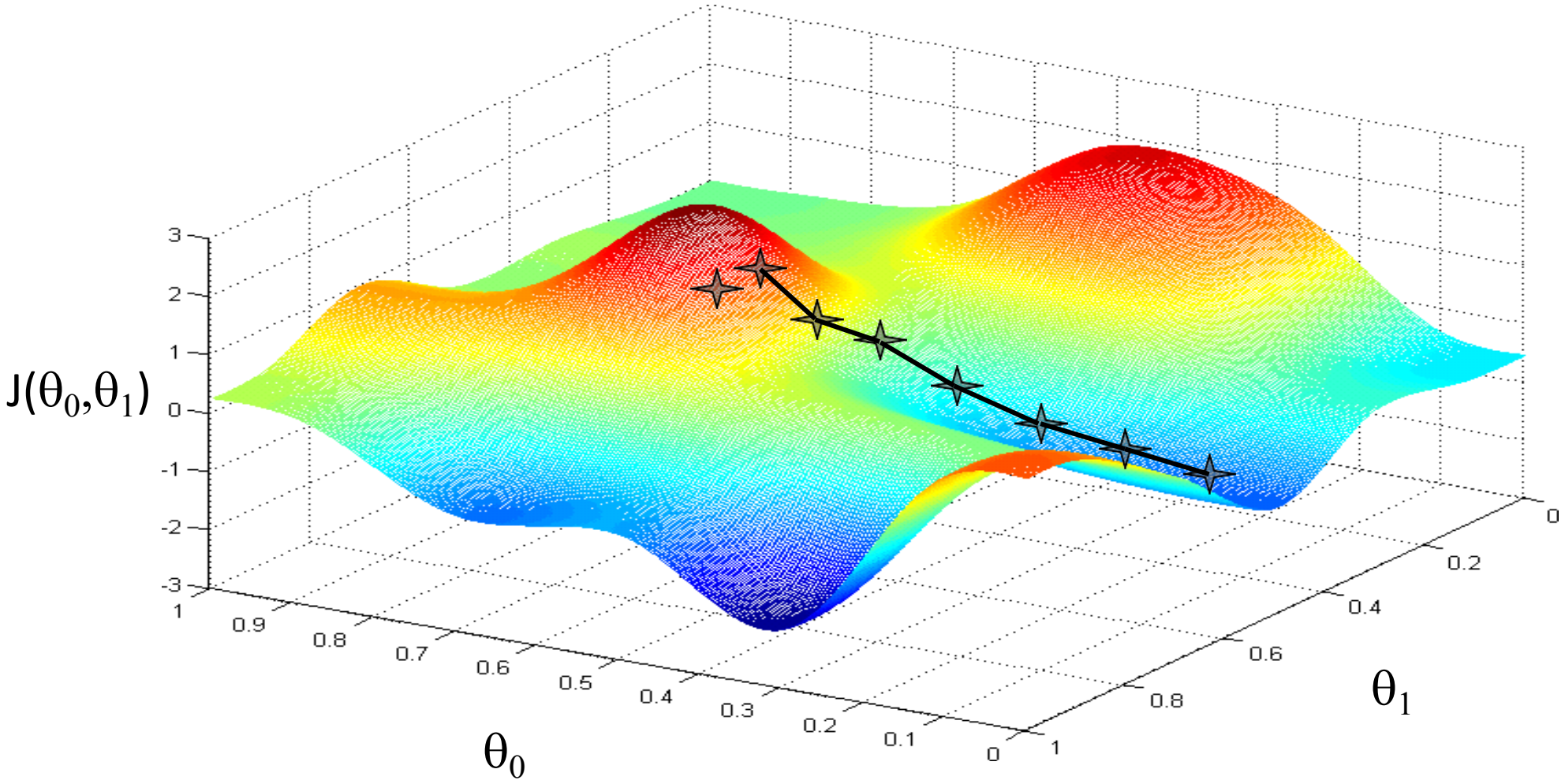
Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)}$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.




$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Clustering

Clustering

Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Clustering: a definition

"The process of organizing objects into *groups* whose members are *similar in some way*"

J.A. Hartigan, 1975

"An algorithm by which objects are grouped in *classes*, so that intra-class *similarity* is maximized and inter-class similarity is minimized"

J. Han and M. Kamber, 2000

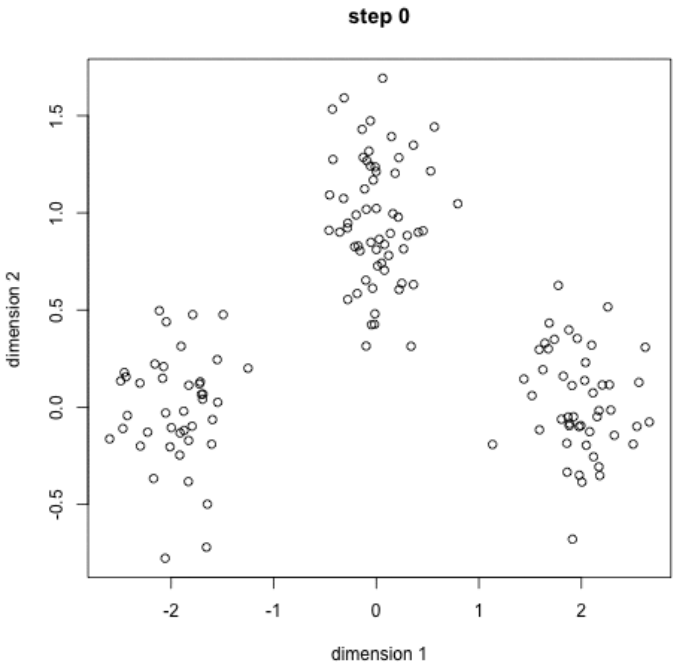
"... grouping or segmenting a collection of objects into subsets or *clusters*, such that those within each cluster are more closely *related* to one another than objects assigned to different clusters"

T. Hastie, R. Tibshirani, J. Friedman, 2009

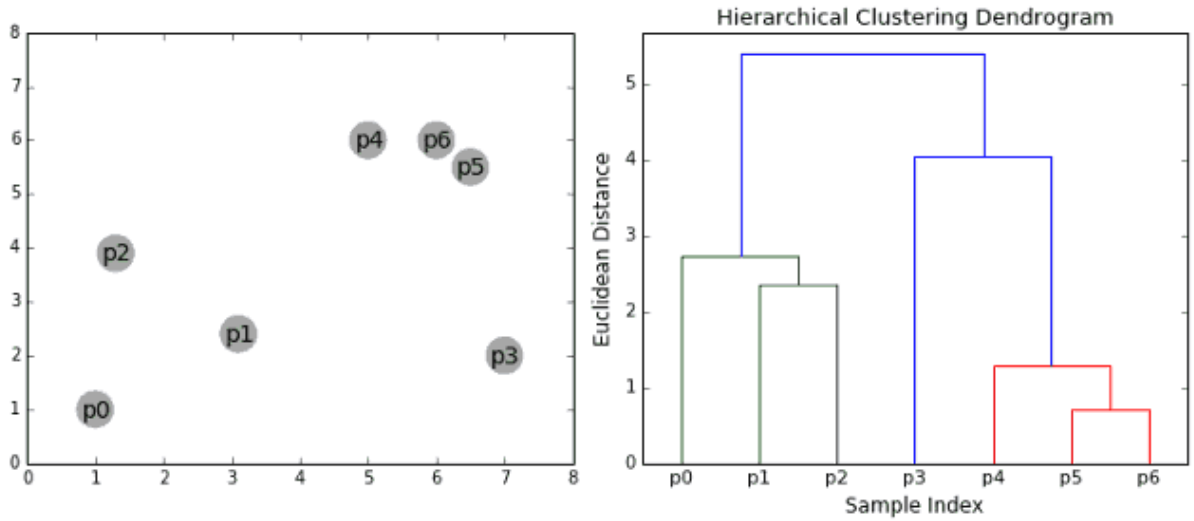
(Some) Applications of Clustering

- Market research/Insurance/Telephone Companies
  - find groups of customers with similar behavior for targeted advertising
- Biology
  - classification of plants and animals given their features
- Social Media
  - identify suggestions
  - cluster the blocked users
- On the Web:
  - document classification
  - cluster Web log data to discover groups of similar access patterns
  - recommendation systems ("If you liked this, you might also like that")

Types of Clustering

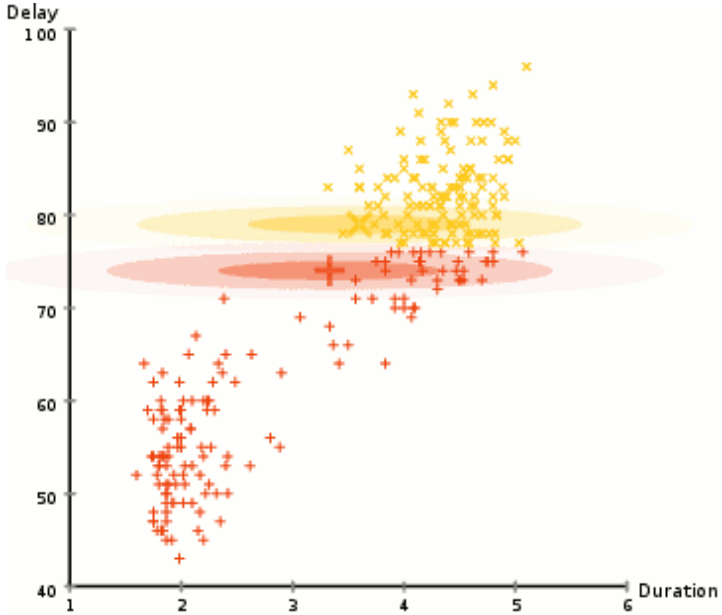


Centroid-based Clustering
(K-means)

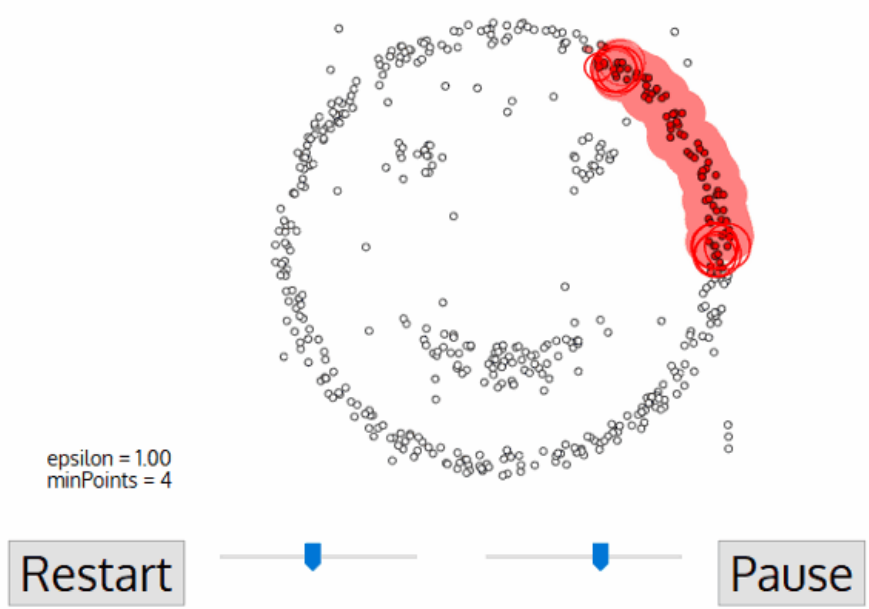


Hierarchical Clustering
(Agglomerative Hierarchical Clustering)

Types of Clustering



Distribution-based Clustering
(Gaussian Mixture-model based)



Density-based Clustering
(DB-SCAN)

Distance Measures

Two major classes of distance measure:

- Euclidean
  - A Euclidean space has some number of real-valued dimensions and "dense" points
  - There is a notion of *average* of two points
  - A Euclidean distance is based on the locations of points in such a space

$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

- Non-Euclidean
  - A Non-Euclidean distance is based on properties of points, but not on their *location* in a space

Distances vs Similarities

- Distances are normally used to measure the similarity or dissimilarity between two data objects...
- ... However they are two different things!
- i.e. Dissimilarities can be judged by a set of users in a survey
  - they do not necessarily satisfy the triangle inequality
  - they can be 0 even if two objects are not the same
  - they can be asymmetric (in this case their average can be calculated)

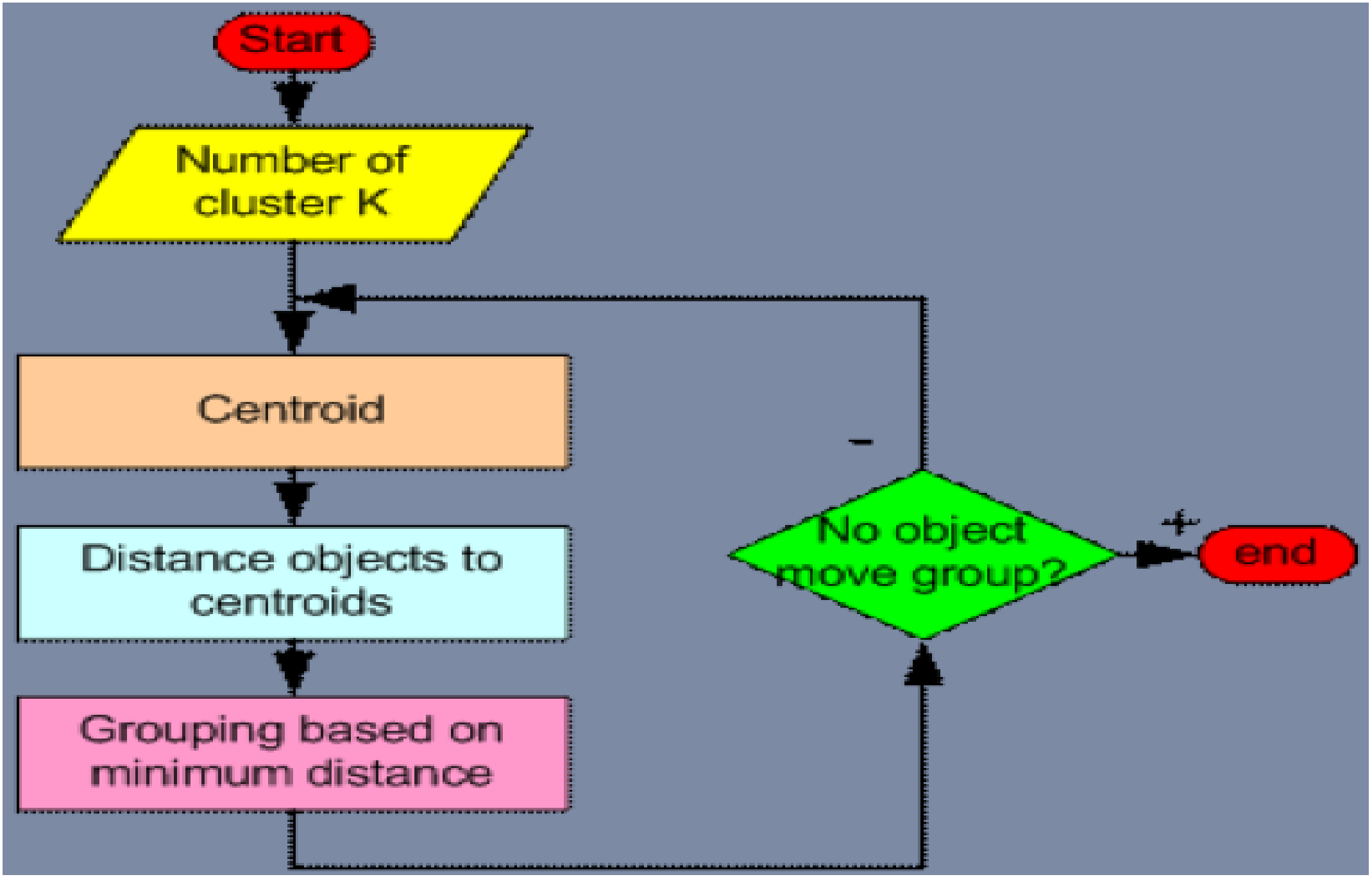
K-Means Algorithm

- One of the simplest unsupervised learning algorithms
- Assumes Euclidean space *(works with numeric data only)*
- Number of clusters fixed a priority
- **How does it work?**

K-Means Algorithm

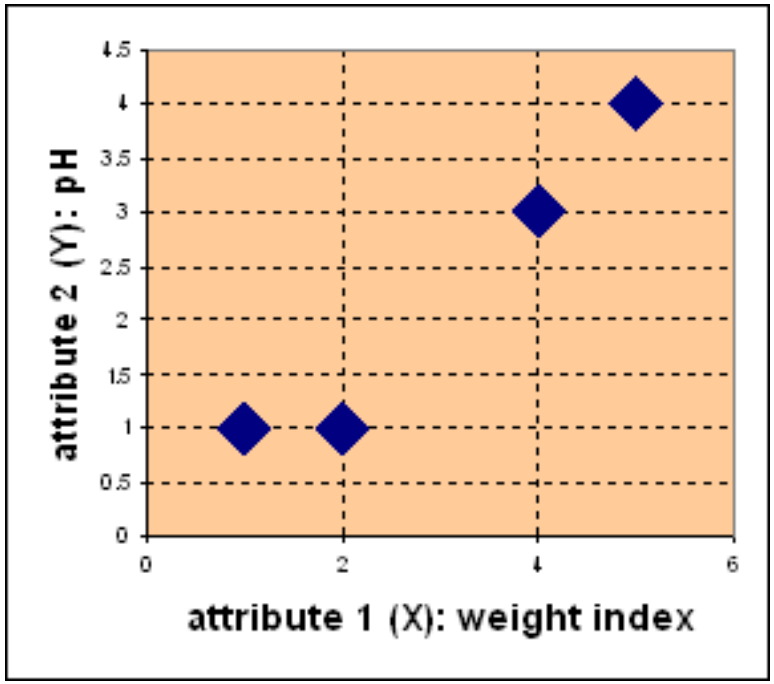
- One of the simplest unsupervised learning algorithms
- Assumes Euclidean space *(works with numeric data only)*
- Number of clusters fixed a priority
- **How does it work?**

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group *centroids*.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move.


