Machine Learning using Python

ಡಾ|| ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್
Head Dept of CSE
SDMIT Ujire

Agenda

1. **Machine Learning Engineering - Introduction**
2. **Python Fundamentals – Hands On**
3. **Decision Tree Learning**
4. **Machine Learning Lab Problems and Programs**

Machine Learning Engineering

ಡಾ|| ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್
Head Dept of CSE
SDMIT Ujire

Contents

1. What is Machine Learning ?

- Study of algorithms that
  - Improve their performance **P**
  - At some task **T**
  - With Experience **E**
- Well-defined learning task<P,T,E>

Machine learning (ML) — a subset of artificial intelligence (AI) — is more than a technique for analyzing data. It's a system that is fueled by data, with the ability to learn and improve by using algorithms that provide new insights without being explicitly programmed to do so.

Figure 1. Learning From Data Without Being Explicitly Programmed

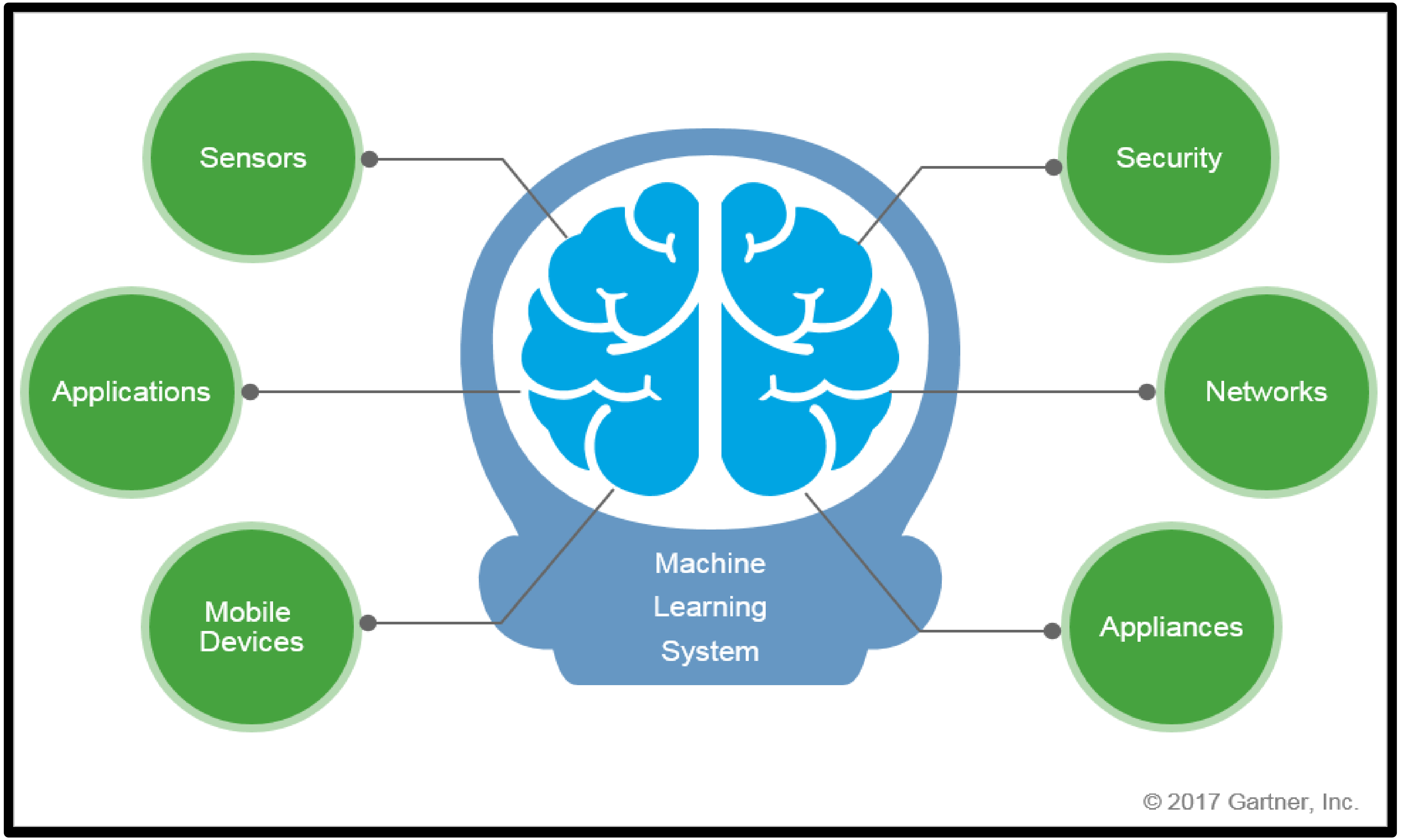
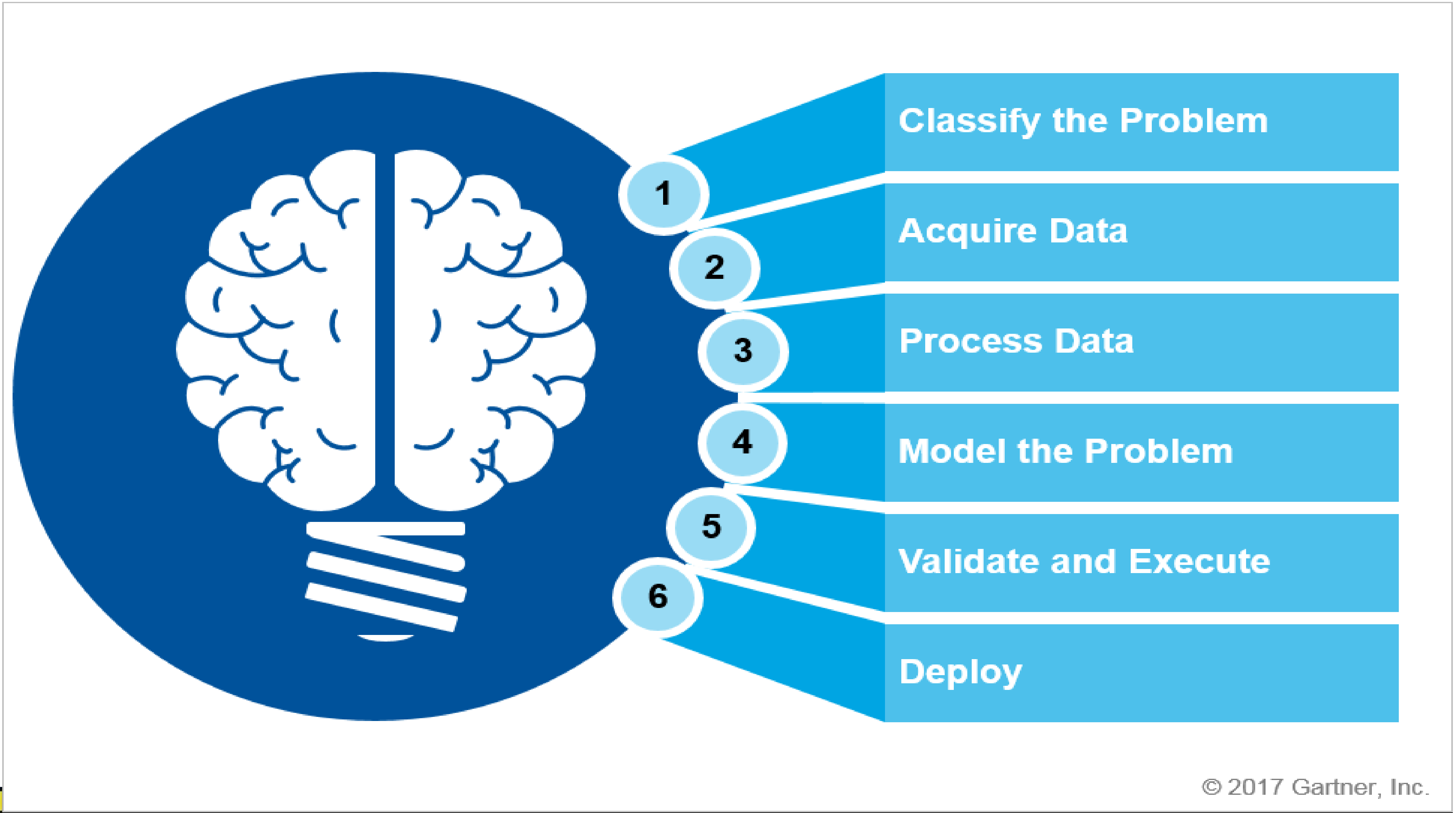


Figure 2. Stages of the Machine Learning Process



Context

2. Machine Learning Engineer

- Machine learning engineers are sophisticated programmers who ***develop machines and systems that can learn and apply knowledge*** without specific direction.
- The algorithms they create allow a machine to find patterns in its own programming data, teaching it to understand commands and even think for itself.

Example : An example of a system a machine learning engineer would work on is a ***self-driving car and*** *automatic vacuum cleaner****.***

3. Skills required for MLE

1. **Math Skills**
2. **Programming Skills**
3. **Data Engineering Skills**
4. **Knowledge of Machine Learning Algorithms**
5. **Knowledge of Machine Learning Frameworks**
6. **Knowledge of IoT and Cloud Computing**
7. **Excellent Communication Skills**

3.1. Math Skills

- **Probability and Statistics** : Descriptive Statistics , Bayes Rule and Random Variables , Probability Distributions , Sampling Hypothesis , Testing Regression and Decision Analysis.

- **Linear Algebra :** Matrices , Vector Spaces

- **Calculus :** Basics of Differential and Integral calculus

3.2. Programming Skills

- Coding Skills , Algorithms , Data Structure and OOPS Concepts
- Languages like Python , R , Java and C (Master of any one or two)

3.3. Data Engineering Skills

- Ability to work with large amount of Data , Data Preprocessing , Knowledge of SQL an No SQL .
- ETL (Extract , Transform and Load) Operations
- Data Analysis and Visualization Skills

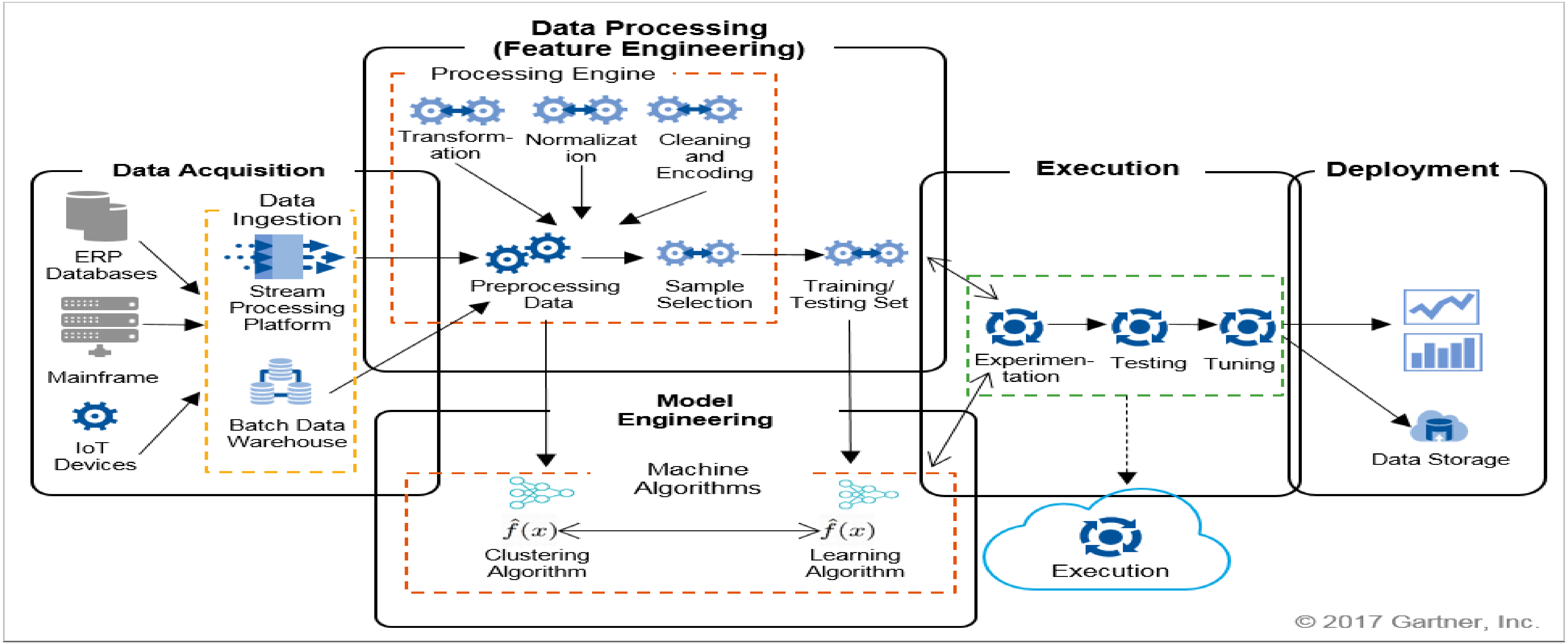
3.4. Knowledge of Machine Learning Algorithms

- Shallow and Deep Learning
- Supervised
- Unsupervised
- Semi Supervised
- Reinforcement

3.5. Knowledge of Machine Learning Frameworks

- Familiar with popular machine learning Frameworks such as
  - SCIKIT Learn,
  - TensorFlow ,
  - Azure ,
  - Caffe ,
  - Theano,
  - Spark and
  - Torch

4.Machine Learning Architecture



Terms Frequently Used

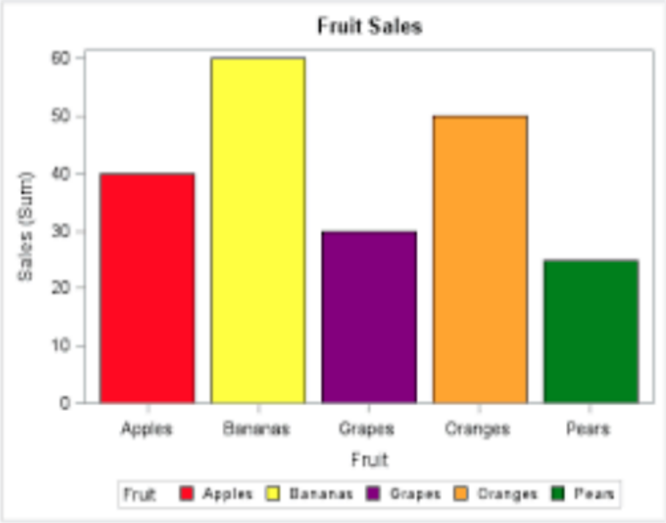
- **Accuracy** : Accuracy is a metric by which one can examine how good is the machine learning model.

$$\frac{\text{True Positive} + \text{True Negatives}}{\text{True Positive} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

- **Backpropagation:** In neural networks, if the estimated output is far away from the actual output (high error), we update the biases and weights based on the error. This weight and bias updating process is known as Back Propagation. 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.

Terms Frequently Used

- **Bar Chart :** Bar charts are a type of graph that are used to display and compare the numbers, frequency or other measures (e.g. mean) for different discrete categories of data. They are used for categorical variables. Simple example of a bar chart:



Terms Frequently Used

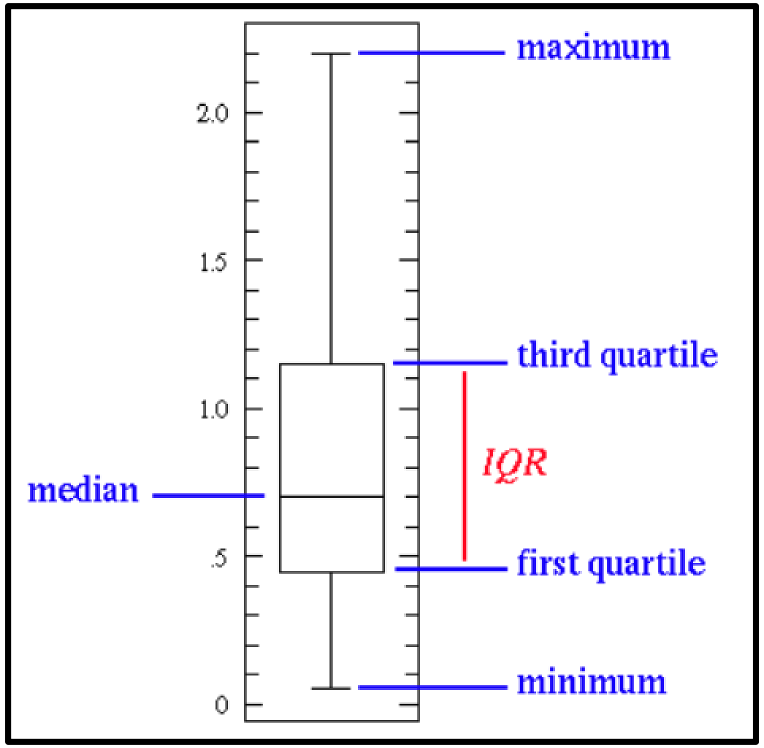
- **Bayes Theorem :** Bayes' theorem is used to calculate the conditional probability. Conditional probability is the probability of an event 'B' occurring given the related event 'A' has already occurred.

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

- **Boosting :** Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model. Some of the boosting algorithms are:
    - AdaBoost
    - GBM
    - XGBM
    - LightGBM
    - CatBoost

Terms Frequently Used

- **Box Plot :** It displays the full range of variation (from min to max), the likely range of variation (the Interquartile range), and a typical value (the median). Below is a visualization of a box plot:



- Some of the inferences that can be made from a box plot:
  - **Median:** Middle quartile marks the median.
  - **Middle box** represents the 50% of the data
  - **First quartile:** 25% of data falls below these line
  - **Third quartile:** 75% of data falls below these line.

Terms Frequently Used

- **Classification:** The goal is to predict discrete values, e.g. {1,0}, {True, False}, {spam, not spam}. It is supervised learning method where the output variable is a category, such as “Male” or “Female” or “Yes” and “No”.
  - For example: Classification Algorithms like Logistic Regression, Decision Tree, K-NN, SVM etc.
- **Clustering :** Clustering is an unsupervised learning method used to discover the inherent groupings in the data. For example: Grouping customers on the basis of their purchasing behaviour which is further used to segment the customers. And then the companies can use the appropriate marketing tactics to generate more profits.
  - Example of clustering algorithms: K-Means, hierarchical clustering, etc.

Terms Frequently Used

- **Confusion Matrix :**A confusion matrix is a table that is often used to describe the performance of a classification model. It is a N * N matrix, where N is the number of classes. We form confusion matrix between prediction of model classes Vs actual classes. The 2nd quadrant is called type II error or False Negatives, whereas 3rd quadrant is called type I error or False positives.

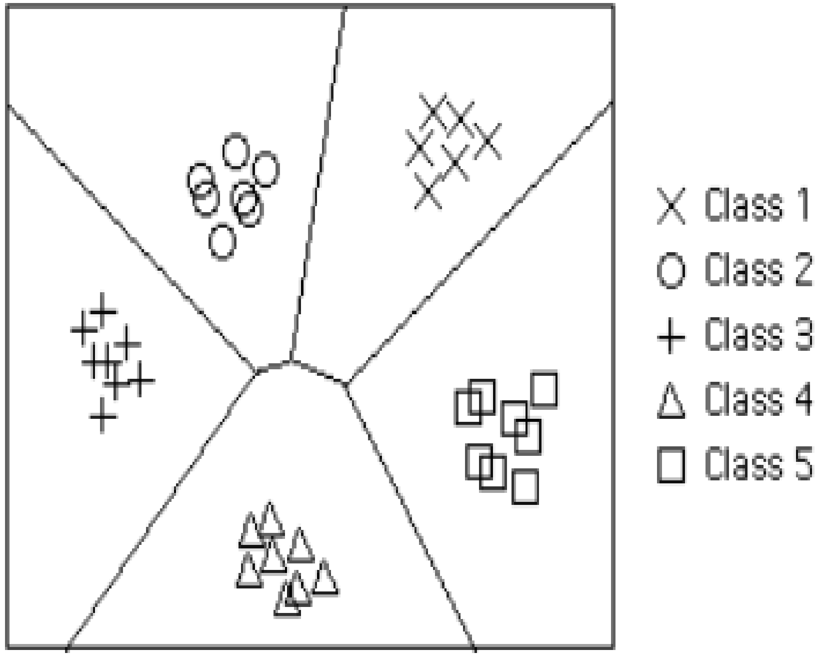
Confusion Matrix		**Target**			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Terms Frequently Used

- **Continuous Variable :** Continuous variables are those variables which can have infinite number of values but only in a specific range. For example, height is a continuous variable.
- **Convergence :**Convergence refers to moving towards union or uniformity. An iterative algorithm is said to converge when as the iterations proceed the output gets closer and closer to a specific value.
- **Cross Validation:** Cross Validation is a technique which involves reserving a particular sample of a dataset which is not used to train the model. Later, the model is tested on this sample to evaluate the performance. There are various methods of performing cross validation such as:
    - Leave one out cross validation (LOOCV)
    - k-fold cross validation
    - Stratified k-fold cross validation
    - Adversarial validation

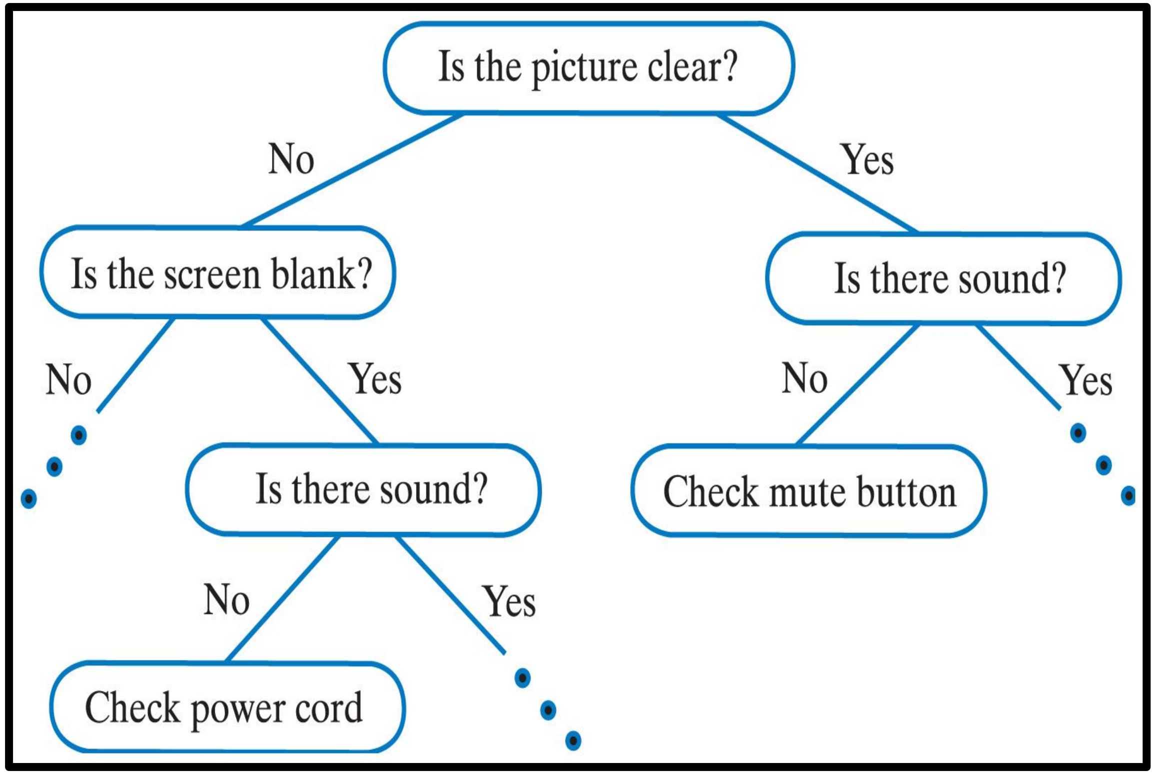
Terms Frequently Used

- **Decision Boundary :** In a statistical-classification problem with two or more classes, a decision boundary or decision surface is a hypersurface that partitions the underlying vector space into two or more sets, one for each class. How well the classifier works depends upon how closely the input patterns to be classified resemble the decision boundary. In the example sketched below, the correspondence is very close, and one can anticipate excellent performance.
- Here the lines separating each class are decision boundaries.



Terms Frequently Used

- **Decision Tree :** Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input & output variables. In this technique, we split the population (or sample) into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.
-



Terms Frequently Used

- **Deep Learning :** Deep Learning is associated with a machine learning algorithm (Artificial Neural Network, ANN) which uses the concept of human brain to facilitate the modeling of arbitrary functions. ANN requires a vast amount of data and this algorithm is highly flexible when it comes to model multiple outputs simultaneously.
- **Dependent Variable :** A dependent variable is what you measure and which is affected by independent / input variable(s). It is called dependent because it “depends” on the independent variable. For example, let’s say we want to predict the smoking habits of people. Then the person smokes “yes” or “no” is the dependent variable.
- **False Negative :** Points which are actually true but are incorrectly predicted as false. For example, if the problem is to predict the loan status. (Y-loan approved, N-loan not approved). False negative in this case will be the samples for which loan was approved but the model predicted the status as not approved.
- **False Positive:** Points which are actually false but are incorrectly predicted as true. For example, if the problem is to predict the loan status. (Y-loan approved, N-loan not approved). False positive in this case will be the samples for which loan was not approved but the model predicted the status as approved.

Terms Frequently Used

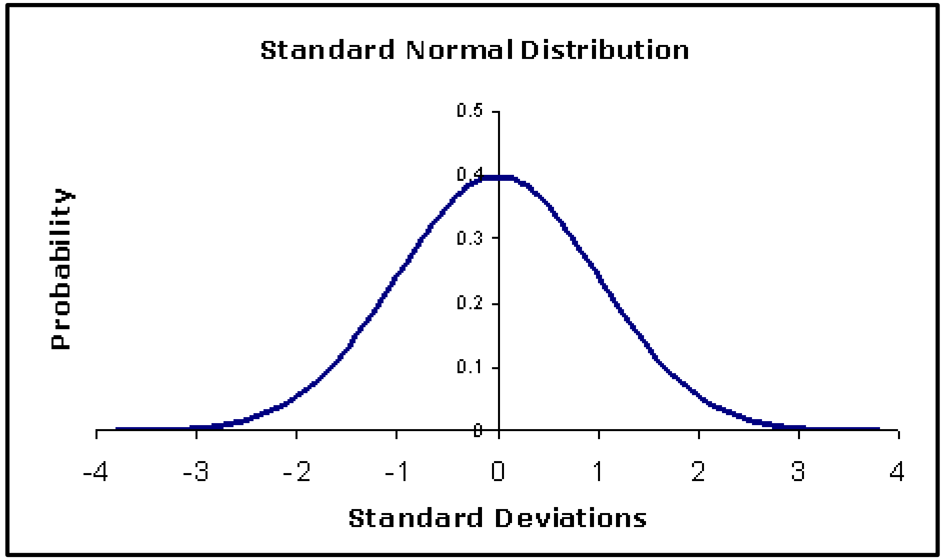
- **Gradient Descent:** Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. In machine learning algorithms, we use gradient descent to minimize the cost function. It find out the best set of parameters for our algorithm. Gradient Descent can be classified as follows:
- On the basis of data ingestion:
  - Full Batch Gradient Descent Algorithm
  - Stochastic Gradient Descent Algorithm
    - *In full batch gradient descent algorithms, we use whole data at once to compute the gradient, whereas in stochastic we take a sample while computing the gradient.*
- On the basis of differentiation techniques:
  - First order Differentiation
  - Second order Differentiation

Terms Frequently Used

- **Hypothesis :** Simply put, a hypothesis is a possible view or assertion of an analyst about the problem he or she is working upon. It may be true or may not be true
- **Labeled Data:** Data consisting of a set of *training examples,* where each example is a *pair* consisting of an input and a desired output value (also called the *supervisory signal, labels, etc*). A **label** is the thing we're predicting—the y variable in simple linear regression. The label could be the future price of wheat, the kind of animal shown in a picture, the meaning of an audio clip, or just about anything.
- **Maximum Likelihood Estimation :** It is a method for finding the values of parameters which make the likelihood maximum. The resulting values are called maximum likelihood estimates (MLE).
- **Multivariate Analysis :** Multivariate analysis is a process of comparing and analyzing the dependency of multiple variables over each other.

Terms Frequently Used

- **Naive Bayes :** It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier would consider all of these properties to independently contribute to the probability that this fruit is an apple.
- **Normal Distribution :** The normal distribution is the most important and most widely used distribution in statistics. It is sometimes called the bell curve, because it has a peculiar shape of a bell. Mostly, a binomial distribution is similar to normal distribution. The difference between the two is normal distribution is continuous.



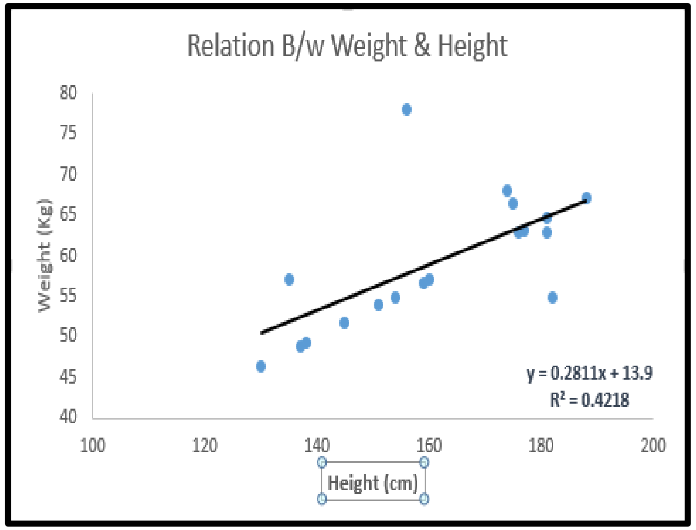
Terms Frequently Used

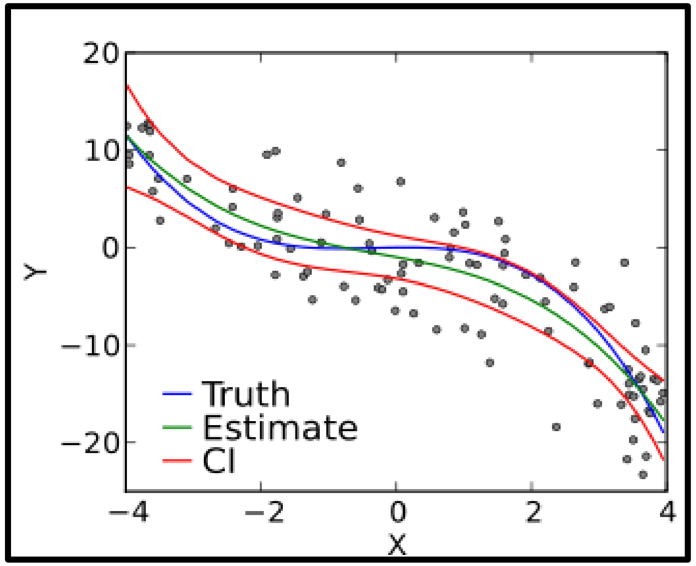
- **Normalization :** Normalization is the process of rescaling your data so that they have the same scale. Normalization is used when the attributes in our data have varying scales.
  - For example, if you have a variable ranging from 0 to 1 and other from 0 to 1000, you can normalize the variable, such that both are in the range 0 to 1.

- **Overfitting :** A model is said to overfit when it performs well on the train dataset but fails on the test set. This happens when the model is too sensitive and captures random patterns which are present only in the training dataset. There are two methods to overcome overfitting:
  - Reduce the model complexity
  - Regularization

Terms Frequently Used

- **Regression:** It is supervised learning method where the output variable is a real value, such as "amount" or "weight".
  - Example of Regression: Linear Regression, Ridge Regression, Lasso Regression
- **Linear Regression :** is a method for finding the straight line or hyperplane that best fits a set of points. **Y=aX+b ,** where:
  - Y – Dependent Variable
  - a – Slope
  - X – Independent variable
  - b – Intercept
- **Polynomial Regression :**In this technique, a curve fits into the data points. In a polynomial regression equation, the power of the independent variable is greater than 1. Although higher degree polynomials give lower error, they might also result in over-fitting.





Terms Frequently Used

- **Root Mean Squared Error (RMSE):** RMSE is a measure of the differences between values predicted by a model or an estimator and the values actually observed. It is the standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are. The formula for RMSE is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

- Here,
  - Predicted -> value predicted by the model
  - Actual -> observed values
  - N -> Total number of observations

Terms Frequently Used

- **Underfitting:** Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. It refers to a model that can neither model on the training data nor generalize to new data. An underfit model is not a suitable model as it will have poor performance on the training data.
- **Univariate Analysis :** Univariate analysis is comparing and analyzing the dependency of a single predictor and a response variable
- **True Negative :** These are the points which are actually false and we have predicted them false.
- **True Positive : T**hese are the points which are actually true and we have predicted them true.

Terms Frequently Used

- **Model:** A model defines the relationship between features and label. For example, a spam detection model might associate certain features strongly with "spam".
- **Training** means creating or **learning** the model. That is, you show the model labeled examples and enable the model to gradually learn the relationships between features and label
- Loss : **Loss** for a given example is also called squared error .
    - = Square of the difference between prediction and label
    - = (observation - prediction)^2
    - = (y - y')^2
- **Mean square error** (**MSE**) is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediction(x))^2$$

5. Types of Machine Learning Algorithms

1. Based on Depth of Learning
2. Based on Type of learning

5.1 Based on Depth of Learning

1. Shallow Learning
   - Algorithms with Few Layers
   - Better for Less Complex and Smaller Data sets
   - **Eg: Logistic Regression and Support vector Machines**
2. Deep Learning
   - New technique that uses many layers of neural network (a model based on the structure of human brain)
   - Useful when the target function is very complex and data sets are very large.

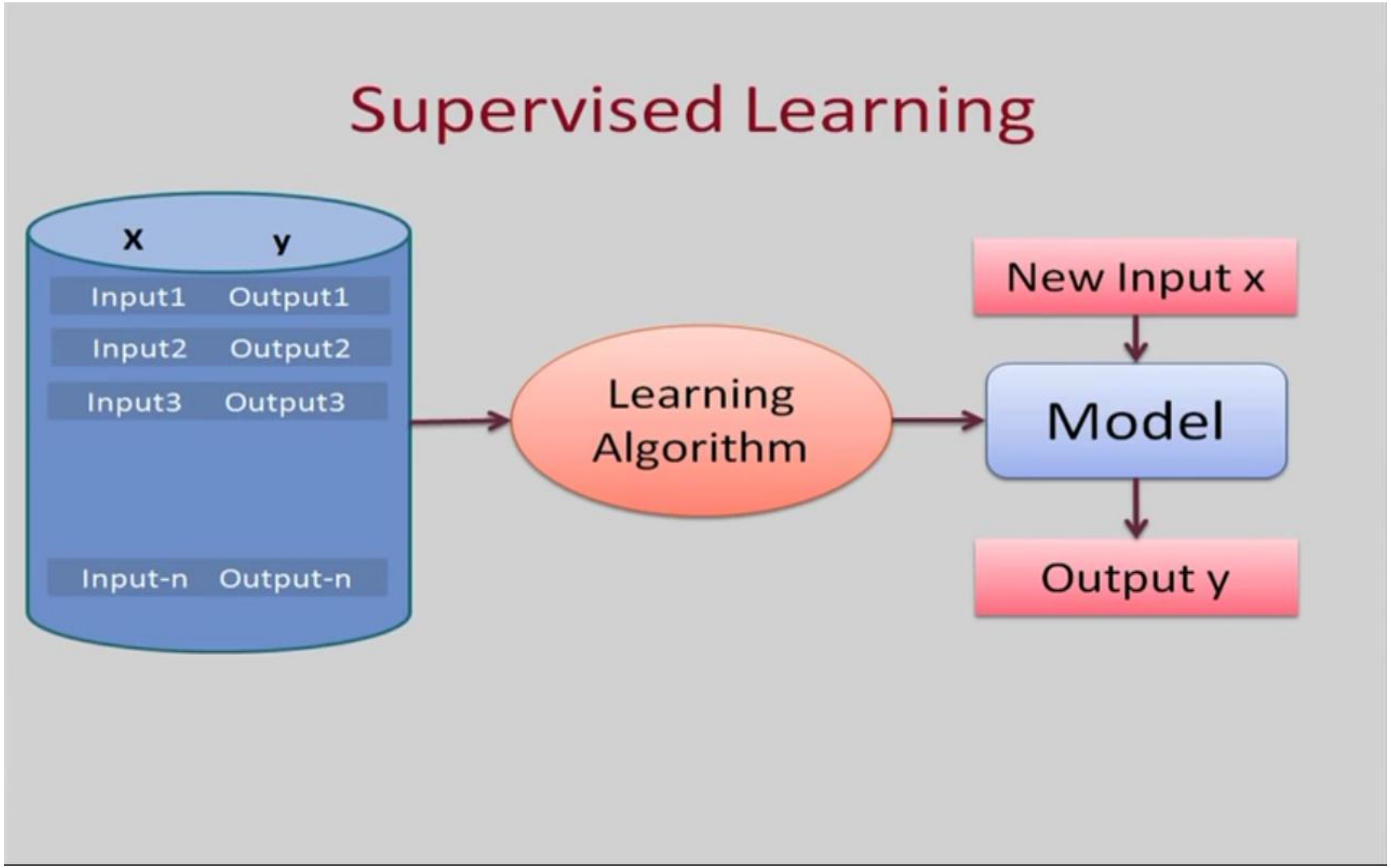
5.2 Based on Type of Learning

1. **Supervised Learning**
   - X and Y
   - Given an observation X what is the best label for Y
2. **Unsupervised Learning**
   - X
   - Given a set of X cluster or summarize them
3. **Semi Supervised Learning**
4. **Reinforcement Learning**
   - Determine what to do based on Rewards and punishments

5.2.1. Supervised Learning

- Here the training (labeled) data set containing ***input/predictors and output*** will be fed to the machine. The machine with the help of algorithm analyze the data set and ***generates a suitable model /function*** that best describes the input data. i.e it generates a function ***f(X) which makes best estimation of the output X for given X.***

- The generated model / function can be used to predict the output values for new data based on those relationships which it learned from the previous data sets.
  - When Y is discrete (True /False, ..) – **Classification**
  - When X is continuous (Real Numbers)- **Regression**


Supervised Learning
x
y
Input1 Output1
Input2 Output2
Input3 Output3
Input-n Output-n
Learning Algorithm
New Input x
Model
Output y

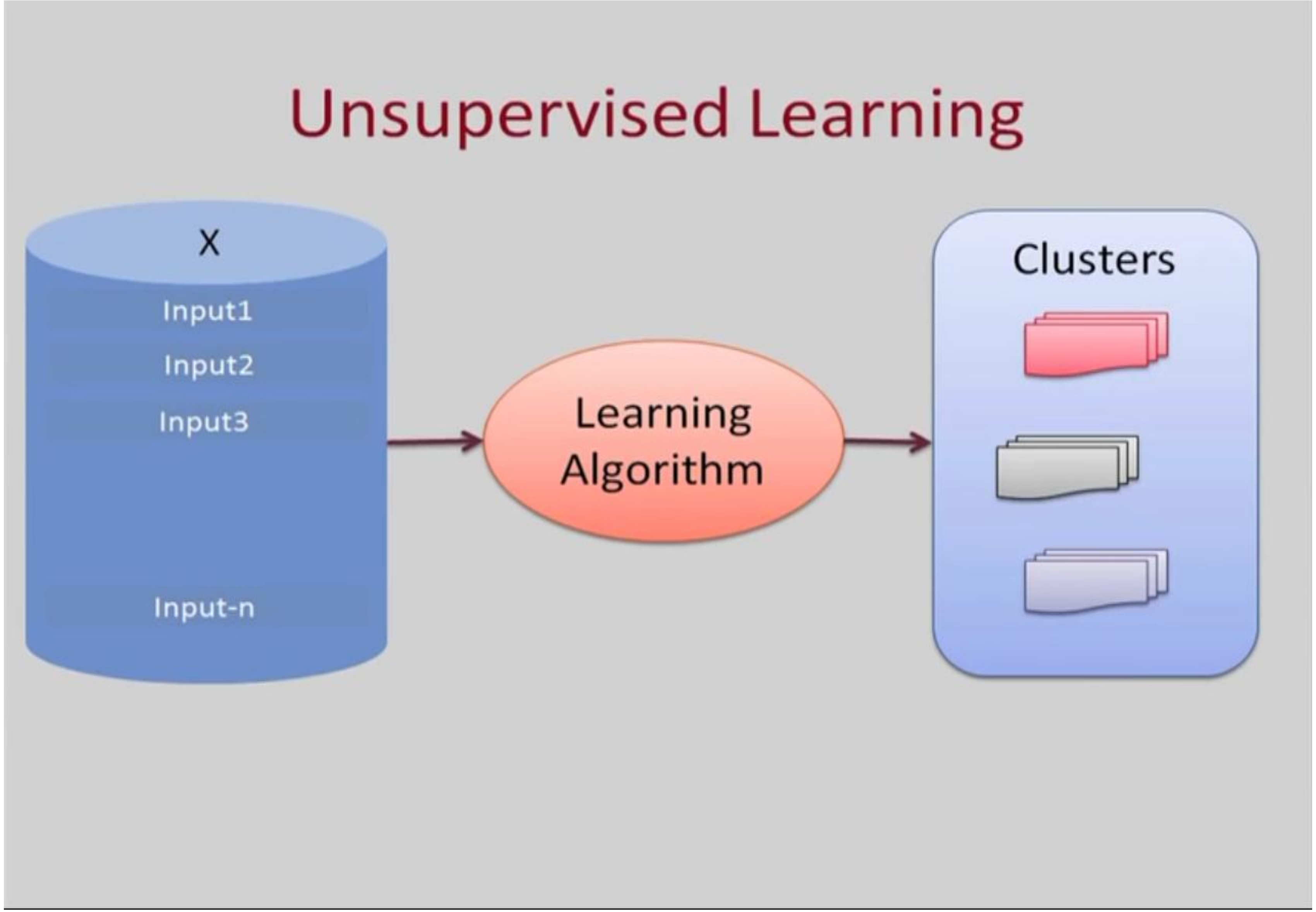
Types of Supervised Learning
(Task Driven . Develop Prediction Model based on Input and Output Data)

1. **Classification (Discrete)**
   a) Logistic Regression
   b) KNN
   c) Decision Trees
   d) Support Vector Machines
   e) Naïve Bayesian
   f) Discriminant Analysis
   g) Random Forest
   h) AdaBoost
   i) Neural Networks

2. **Regression (Continuous)**
   a) Linear Regression
   b) SVR
   c) GPR
   d) Ensemble Methods

5.2.2: Unsupervised Learning

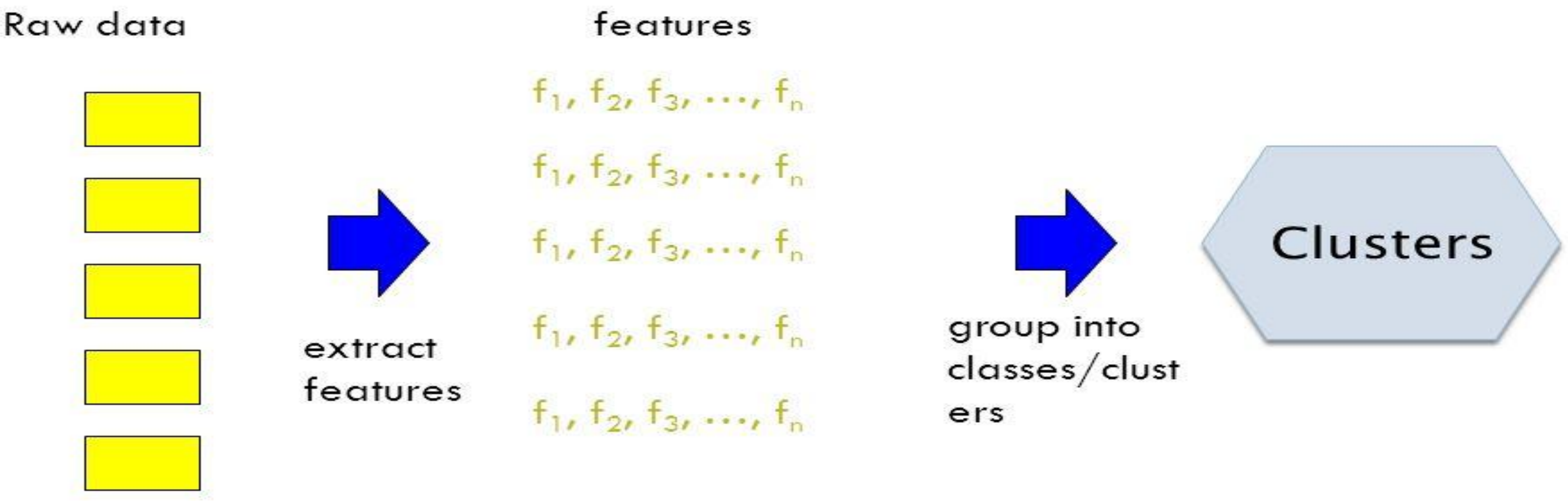
- This approach is data driven . The computer is trained with ***unlabeled*** input data.
- These algorithms try to use techniques on the input data to *mine for rules*, *detect patterns*, and *summarize and group the data points* which help in deriving meaningful insights and describe the data better to the users.


