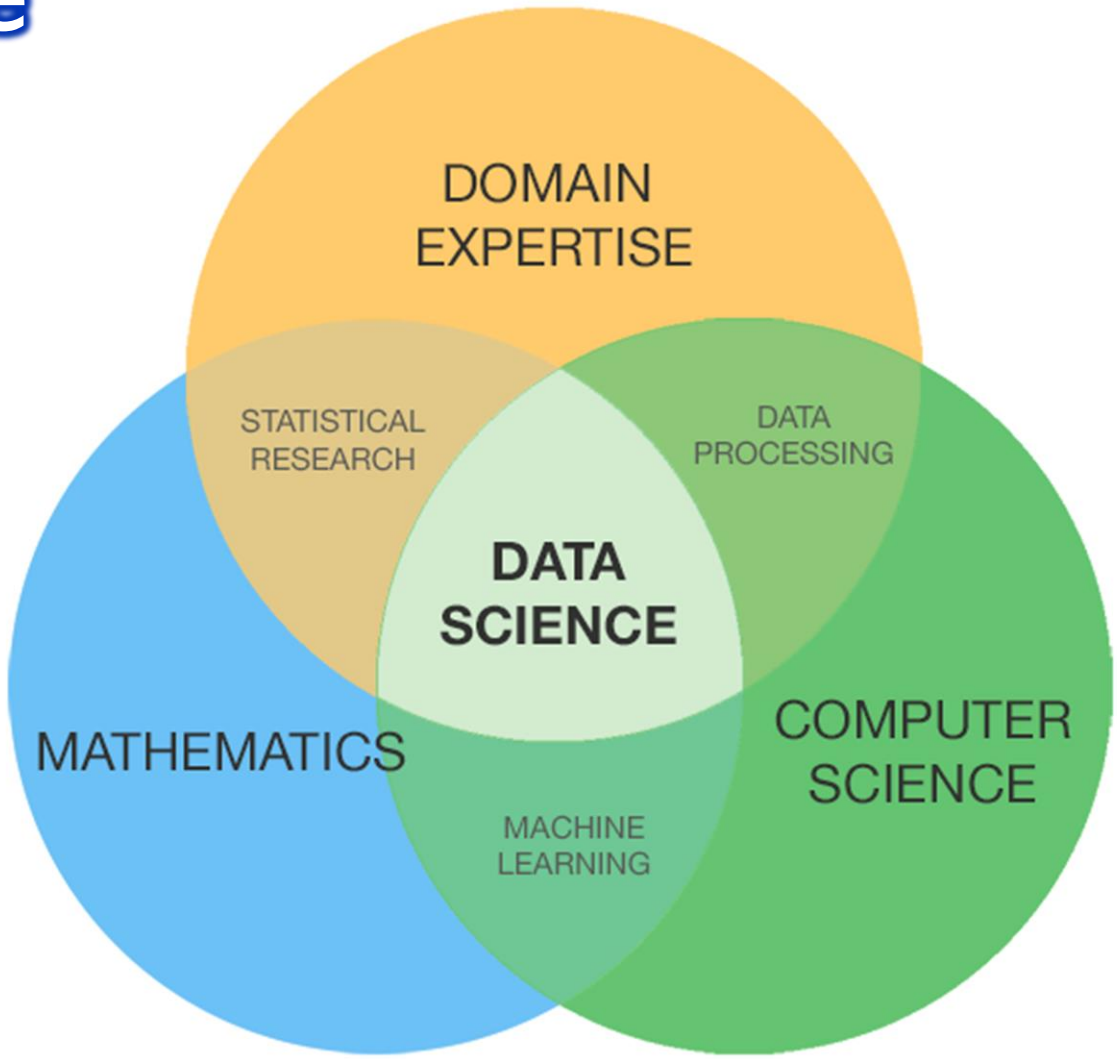
Machine Learning

Parameswari Ettiappan

Data Science – A Definition

Data Science is the science which uses computer science, statistics and machine learning, visualization and human-computer interactions to collect, clean, integrate, analyze, visualize, interact with data to create data products.

Data Science



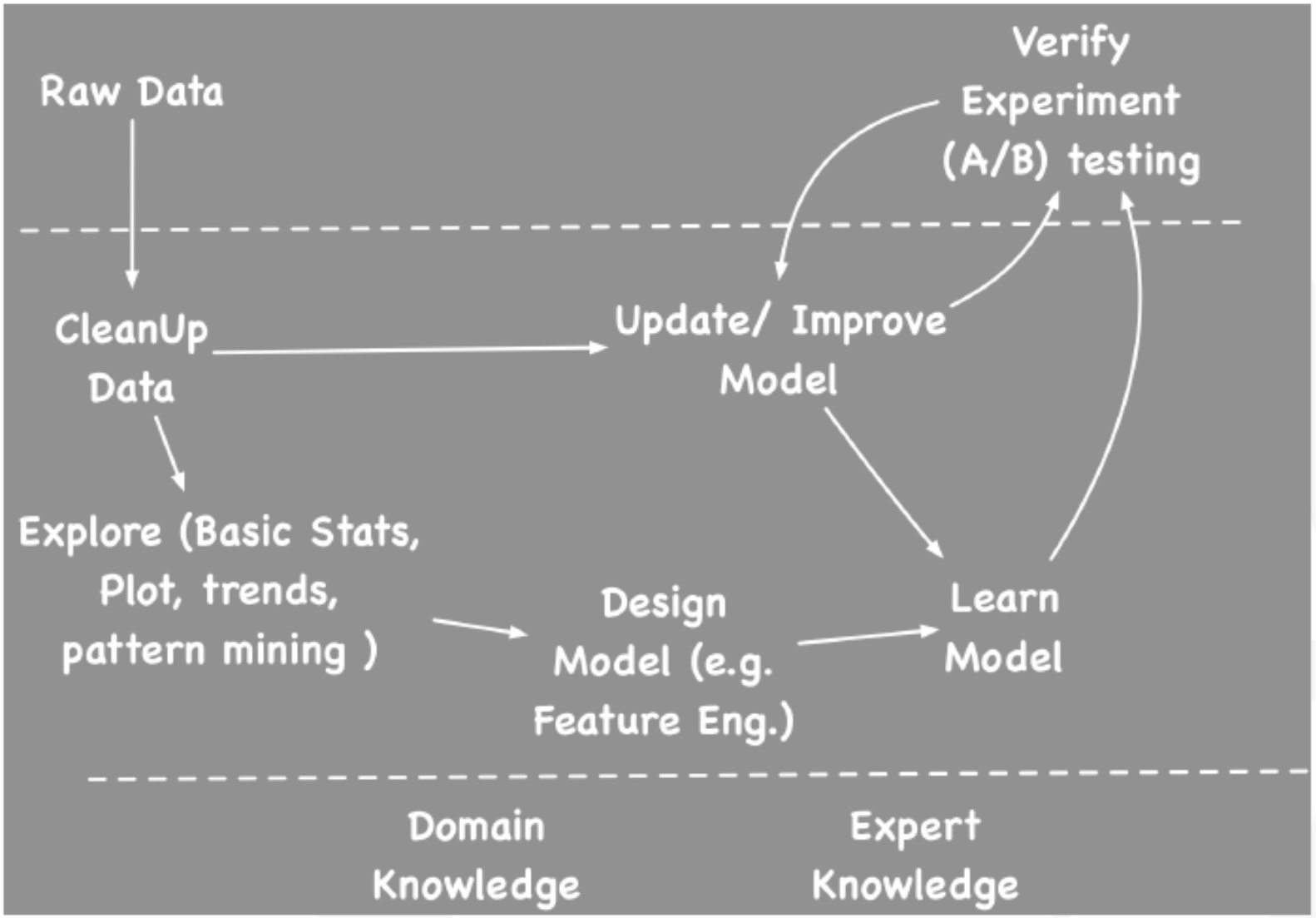
Source: Palmer, Shelly. Data Science for the C-Suite. New York: Digital Living Press, 2015. Print.

What is Data Science?

Extraction of knowledge from large volumes of data that are structured or unstructured.

It is a continuation of the fields **data mining** and **predictive analytics**

Data Science Pipeline

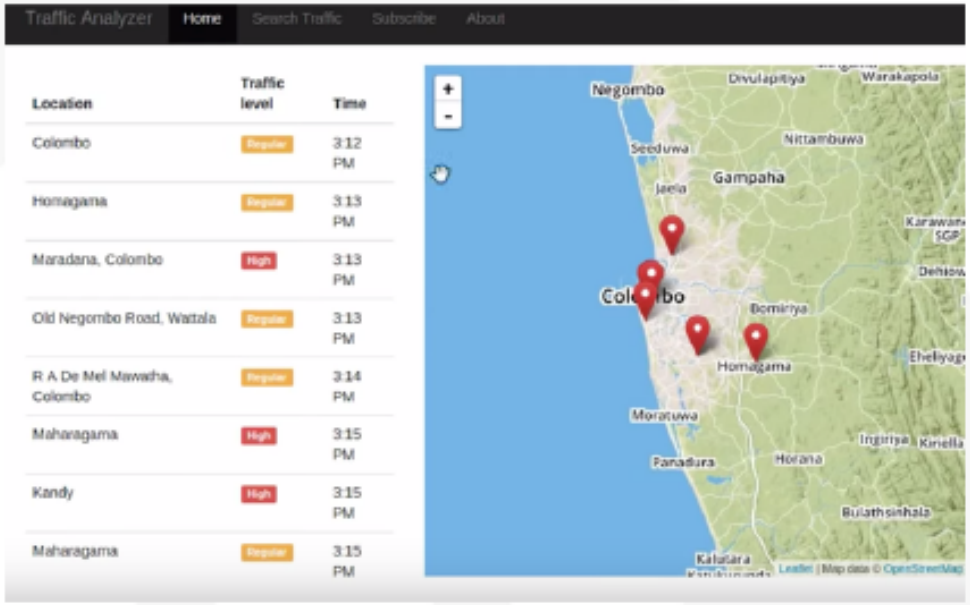


Data Science and Others

- Business Intelligence
- Statistics
- Data(base) Management
- Visualization
- Machine Learning
- Data Mining
- Artificial Intelligence
- Big Data

Example (Road.lk) traffic Feed

1. Data as tweets
2. Extract time, location, and traffic level using NLP
3. Explore data
4. Model based on time, and it is a holiday
5. Predict traffic given a time and location.



Big Data Science Tasks

- Facebook
- Amazon
- Google
- LinkedIn
- Netflix
- Rozetka
- Microsoft

Regular Data Science Tasks

- Data analysis
  - What percentage of users back to our site?
  - Which products usually bought together?
- Modeling/statistics
  - How many cars we are going to sell next year?
  - Which city is better for opening new office?
- Engineering/prototyping
  - Product to use a prediction model
  - Visualization of analytics


Human VS. Machine
VS

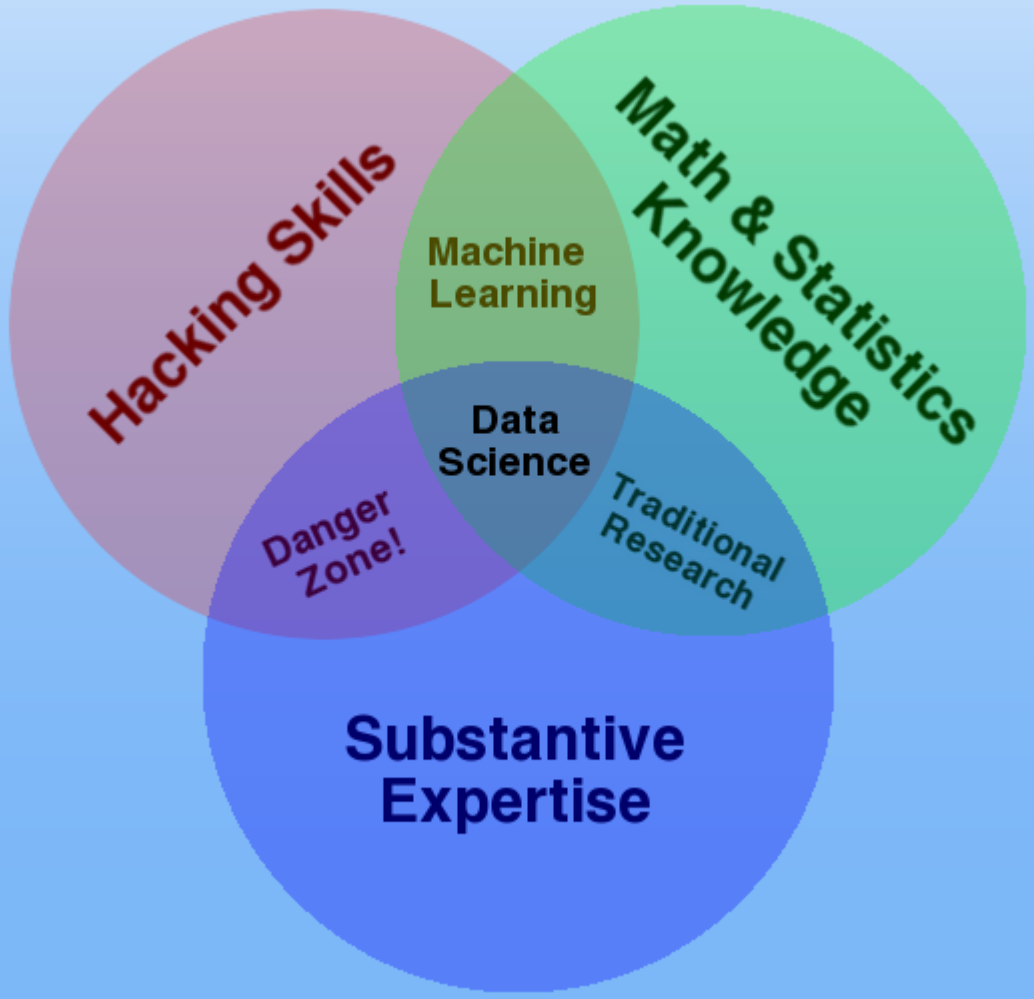
Human vs. Machine

- Human
  - Naturally can work with small amount of data
  - Have a knowledge about domain
  - Good image recognition
- Machines
  - Can make intensive computations
  - Knows only numbers and strings (well, actually only numbers)

Goal of Data Science

Turn data into data products.

Data Science – A Visual Definition



Contrast: Databases

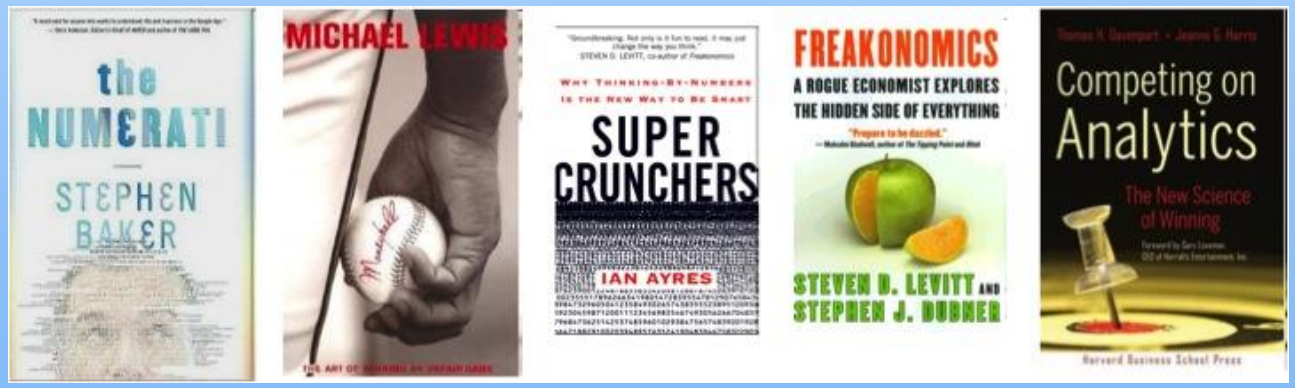
	Databases	Data Science
Data Value	"Precious"	"Cheap"
Data Volume	Modest	Massive
Examples	Bank records, Personnel records, Census, Medical records	Online clicks, GPS logs, Tweets, Building sensor readings
Priorities	Consistency, Error recovery, Auditability	Speed, Availability, Query richness
Structured	Strongly (Schema)	Weakly or none (Text)
Properties	Transactions, ACID*	CAP* theorem (2/3), eventual consistency
Realizations	SQL	NoSQL: MongoDB, CouchDB, Hbase, Cassandra, Riak, Memcached, Apache River, …

ACID = Atomicity, Consistency, Isolation and Durability

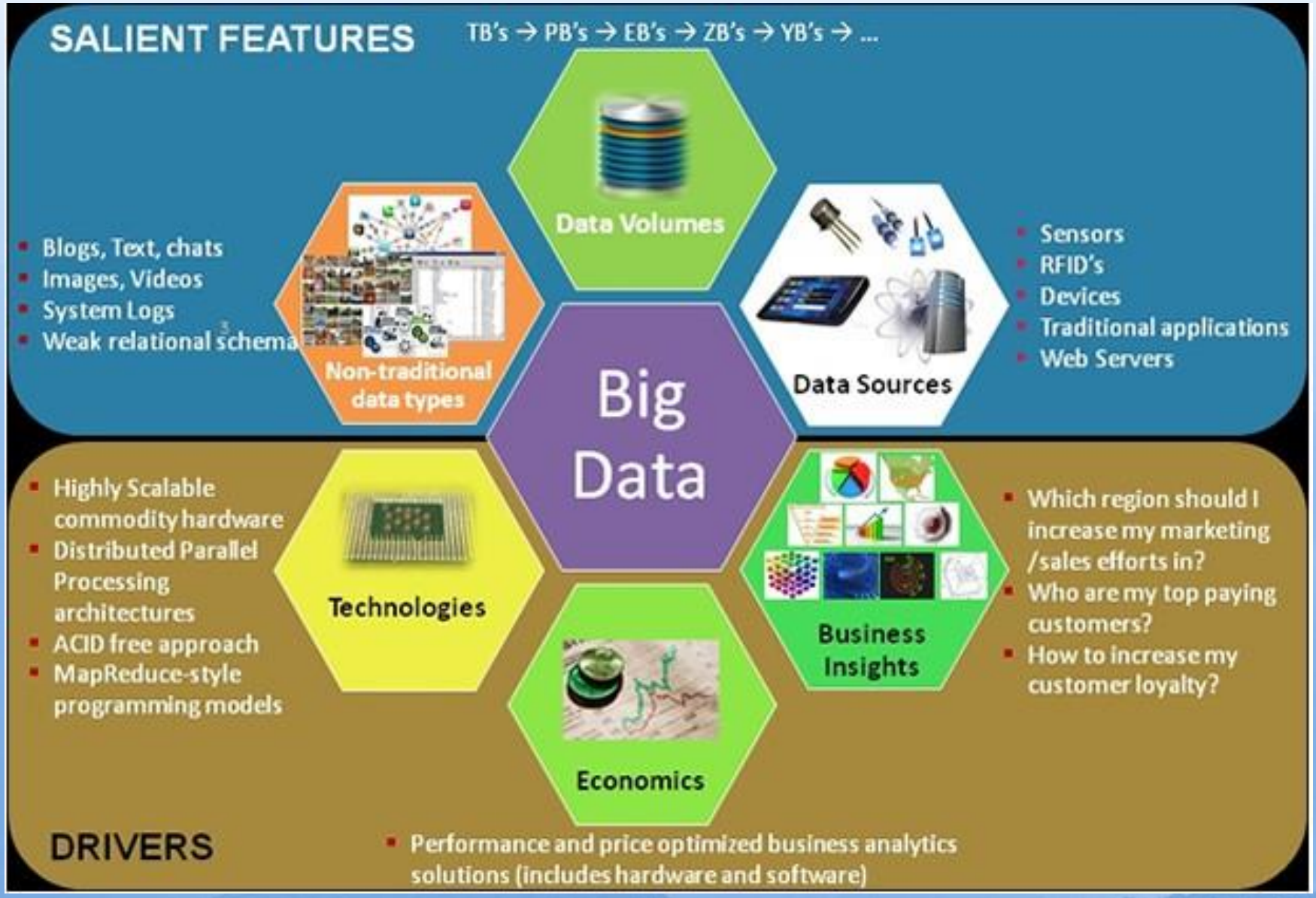
CAP = Consistency, Availability, Partition Tolerance

Contrast: Business Intelligence

Business Intelligence	Data Science
Querying the past	Querying the past present and future



	Business Intelligence	Data Science
Perspective	Looking backwards	Looking forwards
Actions	Slice and Dice	Interact
Expertise	Business User	Data Scientist
Data	Warehoused, Siloed	Distributed, real-time
Scope	Unlimited	Specific business question
Questions	What happened?	What will happen? What if?
Output	Table	Answer
Applicability	Historic, possible confounding factors	Future, correcting for influences
Tools	SAP, Cognos, Microstrategy, SAS	Revolution R Enterprise QlikView, Tableau, Jaspersoft
Hot or not?	*So* 1997	Transformational


SALIENT FEATURES
TB's → PB's → EB's → ZB's → YB's → ...
Data Volumes
Blogs, Text, chats
Images, Videos
System Logs
Weak relational schema
Non-traditional data types
Big Data
Data Sources
Sensors
RFID's
Devices
Traditional applications
Web Servers
Highly Scalable commodity hardware
Distributed Parallel Processing architectures
ACID free approach
MapReduce-style programming models
Technologies
Economics
Business Insights
Which region should I increase my marketing /sales efforts in?
Who are my top paying customers?
How to increase my customer loyalty?
DRIVERS
Performance and price optimized business analytics solutions (includes hardware and software)

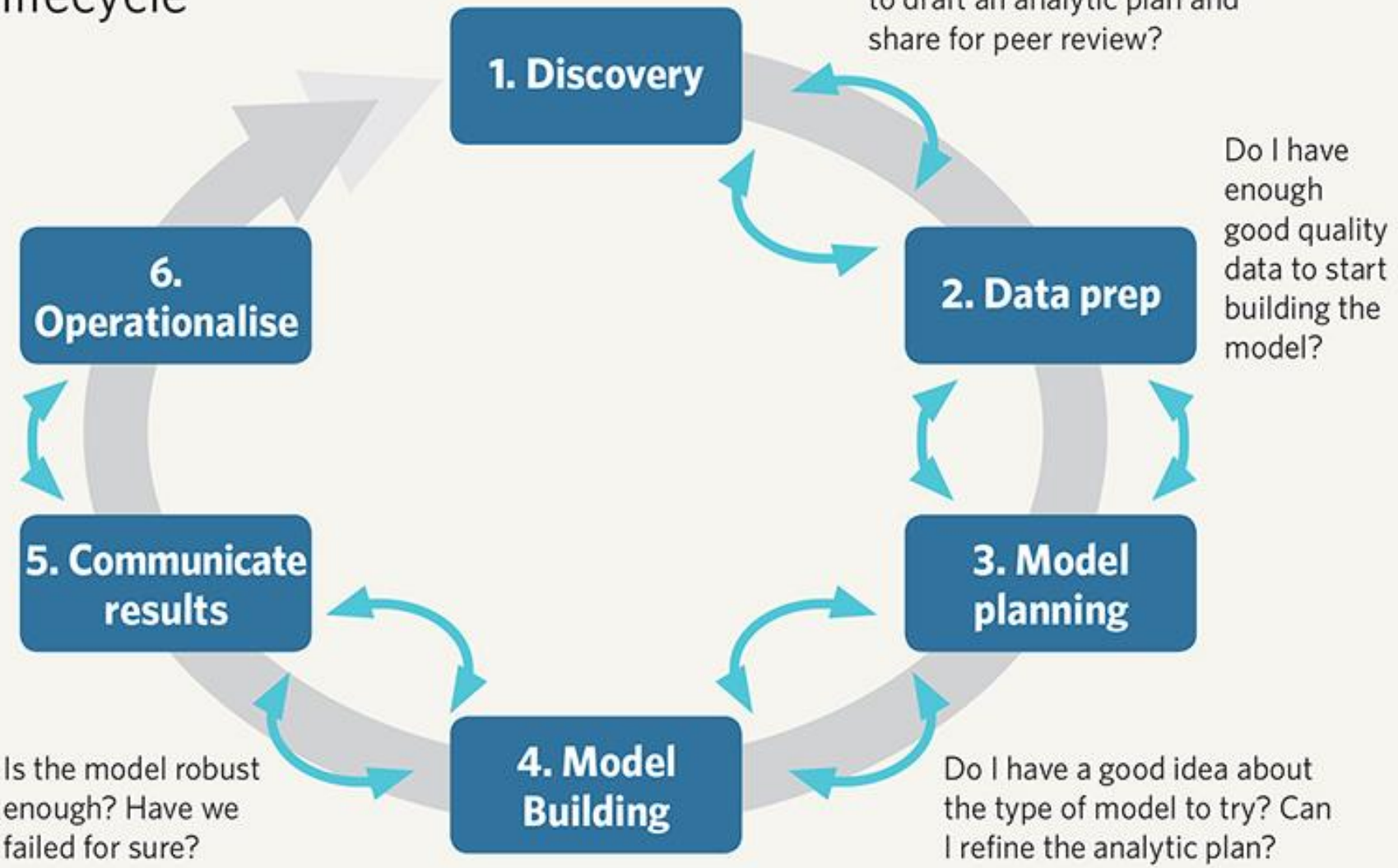
Examples of Big Data Analytics

- Using Big Data to win elections
- Big Data for finding a perfect match(Matrimony.com)
- Big Data for detecting water leakages(Bangalore Water supply and sewage system)
- Big Data for gaining insights into shopping behavior

Examples of Big Data Analytics

- **Big Data for ensuring proper water supply(Kerala Water Supply System)**
- **Using Big Data to improve India's financial inclusion ratio(**Micro-finance firm, Janalakshmi Financial Services)
- **Using Big Data to improve product development(Reliance Games)**
- **Using Big Data to predict ticket confirmations for trains(**PNR prediction)

Data Analytics lifecycle



Data Modeling Tools

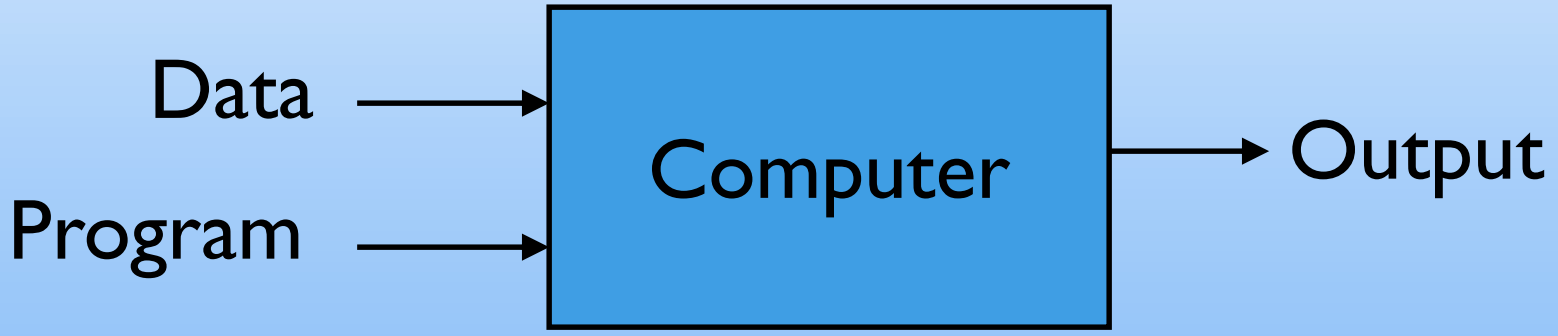
- PowerDesigner
- ER/Studio
- Sparx Enterprise Architect
- Oracle SQL Developer Data Modeler
- CA Erwin
- IBM - InfoSphere Data Architect
- MagicDraw

Contrast: Machine Learning

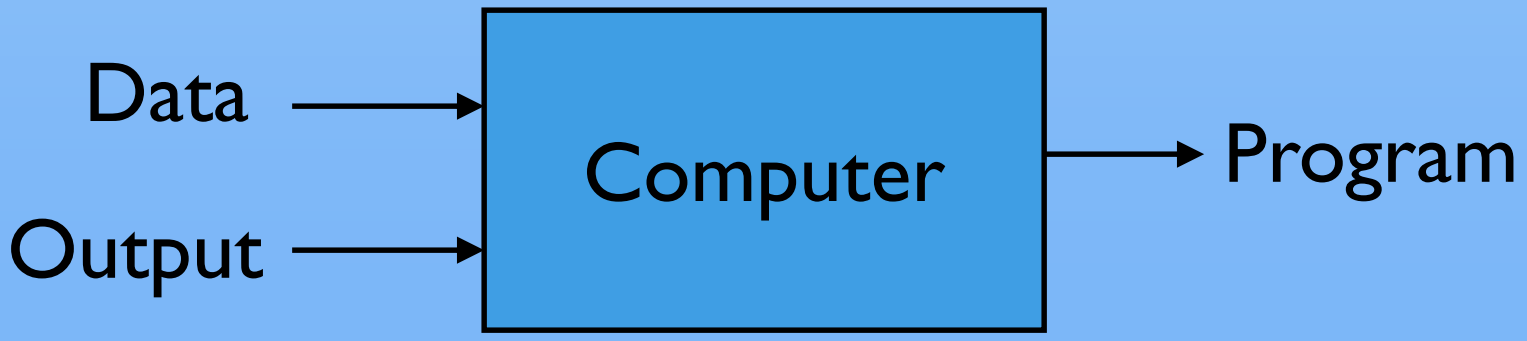
Machine Learning
Develop new (individual) models
Prove mathematical properties of models
Improve/validate on a few, relatively clean, small datasets
Publish a paper

Data Science
Explore many models, build and tune hybrids
Understand empirical properties of models
Develop/use tools that can handle massive datasets
Take action!

Traditional Programming



Machine Learning



AI And Machine Learning Use Cases

- **Data Security**
- **Personal Security**
- **Healthcare**
- **Financial Trading**
- **Marketing Personalization**
- **Fraud Detection**
- **Recommendations**
- **Online Search**
- **Natural Language Processing (NLP)**
- **Smart Cars**

Magic?

No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs

ML in a Nutshell

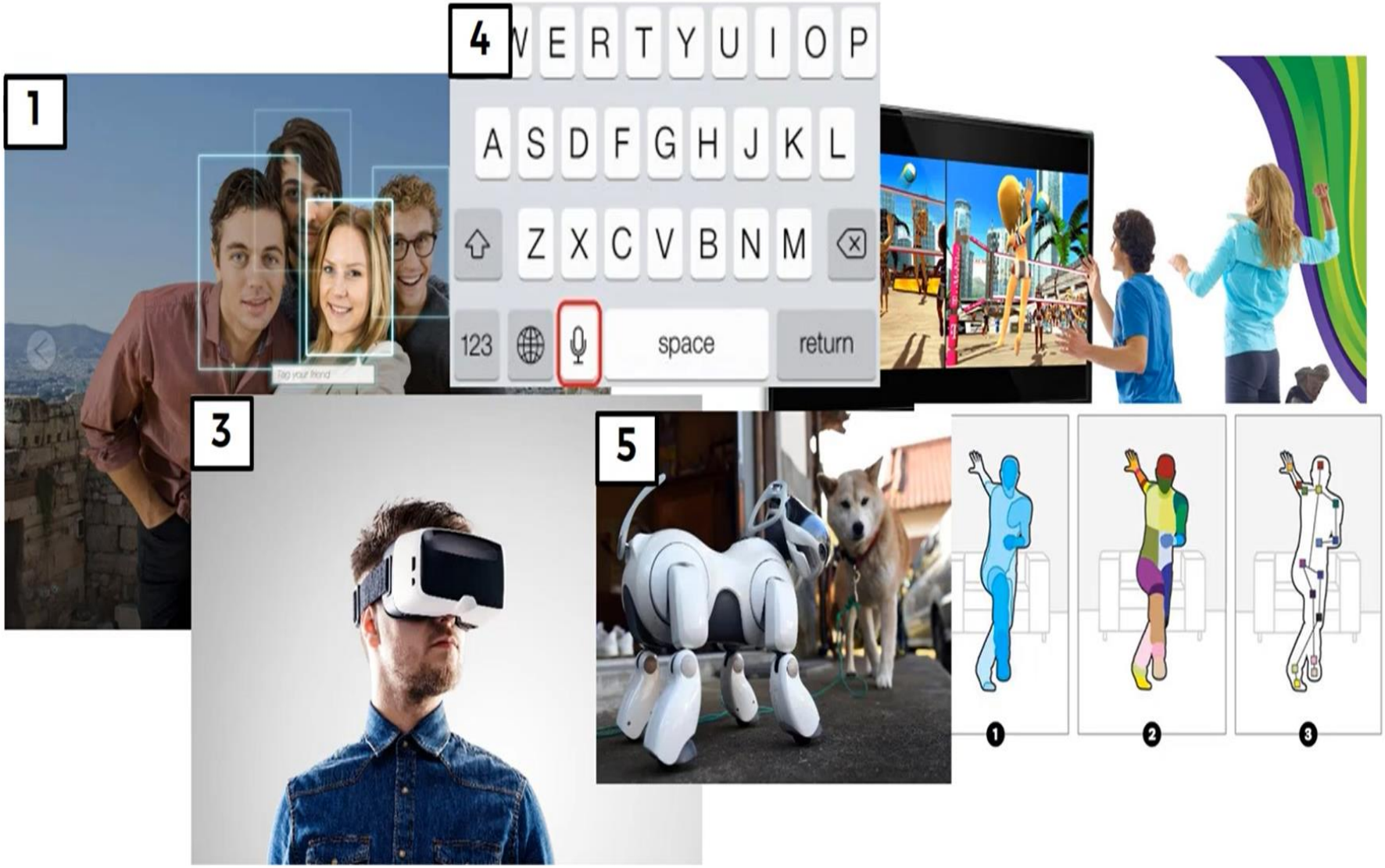
- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

What is machine learning?