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

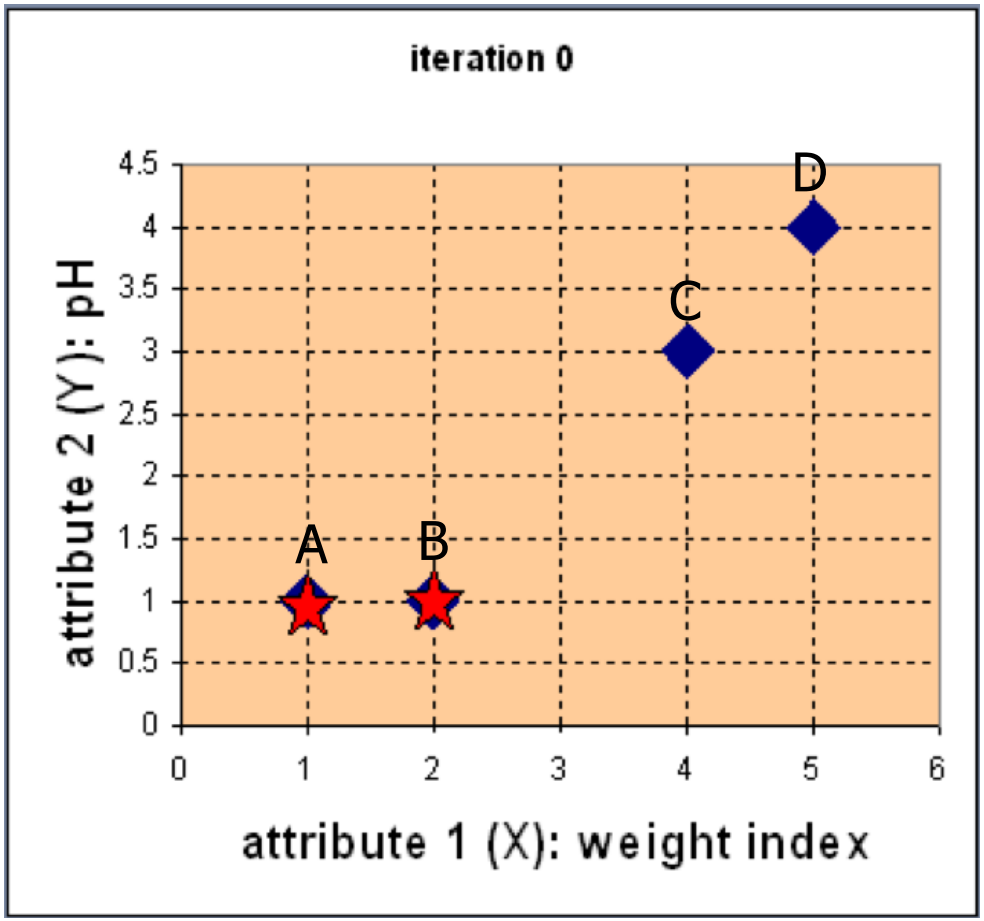
K-Means: A numerical example

Object	Attribute 1 (X)	Attribute 2 (Y)
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4



Example

Step 1: Use initial seed points for partitioning



$$c_1 = A, c_2 = B$$

$$\mathbf{D}^0 = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 3.61 & \boxed{5} \\ 1 & 0 & 2.83 & \boxed{4.24} \end{bmatrix} \end{array} \quad \begin{array}{ll} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{array}$$

Euclidean distance

$$\begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \end{array} \begin{array}{l} X \\ Y \end{array}$$

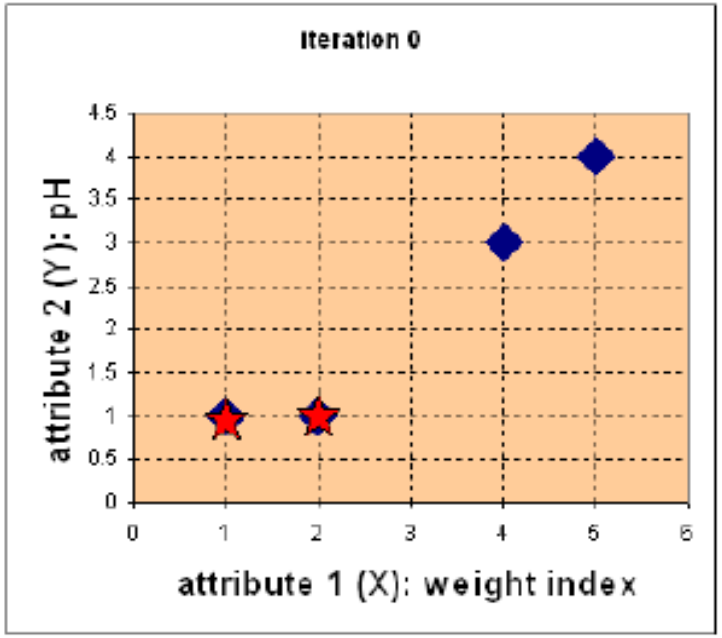
$$d(D,c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D,c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

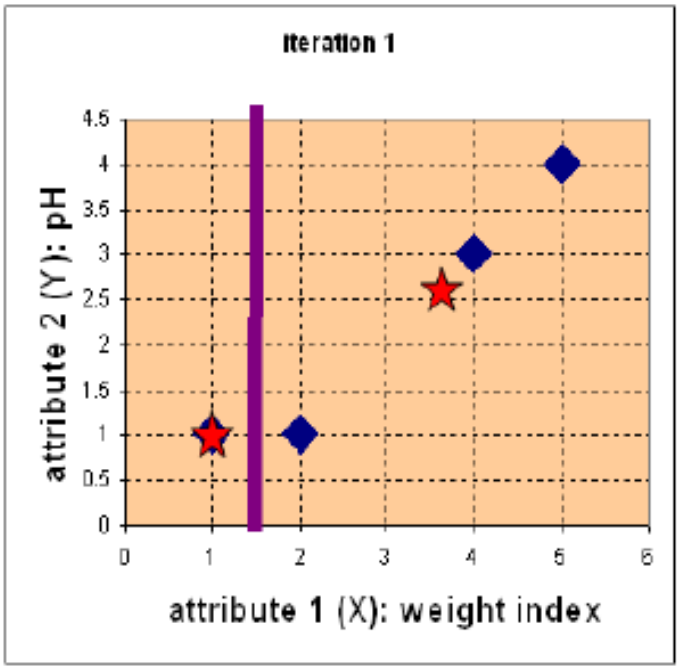
K-Means: A numerical example

- Calculate Objects-Centroids distance
  - $D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) \\ c_2 = (2,1) \end{matrix}$



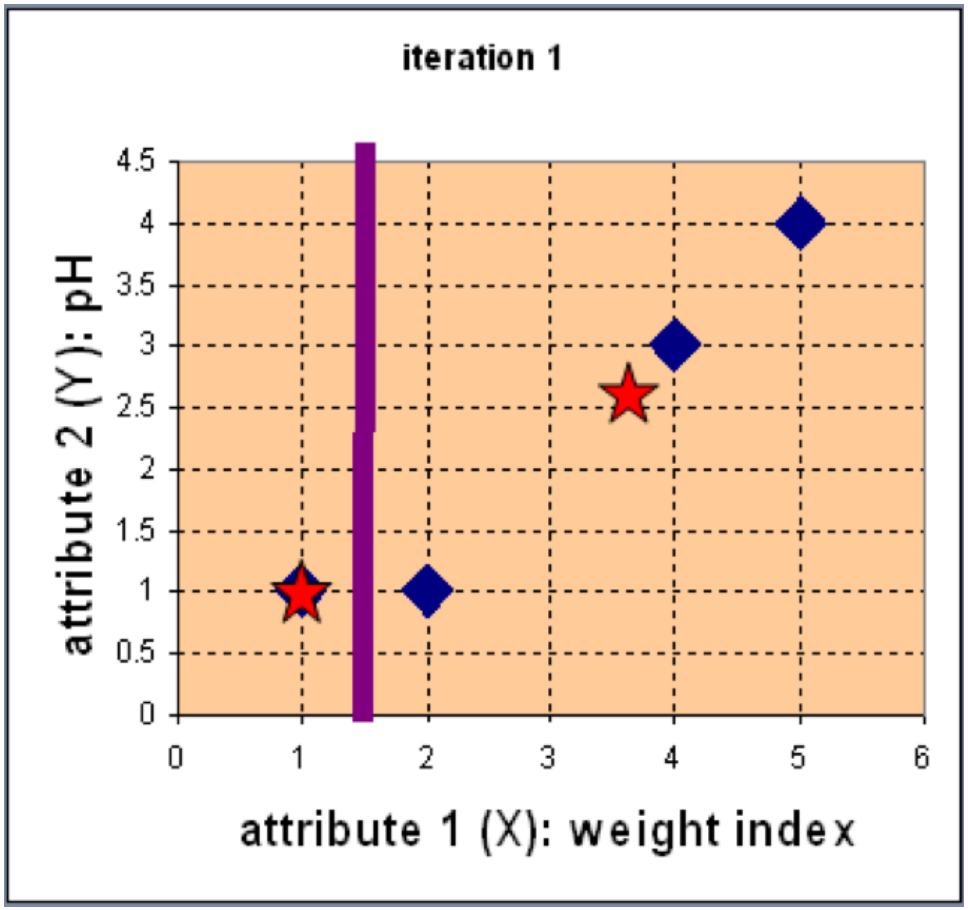
K-Means: A numerical example

- Object Clustering
  - $G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} group1 \\ group2 \end{matrix}$



Example

Step 2: Compute new centroids of the current partition



Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = (1,\ 1)$$

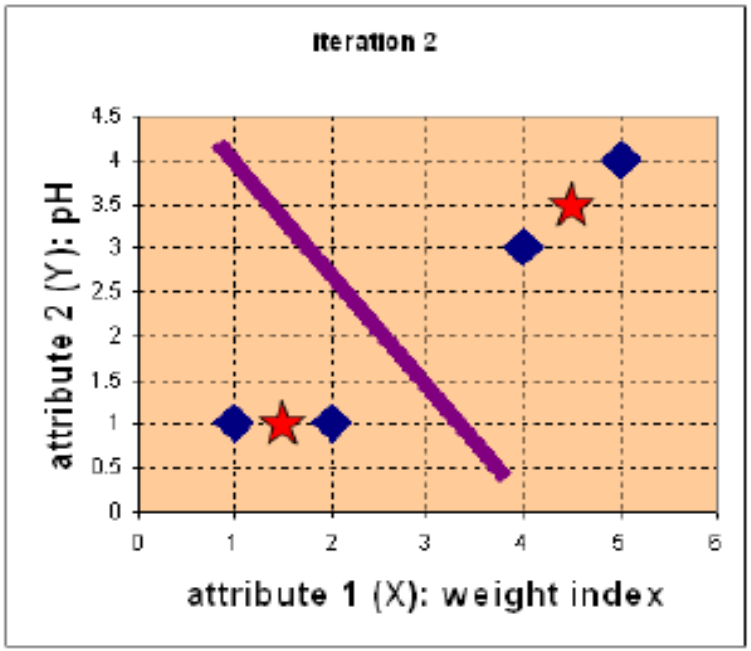
$$c_2 = \left(\frac{2+4+5}{3},\ \frac{1+3+4}{3} \right)$$

$$= (11/3,\ 8/3)$$

$$= (3.67,\ 2.67)$$

K-Means: A numerical example

- $D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix}$ $\quad c_1 = (1,1)$, $c_2 = (\frac{11}{3}, \frac{8}{3})$
- $G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \Rightarrow$ $c_1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1.5, 1)$, $c_2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4.5, 3.5)$

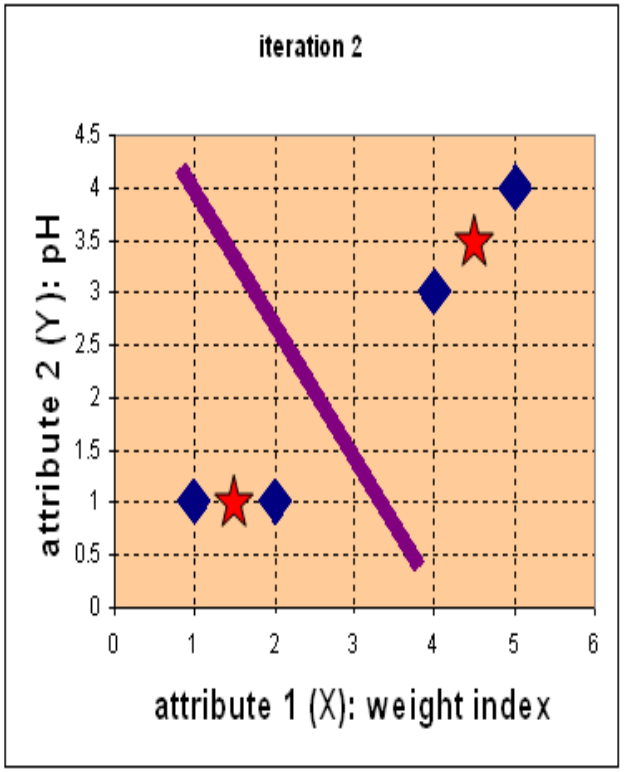


Next: determine next centroids:

Now we repeat the step to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are

$$\mathbf{c}_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1\tfrac{1}{2}, 1)$$

$$\mathbf{c}_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4\tfrac{1}{2}, 3\tfrac{1}{2})$$



Repeat step 2 again, we have new distance matrix at iteration 2 as

$$\mathbf{D}^2 = \begin{array}{c} \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array} \quad \begin{array}{ll} \mathbf{c}_1 = (1\tfrac{1}{2}, 1) & group-1 \\ \mathbf{c}_2 = (4\tfrac{1}{2}, 3\tfrac{1}{2}) & group-2 \end{array}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{array}{l} X \\ Y \end{array}$$

$$\mathbf{G}^2 = \begin{array}{c} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array} \quad \begin{array}{l} group-1 \\ group-2 \end{array}$$

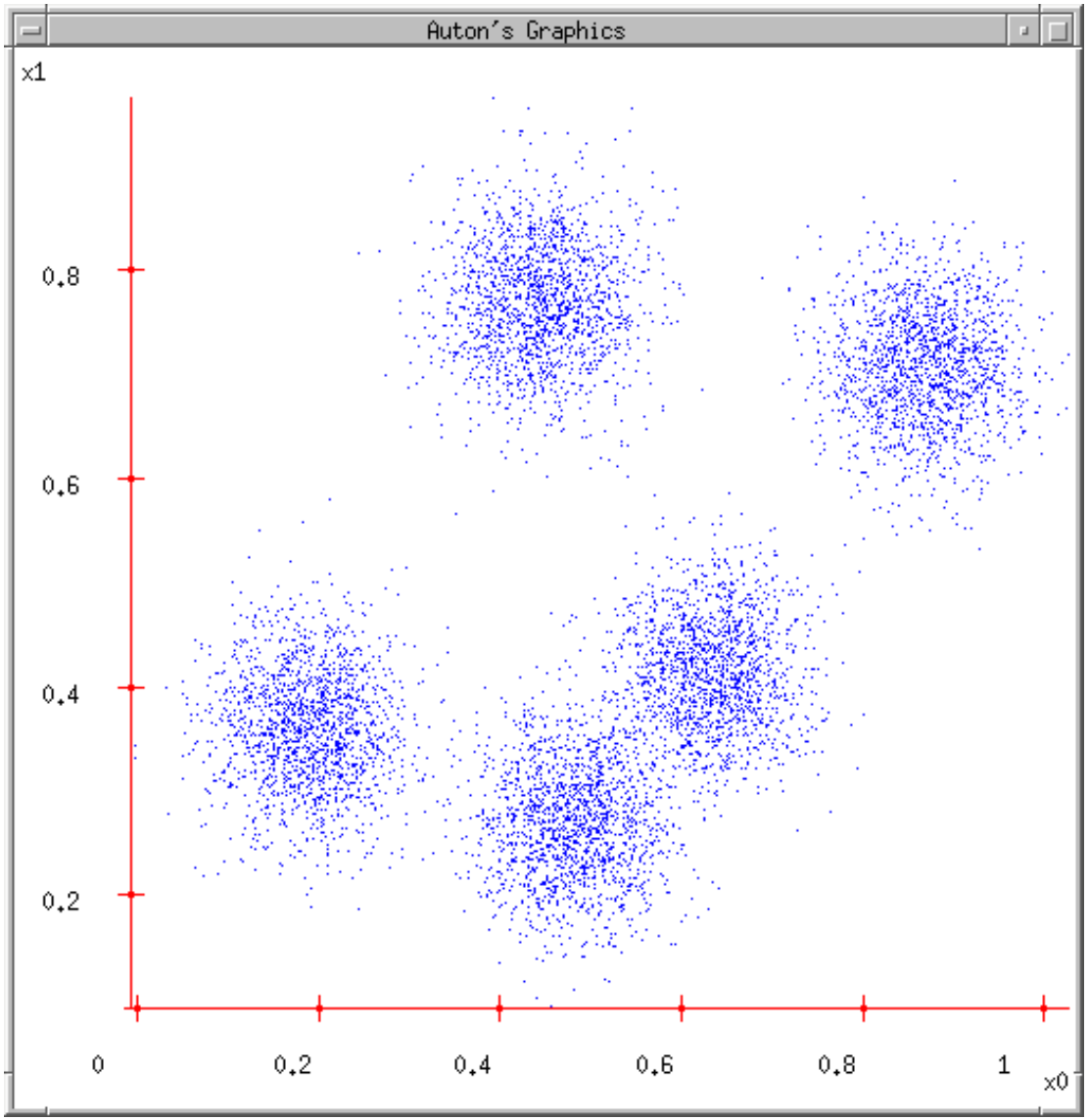
$$\mathbf{G}^2 = \mathbf{G}^1$$

- Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore.
- Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed..