Unsupervised Learning
X
Input1
Input2
Input3
Input-n
Learning
Algorithm
Clusters

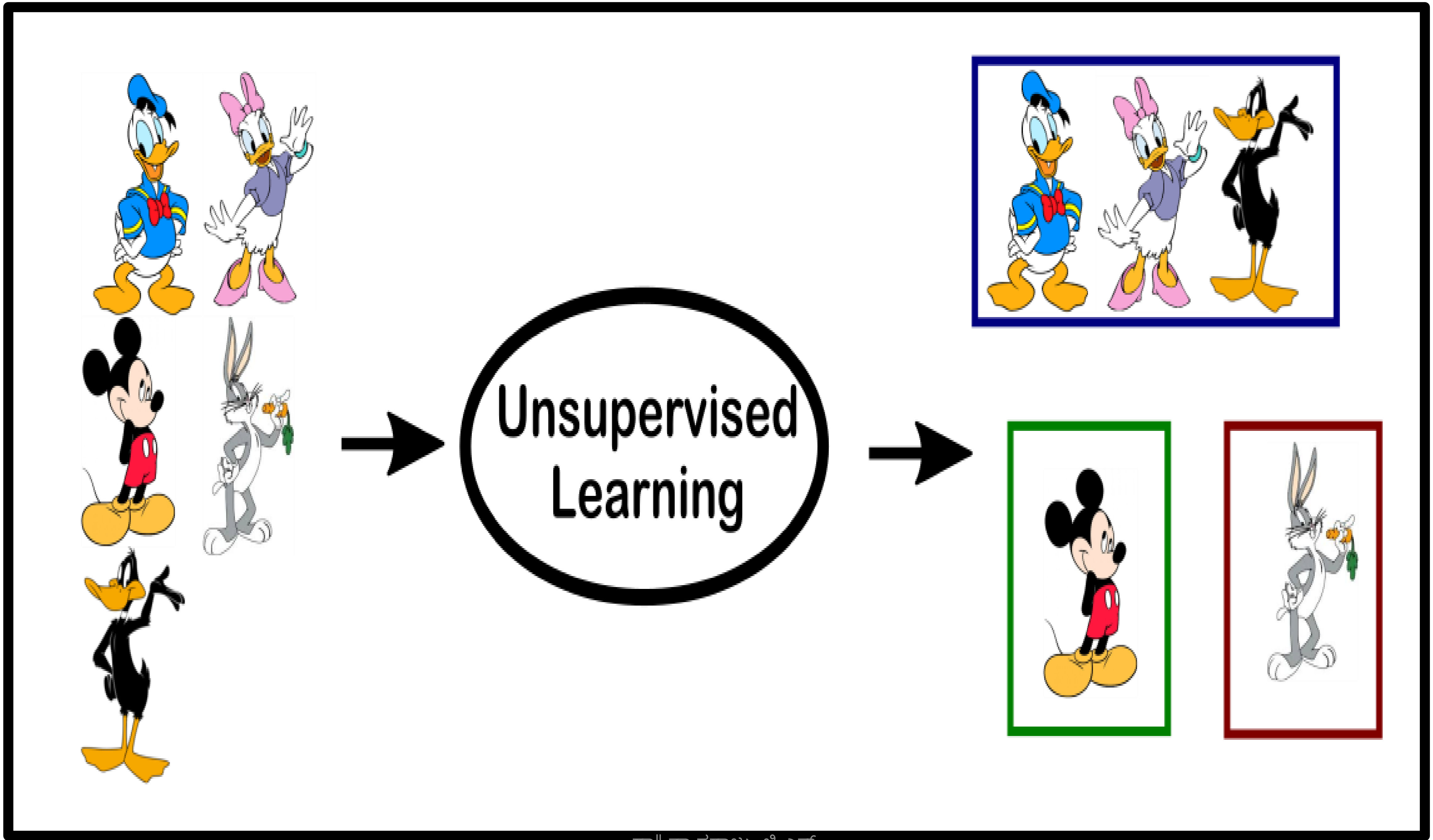
Types of Unsupervised Learning

1. Clustering
   - K Means Clustering
   - Hierarchical Clustering
   - Gaussian Mixture Models
   - Genetic Algorithms
   - Artificial Neural Networks
2. Dimensionality Reduction
   - Tensor Decomposition
   - Principal Component Analysis
   - Multidimensional statistics
   - Random Projection
   - Artificial Neural Networks
3. Association Rules

Unsupervised learning: clustering



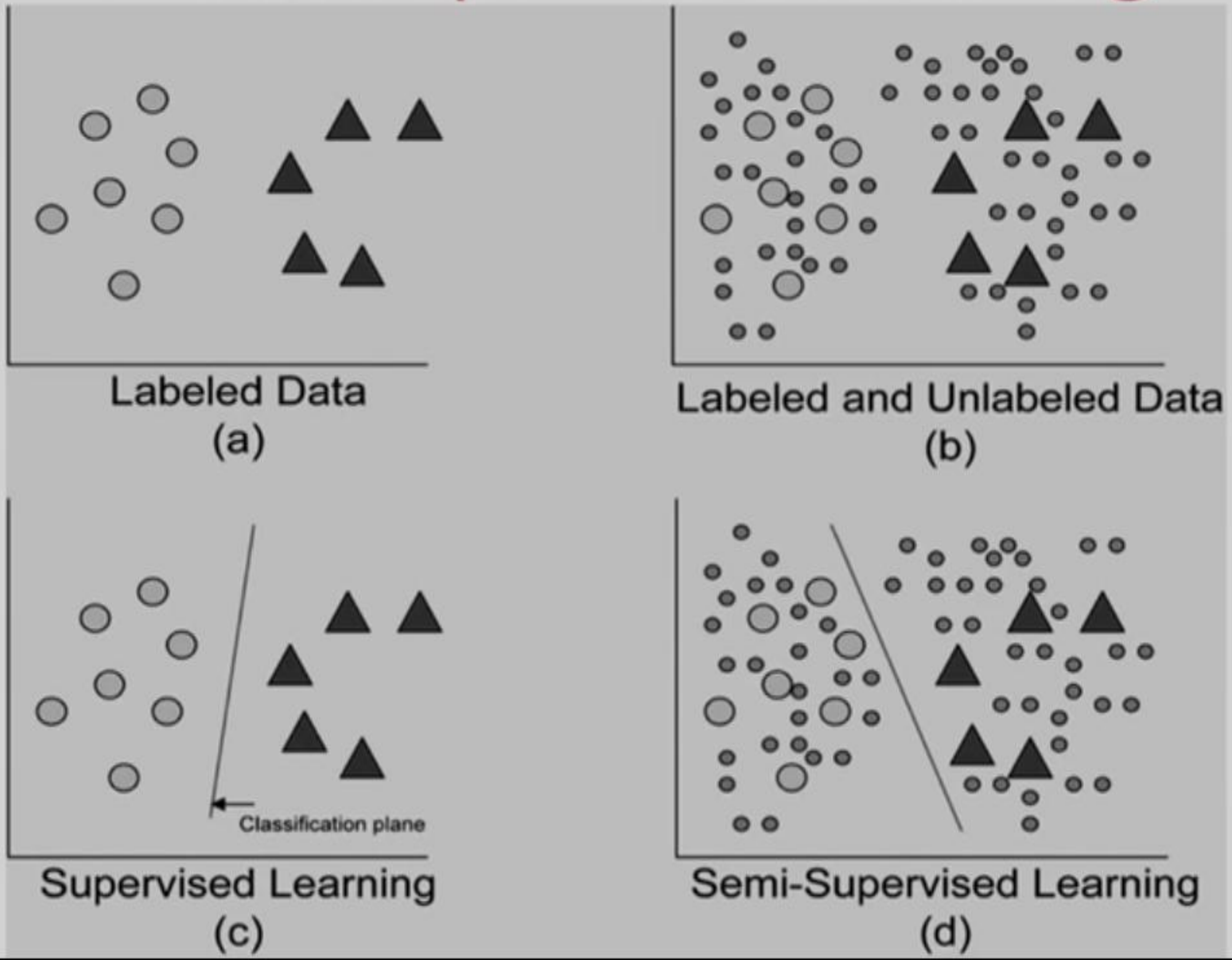
No "supervision", we're only given data and want to find natural groupings


Unsupervised
Learning

5.2.3. Semi Supervised

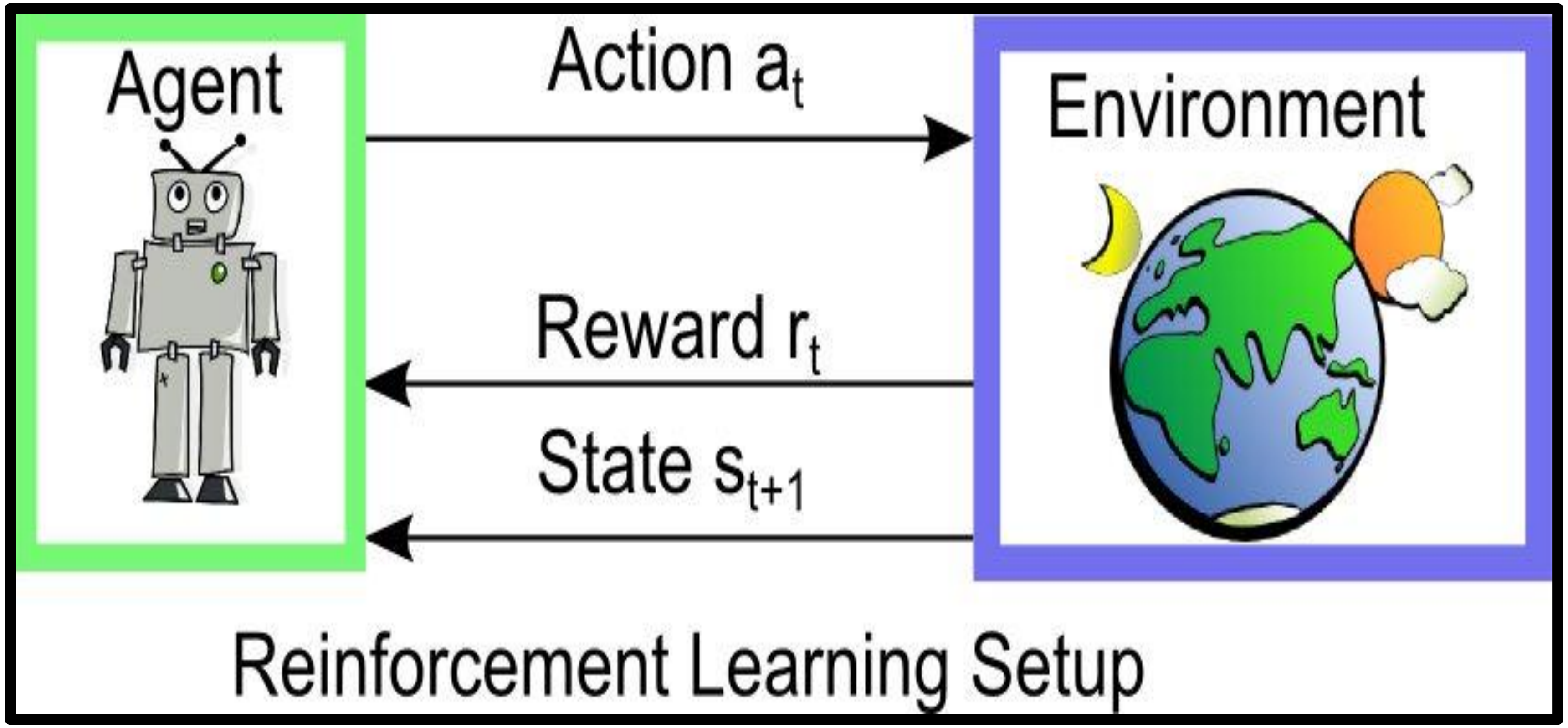
- In the previous two types, either there are no labels for all the observation in the dataset or labels are present for all the observations.
- Semi-supervised learning falls in between these two.

Semi-supervised learning



5.2.4. Reinforcement Learning

- This method aims at using observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk.
- Reinforcement learning algorithm (called the agent) continuously learns from the environment in an iterative fashion.


Agent
Action a_t
Environment
Reward r_t
State s_{t+1}
Reinforcement Learning Setup

Types of Reinforcement Learning Algorithms

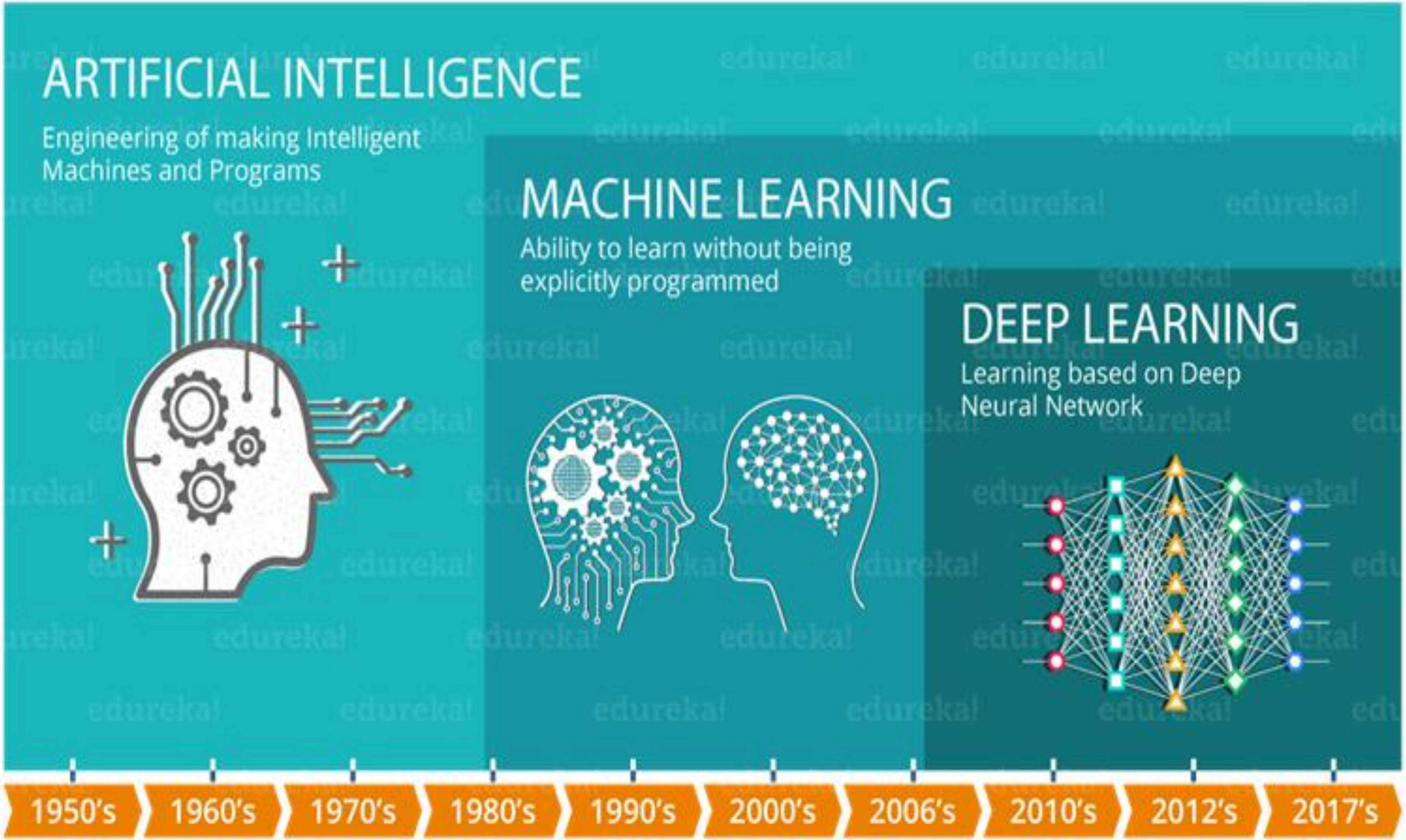
- **Q-Learning**
- **Temporal Difference (TD)**
- **Deep Adversarial Networks**

Table1: Examples of Types of Machine Learning Algorithms / Problem Solving Approaches

Type	Model /Algorithm or Task	Usage Examples in Business
Supervised	Neural network	■ Predicting financial results ■ Fraud detection
Supervised	Classification and/or Regression	■ Spam filtering ■ Fraud detection
Supervised	Decision tree	■ Risk assessment ■ Threat management systems ■ Any optimization problem where an exhaustive search is not feasible

Table1: Examples of Types of Machine Learning Algorithms / Problem Solving Approaches

Type	Model /Algorithm or Task	Usage Examples in Business
Unsupervised	Cluster analysis	■ Financial transactions ■ Streaming analytics in IoT ■ Underwriting in insurance
Unsupervised	Pattern recognition	■ Spam detection ■ Biometrics ■ Identity management
Unsupervised	Association rule learning	■ Security and intrusion detection ■ Bioinformatics ■Manufacturing and Assembly


ARTIFICIAL INTELLIGENCE
Engineering of making Intelligent Machines and Programs
MACHINE LEARNING
Ability to learn without being explicitly programmed
DEEP LEARNING
Learning based on Deep Neural Network
1950's
1960's
1970's
1980's
1990's
2000's
2006's
2010's
2012's
2017's

6. Open Source and Commercial Tools

Open Source Tools

1. Scikit Learn
2. Shogun
3. Accord.NET Framework
4. Spark MLlib
5. H20
6. Coudera Oryx
7. GoLearn
8. Weka
9. Deep Learn.js
10. ConvNet.Js
11. OpenAI
12. TensorFlow
13. Keras
14. Charnn
15. Paddle
16. CNTK
17. R
18. Monte Carlo ML Library
19. Octave Forge

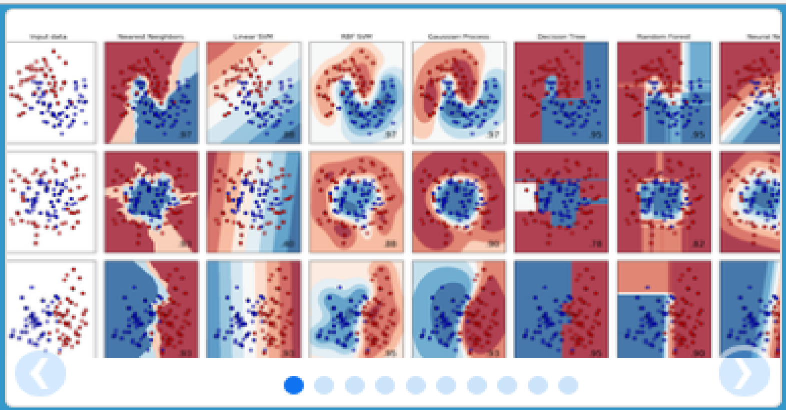
Commercial Tools

1. Microsoft Azure Machine Learning
2. SAS Enterprise Miner
3. IBM SPSS Modeler
4. RapidMiner
5. Apache Mahout
6. MATLAB
7. Oracle Data Mining

7. Introduction to Scikit Learn –
A Python Machine Learning Library

Scikit Learn : http://scikit-learn.org/stable/index.html

scikit-learn.org/stable/index.html



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

What is scikit-learn?

- Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
- It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.
- The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn.

Example : Scikit learn Machine Learning

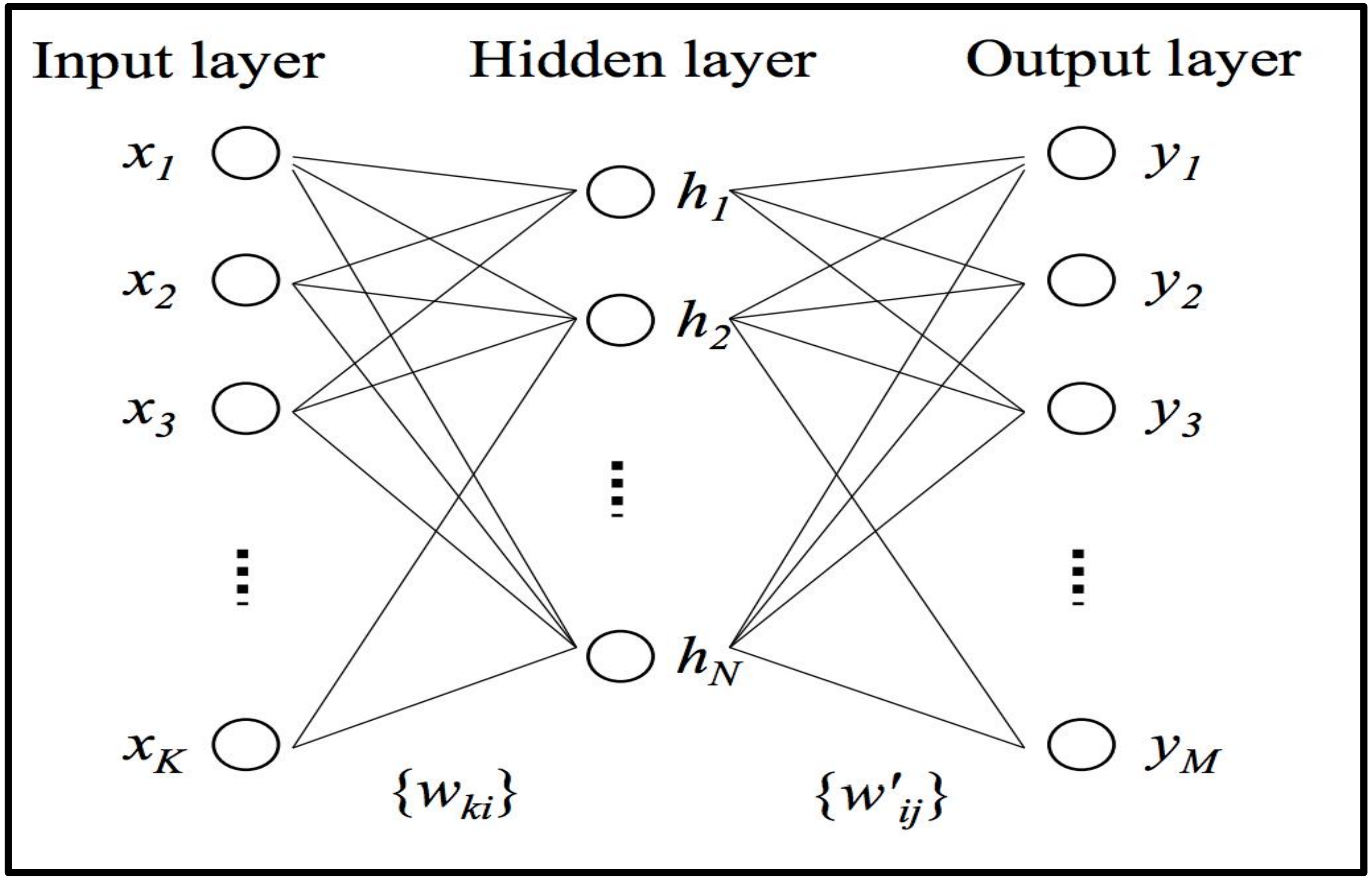
There are 5 key libraries that you will need to install. Below is a list of the Python SciPy libraries required for this tutorial:

1. scipy
2. numpy
3. matplotlib
4. pandas
5. sklearn

8. Introduction to Deep Learning

- Deep learning is a type of machine learning that is based on algorithms with extensive connections or layers between inputs and outputs.
- ML neural nets have inputs (variables), hidden layers (functions that compute the output) and output (results).
- In a simple example, imagine an ML neural net to detect a dog. This seemingly simple example may include a tail detector, ear detector, hair detector and so on.
- These detectors are combined into layers that contribute to detecting a dog. ***The more "detectors" you have, the deeper the neural net***.

Deep Learning



Deep Learning Tools

- Neural Designer
- Torch
- Apache SINGA
- Microsoft Cognitive ToolKit
- Keras
- Deeplearning4j
- Theano
- MXNet
- H2O.ai
- ConvNetjs
- DeepLEarningkit
- Gensim
- Caffe
- ND4J
- DeepLearnToolBox

2.Python Fundamentals

Python Introduction

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- It was created by *Guido van Rossum* during 1985- 1990.
- Python is named after a TV Show called '*Monty Python's Flying Circus*' and not after Python-the snake.

Features of Python

- **High-level Language and Simple**
- **Free and Open Source**
- **Portable**
- **Interactive and Interpreted**
- **Object Oriented**
- **Scalable**
- **Embeddable**
- **Extensive Libraries**
- **Database Support**
- **Programming GUI/APP**

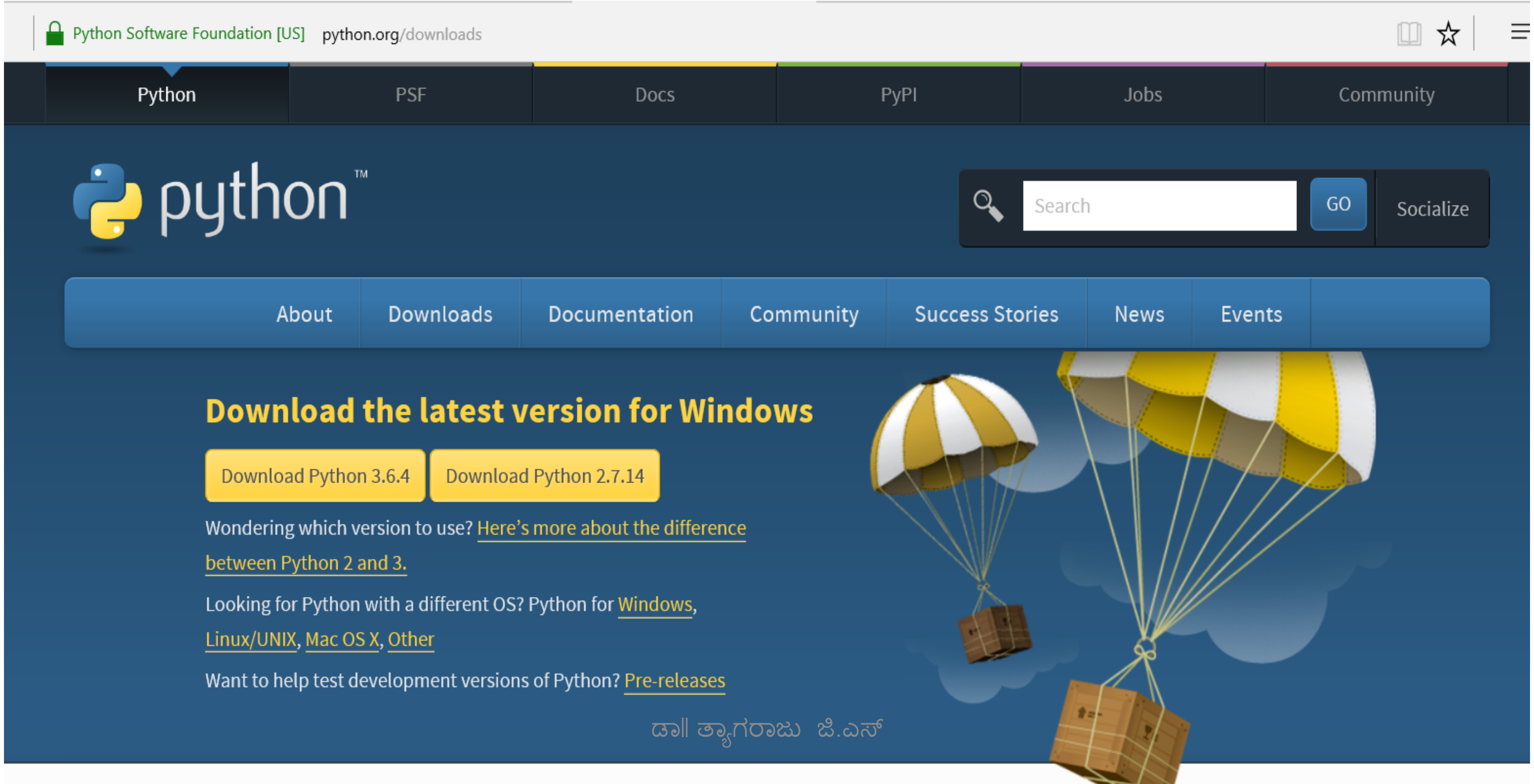
Programming Editors

- Python 3.6 Command Prompt
- Python 3.6 Idle
- Anaconda Prompt
- **Anaconda Jupyter Notebook**
- Anaconda Spider
- **JetBrains PyCharm**

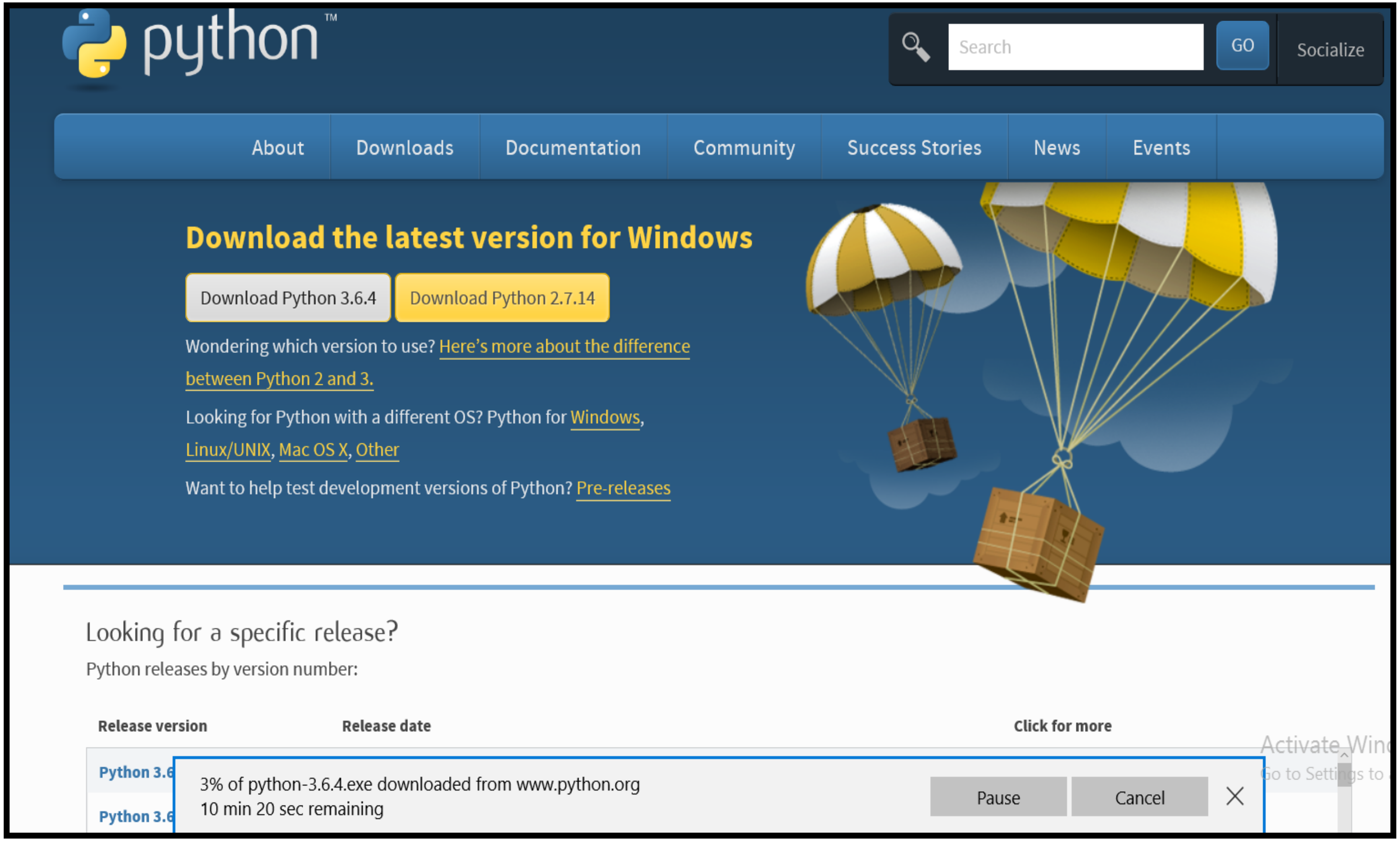
Python 3.6

Python3.6 Installation

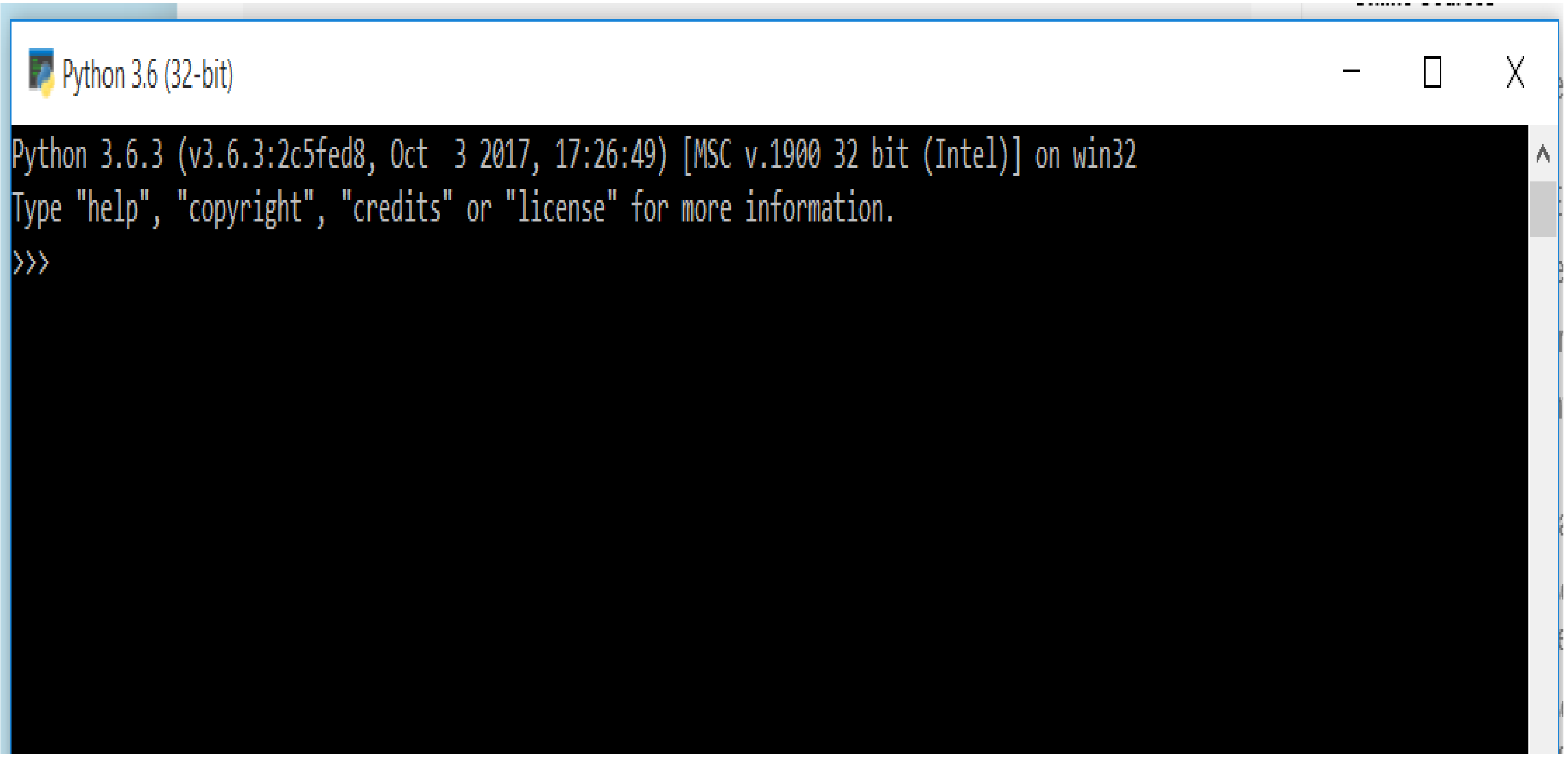
- Go to web page : https://www.python.org/downloads/

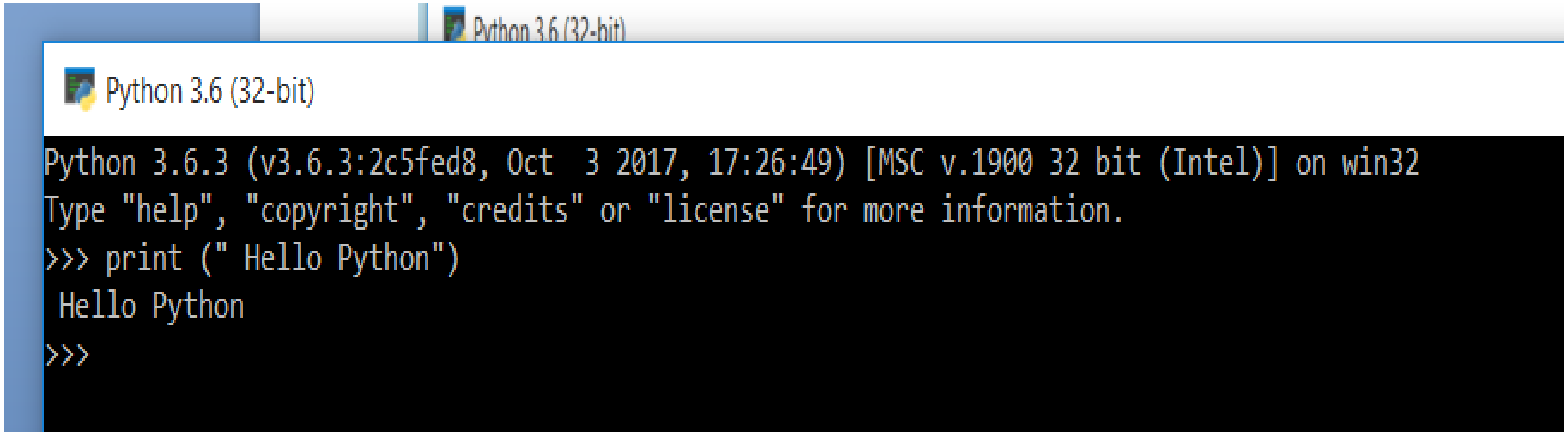


ಡಾ॥ ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್


python™
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.6.4
Download Python 2.7.14
Wondering which version to use? Here's more about the difference between Python 2 and 3.
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Pre-releases
Looking for a specific release?
Python releases by version number:
Release version
Release date
Click for more
3% of python-3.6.4.exe downloaded from www.python.org
10 min 20 sec remaining
Pause
Cancel

Interactive Python Programming




Python 3.6 (32-bit)
Python 3.6 (32-bit)
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print (" Hello Python")
Hello Python
>>>

What is Anaconda?

- The open source Anaconda Distribution is the easiest way to do Python data science and machine learning.
- It includes hundreds of popular data science packages and the *conda* package and virtual environment manager for Windows, Linux, and MacOS.

Use Python for…

- **Web Development**: Django, Pyramid, Bottle, Tornado, Flask, web2py
- **GUI Development**: tkInter, PyGObject, PyQt, PySide, Kivy, wxPython
- **Scientific and Numeric**: SciPy, Pandas, IPython
- **Software Development**: Buildbot, Trac, Roundup
- **System Administration**: Ansible, Salt, OpenStack

Some Python Packages/Libraries

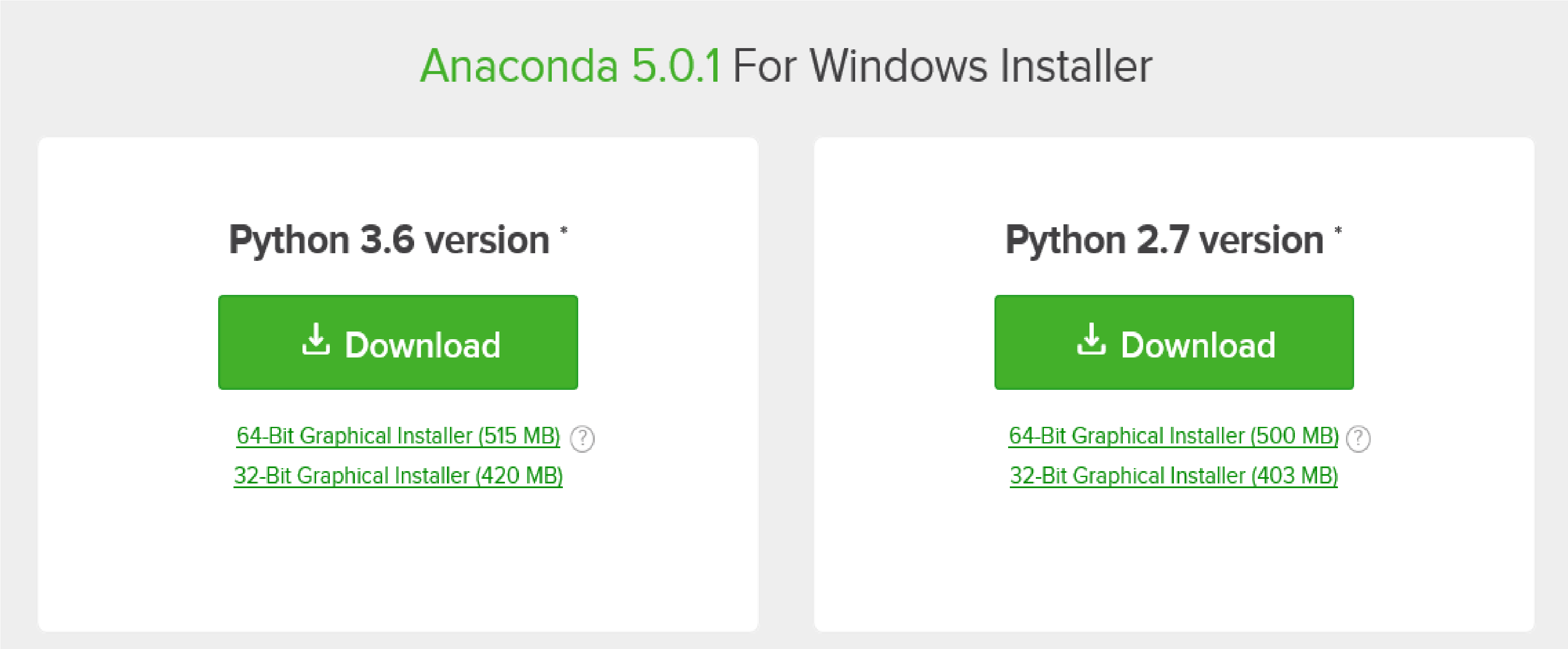
- Numpy
- Scipy
- Pandas
- Matplotlib
- Seaborn
- Bokeh
- Scikit Learn
- Pygames
- dJango
- Flask
- Bottle
- Pyramid
- PyBrain
- PyMongo

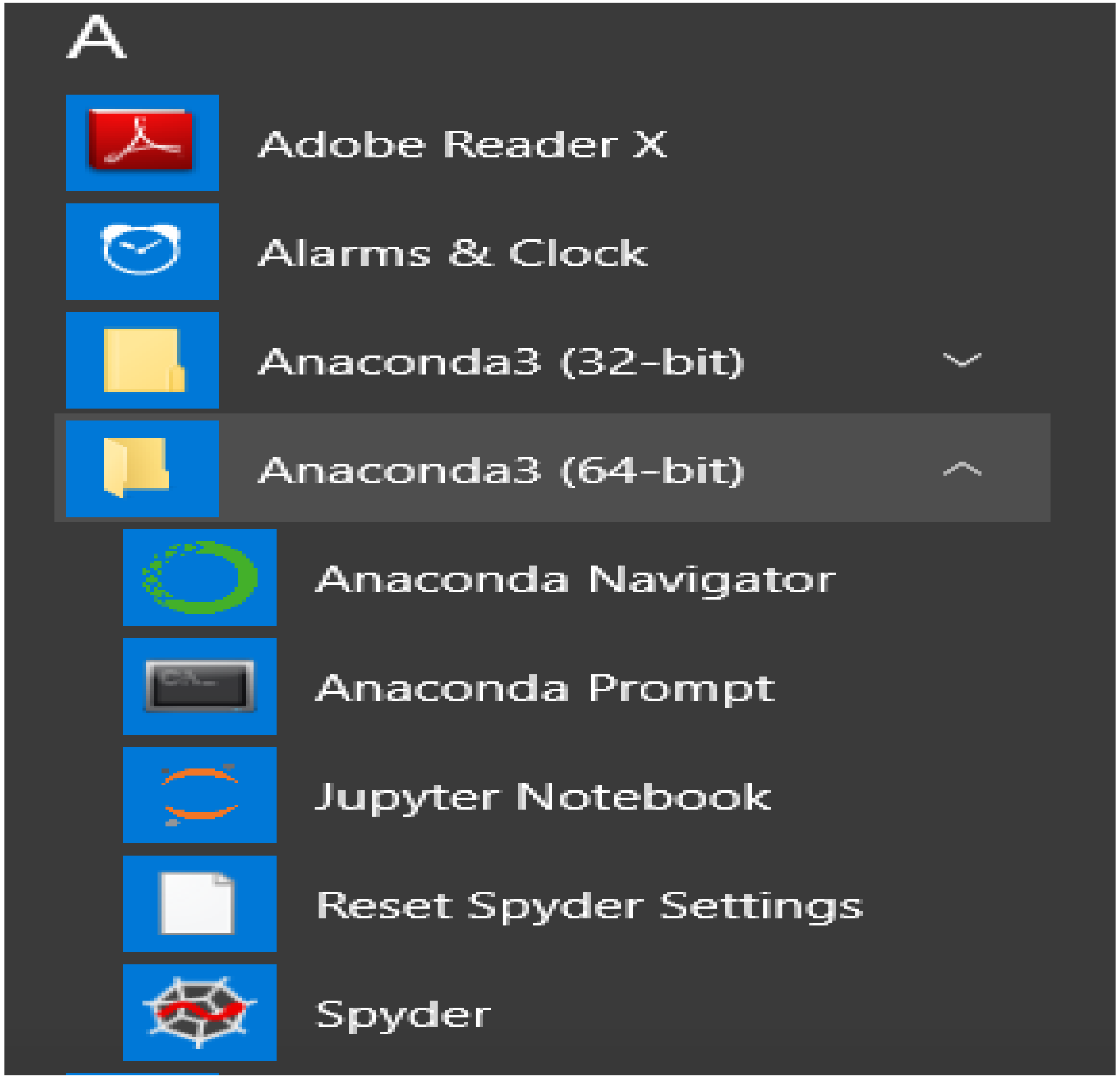
- Tornado
- Web2py
- Json
- tKinter
- OpenCV
- PyGObject
- PyQt
- wxPython
- Kivy
- Buildozer
- Buildbot
- Trac
- Roundup