- A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

Applications of ML



Applications of ML

ML Is The Future

ML Is The Future

ML Is The Future

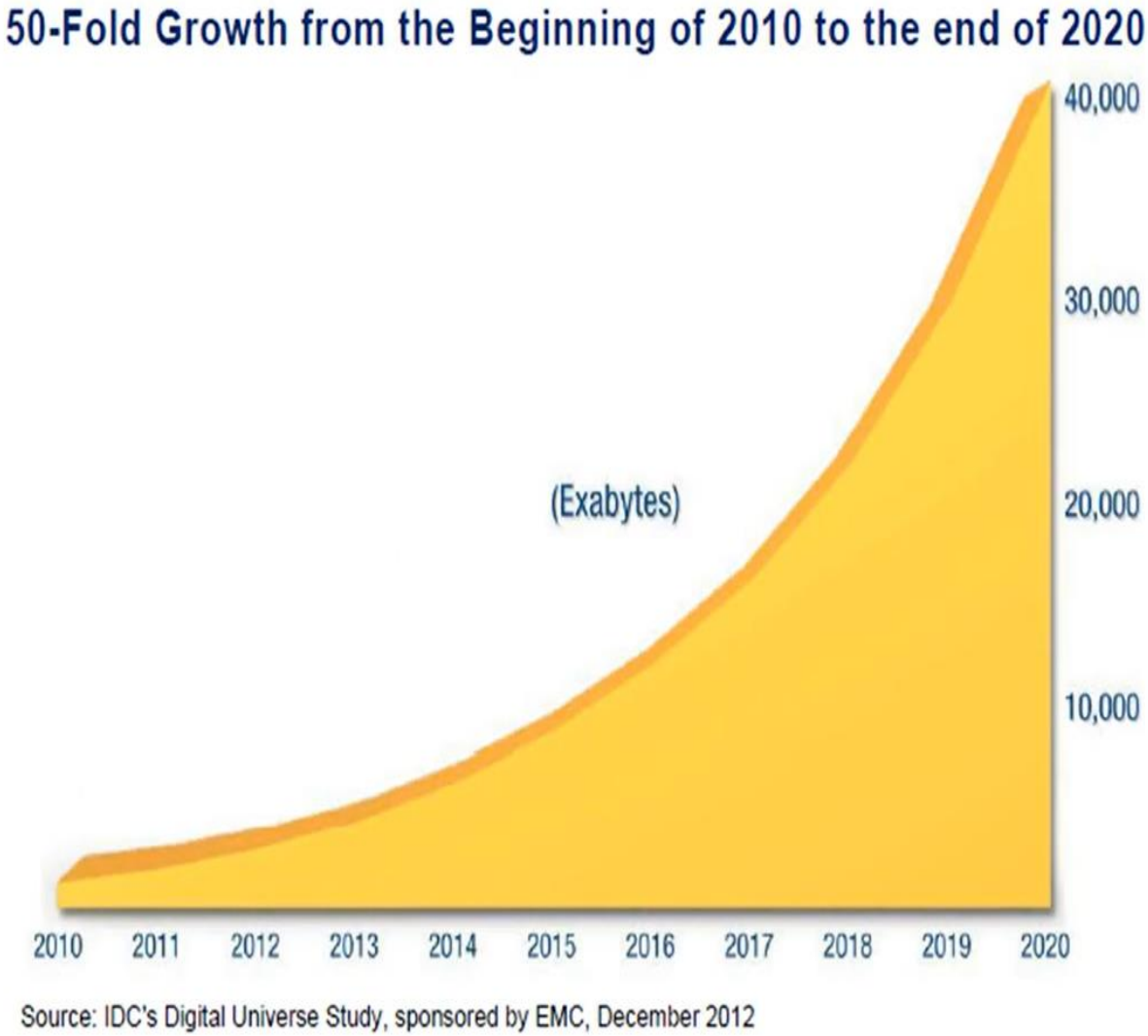
2005 – 130 EXABYTES

2010 – 1,200 EXABYTES

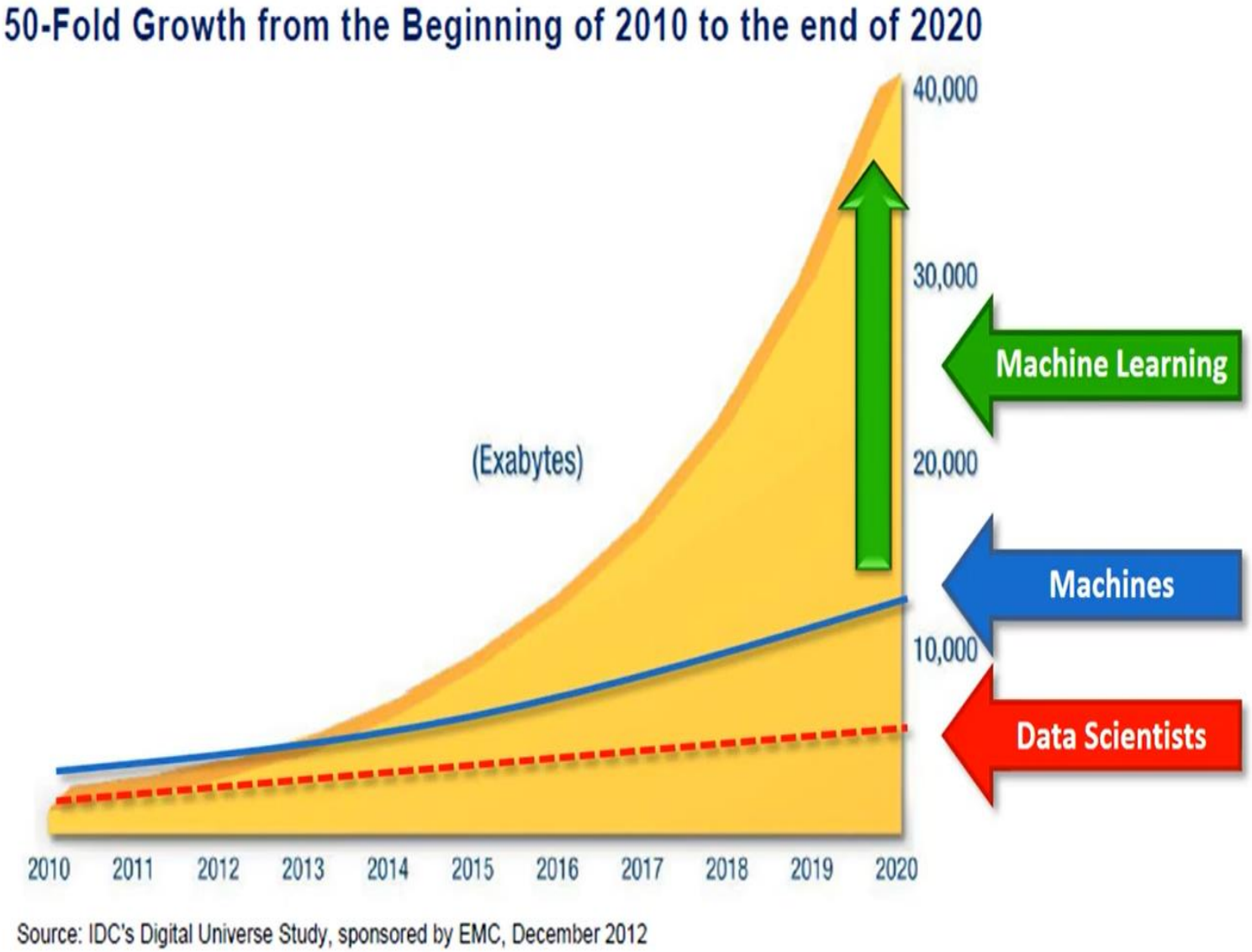
2015 – 7,900 EXABYTES

2020 – 40,900 EXABYTES

ML Is The Future



ML Is The Future



ML Is The Future

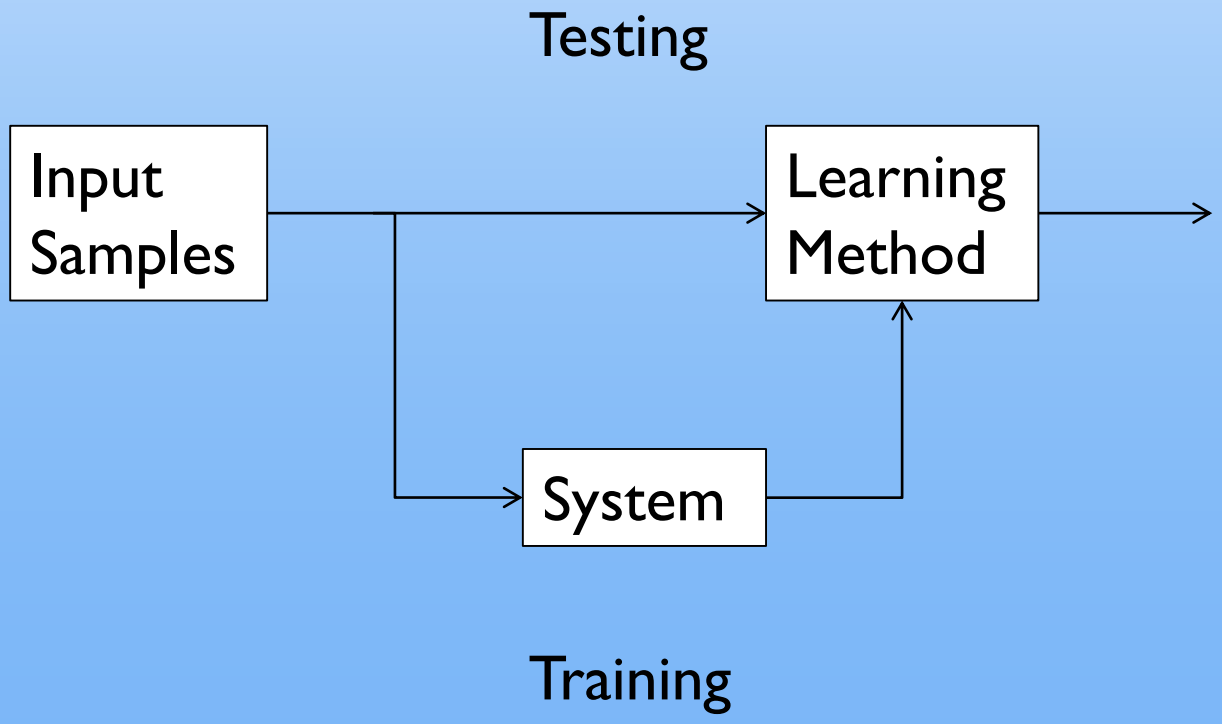
Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

Applications

- Face detection
- Object detection and recognition
- Image segmentation
- Multimedia event detection
- Economical and commercial usage

Learning system model



Descriptive Statistics

- *Descriptive statistics* are numbers that are used to summarize and describe data.
- The word "data" refers to the information that has been collected from an experiment, a survey, a historical record, etc. (By the way, "data" is plural.
- One piece of information is called a "datum.")
- If we are analyzing birth certificates, for example, a descriptive statistic might be the percentage of certificates issued in New York State, or the average age of the mother.

Descriptive Statistics

Average salaries for various occupations in 1999.

$112,760	pediatricians
$106,130	dentists
$100,090	podiatrists
$ 76,140	physicists
$ 53,410	architects
$ 49,720	school, clinical, and counseling psychologists
$ 47,910	flight attendants
$ 39,560	elementary school teachers
$ 38,710	police officers
$ 18,980	floral designers

Types of Statistical Variables

- **Categorical variable**: variables than can be put into categories. For example, the category "Toothpaste Brands" might contain the variables Colgate and Aquafresh.
- **Confounding variable**: extra variables that have a hidden effect on your experimental results.
- **Continuous variable**: a variable with infinite number of values, like "time" or "weight".
- **Control variable**: a factor in an experiment which must be held constant. For example, in an experiment to determine whether light makes plants grow faster, you would have to control for soil quality and water.
- **Dependent variable**: the outcome of an experiment. As you change the independent variable, you watch what happens to the dependent variable.

Types of Statistical Variables

- **Discrete variable**: a variable that can only take on a certain number of values. For example, "number of cars in a parking lot" is discrete because a car park can only hold so many cars.
- **Independent variable**: a variable that is not affected by anything that you, the researcher, does. Usually plotted on the x-axis.
- A **measurement variable** has a number associated with it. It's an "amount" of something, or a"number" of something.
- **Nominal variable**: another name for categorical variable.
- **Ordinal variable**: similar to a categorical variable, but there is a clear order. For example, income levels of low, middle, and high could be considered ordinal.
- **Qualitative variable**: a broad category for any variable that can't be counted (i.e. has no numerical value). Nominal and ordinal variables fall under this umbrella term.

Types of Statistical Variables

- **Quantitative variable:** A broad category that includes any variable that can be counted, or has a numerical value associated with it. Examples of variables that fall into this category include discrete variables and ratio variables.
- **Ratio variables**: similar to interval variables, but has a meaningful zero.

Summary Statistics of Numeric Variables

- **Mean.** A measure of central tendency. It is the arithmetic average; the sum divided by the number of cases.
- **Sum.** The sum or total of the values.
- **Minimum.** The smallest value.
- **Maximum.** The largest value.
- **Range.** The difference between the largest and smallest values--the maximum minus the minimum.
- **Mode.** The most frequently occurring value. If several values share the greatest frequency of occurrence, each of them is a mode.

Summary Statistics of Numeric Variables

- **Median.** The value above and below which half the cases fall; the 50th percentile. If there is an even number of cases, the median is the average of the two middle cases when they are sorted in ascending or descending order. The median is a measure of central tendency not sensitive to outlying values--unlike the mean, which can be affected by one or more extremely high or low values.
- **Percentile.** A value that divides cases according to values below which certain percentages fall. For example, the 25th percentile is the value below which 25% of cases fall.

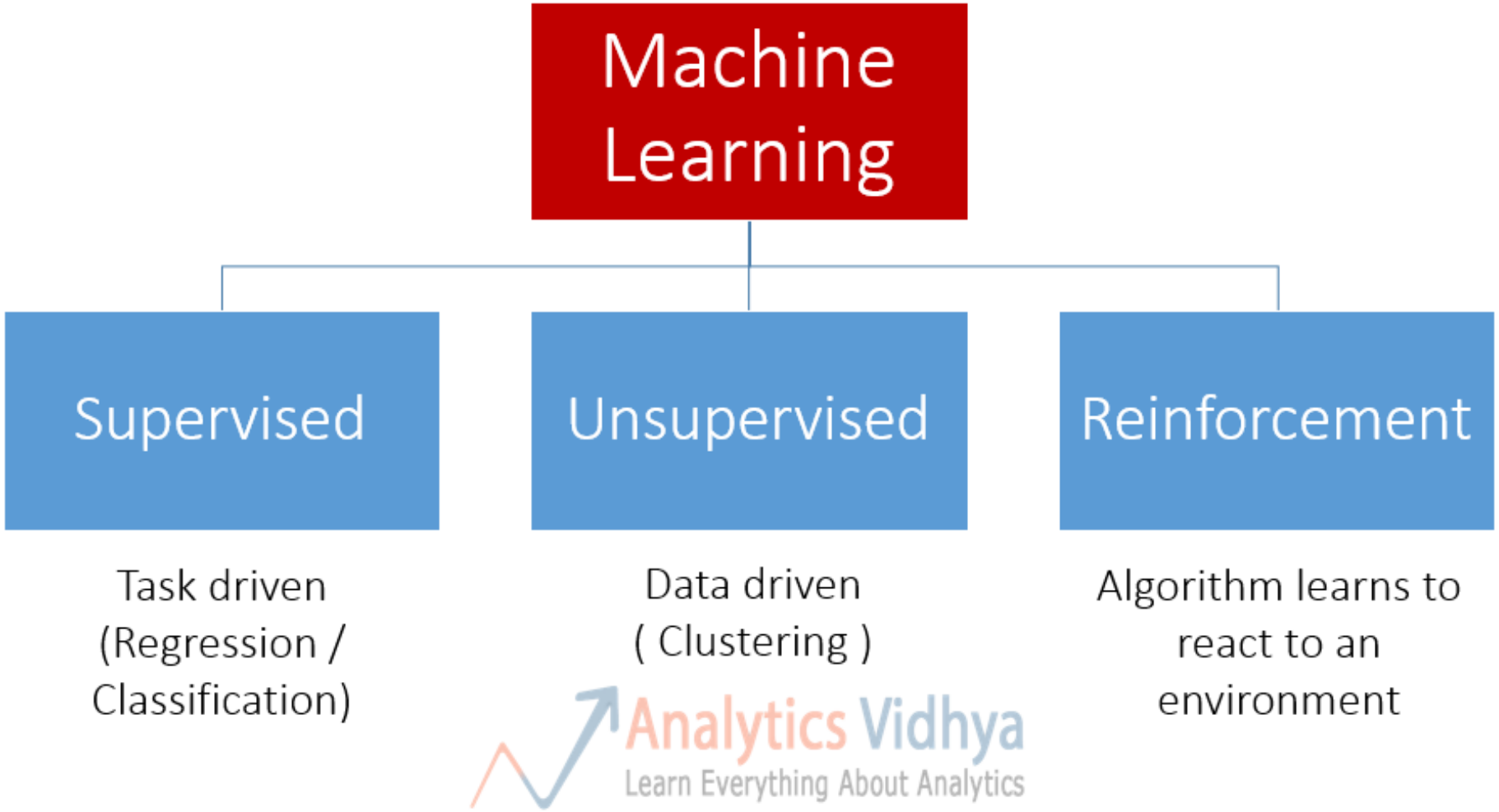
Summary Statistics of Numeric Variables

- **Standard error.** A measure of how much the value of the mean may vary from sample to sample taken from the same distribution. The standard error of the sample mean can be used to estimate a mean value for the population as a whole. In a normal distribution, 95% of the values of the mean should lie in the range of plus and minus two times the standard error from the mean. Additionally, the standard error can be used to roughly compare the observed mean to a hypothesized value of another mean (that is, you can conclude the two values are different if the ratio of the difference to the standard error is less than -2 or greater than +2).
- • **Variance.** This is the sample variance, which is a measure of dispersion around the mean, equal to the sum of squared deviations from the mean divided by one less than the number of cases. The sample variance is measured in units that are the square of those of the variable itself.

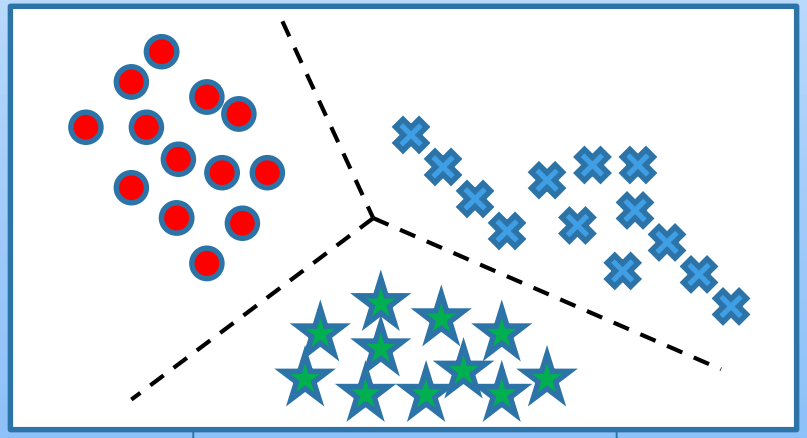
Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

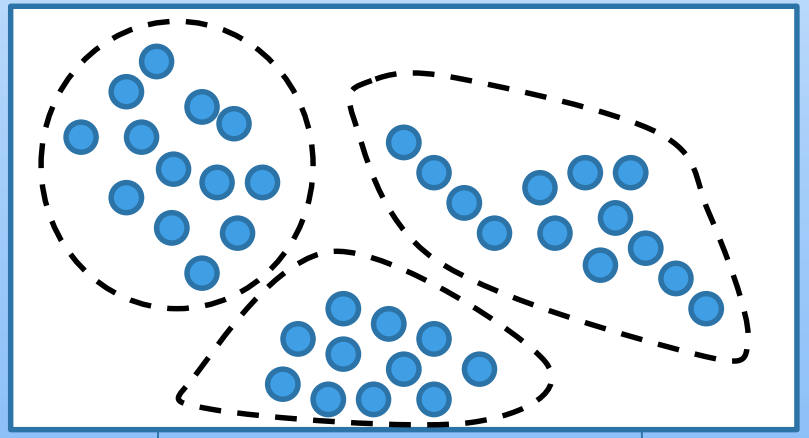
Types of Machine Learning



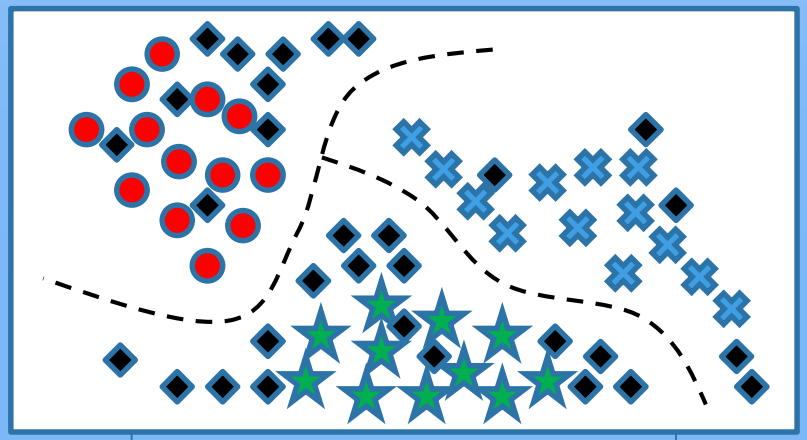
Algorithms



Supervised learning



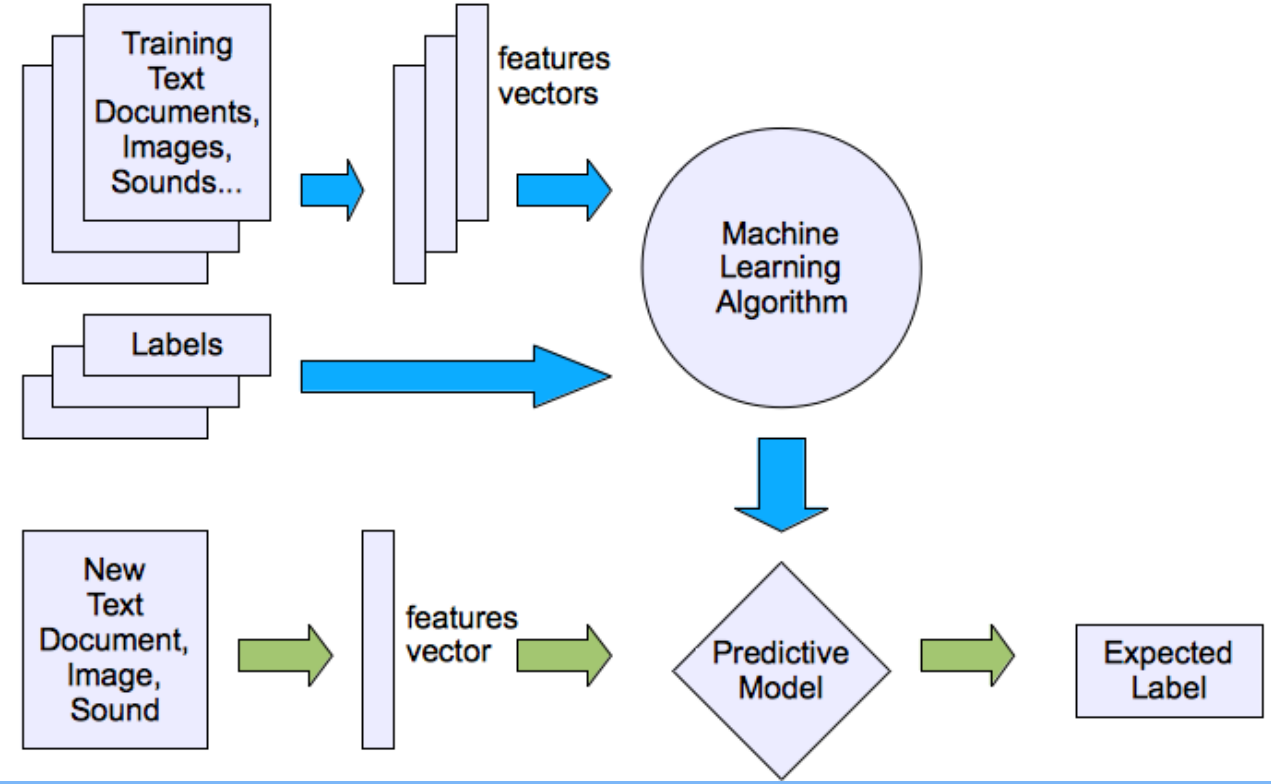
Unsupervised learning



Semi-supervised learning

Machine learning structure

- Supervised learning



What are we seeking?

- Supervised: Low E-out or maximize probabilistic terms

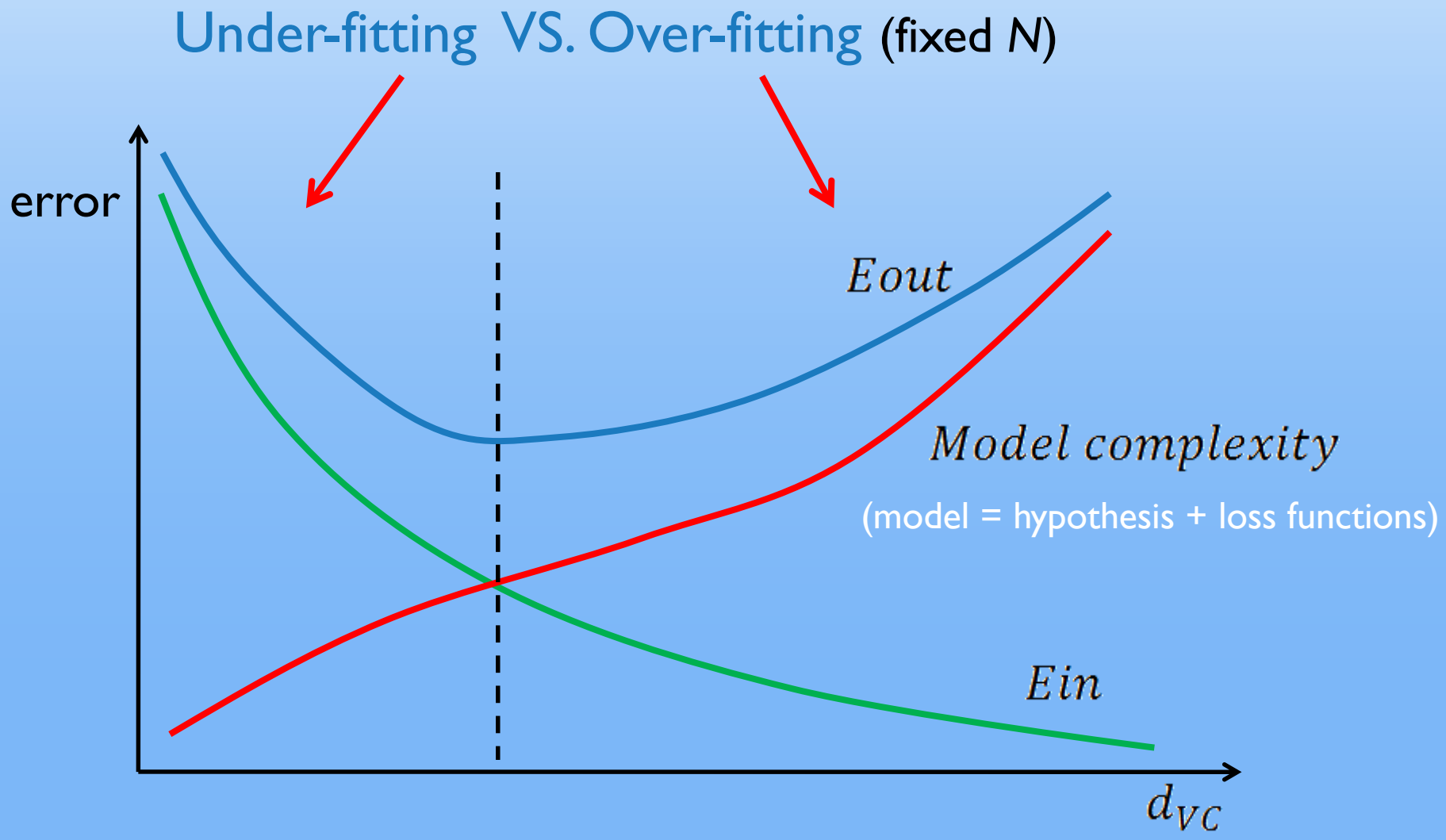
$$error = \frac{1}{N}\sum_{n=1}^{N}[y_n \neq g(x_n)]$$

E-in: for training set
E-out: for testing set

$$Eout(g) \leq Ein(g) \pm O\left(\sqrt{\frac{d_{VC}}{N} ln\, N}\right)$$

- Unsupervised: Minimum quantization error, Minimum distance, MAP, MLE(maximum likelihood estimation)

What are we seeking?

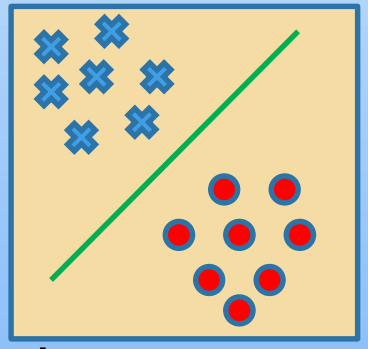


Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

- Linear classifier



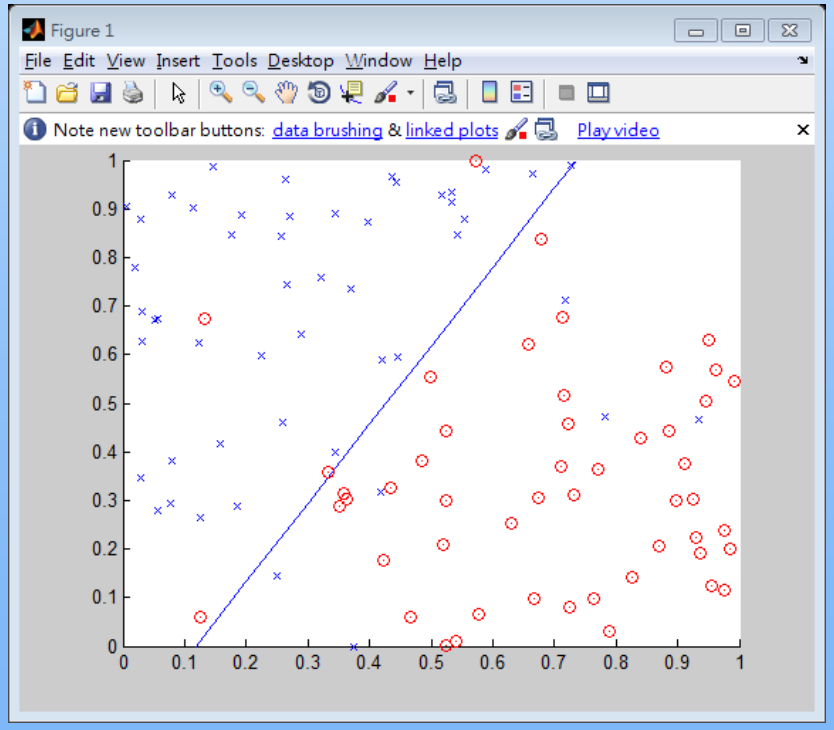
$$g(x_n) = sign(w^T x_n)$$

, where *w* is an *d*-dim vector (learned)

- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Ada-line
  - Multi-layer perceptron (MLP)

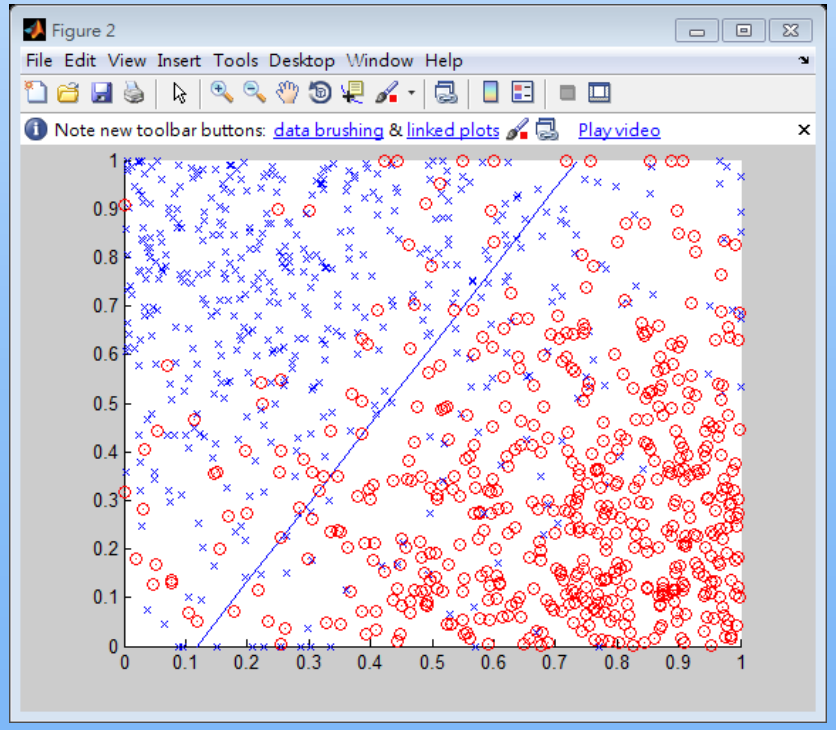
Learning techniques

Using **perceptron learning algorithm**(PLA)



Training

Error rate: 0.10

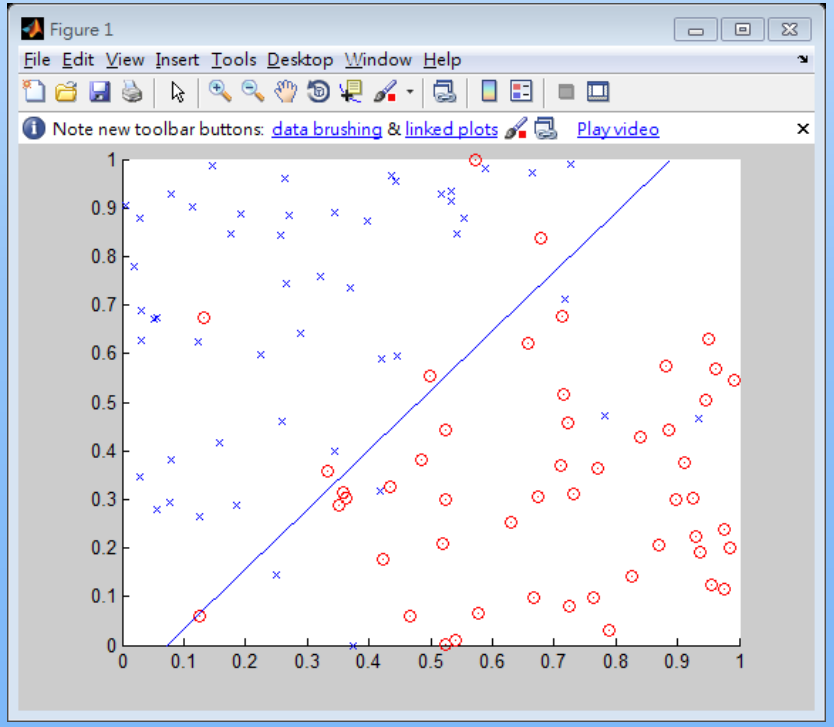


Testing

Error rate: 0.156

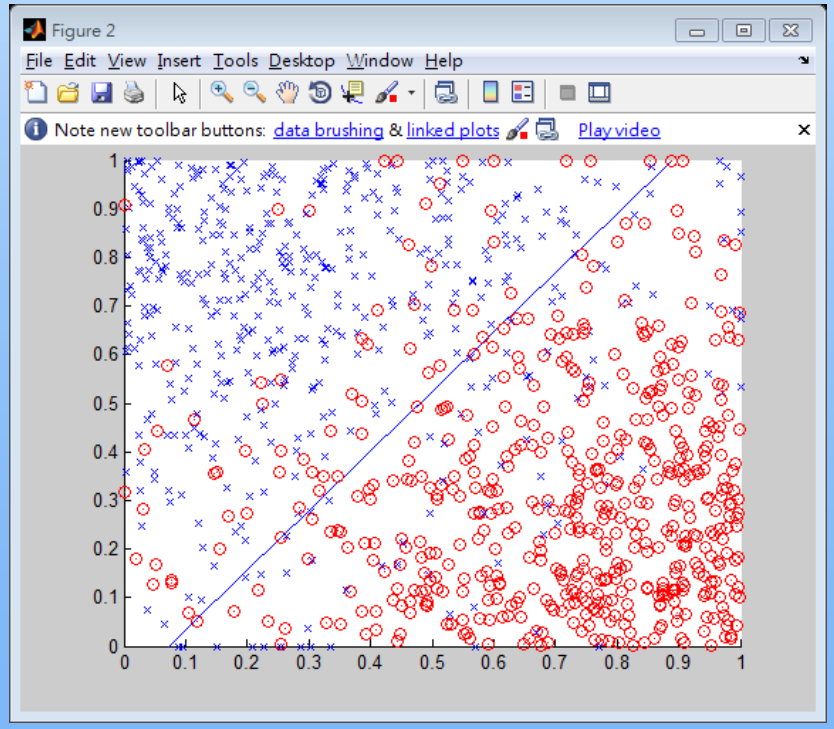
Learning techniques

Using **logistic regression**



Training

Error rate: 0.11

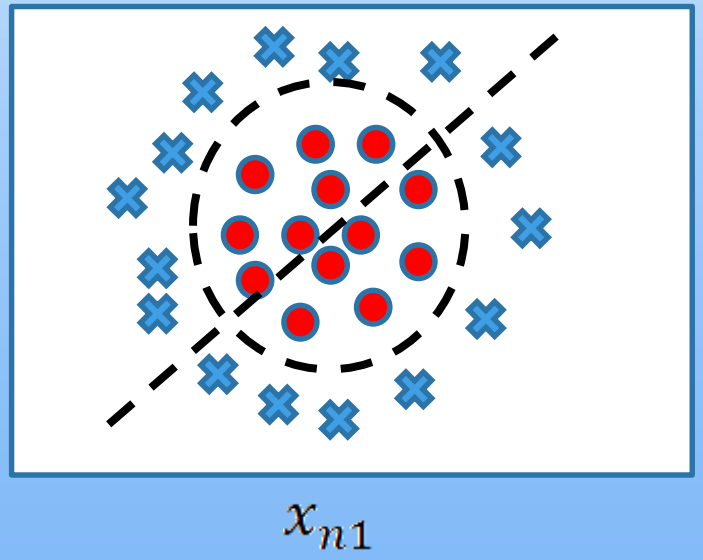


Testing

Error rate: 0.145

Learning techniques

- Non-linear case



$$x_n = [x_{n1}, x_{n2}]$$

$$x_n = [x_{n1}, x_{n2}, x_{n1} * x_{n2}, {x_{n1}}^2, {x_{n2}}^2]$$

$$g(x_n) = sign(w^T x_n)$$

- Support vector machine (SVM):
  - Linear to nonlinear: **Feature transform** and **kernel function**

Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density Estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

Supervised Learning

An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have "experiences".
- A computer system learns from data, which represent some "past experiences" of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: $A_1, A_2, \ldots A_k$.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

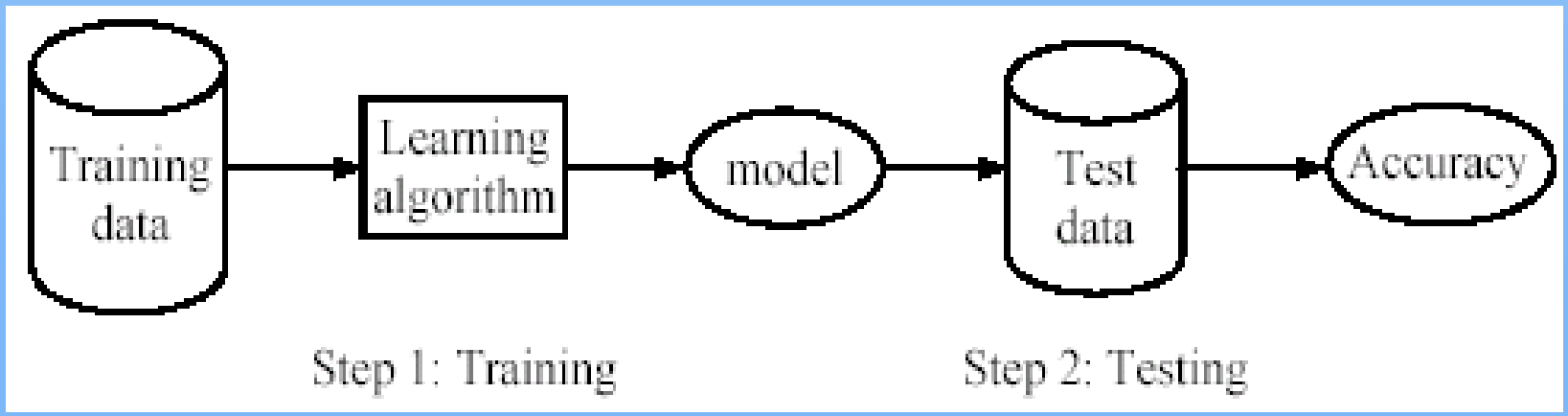
Supervised vs. unsupervised Learning

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a "teacher" gives the classes (supervision).
  - Test data are classified into these classes too.
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

Supervised learning process: two steps

- Learning (training): Learn a model using the training data
- Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



What do we mean by learning?