We get the final grouping as the results as:

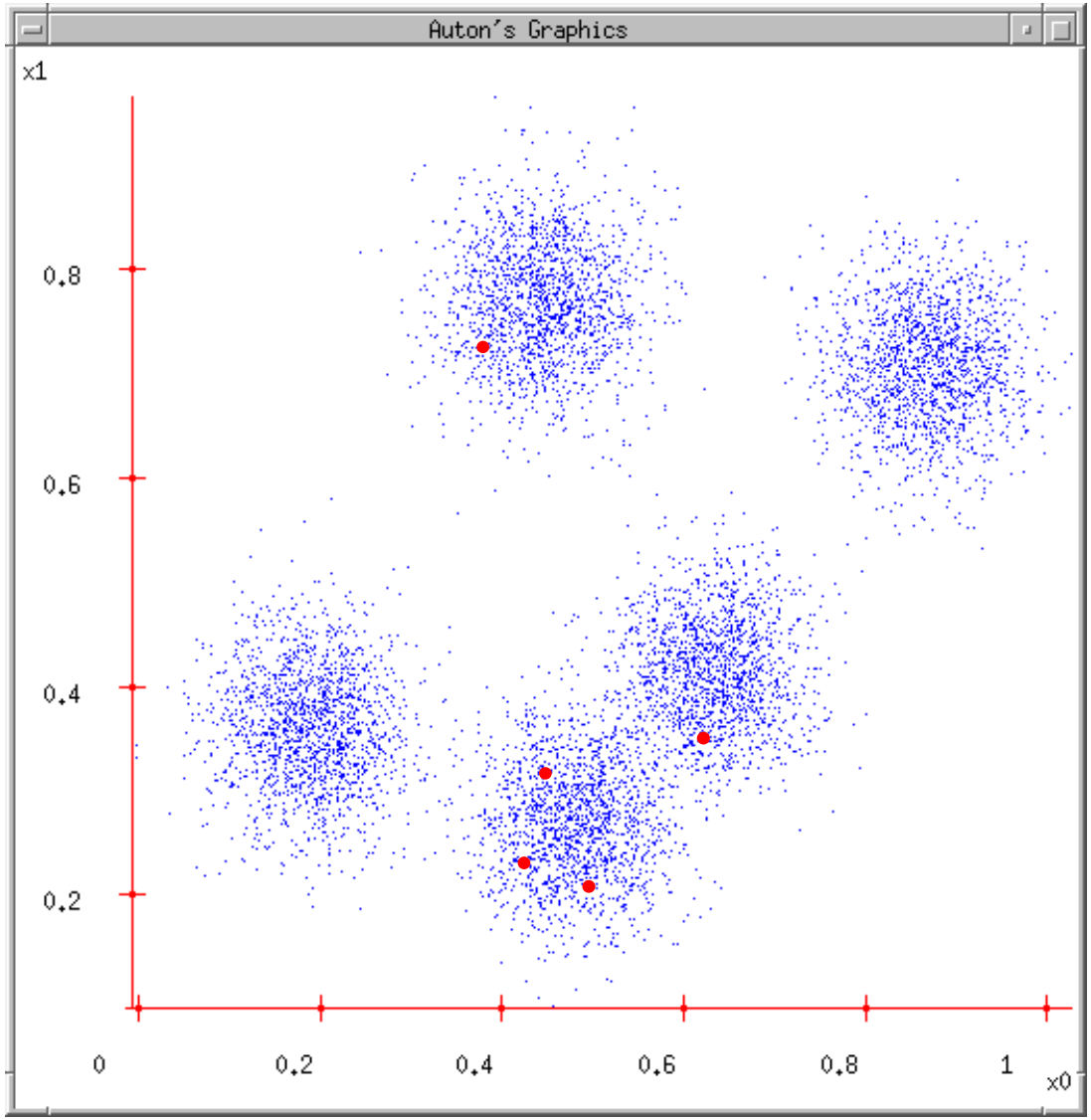
Object	**Group (result)**
A	**1**
B	**1**
C	**2**
D	**2**

K-means Demo



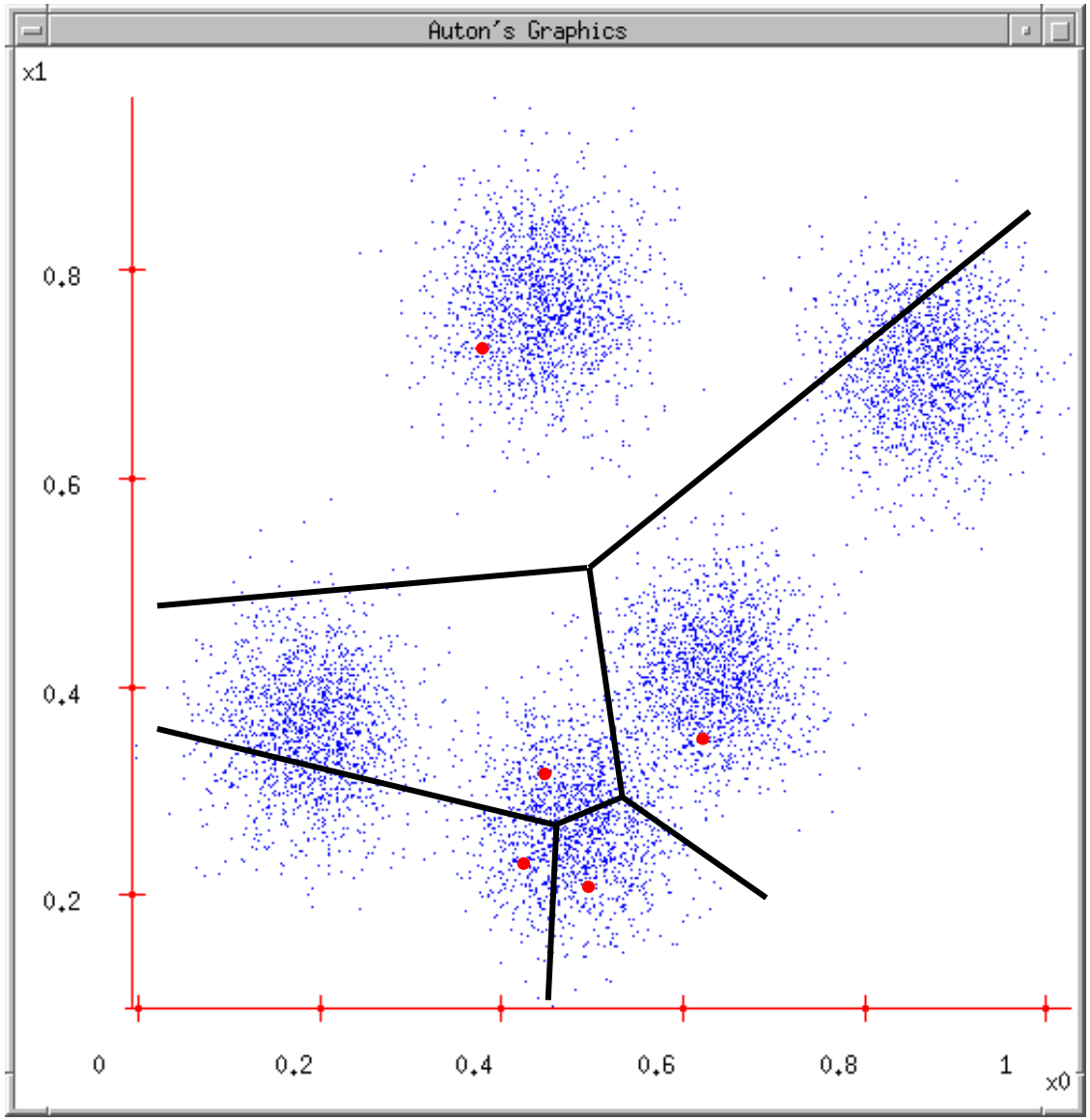
1. User set up the number of clusters they'd like. *(e.g. k=5)*

K-means Demo



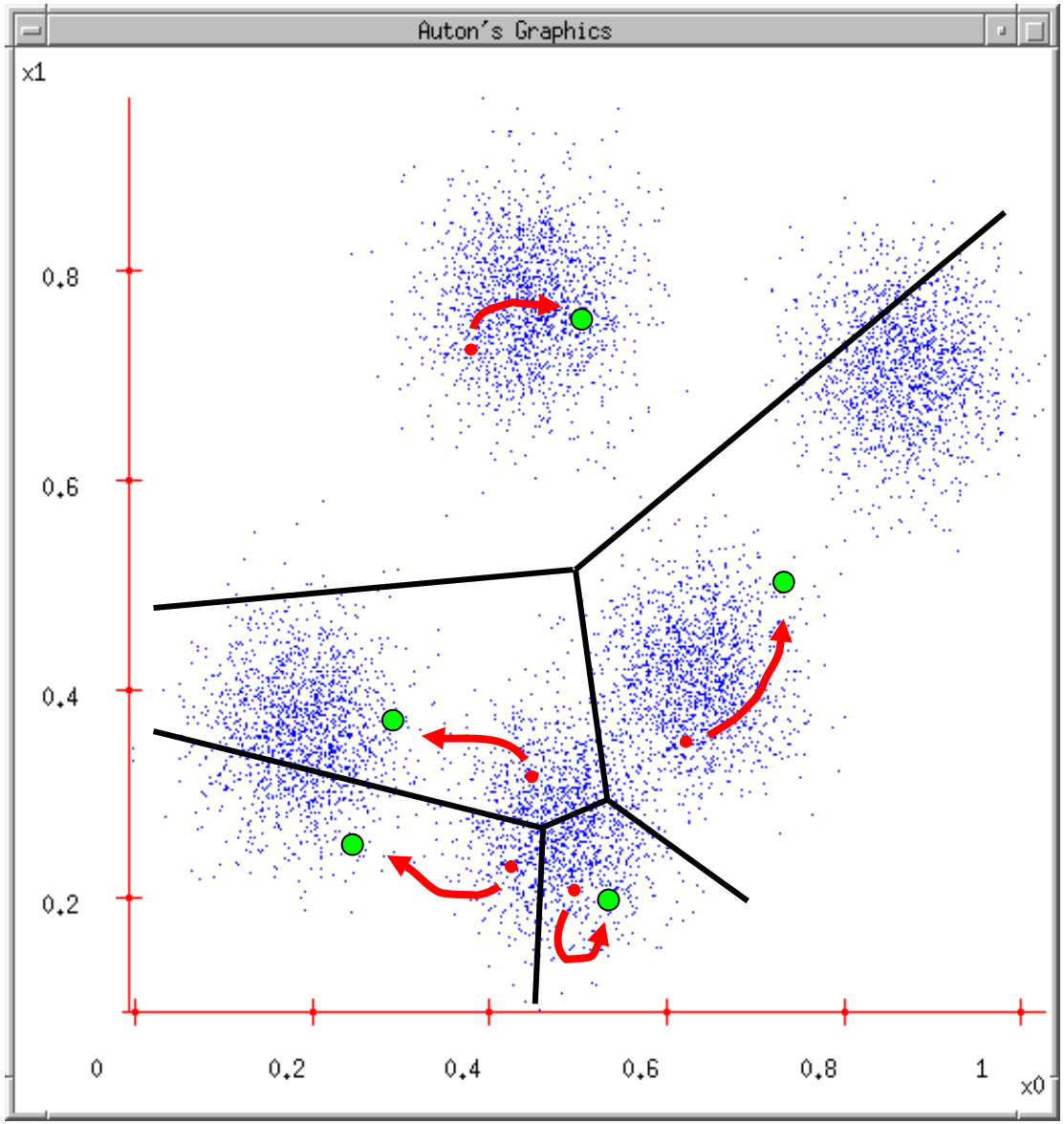
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess K cluster Center locations

K-means Demo



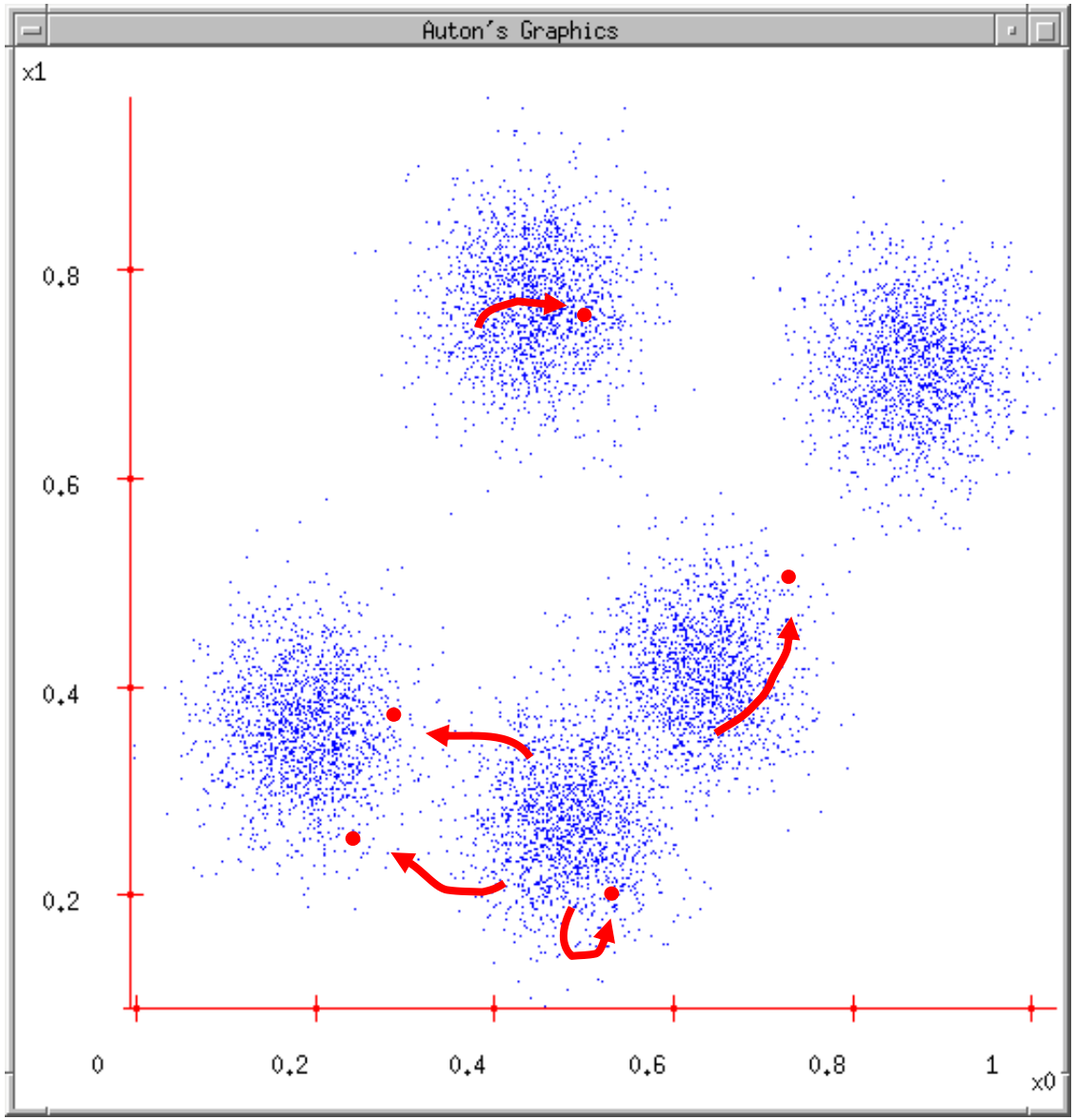
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess *K* cluster Center locations
3. Each data point finds out which Center it's closest to. (Thus each Center "owns" a set of data points)