- Ansible
- Salt
- OpenStack
- Keras
- Tensor Flow
- Theano
- nltk
- Spacy
- TextBlog
- ScikitLearn
- Pattern
- SQLalchemy
- pyMySql

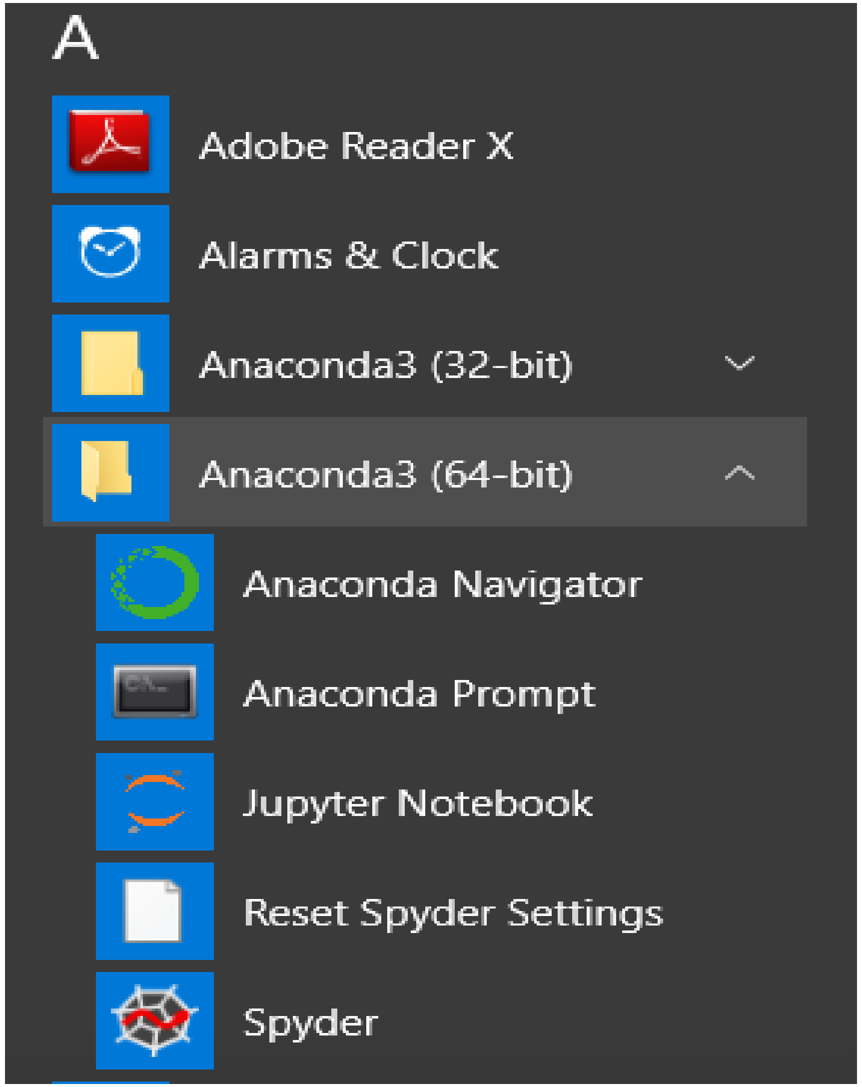
Downloading Anaconda

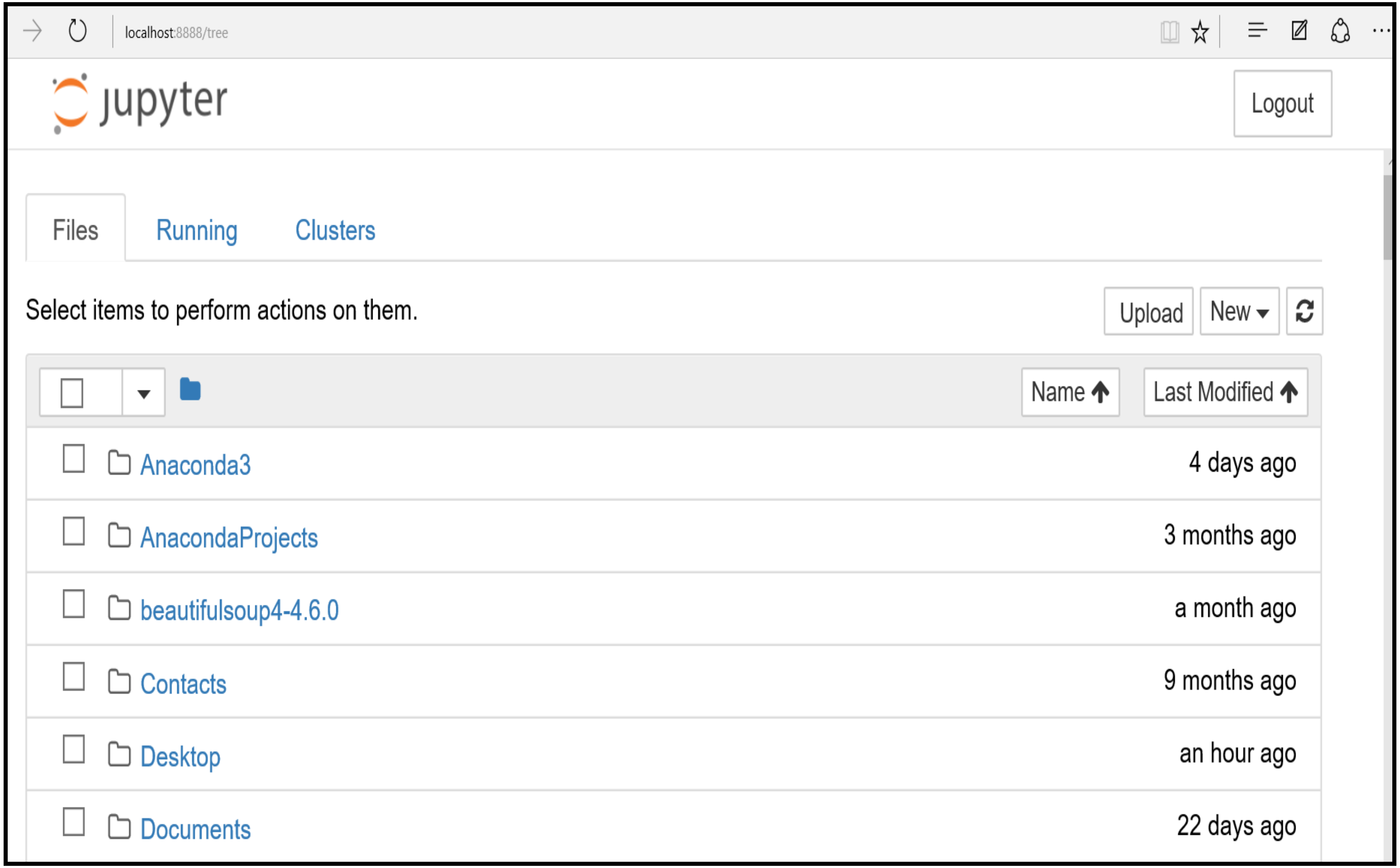
- The latest version of Anaconda is available for download from **https://www.anaconda.com/download/**

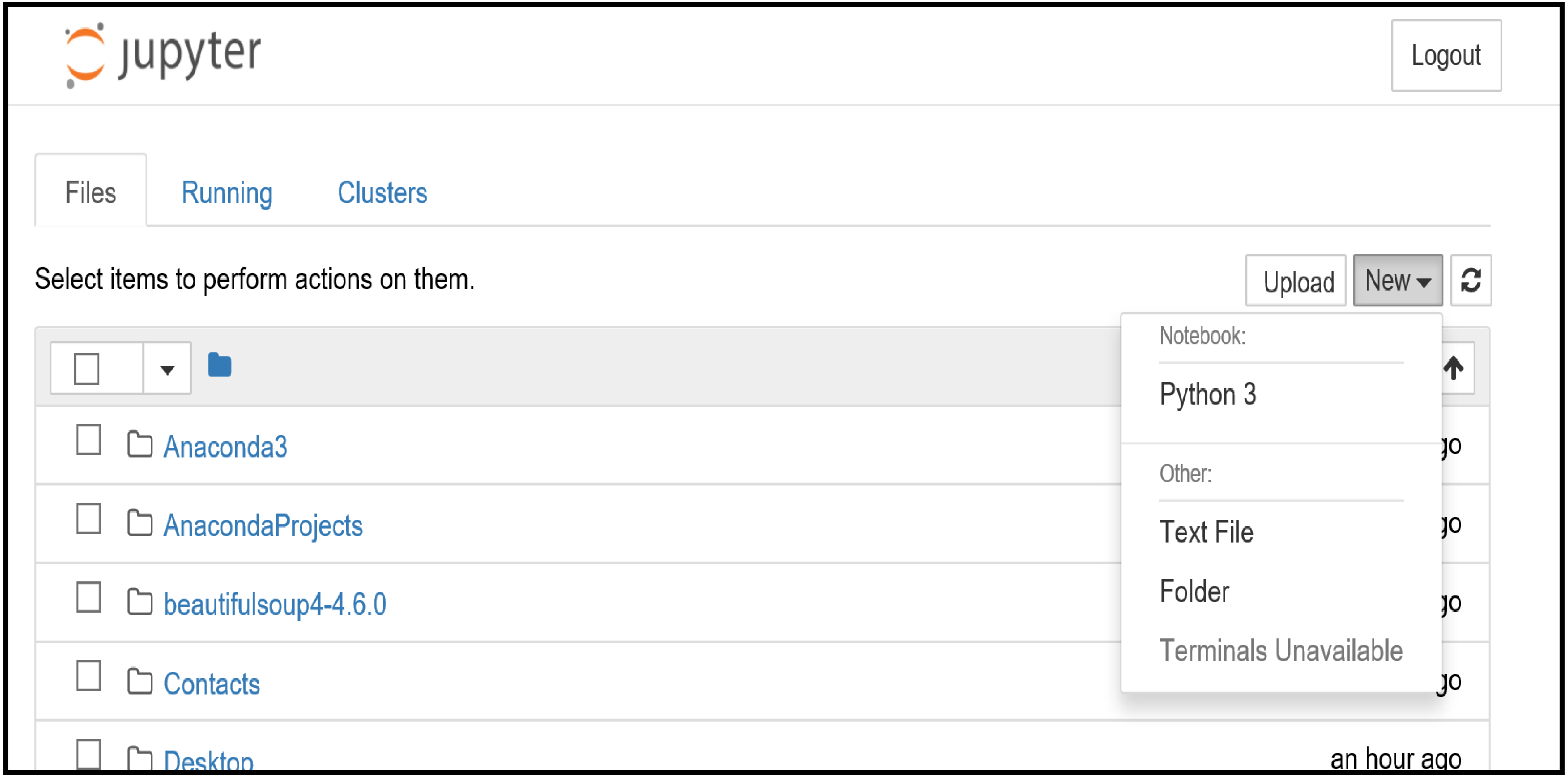



A
Adobe Reader X
Alarms & Clock
Anaconda3 (32-bit)
Anaconda3 (64-bit)
Anaconda Navigator
Anaconda Prompt
Jupyter Notebook
Reset Spyder Settings
Spyder

JUPYTER NOTEBOOK




localhost:8888/tree
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Name
Last Modified
Anaconda3
4 days ago
AnacondaProjects
3 months ago
beautifulsoup4-4.6.0
a month ago
Contacts
9 months ago
Desktop
an hour ago
Documents
22 days ago


jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Notebook:
Python 3
Other:
Text File
Folder
Terminals Unavailable
Anaconda3
AnacondaProjects
beautifulsoup4-4.6.0
Contacts
Desktop
an hour ago


jupyter
Untitled7 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Code
In [ ]:


jupyter
Python Foss Examples (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Code
In [ ]:

jupyter First Program Last Checkpoint: 11 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Code

In [ ]:

```
# Program to find the simple interest

p = int(input("Enter the principal amount "))

t = int(input ("Enter the time period"))

r = float(input("Enter the rate of interest"))

si = p*t*r/100

print("The simple interest = ",si)
```

jupyter First Program Last Checkpoint: 16 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Code

```
In [*]: # Program to find the simple interest

        p = int(input("Enter the principal amount "))

        t = int(input ("Enter the time period"))

        r = float(input("Enter the rate of interest"))

        si = p*t*r/100

        print("The simple interest = ",si)

Enter the principal amount 1000
Enter the time period2

Enter the rate of interest 2.56
```

```
In [1]: # Program to find the simple interest

p = int(input("Enter the principal amount "))

t = int(input ("Enter the time period"))

r = float(input("Enter the rate of interest"))

si = p*t*r/100

print("The simple interest = ",si)
```

```
Enter the principal amount 1000
Enter the time period 2
Enter the rate of interest 2.56
The simple interest =  51.2
```

Data Types in Python

Floating Point Numbers	**1.2**	Immutable
Integer Numbers	**2**	Immutable
Complex Numbers	**1+2j**	Immutable
Strings	**“context”**	Immutable
Lists	**a = [1, 2.2, 'python']**	Mutable
Tuples	**t = (5,'program', 1+3j)**	Immutable
Dictionaries	**d = {1:'value','key':2}**	Immutable + Mutable

Fundamentals of Python for ML

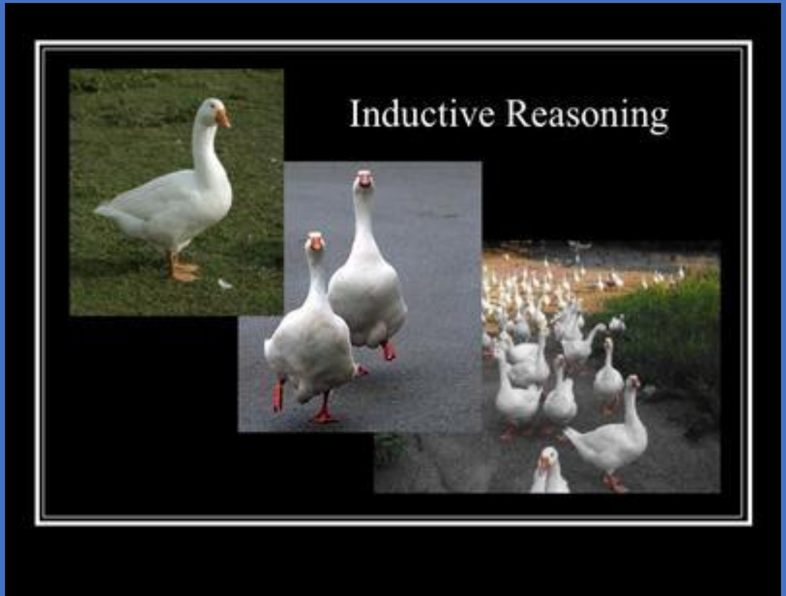
- Hands On

3.Decision Tree Learning

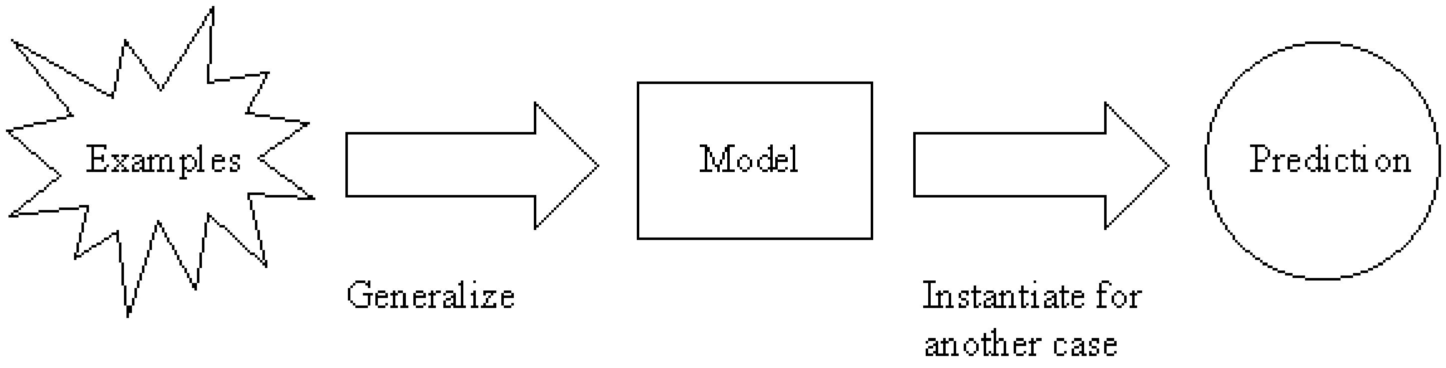
Topics

1. Decision Tree Learning
2. Decision Trees
3. Decision Tree Representation
4. Appropriate Problems for Decision Tree Learning
5. Basic Decision Tree Learning Algorithm (ID3)
6. Advantages and Disadvantages of ID3 based Decision Tree Learning

1. Decision Tree Learning



- **Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.**

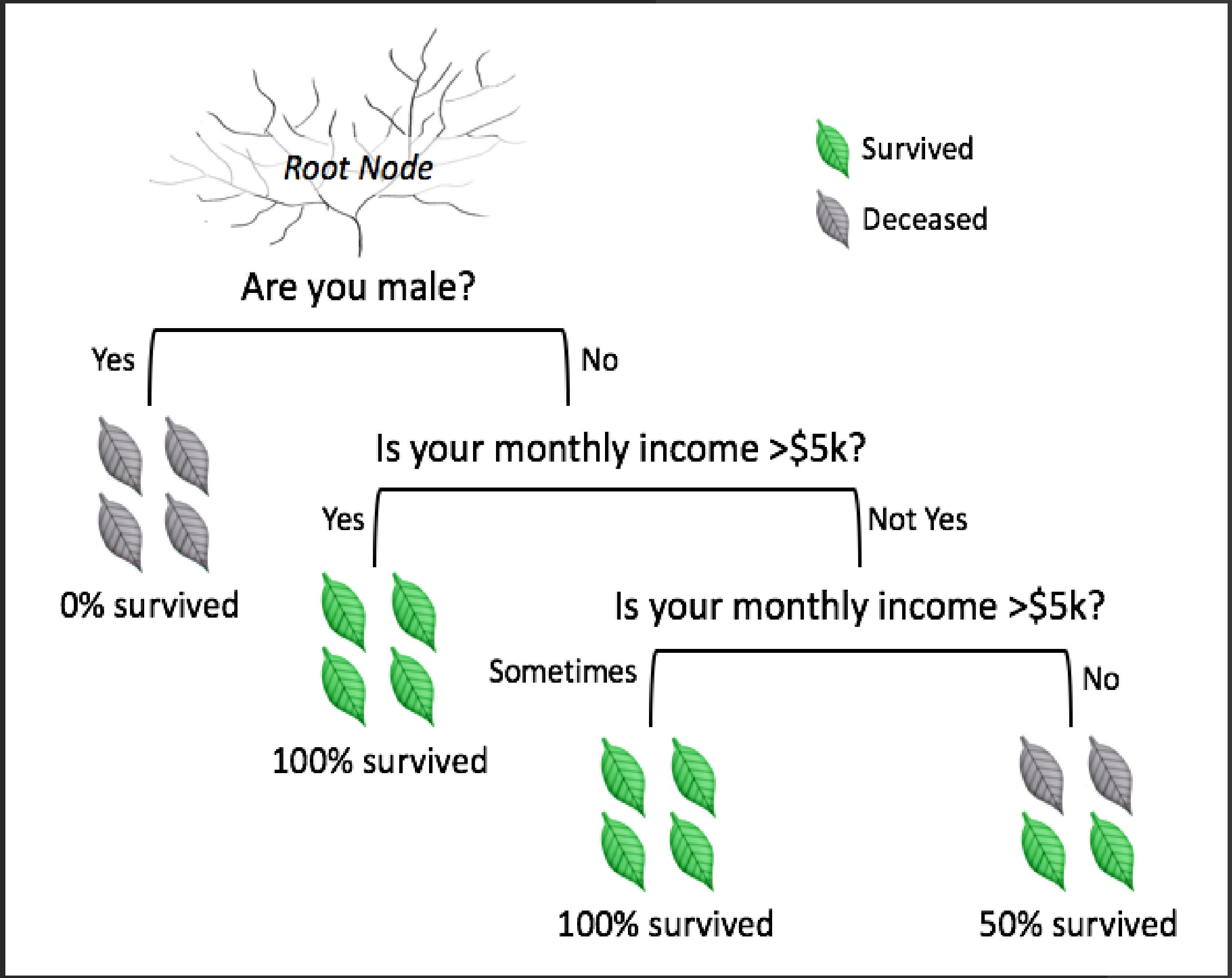


Inductive inference is the process of reaching a general conclusion from specific examples.

- **Is a type of supervised machine learning that is mostly used in classification .**

2. Decision Trees

- A decision tree is a graphical representation of all possible solutions to decisions based on certain conditions.
- A tree structured classifier with two types of nodes :
  - **Decision Nodes** and
  - **Leaf Nodes**.



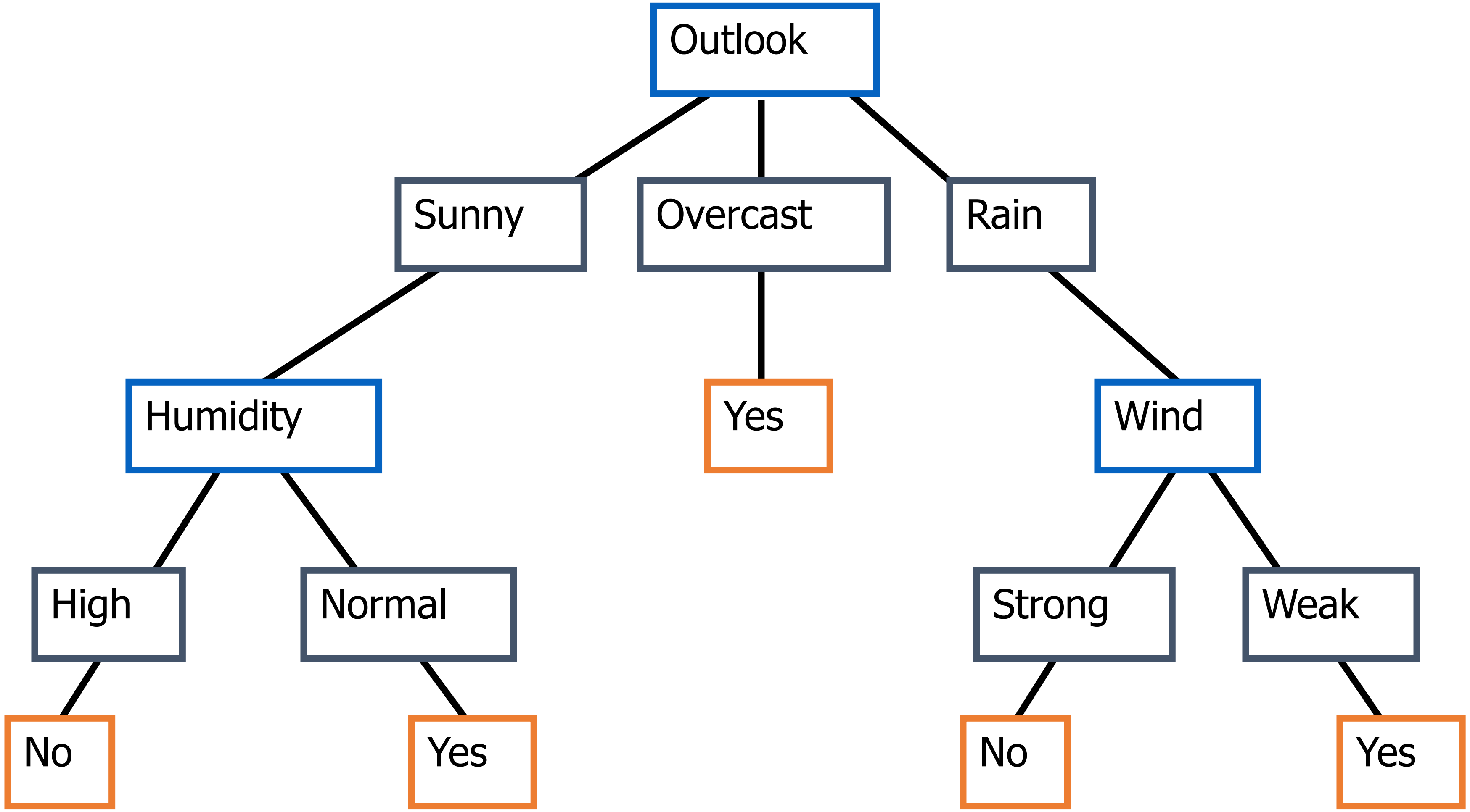
3. Decision Tree Representation

- **Decision trees** classify instances by sorting them down the tree *from the root to some leaf node*, which provides the classification of the instance.
- In the **decision tree representation**:
  - An inner node represents an attribute
  - Edge / branch represents an attribute value
  - Leaf represents a class
  - The paths from root to leaf represent **classification rules**.
- **Decision trees represent a disjunction of conjunctio**ns.
  - Each path from root to a leaf is a conjunction of attribute tests.
  - The tree itself is a disjunction of these conjunctions.

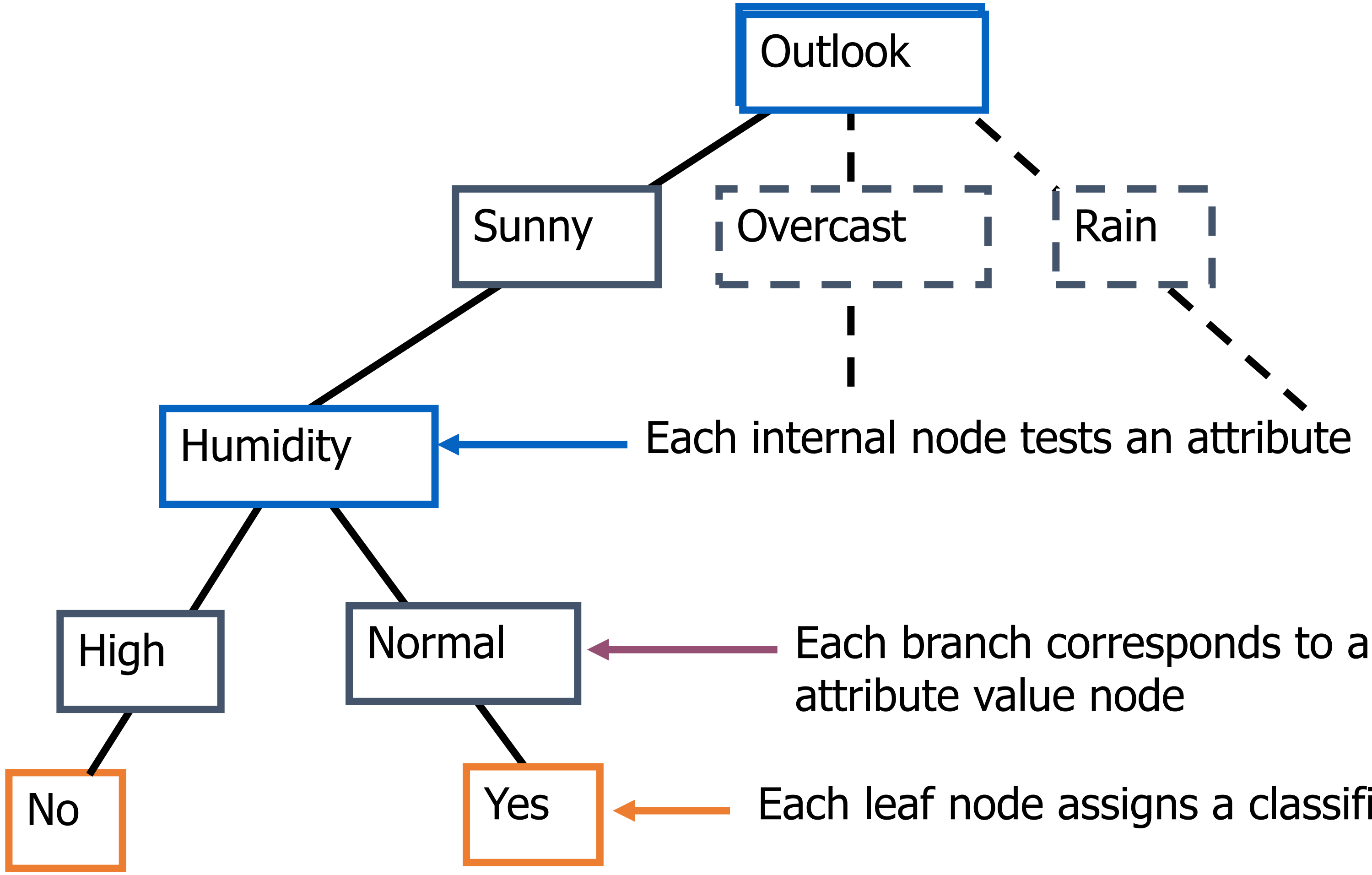
Training Examples (Data Set)

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree for PlayTennis



Decision Tree for PlayTennis



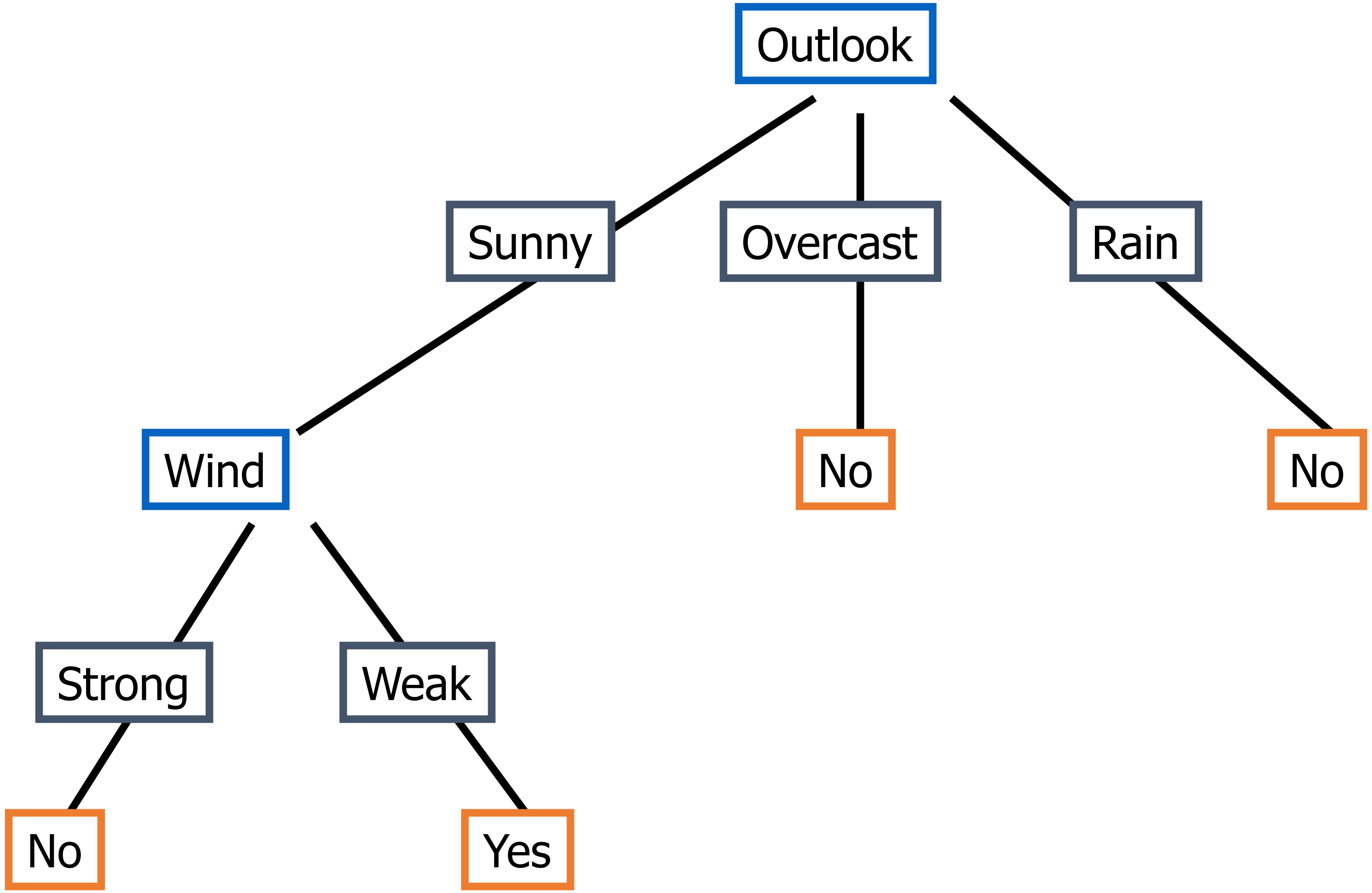
Each internal node tests an attribute

Each branch corresponds to an attribute value node

Each leaf node assigns a classification

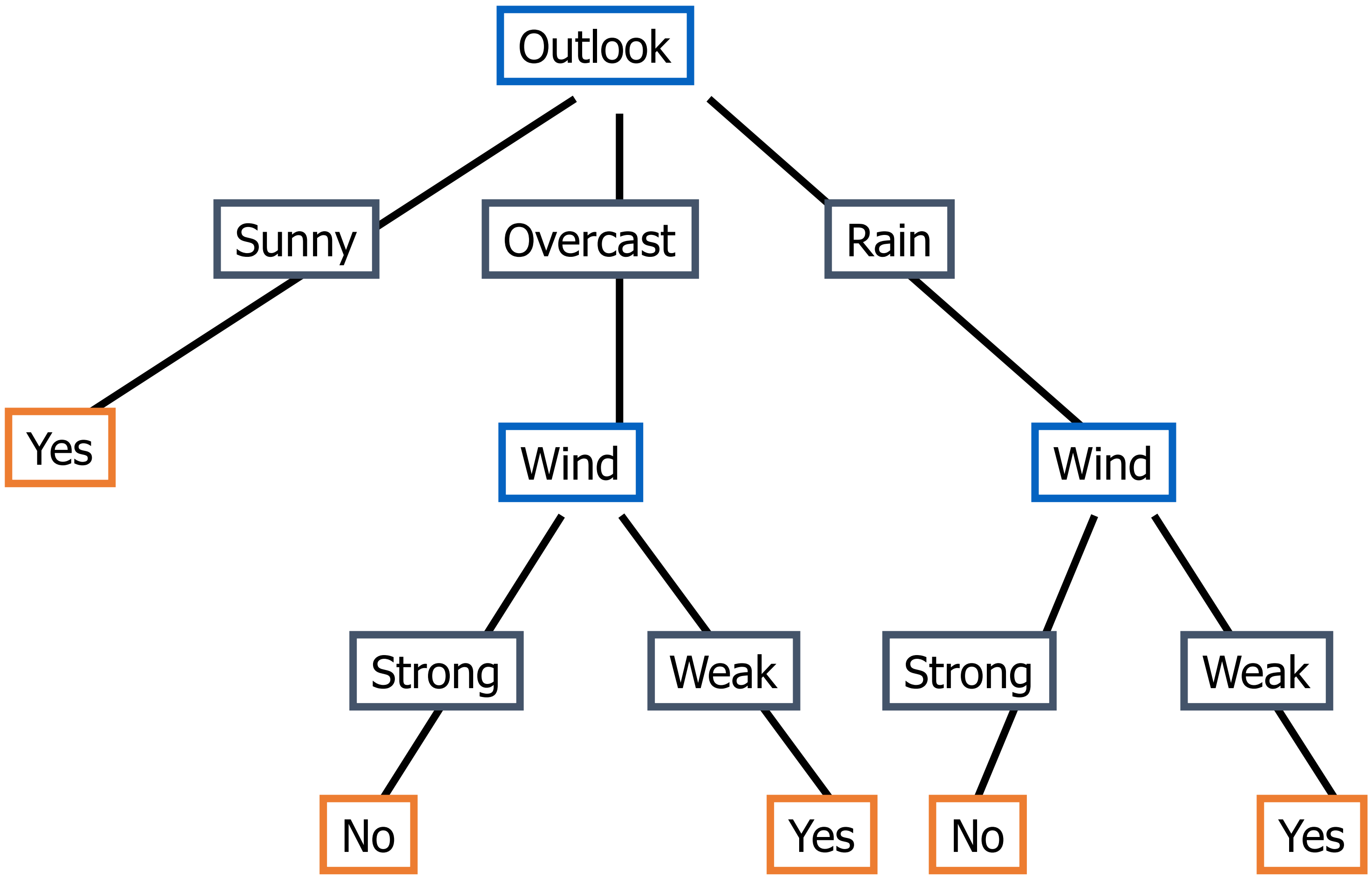
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



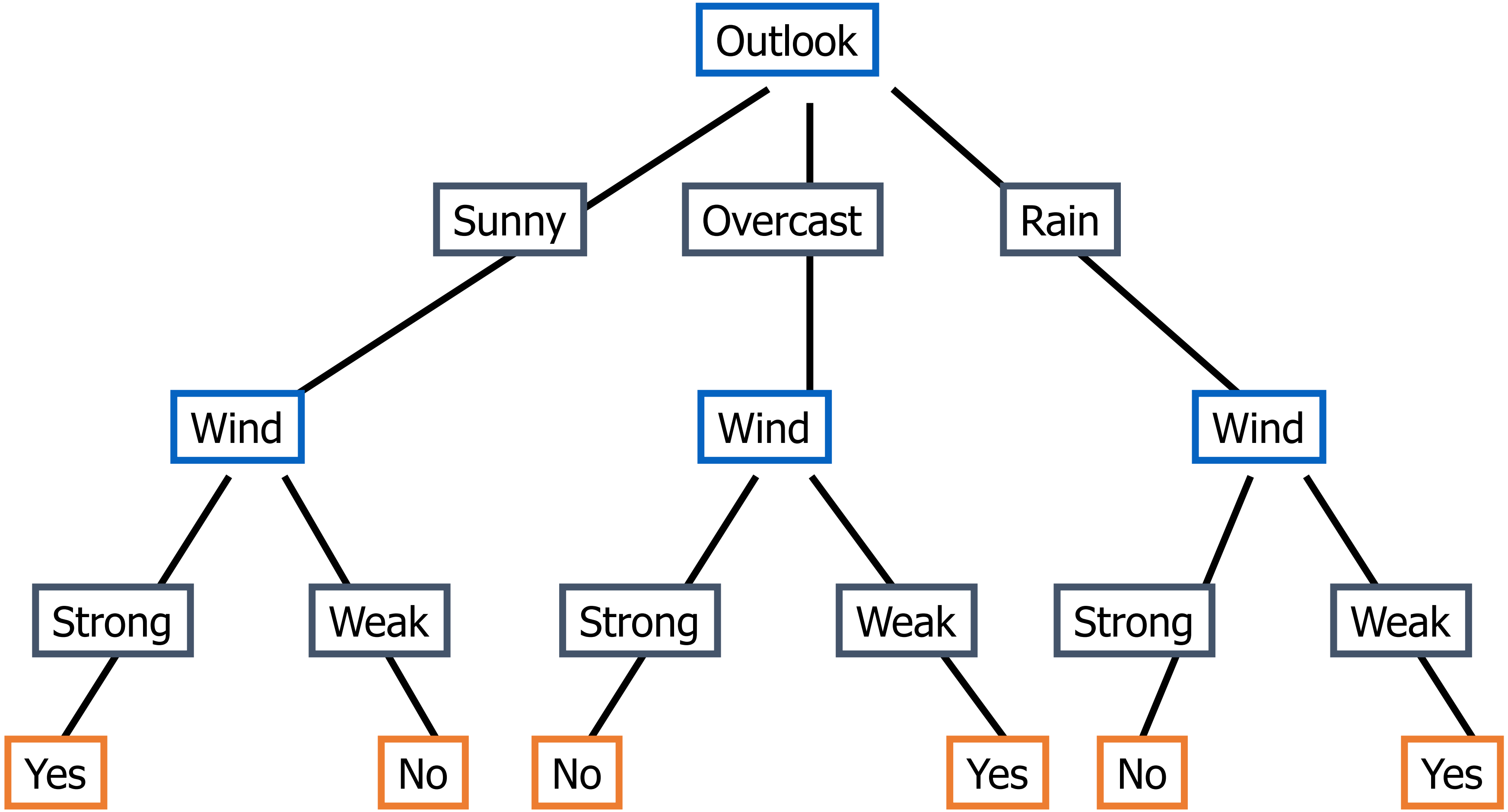
Decision Tree for Disjunction

Outlook=Sunny $\vee$ **Wind=Weak**



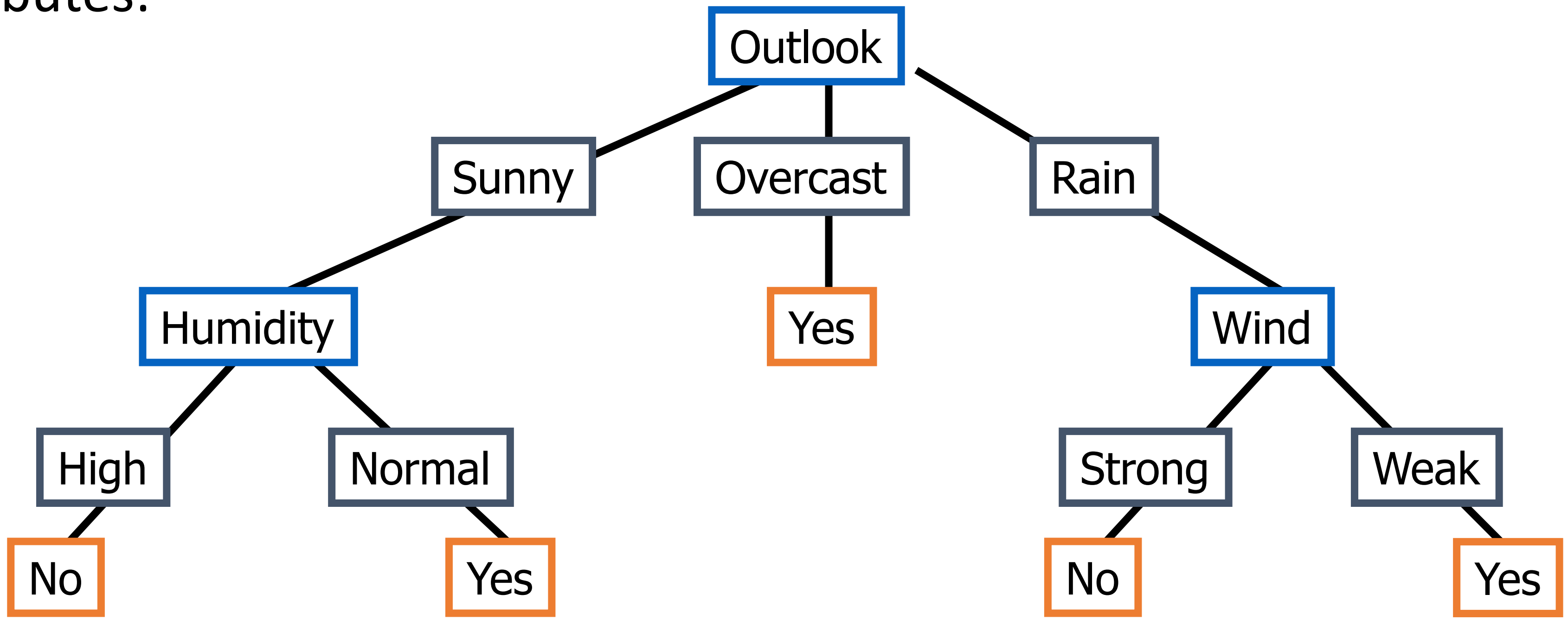
Decision Tree for XOR

Outlook = Sunny XOR Wind = Weak



Decision Tree Expressivity

- Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

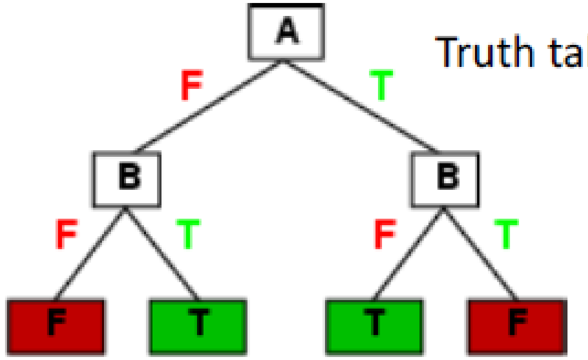


(Outlook=Sunny ∧ Humidity=Normal)
∨ (Outlook=Overcast)
∨ (Outlook=Rain ∧ Wind=Weak)

Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Truth table row → path to leaf

- In the worst case, the tree will require exponentially many nodes

4. Appropriate Problems for Decision Tree Learning

- **Decision tree learning is generally best suited to problems with the following characteristics**:
  - Instances describable by attribute-value pairs. (Hot, Mild, Cold)
  - Target function has discrete output values (yes or no).
  - Disjunctive hypothesis/description may be required.
  - The training data may contain errors/noisy data.
  - The training data may contain missing attribute values.
- **Some examples of problems that fit to these characteristics are:**
  - Medical or equipment diagnosis
  - Credit risk analysis
  - Modelling calendar scheduling preferences

5.Basic Decision Tree Learning Algorithm (ID3)

- The basic decision tree learning algorithm, ID3, employs a top-down, greedy search through the space of possible decision trees, beginning with the question ”**which attribute should be tested at the root of the tree?**”.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Description of PlayTennis Training Data Examples

- **No of Data Examples : 14**
  - No of Yes (+) : 9
  - No of No (-) : 5
- **Data Example Attributes and their possible Values :**
  - [Outlook] Values : { Sunny ,Overcast ,Rain }
  - [Temperature] Values : { Hot , Mild , Cold }
  - [Humidity] Values : { High, Normal }
  - [Wind] Values : { Weak, Strong }
- **Target Attribute :**
  - [PlayTennis] Values : {Yes, No}

Choosing the Best Attribute

Key problem: choosing which attribute to split a given set of examples

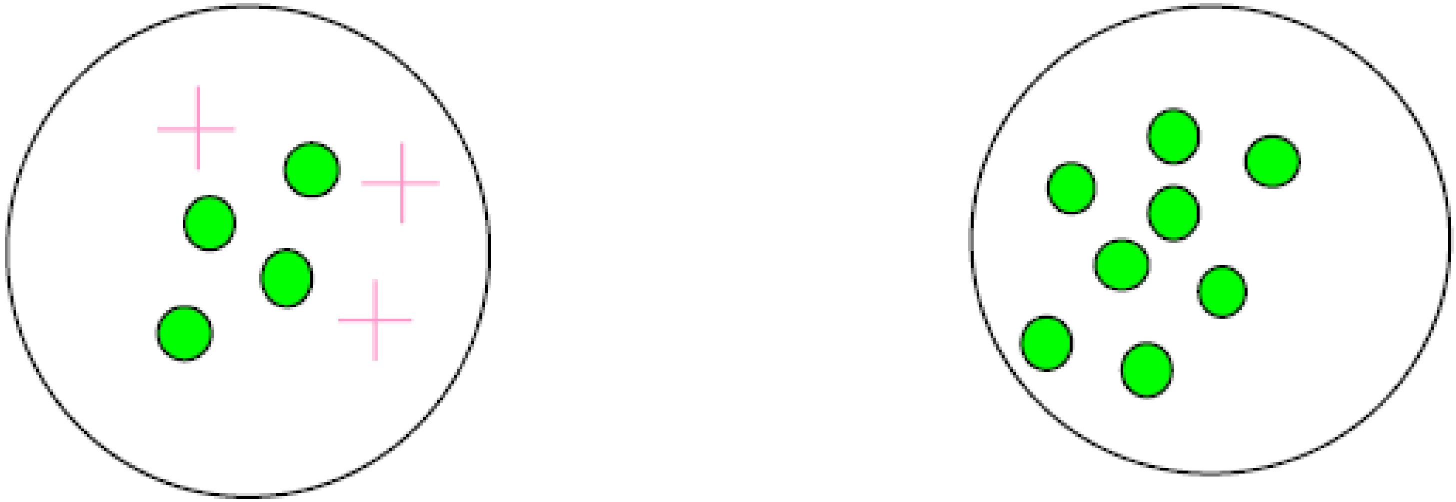
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Which attribute is the Best Classifier ?