- Given
  - a data set *D*,
  - a task *T*, and
  - a performance measure *M*,

  a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.
- In other words, the learned model helps the system to perform *T* better as compared to no learning.

An example

- Data: Loan application data
- Task: Predict whether a loan should be approved or not.
- Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

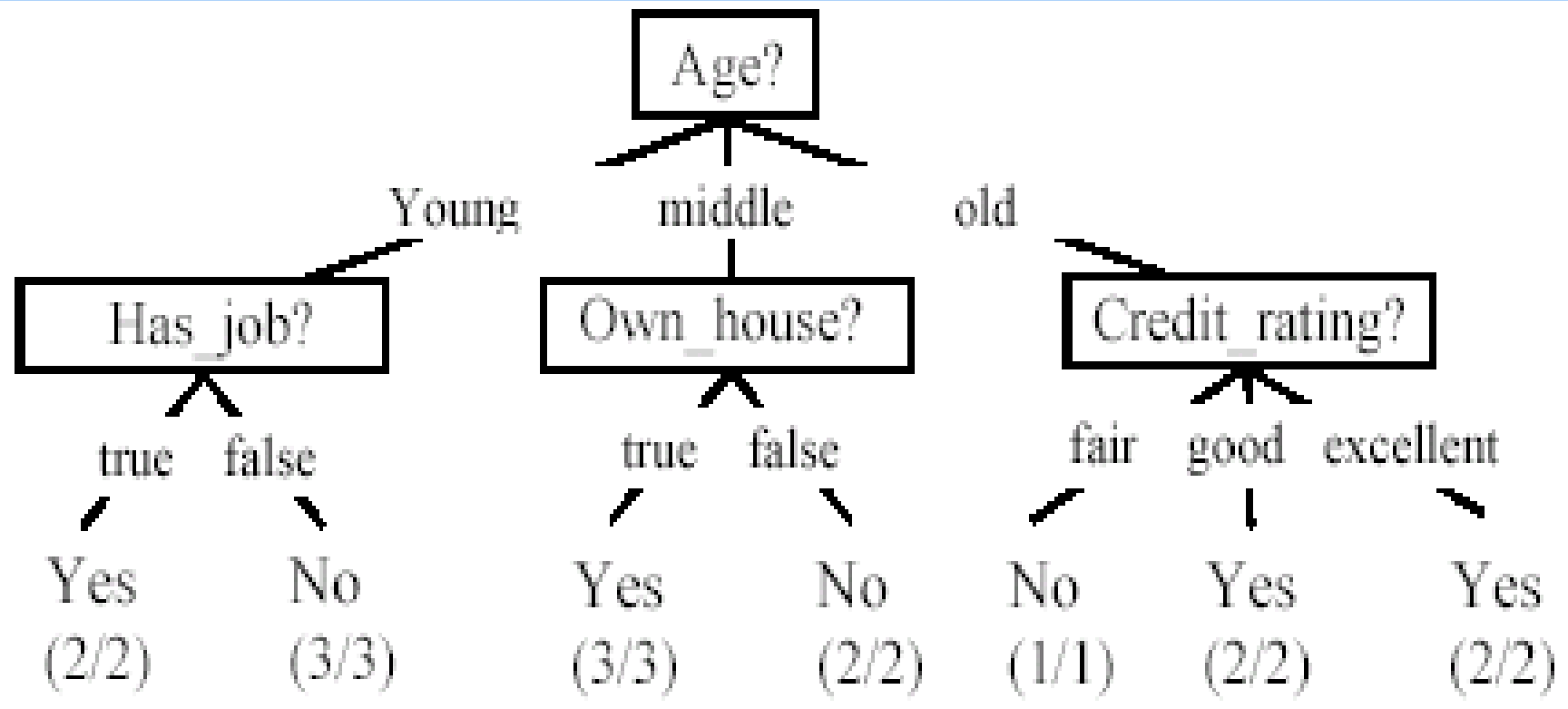
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

A decision tree from the loan data

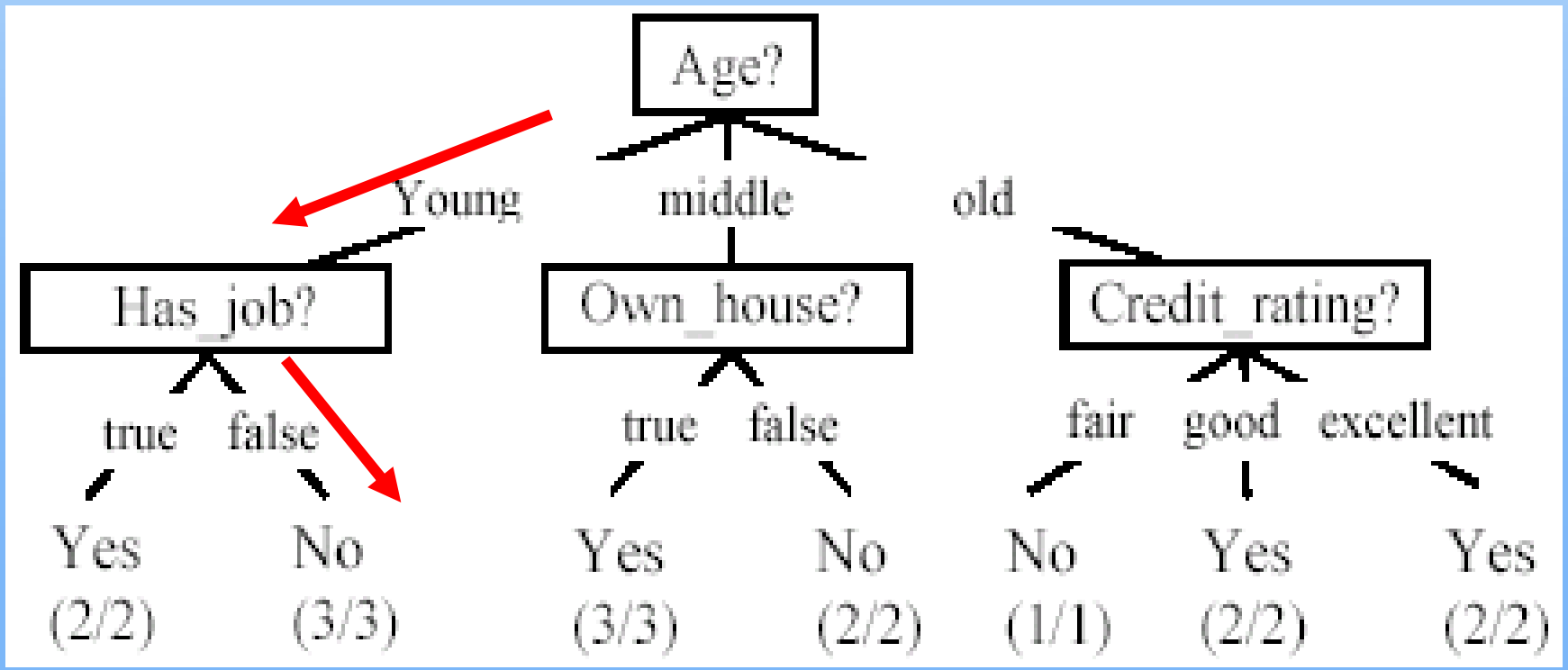
- Decision nodes and leaf nodes (classes)



Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

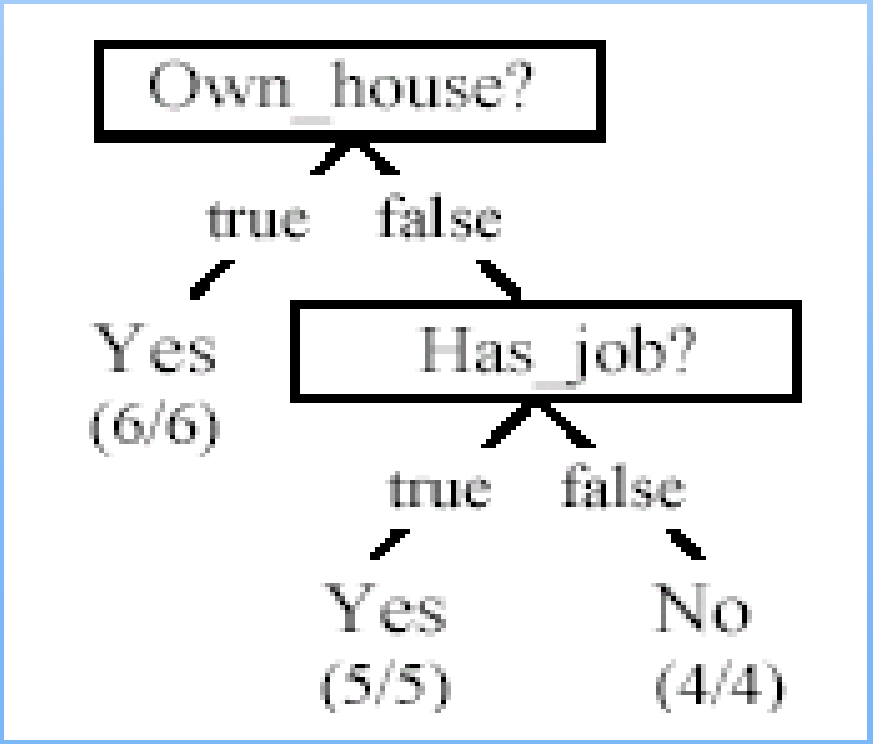
No



Is the decision tree unique?

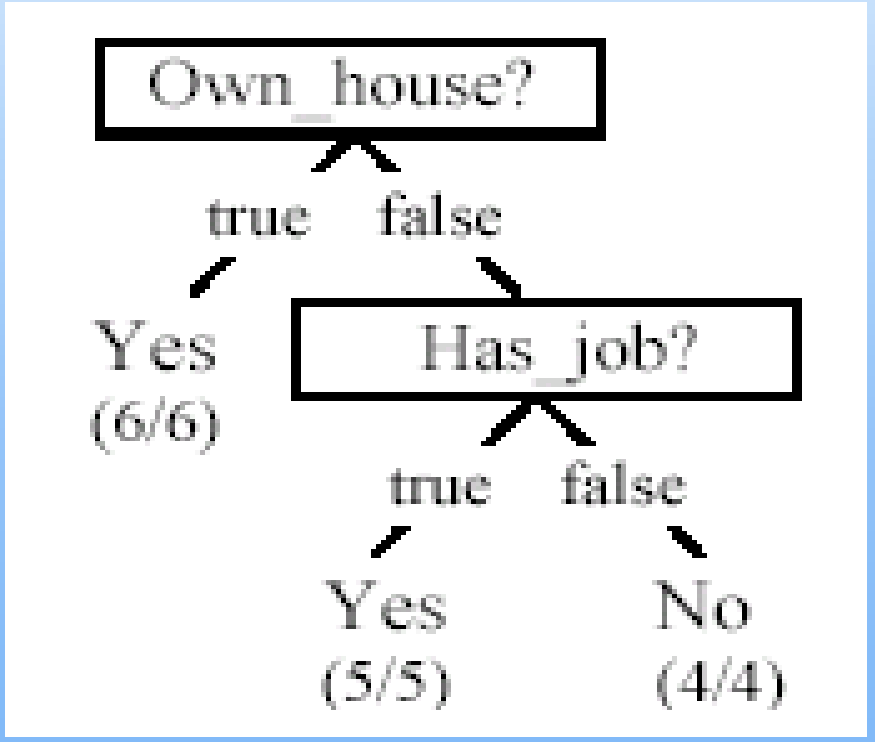
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



Own_house = true → Class =Yes [sup=6/15, conf=6/6]
Own_house = false, Has_job = true → Class = Yes [sup=5/15, conf=5/5]
Own_house = false, Has_job = false → Class = No [sup=4/15, conf=4/4]

Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

Decision tree learning algorithm

. **Algorithm** decisionTree(D, A, T)

```
1    if D contains only training examples of the same class c_j ∈ C then
2        make T a leaf node labeled with class c_j;
3    elseif A = ∅ then
4        make T a leaf node labeled with c_j, which is the most frequent class in D
5    else  // D contains examples belonging to a mixture of classes. We select a single
6          // attribute to partition D into subsets so that each subset is purer
7        p_0 = impurityEval-1(D);
8        for each attribute A_i ∈ {A_1, A_2, ..., A_k} do
9            p_i = impurityEval-2(A_i, D)
10       end
11       Select A_g ∈ {A_1, A_2, ..., A_k} that gives the biggest impurity reduction,
             computed using p_0 − p_i;
12       if p_0 − p_g < threshold then    // A_g does not significantly reduce impurity p_0
13         make T a leaf node labeled with c_j, the most frequent class in D.
14       else                             // A_g is able to reduce impurity p_0
15           Make T a decision node on A_g;
16           Let the possible values of A_g be v_1, v_2, ..., v_m. Partition D into m
                 disjoint subsets D_1, D_2, ..., D_m based on the m values of A_g.
17           for each D_j in {D_1, D_2, ..., D_m} do
18               if D_j ≠ ∅ then
19                   create a branch (edge) node T_j for v_j as a child node of T;
20                   decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
21               end
22           end
23       end
24   end
```

Choose an attribute to partition data

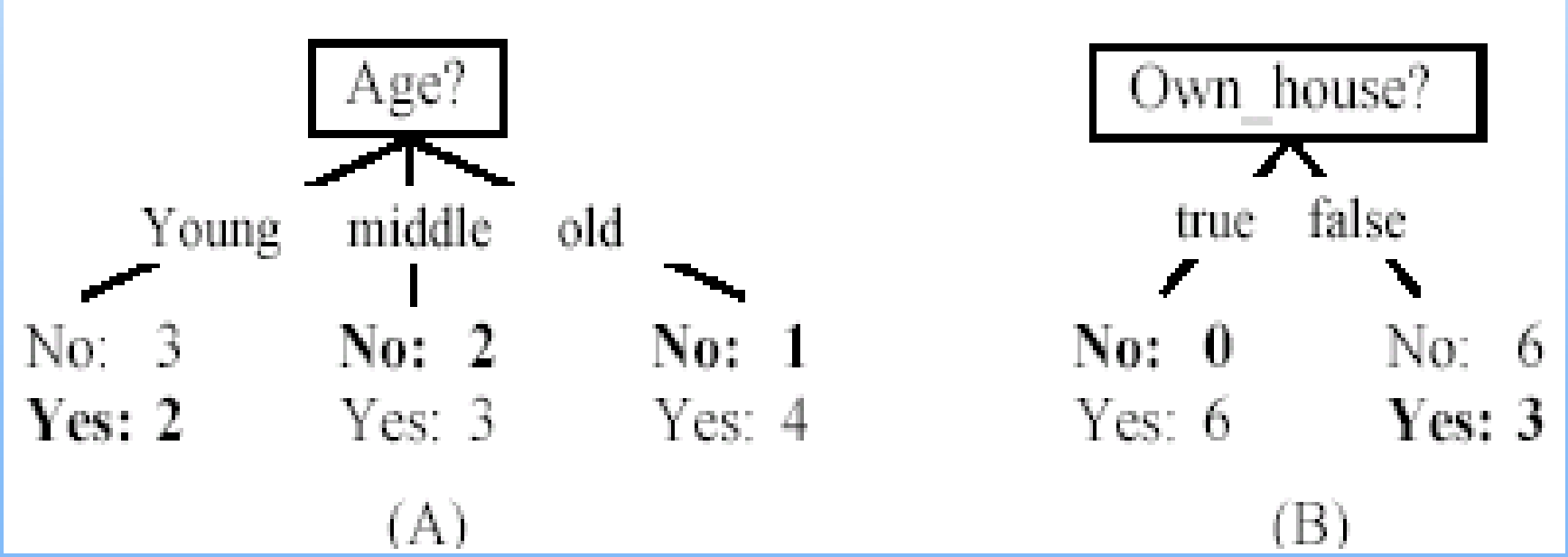
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum Information Gain or Gain Ratio based on information theory.

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

Two possible roots, which is better?



- Fig. (B) seems to be better.

Information theory

- Information theory provides a mathematical basis for measuring the information content.
- To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.
  - If one already has a good guess about the answer, then the actual answer is less informative.
  - If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

Information theory (cont …)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of $) for advanced information - less you know, the more valuable the information.
- Information theory uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

Information theory: Entropy measure

- The entropy formula,

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class c_j in data set D
- We use entropy as a measure of impurity or disorder of data set D. (Or, a measure of information in a tree)

Entropy measure: let us get a feeling

1. The data set D has 50% positive examples ($\Pr(positive) = 0.5$) and 50% negative examples ($\Pr(negative) = 0.5$).

$$entropy(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(positive) = 0.2$) and 80% negative examples ($\Pr(negative) = 0.8$).

$$entropy(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(positive) = 1$) and no negative examples, ($\Pr(negative) = 0$).

$$entropy(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

Information gain

- Given a set of examples D, we first compute its entropy:

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- If we make attribute A_i, with v values, the root of the current tree, this will partition D into v subsets $D_1, D_2 \ldots, D_v$. The expected entropy if A_i is used as the current root:

$$entropy_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j)$$

Information gain (cont …)

- Information gained by selecting attribute A_i to branch or to partition the data is

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

Tested all the calculations

An example

$$entropy(D) = \frac{6}{15} \times \log_2 \frac{6}{15} + \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

Entropy age -→refer manual cal →2/5

log(2/5) – 3/5 log(3/5)

$$entropy_{Own\_house}(D) = \frac{6}{15} \times entropy(D_1) + \frac{9}{15} \times entropy(D_2)$$

$$= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918$$

$$= 0.551$$

$$entropy_{Age}(D) = \frac{5}{15} \times entropy(D_1) + \frac{5}{15} \times entropy(D_2) + \frac{5}{15} \times entropy(D_3)$$

$$= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722$$

$$= 0.888$$

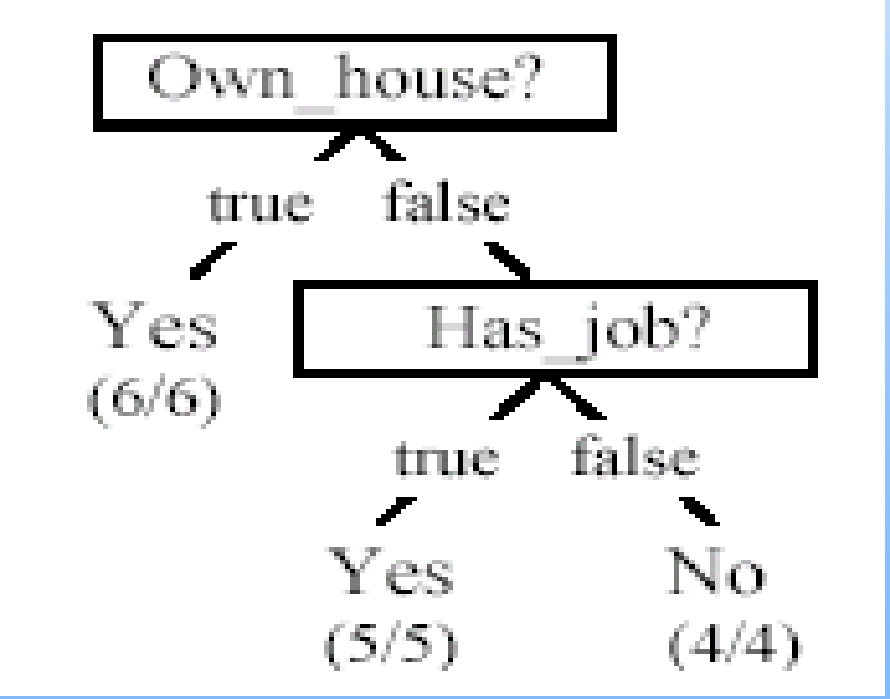
ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

	Age	Yes	No	entropy(Di)
5	young	2	3	0.971
5	middle	3	2	0.971
5	old	4	1	0.722

- Own_house is the best choice for the root.

$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$
$$gain(D, \text{Own\_house}) = 0.971 - 0.551 = 0.420$$
$$gain(D, \text{Has\_Job}) = 0.971 - 0.647 = 0.324$$
$$gain(D, \text{Credit\_Rating}) = 0.971 - 0.608 = 0.363$$

We build the final tree

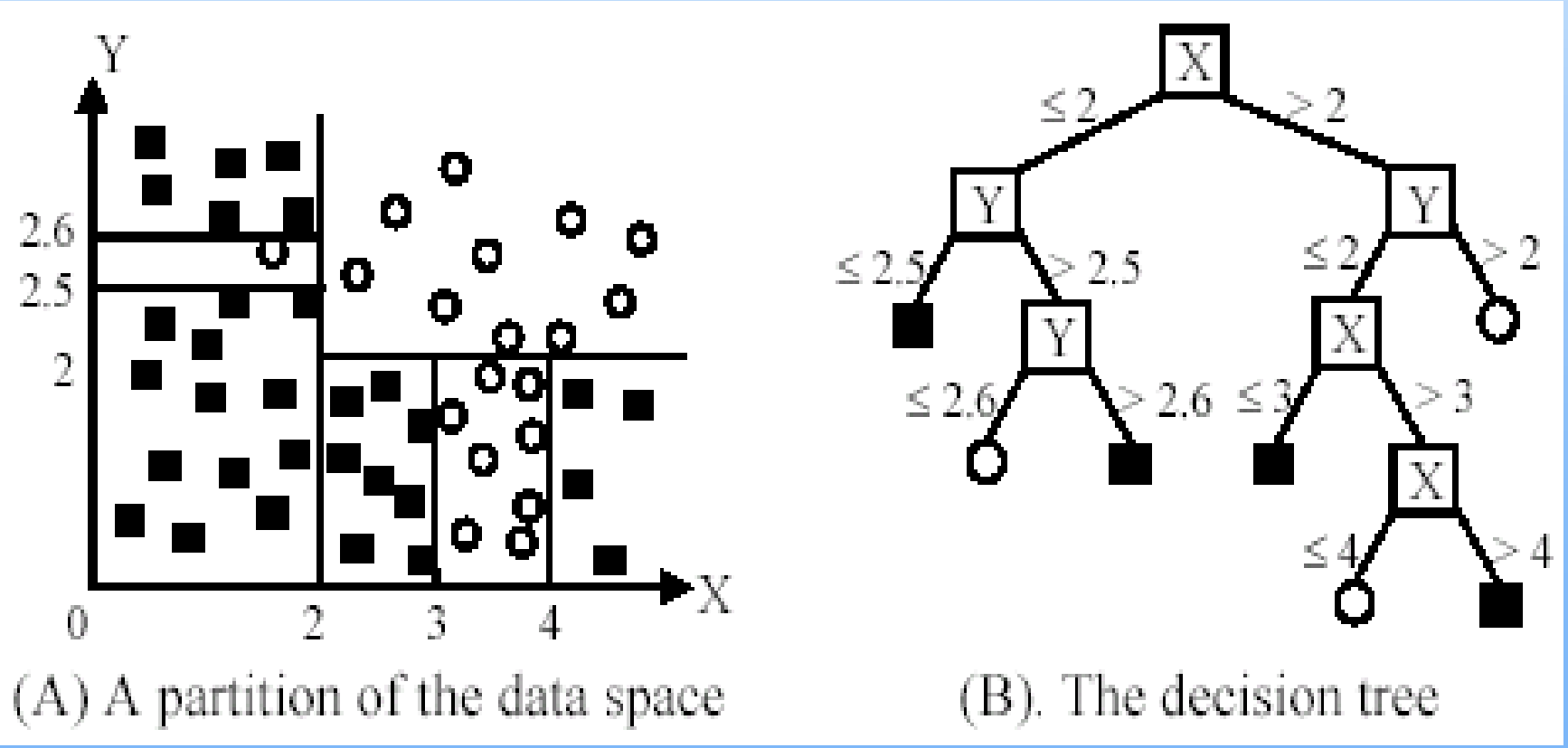


- We can use information gain ratio to evaluate the impurity as well (see the handout)

Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
  - Use information gain or gain ratio again
  - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \ldots, v_r\}$,
  - One possible threshold between two adjacent values v_i and v_{i+1}. Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

An example in a continuous space

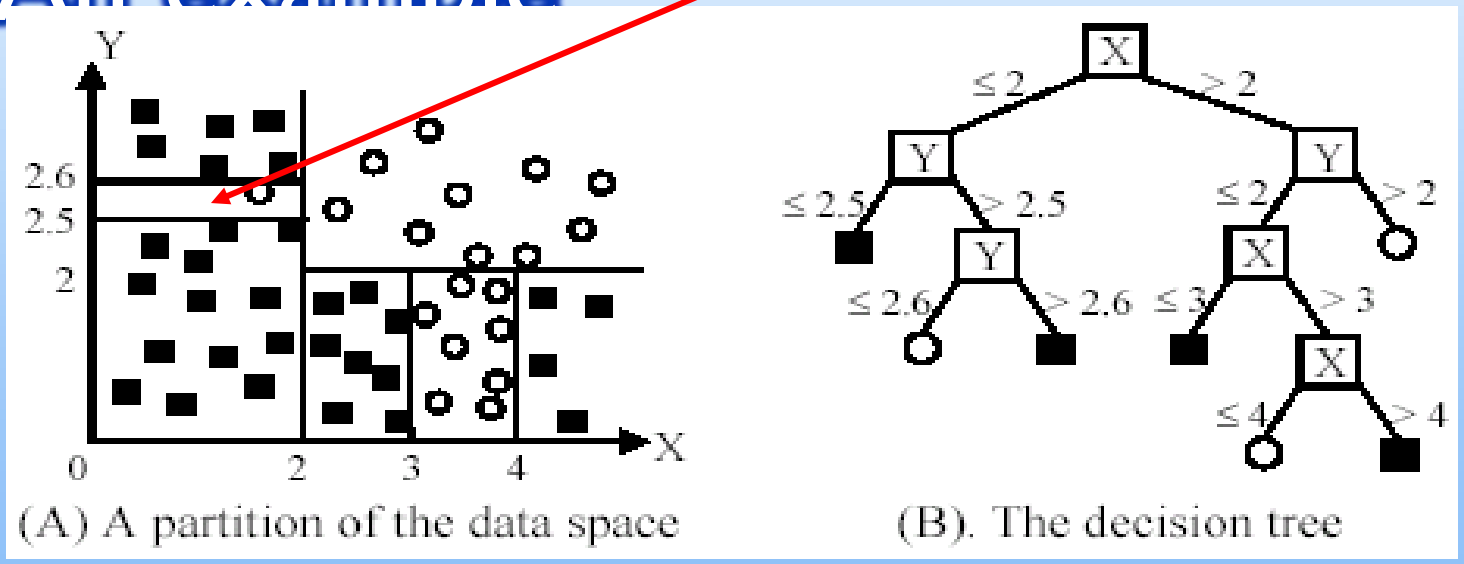


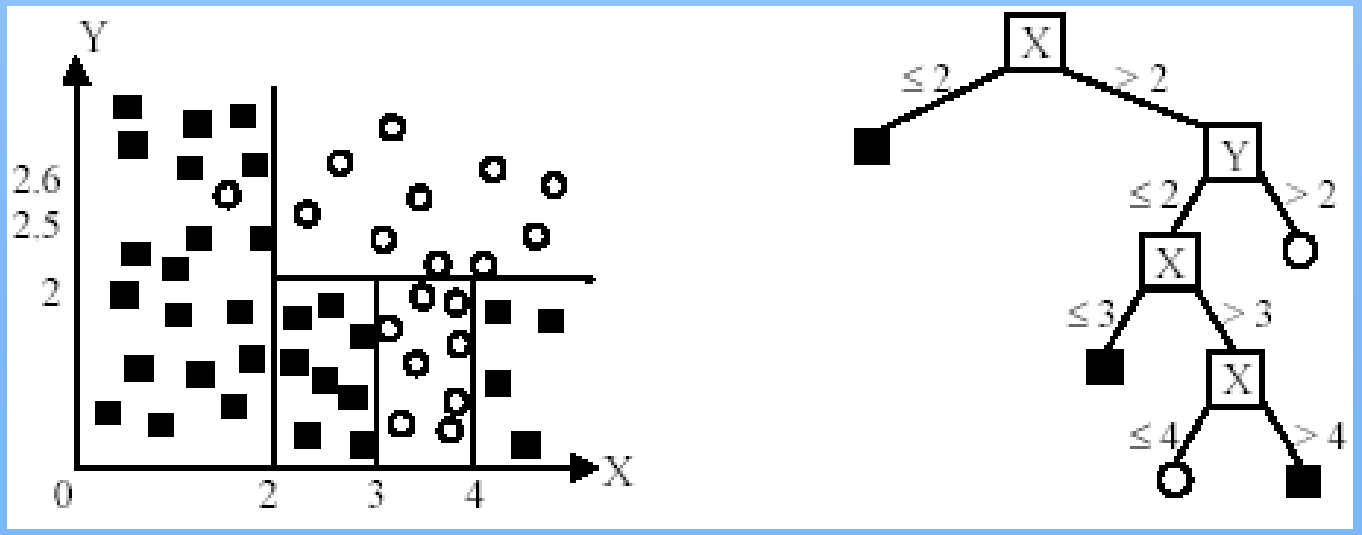
Avoid overfitting in classification

- Overfitting: A tree may overfit the training data
  - Good accuracy on training data but poor on test data
  - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
  - Pre-pruning: Halt tree construction early
    - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
  - Post-pruning: Remove branches or sub-trees from a "fully grown" tree.
    - This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.
    - A validation set may be used for pruning as well.