K-means Demo



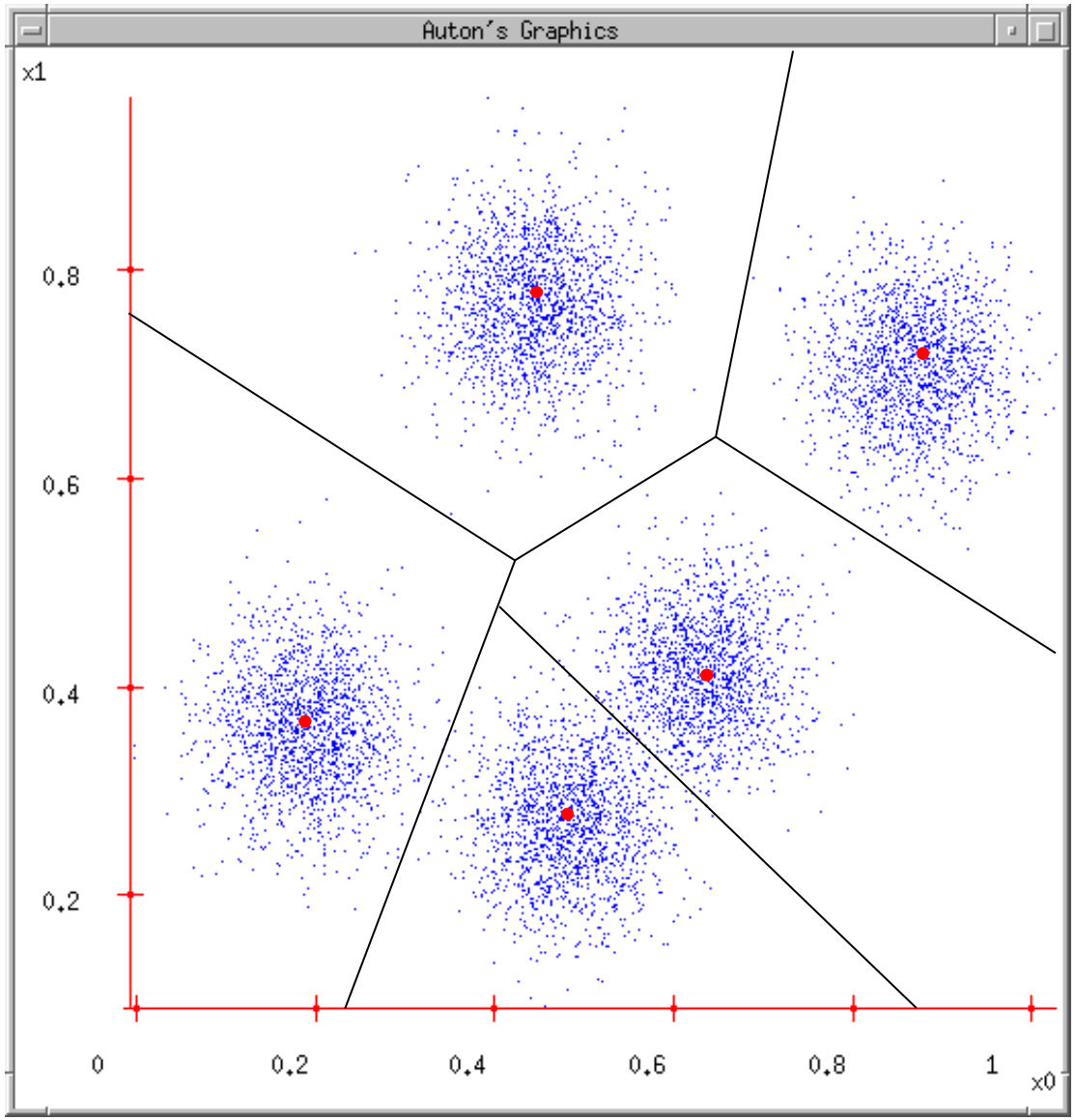
1. User set up the number of clusters they'd like. *(e.g. K=5)*
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each Center "owns" a set of data points)
4. Each centre finds the centroid of the points it owns

K-means Demo



1. User set up the number of clusters they'd like. (e.g. *K=5*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there

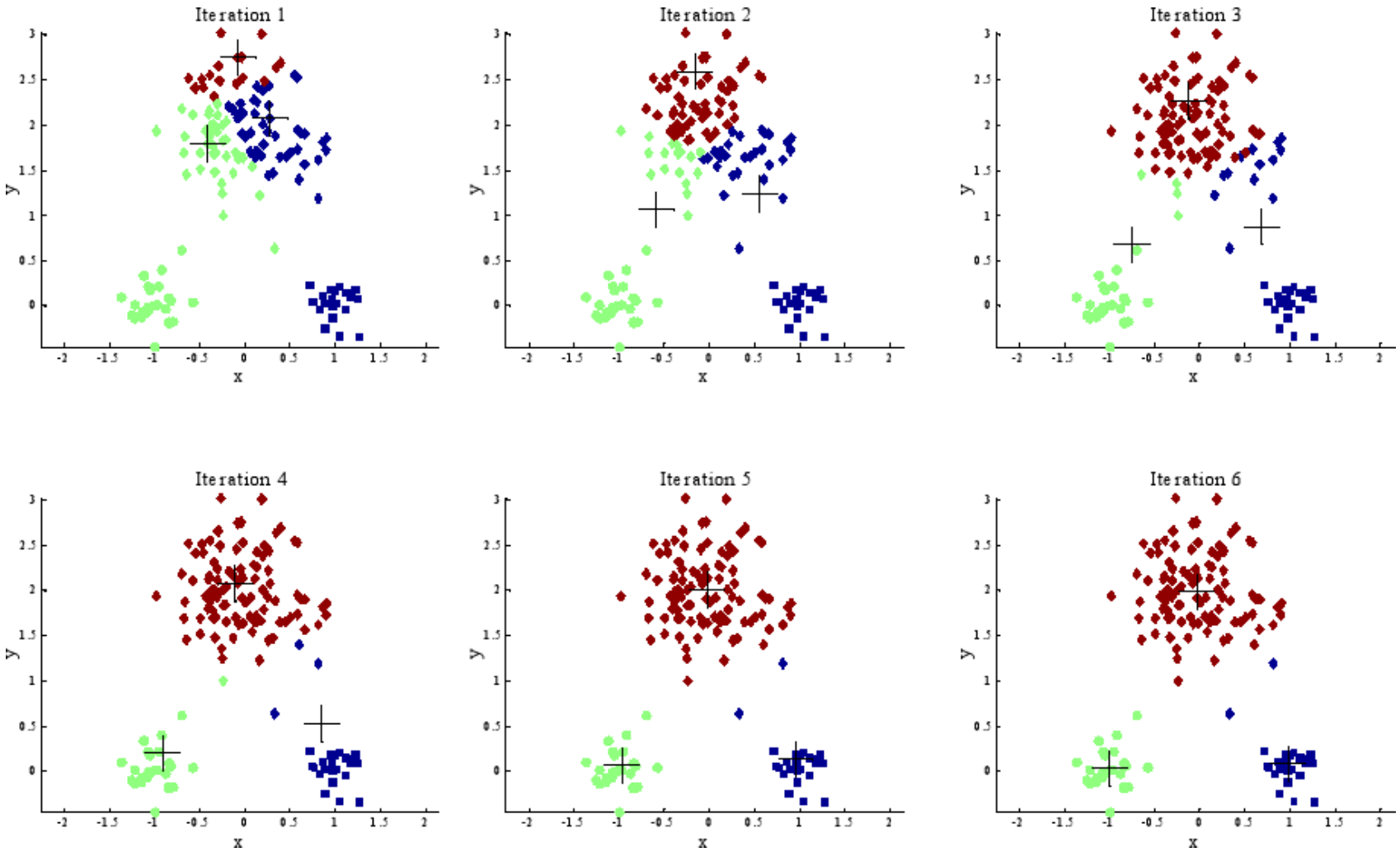
K-means Demo



1. User set up the number of clusters they'd like. (e.g. *K=5*)
2. Randomly guess *K* cluster centre locations
3. Each data point finds out which centre it's closest to. (Thus each centre "owns" a set of data points)
4. Each centre finds the centroid of the points it owns
5. …and jumps there
6. …Repeat until terminated!