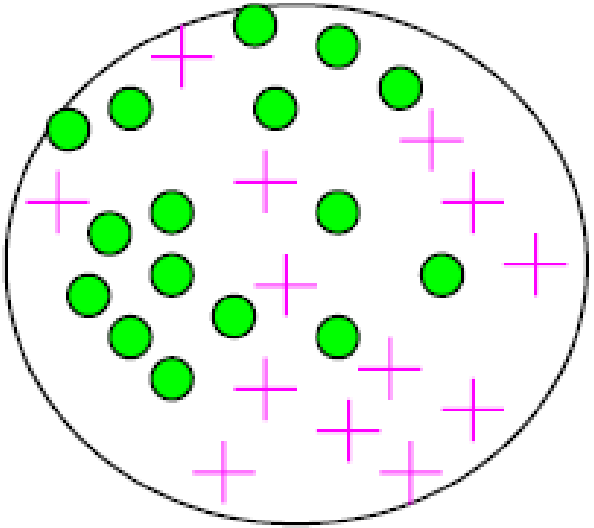
- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.
  - The attribute must be selected that is most useful for classifying examples .
- The statistical property called *information gain* is the good quantitative measure of the worth of an attribute .
- Information Gain measures how well a given attribute separates the training examples according to their target classification.
- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.
- Information gain uses the notion of *entropy*, commonly used in information theory
- Information gain = expected reduction of entropy

Impurity/Entropy (informal)

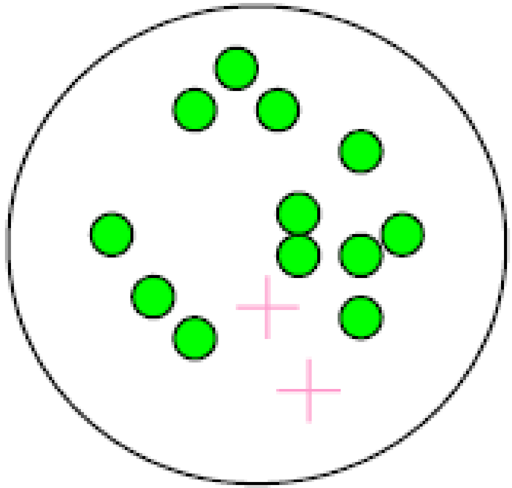
– Measures the level of **impurity** in a group of examples



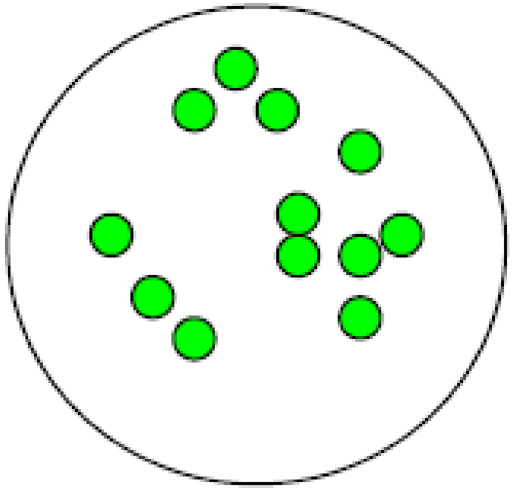
Very impure group



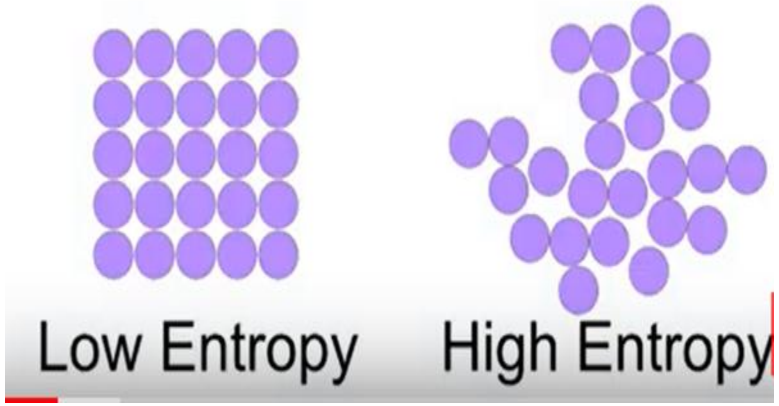
Less impure



Minimum impurity

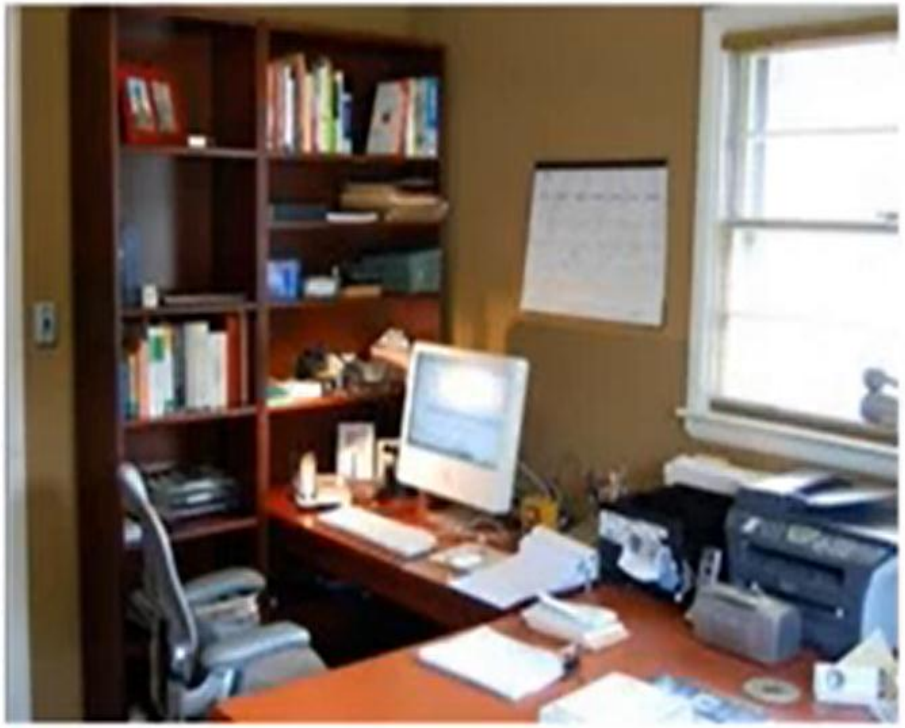


What is Entropy ?



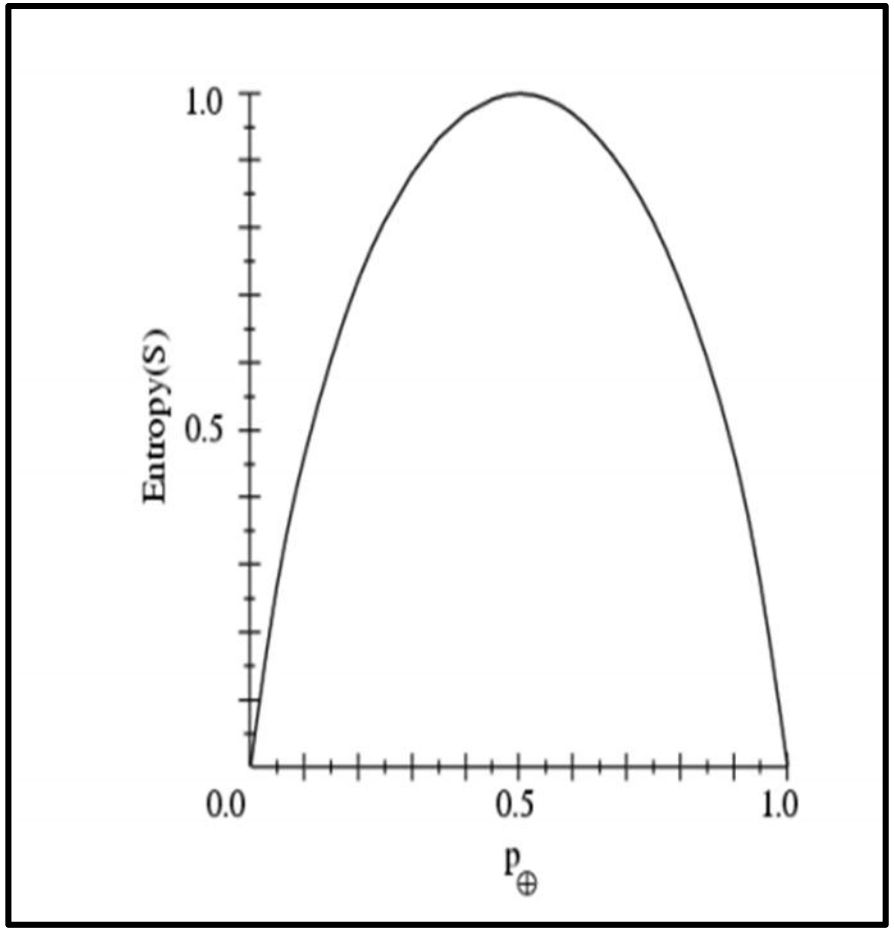
- Define and measures Randomness/impurity in the Data
- Minimum number of bits of information needed to encode the classification of an arbitrary member of Examples Space S.
- If the target attribute can take on **c** possible values the entropy can be as large as $\boldsymbol{log_2\ c}$

High Entropy (messy)



Low Entropy (Clean)

1. Entropy measures Homogeneity of Examples



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

$$\text{Entropy(S)} \equiv -p_{\oplus} log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus}$$

S is a sample of training examples

$p_{\oplus}$ is the proportion of positive examples in S

$p_{\ominus}$ is the proportion of negative examples in S

Examples

- E(S) = - $\boldsymbol{p_+ log_2 p_+}$ - $\boldsymbol{p_- \, log_2 \, p_-}$
- E(9+,5-) = -(9/14)$\boldsymbol{log_2}$(9/14) - (5/14) $\boldsymbol{log_2}$(5/14)

  = 0.940

$$Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$Entropy\ ([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy\ ([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$Entropy\ ([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14)$$

$$= 1/2 + 1/2 = 1$$

$[\log_2 1/2 = -1]$

2. Information Gain Measures the expected reduction in Entropy

- **Information Gain** is the *expected* reduction in entropy caused by partitioning the examples on an attribute. The higher the information gain the more effective the attribute in classifying training data.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Values(A)$ is the set of all possible values for attribute A
S_v is the subset os S for which attribute A has value v

An Illustrative Example

To illustrate the operation of ID3, let's consider the learning task represented by the below examples.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	**No**
D2	Sunny	Hot	High	Strong	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Weak	**Yes**
D5	Rain	Cool	Normal	Weak	**Yes**
D6	Rain	Cool	Normal	Strong	**No**
D7	Overcast	Cool	Normal	Strong	**Yes**
D8	Sunny	Mild	High	Weak	**No**
D9	Sunny	Cool	Normal	Weak	**Yes**
D10	Rain	Mild	Normal	Weak	**Yes**
D11	Sunny	Mild	Normal	Strong	**Yes**
D12	Overcast	Mild	High	Strong	**Yes**
D13	Overcast	Hot	Normal	Weak	**Yes**
D14	Rain	Mild	High	Strong	**No**

Compute the Gain and identify which attribute is the best as illustrated below

- Values (Wind) = Weak, Strong
- S = [9+,5-]
- $S_{weak} = [6+, 2-]$
- $S_{strong} = [3+, 3-]$

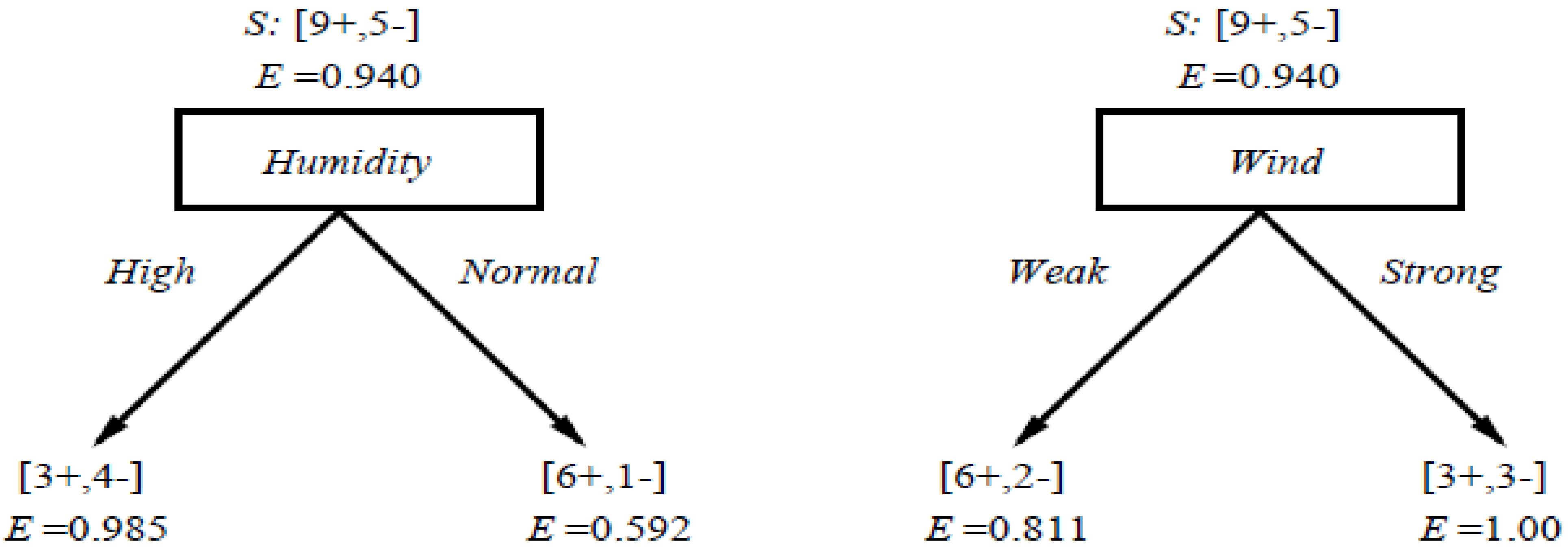
$$\text{Gain}(S, \text{wind}) = \text{Entropy}(S) - \sum_{v \in \{weak, strong\}} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

= Entropy (S) – (8/14)Entropy(S_{weak}) – (6/14)Entropy(S_{strong})

= 0.940 – (8/14)0.811 – (6/14)1.00

= 0.048

Which attribute is the best classifier?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

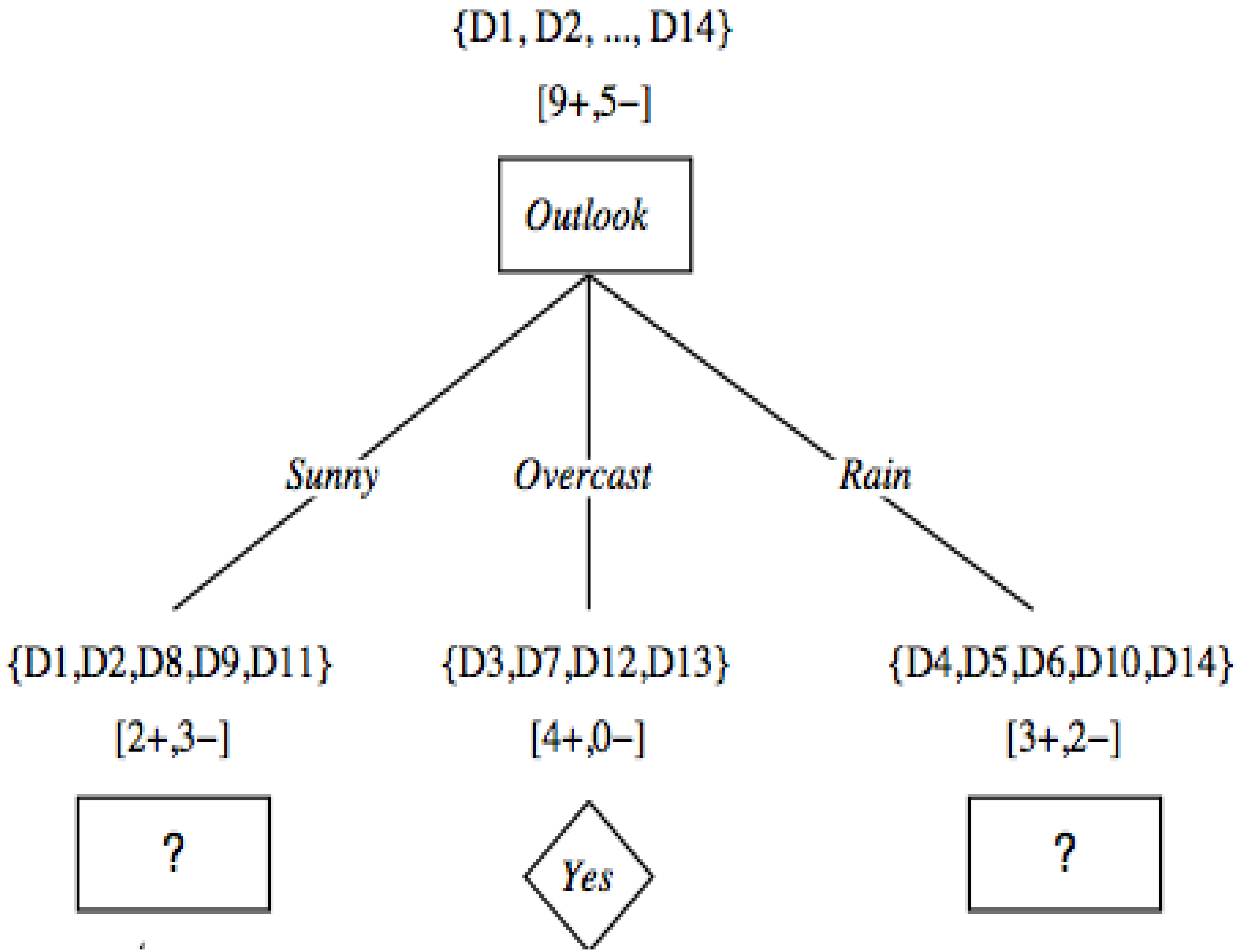
Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Which attribute to test at the root?

  - ***Gain*(*S*, *Outlook*) = 0.246**
  - ***Gain*(*S*, *Humidity*) = 0.151**
  - ***Gain*(*S*, *Wind*) = 0.048**
  - ***Gain*(*S*, *Temperature*) = 0.029**
- *Outlook* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Outlook*
  - partition the training samples according to the value of *Outlook*

After first step



Second step

- *Ssunny* = {D1,D2,D8,D9,D11}
- Working on *Outlook=Sunny* node:

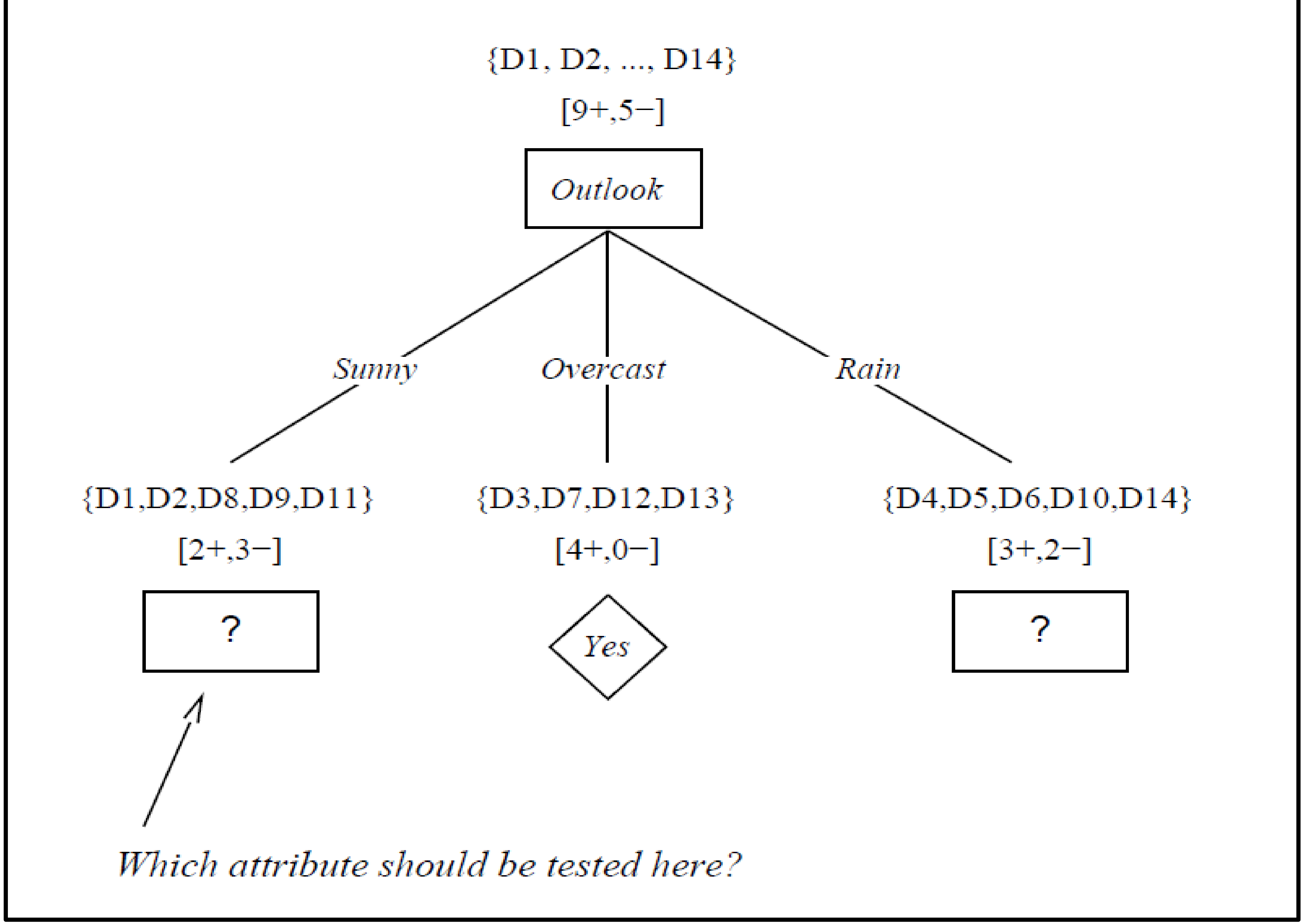
  $Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = \mathbf{0.970}$

  $Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = \mathbf{0\ .019}$

  $Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = \mathbf{0.570}$

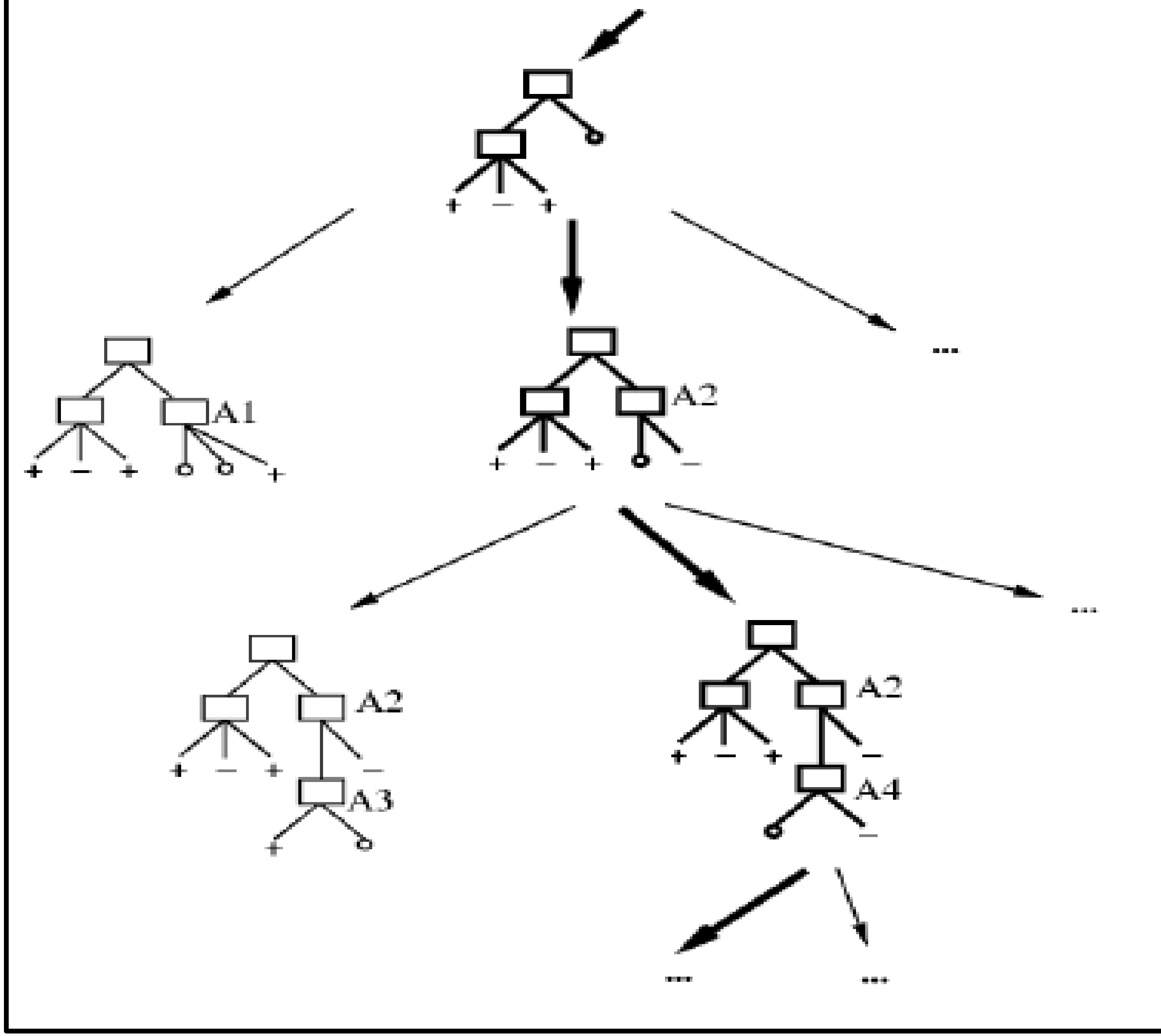
- *Humidity* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Humidity*
  - partition the training samples according to the value of *Humidity*

Second and third steps



Which Tree Should We Output?

- **ID3 performs heuristic search through space of decision trees**
- **It stops at smallest acceptable tree. Why?**



Algorithm : ID3(*Training Example, Target Attributes, Attributes*)

- Create *Root* node
- *If* all X's are +, *return* *Root* with class +
- *If* all X's are –, *return* *Root* with class –
- *If* *Attrs* is empty *return* *Root* with class most common value of *T* in *X*
- *else*

  $A \leftarrow$ best attribute; decision attribute for Root $\leftarrow A$

  For each possible value v_i of A:

  - add a new branch below *Root*, for test $A = v_i$

  - $X_i \leftarrow$ subset of X with $A = v_i$

  - *If* X_i is empty *then* add a new leaf with class the most common value of *T* in *X*

  ***else* add the subtree generated by ID3(X_i, T, $Attrs - \{A\}$)**

- ***return Root***

ID3 - Algorithm

ID3(*Examples, TargetAttribute, Attributes*)

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = -
- If *Attributes* is empty, Return the single-node tree Root, with label = most common value of *TargetAttribute* in *Examples*
- Otherwise Begin
  - A ← the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* ← A
  - For each possible value, vi, of A,
    - Add a new tree branch below *Root*, corresponding to the test A = vi
    - Let $Examples_{vi}$ be the subset of *Examples* that have value vi for A
    - If $Examples_{vi}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *TargetAttribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{vi}$, *TargetAttribute*, *Attributes* – {A})
- End
- Return *Root*

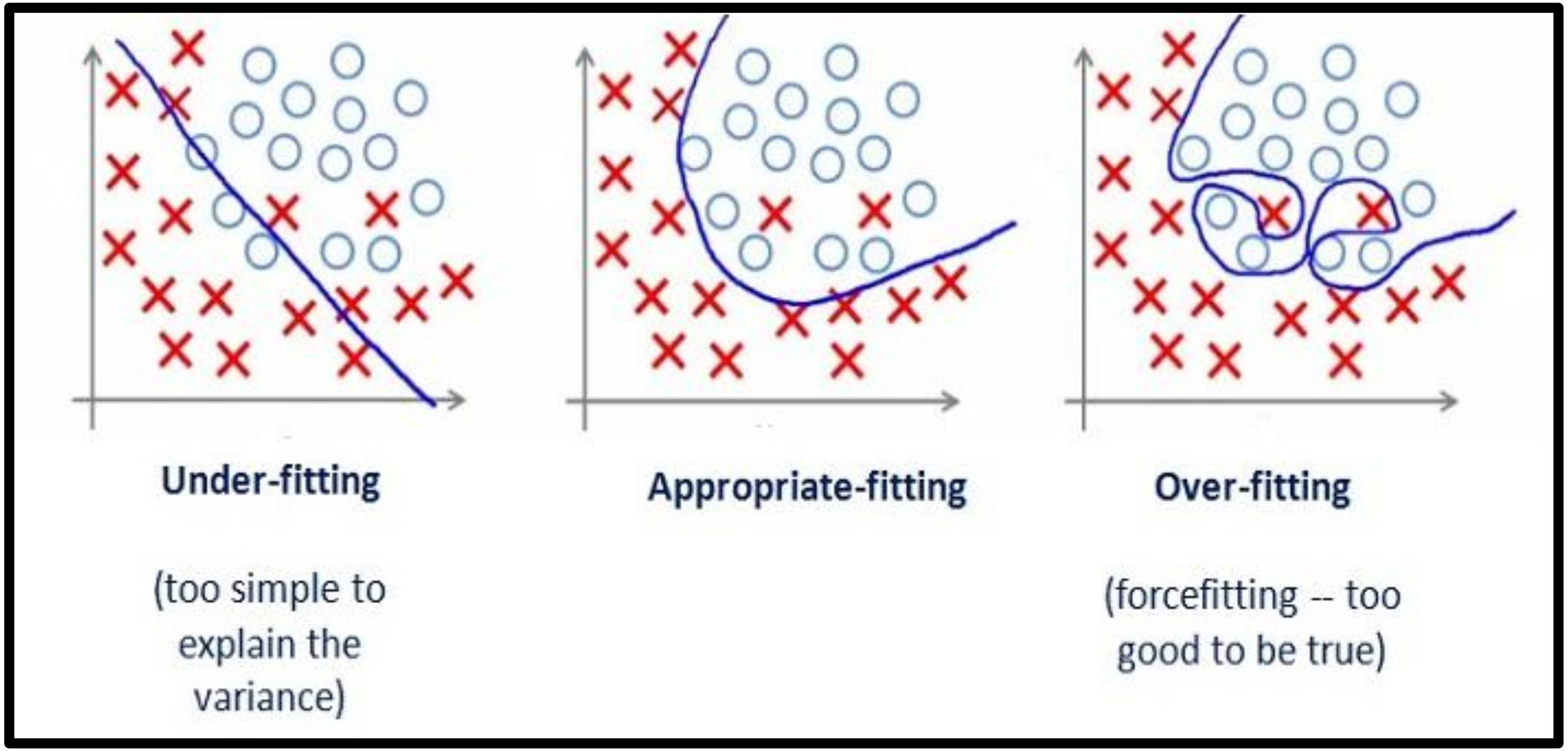
6. Advantages and Disadvantages

Advantages :

- **Computationally Inexpensive**
- **Handles both numerical and categorical attributes**
- **Outputs are easy to interpret**
- **Works well with both linear and nonlinear data**
- **Sensitive to small variations in the training data**
- **Robust with redundant and correlated data**

Disadvantages :

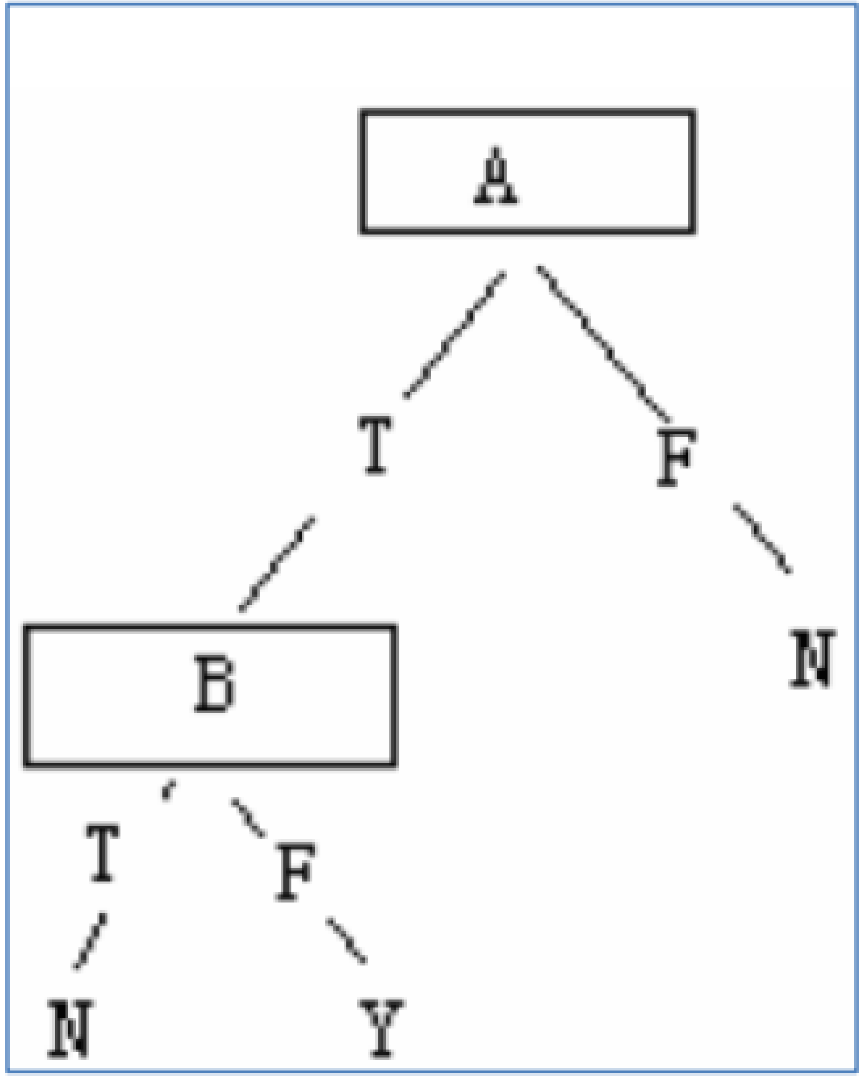
- **Overfitting**
- **Too many Layers**
- **Lack of training Data**
- **Biased Data in training set**
- **Multicollinearity among variables**


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

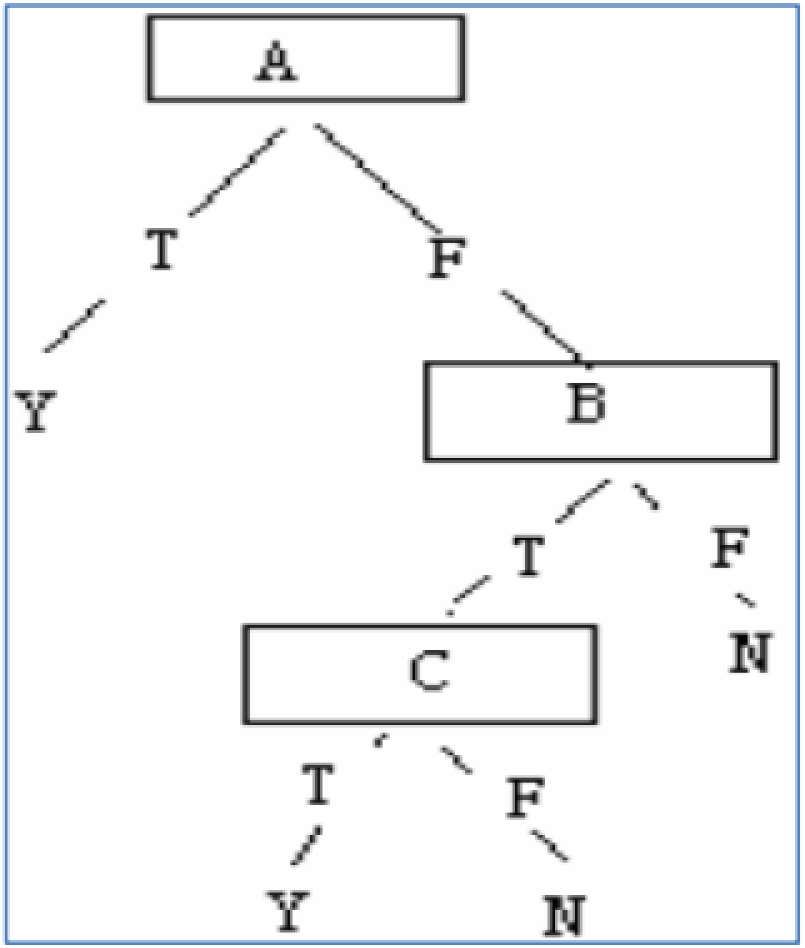
Example : Give decision trees to represent the following Boolean functions:

1) $A \wedge \neg B$
2) $A \vee [B \wedge C]$
3) A XOR B
4) $[A \wedge B] \vee [C \wedge D]$

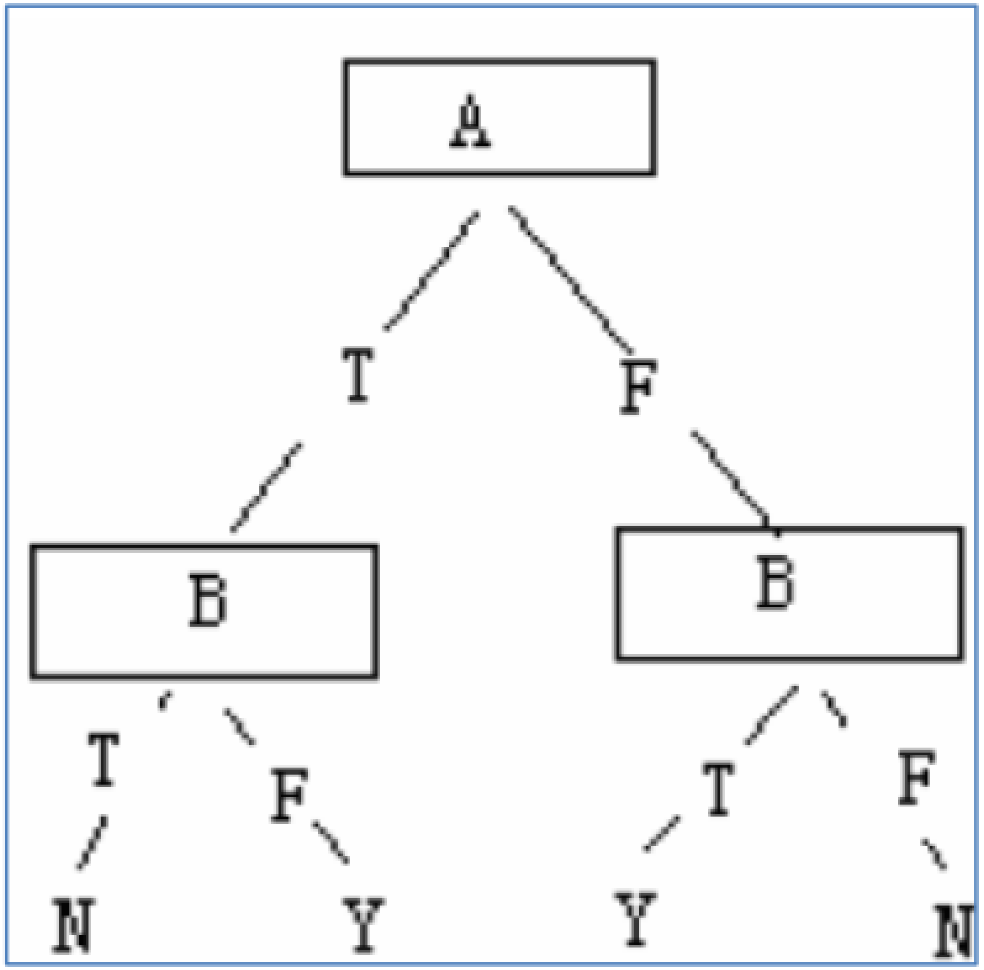
1) $A \wedge \neg B$



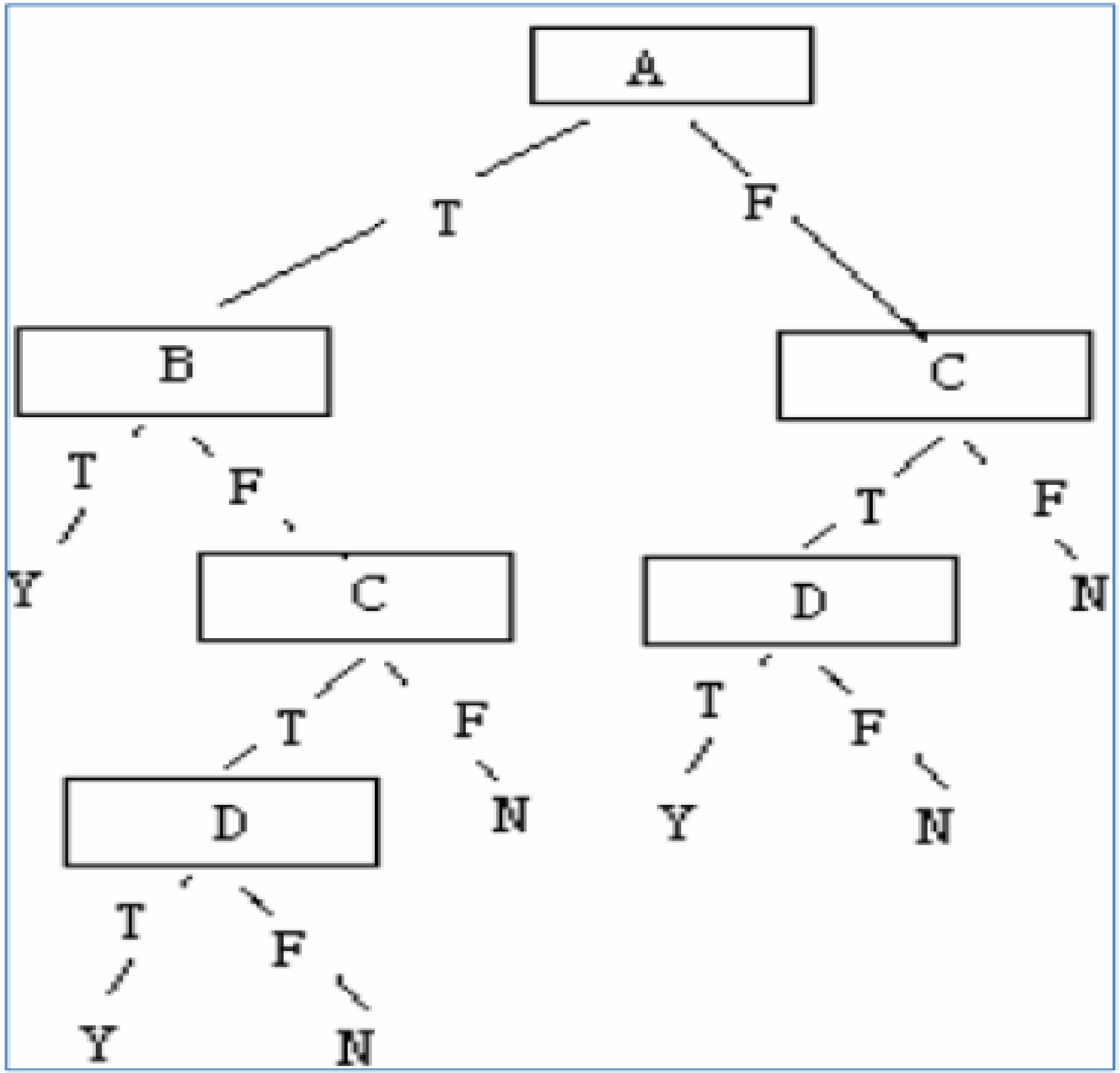
2) $A \vee [B \wedge C]$



3) A XOR B = $(A \wedge \neg B) \vee (\neg A \wedge B)$



4) $[A \wedge B] \vee [C \wedge D]$



2. Consider the following set of training examples:

Example	A1	A2	A3	class
1	T	T	T	+
2	T	T	F	+
3	T	F	T	-
4	T	F	F	+
5	F	T	T	-
6	F	T	F	-
7	F	F	T	+
8	F	F	F	-

A) What is the entropy of this collection of training examples with respect to the target function classification?