An example

Likely to overfit the data





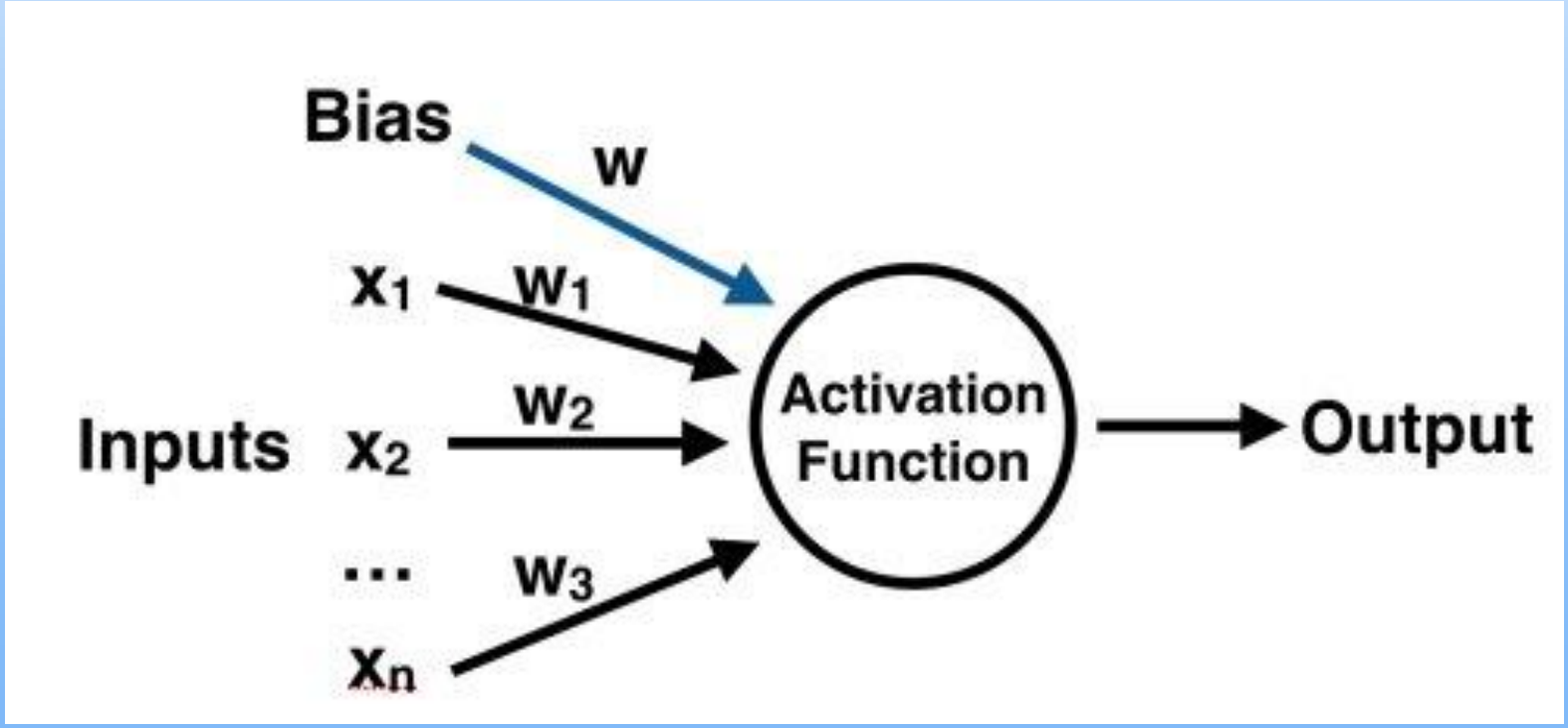
Other issues in decision tree learning

- From tree to rules, and rule pruning
- Handling of miss values
- Handing skewed distributions
- Handling attributes and classes with different costs.
- Attribute construction
- Etc.

Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

Neural Network Algorithm



A perceptron has one or more inputs, a bias, an activation function, and a single output. The perceptron receives inputs, multiplies them by some weight, and then passes them into an activation function to produce an output.

What is an activation function?

$$a = f\left(\sum_{i=0}^{N} w_i x_i\right)$$

The activation function is mostly used to make a non-linear transformation which allows us to fit nonlinear hypotheses or to estimate the complex functions.

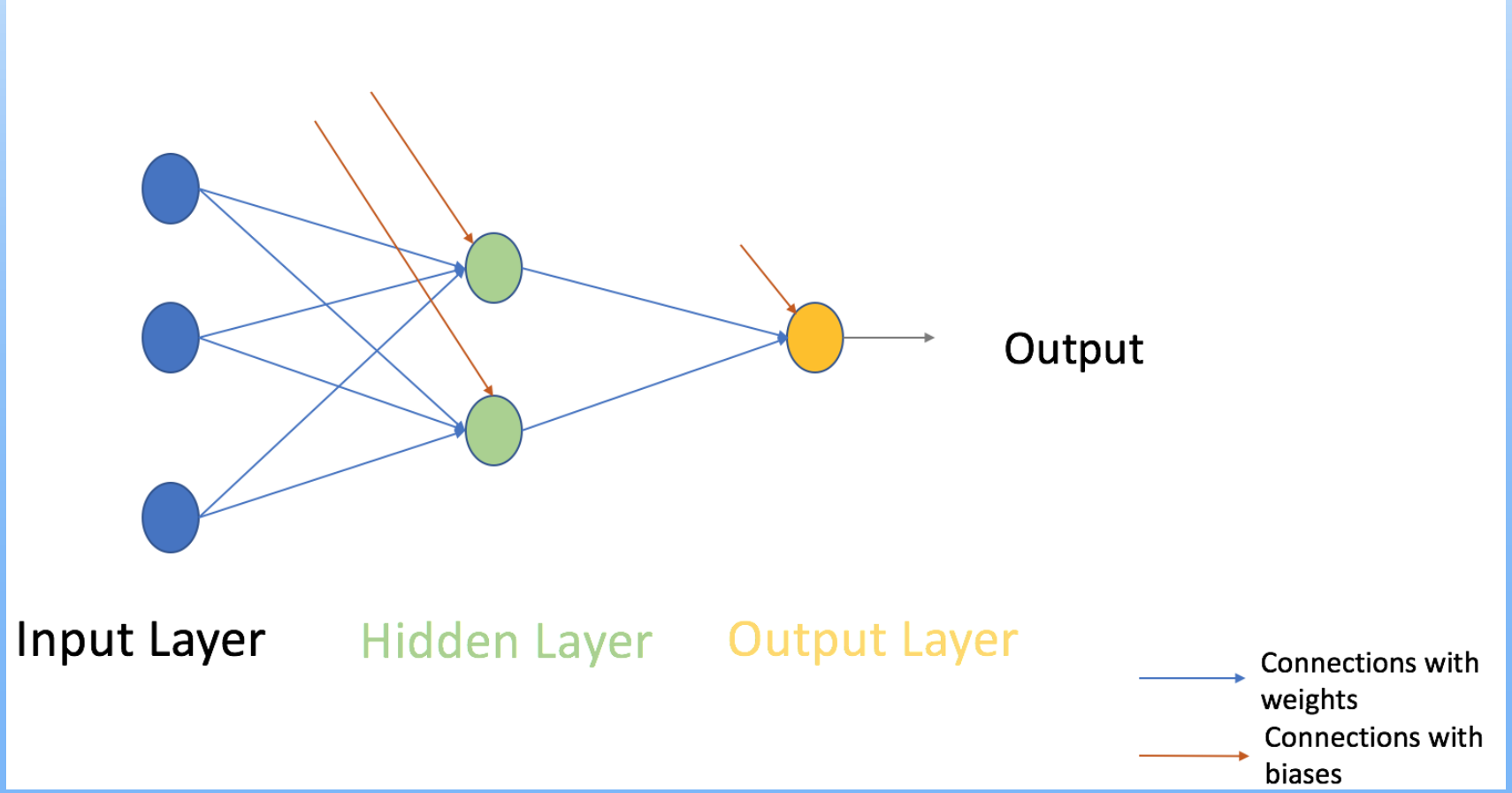
Forward Propagation, Back Propagation and Epochs

- Activation function's output process is known as "**Forward Propagation**". But what if the estimated output is far away from the actual output (high error).
- In the neural network what we do, we update the biases and weights based on the error. This weight and bias updating process is known as "**Back Propagation**".

Forward Propagation, Back Propagation and Epochs

- Back-propagation (BP) algorithms work by determining the loss (or error) at the output and then propagating it back into the network.
- The weights are updated to minimize the error resulting from each neuron.
- The first step in minimizing the error is to determine the gradient (Derivatives) of each node w.r.t. the final output.

Multi-layer perceptron



K-Nearest Neighbour Algorithm

Type	Acidity	Strength	Class
Type1 _tissue	7	7	Bad
Type2_tissue	7	4	Bad
Type3_tissue	3	4	Good
Type4_tissue	1	4	Good
Test Data	3	7	?

X1 = Acid Durability (seconds)	(kg/square meter)	Square Distance to query inst
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

K-Nearest Neighbour Algorithm

X1 = Acid Durability (seconds)	Strength (kg/square meter)	Square Distance to query instance (3, 7)	Rank minimum distance	Is it included in 3-Nearest neighbors?
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes
7	4	$(7-3)^2+(4-7)^2=25$	4	No
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes

Label it as Type_3 tissue closeset to rank 1

Random Forest

- Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model.
- The method of combining trees is known as an ensemble method.
- Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

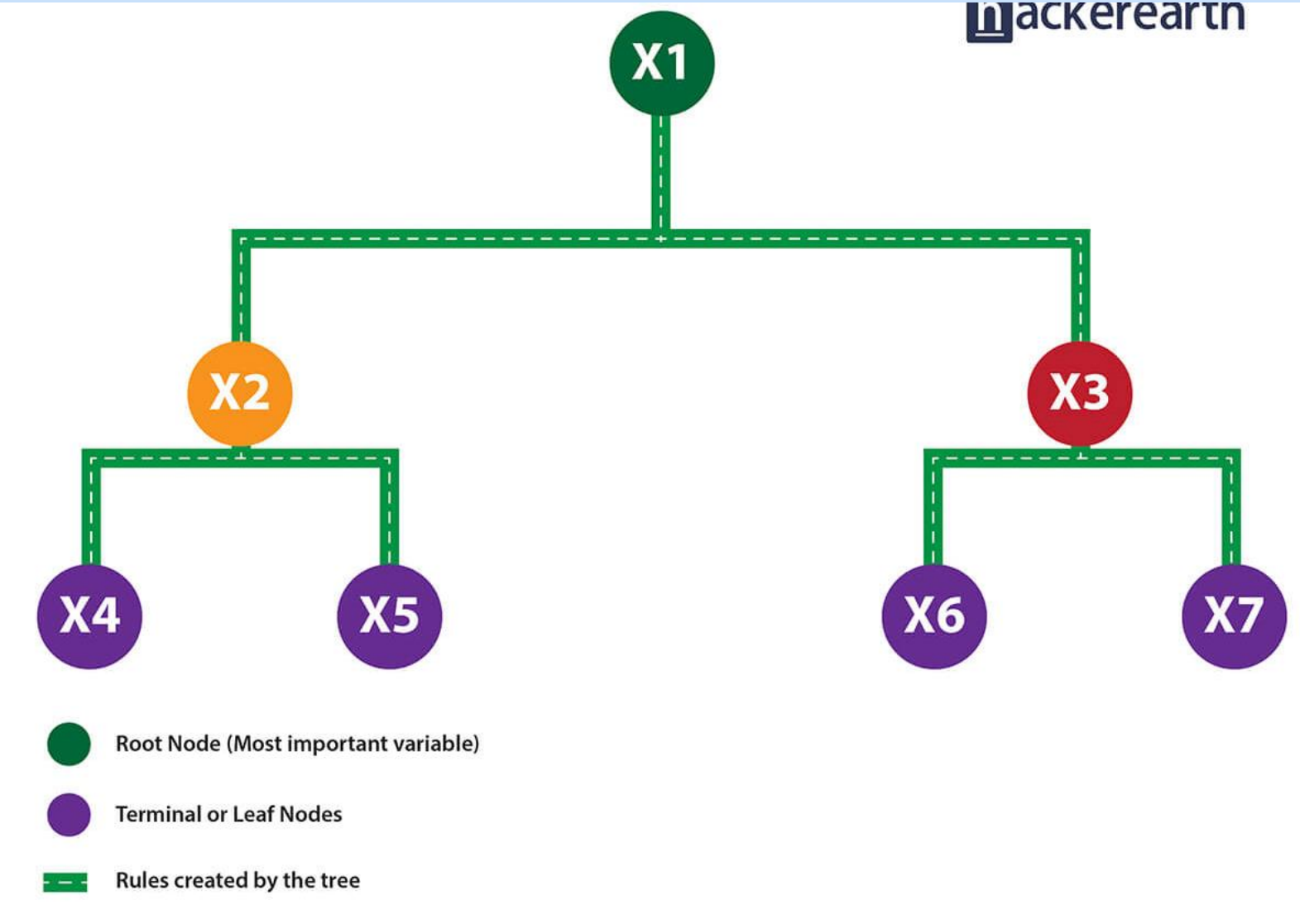
Random Forest

- Say, you want to watch a movie. But you are uncertain of its reviews.
- You ask 10 people who have watched the movie.
- 8 of them said " the movie is fantastic." Since the majority is in favor, you decide to watch the movie.
- This is how we use ensemble techniques in our daily life too.

Random Forest

- Random Forest can be used to solve regression and classification problems.
- In regression problems, the dependent variable is continuous.
- In classification problems, the dependent variable is categorical.

Random Forest



Random Forest Algorithm

- Step 1:Given a data frame (n x p), a tree stratifies or partitions the data based on rules (if-else).
  - Yes, a tree creates rules. These rules divide the data set into distinct and non-overlapping regions.
  - These rules are determined by a variable's contribution to the homogenity or pureness of the resultant child nodes (X2,X3).
- 2. In the image above, the variable X1 resulted in highest homogeneity in child nodes, hence it became the root node.
  - A variable at root node is also seen as the most important variable in the data set.

Random Forest Algorithm

- Step 3. But how is this homogeneity or pureness determined? In other words, how does the tree decide at which variable to split?

Random Forest Algorithm

- In **regression trees** (where the output is predicted using the mean of observations in the terminal nodes), the splitting decision is based on minimizing RSS.
- The variable which leads to the greatest possible reduction in RSS is chosen as the root node.
- The tree splitting takes a **top-down greedy** approach, also known as *recursive binary splitting*.
- We call it "greedy" because the algorithm cares to make the best split at the current step rather than saving a split for better results on future nodes.

Random Foresto

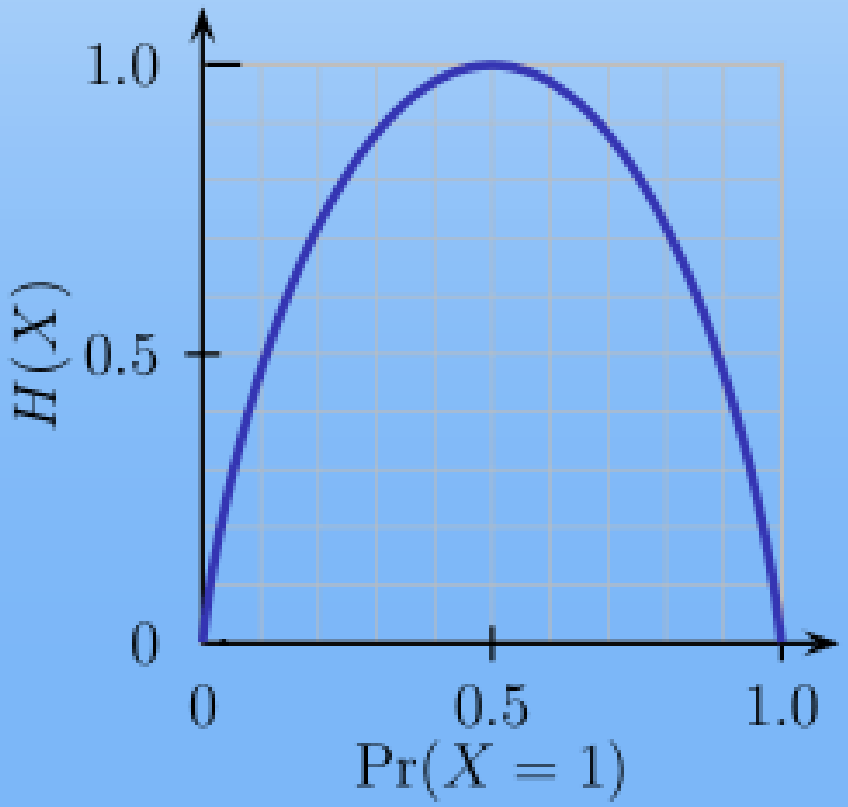
- In **classification trees** (where the output is predicted using mode of observations in the terminal nodes), the splitting decision is based on the following methods:
- **Gini Index** - It's a measure of node purity. If the Gini index takes on a smaller value, it suggests that the node is pure.
- For a split to take place, the Gini index for a child node should be less than that for the parent node.

Random Forest

- **Entropy** - Entropy is a measure of node impurity.
- For a binary class (a,b), the formula to calculate it is shown below. Entropy is maximum at p = 0.5. For p(X=a)=0.5 or p(X=b)=0.5 means, a new observation has a 50%-50% chance of getting classified in either classes.
- The entropy is minimum when the probability is 0 or 1.

Random Forest

- Entropy = - p(a)*log(p(a)) - p(b)*log(p(b))



Random Forest

- In a nutshell, every tree attempts to create rules in such a way that the resultant terminal nodes could be as pure as possible.
- Higher the purity, lesser the uncertainity to make the decision.

Random Forest

- But a decision tree suffers from high variance.
- "High Variance" means getting high prediction error on unseen data.
- We can overcome the variance problem by using more data for training.
- But since the data set available is limited to us, we can use resampling techniques like bagging and random forest to generate more data.

Random Forest

- Building many **decision trees** results in a **forest**. A random forest works the following way:
- First, it uses the Bagging (Bootstrap Aggregating) algorithm to create random samples.
- Given a data set D1 (n rows and p columns), it creates a new dataset (D2) by sampling n cases at random with replacement from the original data.
- About 1/3 of the rows from D1 are left out, known as Out of Bag(OOB) samples.

Random Forest

- Then, the model trains on D2. OOB sample is used to determine unbiased estimate of the error.
- Out of p columns, P << p columns are selected at each node in the data set. The P columns are selected at random. Usually, the default choice of P is p/3 for regression tree and P is sqrt(p) for classification tree.

What is Boosting?

- The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners.
- How would you classify an email as SPAM or not?
- our initial approach would be to identify 'spam' and 'not spam' emails using following criteria. If:
- Email has only one image file (promotional image), It's a SPAM
- Email has only link(s), It's a SPAM
- Email body consist of sentence like "You won a prize money of $ xxxxxx", It's a SPAM
- Email from our official domain "rps.com" , Not a SPAM
- Email from known source, Not a SPAM

Random Forest

- Unlike a tree, no pruning takes place in random forest; i.e, each tree is grown fully. I
- In decision trees, pruning is a method to avoid overfitting.
- Pruning means selecting a subtree that leads to the lowest test errror rate.
- We can use cross validation to determine the test error rate of a subtree.
- Several trees are grown and the final prediction is obtained by averaging or voting.

Random Forest

- Each tree is grown on a different sample of original data.
- Since random forest has the feature to calculate OOB error internally, cross validation doesn't make much sense in random forest.

What is the difference between Bagging and Random Forest?

- It creates randomized samples of the data set (just like random forest) and grows trees on a different sample of the original data.
- The remaining 1/3 of the sample is used to estimate unbiased OOB error.
- It considers all the features at a node (for splitting).
- Once the trees are fully grown, it uses averaging or voting to combine the resultant predictions.

Random Forest

- The need for random forest surfaced after discovering that the bagging algorithm results in correlated trees when faced with a data set having strong predictors.
- Unfortunately, averaging several highly correlated trees doesn't lead to a large reduction in variance.

Random Forest

- The **main difference between** random forest and bagging is that random forest considers only a subset of predictors at a split.
- This results in trees with different predictors at top split, thereby resulting in **decorrelated trees** and more reliable average output.

Adavntages and Disadvantages

- t is robust to correlated predictors.
- It is used to solve both regression and classification problems.
- It can be also used to solve unsupervised ML problems.
- It can handle thousands of input variables without variable selection.
- It can be used as a feature selection tool using its variable importance plot.
- It takes care of missing data internally in an effective manner.

Random Forest Disadvantages

- The Random Forest model is difficult to interpret.
- It tends to return erratic predictions for observations out of range of training data. For example, the training data contains two variable x and y. The range of x variable is 30 to 70. If the test data has x = 200, random forest would give an unreliable prediction.
- It can take longer than expected time to computer a large number of trees.

Boosting Algorithm

- Individually, these rules are not powerful enough to classify an email into 'spam' or 'not spam'. Therefore, these rules are called as **weak learner**.
- To convert weak learner to strong learner, combine the prediction of each weak learner using methods like:
  - Using average/ weighted average
  - Considering prediction has higher vote

Boosting Algorithm

- For example: Above, we have defined 5 weak learners. Out of these 5, 3 are voted as 'SPAM' and 2 are voted as 'Not a SPAM'. In this case, by default, we'll consider an email as SPAM because we have higher(3) vote for 'SPAM'.

How Boosting Algorithms works?

- *Step 1:* The base learner takes all the distributions and assign equal weight or attention to each observation.
- *Step 2:* If there is any prediction error caused by first base learning algorithm, then we pay higher attention to observations having prediction error. Then, we apply the next base learning algorithm.
- *Step 3:* Iterate Step 2 till the limit of base learning algorithm is reached or higher accuracy is achieved.

Types of Boosting Algorithms

- AdaBoost (**Ada**ptive **Boost**ing)
- Gradient Tree Boosting
- XGBoost

Bagging (Bootstrap aggregating)

- Take M bootstrap samples (with replacement)
- Train M different classifiers on these bootstrap samples
- For a new query, let all classifiers predict and take an average (or majority vote)

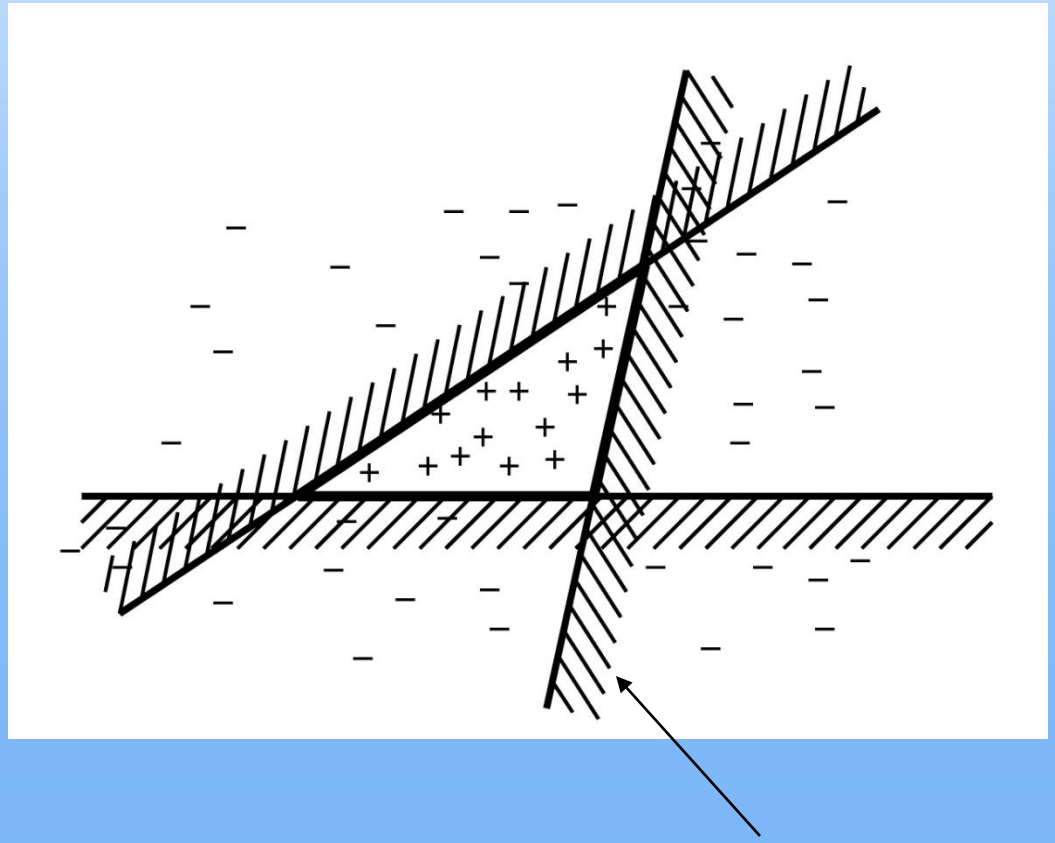
- If the classifiers make independent errors, then their ensemble can improve performance.

- Stated differently: the variance in the prediction is reduced (we don't suffer from the random errors that a single classifier is bound to make).

Boosting

- Train classifiers (e.g. decision trees) in a sequence.
- A new classifier should focus on those cases which were incorrectly classified in the last round.
- Combine the classifiers by letting them vote on the final prediction (like bagging).
- Each classifier is "weak" but the ensemble is "strong."
- AdaBoost is a specific boosting method.

Example

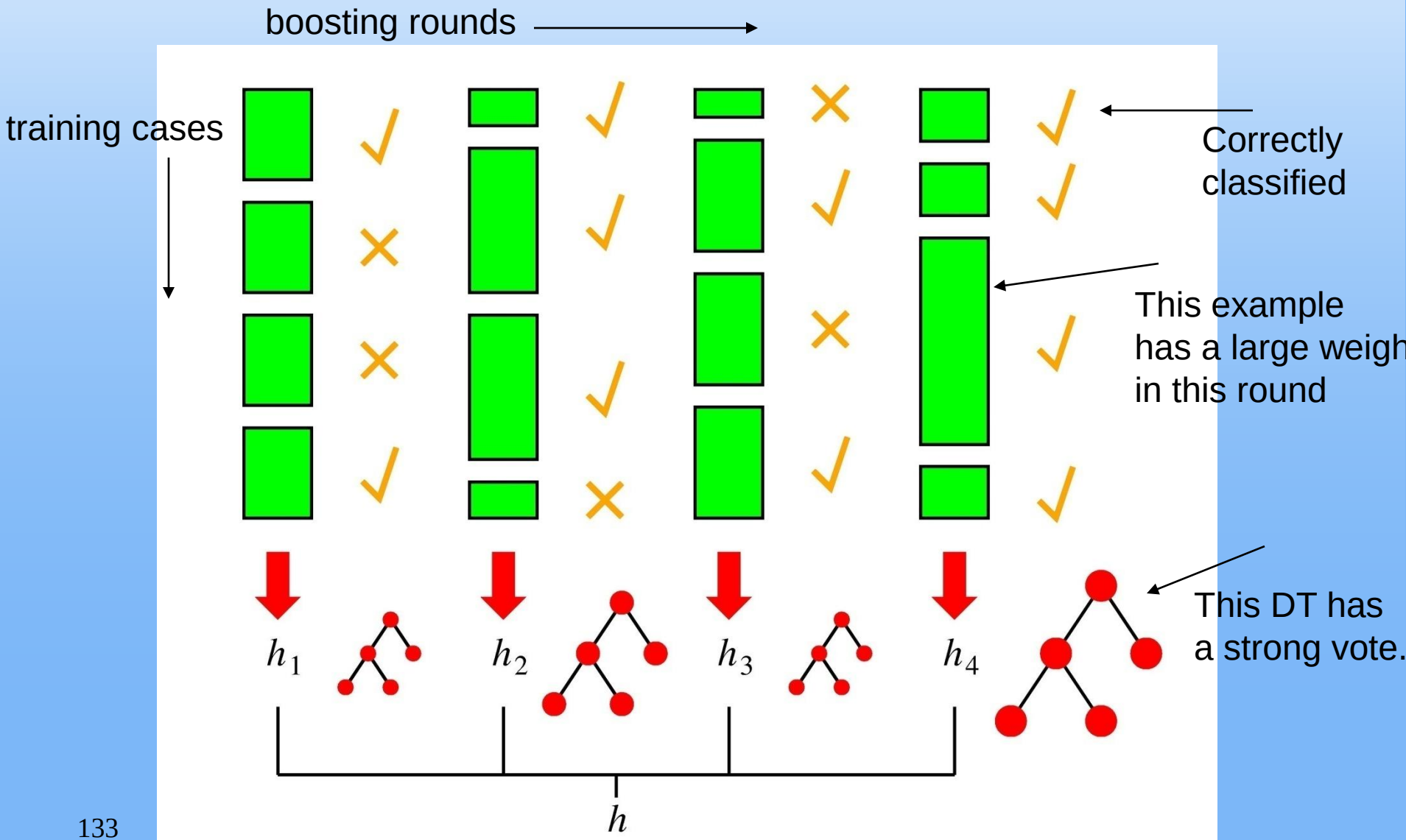


This line is one simple classifier saying that everything to the left + and everything to the right is -

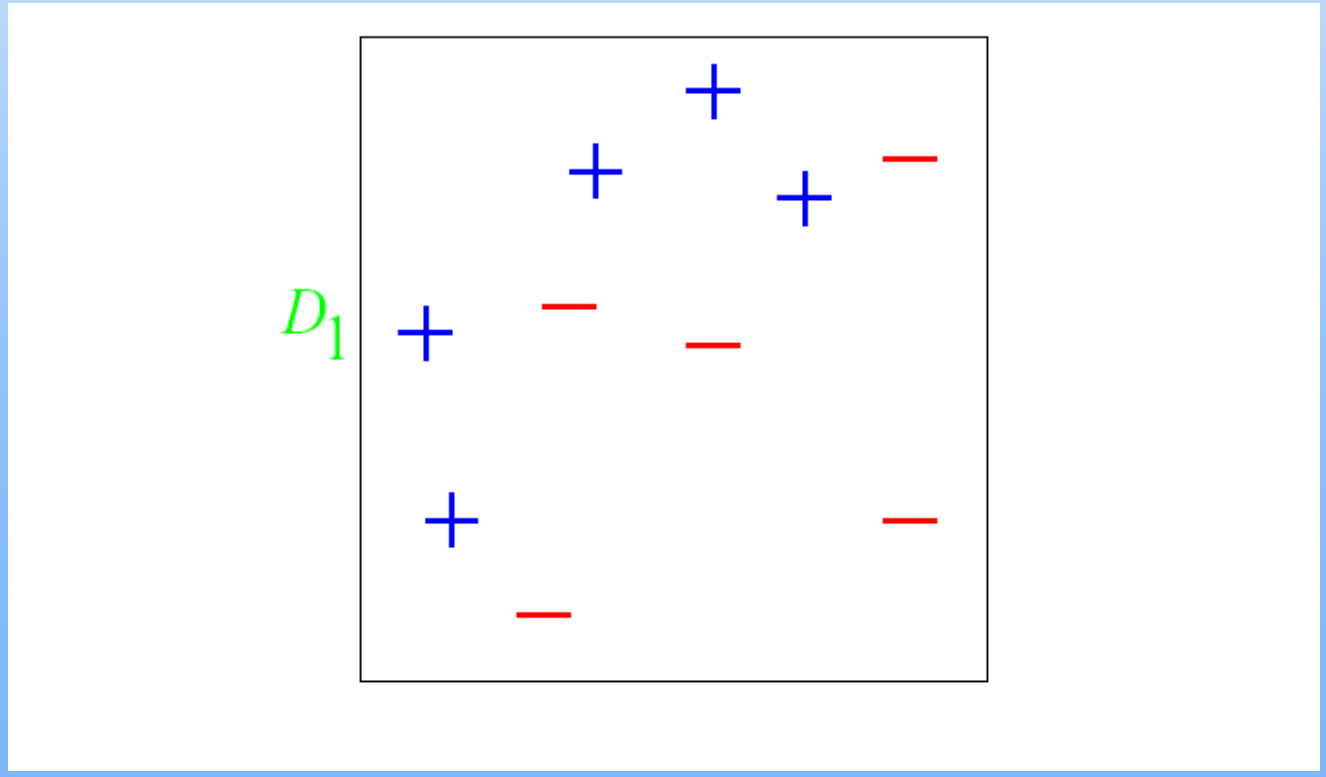
Boosting Intuition

- We adaptively weigh each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that that obtain low training error rate have high weight.
- We stop by using monitoring a hold out set (cross-validation).