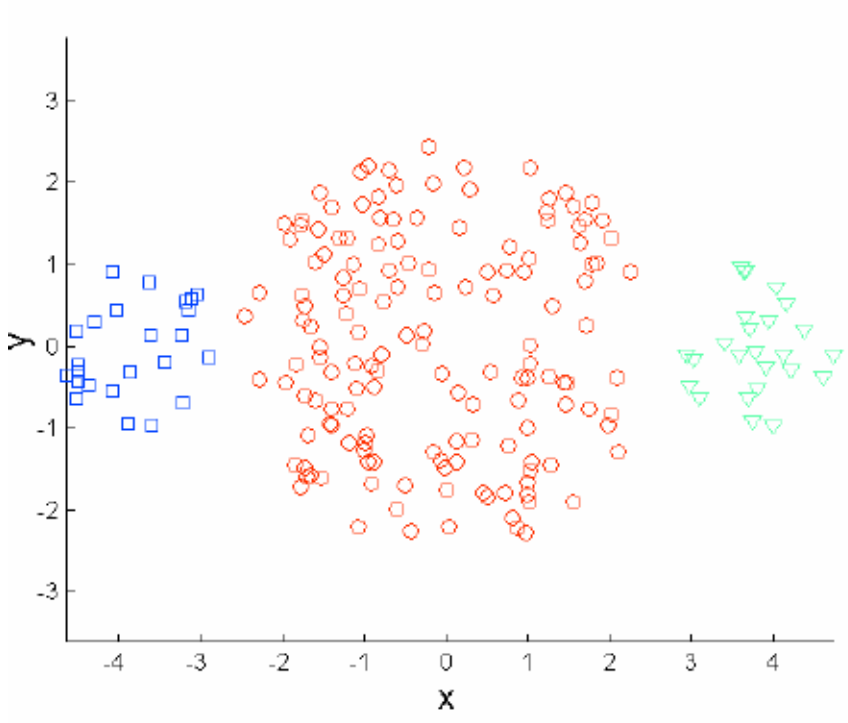
K-Means limits

Importance of choosing initial centroids

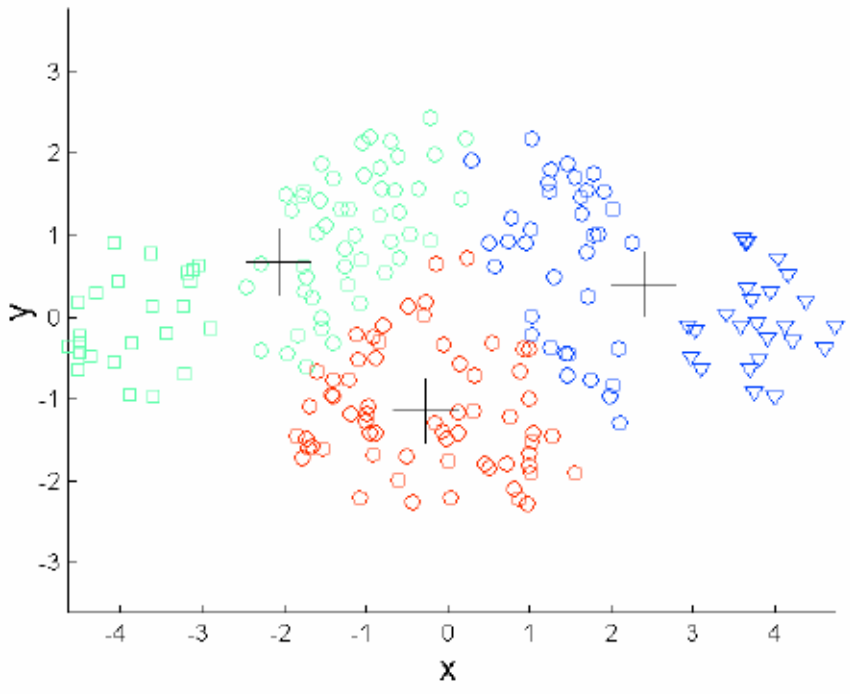


K-Means limits

Differing sizes



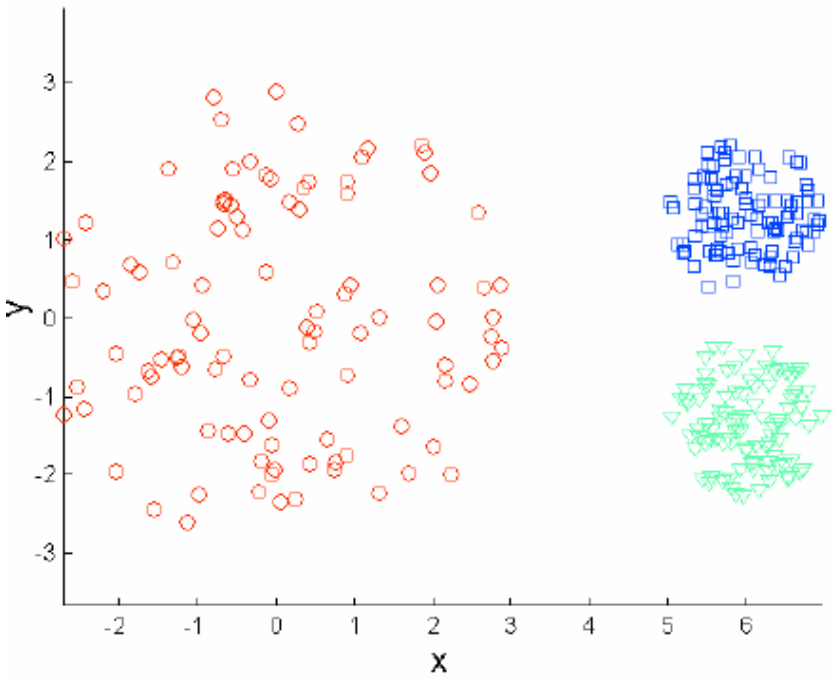
Original Points



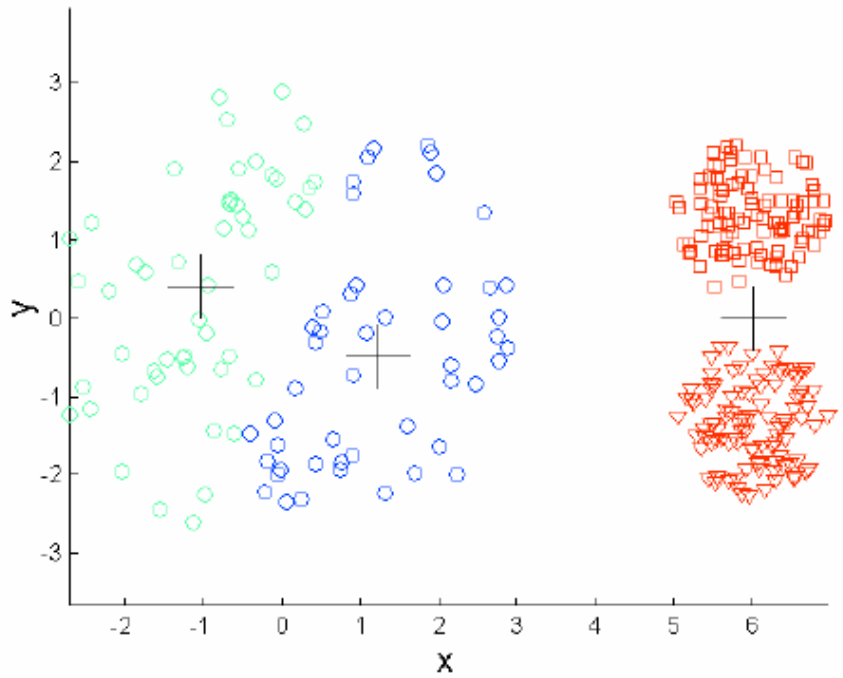
K-means Clusters

K-Means limits

Differing density



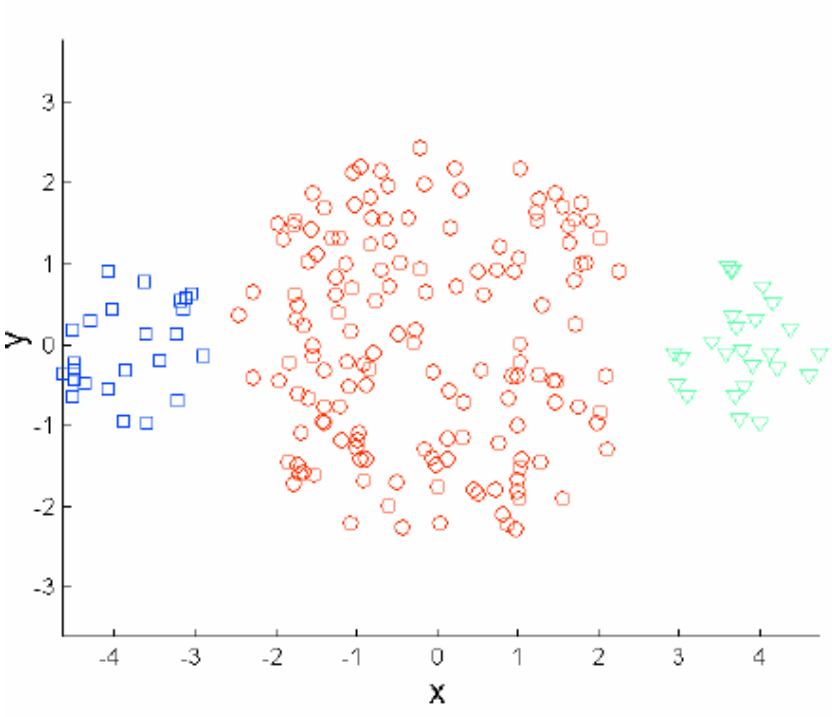
Original Points



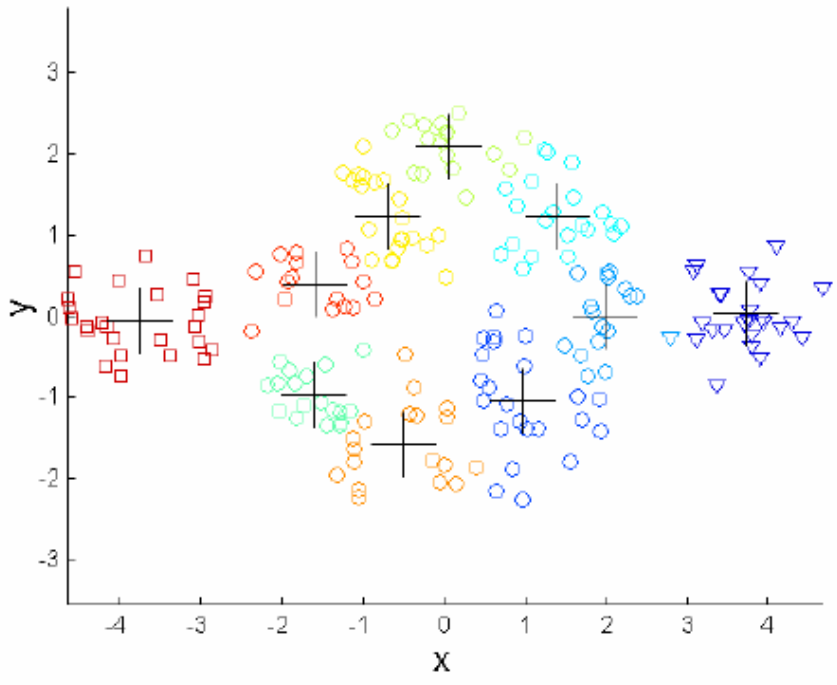
K-means Clusters

K-Means: higher K

What if we tried to increase K to solve K-Means problems?



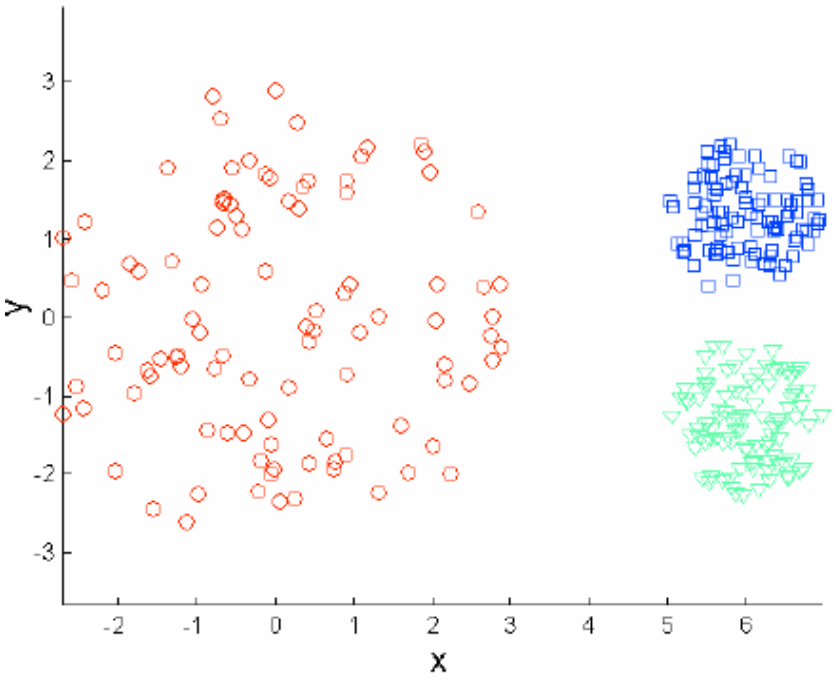
Original Points



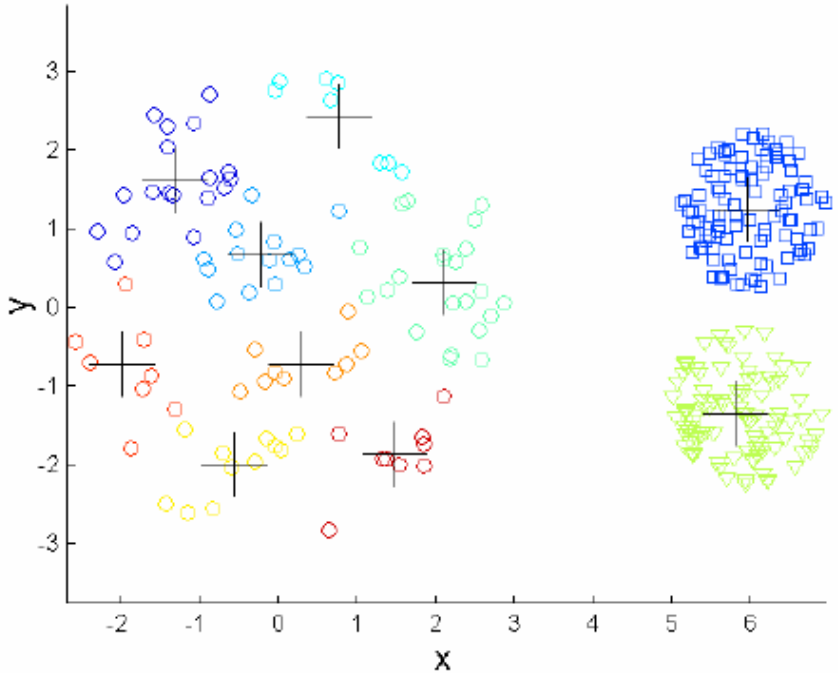
K-means Clusters

K-Means: higher K

What if we tried to increase K to solve K-Means problems?



Original Points

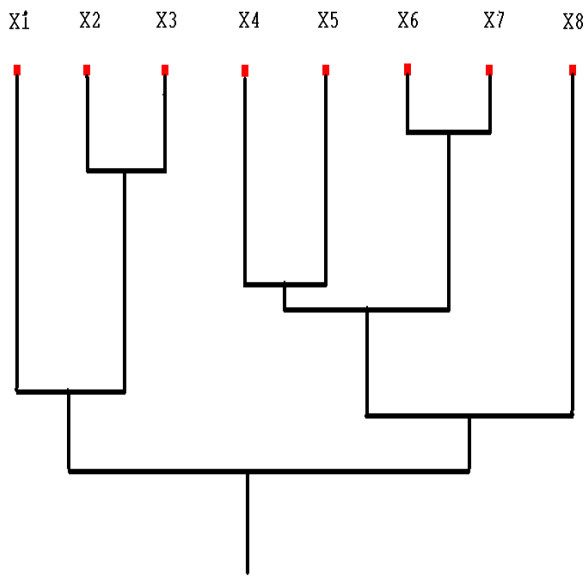


K-means Clusters

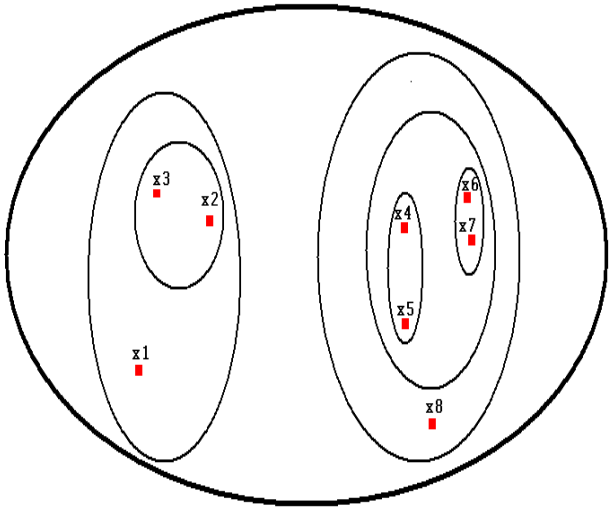
K-Means: Summary

- Advantages:
  - Simple, understandable
  - Relatively efficient: $O(tkn)$, where n is #objects, k is #clusters, and t is #iterations ($k, t \ll n$)
  - Often terminates at a local optimum
- Disadvantages:
  - Works only when mean is defined (what about categorical data?)
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data (too sensible to outliers)
  - Not suitable to discover clusters with non-convex shapes
  - Results depend on the metric used to measure distances and on the value of k
- Suggestions
  - Choose a way to initialize means (i.e. randomly choose k samples)
  - Start with *distant* means, run many times with different starting points
  - Use another algorithm ;-)

Hierarchical Clustering



Dendrogram



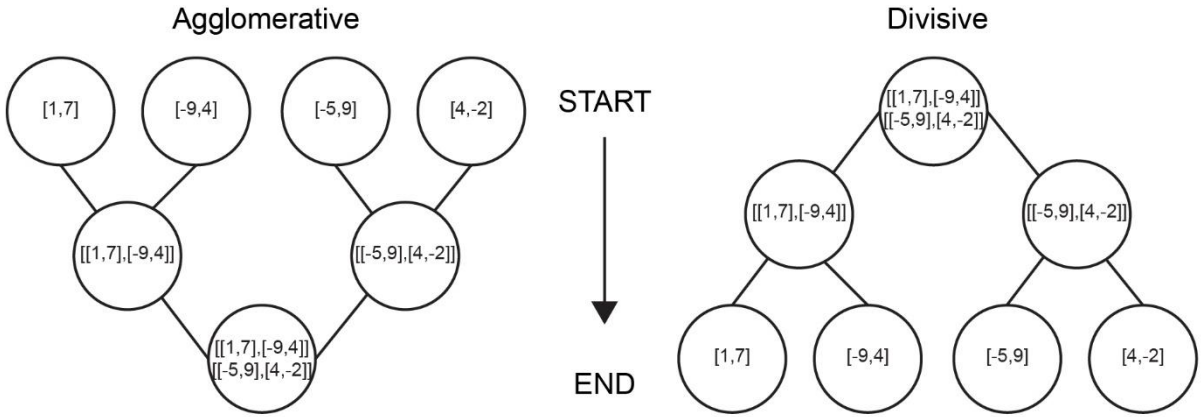
Venn Diagram of Clustered Data

Hierarchical Clustering (Cont.)

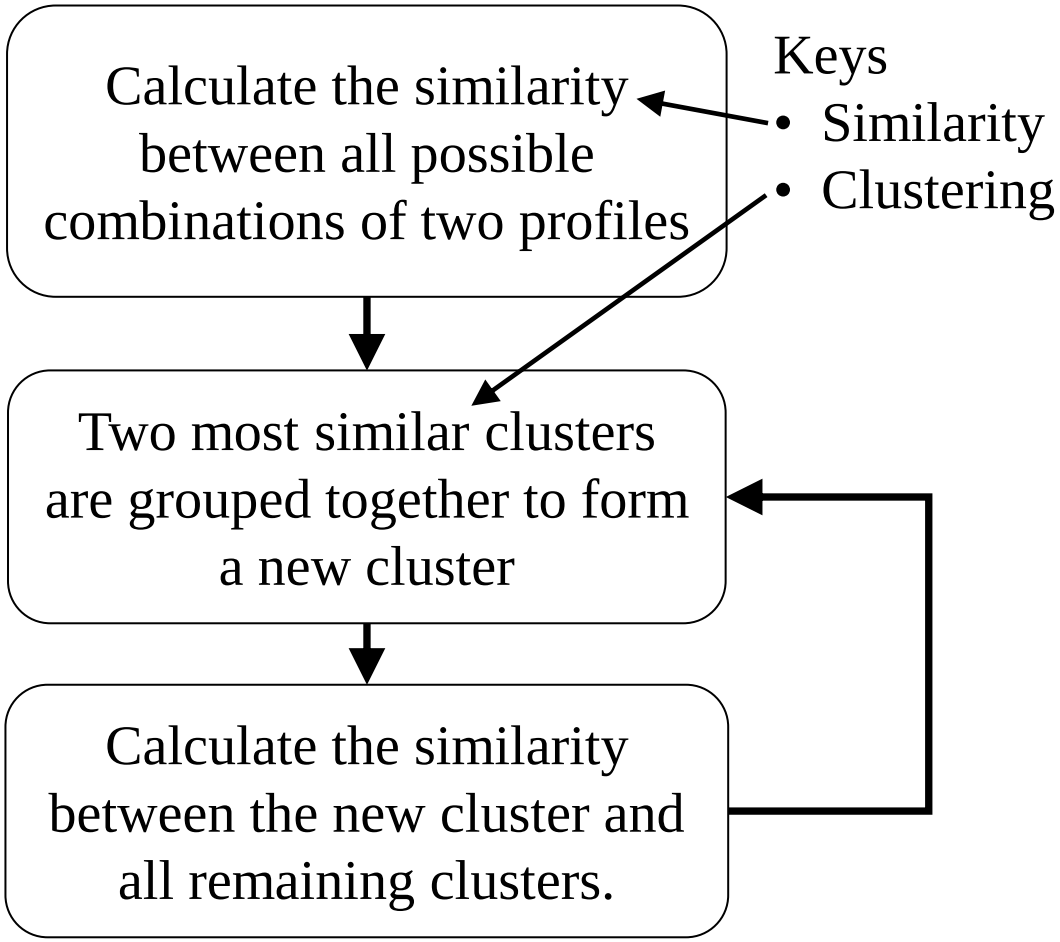
Multilevel clustering:
level 1 has *n* clusters → level *n* has one cluster.

Agglomerative HC:
starts with singleton and merge clusters.

Divisive HC:
starts with one sample and split clusters.

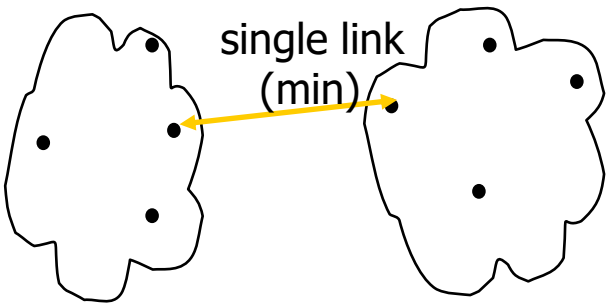


Hierarchical Clustering



Cluster Distance Measures

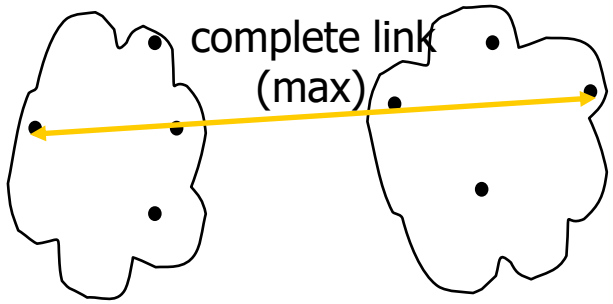
Single link: smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$



Cluster Distance Measures

Complete link: largest distance between an element in one cluster and an element in the other, i.e.,

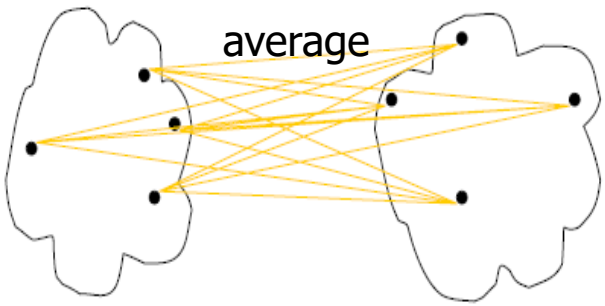
$d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$



Cluster Distance Measures

Average: avg distance between elements in one cluster and elements in the other, i.e.,

$$d(C_i, C_j) = avg\{d(x_{ip}, x_{jq})\}$$



d(C, C)=0

Cluster Distance Measures

Example: Given a data set of five objects characterized by a single continuous feature, assume that there are two clusters: C1: {a, b} and C2: {c, d, e}.

1. Calculate the distance matrix.
2. 2. Calculate three cluster distances between C1 and C2.

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

	a	b	c	d	e
Feature	1	2	4	5	6

Single link

$$\mathrm{dist}(C_1, C_2)$$
$$= \min\{\, d(\mathrm{a,c}), d(\mathrm{a,d}), d(\mathrm{a,e}), d(\mathrm{b,c}), d(\mathrm{b,d}), d(\mathrm{b,e})\}$$
$$= \min\{3, 4, 5, 2, 3, 4\} = 2$$

Cluster Distance Measures

Example: Given a data set of five objects characterized by a single continuous feature, assume that there are two clusters: C1: {a, b} and C2: {c, d, e}.