- The entropy of this collection of training examples with respect to the target function classification is E(S) = 1
- because it contains equal numbers of positive and negative examples.

B) What is the information gain of feature A2 relative to these training examples?

- The information gain of feature A2 relative to these training examples $G(S,A2) = E(S) - \sum|SA2| / |S| * E(S) = 1 – (4/8 * 1 + 4/8 * 1) = 0$

C) What is the best feature relative to these training examples, using Gain Ratio?

- The best feature relative to these training examples is the feature with the maximum information gain, and in this example any feature can selected because the gain of all features are the same.

4.Machine Learning Lab Programs using Python

https://github.com/profthyagu

Lab Programs

1. Decision tree based ID3 algorithm
2. Backpropagation algorithm to build ANN
3. Naïve Bayesian classifier for a sample training data set stored as a .CSV file
4. Naïve Bayesian classifier to cluster the set of documents
5. Bayesian Network for Medical Data Set
6. EM Algorithm + K Means Algorithm for a set of data stored in.csv file
7. K NN Algorithm for iris data
8. Locally Weighted Regression Algorithm
9. FIND-S algorithm *for a given set of training data examples stored in a .CSV file.*
10. Candidate-Elimination algorithm *for a given set of training data examples stored in a .CSV file*

Lab Program 1:

- Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

ID3 - Algorithm

ID3*(Examples, TargetAttribute, Attributes*)

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = -
- If *Attributes* is empty, Return the single-node tree Root, with label = most common value of *TargetAttribute* in *Examples*
- Otherwise Begin
  - A ← the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* ← A
  - For each possible value, vi, of A,
    - Add a new tree branch below *Root*, corresponding to the test A = vi
    - Let $Examples_{vi}$ be the subset of *Examples* that have value vi for A
    - If $Examples_{vi}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *TargetAttribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{vi}$, *TargetAttribute*, *Attributes* – {A})
- End
- Return *Root*

Source Code

- https://github.com/profthyagu

Lab Program 2

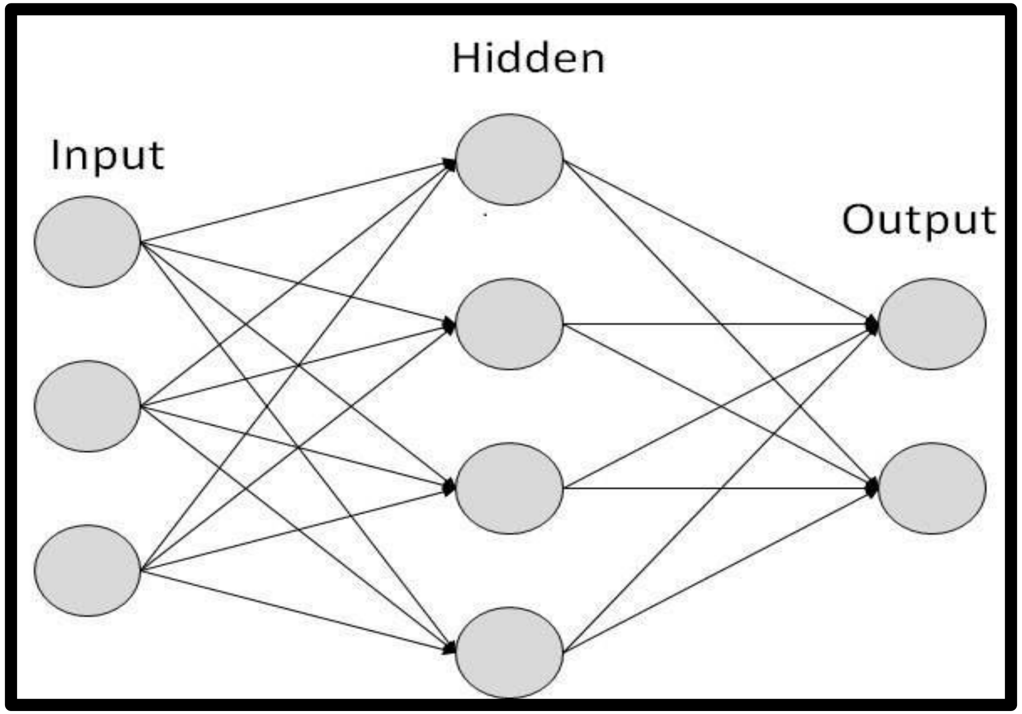
- **Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.**

Artificial neural networks(ANN/NN)

- An ANN is a computational model based on the structure and functions of biological neural networks.
- An ANN can learn from observing data sets.
- ANN takes data samples rather than entire data sets to arrive at solutions, which saves both time and money.

Artificial neural networks

- ANNs have three layers that are interconnected.
- The first layer consists of input neurons. These neurons send data on to the second layer (Hidden) , which in turn sends the data to the third layer (Output).

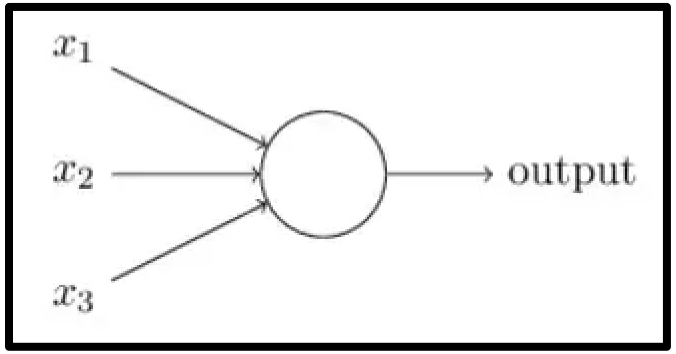


Perceptrons

- **Perceptrons** are single-layer feedforward networks
- Each output unit is independent of the others
- Can assume a single output unit
- Activation of the output unit is calculated by:
- $$O = \mathrm{Step}_0\left(\sum_{j} W_j\, x_j\right)$$

where x_j is the activation of input unit j, and we assume an additional weight and input to represent the threshold

Perceptrons



- A Perceptron is a type of artificial neuron which takes in several binary inputs x1, x2, … , xn and produces a single binary output.
- In order to compute an output, here's what you could do (O = Output):
  a) Sum up the inputs. O = x1 + x2 + x3
  b) Apply a linear mathematical operation to the inputs. O = (x1 * x2) + x3

Note : Thr approach (b) can have its output as O = x3, if x1 or x2 or both are equal to zero. If only one of them is equal to 0, this can lead to a loss of information as the input value gets multiplied by 0.

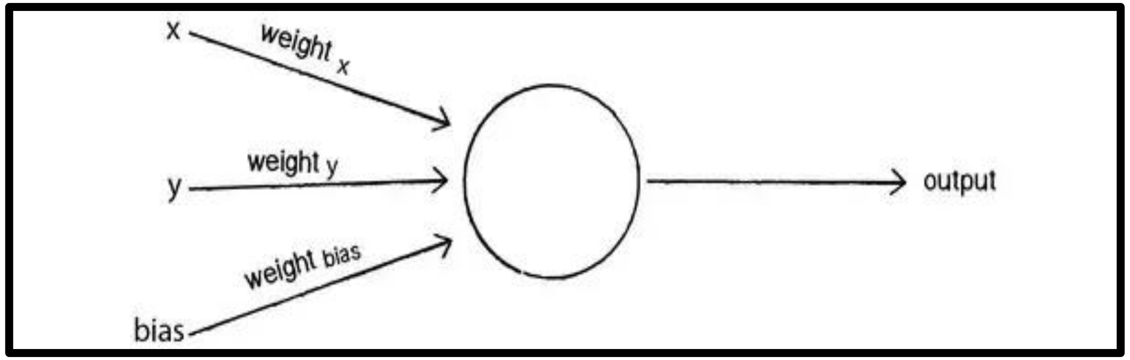
- The simple approach (a), is partially correct and what we can do to make it fully correct.
- Approach (a) sure takes into account all the inputs without one being immediately affected by the other. However, it does not take in the relative importance of the inputs.

What does weight mean in terms of neural networks?

- Consider a simple example where you want to make a perceptron for predicting whether it will rain today or not. You have a binary output as O, which can take values of 0 or 1 and inputs x1 and x2.
- Let x1 be 1 if the weather is humid today, 0 if the opposite.
- Let x2 be 1 if you are wearing a red shirt today and 0 if not.
- We can see here that wearing a red shirt has almost no correlation with the possibility of rainfall. So, a possible output function can be: O = x1 + **0.1** * x2
  - Here's what the factor **0.1** does. If x2 is 0, 0.1*x2 is still 0, but if x2 is 1, 0.1*x2 will be 0.1, not 1.
  - It basically brings down the importance of the input x2 from 1 to 0.1, and hence, it is called the **'weight'** of the input x2.
  - Consider x2 to be 1 for now. O will still be (0 + 0.1) = 0.1 or (1 + 0.1) = 1.1, i.e. not a binary value.
- In order to make it one, what we can do is use a '**threshold'** for the total value of O.
- We can say, like, O is 1 if (x1 + 0.1 * x2) > 1 and O = 0 if (x1 + 0.1 * x2) < 1. We have thus solved our problem.

What does weight mean in terms of neural networks?

In general, $$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

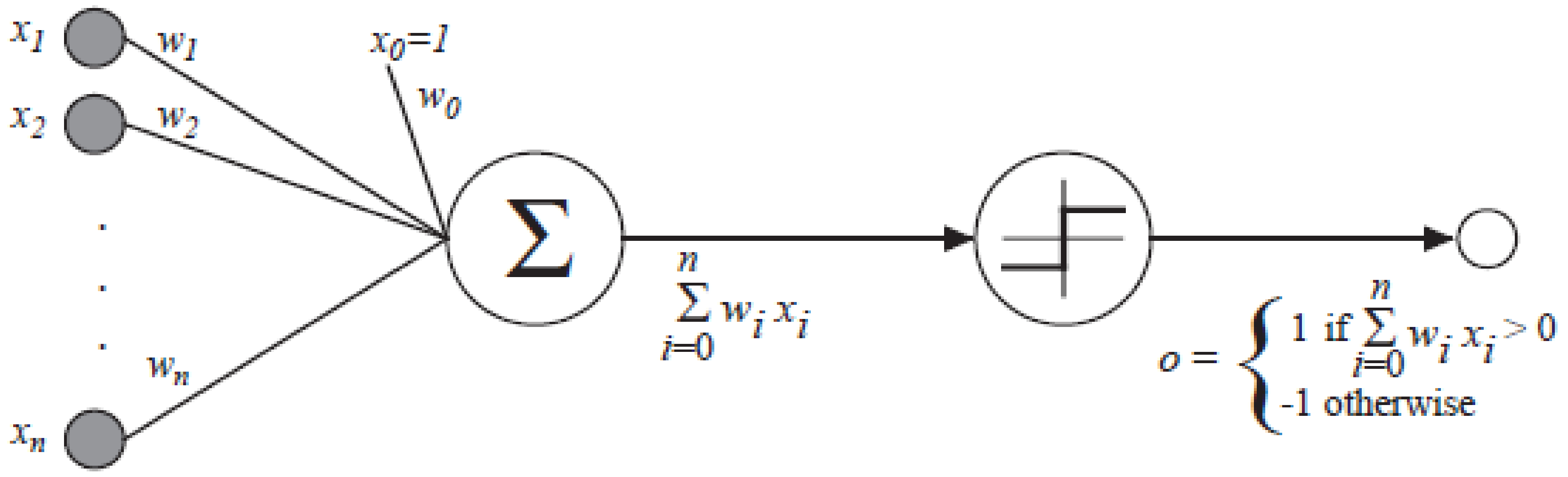


And now making some final notational simplifications, we say

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Where w = [w1 w2 w3 …. wn]T, x = [x1 x2 x3 …. xn] and b = -(threshold)
•**b** is commonly known as bias.

Note: ‘T’ is the transpose operation. And w.x is matrix multiplication.

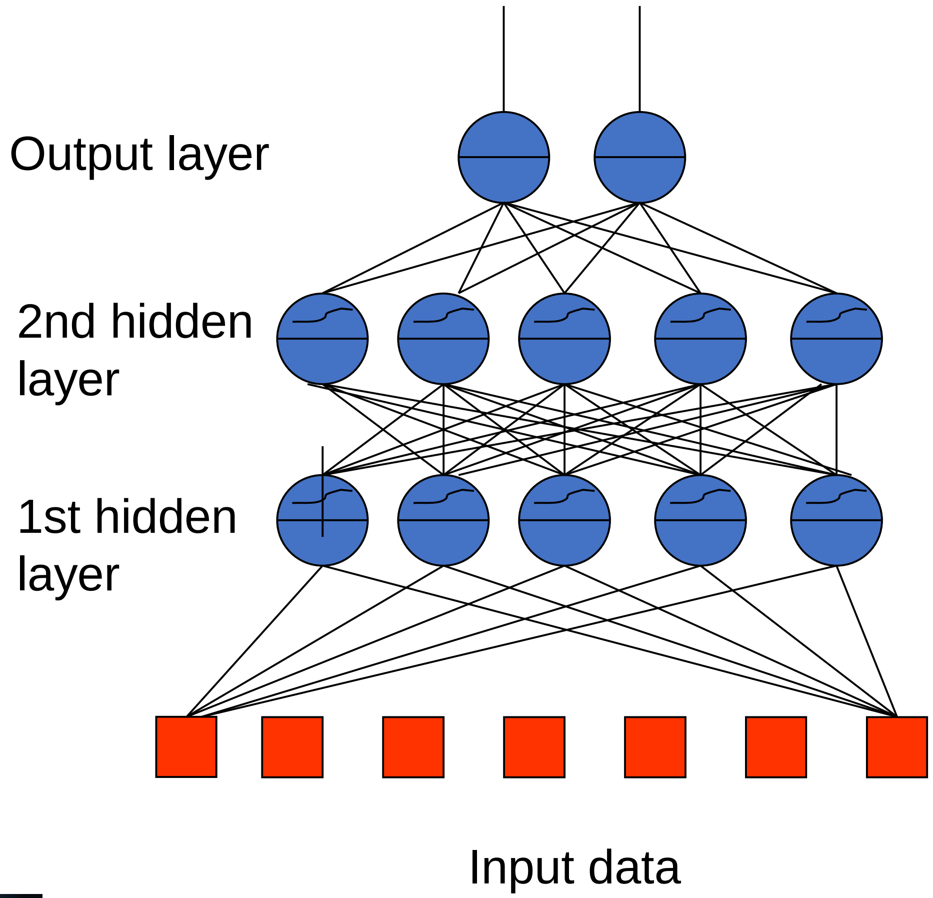


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Multi-Layer Perceptron



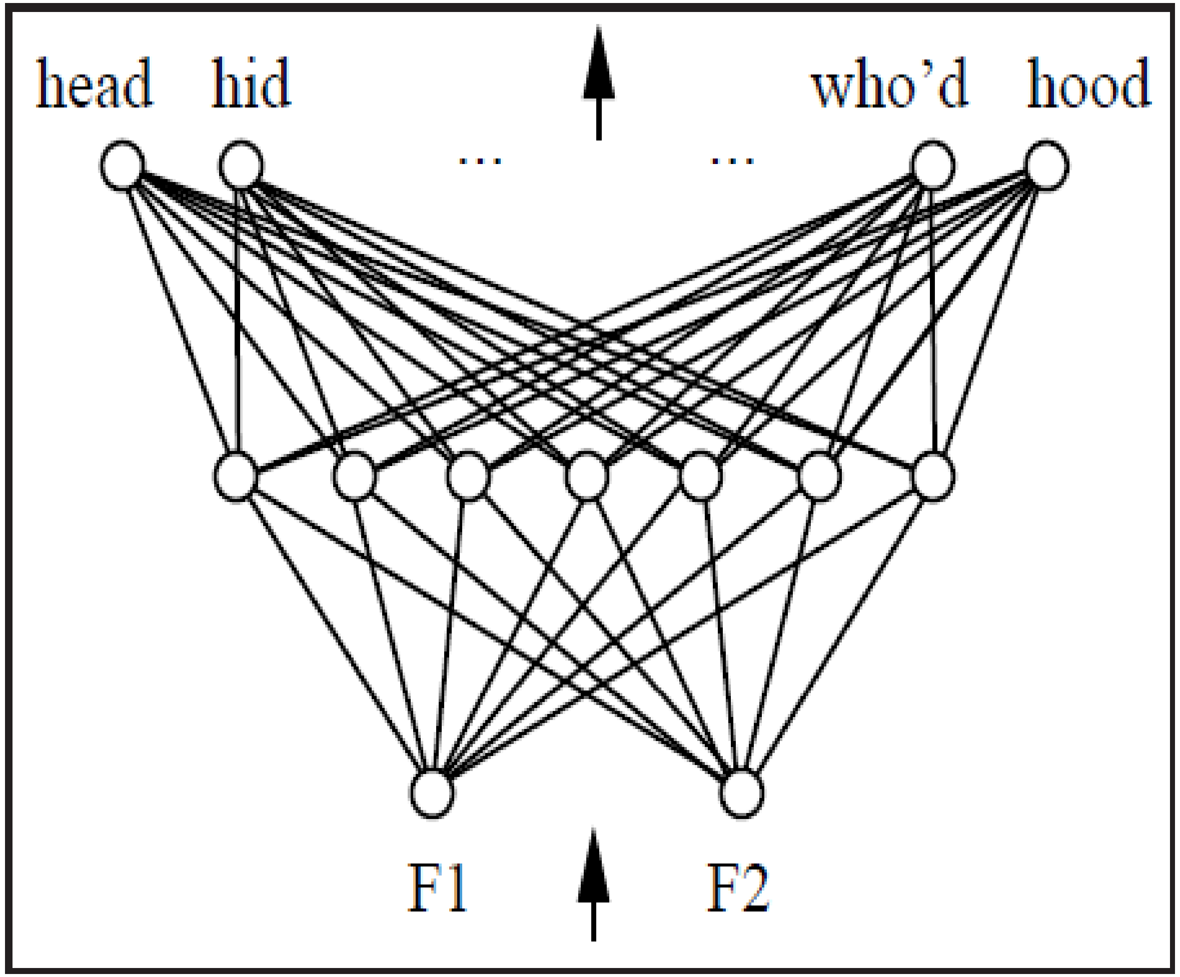
- One or more hidden layers
- Sigmoid activations functions

MULTILAYER NETWORKS AND THE BACKPROPAGATION ALGORITHM

- Single perceptron's can only express linear decision surfaces.
- In contrast, the kind of multilayer networks learned by the **BACKPROPACATION** algorithm are capable of expressing a rich variety of nonlinear decision surfaces.
- For example, a typical multilayer network and decision surface is depicted in Figure
- Here the speech recognition task involves distinguishing among 10 possible vowels, all spoken in the context of "h-d" (i.e., "hid," "had,“ "head," "hood," etc.).

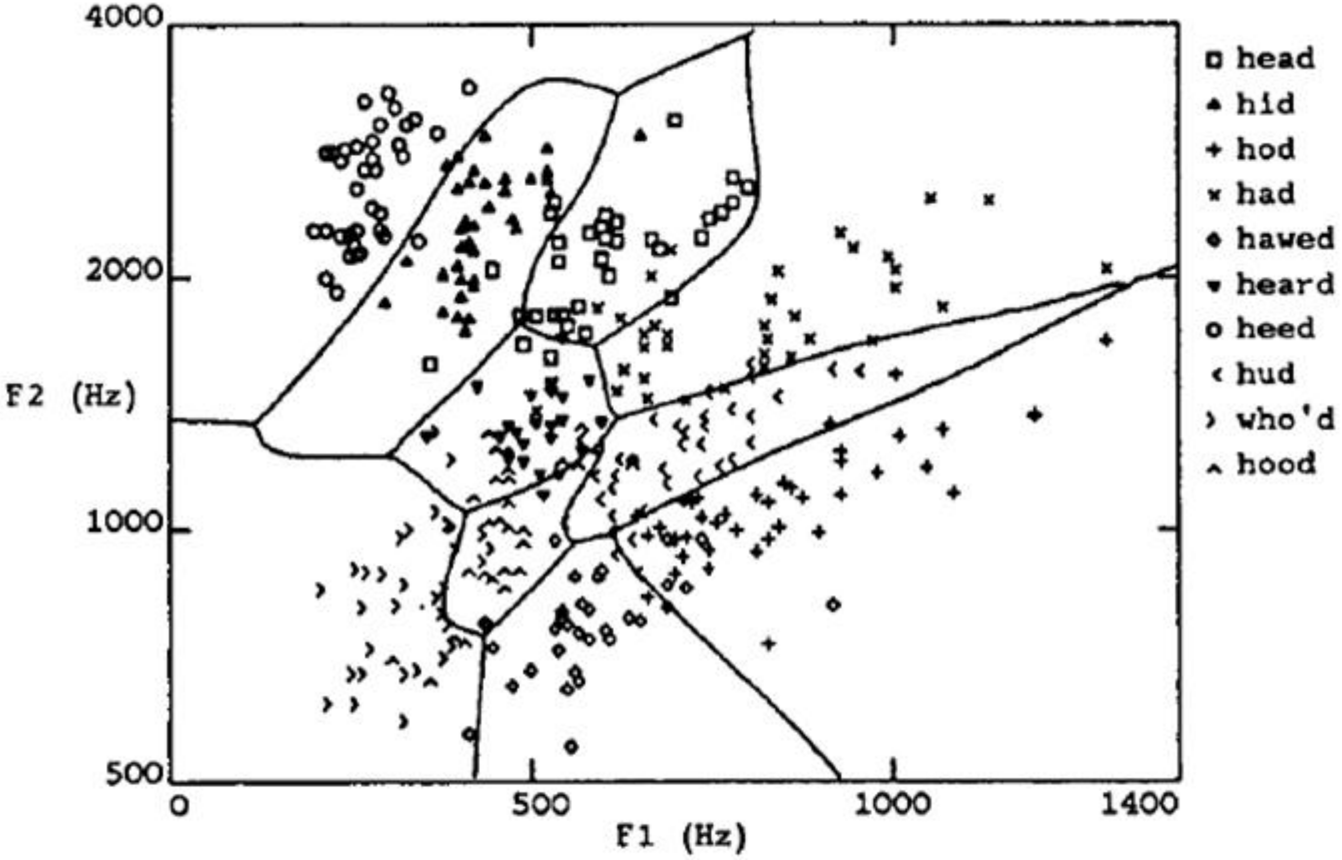
An example

- This network was trained to recognize 1 of 10 vowel sounds occurring in the context “h d” (e.g. “head”, “hid”).
- The inputs have been obtained from a spectral analysis of sound.
- The 10 network outputs correspond to the 10 possible vowel sounds. The net- work prediction is the output whose value is the highest.



An example

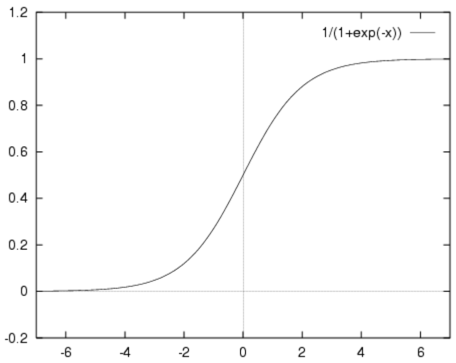
- This plot illustrates the highly non-linear decision surface represented by the learned network.
- Points shown on the plot are test examples distinct from the examples used to train the network.

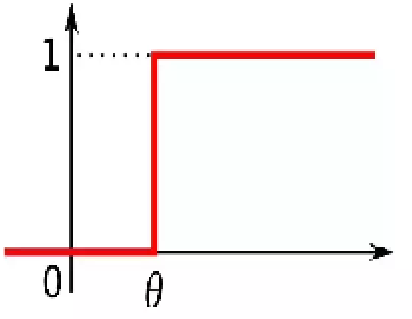


Sigmoid Function

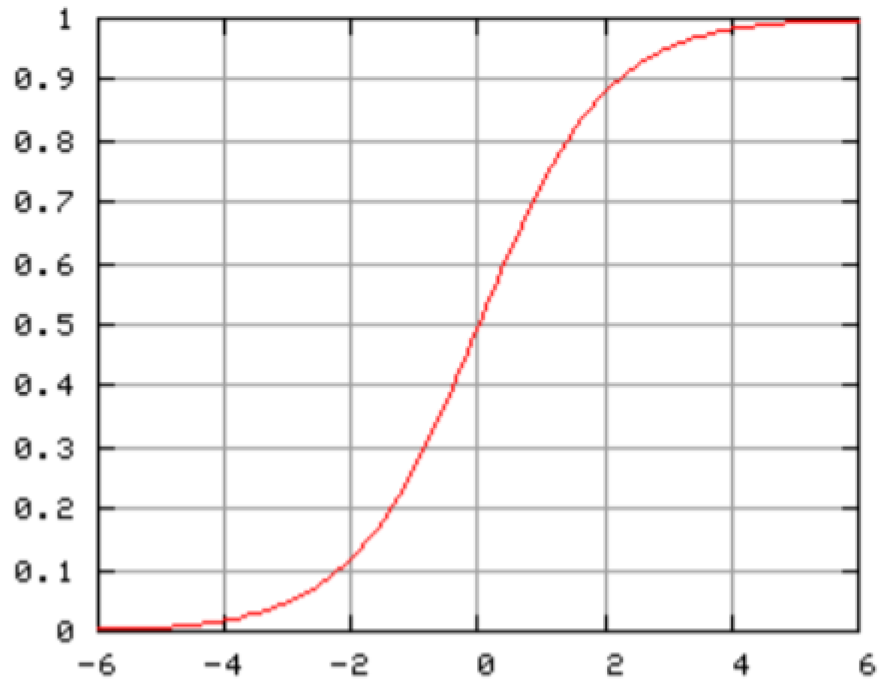
- Sigmoids essentially bring the data within a 0 to 1 range on a smooth curve.

X X X X X X X X X X X X X X
X X X X X X X X X X X X **O** X
X X **O O** X X X X X X X X **O** X
X **O O OO** X X X X X**O O** X
X X X **O O** X X X X X X X X
X X X X X X X X X X X X X X





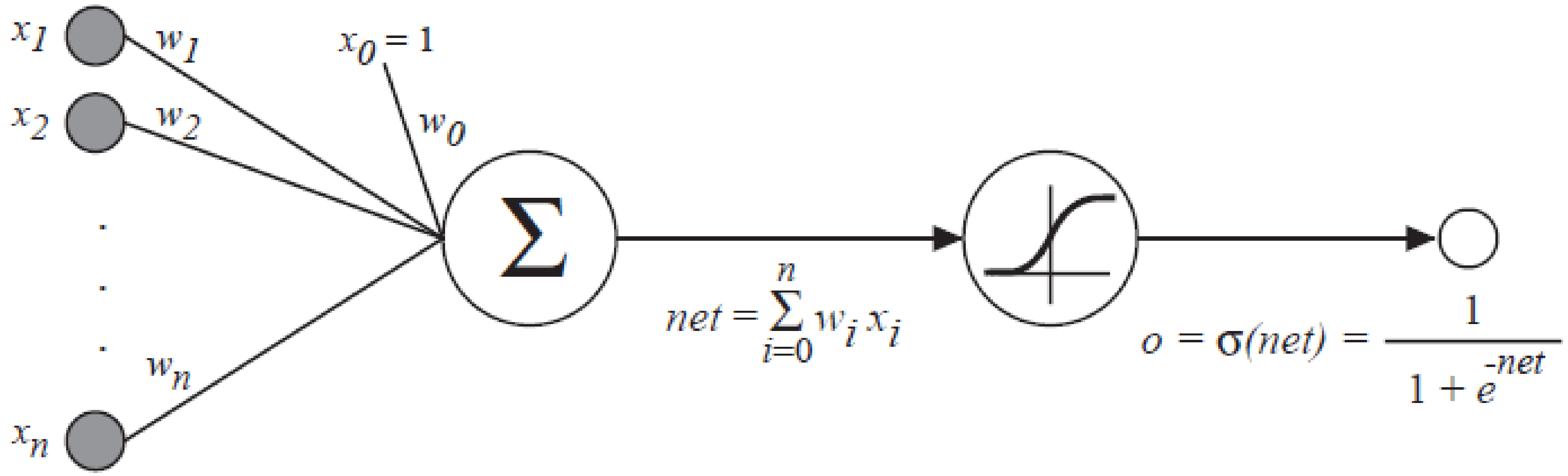
X X X X X X X
X X X **O O O O**
X X **O O O O O O**
O O O O O O O O



$$sig(t) = \frac{1}{1+e^{-t}}$$

•Thinking=**Sigmoid**, Acting=**Threshold**, in general

Sigmoid Threshold Unit



$\sigma(x)$ is the sigmoid function $\frac{1}{1+e^{-x}}$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

We can derive gradient descent rules to train

- One sigmoid unit
- *Multilayer networks* of sigmoid units → Backpropagation

Error Gradient for the Sigmoid Unit

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \left(-\frac{\partial o_d}{\partial w_i} \right) \\
&= -\sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}
\end{aligned}$$

where $net_d = \sum_{i=0}^{n} w_i x_{i,d}$

But

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} o_d(1 - o_d)(t_d - o_d) x_{i,d}$$

Back Propagation Algorithm for Feed Forward networks

function BackProp $(D, \eta, n_{in}, n_{hidden}, n_{out})$

- D is the training set consists of m pairs: $\{(x_i, y_i)^m\}$
- η is the learning rate as an example (0.1)
- n_{in}, n_{hidden} e n_{out} are the numbero of imput hidden and output unit of neural network

Make a feed-forward network with n_{in}, n_{hidden} e n_{out} units

Initialize all the weight to short randomly number (es. [-0.05 0.05])

Repeat until termination condition are verifyed:

For any sample in D:

Forward propagate the network computing the output o_u of every unit u of the network

Back propagate the errors onto the network:

– For every output unit k, compute the error δ_k: $\delta_k = o_k(1-o_k)(t_k - o_k)$

– For every hidden unit h compute the error δ_h: $\delta_h = o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$

– Update the network weight w_{ji}: $w_{ji} = w_{ji} + \Delta w_{ji}, \quad where \;\; \Delta w_{ji} = \eta \delta_j x_{ji}$

(x_{ji} is the input of unit j from coming from unit i)

The Backpropagation Algorithm for a feed-forward 2-layer network of sigmoid units, the stochastic version

Idea: Gradient descent over the entire vector of network weights.

Initialize all weights to small random numbers.

Until satisfied, // *stopping criterion* to be (later) defined

for each training example,

1. input the training example to the network, and compute the network outputs
2. for each output unit k:
   $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
3. for each hidden unit h:
   $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$
4. update each network weight w_{ji}:
   $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$,
   and x_{ji} is the ith input to unit j.

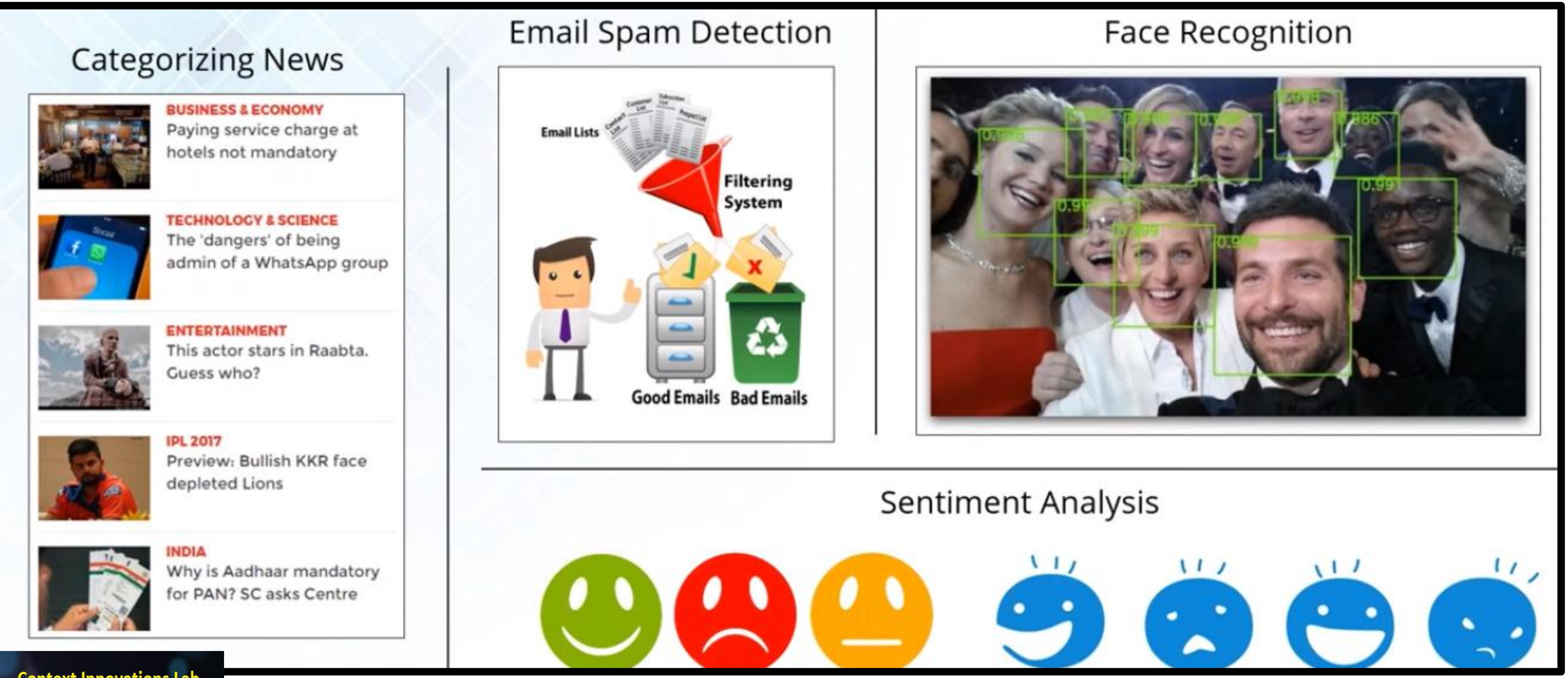
Source Code

- https://github.com/profthyagu

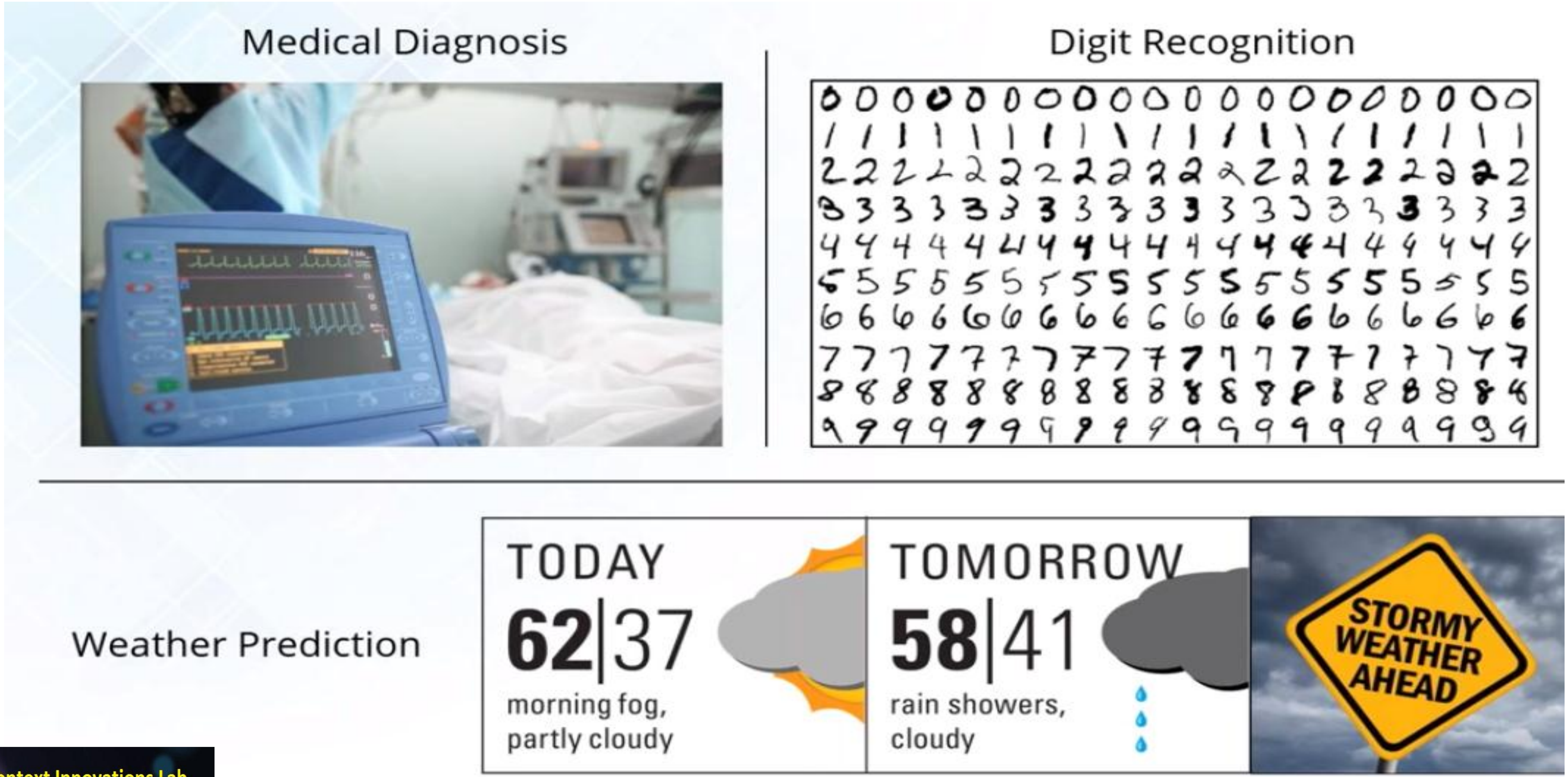
LabProgram 3

- **Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.**

Use cases


Categorizing News
BUSINESS & ECONOMY
Paying service charge at hotels not mandatory
TECHNOLOGY & SCIENCE
The 'dangers' of being admin of a WhatsApp group
ENTERTAINMENT
This actor stars in Raabta. Guess who?
IPL 2017
Preview: Bullish KKR face depleted Lions
INDIA
Why is Aadhaar mandatory for PAN? SC asks Centre
Email Spam Detection
Email Lists
Filtering System
Good Emails
Bad Emails
Face Recognition
Sentiment Analysis

Use cases


Medical Diagnosis
Digit Recognition
Weather Prediction
TODAY
62|37
morning fog,
partly cloudy
TOMORROW
58|41
rain showers,
cloudy
STORMY
WEATHER
AHEAD

Conditional Probability

Definition. The **conditional probability** of an event *A* given that an event *B* has occurred is written: **P(A|B)** and is calculated using:

$P(A|B) = P(A \cap B) / P(B)$ as long as $\boldsymbol{P(B) > 0}$.

Example :

P(A) = 4/52

P(B) = 4/51

P(A and B) = 4/52*4/51= 0.006

$$P\left(\frac{B}{A}\right) = \frac{\text{P(A and B)}}{P(A)} = \frac{0.006}{0.077} = 0.078$$

Bayes Theorem

- Bayes' theorem is stated mathematically as the following equation

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h) =$ prior probability of hypothesis h
- $P(D) =$ prior probability of training data D
- $P(h|D) =$ probability of h given D
- $P(D|h) =$ probability of D given h

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Bayes Theorem

- Bayes' theorem is stated mathematically as the following equation

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B independently of each other;

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Alternative form

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B \mid A)P(A) + P(B \mid \neg A)P(\neg A)}.$$

$P(\neg A)$ is the corresponding probability of the initial degree of belief against *A*, where $1 - P(A) = P(\neg A)$

$P(B \mid \neg A)$ is the conditional probability or likelihood, is the degree of belief in *B*, given that the proposition $\neg A$ is true.

Example of Bayes Theorem $P(A \mid B) = P(B \mid A)P(A) / P(B)$

Consider a drug test that is ***99 percent sensitive***(the true positive rate) and ***99 percent specific*** (the true negative rate). If half a percent ***(0.5 percent)*** of people ***use a drug***, what is the probability a random person with a positive test actually is a user?

$$
\begin{aligned}
P(\text{User} \mid +) &= \frac{P(+ \mid \text{User})P(\text{User})}{P(+)} \\
&= \frac{P(+ \mid \text{User})P(\text{User})}{P(+ \mid \text{User})P(\text{User}) + P(+ \mid \text{Non-user})P(\text{Non-user})} \\
&= \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \\
&\approx 33.2\%
\end{aligned}
$$

Naive Bayes classifier /Bayes Rule

Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

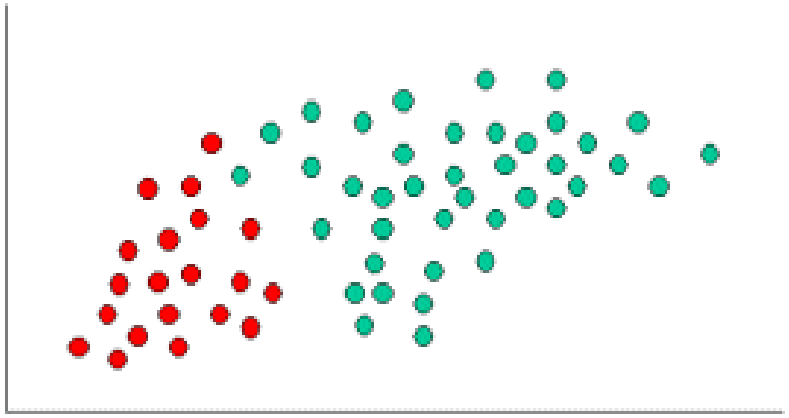
In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C

The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some *k* as follows:

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

Naive Bayes classifier

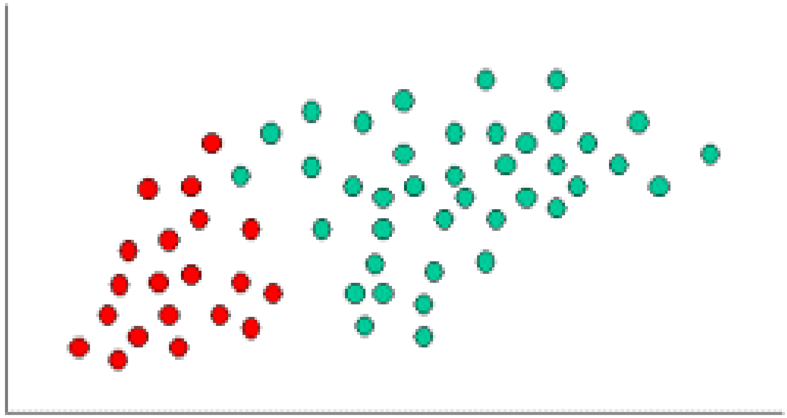
- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.



$$\text{Prior probability for GREEN} \propto \frac{\text{Number of GREEN objects}}{\text{Total number of objects}}$$

$$\text{Prior probability for RED} \propto \frac{\text{Number of RED objects}}{\text{Total number of objects}}$$

Example :Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:



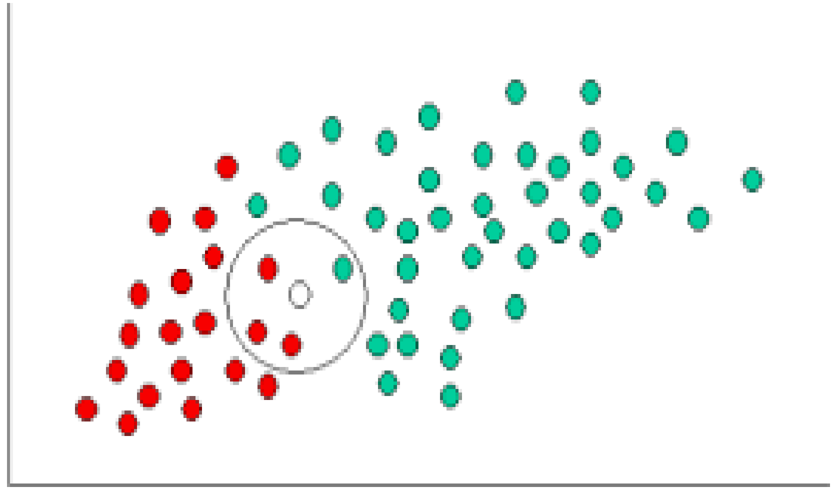
$$\text{Prior probability for GREEN} \propto \frac{40}{60}$$

$$\text{Prior probability for RED} \propto \frac{20}{60}$$

Example :Having formulated our prior probability, we are now ready to classify a new object (WHITE circle).

$$\text{Prior probability for GREEN} \propto \frac{40}{60}$$

$$\text{Prior probability for RED} \propto \frac{20}{60}$$



From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:

$$\text{Probability of } X \text{ given GREEN} \propto \frac{1}{40}$$

$$\text{Probability of } X \text{ given RED} \propto \frac{3}{20}$$

Example :

In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

$$\text{Posterior probability of } X \text{ being GREEN} \propto$$

$$\text{Prior probability of GREEN} \times \text{Likelihood of } X \text{ given GREEN}$$

$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$\text{Posterior probability of } X \text{ being RED} \propto$$

$$\text{Prior probability of RED} \times \text{Likelihood of } X \text{ given RED}$$

$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Event Models

- The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier.
- **For discrete features** like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular.
- For Continuous feature , **Gaussian naive Bayes distributions is popular.**

1. Gaussian naive Bayes

- When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.
- Then, the probability *distribution* of v given a class C_k, $p(x = v \mid C_k)$ can be computed by plugging V into the equation for a Normal distribution parameterized by μ_k and σ_k^2.