Boosting in a Picture



And in animation



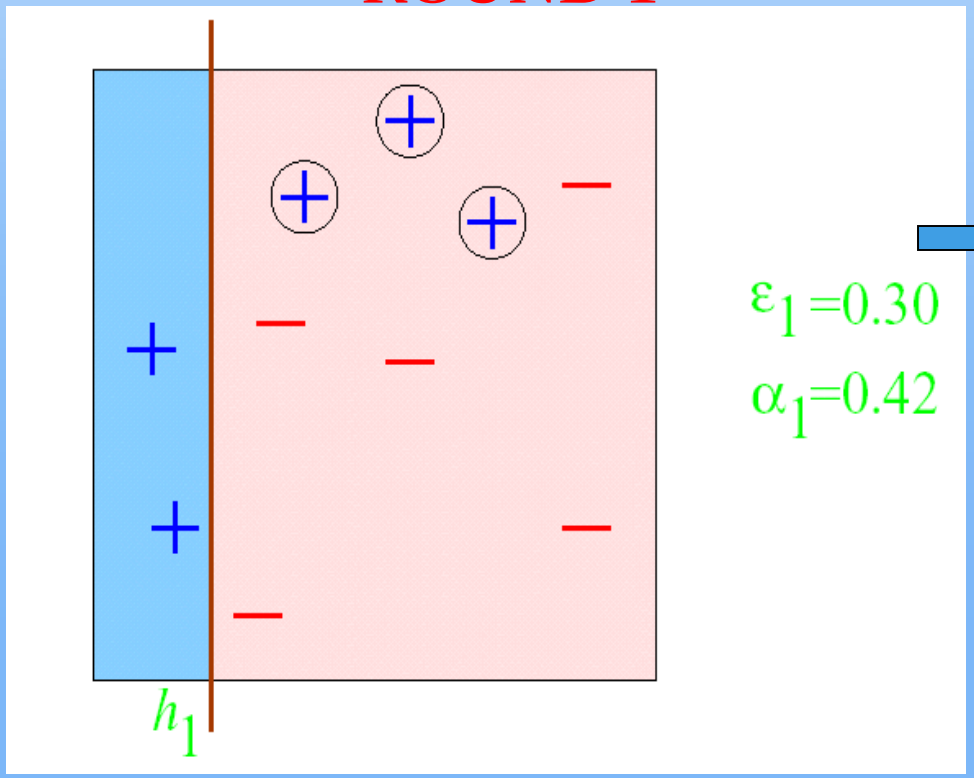
Original training set: equal weights to all training samples

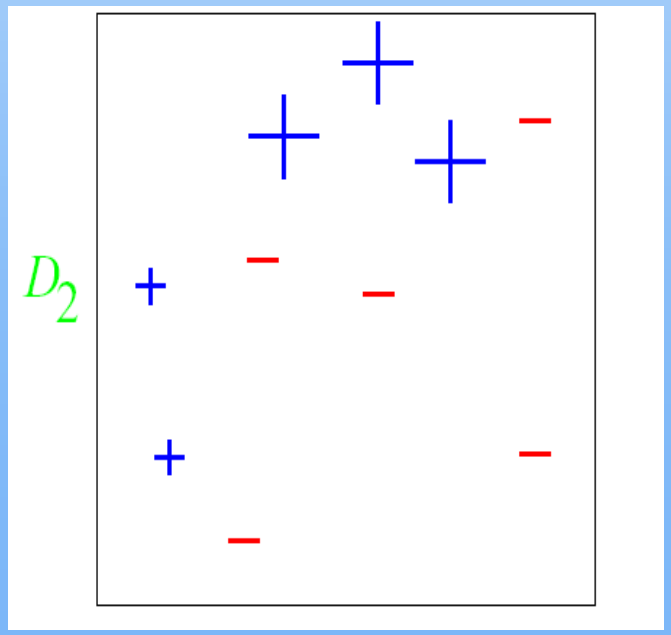
Taken from "**A Tutorial on Boosting**" **by** Yoav Freund and Rob Schapire

AdaBoost example

ε = error rate of classifier
α = weight of classifier

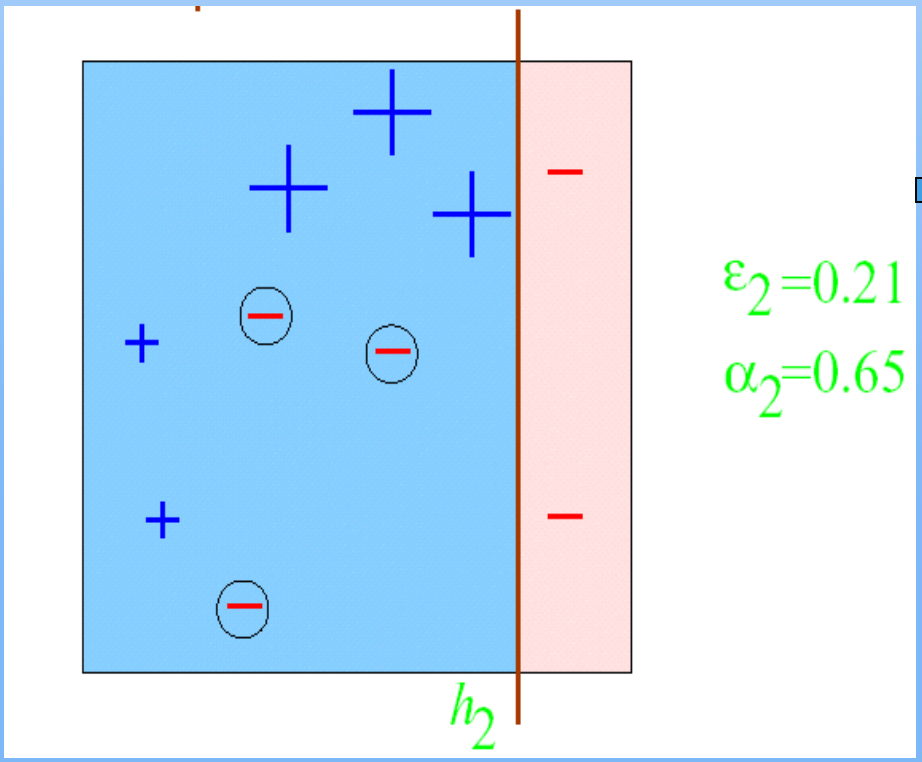
ROUND 1





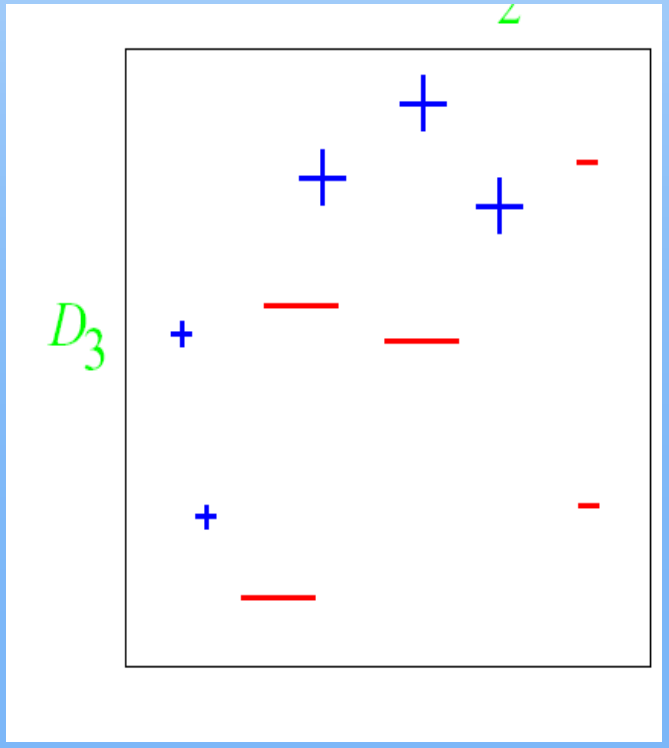
AdaBoost example

ROUND 2



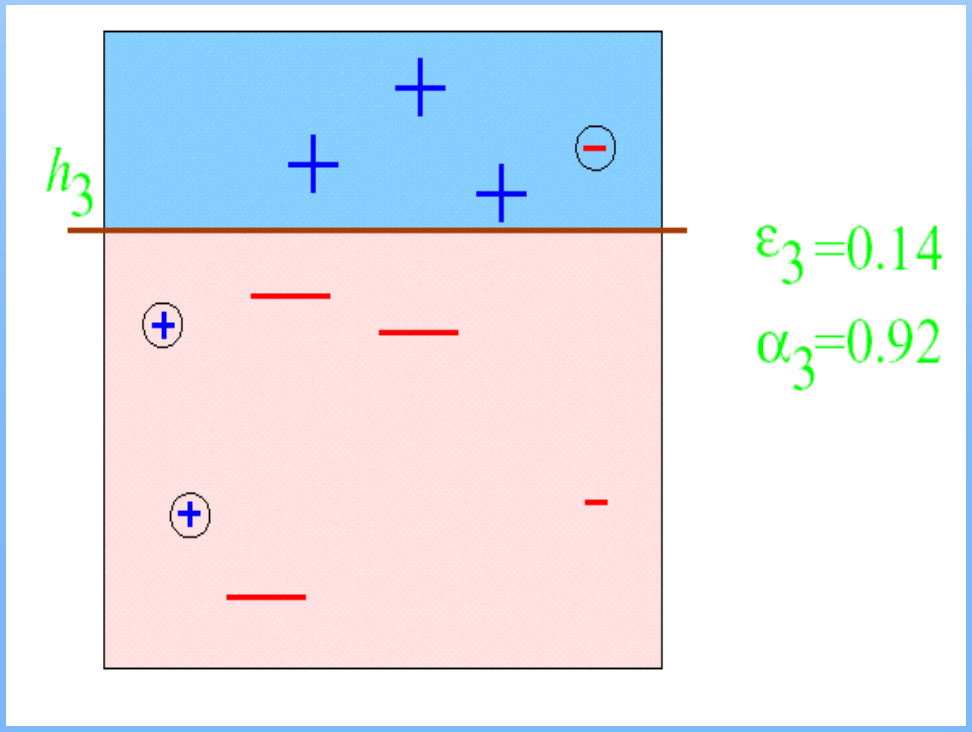
$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

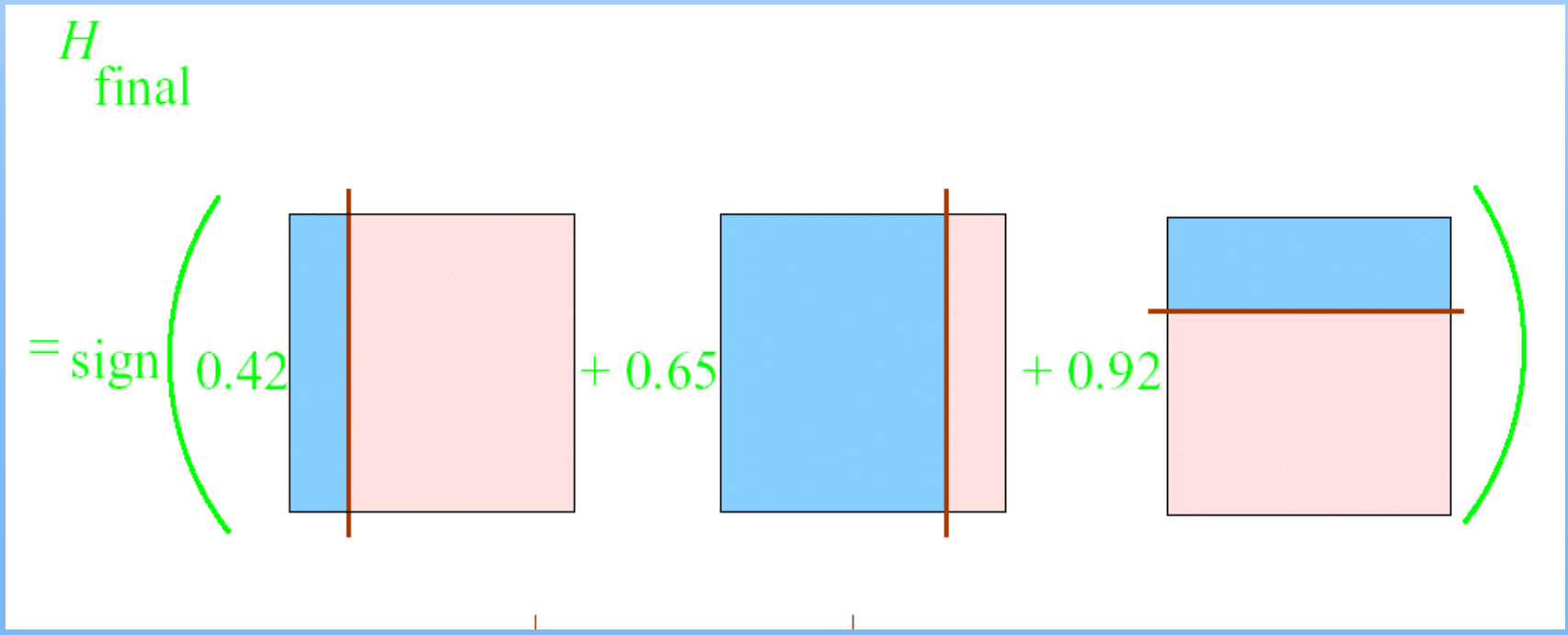


AdaBoost example

ROUND 3



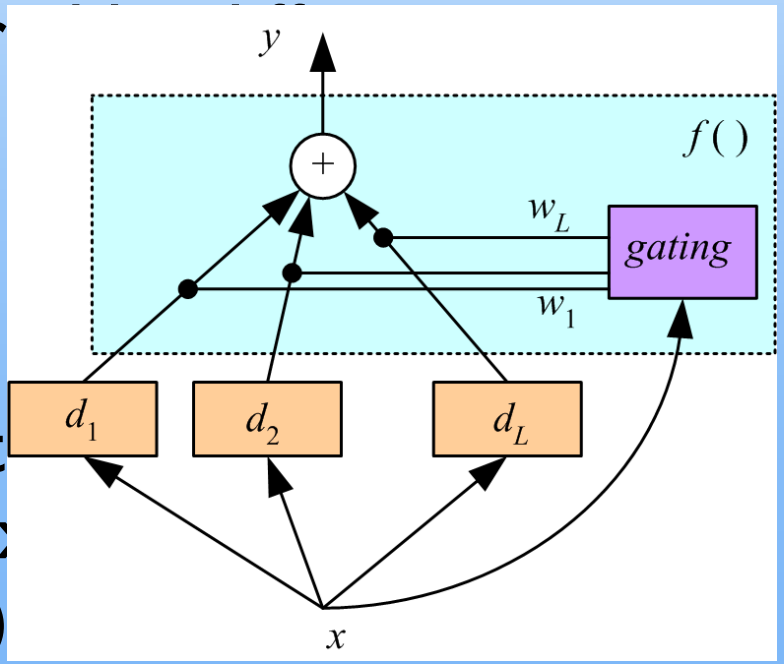
AdaBoost example



Mixture of experts

- Voting where weights are input-dependent (gating)
- Different input regions conver
learners (Jacobs et al., 1991)

$$y = \sum_{j=1}^{L} w_j d_j$$



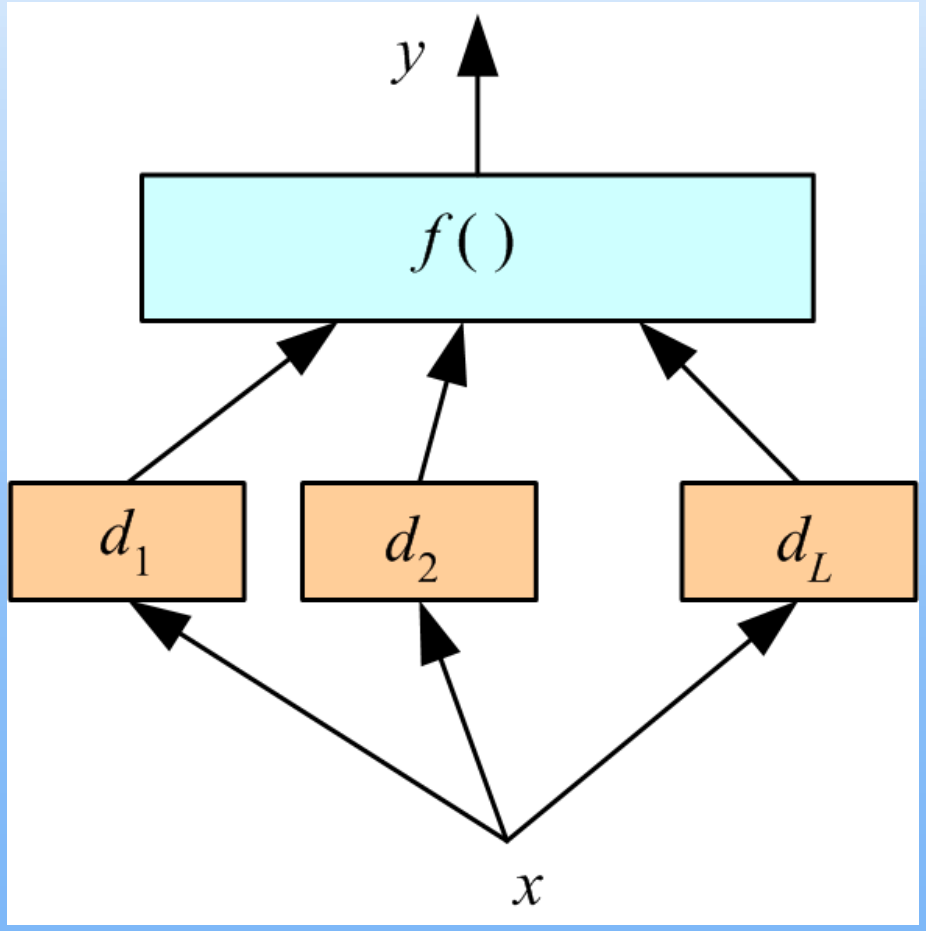
- Gating decides which expert t
- Need to learn the individual e

well as the gating functions $w_i(x)$

$\Sigma w_j(x) = 1$, for all x

Stacking

- Combiner f () is another learner (Wolpert, 1992)



AdaBoost Concept

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

weak classifiers

strong classifier

slightly better than random

Weaker Classifiers

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

weak classifiers

slightly better than random

- Each weak classifier learns by considering one simple feature
- T most beneficial features for classification should be selected

- How to
  - define features?
  - select beneficial features?
  - train weak classifiers?
  - manage (weight) training samples?
  - associate weight to each weak classifier?

The Strong Classifiers

How good the strong one will be?

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

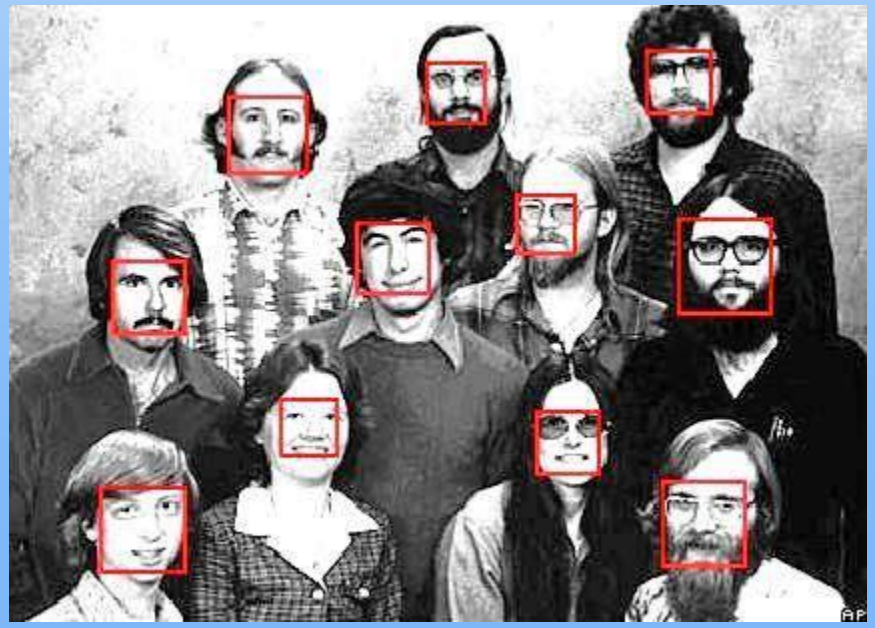
$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

weak classifiers

strong classifier

slightly better than random

The Task of Face Detection



Many slides adapted from P. Viola

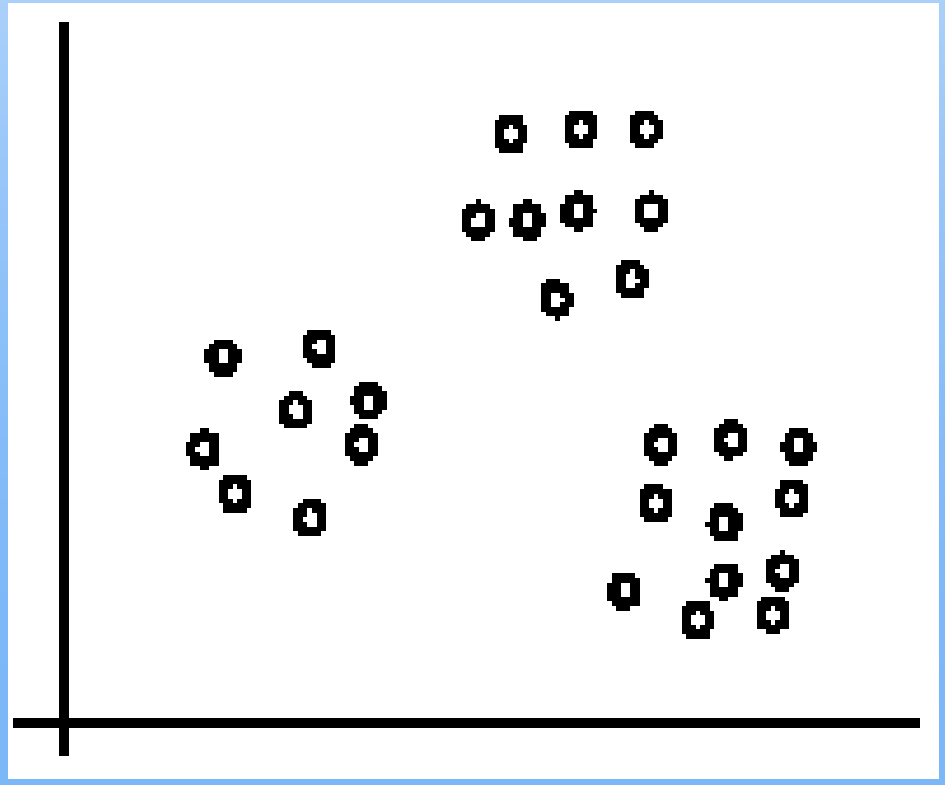
Unsupervised Learning

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, association rule mining is also unsupervised

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make "small", "medium" and "large" T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

Types of Clustering

- Hard Clustering
  - In hard clustering, each data point either belongs to a cluster completely or not.
- Soft Clustering
  - In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example each costumer is assigned a probability to be in either of 10 clusters of the retail store.

Types of clustering algorithms

- **Connectivity models:**
- **Centroid models:**
- **Distribution models:**
- **Density Models:**

Connectivity Models

- These models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away.
- These models can follow two approaches.
  - In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases.
  - In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective.
  - These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

Centroid models

- These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters.
- K-Means clustering algorithm is a popular algorithm that falls into this category.
- In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset.
- These models run iteratively to find the local optima.

Distribution models

- These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian).
- These models often suffer from overfitting.
- A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.

Density Models:

- These models search the data space for areas of varied density of data points in the data space.
- It isolates various different density regions and assign the data points within these regions in the same cluster.
- Popular examples of density models are DBSCAN and OPTICS.

Aspects of clustering

- A clustering algorithm
  - Partitional clustering
  - Hierarchical clustering
  - …
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance ⇒ maximized
  - Intra-clusters distance ⇒ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) D be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1) Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
  2) Assign each data point to the closest centroid
  3) Re-compute the centroids using the current cluster memberships.
  4) If a convergence criterion is not met, go to 2).

K-means algorithm – (cont …)

```
   Algorithm k-means(k, D)
1    Choose k data points as the initial centroids (cluster centers)
2    repeat
3        for each data point x ∈ D do
4            compute the distance from x to each centroid;
5            assign x to the closest centroid          // a centroid represents a cluster
6        endfor
7        re-compute the centroids using the current cluster memberships
8    until the stopping criterion is met
```

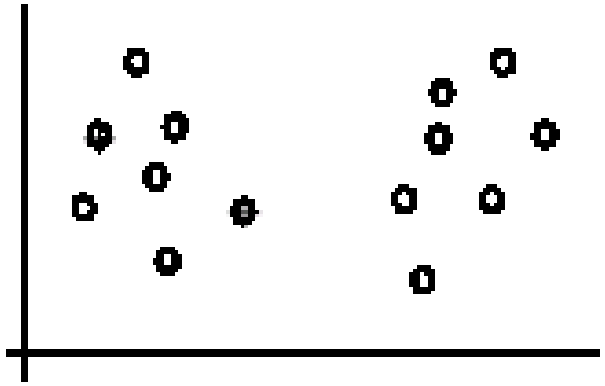
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

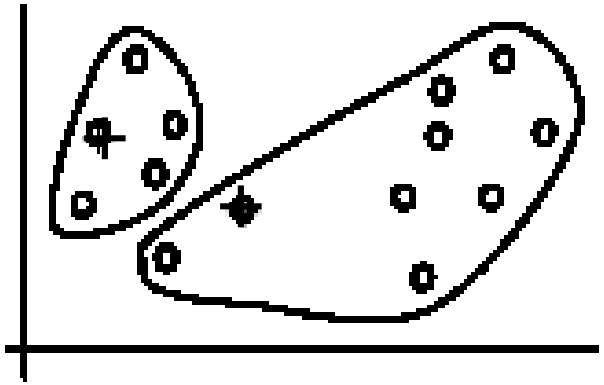
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \qquad (1)$$

- C_j is the jth cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

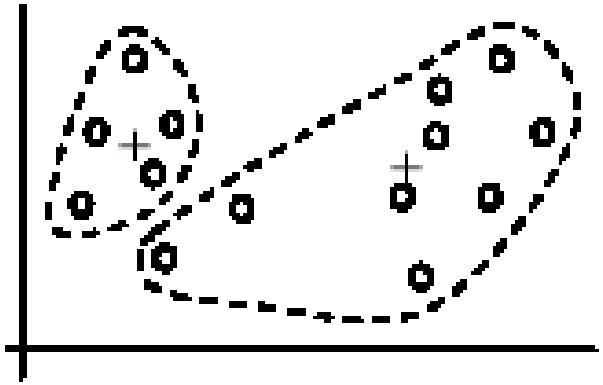
An example



(A). Random selection of k centers

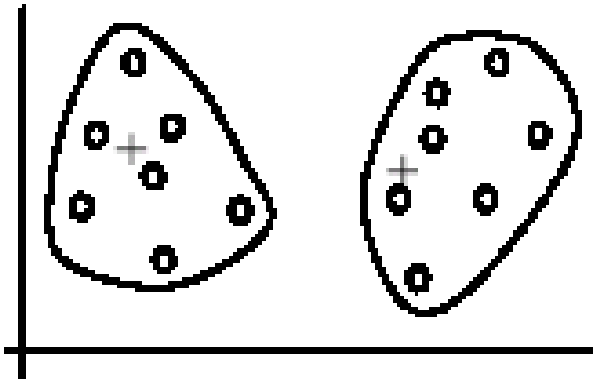


Iteration 1: (B). Cluster assignment

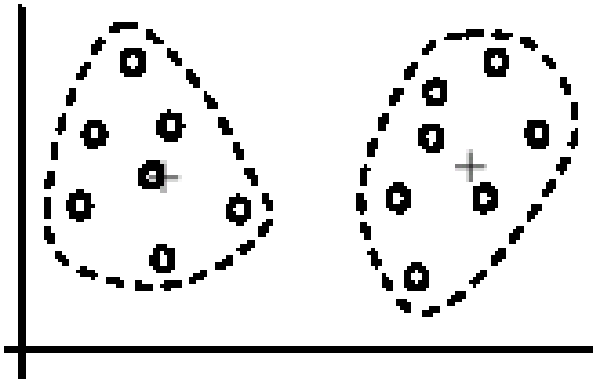


(C). Re-compute centroids

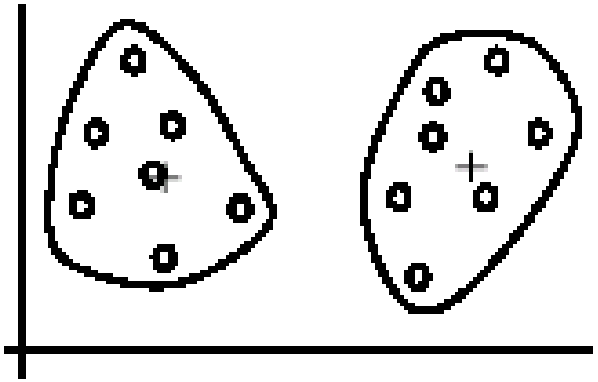
An example (cont …)



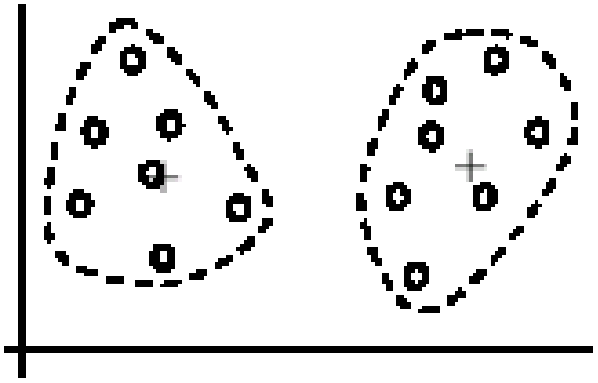
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

The *k*-means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \qquad (2)$$

where $|C_j|$ is the number of data points in cluster C_j. The distance from one data point $\mathbf{x}_i$ to a mean (centroid) $\mathbf{m}_j$ is computed with

$$dist(\mathbf{x}_i, \mathbf{m}_j) = \| \mathbf{x}_i - \mathbf{m}_j \| \qquad (3)$$

$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

A disk version of *k*-means

- K-means can be implemented with data on disk
  - In each iteration, it scans the data once.
  - as the centroids can be computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
  - In practice, a limited is set (< 50).
- Not the best method. There are other scale-up algorithms, e.g., BIRCH.

A disk version of k-means (cont …)

```
Algorithm disk-k-means(k, D)
1   Choose k data points as the initial centriods m_j, j = 1, …, k;
2   repeat
3       initialize s_j = 0, j = 1, …, k;          // 0 is a vector with all 0's
4       initialize n_j = 0, j = 1, …, k;          // n_j is the number points in cluster j
5       for each data point x ∈ D do
6           j = arg min_j dist(x, m_j);
7           assign x to the cluster j;
8           s_j = s_j + x;
9           n_j = n_j + 1;
10      endfor
11      m_i = s_j/n_j, i = 1, …, k;
12  until the stopping criterion is met
```

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,

    where n is the number of data points,

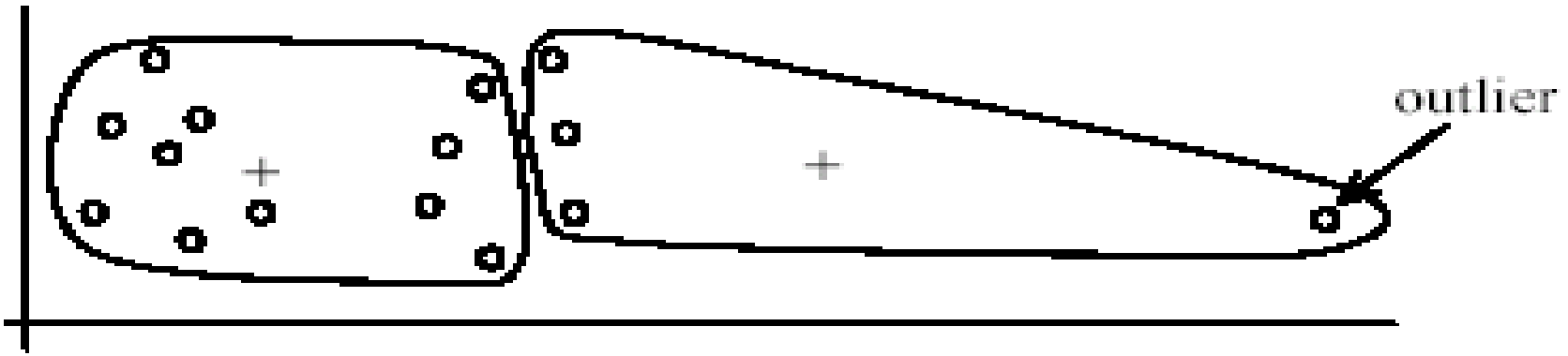
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

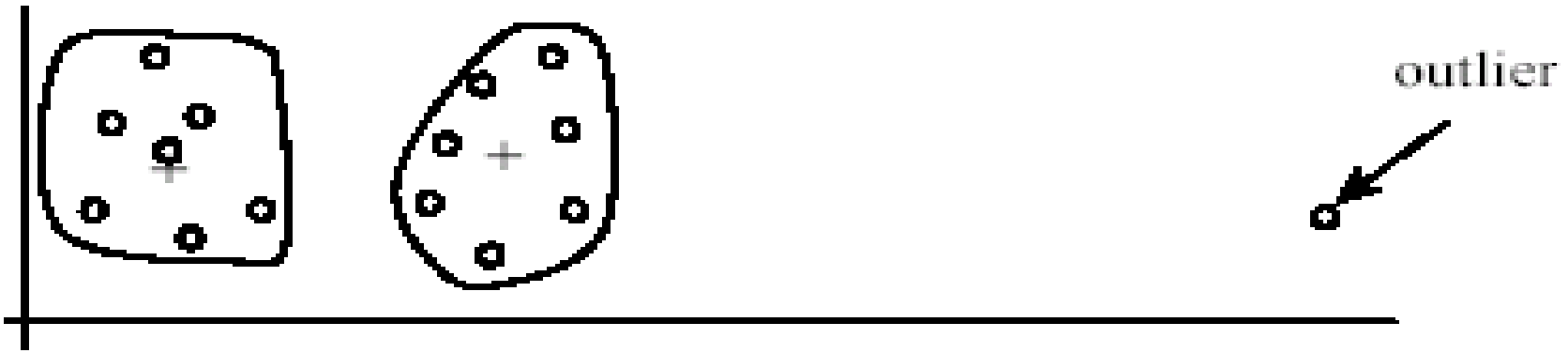
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



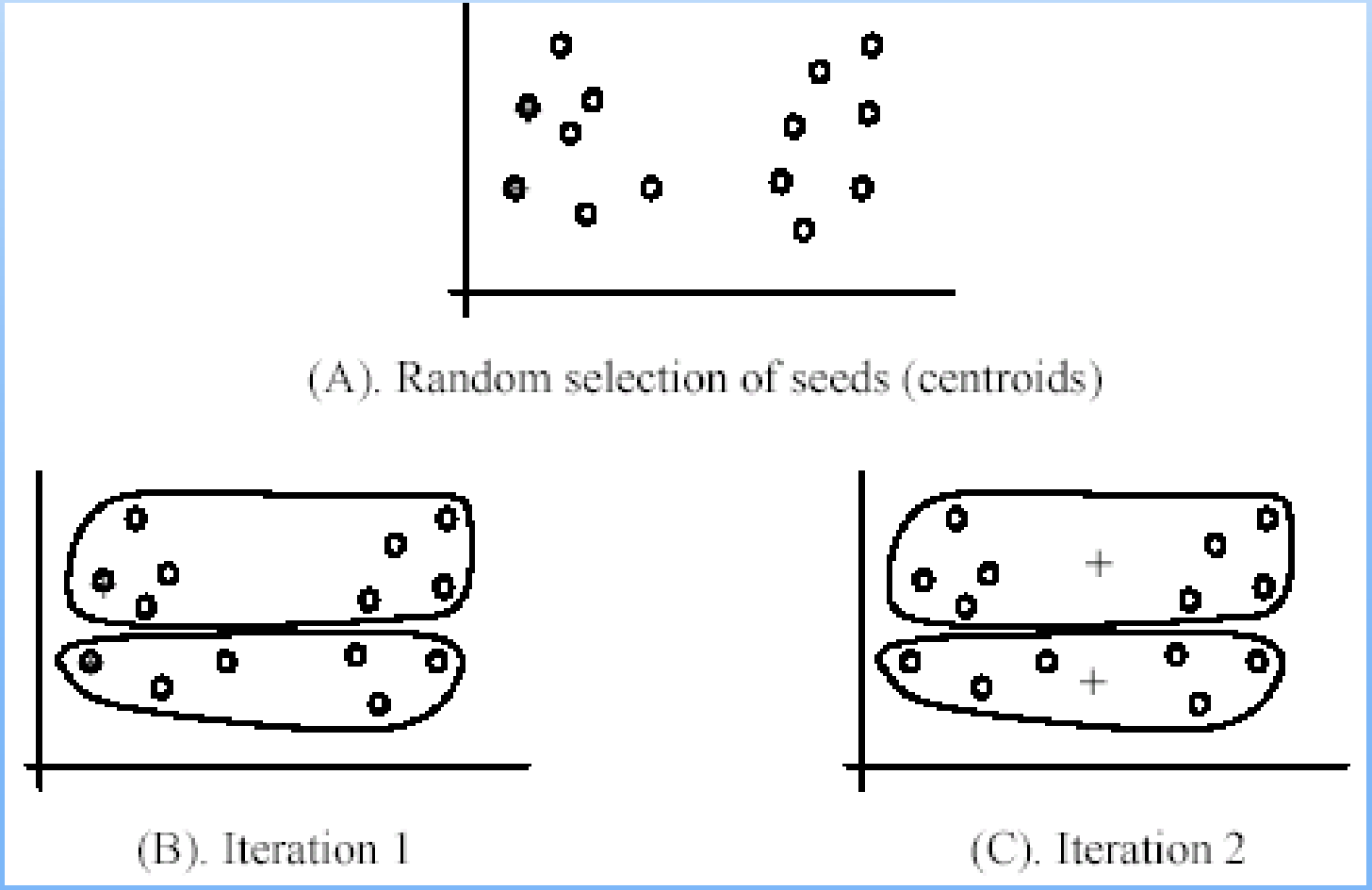
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

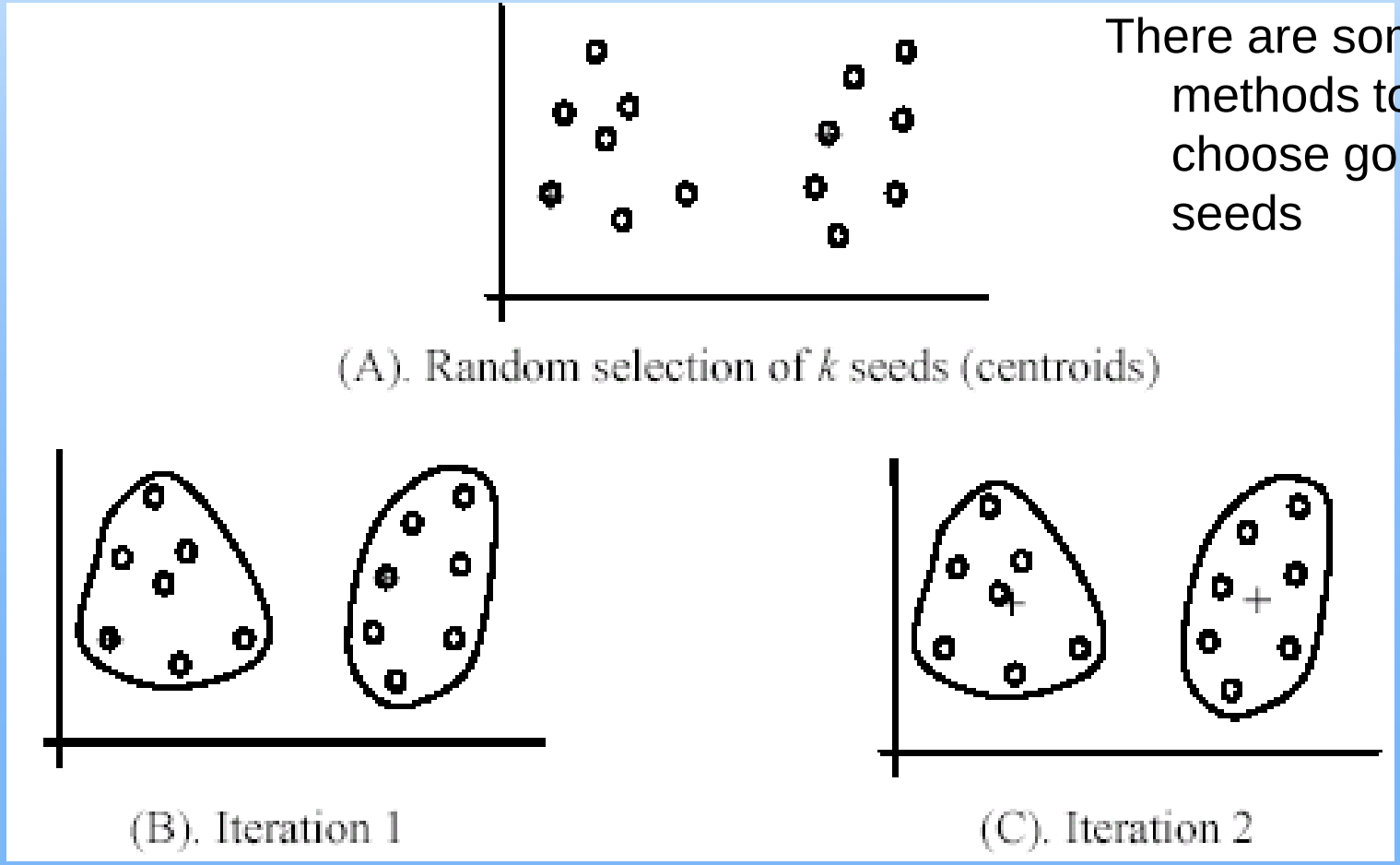
Weaknesses of k-means (cont …)

- The algorithm is sensitive to initial seeds.