1. Calculate the distance matrix.
2. 2. Calculate three cluster distances between C1 and C2.

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

	a	b	c	d	e
Feature	1	2	4	5	6

Complete link

$$\text{dist}(C_1, C_2)$$
$$= \max\{\, d(\text{a,c}), d(\text{a,d}), d(\text{a,e}), d(\text{b,c}), d(\text{b,d}), d(\text{b,e})\}$$
$$= \max\{3, 4, 5, 2, 3, 4\} = 5$$

Cluster Distance Measures

Example: Given a data set of five objects characterised by a single continuous feature, assume that there are two clusters: C1: {a, b} and C2: {c, d, e}.

1. Calculate the distance matrix.
2. 2. Calculate three cluster distances between C1 and C2.

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

	a	b	c	d	e
Feature	1	2	4	5	6

Average

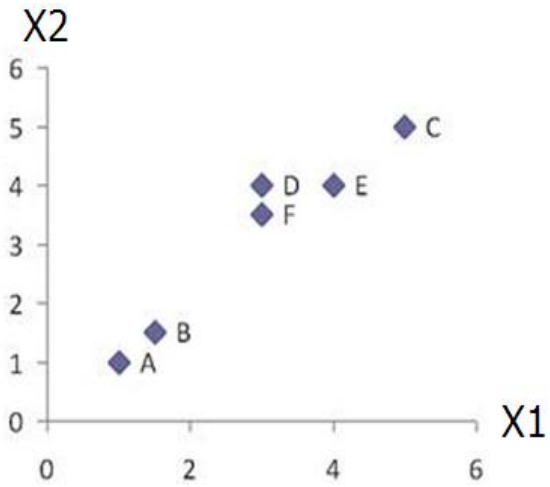
$$\mathrm{dist}(C_1, C_2) = \frac{d(\mathrm{a,c}) + d(\mathrm{a,d}) + d(\mathrm{a,e}) + d(\mathrm{b,c}) + d(\mathrm{b,d}) + d(\mathrm{b,e})}{6}$$

$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

Example

Problem: clustering analysis with agglomerative algorithm

Data	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



$$d_{AB} = ((1-1.5)^2 + (1-1.5)^2)^{\frac{1}{2}} = \sqrt{\frac{1}{2}} = 0.707$$

$$d_{DF} = ((3-3)^2 + (4-3.5)^2)^{\frac{1}{2}} = 0.5$$

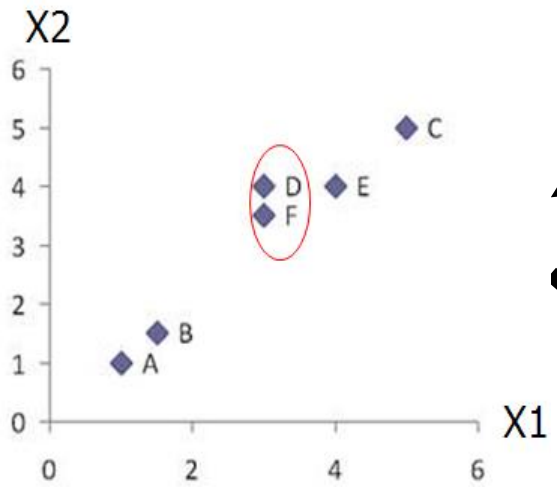
Euclidean distance

Dist	A	B	C	D	E	F
A	0.0	0.71	5.66	3.61	4.24	3.2
B		0.0	4.95	2.92	3.54	2.50
C			0.0	2.24	1.41	2.50
D				0.0	1.00	0.50
E					0.0	1.12
F						0.0

distance matrix

Example

Merge two closest clusters (iteration 1)



Dist	A	B	C	D	E	F
A	0.0	0.71	5.66	3.61	4.24	3.2
B		0.0	4.95	2.92	3.54	2.50
C			0.0	2.24	1.41	2.50
D				0.0	1.00	0.50
E					0.0	1.12
F						0.0

Dist	A	B	C	D,F	E
A	0.0	0.71	5.66	?	4.24
B		0.0	4.95	?	3.54
C			0.0	?	1.41
D,F				0.0	?
E					0.0

Example

Update distance matrix (iteration 1)

Dist	A	B	C	D	E	F
A	0.0	0.71	5.66	3.61	4.24	3.2
B		0.0	4.95	2.92	3.54	2.50
C			0.0	2.24	1.41	2.50
D				0.0	1.00	0.50
E					0.0	1.12
F						0.0

$$d_{(D,F)\to A} = \min(d_{DA}, d_{FA}) = \min(3.61, 3.20) = 3.20$$

$$d_{(D,F)\to B} = \min(d_{DB}, d_{FB}) = \min(2.92, 2.50) = 2.50$$

$$d_{(D,F)\to C} = \min(d_{DC}, d_{FC}) = \min(2.24, 2.50) = 2.24$$

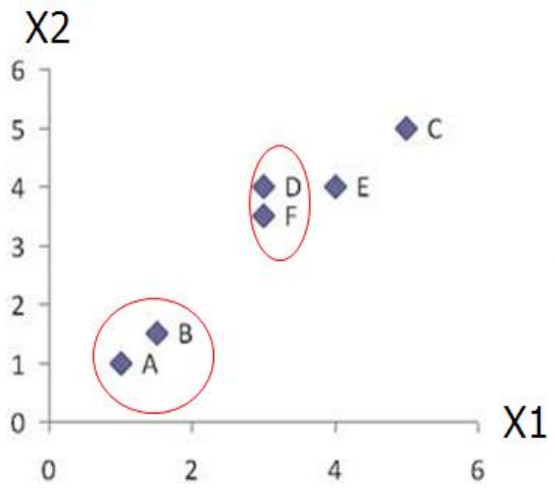
$$d_{(D,F)\to E} = \min(d_{DE}, d_{FE}) = \min(1.00, 1.12) = 1.00$$

Dis	A	B	C	D,F	E
A	0.0	0.71	5.66	?	4.24
B		0.0	4.95	?	3.54
C			0.0	?	1.41
D,F				0.0	?
E					0.0

Dist	A	B	C	D,F	E
A	0.0	0.71	5.66	3.20	4.24
B		0.0	4.95	2.50	3.54
C			0.0	2.24	1.41
D,F				0.0	1.00
E					0.0

Example

Merge two closest clusters (iteration 2)



Dist	A	B	C	D,F	E
A	0.0	**0.71**	5.66	3.20	4.24
B		0.0	4.95	2.50	3.54
C			0.0	2.24	1.41
D,F				0.0	1.00
E					0.0

Dist	A,B	C	D,F	E
A,B	0.0	?	?	?
C		0.0	2.24	1.41
D,F			0.0	1.00
E				0.0

Example

Update distance matrix (iteration 2)

Dist	A	B	C	D,F	E
A	0.0	0.71	5.66	3.20	4.24
B		0.0	4.95	2.50	3.54
C			0.0	2.24	1.41
D,F				0.0	1.00
E					0.0

$$d_{(A,B)\to C} = \min(d_{AC}, d_{BC}) = \min(5.66, 4.95) = 4.95$$

$$d_{(A,B)\to(D,F)} = \min(d_{AD}, d_{AF}, d_{BD}, d_{BF}) = \min(3.61, 2.92, 3.20, 2.50) = 2.50$$

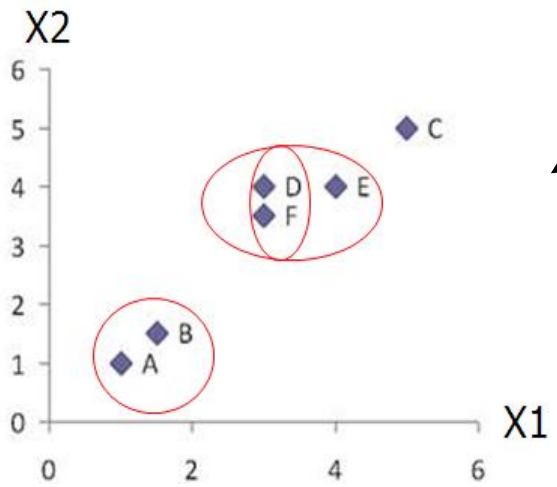
$$d_{(A,B)\to E} = \min(d_{AE}, d_{BE}) = \min(4.24, 3.54) = 3.54$$

Dist	A,B	C	D,F	E
A,B	0.0	?	?	?
C		0.0	2.24	1.41
D,F			0.0	1.00
E				0.0

Dist	A,B	C	D,F	E
A,B	0.0	4.95	2.50	3.54
C		0.0	2.24	1.41
D,F			0.0	1.00
E				0.0

Example

Merge two closest clusters/update distance matrix (iteration 3)

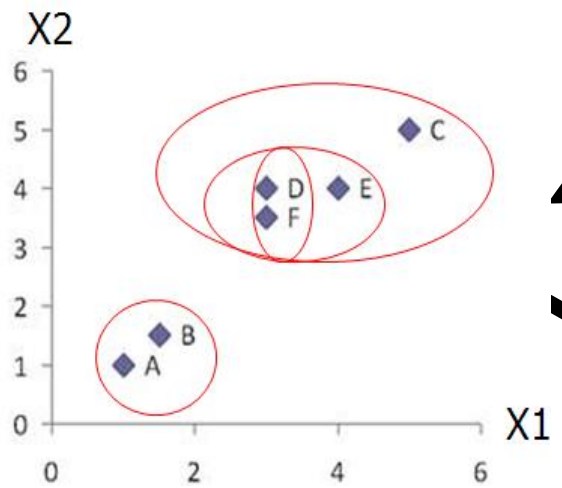


Dist	A,B	C	D,F	E
A,B	0.0	4.95	2.50	3.54
C		0.0	2.24	1.41
D,F			0.0	1.00
E				0.0

Dist	A,B	C	(D,F),E
A,B	0.0	4.95	2.50
C		0.0	1.41
(D,F),E			0.0

Example

Merge two closest clusters/update distance matrix (iteration 4)



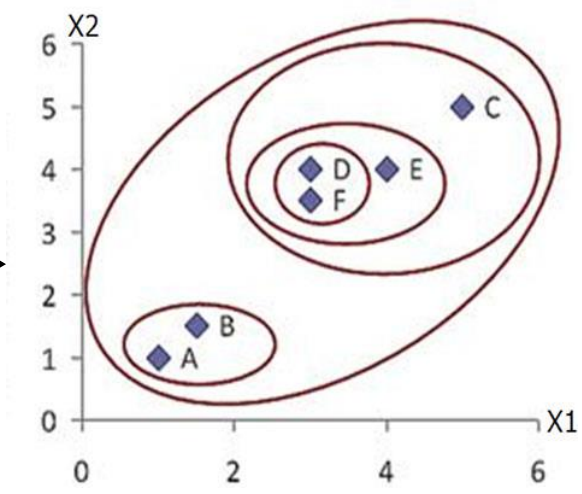
Dist	A,B	C	(D,F),E
A,B	0.0	4.95	2.50
C		0.0	1.41
(D,F),E			0.0

Dist	A,B	((D,F),E),C
A,B	0.0	2.50
((D,F),E),C		0.0

Example

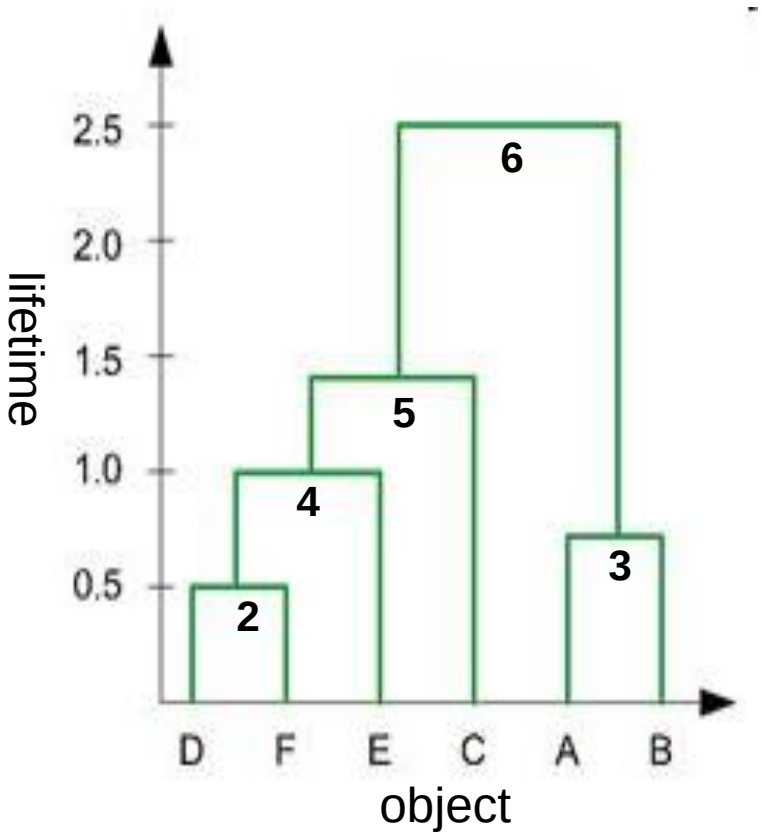
Final result (meeting termination condition)

Data	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Example

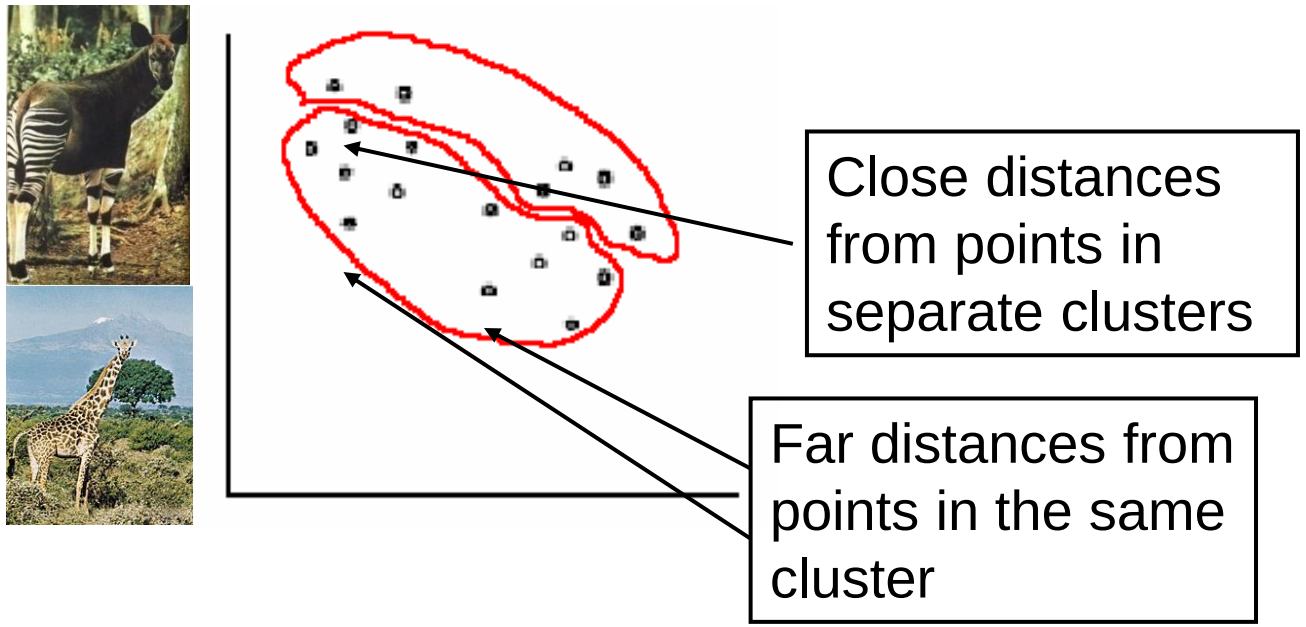
Dendrogram tree representation



1. In the beginning we have 6 clusters: A, B, C, D, E and F
2. We merge clusters D and F into cluster (D, F) at distance 0.50
3. We merge cluster A and cluster B into (A, B) at distance 0.71
4. We merge clusters E and (D, F) into ((D, F), E) at distance 1.00
5. We merge clusters ((D, F), E) and C into (((D, F), E), C) at distance 1.41
6. We merge clusters (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
7. The last cluster contain all the objects, thus conclude the computation