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

2. Multinomial naive Bayes

Its is used when we have **discrete data** (e.g. movie ratings ranging 1 and 5 as each rating will have certain **frequency** to represent). In text learning we have the count of each word to predict the class or label

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:

$$\begin{aligned} \log p(C_k \mid \mathbf{x}) &\propto \log \left(p(C_k) \prod_{i=1}^{n} {p_{ki}}^{x_i} \right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x} \end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

2. Multinomial naive Bayes

Its is used when we have **discrete data** (e.g. movie ratings ranging 1 and 5 as each rating will have certain **frequency** to represent). In text learning we have the count of each word to predict the class or label.
The Multinomial Naive Bayes's conditional distribution is:

$$p(\mathbf{x}|C=k) = \mathrm{Multinomial}(n, \mathbf{p}_k) = \frac{(\sum_d x_d)!}{\prod_d x_d!} \prod_d {p_{kd}}^{x_d}$$

where $\mathbf{x}$ is feature and C is class. d is the number of dimension of feature.

When using in text domain: given an ith document's word feature $\mathbf{x}_i = (w_1, \ldots, w_d)$, $d = |Vocabulary|$.

The term frequencies can then be used to compute the maximum-likelihood estimate based on the training data to estimate the class-conditional probabilities in the multinomial model:

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega j} + \alpha \cdot V}$$

where

- x_i: A word from the feature vector $\mathbf{x}$ of a particular sample.
- $\sum tf(x_i, d \in \omega_j)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class ω_j.
- $\sum N_{d \in \omega j}$: The sum of all term frequencies in the training dataset for class ω_j.
- α: An additive smoothing parameter ($\alpha = 1$ for Laplace smoothing).
- V: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text $\mathbf{x}$ can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \ldots \cdot P(x_n \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j)$$

3. Bernoulli naive Bayes

It assumes that all our features are binary such that they take only two values. Means **0s** can represent “word does not occur in the document” and **1s** as "word occurs in the document" .

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

Example 1

- **Import**

```
import pandas as pd
import numpy as np
```

Create Data

```python
# Create an empty dataframe
data = pd.read_csv("C:\\Users\\Dr.Thyagaraju\\Desktop\\Data\\Gender.csv")
# View the data
data
```

	Gender	Height	Weight	Foot_Size
0	male	6.00	180	12
1	male	5.92	190	11
2	male	5.58	170	12
3	male	5.92	165	10
4	female	5.00	100	6
5	female	5.50	150	8
6	female	5.42	130	7
7	female	5.75	150	9

Creating a New Data

```python
# Create an empty dataframe
person = pd.DataFrame()

# Create some feature values for this single row
person['Height'] = [6]
person['Weight'] = [130]
person['Foot_Size'] = [8]

# View the data
person
```

	Height	Weight	Foot_Size
0	6	130	8

Bayes Theorem

Bayes theorem is a famous equation that allows us to make predictions based on data. Here is the classic version of the Bayes theorem:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

This might be too abstract, so let us replace some of the variables to make it more concrete. In a bayes classifier, we are interested in finding out the class (e.g. male or female, spam or ham) of an observation *given* the data:

$$p(\text{class} \mid \mathbf{data}) = \frac{p(\mathbf{data} \mid \text{class}) * p(\text{class})}{p(\mathbf{data})}$$

where:

- class is a particular class (e.g. male)
- $\mathbf{data}$ is an observation's data
- $p(\text{class} \mid \mathbf{data})$ is called the posterior
- $p(\text{data}|\text{class})$ is called the likelihood
- $p(\text{class})$ is called the prior
- $p(\mathbf{data})$ is called the marginal probability

In a bayes classifier, we calculate the posterior (technically we only calculate the numerator of the posterior, but ignore that for now) for every class for each observation. Then, classify the observation based on the class with the largest posterior value. In our example, we have one observation to predict and two possible classes (e.g. male and female), therefore we will calculate two posteriors: one for male and one for female.

$$p(\text{person is male} \mid \textbf{person's data}) = \frac{p(\textbf{person's data} \mid \text{person is male}) * p(\text{person is male})}{p(\textbf{person's data})}$$

$$p(\text{person is female} \mid \textbf{person's data}) = \frac{p(\textbf{person's data} \mid \text{person is female}) * p(\text{person is female})}{p(\textbf{person's data})}$$

Gaussian Naive Bayes Classifier

A gaussian naive bayes is probably the most popular type of bayes classifier. To explain what the name means, let us look at what the bayes equations looks like when we apply our two classes (male and female) and three feature variables (height, weight, and footsize):

$$\text{posterior (male)} = \frac{P(\text{male})\, p(\text{height} \mid \text{male})\, p(\text{weight} \mid \text{male})\, p(\text{foot size} \mid \text{male})}{\text{marginal probability}}$$

$$\text{posterior (female)} = \frac{P(\text{female})\, p(\text{height} \mid \text{female})\, p(\text{weight} \mid \text{female})\, p(\text{foot size} \mid \text{female})}{\text{marginal probability}}$$

$$p(\text{height} \mid \text{female}) = \frac{1}{\sqrt{2\pi\text{variance of female height in the data}}} e^{-\frac{(\text{observation's height} - \text{average height of females in the data})^2}{2\text{variance of female height in the data}}}$$

- marginal probability marginal probability is probably one of the most confusing parts of Bayesian approaches.

```
# Calculate Priors
```

```python
# Number of males
n_male = data['Gender'][data['Gender'] == 'male'].count()

# Number of males
n_female = data['Gender'][data['Gender'] == 'female'].count()

# Total rows
total_ppl = data['Gender'].count()
```

```python
# Number of males divided by the total rows
P_male = n_male/total_ppl

# Number of females divided by the total rows
P_female = n_female/total_ppl
```

```
# Calculate Likelihood
```

```python
# Group the data by gender and calculate the means of each feature
data_means = data.groupby('Gender').mean()

# View the values
data_means
```

	Height	Weight	Foot_Size
Gender			
female	5.4175	132.50	7.50
male	5.8550	176.25	11.25

```python
# Group the data by gender and calculate the variance of each feature
data_variance = data.groupby('Gender').var()

# View the values
data_variance
```

	Height	Weight	Foot_Size
Gender			
female	0.097225	558.333333	1.666667
male	0.035033	122.916667	0.916667

Create All the vairables Needed

```
# Means for male
male_height_mean = data_means['Height'][data_variance.index == 'male'].values[0]
male_weight_mean = data_means['Weight'][data_variance.index == 'male'].values[0]
male_footsize_mean = data_means['Foot_Size'][data_variance.index == 'male'].values[0]

# Variance for male
male_height_variance = data_variance['Height'][data_variance.index == 'male'].values[0]
male_weight_variance = data_variance['Weight'][data_variance.index == 'male'].values[0]
male_footsize_variance = data_variance['Foot_Size'][data_variance.index == 'male'].values[0]

# Means for female
female_height_mean = data_means['Height'][data_variance.index == 'female'].values[0]
female_weight_mean = data_means['Weight'][data_variance.index == 'female'].values[0]
female_footsize_mean = data_means['Foot_Size'][data_variance.index == 'female'].values[0]

# Variance for female
female_height_variance = data_variance['Height'][data_variance.index == 'female'].values[0]
female_weight_variance = data_variance['Weight'][data_variance.index == 'female'].values[0]
female_footsize_variance = data_variance['Foot_Size'][data_variance.index == 'female'].values[0]
```

```
# Create a function to calculate the probability density of each of the terms of the likelihood (e.g. P(height|female))
```

```
# Create a function that calculates p(x | y):
def p_x_given_y(x, mean_y, variance_y):

    # Input the arguments into a probability density function
    p = 1/(np.sqrt(2*np.pi*variance_y)) * np.exp((-(x-mean_y)**2)/(2*variance_y))

    # return p
    return p
```

Apply Bayes Classifier To New Data Point

Alright! Our bayes classifier is ready. Remember that since we can ignore the marginal probability (the demoninator), what we are actually calculating is this:

$$\text{numerator of the posterior} = P(\text{female})\, p(\text{height} \mid \text{female})\, p(\text{weight} \mid \text{female})\, p(\text{foot size} \mid \text{female})$$

To do this, we just need to plug in the values of the unclassified person (height = 6), the variables of the dataset (e.g. mean of female height), and the function (`p_x_given_y`) we made above:

Apply Bayes Classifier To New Data Point

```python
# Numerator of the posterior if the unclassified observation is a male
P_male * \
p_x_given_y(person['Height'][0], male_height_mean, male_height_variance) * \
p_x_given_y(person['Weight'][0], male_weight_mean, male_weight_variance) * \
p_x_given_y(person['Foot_Size'][0], male_footsize_mean, male_footsize_variance)
```

```
6.197071843878078e-09
```

```python
# Numerator of the posterior if the unclassified observation is a female
P_female * \
p_x_given_y(person['Height'][0], female_height_mean, female_height_variance) * \
p_x_given_y(person['Weight'][0], female_weight_mean, female_weight_variance) * \
p_x_given_y(person['Foot_Size'][0], female_footsize_mean, female_footsize_variance)
```

```
0.0005377909183630018
```

Because the numerator of the posterior for female is greater than male, then we predict that the person is female.

Source Code

- https://github.com/profthyagu

Lab Program 4

- **Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Libraries can be used to write the program. Calculate the accuracy, precision, and recall for your data set.**

Learning to Classify Text – Algorithm

S1: LEARN_NAIVE_BAYES_TEXT (*Examples*, V)

S2: CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

- *Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms* ***$P(w_k \mid v_j)$,*** *describing the probability that a randomly drawn word from a document in* ***class v_j*** *will be the English word* ***wk****. It also learns the class prior probabilities* ***$P(v_j)$.***

S1: LEARN_NAIVE_BAYES_TEXT (*Examples*, V)

[V: Class , W: Word, doc : Documents]

1. *collect all words and other tokens that occur in Examples*

- ***Vocabulary*** $\leftarrow$ all distinct words and other tokens in *Examples*

2. *calculate the required* ***P(v_j) and P(w_k | v_j)*** *probability terms*

- For each target value v_j in *V* do

$$P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$$

  - ***$docs_j$*** $\leftarrow$ **subset of *Examples* for which the target value is *v_j***
  - ***$Text_j$*** $\leftarrow$ **a single document created by concatenating all members of *$docs_j$***
  - ***n*** $\leftarrow$ **total number of words in *$Text_j$* (counting duplicate words multiple times)**
  - **for each word *w_k* in *Vocabulary***

$$P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$$

n_k $\leftarrow$ **number of times word *w_k* occurs in *$Text_j$***

S2:CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

- *positions* $\leftarrow$ all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB} where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

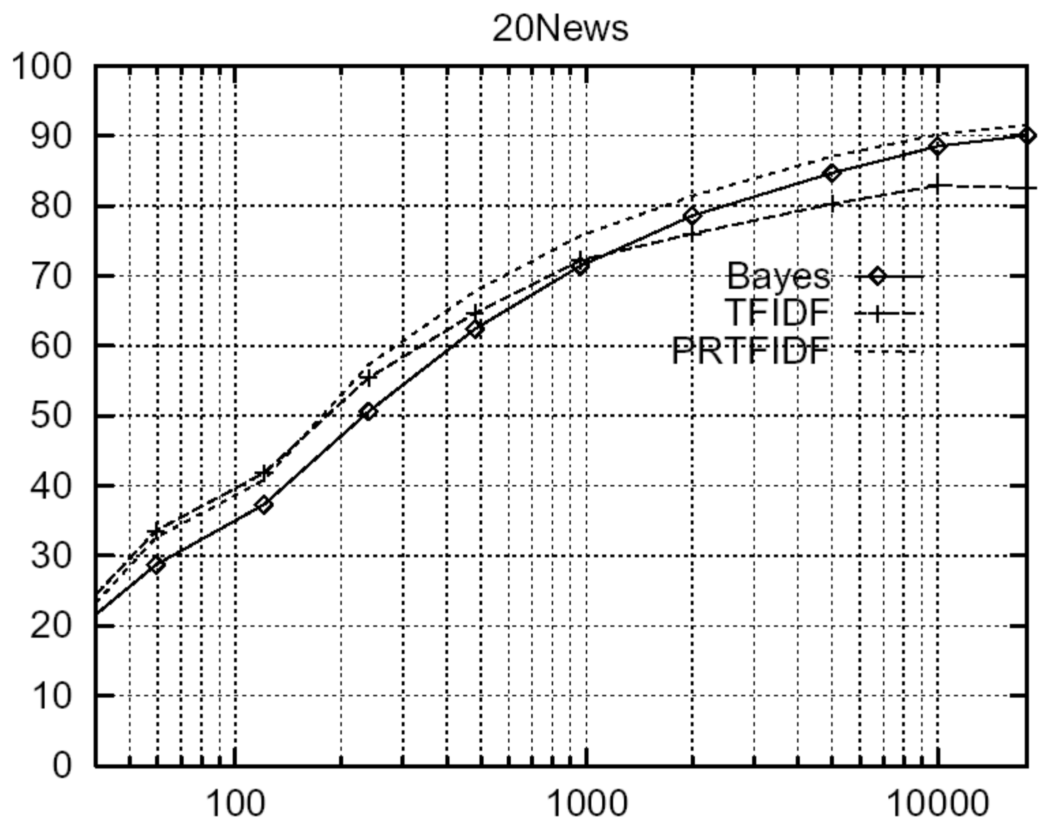
Twenty NewsGroups

- Given 1000 training documents from each group Learn to classify new documents according to which newsgroup it came from

comp.graphics	**misc.forsale**	**alt.atheism**	**sci.space**
comp.os.ms-windows.misc	**rec.autos**	**soc.religion.christian**	**sci.crypt**
comp.sys.ibm.pc.hardware	**rec.motorcycles**	**talk.religion.misc**	**sci.electronics**
comp.sys.mac.hardware	**rec.sport.baseball**	**talk.politics.mideast**	**sci.med**
comp.windows.x	**rec.sport.hockey**	**talk.politics.misc**	
		talk.politics.guns	

- Naive Bayes: 89% classification accuracy

Learning Curve for 20 Newsgroups



- Accuracy vs. Training set size (1/3 withheld for test)

Example :

- In the example, we are given a sentence " ***A very close game***", a training set of five sentences (as shown below), and their corresponding category (Sports or Not Sports).
- The goal is to build a Naive Bayes classifier that will tell us which category the sentence " ***A very close game***" belongs to.
- Applying a Naive Bayes classifier, thus the strategy would be calculating the probability of both "A very close game **is Sports**", as well as it's ***Not Sports.*** The one with the higher probability will be the result.

Text	Category
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Step 1: Feature Engineering

- **word frequencies**, i.e., counting the occurrence of every word in the document.
- ***P(a very close game)*** = P(a) X P(very) X P(close) X P(game)
- ***P(a very close game | Sports)*** = P(a|Sports) X P(Very|Sports) X P(close|Sports) X P(game|Sports)
- **P(*a very close game* | *Not Sports*)** = P(*a* | *Not Sports*) x P(*very* | *Not Sports*) x P(*close* | *Not Sports*) x P(*game* | *Not Sports*)

Step 2: Calculating the probabilities

- Here , the word "close" does not exist in the category Sports, thus **P(*close* |*Sports*)** = 0, leading to **P(*a very close game* | *Sports*)**=0.

- Given an observation x = (x1, …, xd) from a **multinomial distribution** with N trials and parameter vector θ = (θ1, …, θd), a "smoothed" version of the data gives the estimator.

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

- where the pseudo count **α > 0** is the smoothing parameter (α = 0 corresponds to no smoothing)

Word	P(word | Sports)	P(word | Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

$$P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$$

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 4.61 \times 10^{-5}$$
$$= 0.0000461$$

$$\mathrm{P(a — Not\ \ Sports)} \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times P(game|Not\ Sports) \times P(Not\ Sports)$$
$$= 1.43 \times 10^{-5}$$
$$= 0.0000143$$

As seen from the results shown below, P(a very close game | Sports) gives a higher probability, suggesting that the sentence belongs to the Sports category.

Multinomial Naive Bayes

Term Frequency

A alternative approach to characterize text documents — rather than binary values — is the *term frequency (tf(t, d))*. The term frequency is typically defined as the number of times a given term *t* (i.e., word or token) appears in a document *d* (this approach is sometimes also called *raw frequency*). In practice, the term frequency is often normalized by dividing the raw term frequency by the document length.

$$\text{normalized term frequency} = \frac{tf(t,d)}{n_d}$$

where

- $tf(t,d)$: Raw term frequency (the count of term t in document d).
- n_d: The total number of terms in document d.

The term frequencies can then be used to compute the maximum-likelihood estimate based on the training data to estimate the class-conditional probabilities in the multinomial model:

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega j} + \alpha \cdot V}$$

where

- x_i: A word from the feature vector $\mathbf{x}$ of a particular sample.
- $\sum tf(x_i, d \in \omega_j)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class ω_j.
- $\sum N_{d \in \omega j}$: The sum of all term frequencies in the training dataset for class ω_j.
- α: An additive smoothing parameter ($\alpha = 1$ for Laplace smoothing).
- V: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text $\mathbf{x}$ can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \ldots \cdot P(x_n \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j)$$

Confusion Matrix

- A confusion matrix is a summary of prediction results on a classification problem.
- The number of correct and incorrect predictions are summarized with count values and broken down by each class.
- **The confusion matrix shows the ways in which your classification model is confused when it makes predictions.**

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Source Code

- GITHUB LINK: https://github.com/profthyagu

Lab Program5

- **Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Python ML library API.**

Bayesian Network (BAYESIAN BELIEF NETWORKS)

- Bayesian Belief networks describe conditional independence among *subsets* of variables

$\rightarrow$ allows combining prior knowledge about (in)dependencies among variables with observed training data

(also called Bayes Nets)

Conditional Independence

- **Definition:** X is *conditionally independent* of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z; that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

more compactly, we write

$$P(X|Y, Z) = P(X|Z)$$

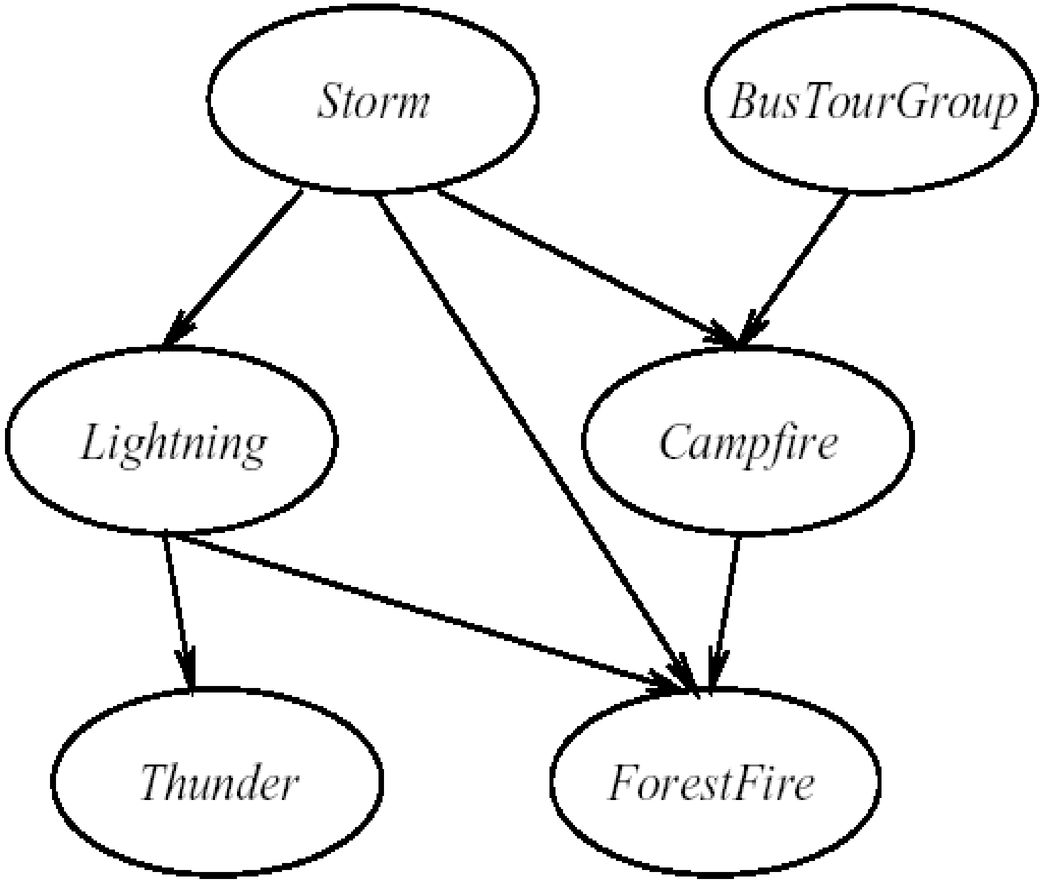
- Example: *Thunder* is conditionally independent of *Rain*, given *Lightning*

$$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$$

- Naive Bayes uses cond. indep. to justify

$$P(X, Y|Z) = P(X|Y, Z)\ P(Y|Z) = P(X|Z)\ P(Y|Z)$$

Bayesian Belief Network (1/2)



	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

- Network represents a set of conditional independence assertions:
  - Each node is asserted to be conditionally independent of its non descendants, given its immediate predecessors.
  - Directed acyclic graph

Bayesian Belief Network (2/2)

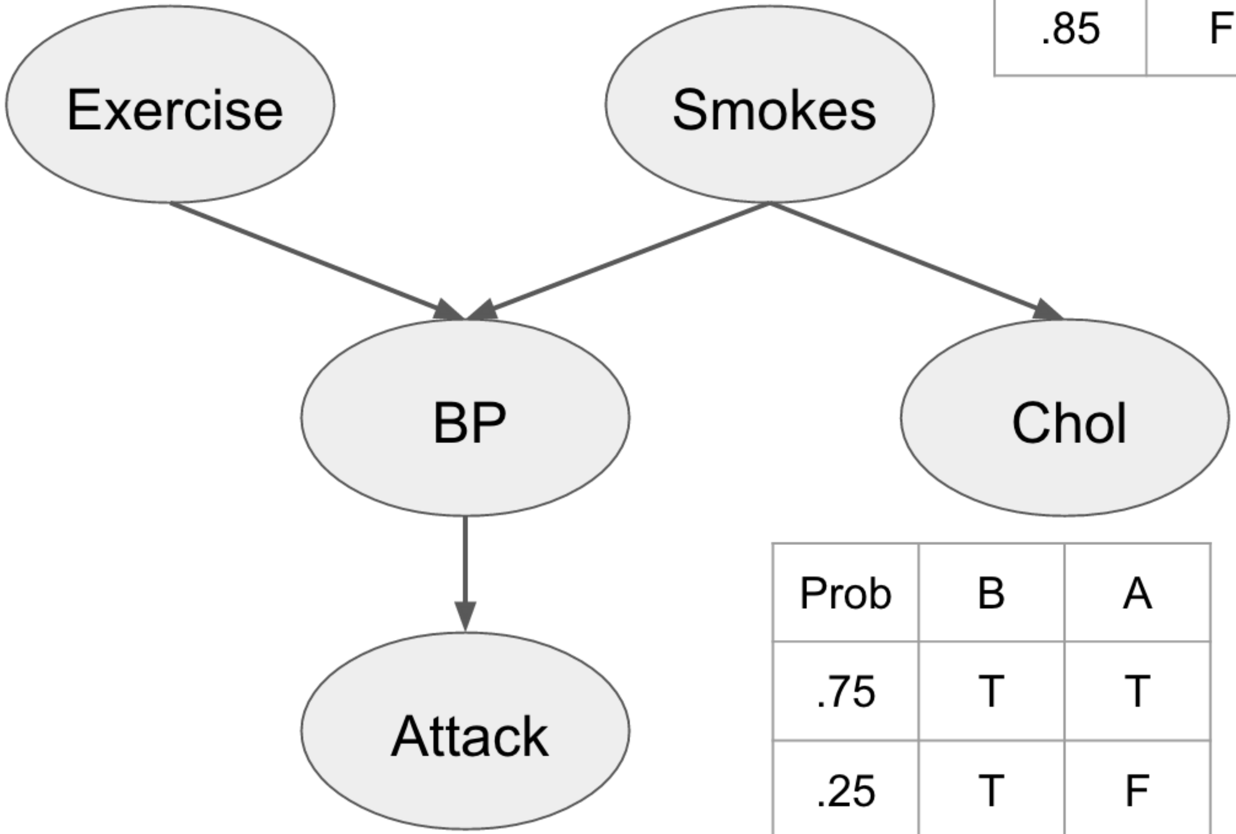
- Represents joint probability distribution over all variables
  - e.g., *P*(*Storm*, *BusTourGroup*, . . . , *ForestFire*)
  - in general,

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

    where *Parents*(Y_i) denotes immediate predecessors of Y_i in graph
  - so, joint distribution is fully defined by graph, plus the P(y_i|*Parents*(Y_i))

Prob	E
.4	T
.6	F

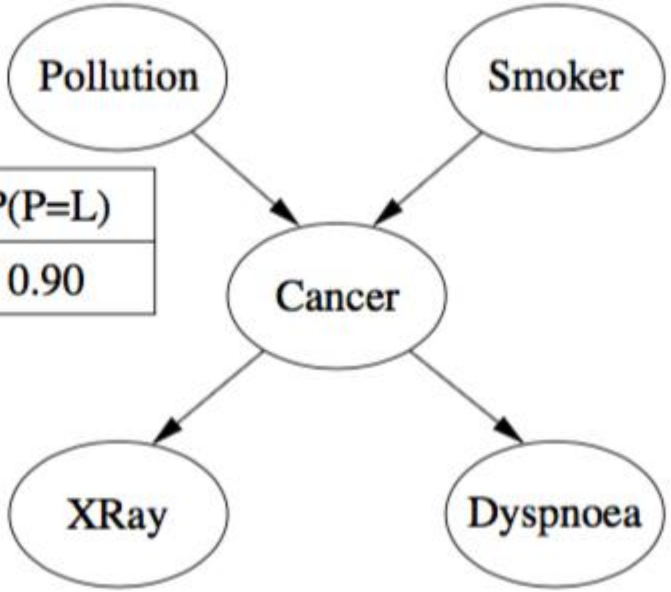
Prob	S
.15	T
.85	F



Prob	E	S	B
.45	T	T	T
.55	T	T	F
.05	T	F	T
.95	T	F	F
.95	F	T	T
.05	F	T	F
.55	F	F	T
.45	F	F	F

Prob	S	C
.8	T	T
.2	T	F
.4	F	T
.6	F	F

Prob	B	A
.75	T	T
.25	T	F
.05	F	T
.95	F	F



P(P=L)
0.90

P(S=T)
0.30

P	S	P(C=T|P,S)
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

C	P(X=pos|C)
T	0.90
F	0.20

C	P(D=T|C)
T	0.65
F	0.30

FIGURE 2.1

A BN for the lung cancer problem.

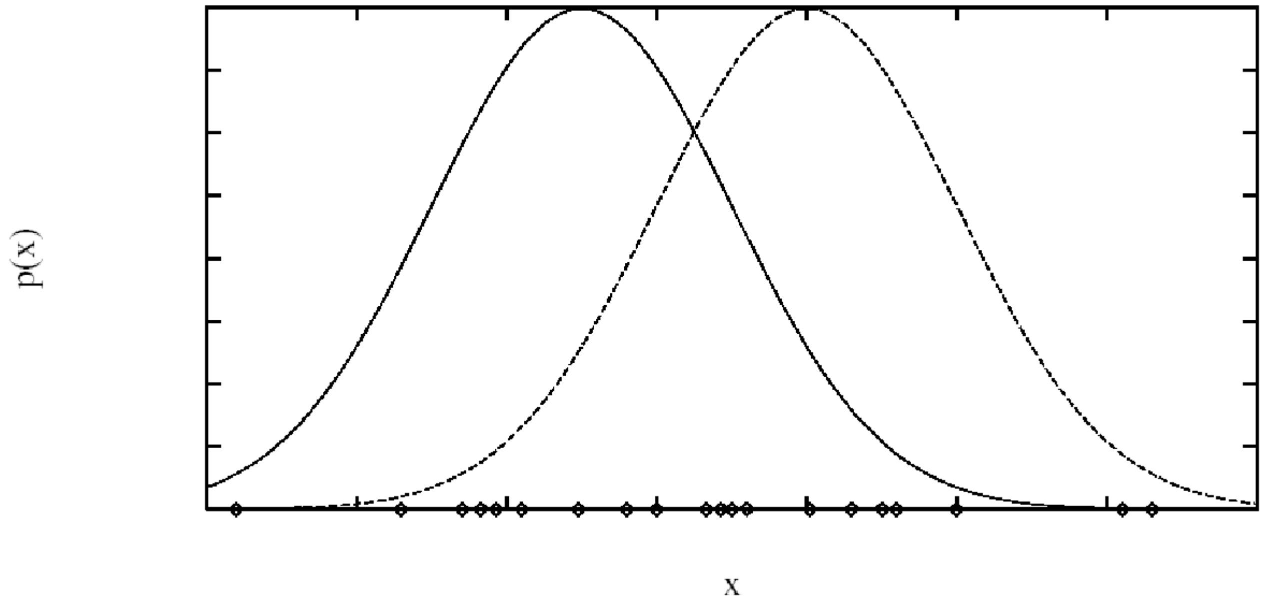
Source Code

- https://github.com/profthyagu

Lab Program 6

- Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Python ML library classes/API in the program.

Generating Data from Mixture of *k* Gaussians



- Each instance x generated by
  1. Choosing one of the *k* Gaussians with uniform probability
  2. Generating an instance at random according to that Gaussian

Gaussian Distribution

- Univariate Gaussian Distribution

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathbf{e}^{-\frac{(\mathbf{x}-\boldsymbol{\mu})^2}{2\sigma^2}}$$

mean

variance

- Multi-Variate Gaussian Distribution

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{T} \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}$$

mean

covariance

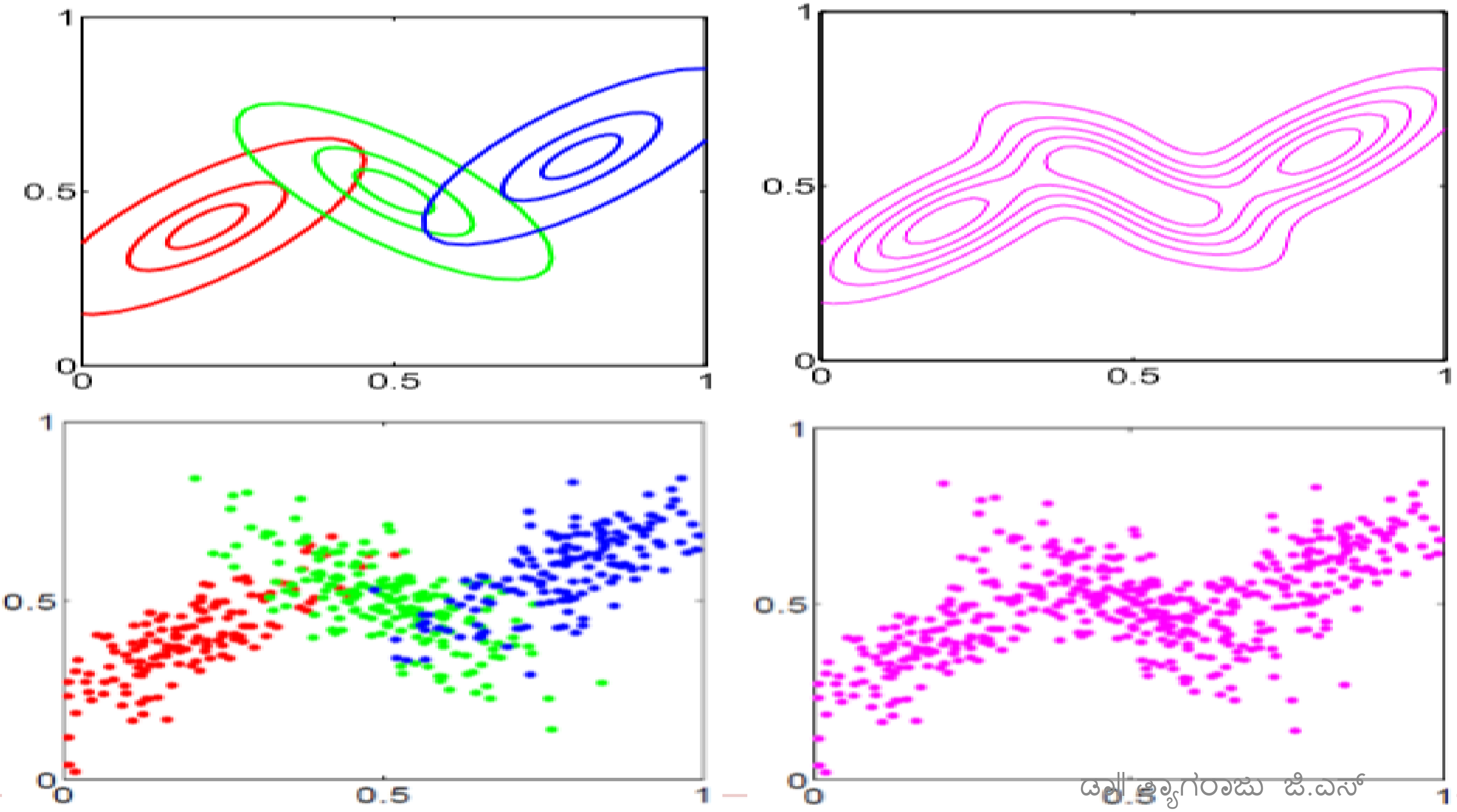
Gaussian Mixtures

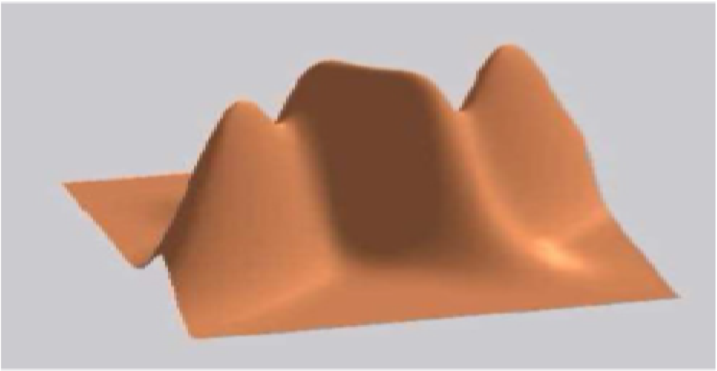
- Linear super-position of Gaussians

$$\mathbf{p(x)} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \pi_{\mathbf{k}} \mathcal{N}(\mathbf{x} \mid \mu_{\mathbf{k}}, \Sigma_{\mathbf{k}})$$

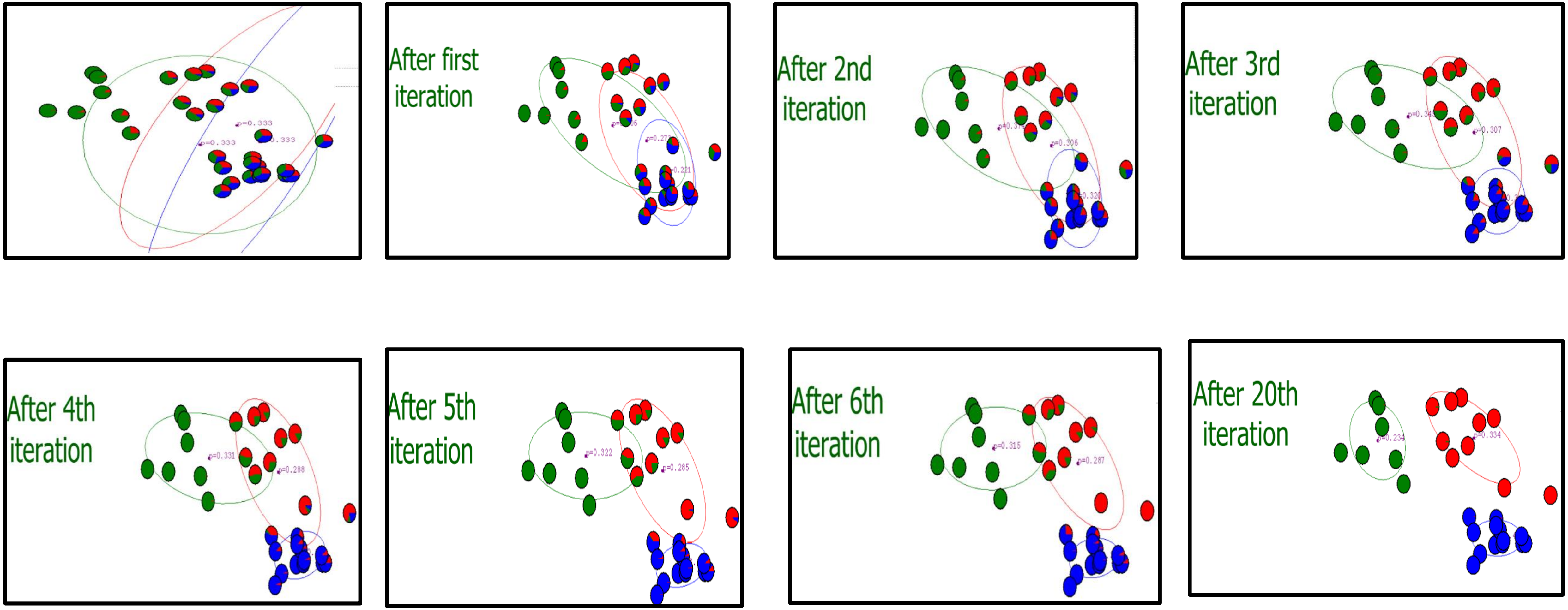
Number of Gaussians

Mixing coefficient: weightage for each Gaussian dist.





GMM : Example


After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
After 20th iteration

Expectation Maximization (EM) Algorithm

- When to use:
  - Filling in missing data in samples
  - Discovering the value of latent variables
  - Estimating the parameters of HMMs
  - Estimating parameters of finite mixtures
  - Unsupervised learning of clusters
  - Semi-supervised classification and clustering

Expectation Maximization (EM) Algorithm

- EM is typically used to compute maximum likelihood estimates given incomplete samples.
- The EM algorithm estimates the parameters of a model iteratively.
    - Starting from some initial guess, each iteration consists of
        - an E step (Expectation step)
        - an M step (Maximization step)

EM Algorithm

- Given:
  - Instances from X generated by mixture of k Gaussian distributions
  - Unknown means $<\mu_1,\ldots,\mu_k>$ of the k Gaussians
  - Don't know which instance x_i was generated by which Gaussian
- Determine:
  - Maximum likelihood estimates of $<\mu_1,\ldots,\mu_k>$

- EM Algorithm:
- Pick random initial $h = <\mu_1, \mu_2>$ then iterate

E step: Calculate the expected value $E[z_{ij}]$ of each ***hidden variable z_ij***, assuming the current hypothesis $h = <\mu_1, \mu_2>$ holds.

$$E[z_{ij}] = \frac{p(x = x_i|\mu = \mu_j)}{\Sigma_{n=1}^{2} p(x = x_i|\mu = \mu_n)} = \frac{e^{-\frac{1}{2\sigma^2}(x_i-\mu_j)^2}}{\Sigma_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i-\mu_n)^2}}$$

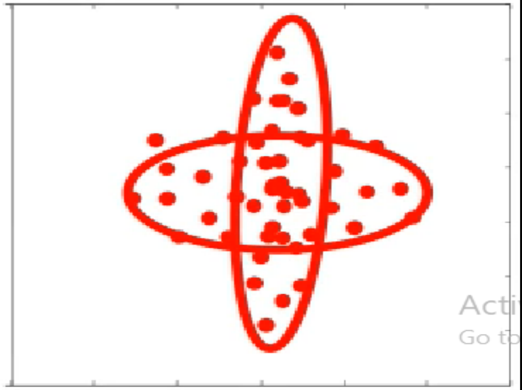
M step: Calculate a new maximum likelihood hypothesis $h' = <\mu'_1, \mu'_2>$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above. Replace $h = <\mu_1, \mu_2>$ by $h' = <\mu'_1, \mu'_2>$.

$$\mu_j \leftarrow \frac{\Sigma_{i=1}^{m} E[z_{ij}]\ x_i}{\Sigma_{i=1}^{m} E[z_{ij}]}$$

Mixtures of Gaussians

- K-means algorithm
  - Assigned each example to exactly one cluster
  - What if clusters are overlapping?
    - Hard to tell which cluster is right
    - Maybe we should try to remain uncertain
  - Used Euclidean distance
  - What if cluster has a non-circular shape?