Weaknesses of k-means (cont …)

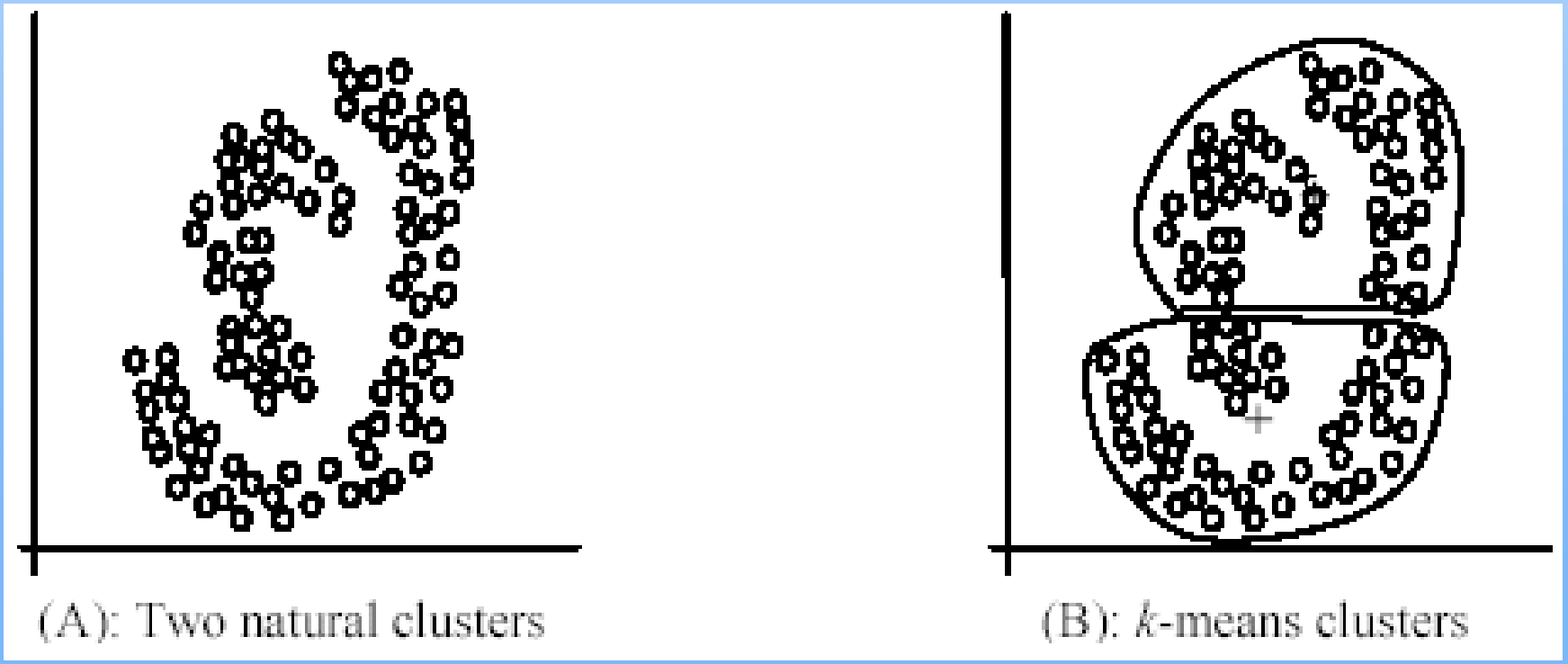
- If we use different seeds: good results



There are some methods to help choose good seeds

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Density Estimation

Density Estimation

Density estimation: is an unsupervised learning

- Learn relations among attributes in the data

Data: $D = \{D_1, D_2, ..., D_n\}$

$D_i = \mathbf{x}_i$ a vector of attribute values

Attributes:

- modeled by random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ with
  - **Continuous or discrete valued variables**

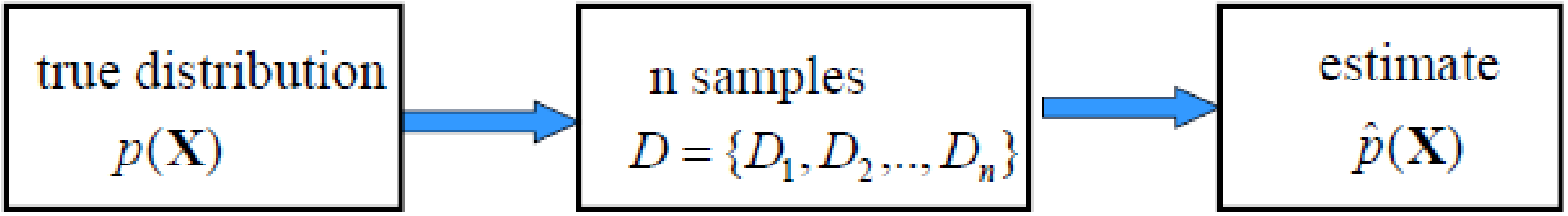
Density estimation attempts to learn the underlying probability distribution: $p(\mathbf{X}) = p(X_1, X_2, \ldots, X_d)$

Density Estimation

Data: $D = \{D_1, D_2, ..., D_n\}$

$D_i = \mathbf{x}_i$ a vector of attribute values

Objective: estimate the underlying probability distribution over variables $\mathbf{X}$, $p(\mathbf{X})$, using examples in D



Standard (iid) assumptions: Samples

- **are independent of each other**
- **come from the same (identical) distribution (fixed $p(\mathbf{X})$)**

Density Estimation

- Types
  - Parametric and Non Parametric

Parametric Approach

Coin example: we have a coin that can be biased

Outcomes: two possible values -- head or tail

Data: D a sequence of outcomes x_i such that

- **head** $x_i = 1$
- **tail** $x_i = 0$

Model: probability of a head θ
probability of a tail $(1-\theta)$

Objective:

We would like to estimate the probability of a **head** $\hat{\theta}$ from data

Parametric Approach

- **Assume** the unknown and possibly biased coin
- Probability of the head is θ
- **Data:**

  H H T T H H T H T H T T T H T H H H H T H H H H T

  - **Heads:** 15
  - **Tails:** 10

What would be your choice of the probability of a head ?

Solution: use frequencies of occurrences to do the estimate

$$\tilde{\theta} = \frac{15}{25} = 0.6$$

This is **the maximum likelihood estimate** of the parameter θ

Non Parametric Approach

- In statistics, **kernel density estimation (KDE)** is a non-parametric way to estimate the probability density function of a random variable.
- Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample.

Kernel Density Estimation

- What is a Kernel?
- A kernel is a special type of probability density function (PDF) with the added property that it must be even. Thus, a kernel is a function with the following properties
- non-negative
- real-valued
- even
- its definite integral over its support set must equal to 1

Even Function

- Let f(x) be a real-valued function of a real variable.
- Then f is even if the following equation holds for all x and -x in the domain of f:[1]
- $f(x)=f(-x)$,
- or
- $f(x)-f(-x)=0$

Even Function

- Geometrically speaking, the graph face of an even function is symmetric with respect to the *y*-axis, meaning that its graph remains unchanged after reflection about the *y*-axis.

What is Kernel Density Estimation?

- Kernel density estimation is a non-parametric method of estimating the probability density function (PDF) of a continuous random variable.
- It is non-parametric because it does not assume any underlying distribution for the variable.
- Essentially, at every datum, a kernel function is created with the datum at its centre – this ensures that the kernel is symmetric about the datum.

What is Kernel Density Estimation?

- The PDF is then estimated by adding all of these kernel functions and dividing by the number of data to ensure that it satisfies the 2 properties of a PDF:
- Every possible value of the PDF (i.e. the function, f(x)), is non-negative.
- The definite integral of the PDF over its support set equals to 1.

Constructing a Kernel Density Estimate: Step by Step

Constructing a Kernel Density Estimate: Step by Step

1) Choose a kernel; the common ones are normal (Gaussian), uniform (rectangular), and triangular.

2) At each datum, x_i, build the scaled kernel function

$$h^{-1}K[(x - x_i)/h],$$

where $K()$ is your chosen kernel function. The parameter h is called the bandwidth, the window width, or the smoothing parameter.

3) Add all of the individual scaled kernel functions and divide by n; this places a probability of $1/n$ to each x_i. It also ensures that the kernel density estimate integrates to 1 over its support set.

$$\hat{f}(x) = n^{-1}h^{-1}\sum_{i=1}^{n} K[(x - x_i)/h]$$

The density() function in R computes the values of the kernel density estimate. Applying the plot() function to an object created by density() will plot the estimate. Applying the summary() function to the object will reveal useful statistics about the estimate.

Frequent Statistics

- **Frequentist Statistics** tests whether an event (hypothesis) occurs or not.

no. of tosses	no. of heads	difference
10	4	-1
50	25	0
100	44	-6
500	255	5
1000	502	2
5000	2533	33
10000	5067	67

Frequent Statistics

We know that probability of getting a head on tossing a fair coin is 0.5.

No. of heads represents the actual number of heads obtained.

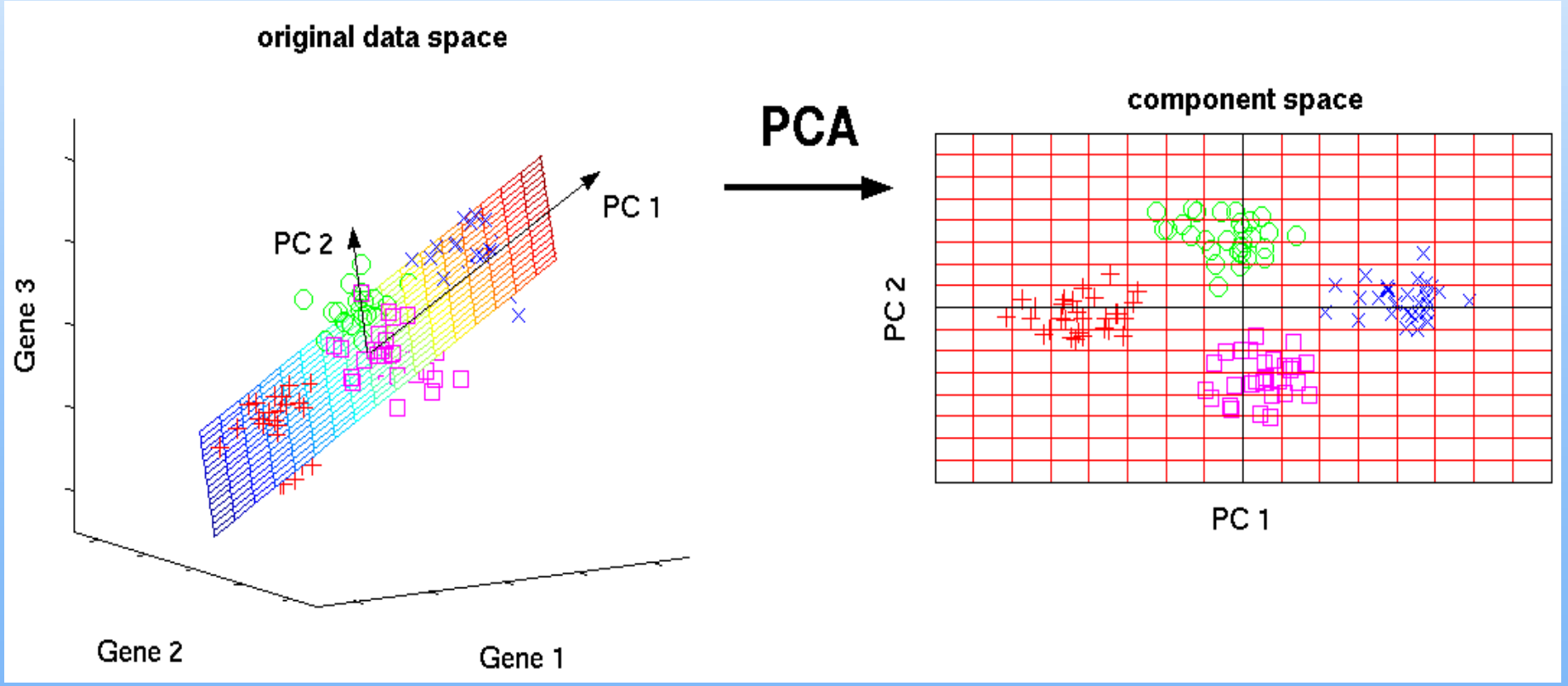
Difference is the difference between 0.5*(No. of tosses) - no. of heads.

Frequent Statistics

- The researcher flips the coin five times and observes heads each time (HHHHH), yielding a test statistic of 5.
- In a one-tailed test, this is the upper extreme of all possible outcomes, and yields a *p*-value of $(1/2)^5 = 1/32 \approx 0.03$.
- Confidence Interval (C.I) like p-value depends heavily on the sample size.
- This makes the stopping potential absolutely absurd since no matter how many persons perform the tests on the same data, the results should be consistent.

PCAWhat is Principal Component Analysis

- Principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set.
- It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible.
- With fewer variables, visualization also becomes much more meaningful.
- PCA is more useful when dealing with 3 or higher dimensional data.


original data space
PC 1
PC 2
Gene 3
Gene 2
Gene 1
PCA
component space
PC 2
PC 1

What are principal components ?

A principal component is a normalized linear combination of the original predictors in a data set. In image above, *PC1* and *PC2* are the principal components. Let's say we have a set of predictors as $X^1, X^2 ..., X^p$

The principal component can be written as:

$$Z^1 = \Phi^{11}X^1 + \Phi^{21}X^2 + \Phi^{31}X^3 + ... + \Phi^{p1}X^p$$

where,

- Z^1 is first principal component
- Φ^{p1} is the loading vector comprising of loadings ($\Phi^1, \Phi^2 ..$) of first principal component. The loadings are constrained to a sum of square equals to 1. This is because large magnitude of loadings may lead to large variance. It also defines the direction of the principal component (Z^1) along which data varies the most. It results in a line in *p* dimensional space which is closest to the *n* observations. Closeness is measured using average squared euclidean distance.
- $X^1..X^p$ are normalized predictors. Normalized predictors have mean equals to zero and standard deviation equals to one.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

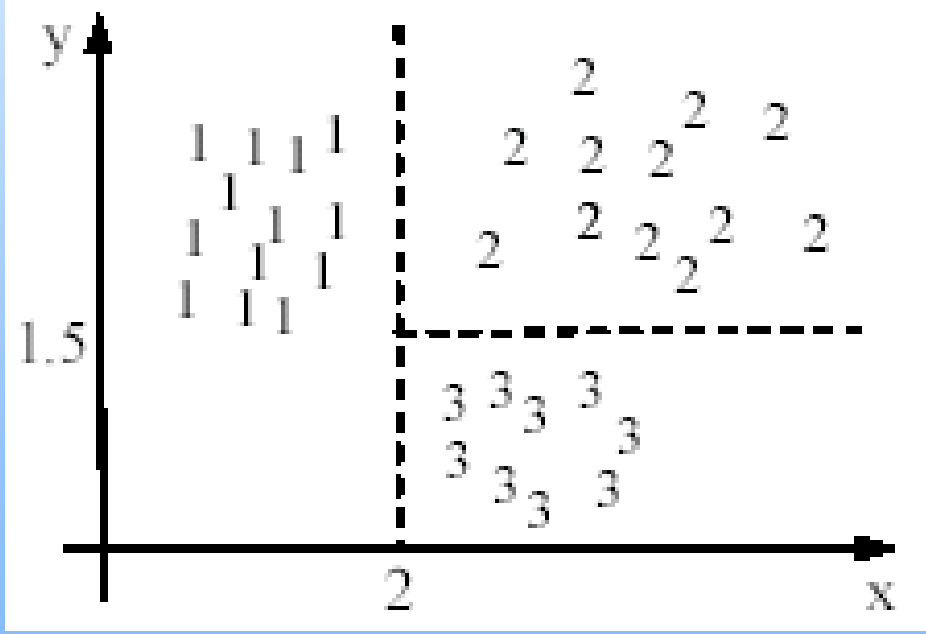
Common ways to represent clusters

- Use the centroid of each cluster to represent the cluster.
  - compute the radius and
  - standard deviation of the cluster to determine its spread in each dimension

  - The centroid representation alone works well if the clusters are of the hyper-spherical shape.
  - If clusters are elongated or are of other shapes, centroids are not sufficient

Using classification model

- All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.
  - run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$ cluster 1

$x > 2, y > 1.5 \rightarrow$ cluster 2

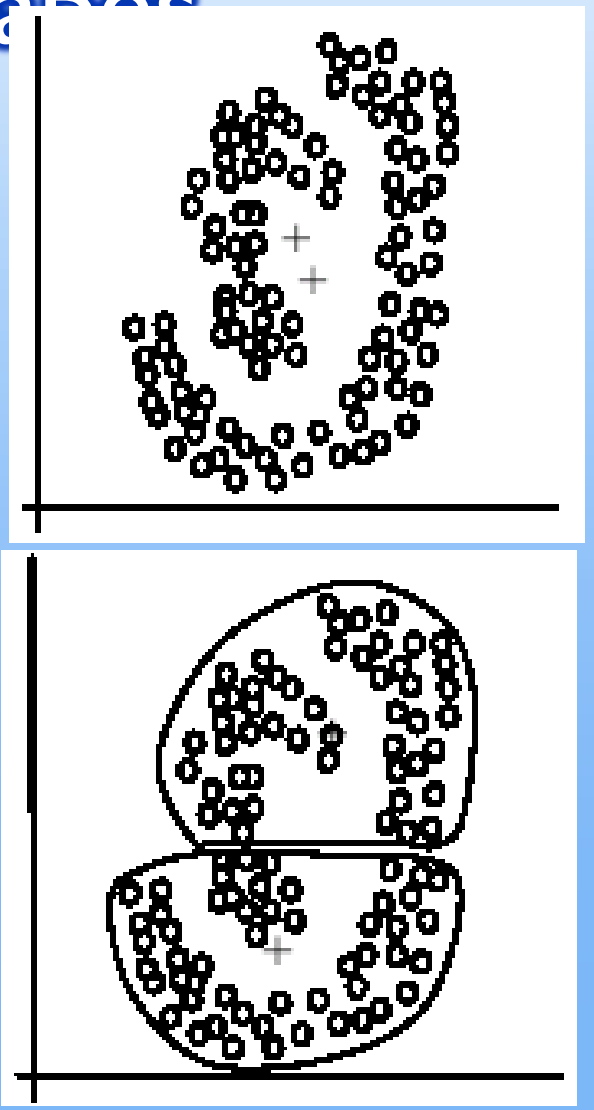
$x > 2, y \leq 1.5 \rightarrow$ cluster 3

Use frequent values to represent cluster

- This method is mainly for clustering of categorical data (e.g., *k*-modes clustering).
- Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

Clusters of arbitrary shapes

- Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.
- Irregular shape clusters are hard to represent. They may not be useful in some applications.
  - Using centroids are not suitable (upper figure) in general
  - K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.

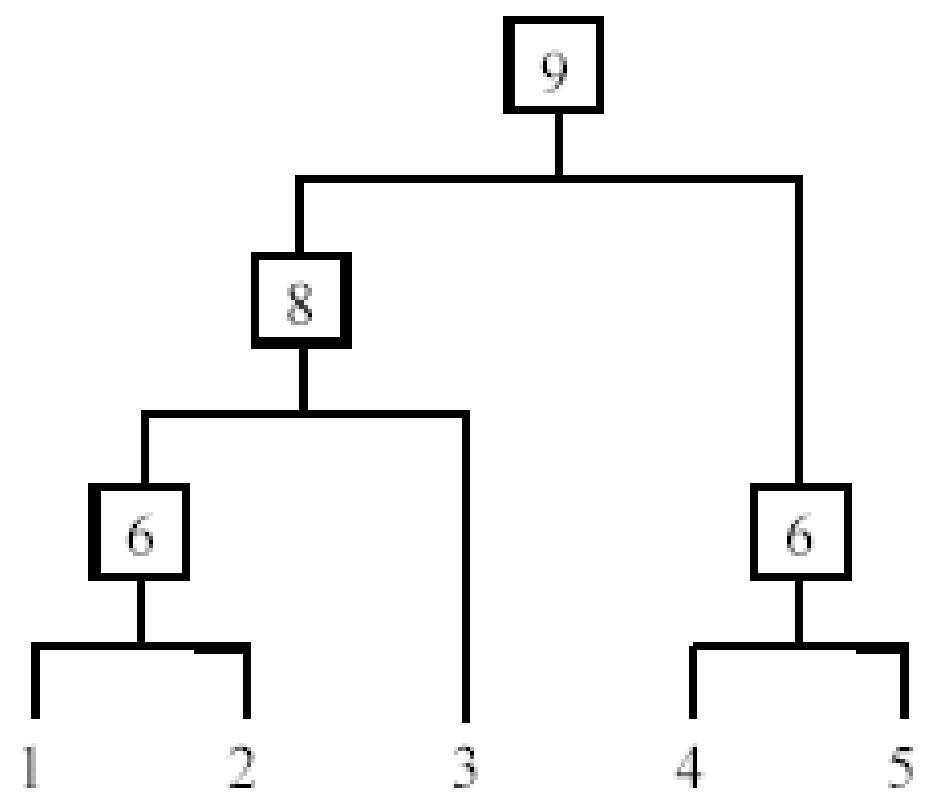


Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Hierarchical Clustering

- Produce a nested sequence of clusters, a tree, also called Dendrogram.



Types of hierarchical clustering

- Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and
  - merges the most similar (or nearest) pair of clusters
  - stops when all the data points are merged into a single cluster (i.e., the root cluster).
- Divisive (top down) clustering: It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Agglomerative clustering

It is more popular then divisive methods.

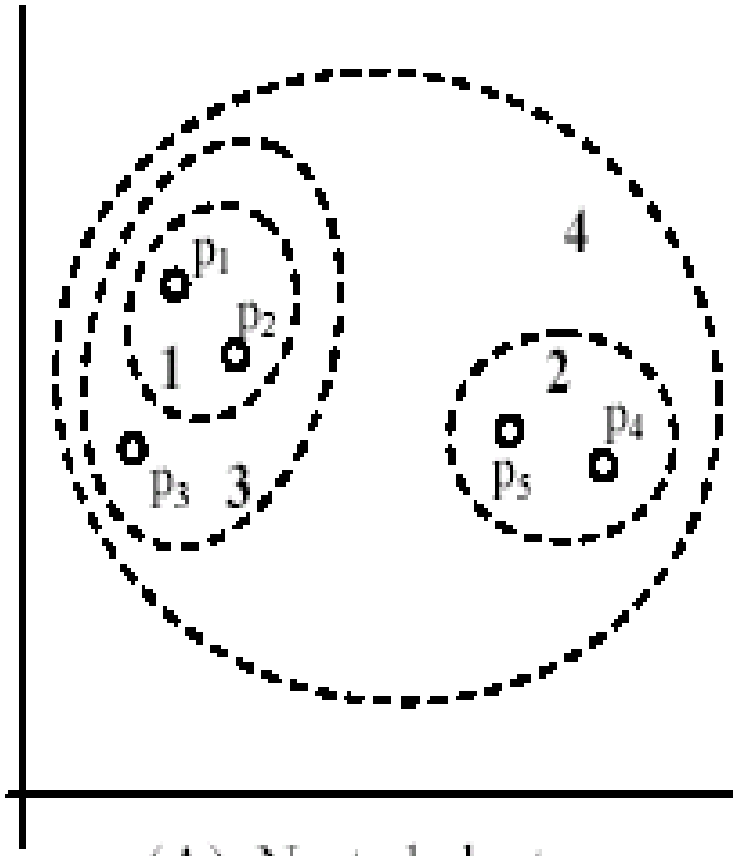
- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

Agglomerative clustering algorithm

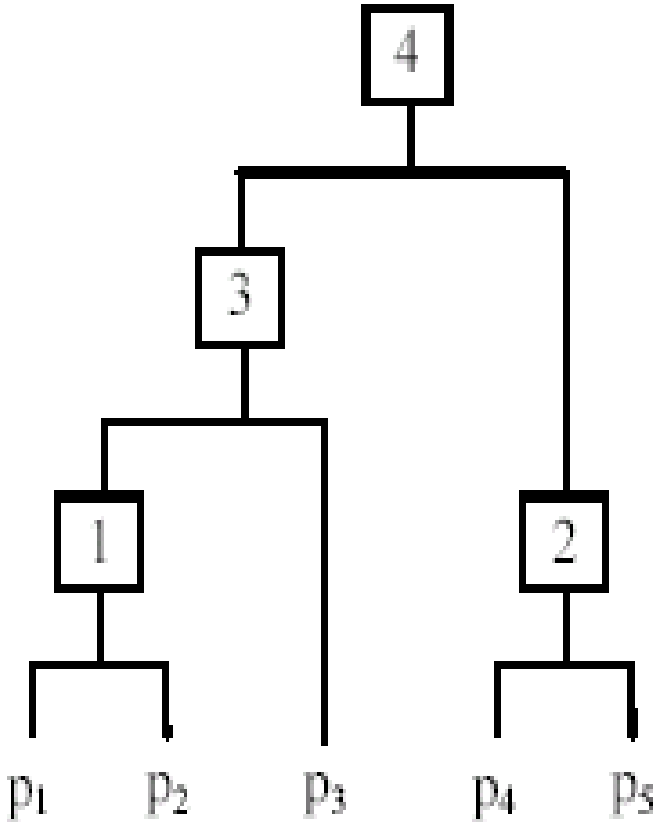
Algorithm Agglomerative(D)

1 Make each data point in the data set D a cluster,
2 Compute all pair-wise distances of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in D$;
2 **repeat**
3 find two clusters that are nearest to each other;
4 merge the two clusters form a new cluster c;
5 compute the distance from c to all other clusters;
12 **until** there is only one cluster left

An example: working of the algorithm



(A). Nested clusters



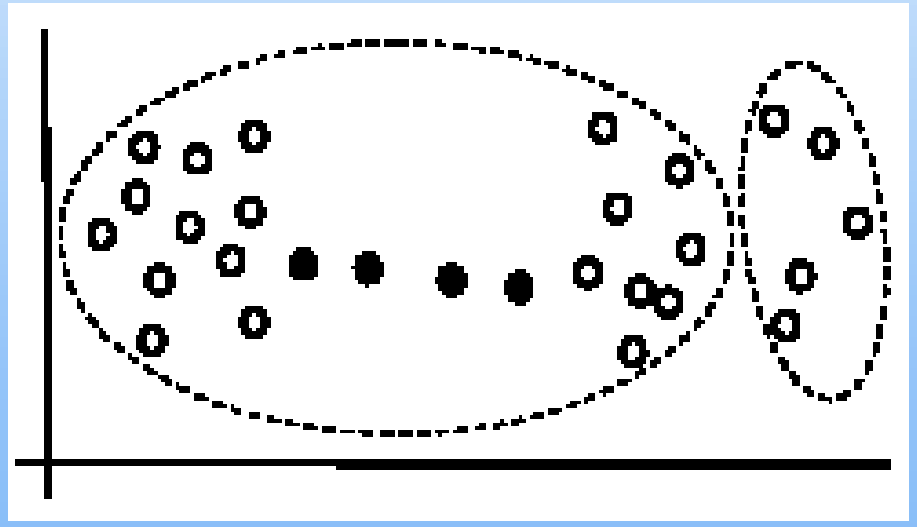
(B) Dendrogram

Measuring the distance of two clusters

- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
  - Single link
  - Complete link
  - Average link
  - Centroids
  - …

Single link method

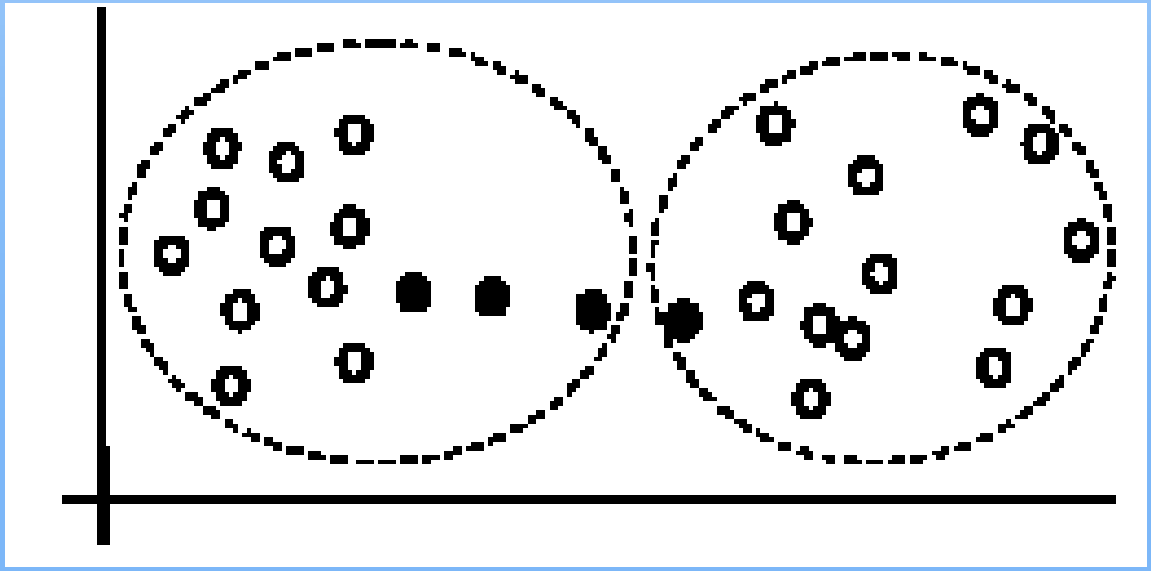
- The distance between two clusters is the distance between two closest data points in the two clusters, one data point from each cluster.
- It can find arbitrarily shaped clusters, but
  - It may cause the undesirable "chain effect" by noisy points



Two natural clusters are split into two

Complete link method

- The distance between two clusters is the distance of two furthest data points in the two clusters.
- It is sensitive to outliers because they are far away



Average link and centroid methods

- Average link: A compromise between
  - the sensitivity of complete-link clustering to outliers and
  - the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.
  - In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.
- Centroid method: In this method, the distance between two clusters is the distance between their centroids

The complexity

- All the algorithms are at least $O(n^2)$. n is the number of data points.
- Single link can be done in $O(n^2)$.
- Complete and average links can be done in $O(n^2 \log n)$.
- Due the complexity, hard to use for large data sets.
  - Sampling
  - Scale-up methods (e.g., BIRCH).

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Distance functions

- Key to clustering. "similarity" and "dissimilarity" can also commonly used terms.
- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications

Distance functions for numeric attributes

- Most commonly used functions are
  - Euclidean distance and
  - Manhattan (city block) distance
- We denote distance with: $dist(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are data points (vectors)
- They are special cases of Minkowski distance. h is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = ((x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + ... + (x_{ir} - x_{jr})^h)^{\frac{1}{h}}$$

Euclidean distance and Manhattan distance

- If $h = 2$, it is the Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2}$$

- If $h = 1$, it is the Manhattan distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ir} - x_{jr}|$$

- Weighted Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \ldots + w_r(x_{ir} - x_{jr})^2}$$

Squared distance and Chebychev distance

- Squared Euclidean distance: to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2$$

- Chebychev distance: one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(| x_{i1} - x_{j1} |, | x_{i2} - x_{j2} |, ..., | x_{ir} - x_{jr} |)$$

Distance functions for binary and nominal attributes

- Binary attribute: has two values or states but no ordering relationships, e.g.,
  - Gender: male and female.
- We use a confusion matrix to introduce the distance functions/measures.
- Let the *i*th and *j*th data points be $\mathbf{x}_i$ and $\mathbf{x}_j$ (vectors)

Confusion matrix

Data point i \ Data point j	1	0	
1	a	b	$a+b$
0	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d$

(10)

a: the number of attributes with the value of 1 for both data points.

b: the number of attributes for which $x_{if} = 1$ and $x_{jf} = 0$, where x_{if} (x_{jf}) is the value of the fth attribute of the data point $\mathbf{x}_i$ ($\mathbf{x}_j$).

c: the number of attributes for which $x_{if} = 0$ and $x_{jf} = 1$.

d: the number of attributes with the value of 0 for both data points.