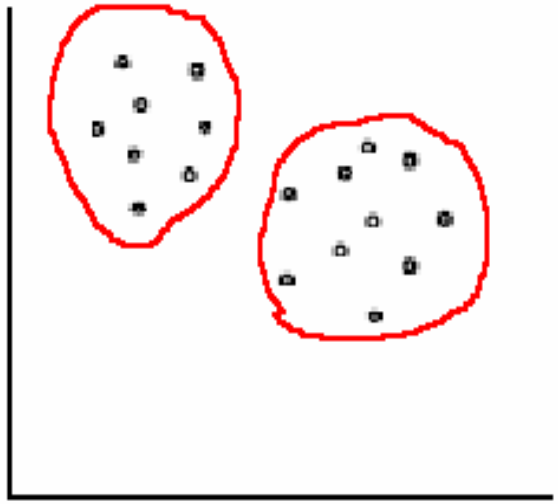
Bad Clustering

This clustering violates both
Homogeneity and Separation principles



Good Clustering

This clustering satisfies both
Homogeneity and Separation principles



- The end

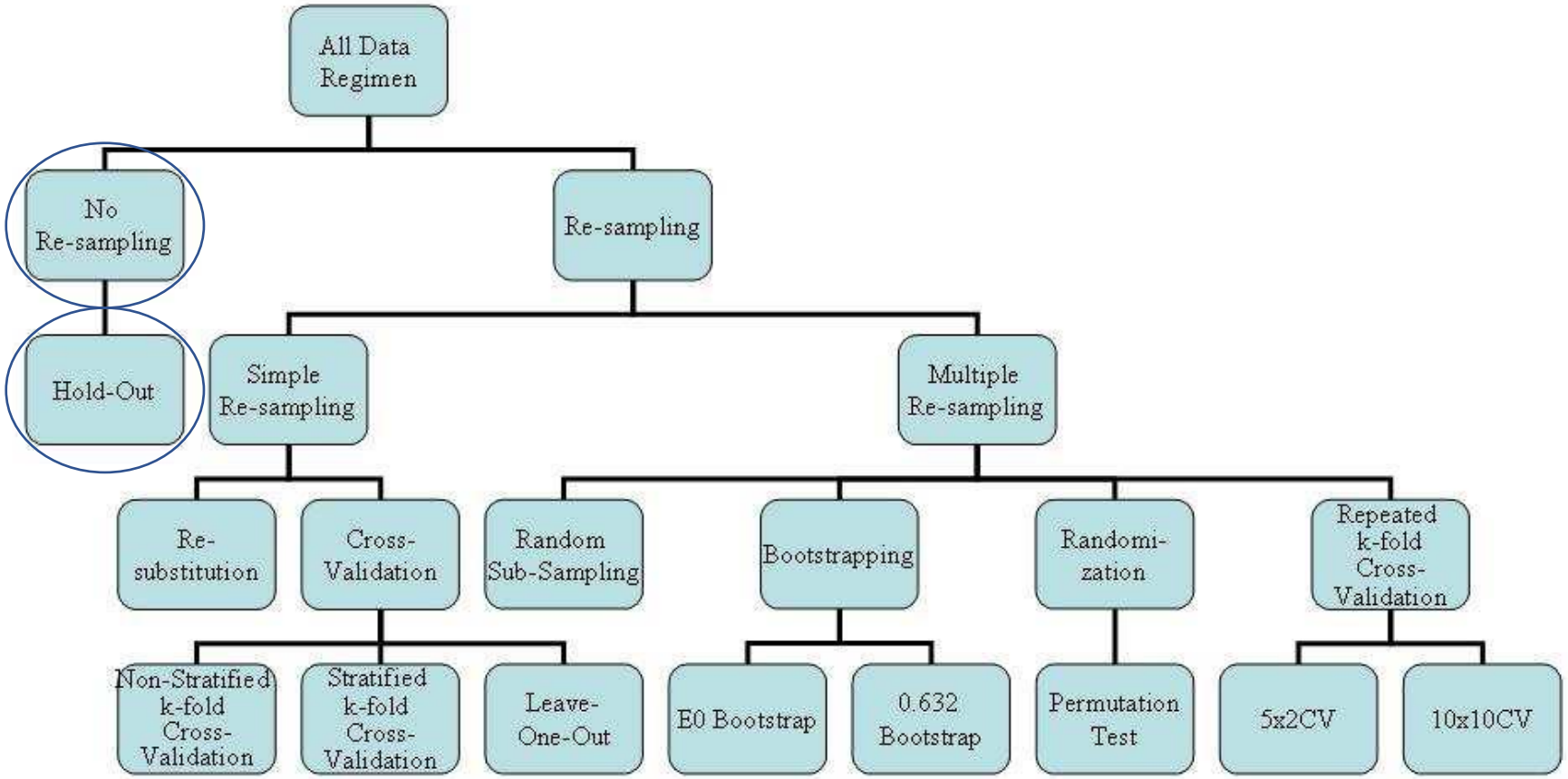
Performance Evaluation

Performance Evaluation

- Dataset Sampling and Re-arrangement
- Performance Metrics
- Performance Charts/Curves

Topic 1:
Dataset Sampling and Arrangement

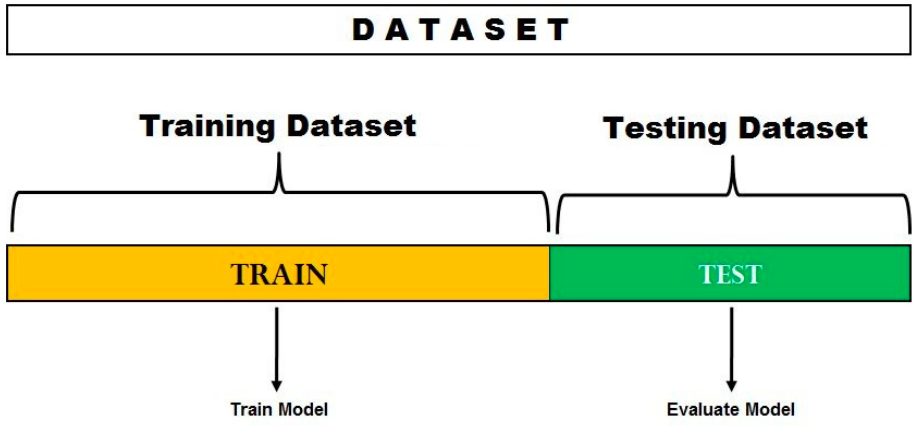
Overview of Re-Sampling Methods

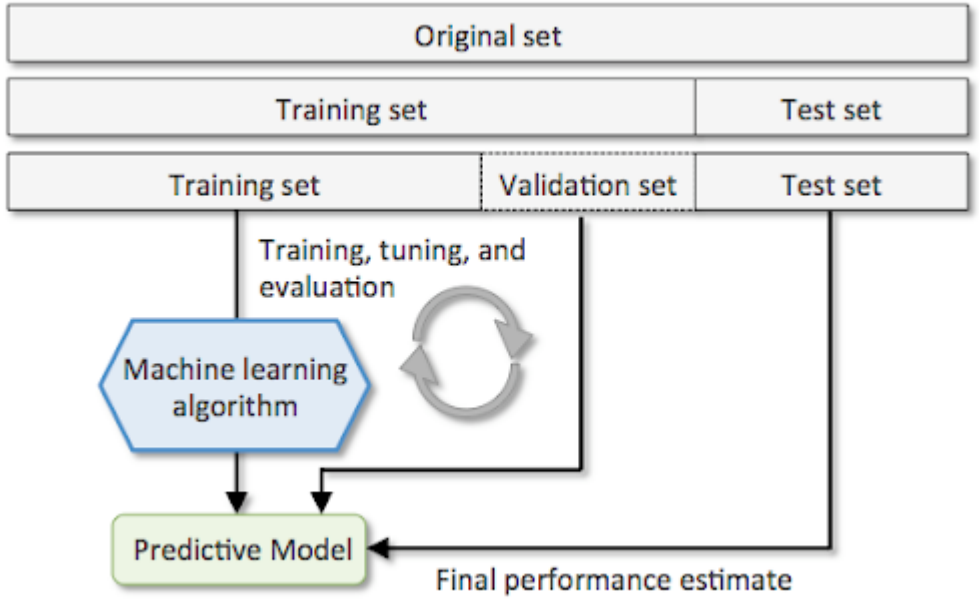


No-Sampling Method: Holdout Methods

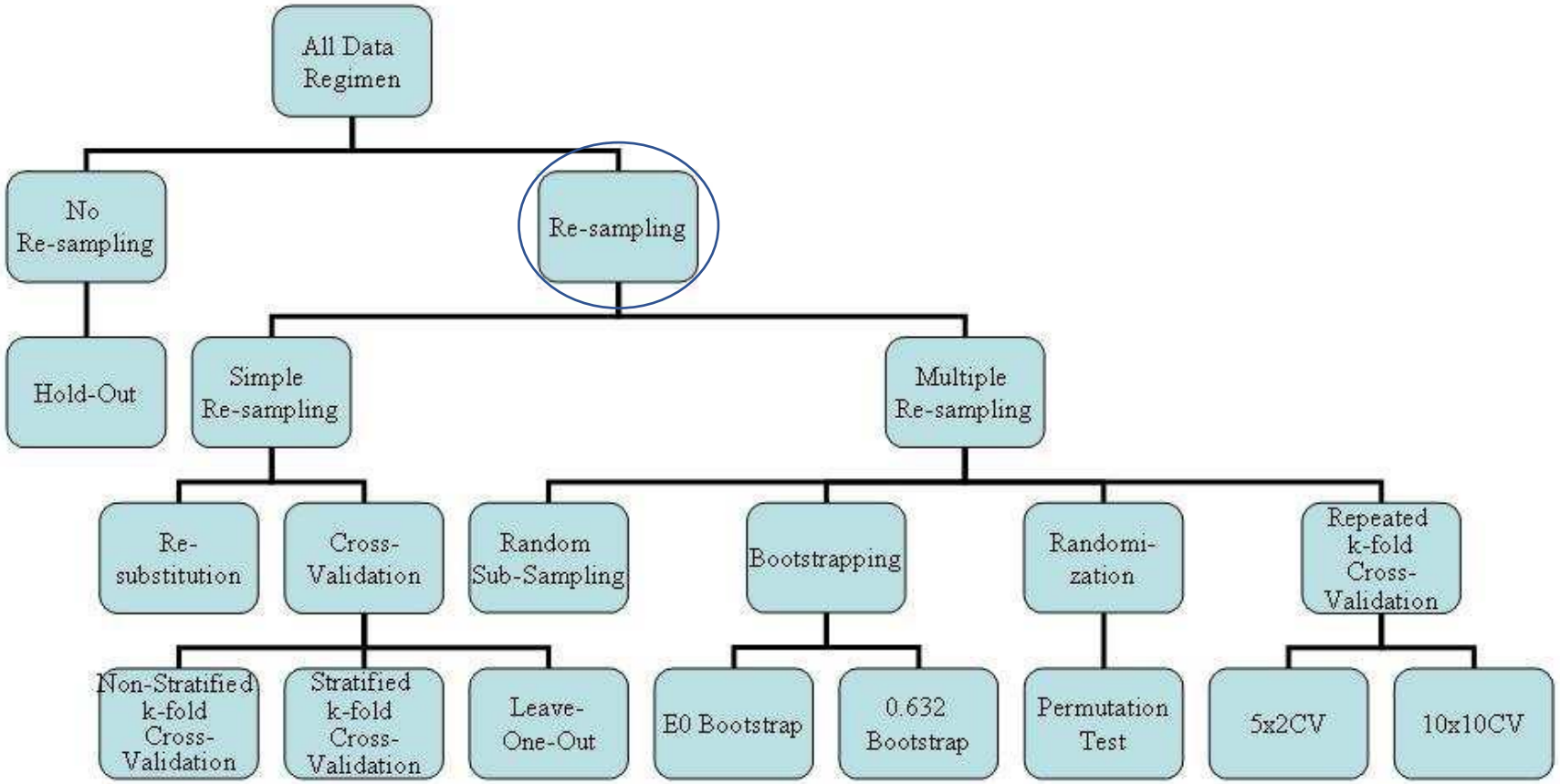
- **No-Sampling:** No explicit or external sampling techniques used. One of the examples is Holdout method.

- **Holdout Method** is the simplest sort of method to evaluate a classifier. In this method, the data set (a collection of data items or examples) is separated into two sets, called the **Training set and Test set**. Typically in 80/20, or 70/30 format.
- When data is sufficiently large, we introduce a validation set, where we typically tune parameters.





Overview of Re-Sampling Methods



What is the Purpose of Resampling?

- Ideally, we would have access to the entire population or a lot of representative data from it.
- This is usually not the case, and the limited data available has to be re-used in clever ways in order to be able to estimate the error of our classifiers as reliably as possible, i.e., to be re-used in clever ways in order to obtain sufficiently large numbers of samples.
- Resampling is divided into two categories:
  - *Simple re-sampling* (where each data point is used for testing only once)
  - *Multiple re-sampling:* (which allows the use of the same data point more than once for testing).

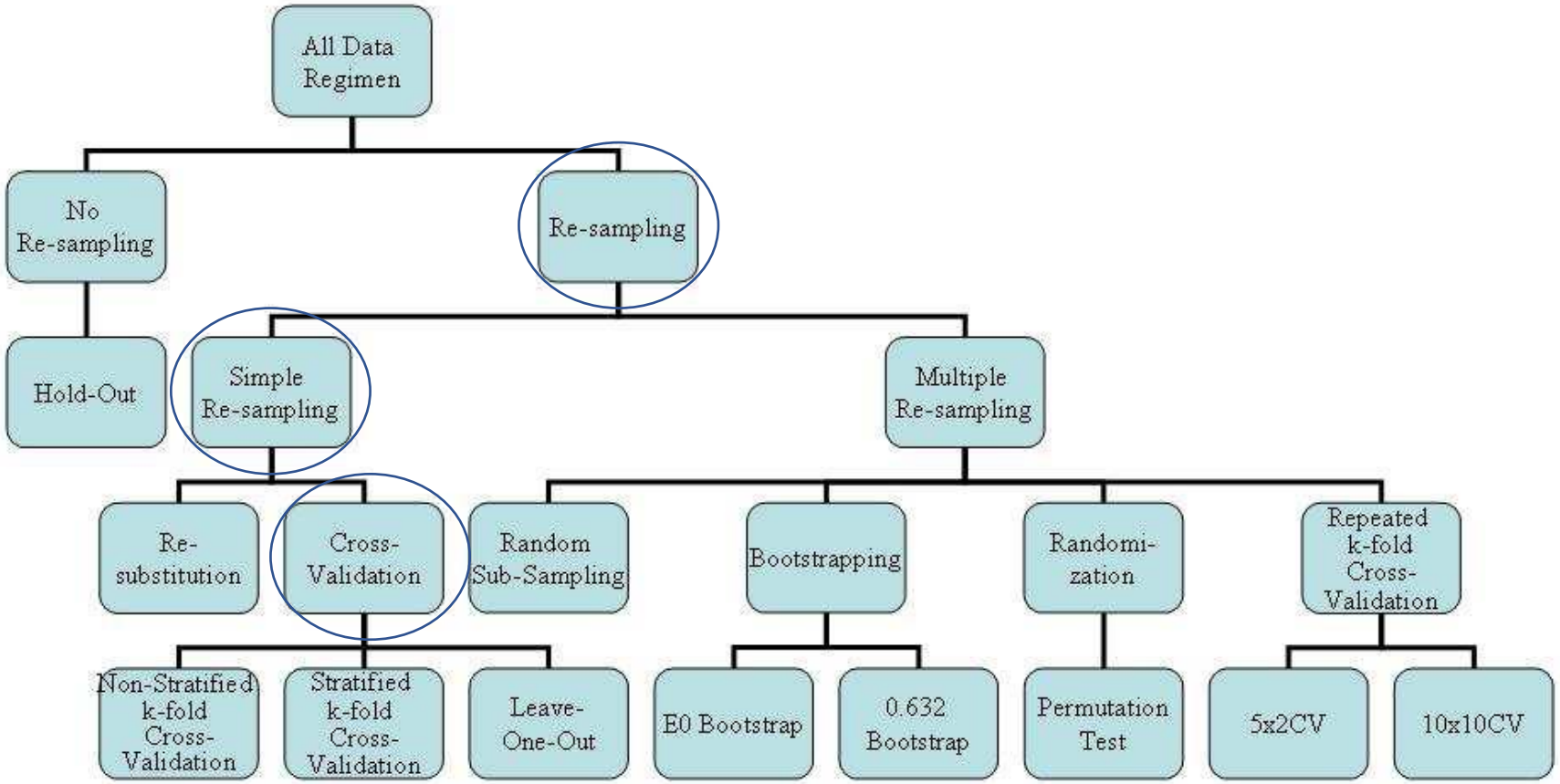
What are the Issues of Resampling?

- Re-sampling is usually followed by Statistical testing. Yet statistical testing relies on the fundamental assumption that the data used to obtain a sample statistics must be independent.
- However, if data is re-used, then this important independence assumption is broken and the result of the statistical test risks being invalid.
- In addition to discussing a few re-sampling approaches, we will underline the issues that may arise when applying them.

Resampling Approaches Discussed in this Tutorial

- Simple Resampling:
  - Cross-Validation and its variants
- Multiple Resampling:
  - The 0.632 Bootstrap
  - The Permutation Test
  - Repeated k-fold Cross-Validation
  - ….