- Gaussian mixture models
  - Clusters modeled as Gaussians
    - Not just by their mean
  - EM algorithm: assign data to cluster with some *probability*
  - Gives probability model of x! ("generative")



Mixtures of Gaussians

- Start with parameters describing each cluster
- Mean μ_c , variance σ_c , "size" π_c
- Probability distribution: $p(x) = \sum_c \pi_c \, \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$

- Equivalent "latent variable" form:

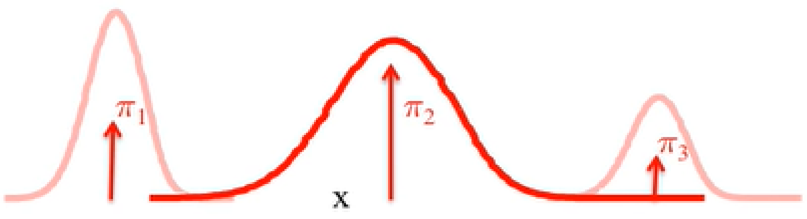
$$p(z = c) = \pi_c$$
$$p(x|z = c) = \mathcal{N}(x \; ; \; \mu_c, \sigma_c)$$

Select a mixture component with probability π

Sample from that component's Gaussian

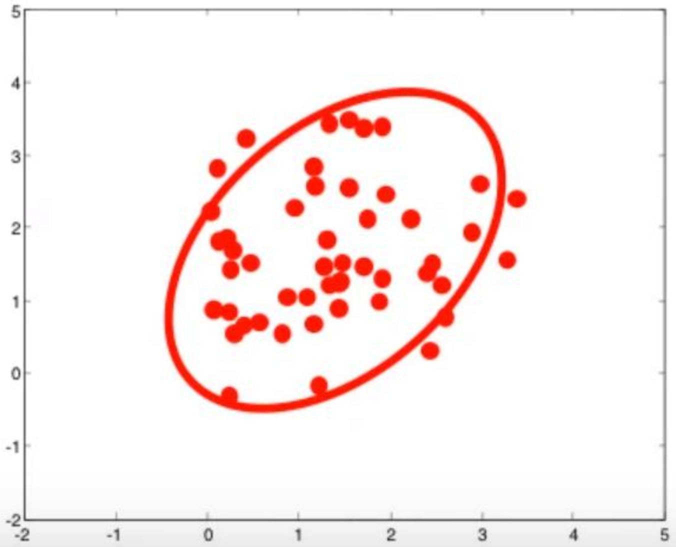
"Latent assignment" z:
we observe x, but z is hidden

p(x) = marginal over x



Multivariate Gaussian models

$$\mathcal{N}(\underline{x}\ ;\ \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} (\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) \right\}$$



Maximum Likelihood estimates

$$\hat{\mu} = \frac{1}{m} \sum_i x^{(i)}$$

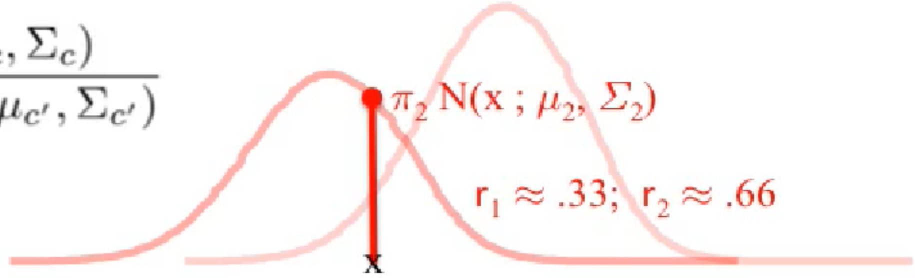
$$\hat{\Sigma} = \frac{1}{m} \sum_i (x^{(i)} - \hat{\mu})^T (x^{(i)} - \hat{\mu})$$

We' ll model each cluster using one of these Gaussian "bells"…

EM Algorithm: E-step

- Start with clusters: Mean μ_c, Covariance Σ_c, "size" π_c
- E-step ("Expectation")
  - For each datum (example) x_i,
  - Compute "r_{ic}", the probability that it belongs to cluster c
    - Compute its probability under model c
    - Normalize to sum to one (over clusters c)

$$r_{ic} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$



  - If x_i is very likely under the c^{th} Gaussian, it gets high weight
  - Denominator just makes r's sum to one

EM Algorithm: M-step

- Start with assignment probabilities r_{ic}
- Update parameters: mean μ_c, Covariance Σ_c, "size" π_c

- M-step ("Maximization")
  - For each cluster (Gaussian) z = c,
  - Update its parameters using the (weighted) data points

$$m_c = \sum_i r_{ic}$$ Total responsibility allocated to cluster c

$$\pi_c = \frac{m_c}{m}$$ Fraction of total assigned to cluster c

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x^{(i)}$$

Weighted mean of assigned data

$$\Sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x^{(i)} - \mu_c)^T (x^{(i)} - \mu_c)$$

Weighted covariance of assigned data (use new weighted means here)

Expectation-Maximization

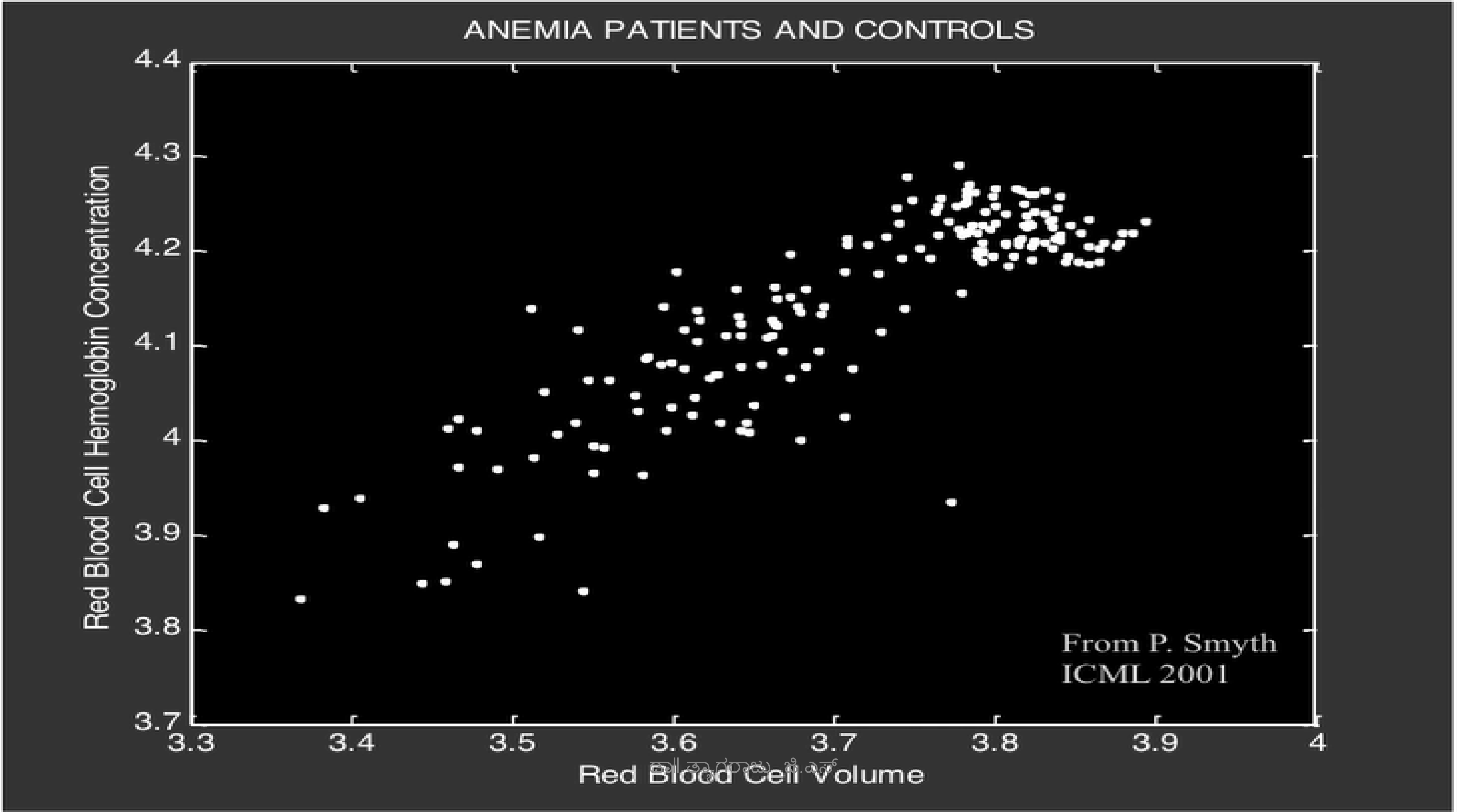
- Each step increases the log-likelihood of our model

$$\log p(\underline{X}) = \sum_i \log \left[\sum_c \pi_c \, \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c) \right]$$

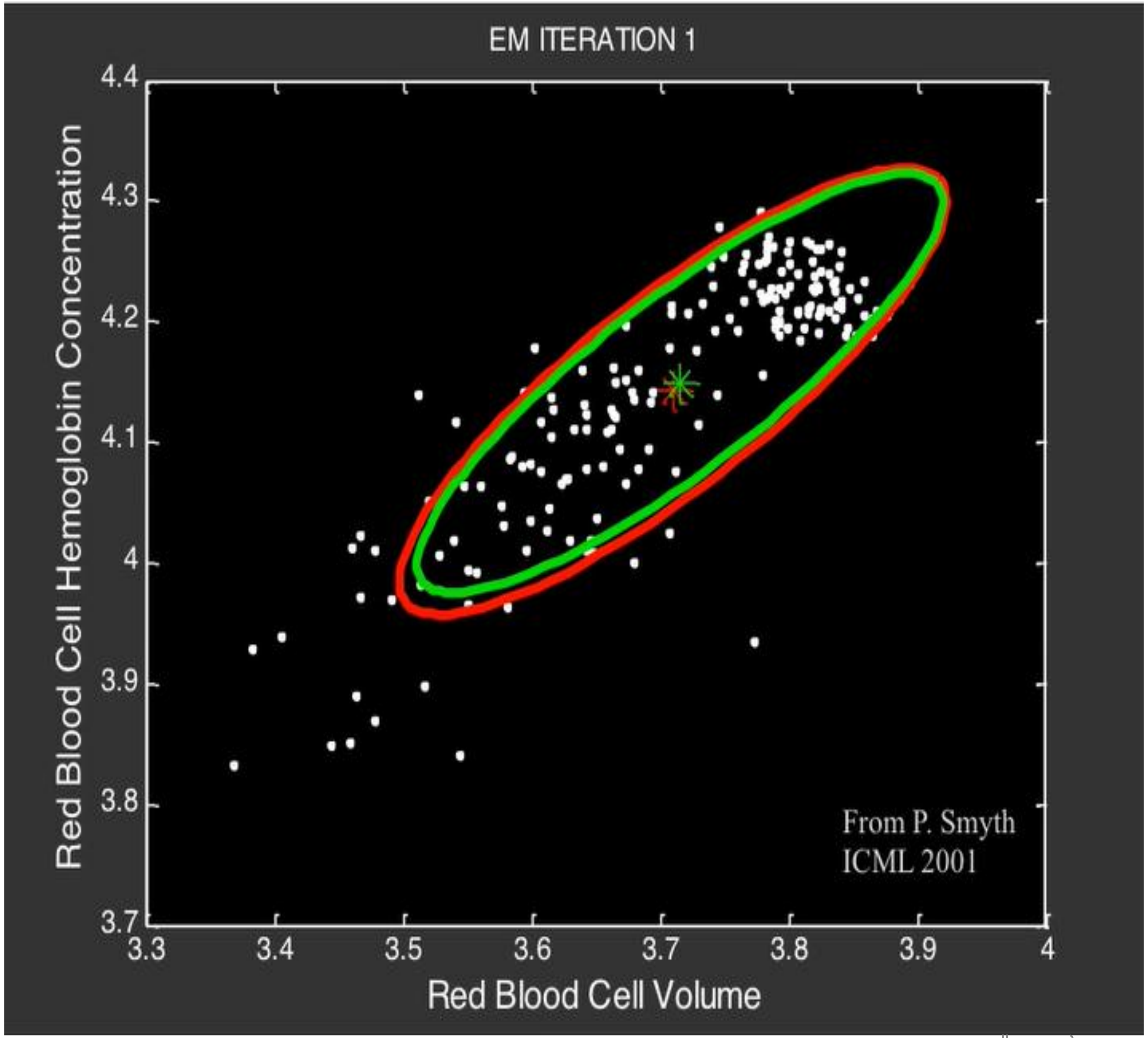
(we won't derive this here, though)

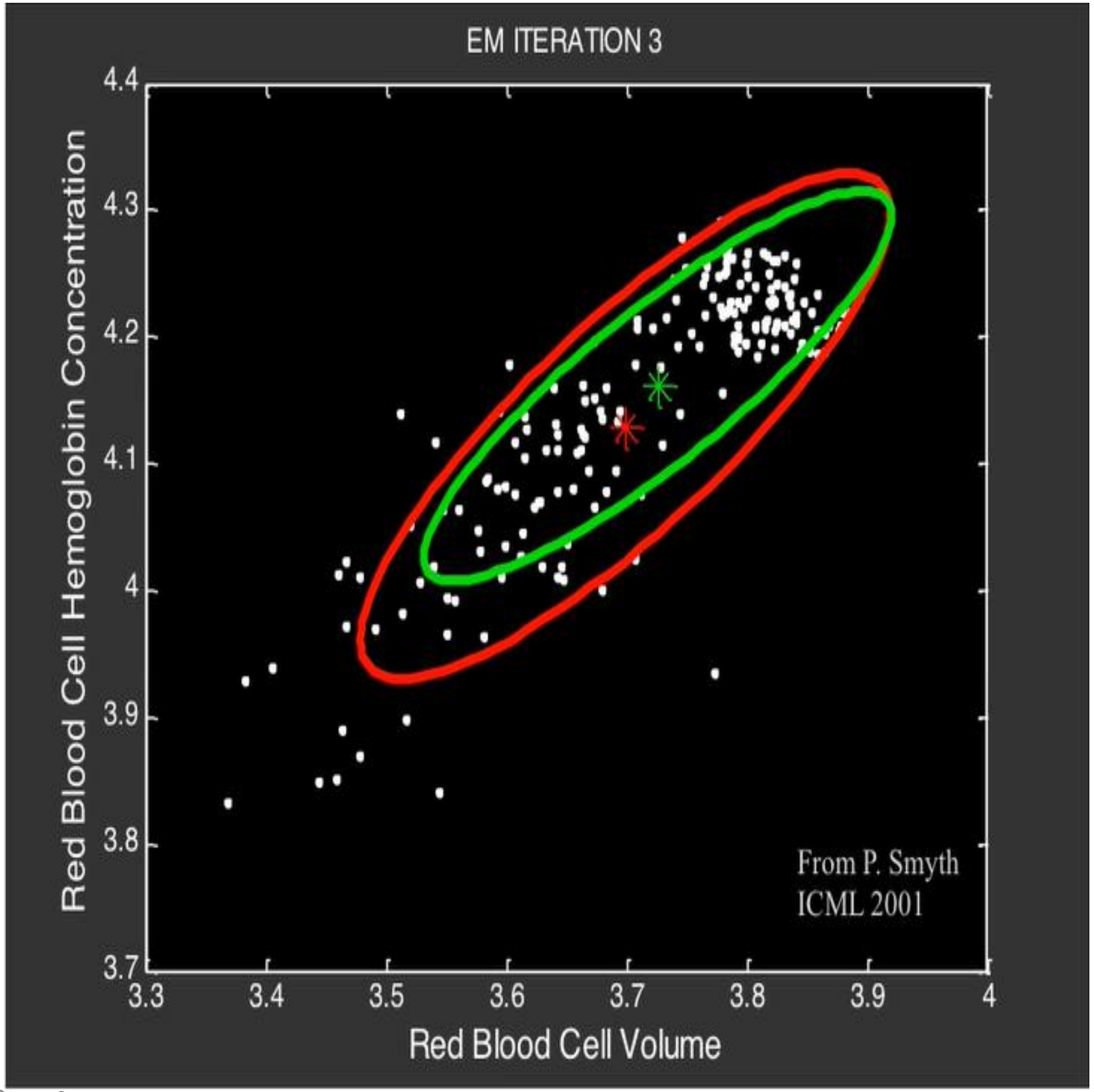
- Iterate until convergence
  - Convergence guaranteed – another ascent method
  - Local optima: initialization often important
- What should we do
  - If we want to choose a single cluster for an "answer"?
  - With new data we didn't see during training?
- Choosing the number of clusters
  - Can use penalized likelihood of training data (like k-means
  - True probability model: can use log-likelihood of test data, log p(x')

GMM : Example2

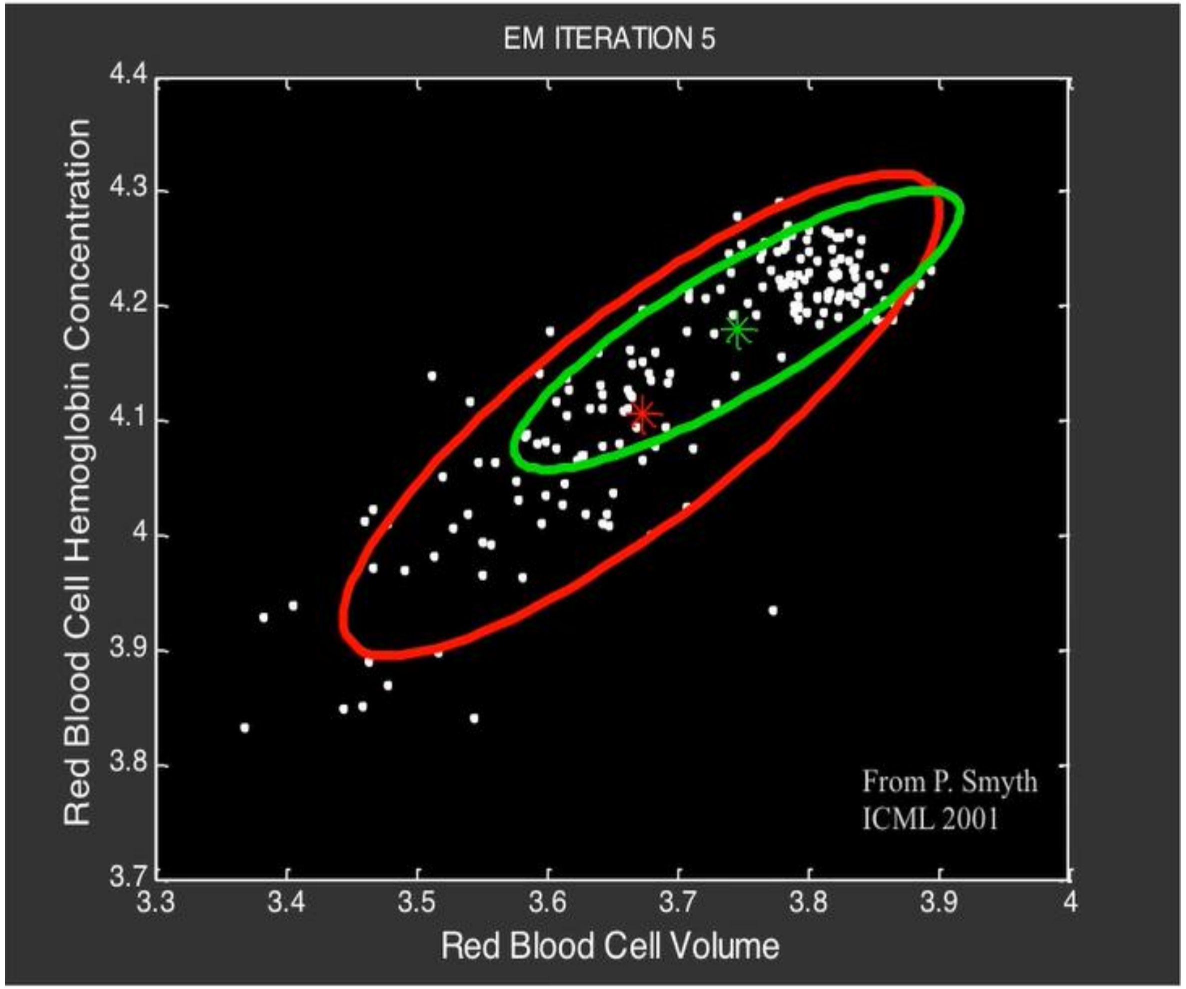


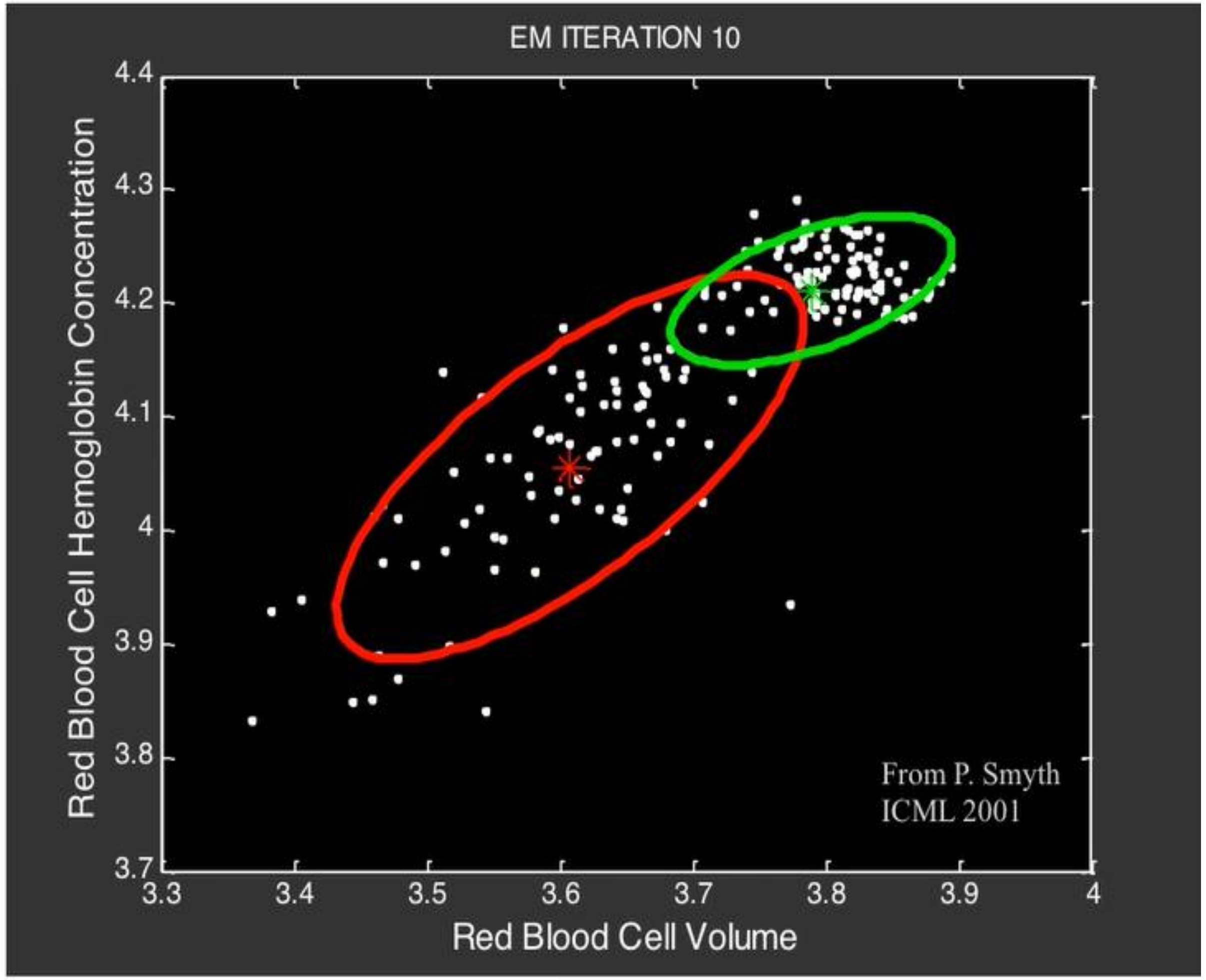
GMM : Example2



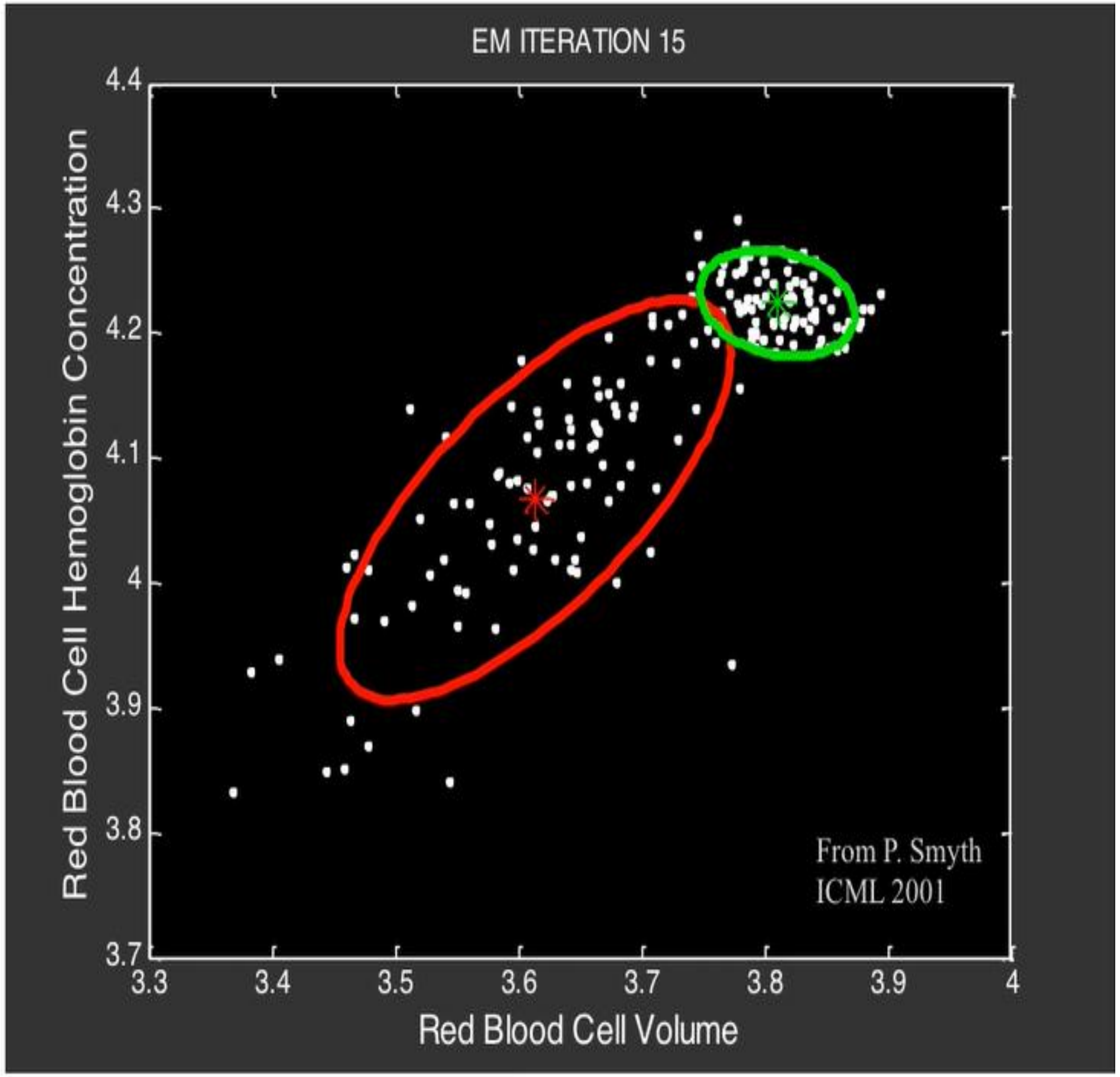


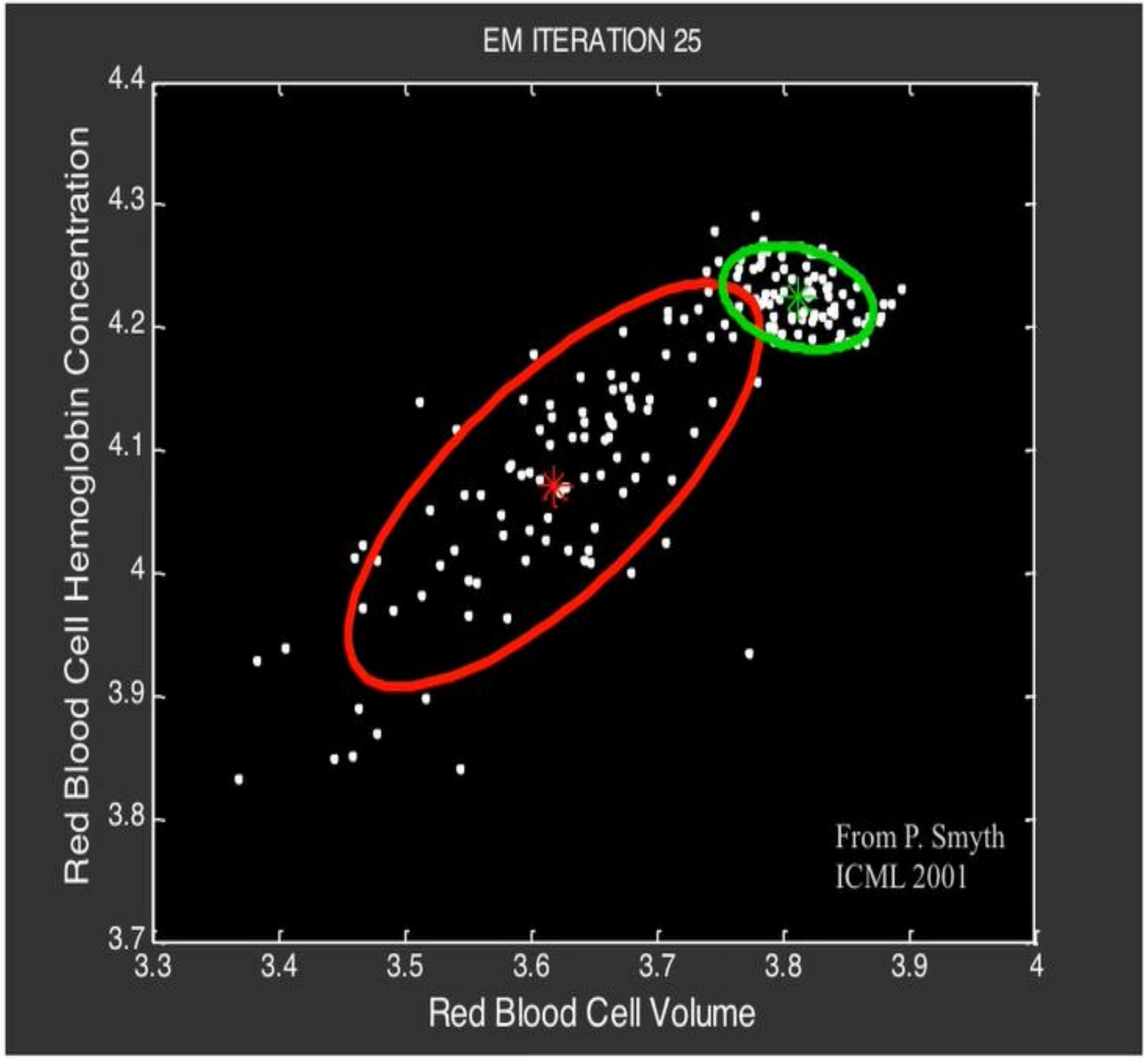
GMM : Example2



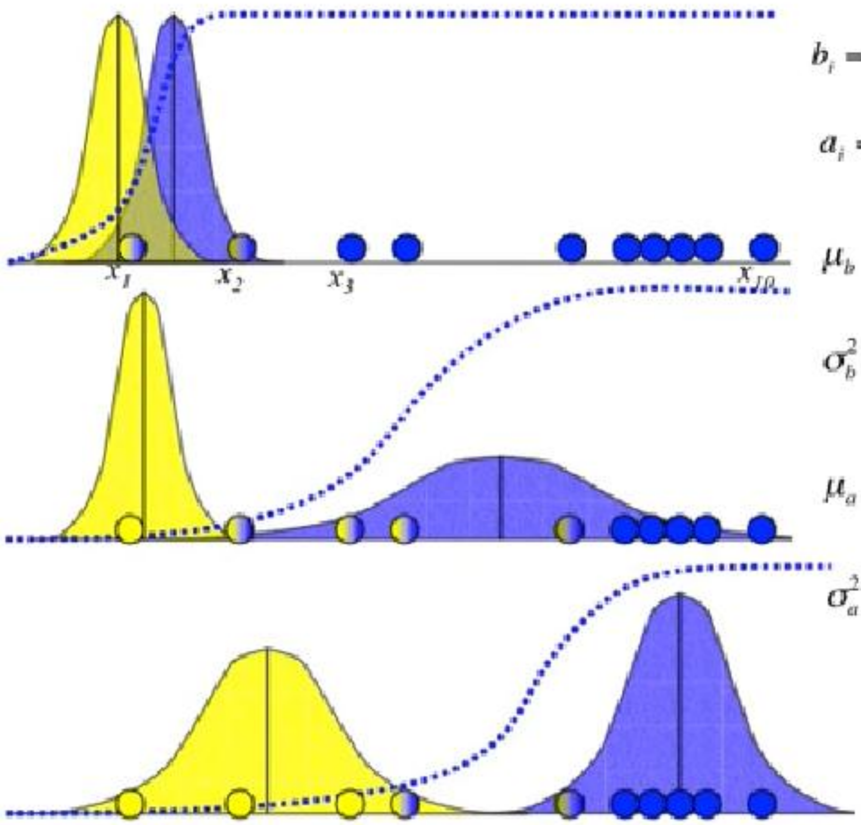


GMM : Example2





EM:



$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left\{-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right\}$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_n}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1(x_1 - \mu_b)^2 + \ldots + b_n(x_n - \mu_b)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \ldots + a_n x_n}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1(x_1 - \mu_a)^2 + \ldots + a_n(x_n - \mu_a)^2}{a_1 + a_2 + \ldots + a_n}$$

could also estimate priors:

$P(b) = (b_1 + b_2 + \ldots b_n) / n$

$P(a) = 1 - P(b)$

K Means Algorithm

- 1. The sample space is initially partitioned into K clusters and the observations are randomly assigned to the clusters.
- 2. For each sample:
  - Calculate the distance from the observation to the centroid of the cluster.
  - IF the sample is closest to its own cluster THEN leave it ELSE select another cluster.
- 3. Repeat steps 1 and 2 untill no observations are moved from one cluster to another

Distance functions

Euclidean
$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan
$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski
$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

Basic Algorithm of K-means

Algorithm 1 Basic K-means Algorithm.

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Details of K-means

1. Initial centroids are often chosen randomly.
   - *Clusters produced vary from one run to another*

2. The centroid is (typically) the mean of the points in the cluster.

3. 'Closeness' is measured by **Euclidean distance**, cosine similarity, correlation, etc.

4. K-means will converge for common similarity measures mentioned above.

5. Most of the convergence happens in the first few iterations.
   - *Often the stopping condition is changed to 'Until relatively few points change clusters'*

Euclidean Distance

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

A simple example: Find the distance between two points, the original and the point (3,4)

$$d_E(O,A)=\sqrt{3^2+4^2}=5$$

Update Centroid

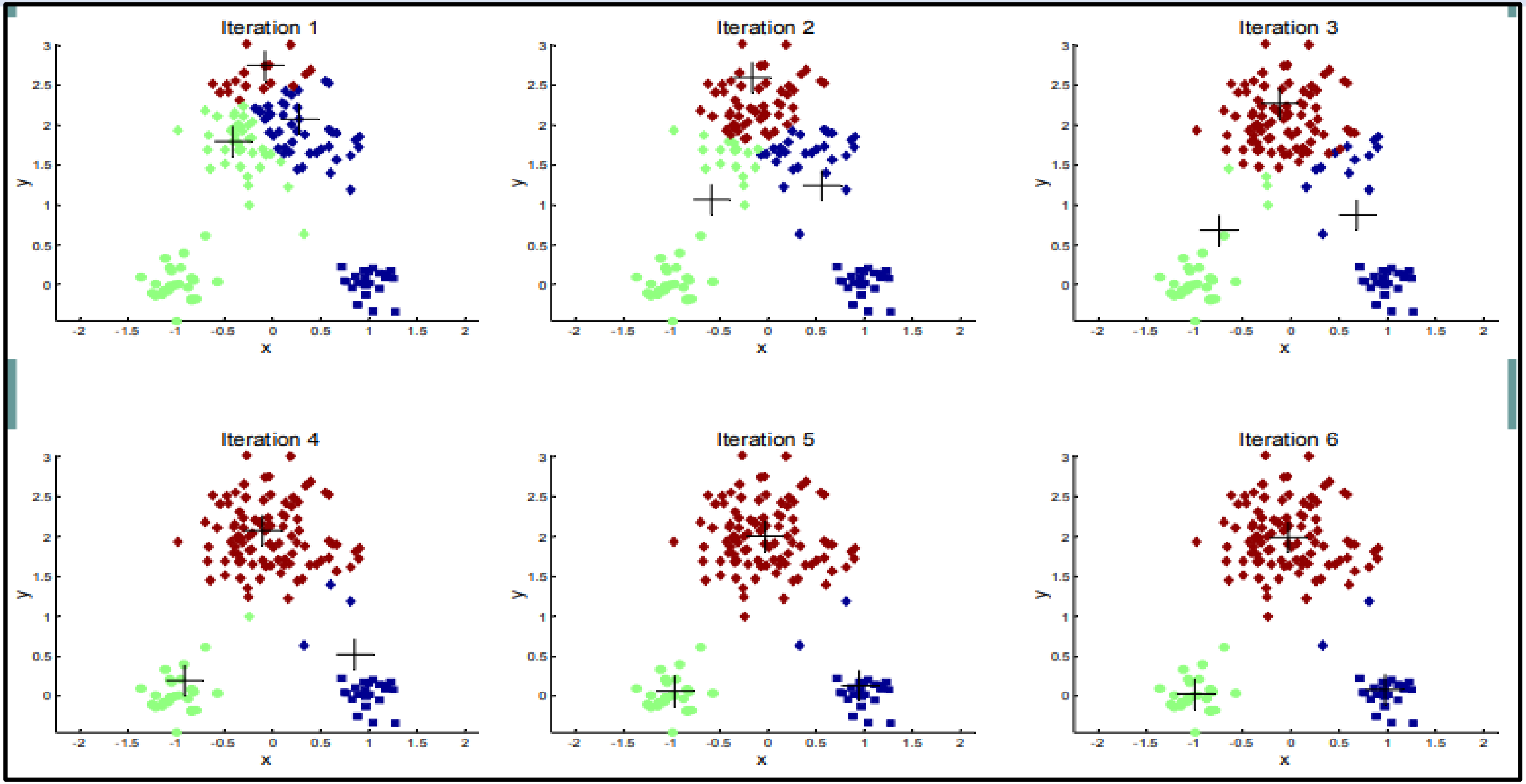
We use the following equation to calculate the n dimensional centroid point amid k n-dimensional points

$$CP(x_1, x_2, \ldots, x_k) = \left(\frac{\sum_{i=1}^{k} x1st_i}{k}, \frac{\sum_{i=1}^{k} x2nd_i}{k}, \ldots, \frac{\sum_{i=1}^{k} xnth_i}{k}\right)$$

Example: Find the centroid of 3 2D points, (2,4), (5,2) and (8,9)

$$CP = \left(\frac{2+5+8}{3}, \frac{4+2+9}{3}\right) = (5,5)$$

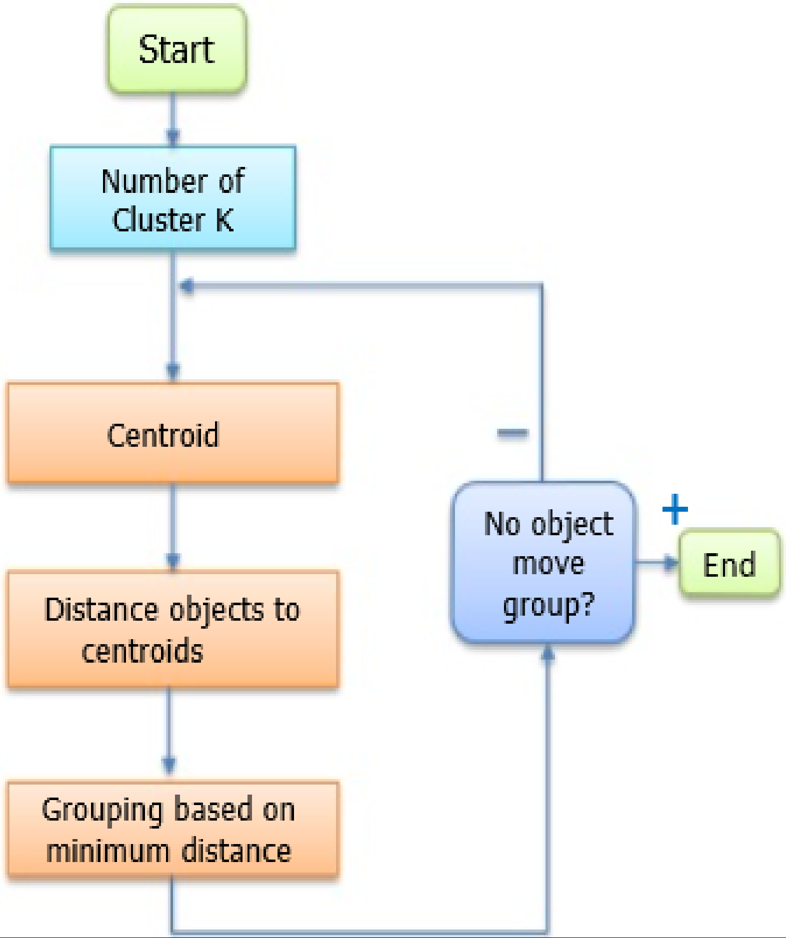
Examples of K Means



How the K-Mean Clustering algorithm works?

$$\boxed{\left[\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}\right]}$$

Process Flow of K-means



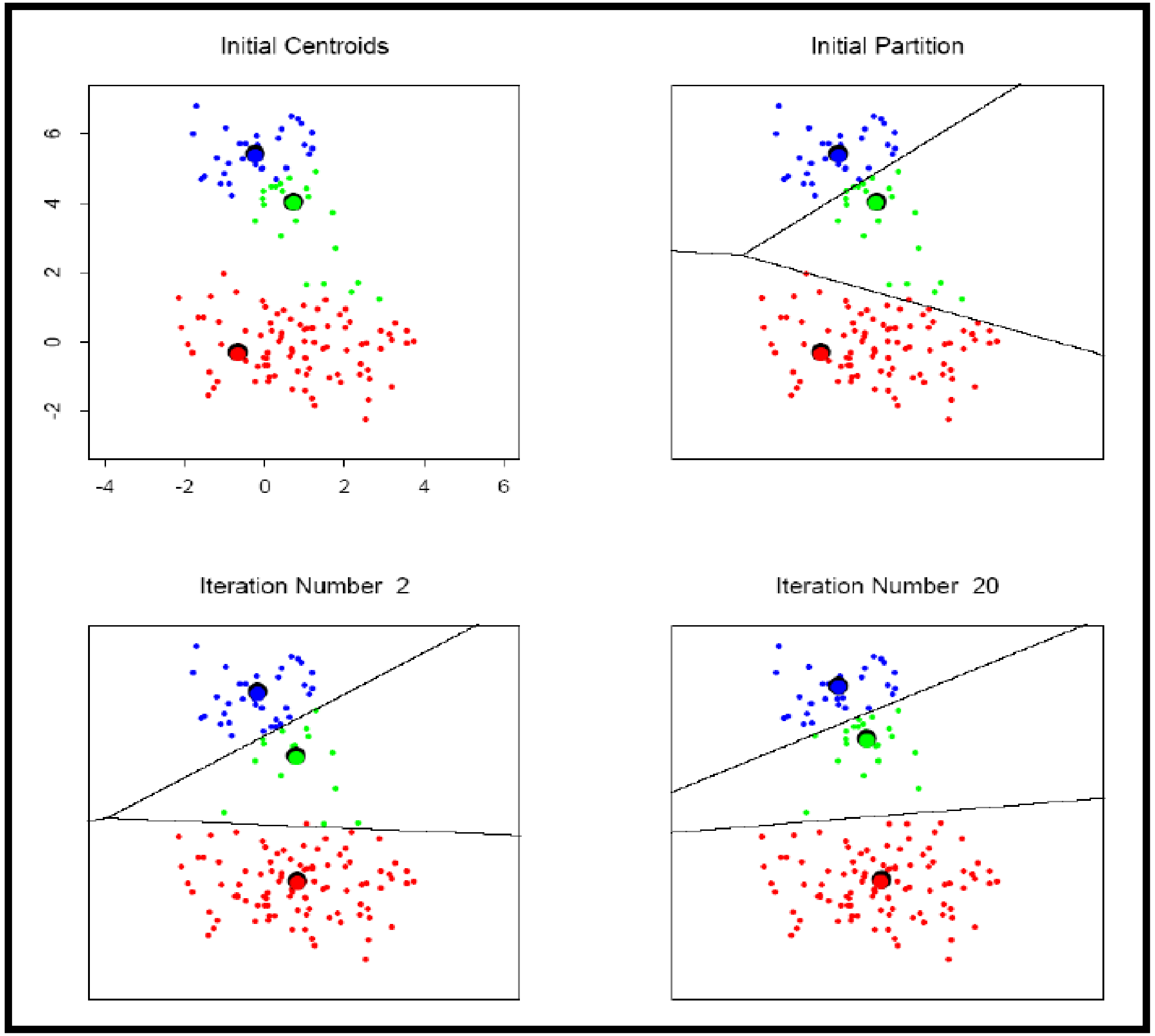
Iterate until *stable* (cluster centers converge):

1. Determine the centroid coordinate.

2. Determine the distance of each object to the centroids.

3. Group the object based on minimum distance (find the closest centroid)

K-means clustering example



Source Code

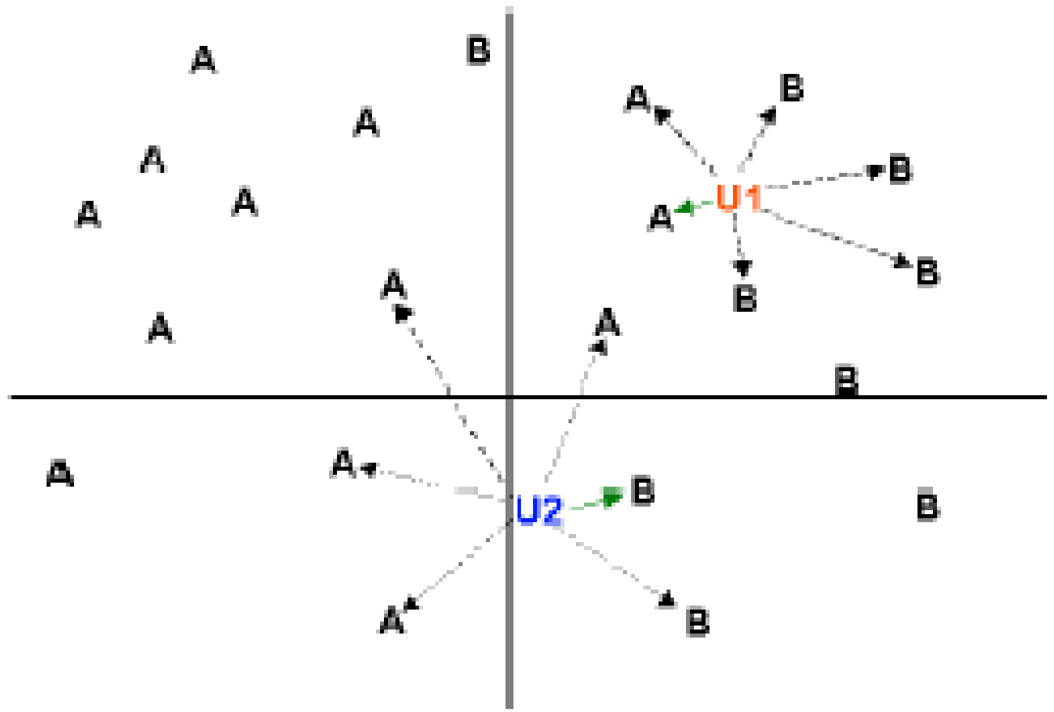
•

Lab Program 7

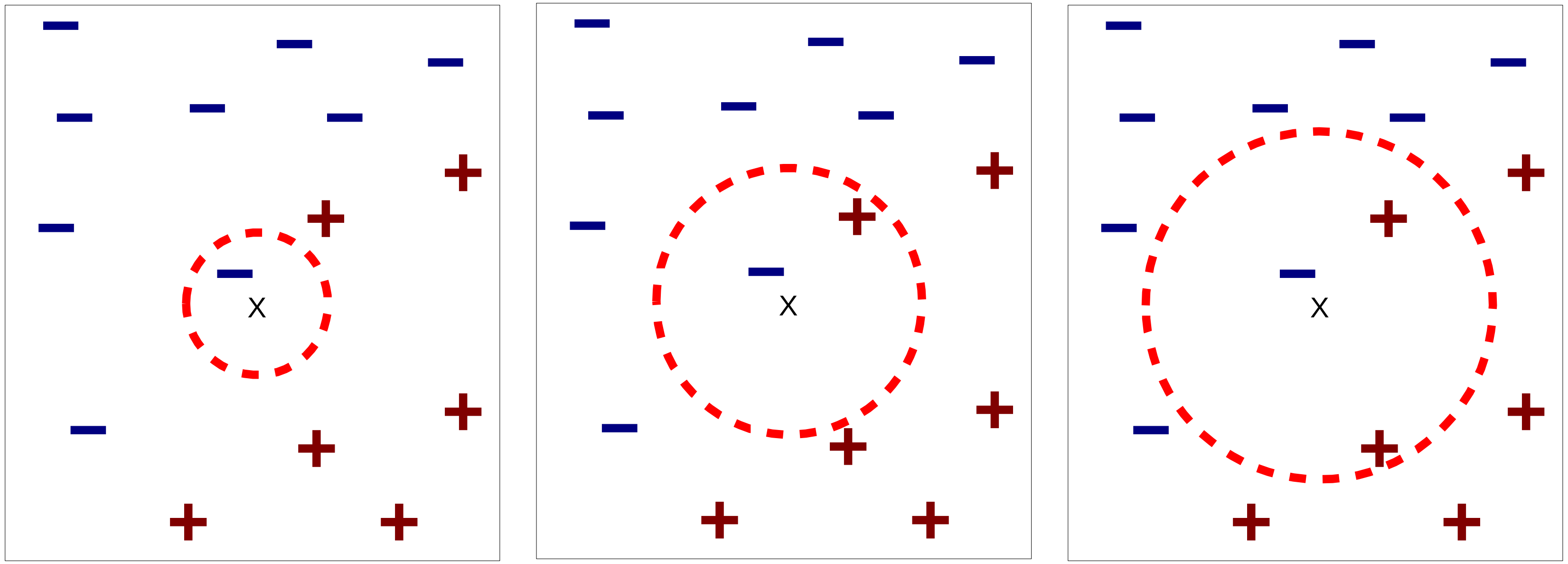
- Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Python ML library classes can be used for this problem.