Symmetric binary attributes

- A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender
- Distance function: Simple Matching Coefficient, proportion of mismatches of their values

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c + d}$$

Symmetric binary attributes: example

x_1	1	1	1	0	1	0	0
x_2	0	1	1	0	0	1	0

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{2+1}{2+2+1+2} = \frac{3}{7} = 0.429$$

Asymmetric binary attributes

- Asymmetric: if one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state, which is typically the rare or infrequent state.
  - Jaccard coefficient is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c}$$

  - We can have some variations, adding weights

Nominal attributes

- Nominal attributes: with more than two states or values.
  - the commonly used distance measure is also based on the simple matching method.
  - Given two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, let the number of attributes be r, and the number of values that match in $\mathbf{x}_i$ and $\mathbf{x}_j$ be q.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{r - q}{r}$$

Distance function for text documents

- A text document consists of a sequence of sentences and each sentence consists of a sequence of words.
- To simplify: a document is usually considered a "bag" of words in document clustering.
  - Sequence and position of words are ignored.
- A document is represented with a vector just like a normal data point.
- It is common to use similarity to compare two documents rather than distance.
  - The most commonly used similarity function is the **cosine similarity**. We will study this later.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points
  - $\mathbf{x}_i$: (0.1, 20) and $\mathbf{x}_j$: (0.9, 720).

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9 - 0.1)^2 + (720 - 20)^2} = 700.000457,$$

- The distance is almost completely dominated by (720-20) = 700.
- Standardize attributes: to force the attributes to have a common value range

Interval-scaled attributes

- Their values are real numbers following a linear scale.
  - The difference in Age between 10 and 20 is the same as that between 40 and 50.
  - The key idea is that intervals keep the same importance through out the scale
- Two main approaches to standardize interval scaled attributes, **range** and **z-score**. *f* is an attribute

$$range(x_{if}) = \frac{x_{if} - \min(f)}{\max(f) - \min(f)},$$

Interval-scaled attributes (cont …)

- Z-score: transforms the attribute values so that they have a mean of zero and a **mean absolute deviation** of 1. The mean absolute deviation of attribute *f*, denoted by s_f, is computed as follows

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|\right),$$

$$m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \ldots + x_{nf}\right),$$

Z-score: $$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

Ratio-scaled attributes

- Numeric attributes, but unlike interval-scaled attributes, their scales are exponential,
- For example, the total amount of microorganisms that evolve in a time *t* is approximately given by

  Ae^{Bt},

  - where *A* and *B* are some positive constants.
- Do log transform:

  $$\log(x_{if})$$

  - Then treat it as an interval-scaled attribuete

Nominal attributes

- Sometime, we need to transform nominal attributes to numeric attributes.
- Transform nominal attributes to binary attributes.
  - The number of values of a nominal attribute is *v*.
  - Create *v* binary attributes to represent them.
  - If a data instance for the nominal attribute takes a particular value, the value of its binary attribute is set to 1, otherwise it is set to 0.
- The resulting binary attributes can be used as numeric attributes, with two values, 0 and 1.

Nominal attributes: an example

- Nominal attribute *fruit*: has three values,
  - Apple, Orange, and Pear
- We create three binary attributes called, Apple, Orange, and Pear in the new data.
- If a particular data instance in the original data has Apple as the value for *fruit*,
  - then in the transformed data, we set the value of the attribute Apple to 1, and
  - the values of attributes Orange and Pear to 0

Ordinal attributes

- Ordinal attribute: an ordinal attribute is like a nominal attribute, but its values have a numerical ordering. E.g.,
  - Age attribute with values: Young, MiddleAge and Old. They are ordered.
  - Common approach to standardization: treat is as an interval-scaled attribute.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Mixed attributes

- Our distance functions given are for data with all numeric attributes, or all nominal attributes, etc.
- Practical data has different types:
  - Any subset of the 6 types of attributes,
    - **interval-scaled**,
    - **symmetric binary**,
    - **asymmetric binary**,
    - **ratio-scaled**,
    - **ordinal** and
    - **nominal**

Convert to a single type

- One common way of dealing with mixed attributes is to
  - Decide the dominant attribute type, and
  - Convert the other types to this type.
- E.g, if most attributes in a data set are interval-scaled,
  - we convert ordinal attributes and ratio-scaled attributes to interval-scaled attributes.
  - It is also appropriate to treat symmetric binary attributes as interval-scaled attributes.

Convert to a single type (cont …)

- It does not make much sense to convert a nominal attribute or an asymmetric binary attribute to an interval-scaled attribute,
  - but it is still frequently done in practice by assigning some numbers to them according to some hidden ordering, e.g., prices of the fruits
- Alternatively, a nominal attribute can be converted to a set of (symmetric) binary attributes, which are then treated as numeric attributes.

Combining individual distances

- This approach computes individual attribute distances and then combine them.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{f=1}^{r} \delta_{ij}^f d_{ij}^f}{\sum_{f=1}^{r} \delta_{ij}^f}$$

This distance value is between 0 and 1. r is the number of attributes in the data set. The indicator δ_{ij}^f is 1 when both values x_{if} and x_{jf} for attribute f are non-missing, and it is set to 0 otherwise. It is also set to 0 if attribute f is asymmetric and the match is 0-0. Equation (25) cannot be computed if all δ_{ij}^f 's are 0. In such a case, some default value may be used or one of the data points is removed.

d_{ij}^f is the distance contributed by attribute f, and it is in the 0-1 range.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced several main algorithms.
- Choosing the "best" algorithm is a challenge.
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any "ideal" structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

Choose a clustering algorithm (cont …)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly application dependent and to certain extent subjective (personal preferences).

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Cluster Evaluation: hard problem

- The quality of a clustering is very hard to evaluate because
  - We do not know the correct clusters
- Some methods are used:
  - User inspection
    - Study centroids, and spreads
    - Rules from a decision tree.
    - For text documents, one can read some documents in clusters.

Cluster evaluation: ground truth

- We use some labeled data (for classification)
- Assumption: Each class is a cluster.
- After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, entropy, purity, precision, recall and F-score.
  - Let the classes in the data D be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces k clusters, which divides D into k disjoint subsets, $D_1, D_2, \ldots, D_k$.

Evaluation measures: Entropy

Entropy: For each cluster, we can measure its entropy as follows:

$$entropy(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j) \log_2 \Pr_i(c_j). \quad (29)$$

where $\Pr_i(c_j)$ is the proportion of class c_j data points in cluster i or D_i. The total entropy of the whole clustering (which considers all clusters) is

$$entropy_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times entropy(D_i) \quad (30)$$

Evaluation measures: purity

Purity: This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j(\Pr_i(c_j)) \quad (31)$$

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times purity(D_i) \quad (32)$$

An example

Example 14: Assume we have a text collection D of 900 documents from three topics (or three classes), Science, Sports, and Politics. Each class has 300 documents. Each document in D is labeled with one of the topics (classes). We use this collection to perform clustering to find three clusters. Note that class/topic labels are not used in clustering. After clustering, we want to measure the effectiveness of the clustering algorithm.

Cluster	Science	Sports	Politics		Entropy	Purity
1	250	20	10		0.589	0.893
2	20	180	80		1.198	0.643
3	30	100	210		1.257	0.617
Total	300	300	300		1.031	0.711

A remark about ground truth evaluation

- Commonly used to compare different clustering algorithms.
- A real-life data set for clustering has no class labels.
  - Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.
- The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.
- This evaluation method is said to be based on **external data** or information.

Evaluation based on internal information

- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key.

Indirect evaluation

- In some applications, clustering is not the primary task, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

Holes in data space

- All the clustering algorithms only group data.
- Clusters only represent one aspect of the knowledge in the data.
- Another aspect that we have not studied is the **holes**.
  - A hole is a region in the data space that contains no or few data points. Reasons:
    - insufficient data in certain areas, and/or
    - certain attribute-value combinations are not possible or seldom occur.

Holes are useful too

- Although clusters are important, holes in the space can be quite useful too.
- For example, in a disease database
  - we may find that certain symptoms and/or test values do not occur together, or
  - when a certain medicine is used, some test values never go beyond certain ranges.
- Discovery of such information can be important in medical domains because
  - it could mean the discovery of a cure to a disease or some biological laws.

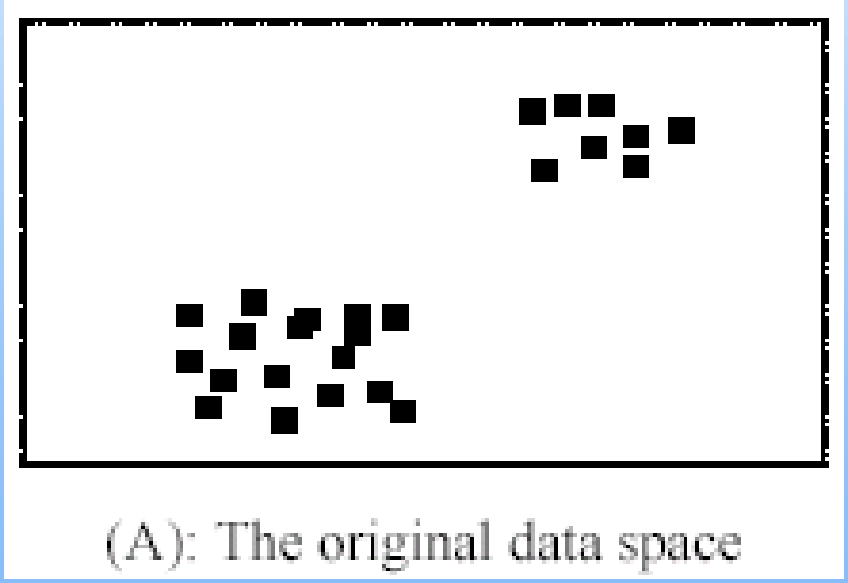
Data regions and empty regions

- Given a data space, separate
  - data regions (clusters) and
  - empty regions (holes, with few or no data points).
- Use a supervised learning technique, i.e., decision tree induction, to separate the two types of regions.
- Due to the use of a supervised learning method for an unsupervised learning task,
  - an interesting connection is made between the two types of learning paradigms.

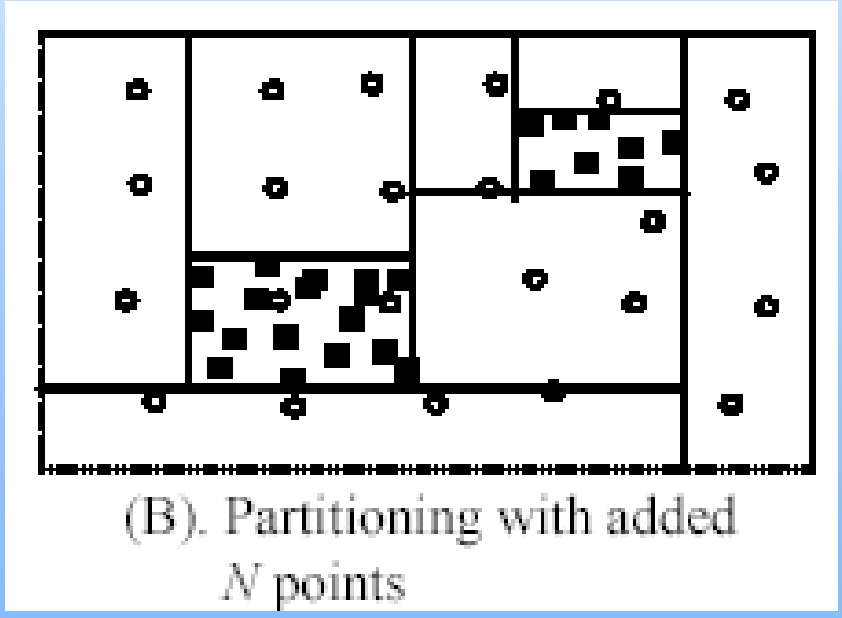
Supervised learning for unsupervised learning

- Decision tree algorithm is not directly applicable.
  - it needs at least two classes of data.
  - A clustering data set has no class label for each data point.
- The problem can be dealt with by a simple idea.
  - Regard each point in the data set to have a class label *Y*.
  - Assume that the data space is uniformly distributed with another type of points, called **non-existing points**. We give them the class, *N*.
- With the *N* points added, the problem of partitioning the data space into data and empty regions becomes a supervised classification problem.

An example



(A): The original data space



(B). Partitioning with added N points

A decision tree method is used for partitioning in (B).

Can it done without adding *N* points?

- Yes.
- Physically adding *N* points increases the size of the data and thus the running time.
- More importantly: it is unlikely that we can have points truly uniformly distributed in a high dimensional space as we would need an exponential number of points.
- Fortunately, no need to physically add any *N* points.
  - We can compute them when needed

Characteristics of the approach

- It provides representations of the resulting data and empty regions in terms of hyper-rectangles, or rules.
- It detects outliers automatically. Outliers are data points in an empty region.
- It may not use all attributes in the data just as in a normal decision tree for supervised learning.
  - It can automatically determine what attributes are useful. Subspace clustering …
- Drawback: data regions of irregular shapes are hard to handle since decision tree learning only generates hyper-rectangles (formed by axis-parallel hyper-planes), which are rules.

Building the Tree

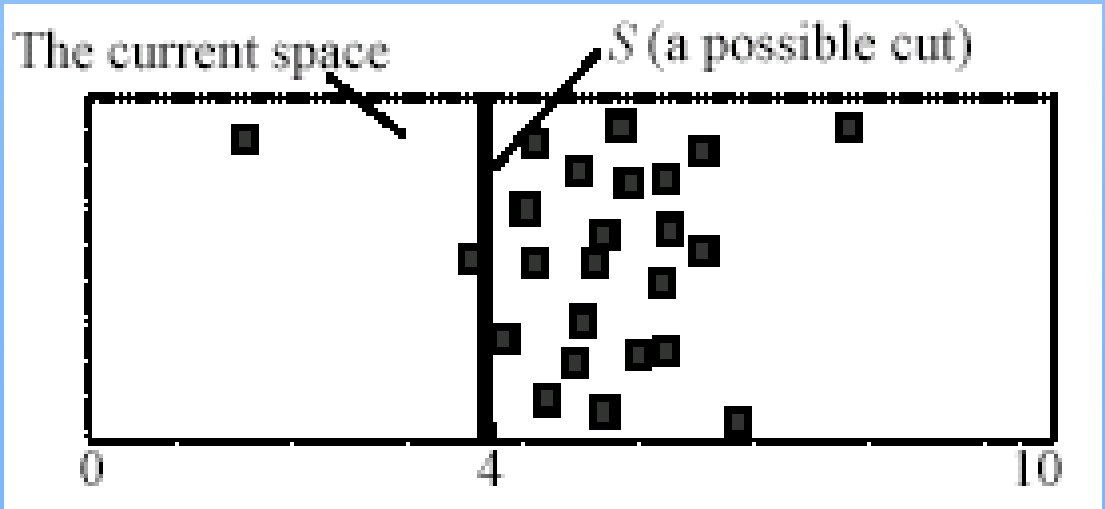
- The main computation in decision tree building is to evaluate entropy (for information gain):

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- Can it be evaluated without adding *N* points? Yes.
- Pr(c_j) is the probability of class c_j in data set *D*, and |*C*| is the number of classes, *Y* and *N* (2 classes).
  - To compute Pr(c_j), we only need the number of *Y* (data) points and the number of *N* (non-existing) points.
  - We already have *Y* (or data) points, and we can compute the number of *N* points on the fly. Simple: as we assume that the *N* points are uniformly distributed in the space.

An example

- The space has 25 data (*Y*) points and 25 *N* points. Assume the system is evaluating a possible cut *S*.
  - # *N* points on the left of *S* is 25 * 4/10 = 10. The number of *Y* points is 3.
  - Likewise, # *N* points on the right of *S* is 15 (= 25 - 10). The number of *Y* points is 22.
- With these numbers, entropy can be computed.



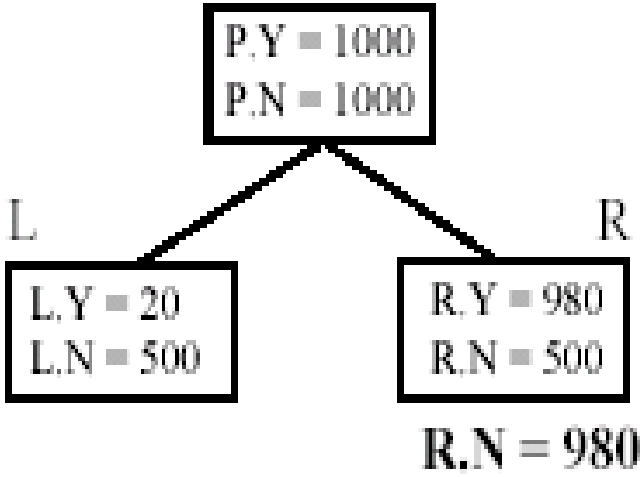
How many *N* points to add?

- We add a different number of *N* points at each different node.
  - The number of *N* points for the current node *E* is determined by the following rule (note that at the root node, the number of inherited *N* points is 0):

```
1  If the number of N points inherited from the parent node of E is less than the
       number of Y points in E then
2      the number of N points for E is increased to the number of Y points in E
3  else the number of inherited N points is used for E
```

An example

Example 17: Fig. 20 gives an example. The (parent) node P has two children nodes L and R. Assume P has 1000 Y points and thus 1000 N points, stored in $P.Y$ and $P.N$ respectively. Assume after splitting, L has 20 Y points and 500 N points, and R has 980 Y points and 500 N points. According to the above rule, for subsequent partitioning, we increase the number of N points at R to 980. The number of N points at L is unchanged.



How many *N* points to add? (cont…)

- Basically, for a *Y* node (which has more data points), we increase *N* points so that

  #Y = #N

- The number of *N* points is not reduced if the current node is an *N* node (an *N* node has more *N* points than *Y* points).
  - A reduction may cause outlier *Y* points to form *Y* nodes (a *Y* node has an equal number of *Y* points as *N* points or more).
  - Then data regions and empty regions may not be separated well.

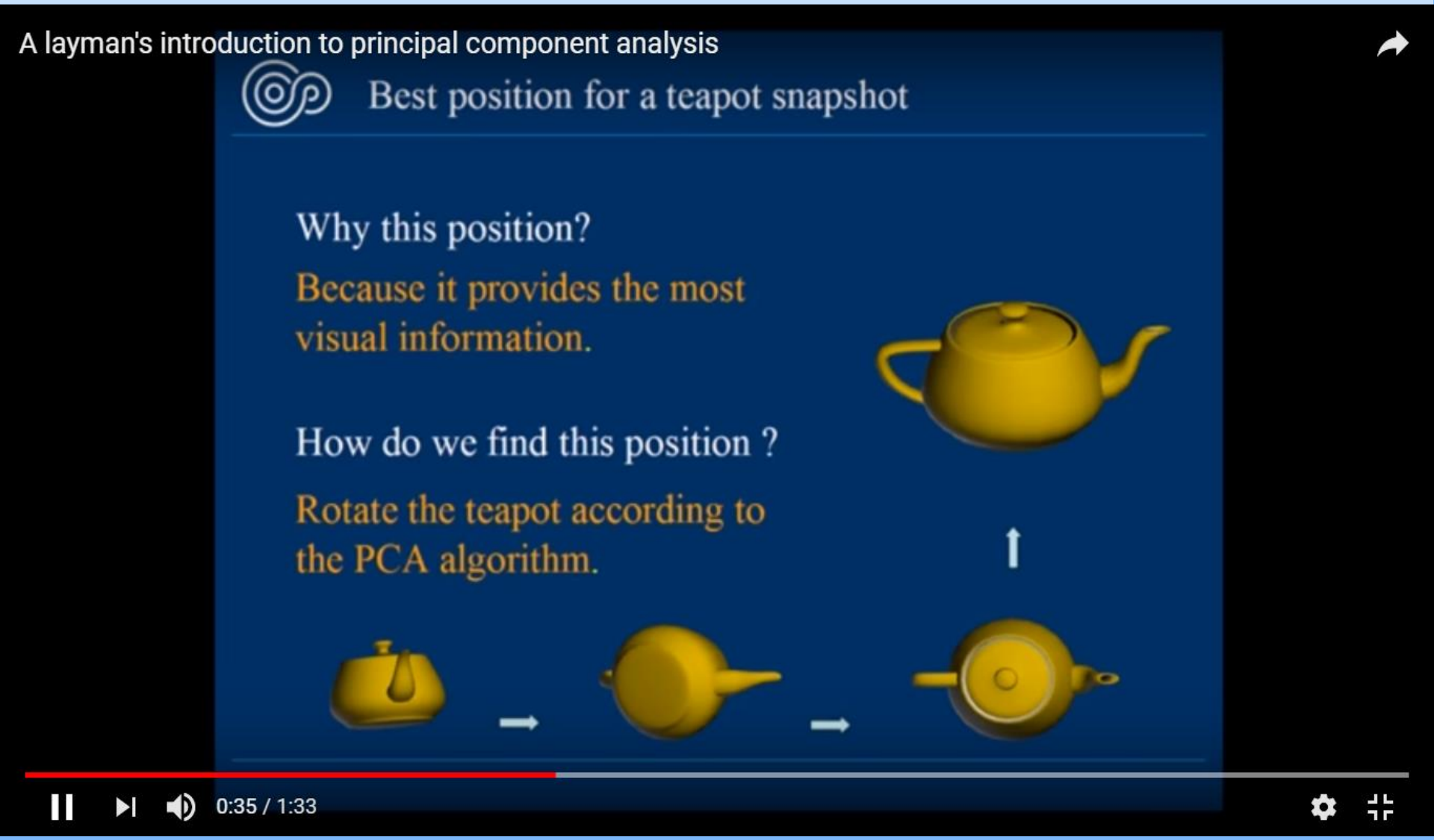
Building the decision tree

- Using the above ideas, a decision tree can be built to separate data regions and empty regions.
- The actual method is more sophisticated as a few other tricky issues need to be handled in
  - tree building and
  - tree pruning.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

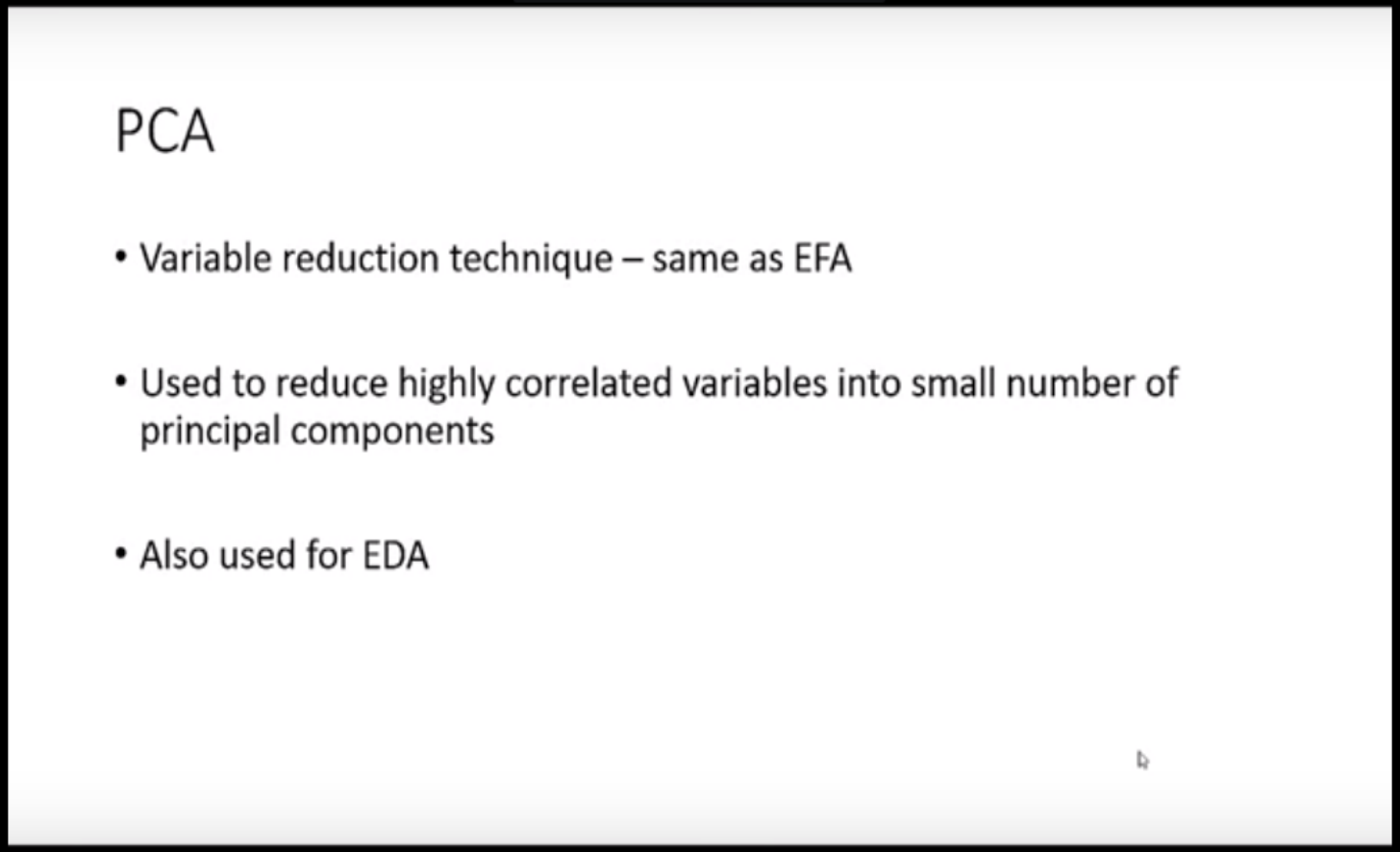
PCA


A layman's introduction to principal component analysis
Best position for a teapot snapshot
Why this position?
Because it provides the most visual information.
How do we find this position ?
Rotate the teapot according to the PCA algorithm.
0:35 / 1:33

PCA

Principal Component Analysis

Press Esc to exit full screen



Analytics University

0:40 / 10:03

PCA

Example of Principal Component Analysis (PCA).mp4

Press Esc to exit full screen

- We can use var function of excel to obtain variance.
- For covariance, we will use function
  - $(\Sigma (x1 - x1bar)*(x2-x2bar))/ (\#\ of\ terms\ -1)$
  - Please don't use excel's covar formula, as that is for population not for sample.

	x1	x2	x1-x1bar	x2-x2bar	(x1-x1bar)*(x2-x2bar)	
	1.4000	1.6500	1.8500	2.2175	4.1024	
	1.6000	1.9750	2.0500	2.5425	5.2121	
	-1.4000	-1.7750	-0.9500	-1.2075	1.1471	
	-2.0000	-2.5250	-1.5500	-1.9575	3.0341	
	-3.0000	-3.9500	-2.5500	-3.3825	8.6254	
	2.4000	3.0750	2.8500	3.6425	10.3811	
	1.5000	2.0250	1.9500	2.5925	5.0554	
	2.3000	2.7500	2.7500	3.3175	9.1231	
	-3.2000	-4.0500	-2.7500	-3.4825	9.5769	
	-4.1000	-4.8500	-3.6500	-4.2825	15.6311	
avg	-0.4500	-0.5675			7.9876	<-- covariance
var	6.4228	9.9528	6.4228	9.9528	16.3756	<-- sum of var

Step1 : covariance matrix is

$$\begin{bmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{bmatrix}$$

2:43 / 9:55

PCA

- **Step 2: is to obtain Eigen Values by solving function determinant $(A - \lambda I) = 0$**

i.e. $(var1 - \lambda) * (var2 - \lambda) - covariance(1,2)^2 = 0$

Or $\lambda^2 - (var1 + var2) * \lambda + [var1 * var2 - cov(1,2)^2]$

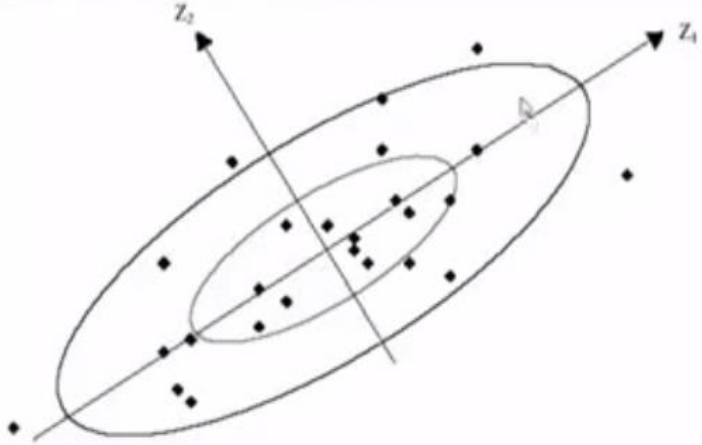
In our case, it would be $\lambda^2 - 16.3756 * \lambda + 0.12214]$.....(A)

- Remember solution of $ax^2 + bx + c = 0$
- It used to be $x =$
- $[-b + \sqrt{(b^2 - 4ac)}]/2a$ or $[-b - \sqrt{(b^2 - 4ac)}]/2a$
- Note an important characteristic, If you sum the above two values of x, you will get $-b/a$ (why?)
- Solving eq A, we get $\lambda = 16.36809984, 0.007462657$
- These two are Eigen Values.
- Note $\lambda 1 + \lambda 2 = 16.3756$, which is the sum of variance. Sum of Eigen Values will always be equal to sum of variances (why?).

PCA

Example of Principal Component Analysis (PCA).mp4

Visualization of PCA action:

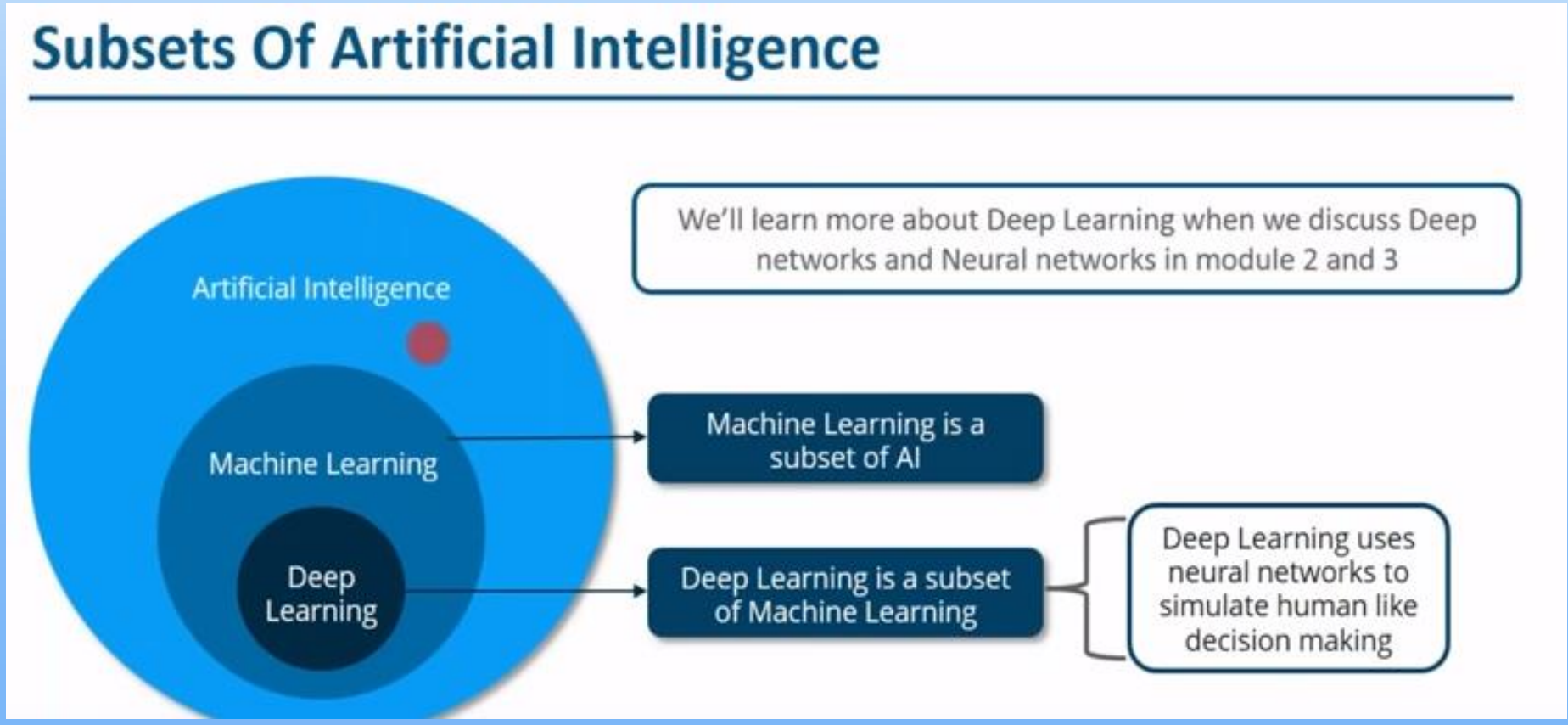


First principal .

1. Among all possible axis passing through the center of data i.e center of x1 and x2, it is the direction, on which if we draw perpendicular lines from all data points, the variance of projection points on the line z1 will be maximum.