Overview of Re-Sampling Methods

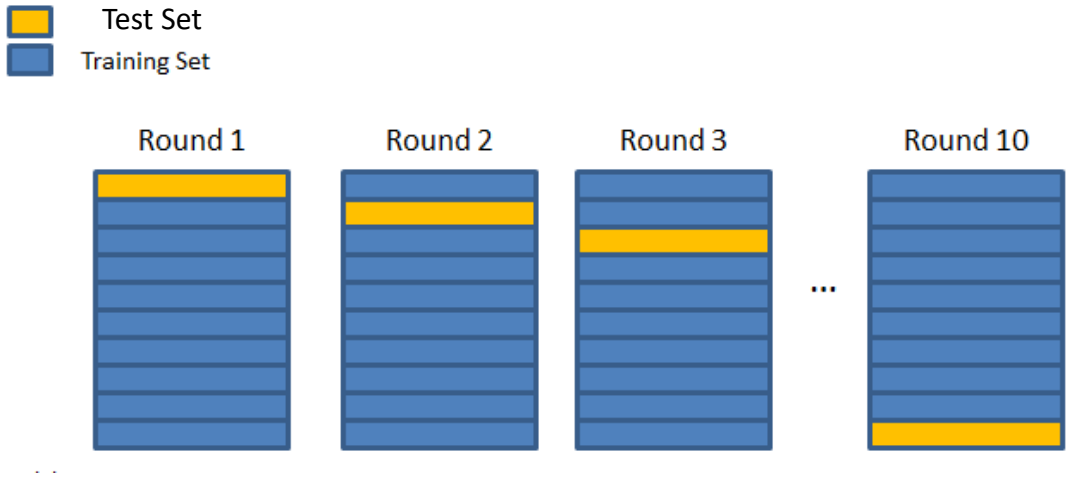


k-fold Cross-Validation

In Cross-Validation, the data set is divided into **k** folds and at each iteration, a different fold is reserved for testing and all the others, used for training the classifiers.

The procedure has a single parameter called **k** that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.
k=10 becoming 10-fold cross-validation.
If k=5 the dataset will be divided into 5 equal parts and the below process will run 5 times, each time with a different holdout set.

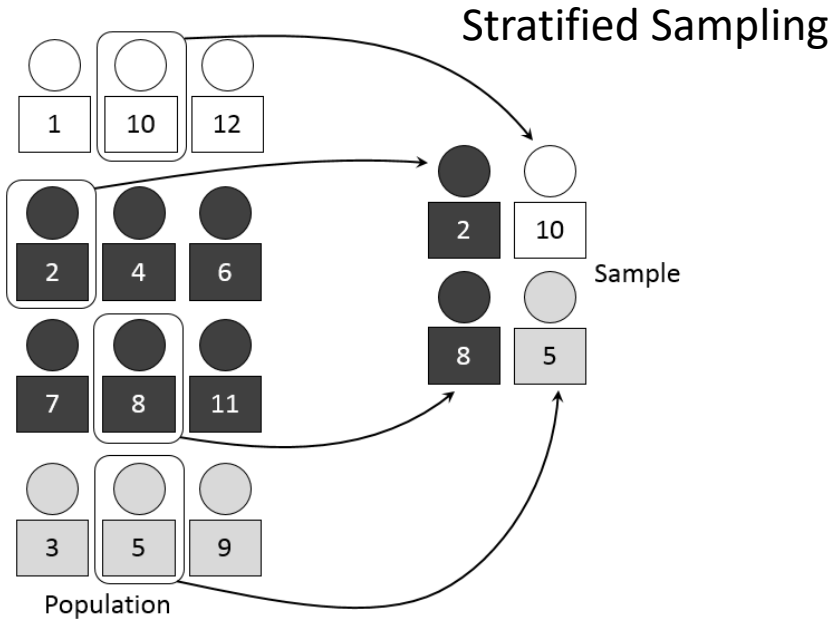
Procedure:
1. Take the group as a holdout or test data set
2. Take the remaining groups as a training data set
3. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model



Issues: k-fold Cross-Validation

- Using simple K-Fold (non-stratified CV) on a classification problem can be tricky.
- Since we are randomly shuffling the data and then dividing it into folds, chances are we may get highly imbalanced folds which may cause our training to be biased.
- For example, let us somehow get a fold that has majority belonging to one class(say positive) and only a few as negative class.
- This will certainly ruin our training and to avoid this we make stratified folds using stratification.

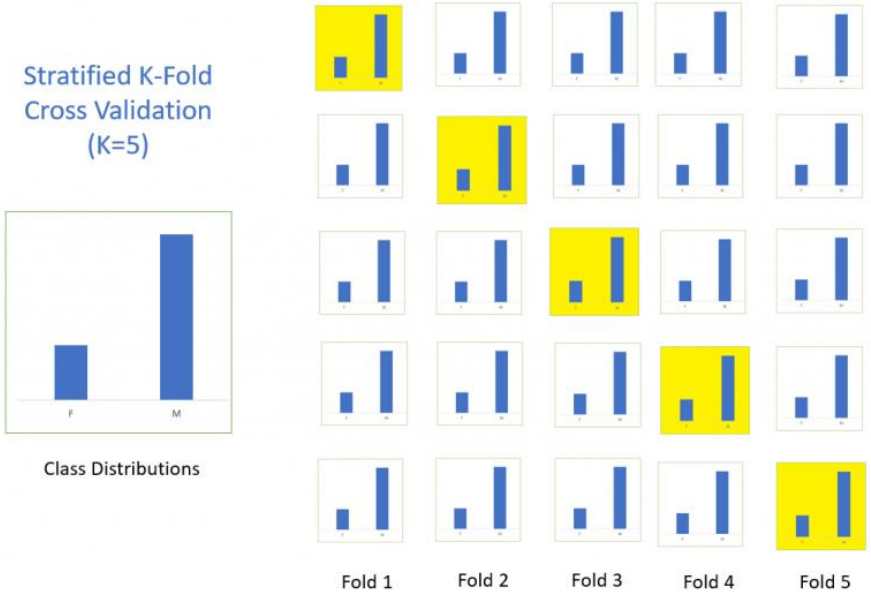
Stratified k-fold Cross-Validation:



Stratified sampling is a sampling technique where the samples are selected in the same proportion (by dividing the population into groups called 'strata' based on a characteristic) as they appear in the population. For example, if the population of interest has 30% male and 70% female subjects, then we divide the population into two ('male' and 'female') groups and choose 30% of the sample from the 'male' group and '70%' of the sample from the 'female' group.

Stratified k-fold Cross-Validation: The splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value, such as the class outcome value. This is called stratified cross-validation.

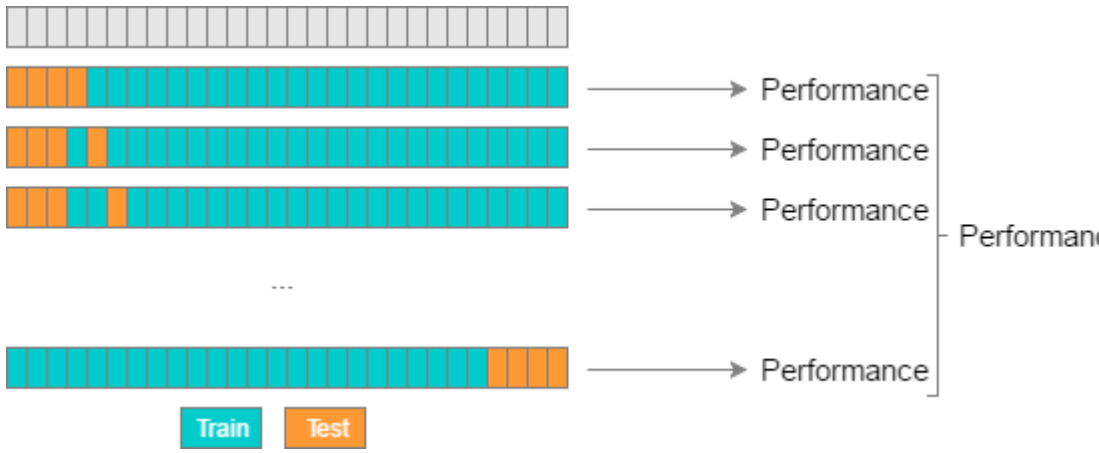
.



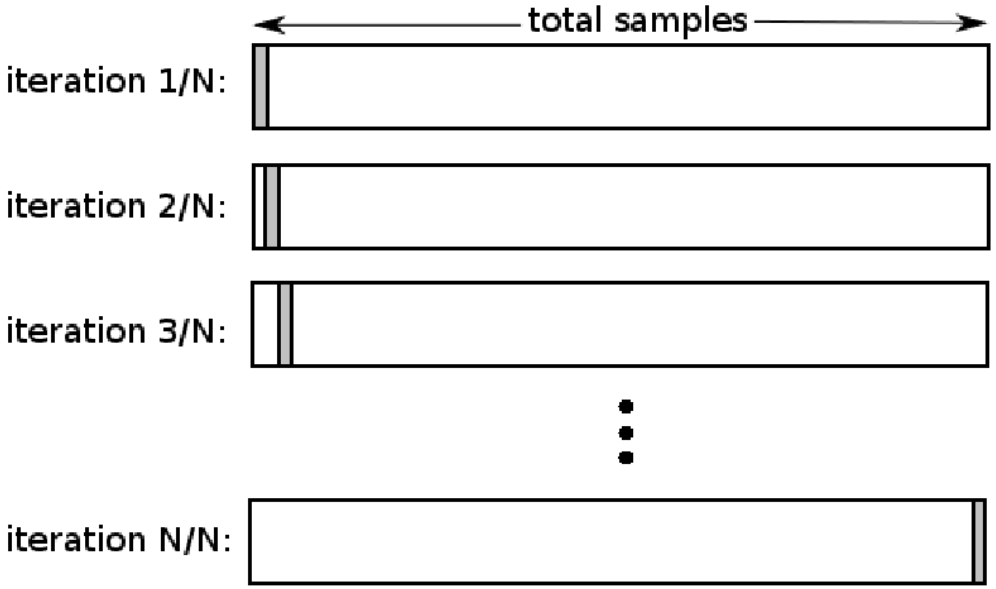
Leave-P-Out cross validation

- When using this exhaustive method, we take **p-number of points out** from the **total number of data (n)**. While training the model we **train it on these (n – p)** data points and **test the model on p data**.
- We repeat this process for **all the possible combinations of *p*** from the original dataset. Then to get the final accuracy, we average the accuracies from all these iterations.
- This is an exhaustive method as we train the model on every possible combination of data points. Remember if we choose a higher value for p, then the number of combinations will be more and we can say the method gets a lot more exhaustive.
- **Leave-one-out cross validation**

  This is a simple variation of Leave-P-Out cross validation and the value of p is set as 1. This makes the method much less exhaustive as now for n data points and p = 1, we have n number of combinations.



Leave-P-out cross validation



Leave-one-out cross validation

Considerations about k-fold Cross-Validation and its variants

- k-fold Cross-Validation is the best known and most commonly used resampling technique.
- K-fold Cross-Validation is less computer intensive than Leave-One-Out.
- The testing sets are independent of one another, as required, by many statistical testing methods, but the training sets are highly overlapping. This can affect the bias of the error estimates.
- Leave-One-Out produces error estimates with high variance given that the testing set at each fold contains only one example. The classifier is practically unbiased since each fold trains on almost all the data.

Topic 2:
Performance Metrics

Metrics

It is extremely important to use **quantitative metrics** for evaluating a machine learning model

- Until now, we relied on the **cost function value** for regression and classification
- Other metrics can be used to **better evaluate** and understand the model
- **<u>For classification</u>**
  - ✓ Accuracy/Precision/Recall/F1-score, ROC curves,…
- **<u>For regression</u>**
  - ✓ Normalized RMSE, Normalized Mean Absolute Error (NMAE),…

Confusion matrix

- A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

TP, TN, FP, FN

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.
False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

False Positives (FP) – When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN) – When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP Type-I Error
	Negative (0)	FN Type-II Error	TN

Find out the values of TP, TN, FP, FN

1=Positive class
0=Negative Class

Data	Actual	Predicted
1	1	1
2	1	1
3	0	1
4	0	1
5	1	0
6	0	0
7	1	1

TP =

TN =

FP =

FN =

Accuracy

- Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. For a model, if we have got 0.803 means our model is approx. 80% accurate.

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

- <u>Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.</u>

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

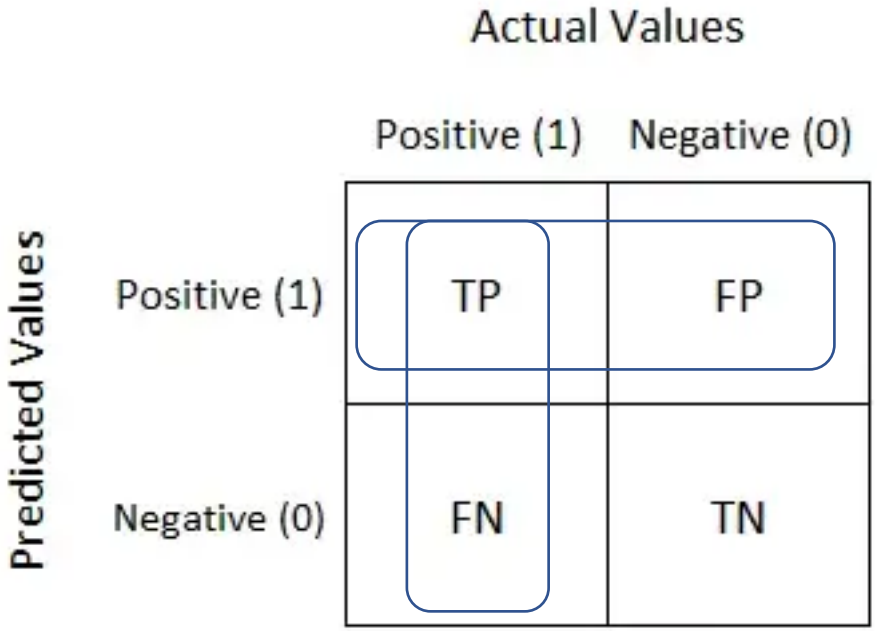
Precision, Recall

- **Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. For ex, a 0.788 precision which is pretty good.

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

- **Recall** (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? For ex, if we get recall of 0.631 which is good for this model as it's above 0.5.

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



F1-score

- F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. A F1 score is 0.701 is a good indicator of performance.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F_β Score

- $F_\beta = (1+\beta^2)\frac{Precission \times Recall}{\beta^2(Precission+Recall)}$

If β=1, it is $F1 - Score$
β=0.5, it is $F0.5 - Score$
β=2, it is $F2 - Score$

When β = 1 $\quad F_1 = (1+1^2)\frac{Precission \times Recall}{1^2 * (Precission + Recall)}$

$$F_1 = 2\frac{Precission \times Recall}{Precission + Recall}$$

It's a harmonic mean of precision and recall

Use- where FP and FN both are important.

When FP (Type-I Error) is having more impact than FN, its better to reduce the value of β.
Typically β=0.5.

When FN (Type-II Error) is having more impact than FP, its better to increase the value of β.
Typically β=2.

Practice Math

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

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

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

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP 30	FP 5
	Negative (0)	FN 10	TN 55

Accuracy = (55 + 30)/(55 + 5 + 30 + 10) = 0.85

precision = 30/(30+ 5) = 0.857

Recall = 30/(30+ 10) = 0.75

F1 Score = 2* (0.857 * 0.75)/(0.857 + 0.75) = 0.799.

At a glance

Recall Sensitivity True positive rate (TPR)	$\frac{TP}{FN+TP} = \frac{TP}{P}$
False positive rate (FPR) False alarm rate	$\frac{FP}{TN+FP} = \frac{FP}{N}$
Specificity True negative rate (TNR)	$\frac{TN}{TN+FP} = \frac{TN}{N} = 1 - FPR$
Precision	$\frac{TP}{TP+FP}$
False negative rate (FNR)	$\frac{FN}{FN+TP} = \frac{FN}{P}$
Accuracy	$\frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

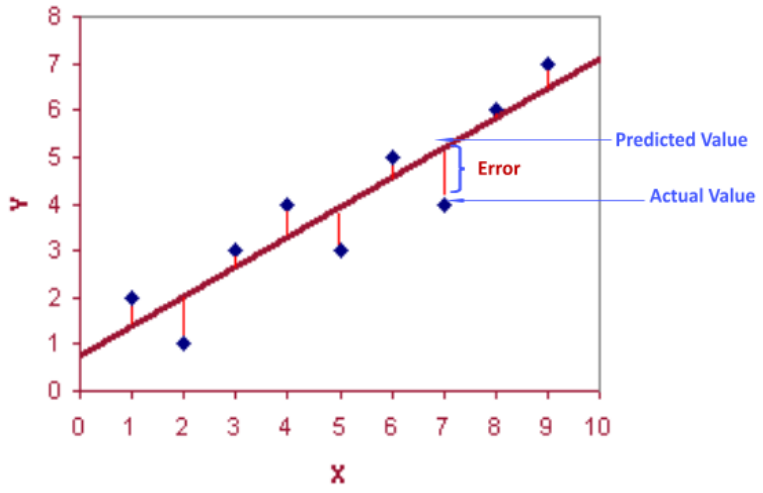
Root-Mean Squared Error (RMSE)

- The Root-Mean Squared Error (RMSE) is usually used for **regression**, but can also be used with probabilistic classifiers. The formula for the RMSE is:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Where,
n =total number of test examples
$\hat{y}$ = predicted output
y_i = the actual label.

No	$\hat{y}$	y_i	$(\hat{y} - y_i)^2$
1	.95	1	.0025
2	.6	0	.36
3	.8	1	.04
4	.75	0	.5625
5	.9	1	.01

$$\text{RMSE(f)} = \frac{\sqrt{(.0025+.36+.04+.5625+.01)}}{5} = 0.4416$$

RMSE calculates total error in a prediction