9:47 / 9:55

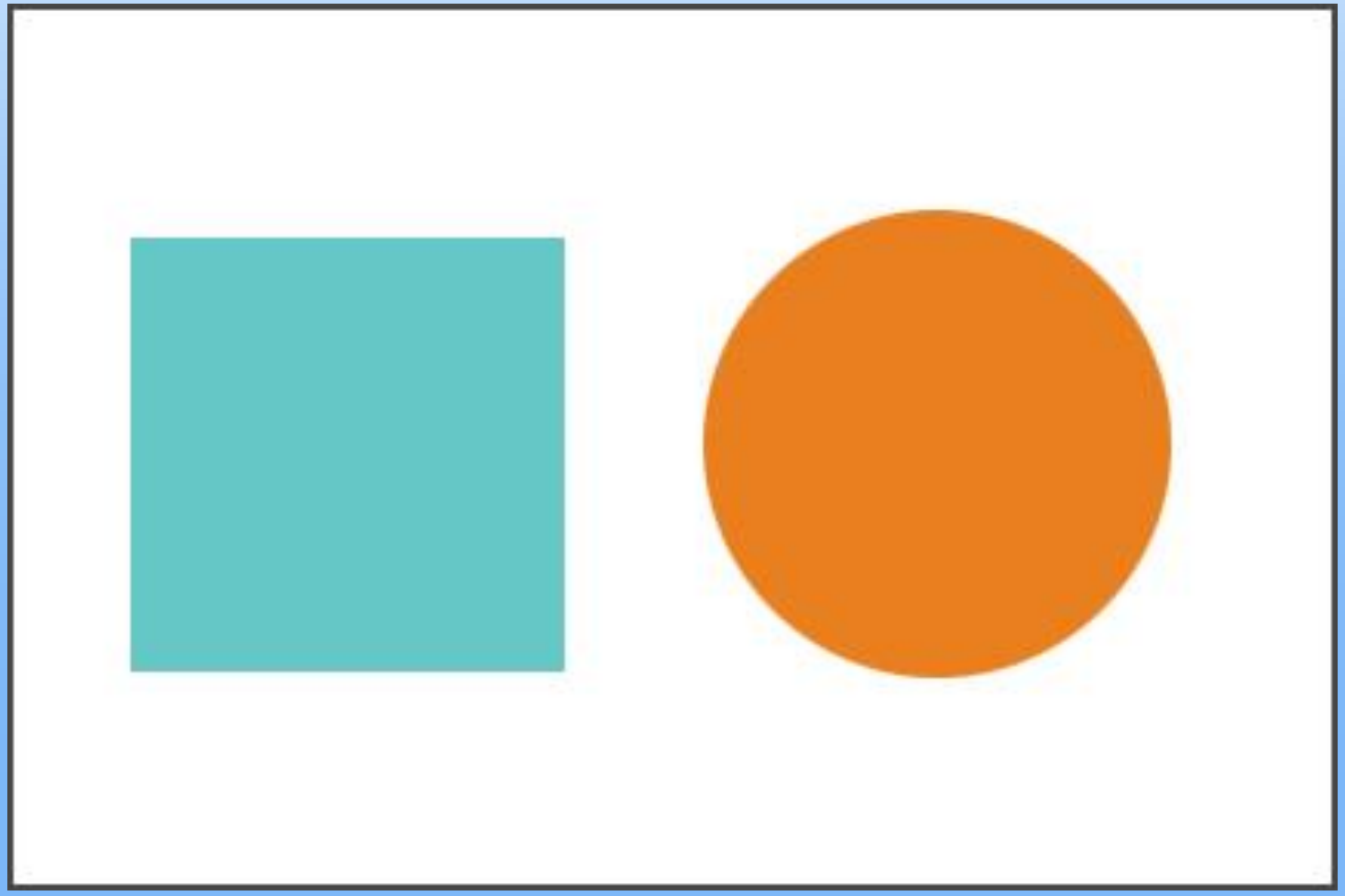
Deep Learning



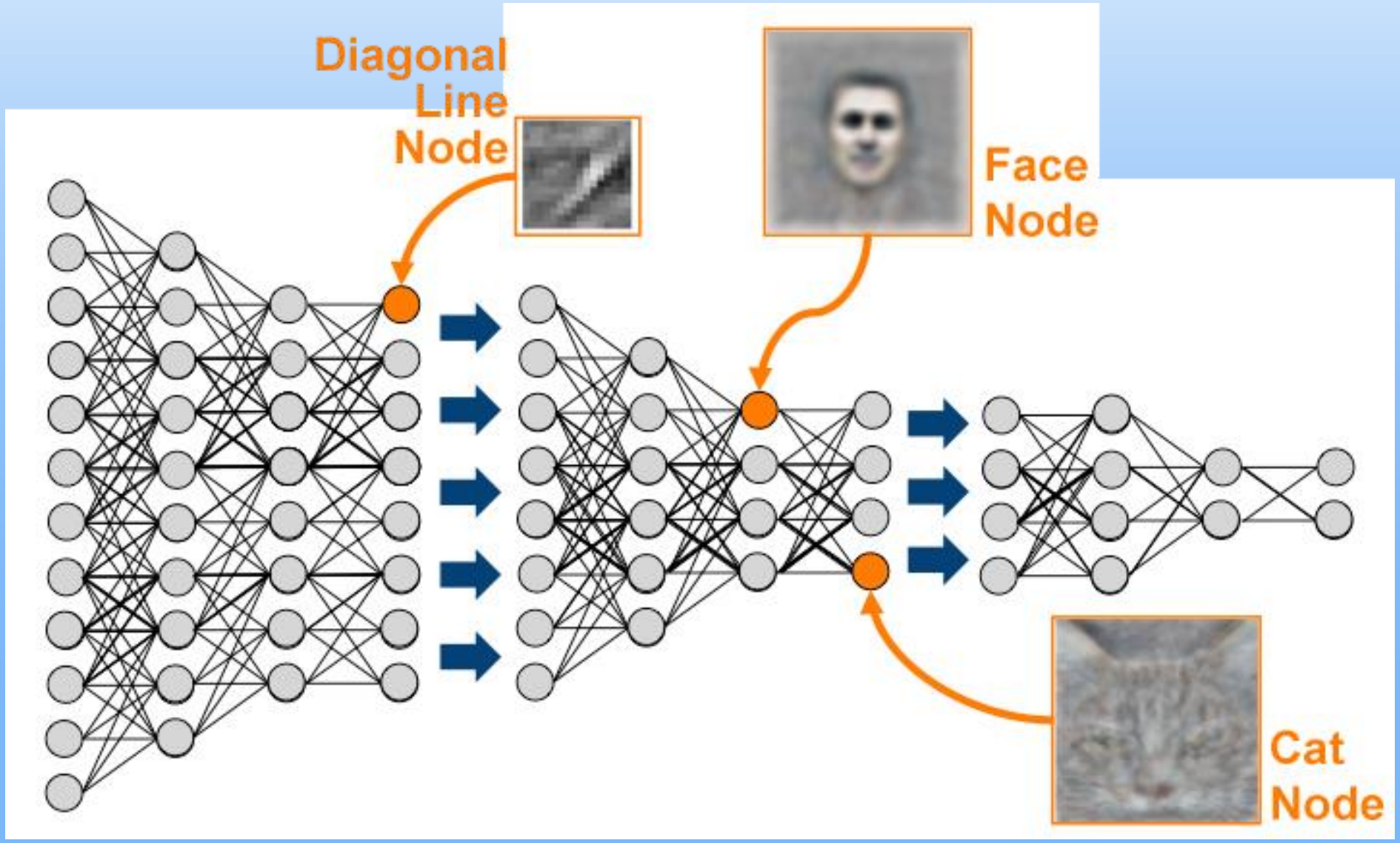
Deep Learning

- Deep learning automatically finds out the features which are important for classification, where in Machine Learning we had to manually give the features.

Deep Learning



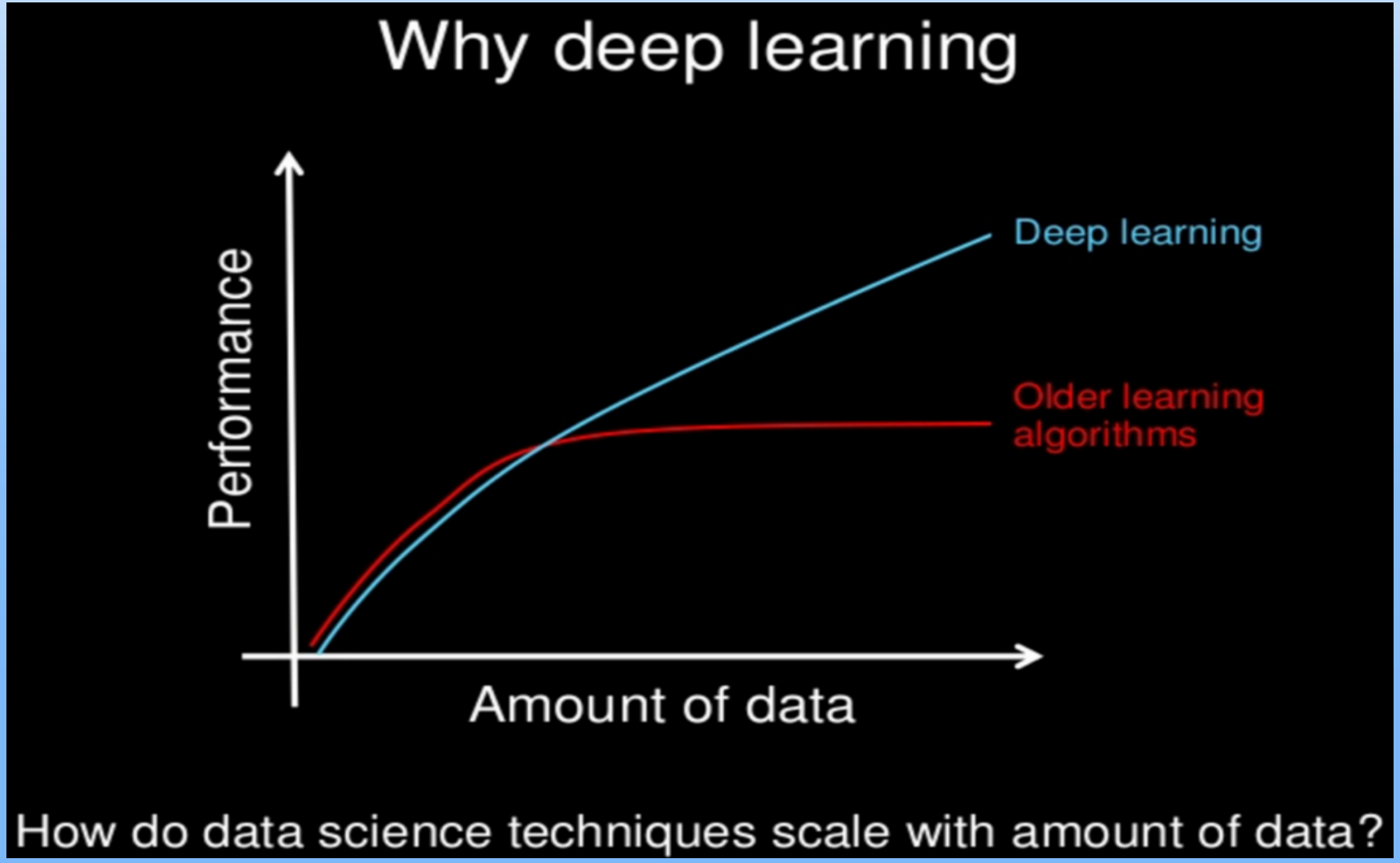
Deep Learning



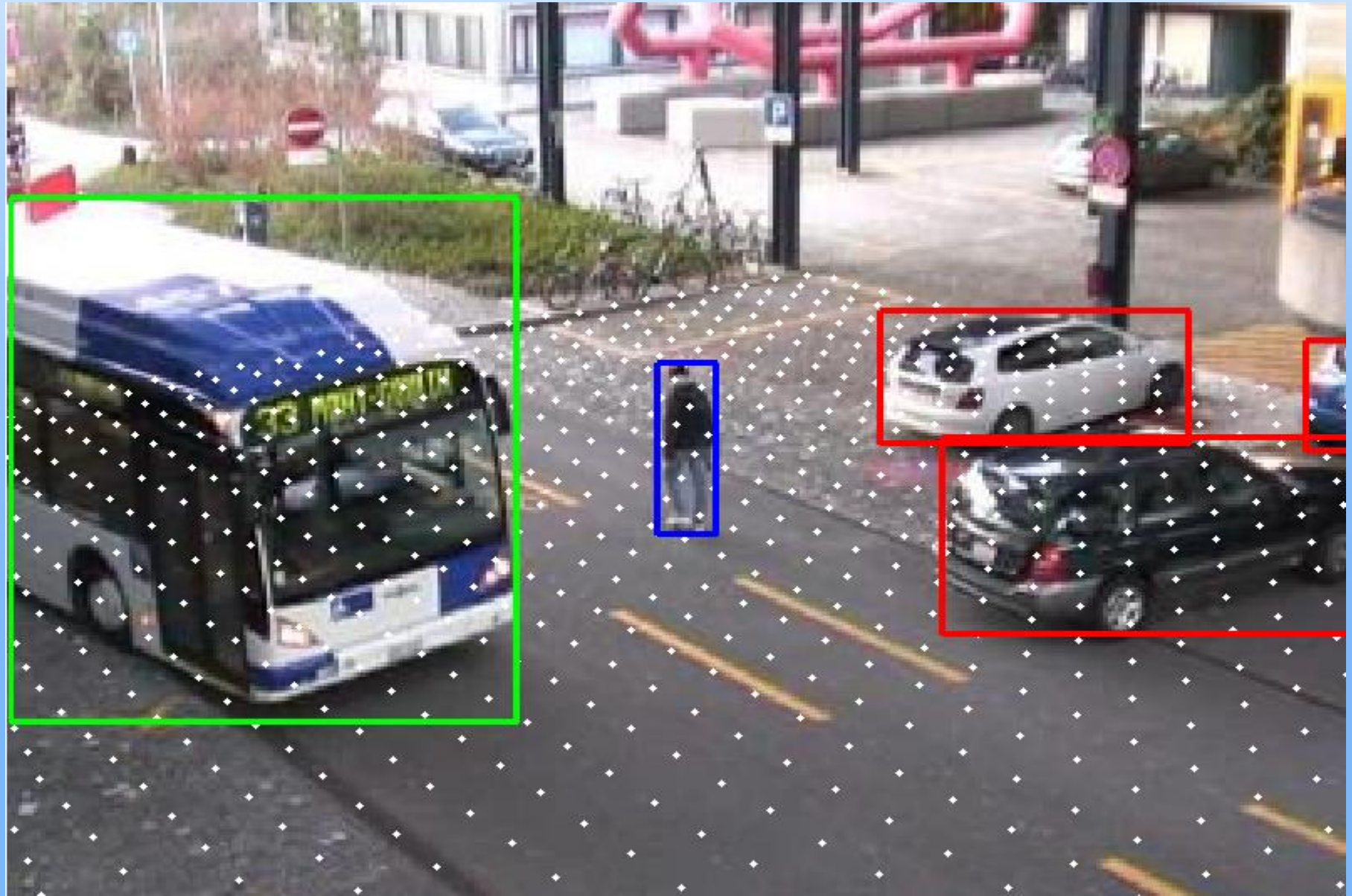
Deep Learning

- It first identifies what are the edges that are most relevant to find out a Cat or a Dog
- It then builds on this hierarchically to find what combination of shapes and edges we can find. For example, whether whiskers are present, or whether ears are present, etc.
- After consecutive hierarchical identification of complex concepts, it then decides which of this features are responsible for finding the answer.

Deep Learning



Feature Engineering



Deep Learning = Learning Hierarchical Representations

Y LeCun

- **Traditional Pattern Recognition:** Fixed/Handcrafted Feature Extractor

Feature Extractor → Trainable Classifier

- **Mainstream Modern Pattern Recognition:** Unsupervised mid-level features

Feature Extractor → Mid-Level Features → Trainable Classifier

- **Deep Learning:** Representations are hierarchical and trained

Low-Level Features → Mid-Level Features → High-Level Features → Trainable Classifier

Applications

- Sentiment Analysis
  - Classifying Customer Reviews to positive or negative

Movie Reviews

The best way to hope for any chance of enjoying this film is by lowering your expectations --- Negative comment

The show starts out as competent but unremarkable and gradually grows in to a considerable power --- **Positive comment**

Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\BALASUBRAMANIAM

Editor

temp.py | standfordnlpdemo.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Sep 25 20:43:48 2017
4
5  @author: BALASUBRAMANIAM
6  """
7
8  from pycorenlp import StanfordCoreNLP
9  nlp = StanfordCoreNLP('http://localhost:7070')
10 res = nlp.annotate("""I love you. I hate him. You are nice. He is dumb.
11                     The best way to hope for any chance of enjoying
12                     this film is by lowering your expectations.The show st
13                     but unremarkable and gradually grows in to a considera
14                     properties={
15                         'annotators': 'sentiment',
16                         'outputFormat': 'json'
17                     })
18 print(res)
19 for s in res["sentences"]:
20     print ("%d: '%s': %s %s" % (
21         s["index"],
22         " ".join([t["word"] for t in s["tokens"]]),
23         s["sentimentValue"], s["sentiment"]))
```

Python console

Python 1

```
'starts', 'originalText': 'starts', 'characterOffsetBegin': 189, 'characterOffs
etEnd': 195, 'pos': 'VBZ', 'before': ' ', 'after': ' '}, {'index': 20, 'word':
'out', 'originalText': 'out', 'characterOffsetBegin': 196, 'characterOffsetEnd'
: 199, 'pos': 'RP', 'before': ' ', 'after': ' '}, {'index': 21, 'word': 'as', '
originalText': 'as', 'characterOffsetBegin': 200, 'characterOffsetEnd': 202, 'p
os': 'IN', 'before': ' ', 'after': ' '}, {'index': 22, 'word': 'competent', 'or
iginalText': 'competent', 'characterOffsetBegin': 203, 'characterOffsetEnd': 21
2, 'pos': 'JJ', 'before': ' ', 'after': ' \n                    '}, {'index': 23
, 'word': 'but', 'originalText': 'but', 'characterOffsetBegin': 233, 'character
OffsetEnd': 236, 'pos': 'CC', 'before': ' \n                    ', 'after': ' '}
, {'index': 24, 'word': 'unremarkable', 'originalText': 'unremarkable', 'charac
terOffsetBegin': 237, 'characterOffsetEnd': 249, 'pos': 'JJ', 'before': ' ', 'a
fter': ' '}, {'index': 25, 'word': 'and', 'originalText': 'and', 'characterOffs
etBegin': 250, 'characterOffsetEnd': 253, 'pos': 'CC', 'before': ' ', 'after':
' '}, {'index': 26, 'word': 'gradually', 'originalText': 'gradually', 'characte
rOffsetBegin': 254, 'characterOffsetEnd': 263, 'pos': 'RB', 'before': ' ', 'aft
er': ' '}, {'index': 27, 'word': 'grows', 'originalText': 'grows', 'characterOf
fsetBegin': 264, 'characterOffsetEnd': 269, 'pos': 'VBZ', 'before': ' ', 'after
': ' '}, {'index': 28, 'word': 'in', 'originalText': 'in', 'characterOffsetBegi
n': 270, 'characterOffsetEnd': 272, 'pos': 'IN', 'before': ' ', 'after': ' '},
{'index': 29, 'word': 'to', 'originalText': 'to', 'characterOffsetBegin': 273,
'characterOffsetEnd': 275, 'pos': 'TO', 'before': ' ', 'after': ' '}, {'index':
30, 'word': 'a', 'originalText': 'a', 'characterOffsetBegin': 276, 'characterOf
fsetEnd': 277, 'pos': 'DT', 'before': ' ', 'after': ' '}, {'index': 31, 'word':
'considerable', 'originalText': 'considerable', 'characterOffsetBegin': 278, 'c
haracterOffsetEnd': 290, 'pos': 'JJ', 'before': ' ', 'after': ' '}, {'index': 3
2, 'word': 'power', 'originalText': 'power', 'characterOffsetBegin': 291, 'char
acterOffsetEnd': 296, 'pos': 'NN', 'before': ' ', 'after': ''}]}]}
0: 'I love you .': 3 Positive
1: 'I hate him .': 1 Negative
2: 'You are nice .': 3 Positive
3: 'He is dumb .': 1 Negative
4: 'The best way to hope for any chance of enjoying this film is by lowering yo
ur expectations.The show starts out as competent but unremarkable and gradually
grows in to a considerable power': 3 Positive
>>>
```

Python console | IPython console

Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 13 | Column: 20 | Memory: 59 %

Type here to search

ENG 20:56 25/09/2017

Stanford nlp for python | localhost:14000

localhost:14000

Stanford CoreNLP

— Text to annotate —

I going to Bangalore

— Annotations —

parts-of-speech | named entities | dependency parse | openie

— Language —

English

Submit

Part-of-Speech:

Named Entity Recognition:

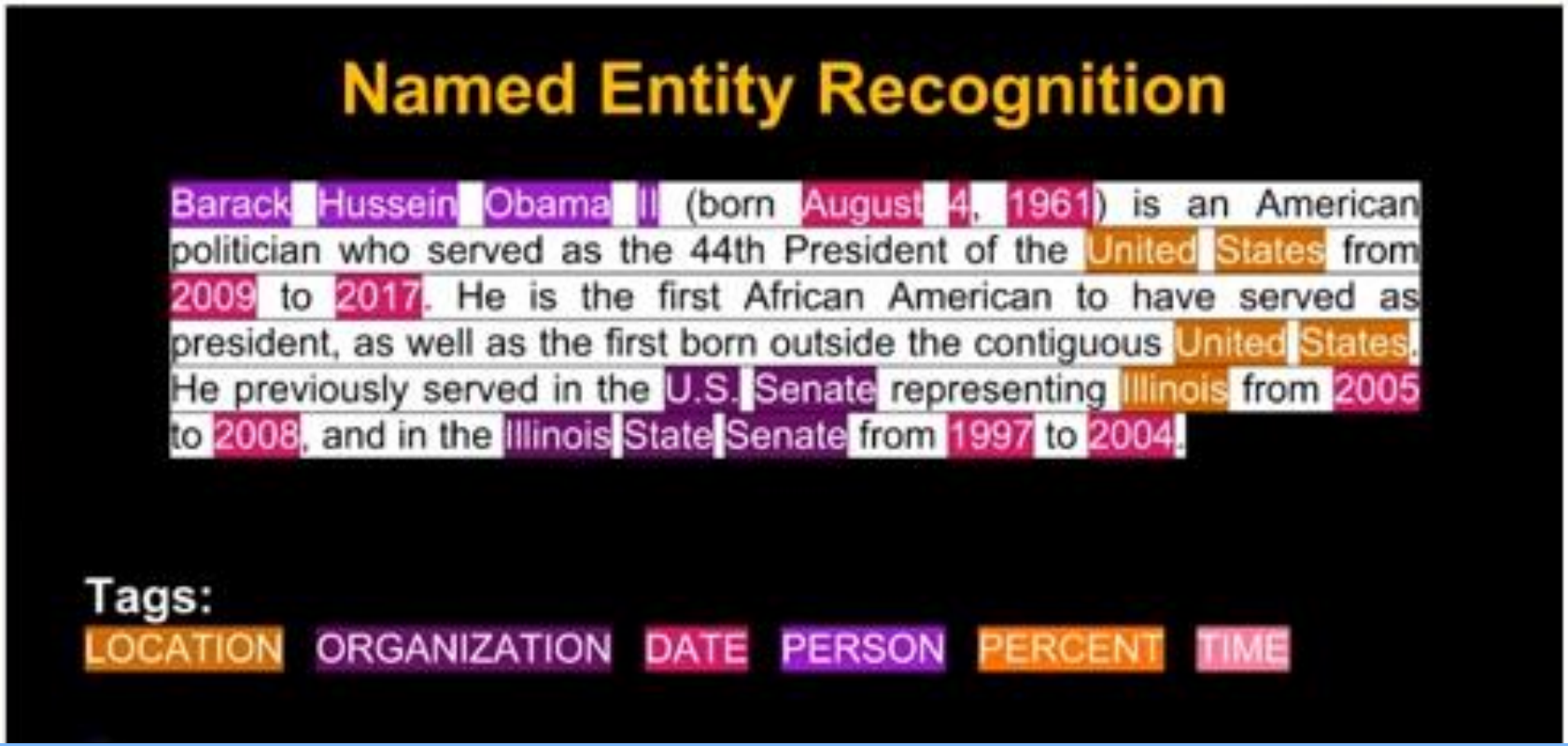
Basic Dependencies:

Enhanced++ Dependencies:

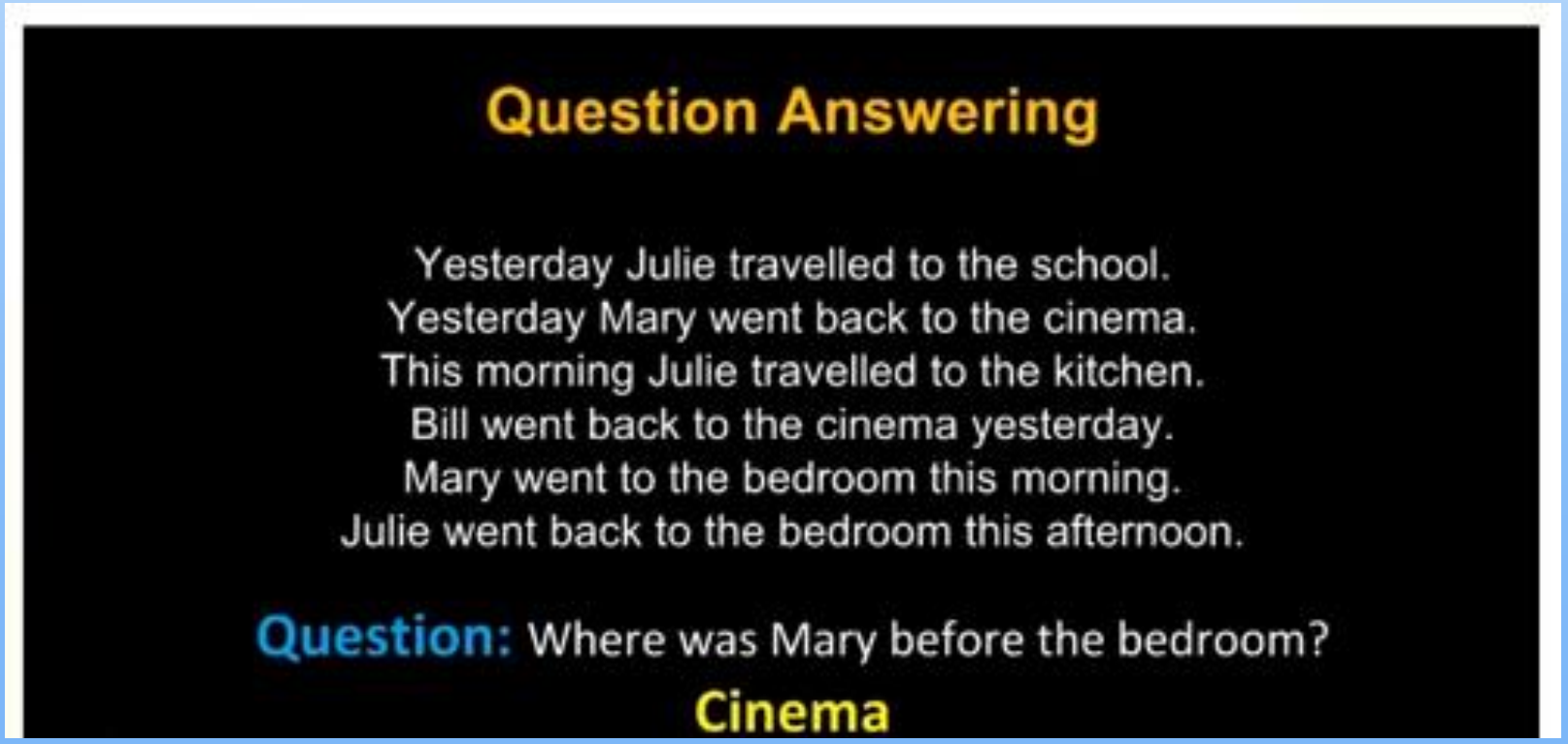
Type here to search

ENG 22:08 25/09/2017

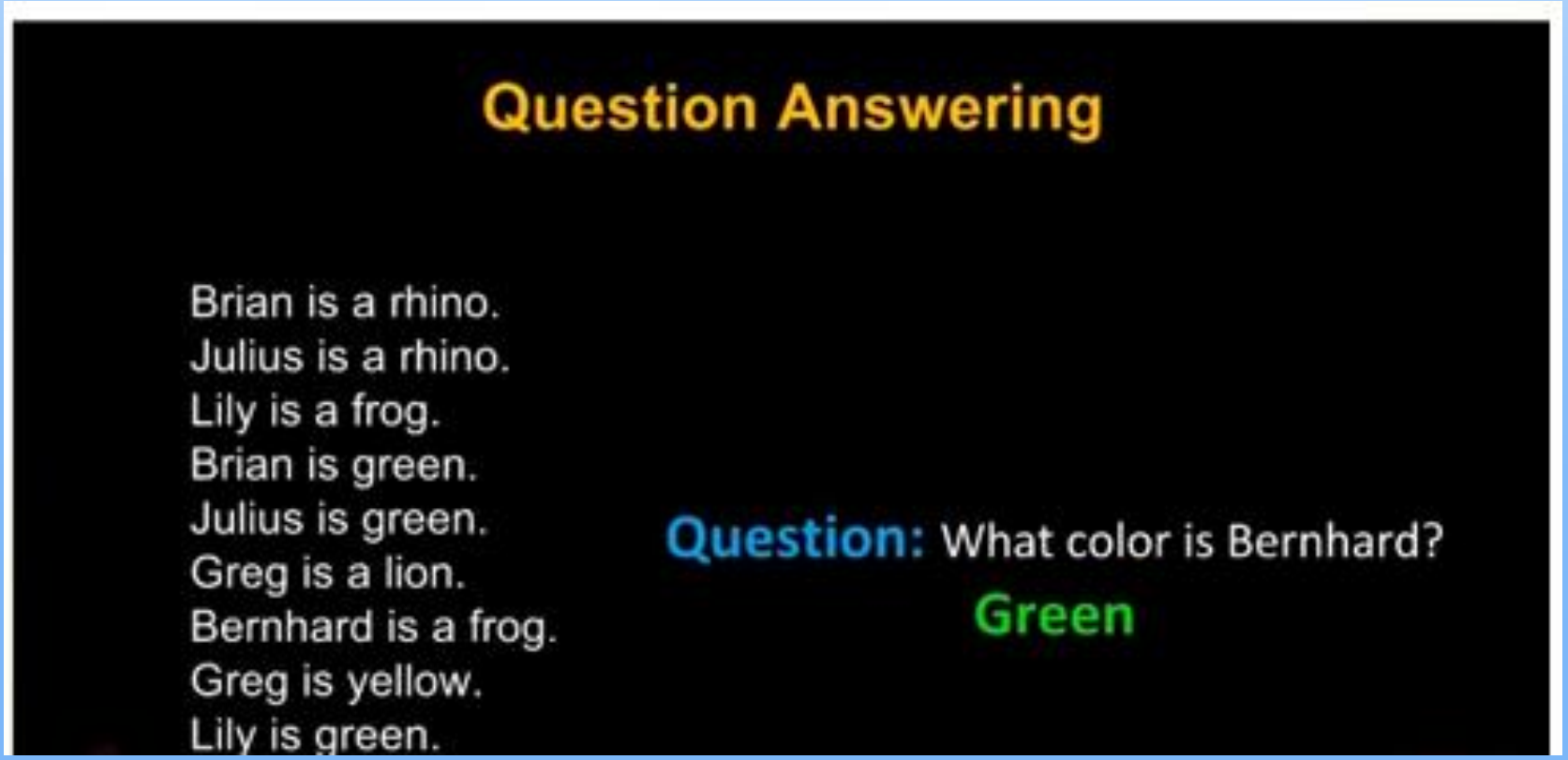
Applications



Applications



Applications



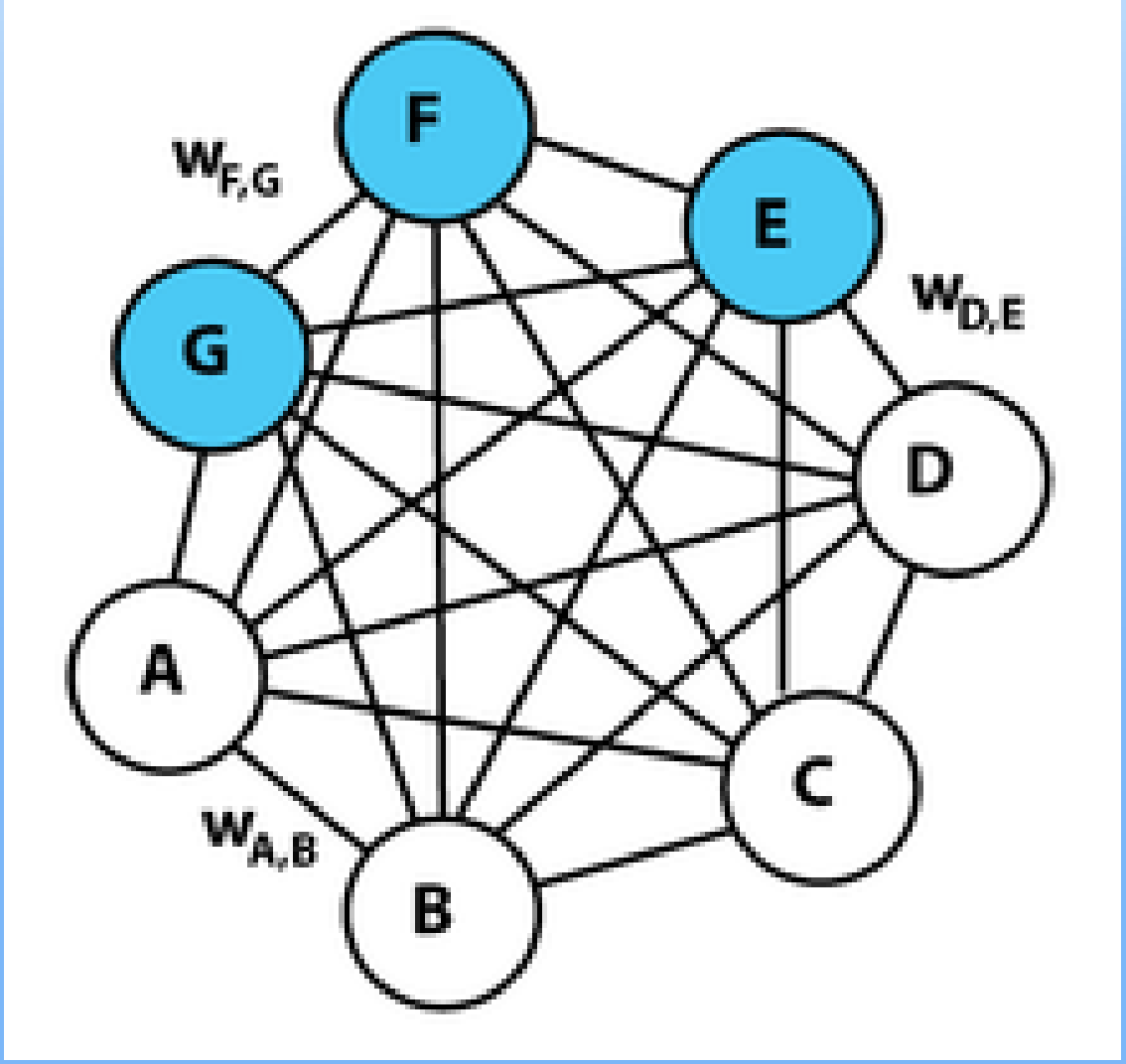
Applications

Applications

Boltzmann Learning Algorithm

- A Boltzmann machine, like a Hopfield network, is a network of units with an "energy" defined for the overall network.
- Its units produce binary results.
- The global energy, E, in a Boltzmann machine is identical in form to that of a Hopfield network

Boltzmann Learning



Boltzmann Learning

$$E = -\left(\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i\right)$$

w: weights s: outputs θ: Bias

Where:

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state, $s_i \in \{0, 1\}$, of unit i.
- θ_i is the bias of unit i in the global energy function. ($-\theta_i$ is the activation threshold for the unit.)

Often the weights are represented as a symmetric matrix W, with zeros along the diagonal.

Hopfield Network

- Hopfield nets serve as content-addressable memory systems with binarythreshold nodes.
- They are guaranteed to converge to a local minimum, but will sometimes converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum).
- Hopfield networks also provide a model for understanding human memory.

Disadvantages of Hopfield

- Hopfield networks suffer from spurious local minima that form on the energy hypersurface
- require the input patterns to be uncorrelated
- are limited in capacity of patterns that can be stored
- are usually fully connected and not stacked