K-Nearest-Neighbor Algorithm

- **Principle:** points (documents) that are close in the space belong to the same class



Definition of Nearest Neighbor



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

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

Context Innovations Lab

Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

Distance Metrics

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T Q(\boldsymbol{x}-\boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x}-\boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

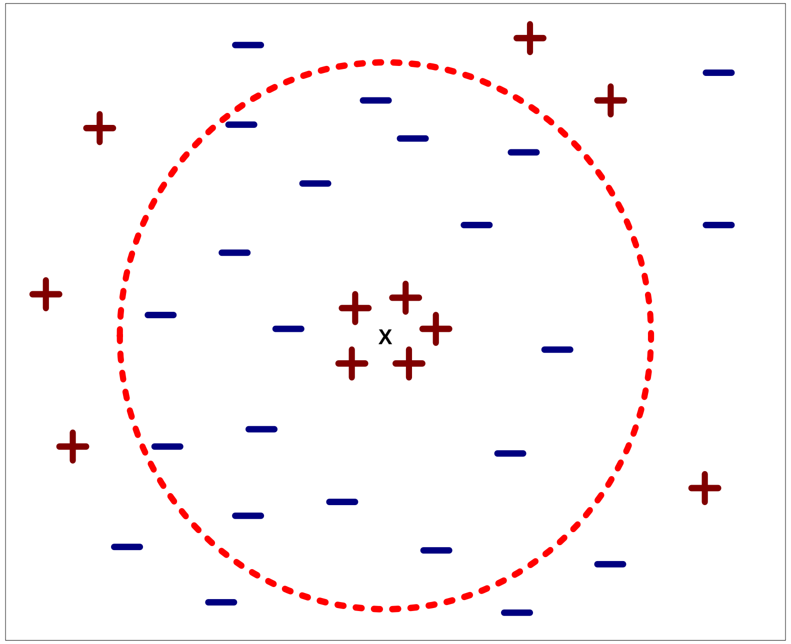
Figure 1. Equations of selected distance functions. ($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Selection of Distance Metrics

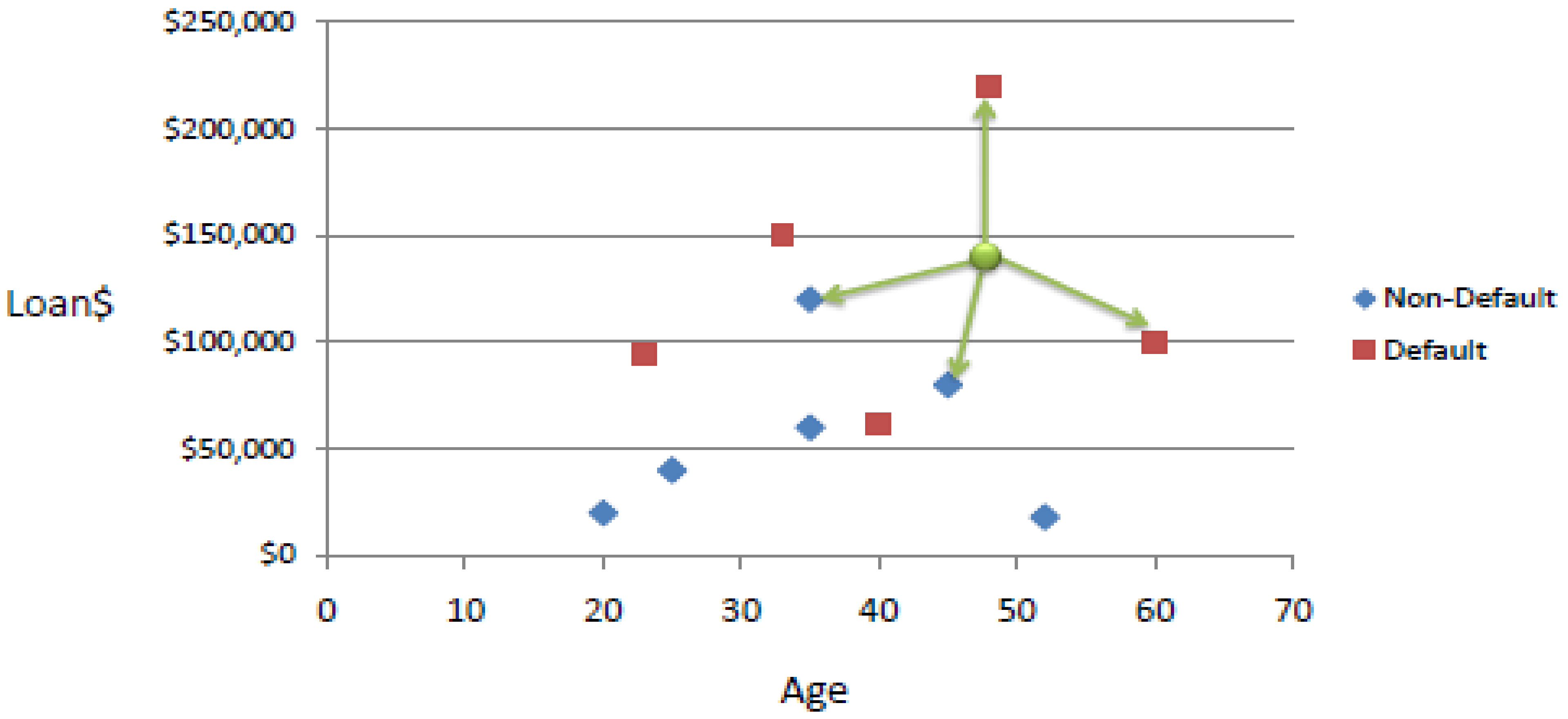
- You can choose the best distance metric based on the properties of your data. If you are unsure, you can experiment with different distance metrics and different values of K together and see which mix results in the most accurate models.
- Euclidean is a good distance measure to use if the input variables are similar in type (e.g. all measured widths and heights).
- Manhattan distance is a good measure to use if the input variables are not similar in type (such as age, gender, height, etc.).

Nearest Neighbor Classification…

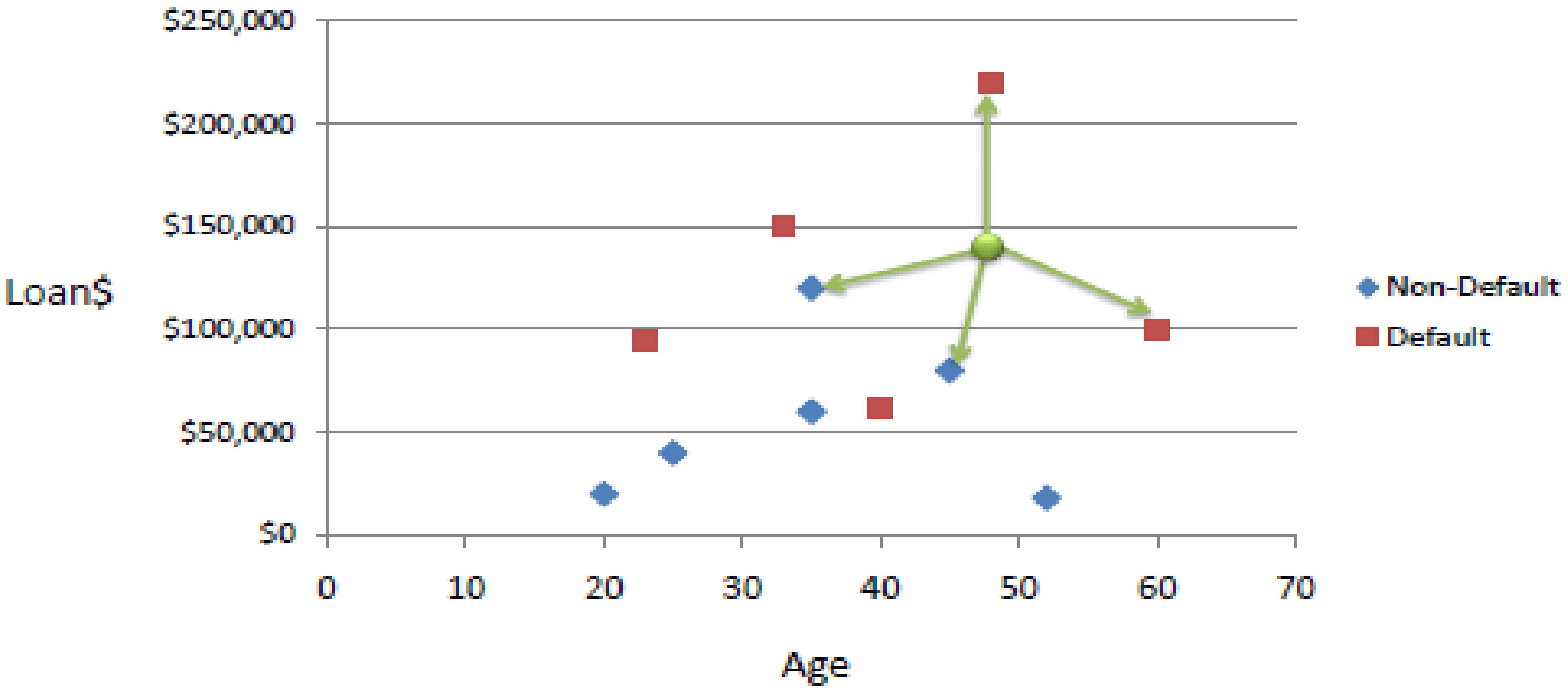
- **Choosing the value of k:**
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Example: Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



Example: We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.



D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	$142,000	?		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Source Code

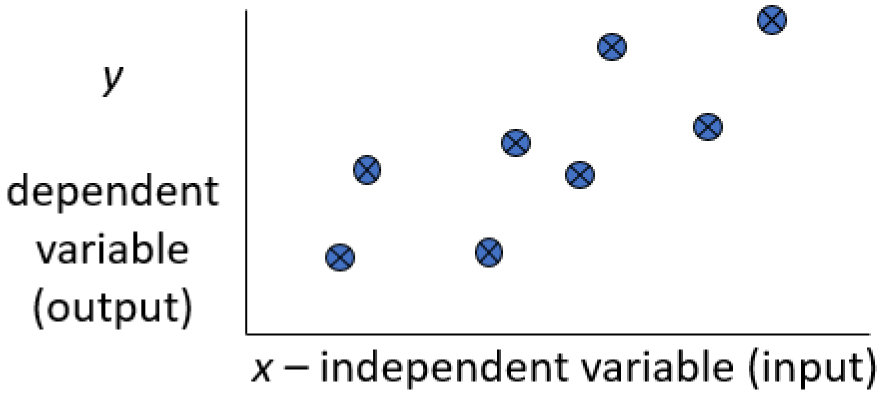
- https://github.com/profthyagu

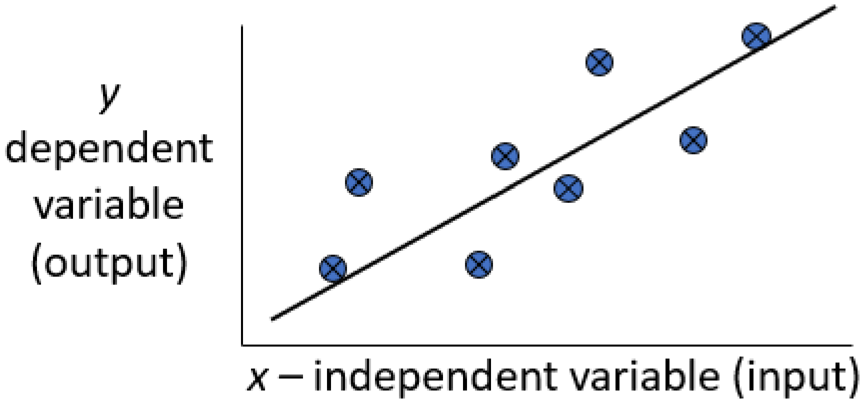
Lab Program 8

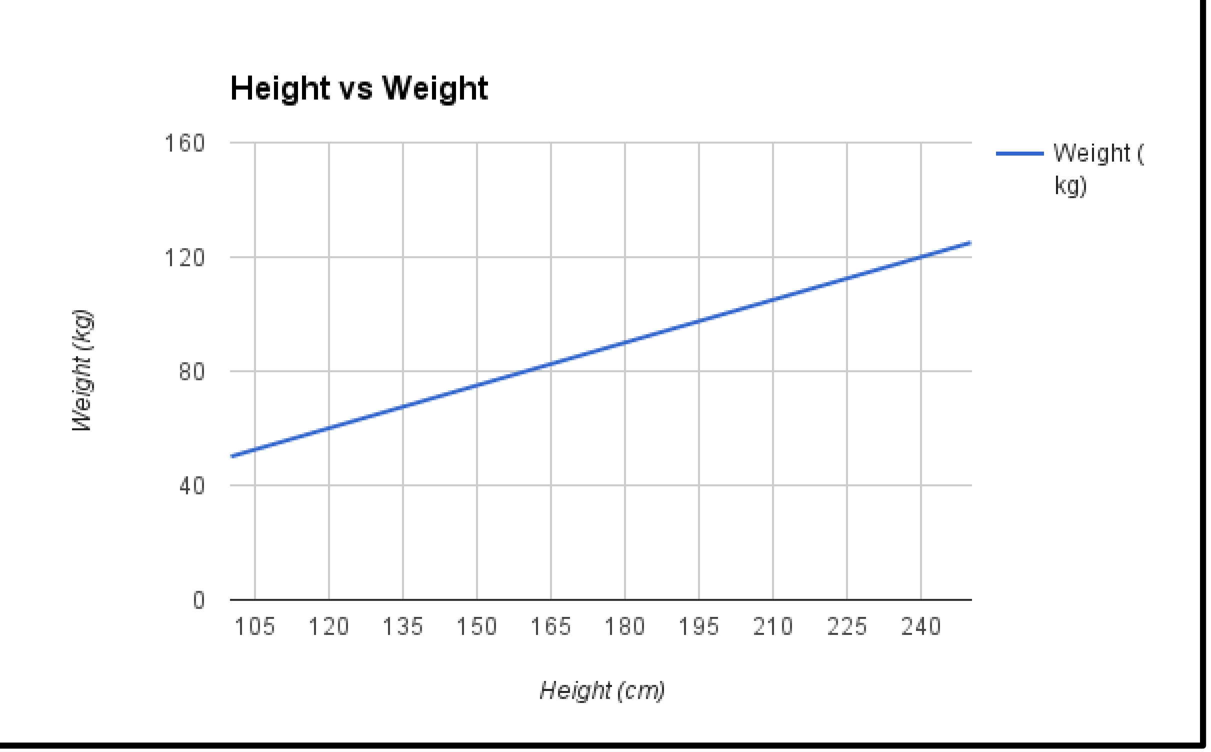
- **Implement the non-parametric Locally Weighted Regression (LOWESS) algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.**

Regression

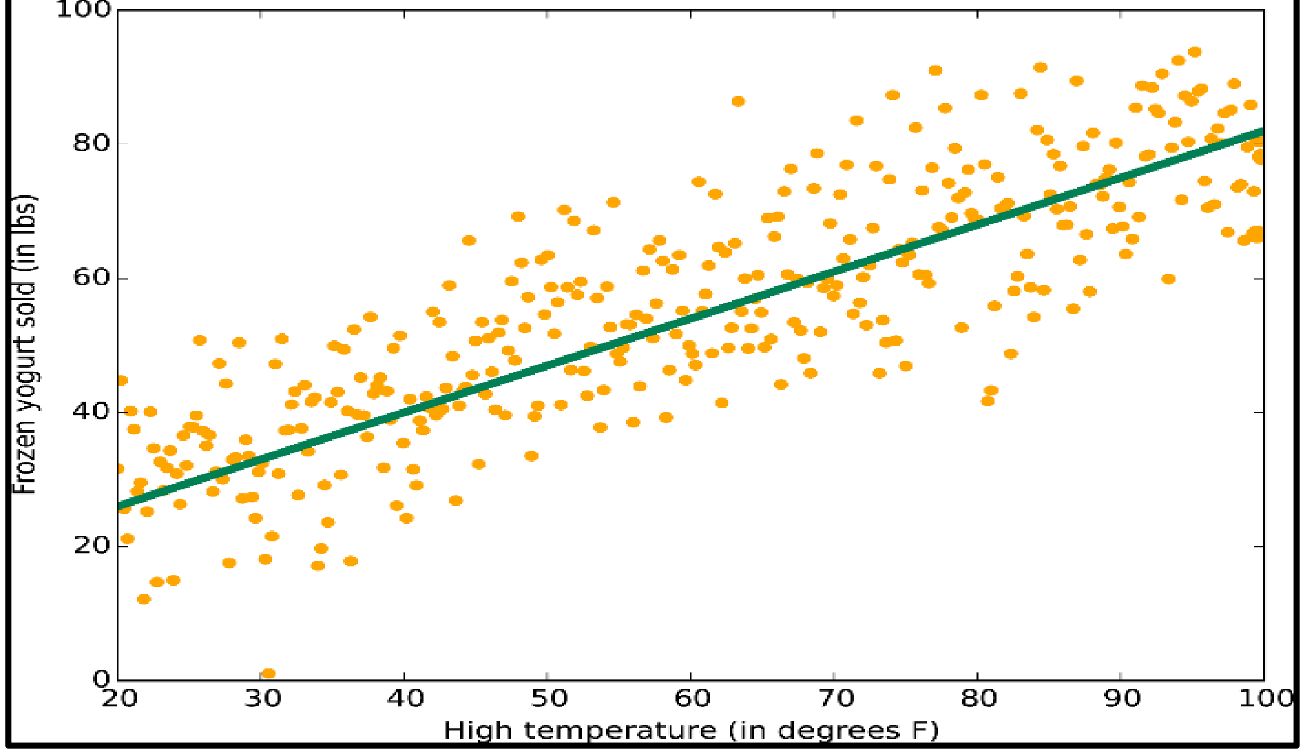
- Regression is a technique from statistics that is used to predict values of a desired target quantity when the target quantity is continuous .
- In regression, we seek to identify (or estimate) a continuous variable y associated with a given input vector x.
  - y is called the dependent variable.
  - x is called the independent variable.



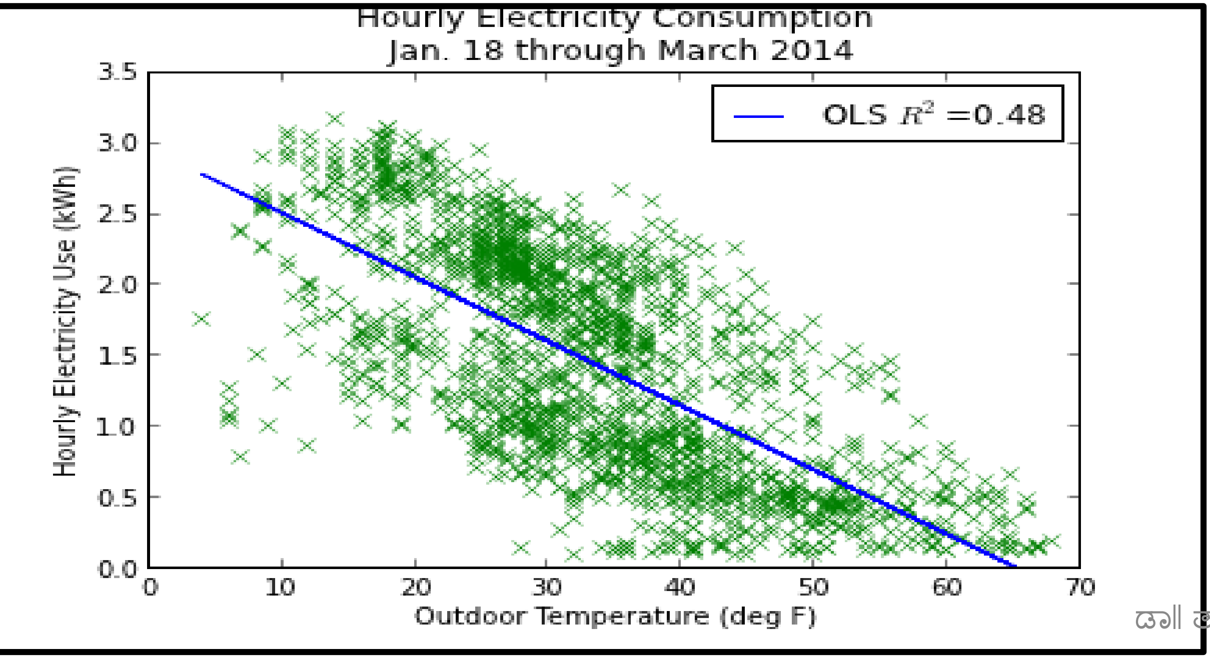




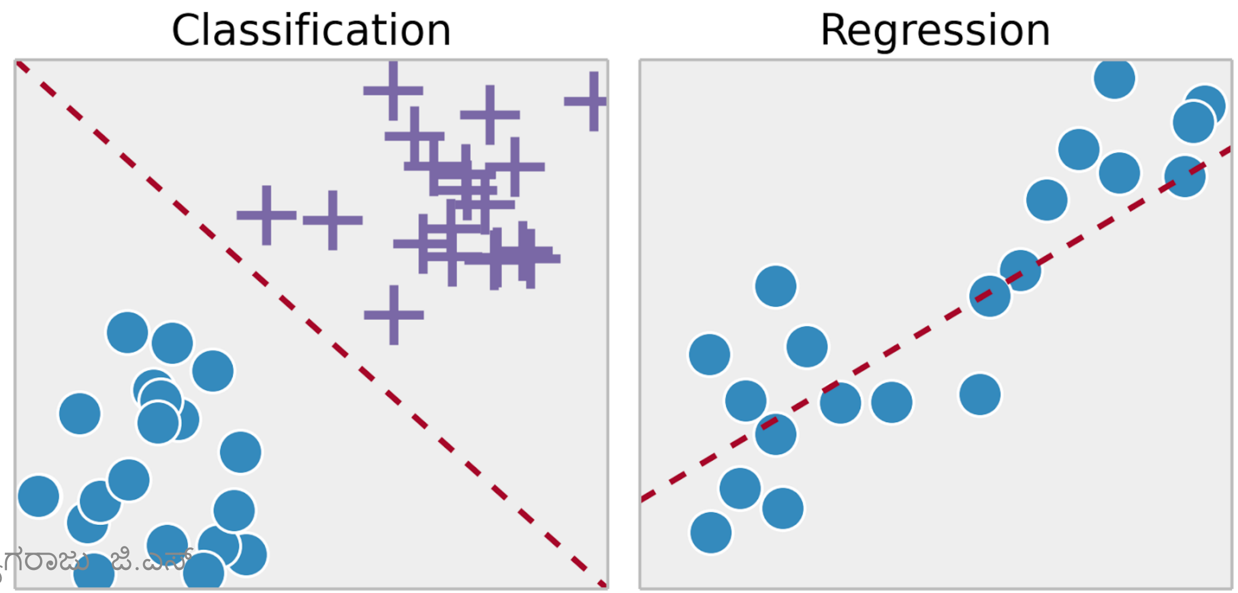
Height vs Weight
Weight (kg)
Weight (kg)
0
40
80
120
160
105
120
135
150
165
180
195
210
225
240
Height (cm)



Frozen yogurt sold (in lbs)
0
20
40
60
80
100
20
30
40
50
60
70
80
90
100
High temperature (in degrees F)

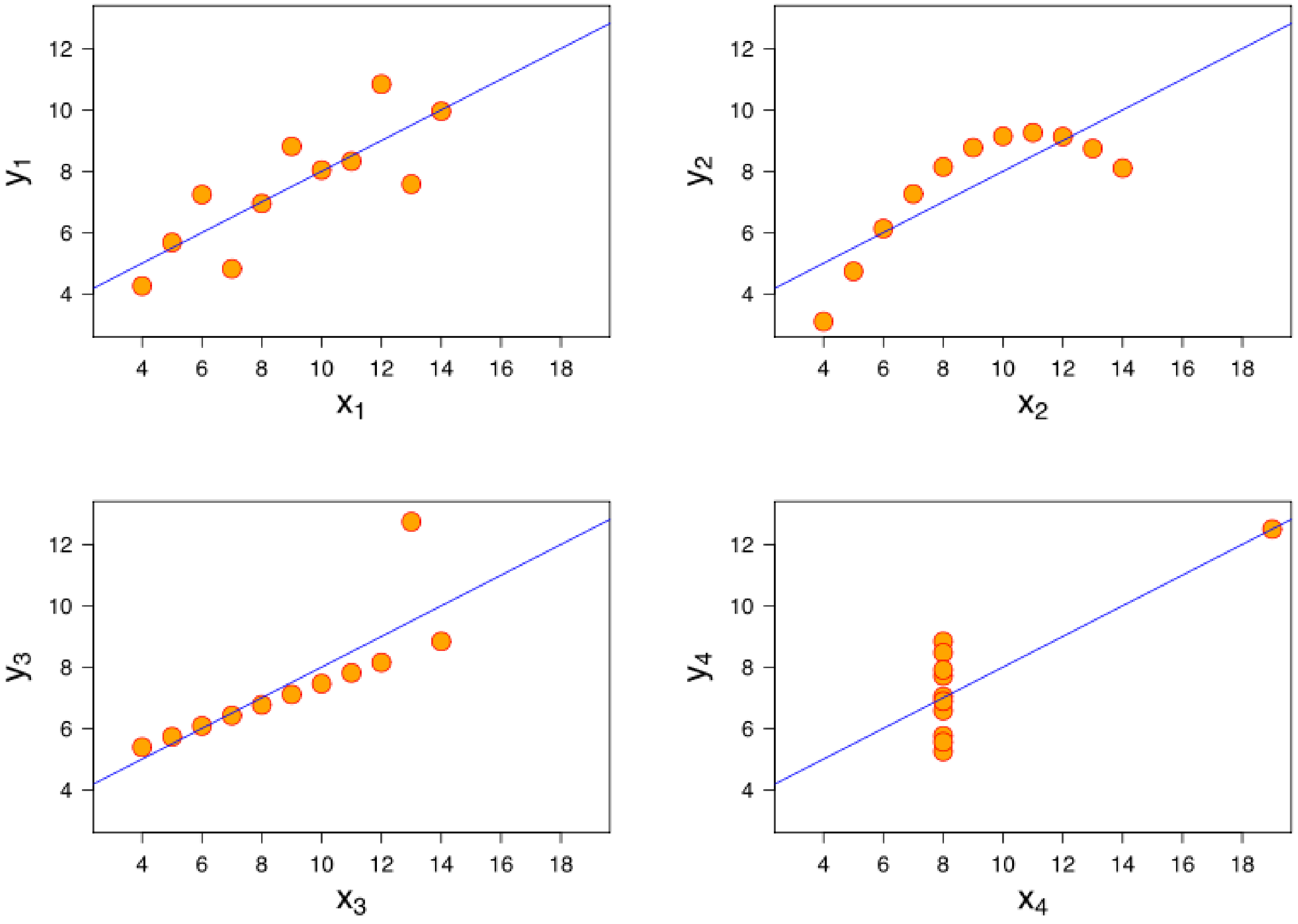


Hourly Electricity Consumption
Jan. 18 through March 2014
OLS R^2 =0.48
Hourly Electricity Use (kWh)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
10
20
30
40
50
60
70
Outdoor Temperature (deg F)



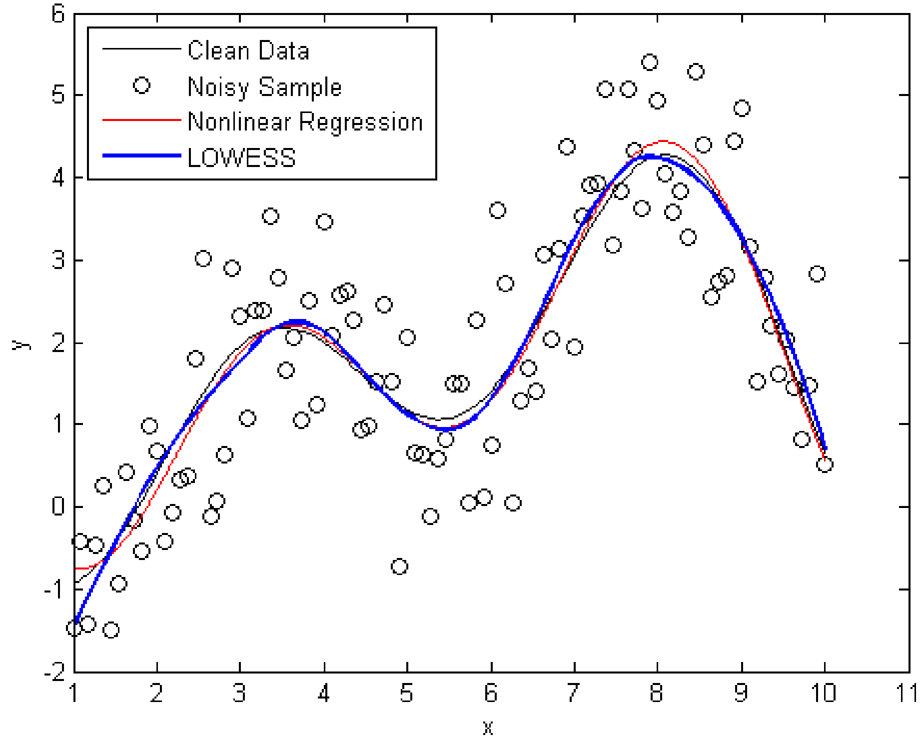
Classification
Regression

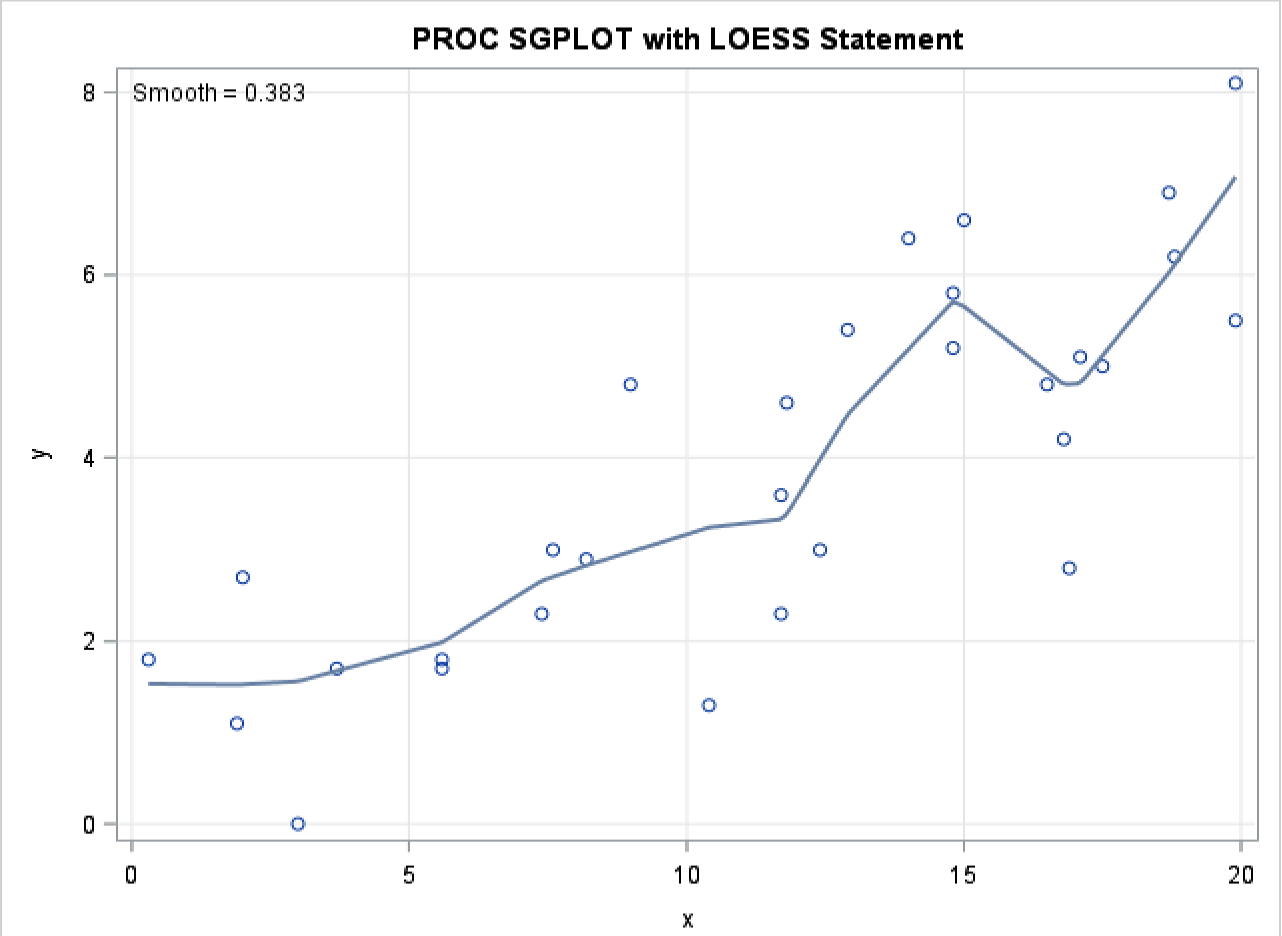
What lines "really" best fit each case?



Loess/Lowess Regression

- Loess regression is a nonparametric technique that uses ***local weighted*** regression to fit a **smooth curve** through points in a scatter plot.





Lowess Algorithm

- Locally weighted regression is a very powerful non-parametric model used in statistical learning .Given a *dataset* **X**, **y**, we attempt to find a *model* parameter β**(x)** that minimizes *residual sum of* ***weighted*** *squared errors*. The weights are given by a *kernel function(k or w)* which can be chosen arbitrarily .

Algorithm

1. Read the Given data Sample to **X** and the curve (linear or non linear) to **Y**
2. Set the value for Smoothening parameter or Free parameter say **τ**
3. Set the bias /Point of interest set **X0** which is a subset of **X**
4. Determine the weight matrix using :

$$w(x, x_o) = e^{-\frac{(x-x_o)^2}{2\tau^2}}$$

5. Determine the value of model term parameter β using :

$$\hat{\beta}(x_o) = (X^T W X)^{-1} X^T W y$$

6. Prediction = $X_0 * \beta$

Source Code

https://github.com/profthyagu

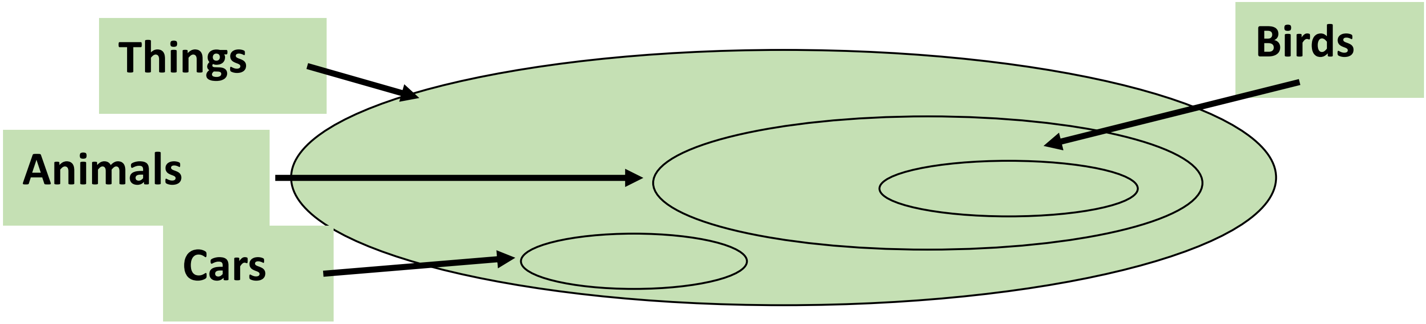
Lab Program 9

- **Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.**

Day	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

What is a Concept?

- A **Concept** is a subset of objects or events defined over a larger set [Example: The concept of a *bird* is the subset of all objects (i.e., the set of all *things* or all *animals*) that belong to the category of bird.]



- Alternatively, a concept is a boolean-valued function defined over this larger set [**Example:** a function defined over all *animals* whose value is *true* for birds and *false* for every other animal].

What is Concept-Learning?

- Given a set of examples labeled as members or non-members of a concept, *concept-learning* consists of automatically inferring the general definition of this concept.
- In other words, *concept-learning* consists of approximating a boolean-valued function from training examples of its input and output.

A CONCEPT LEARNING TASK

- Consider the example task of learning the target concept "*days on which my friend Aldo enjoys his favorite water sport*.“
- Table describes a set of example days, each represented by a set of attributes.
- The attribute EnjoySport indicates whether or not Aldo enjoys his favorite water sport on this day.
- The task is to learn to predict the value of EnjoySport for an arbitrary day, based on the values of its other attributes.

Database:

Day	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	YEs

Example of a Concept Learning

- **Concept:** Good Days for Water Sports (values: Yes, No)
- **Attributes/Features:**
  - **Sky** (values: Sunny, Cloudy, Rainy)
  - **AirTemp** (values: Warm, Cold)
  - **Humidity** (values: Normal, High)
  - **Wind** (values: Strong, Weak)
  - **Water** (Warm, Cool)
  - **Forecast** (values: Same, Change)
- **Example of a Training Point:**

class

<Sunny, Warm, High, Strong, Warm, Same, Yes>

Concept Learning: Search in Hypothesis Space

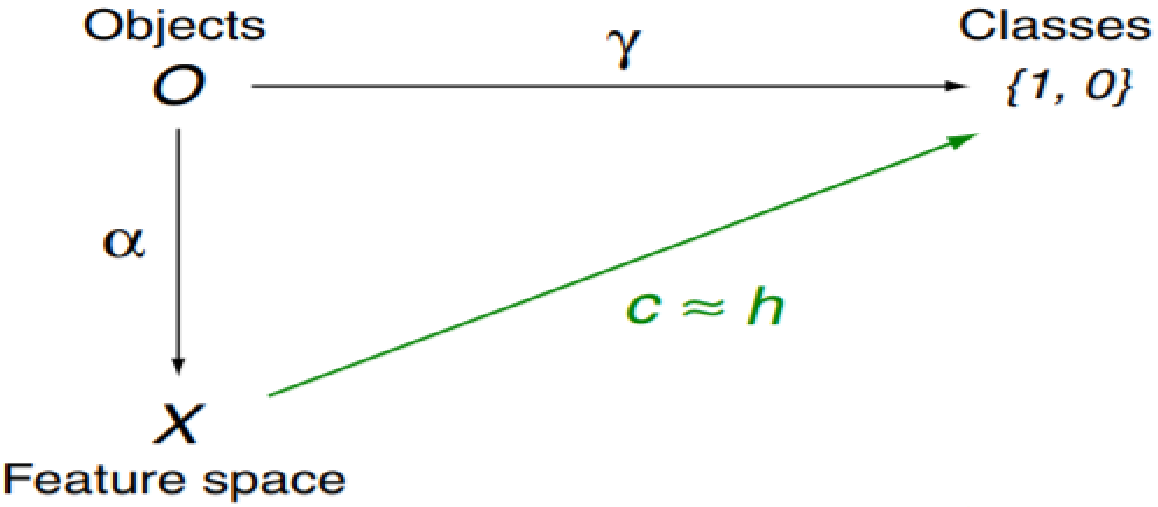
A Learning Task

Given is a set D of examples: days that are characterized by the six features “Sky”, “Temperature”, “Humidity”, “Wind”, “Water”, and “Forecast”, along with a statement (in fact: a feature) whether or not our friend will enjoy her favorite sport.

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
4	sunny	warm	high	strong	cool	change	yes

- What is the concept behind “EnjoySport” ?
- What are possible hypotheses to formalize the concept “EnjoySport” ?
  Similarly: What are the elements of the set or class “EnjoySport” ?

Concept Learning: Search in Hypothesis Space



Definition 1 (Concept, Hypothesis, Hypothesis Space)

A concept is a subset of an object set O and hence determines a subset of the feature space $X = \alpha(O)$. Concept learning is the approximation of the ideal classifier $c : X \to \{0, 1\}$ by a function h, where c is defined as follows:

$$c(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha^{-1}(\mathbf{x}) \text{ belongs to the concept} \\ 0 & \text{otherwise} \end{cases}$$

The approximation function $h : X \to \{0, 1\}$ is called hypothesis here. A set H of hypotheses among which h is searched is called hypothesis space.

Concept Learning: Search in Hypothesis Space

Usually, an example set D, $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\}$, contains positive ($c(\mathbf{x}) = 1$) and negative ($c(\mathbf{x}) = 0$) examples.

Definition 2 (Hypothesis-Fulfilling, Consistency)

An example $(\mathbf{x}, c(\mathbf{x}))$ fulfills a hypothesis h iff $h(\mathbf{x}) = 1$. A hypothesis h is consistent with an example $(\mathbf{x}, c(\mathbf{x}))$ iff $h(\mathbf{x}) = c(\mathbf{x})$.

A hypothesis h is consistent with a set D of examples, denoted as *consistent*(h, D), iff:

$$\forall (\mathbf{x}, c(\mathbf{x})) \in D : \; h(\mathbf{x}) = c(\mathbf{x})$$

Terminology and Notation

Chosen Hypothesis Representation:

Conjunction of constraints on each attribute where:

- “?” means “any value is acceptable”
- “0” means “no value is acceptable”

- **Example of a hypothesis: <?, Cold, High, ?, ?, ?>** (If the air temperature is cold and the humidity high then it is a good day for water sports)
- **Goal:** To infer the “best” concept-description from the set of all possible hypotheses (“best” means “which best generalizes to all (known or unknown) elements of the instance space”)
- **The most general hypothesis-that every day is a good day for water sports, positive example-is represented by <?, ?, ?, ?, ?, ?> and**
- **The most specific possible hypothesis-that no day is a positive example-is represented by <0,0,0,0,0,0>**

Terminology and Notation

- The set of items over which the concept is defined is called the set of ***instances*** (denoted by X)
- The concept to be learned is called the *Target Concept* (denoted by c: X--> {0,1})
- The set of ***Training Examples D*** is a set of instances, x, along with their target concept value c(x).
- Members of the concept (instances for which c(x)=1) are called ***positive examples***.
- Non members of the concept (instances for which c(x)=0) are called ***negative examples***.
- **H** represents the set of ***all possible hypotheses***. **H** is determined by the human designer's choice of a hypothesis representation.
- **The goal of concept-learning is to find a hypothesis h:X --> {0,1} such that h(x)=c(x) for all x in X.**

Structure of a hypothesis h:

1. conjunction of feature-value pairs
2. three kinds of values: literal, ? (wildcard), **0** (contradiction)

A hypothesis for EnjoySport $\Longrightarrow$ $\langle$ *sunny, ?, ?, strong, ?, same* $\rangle$

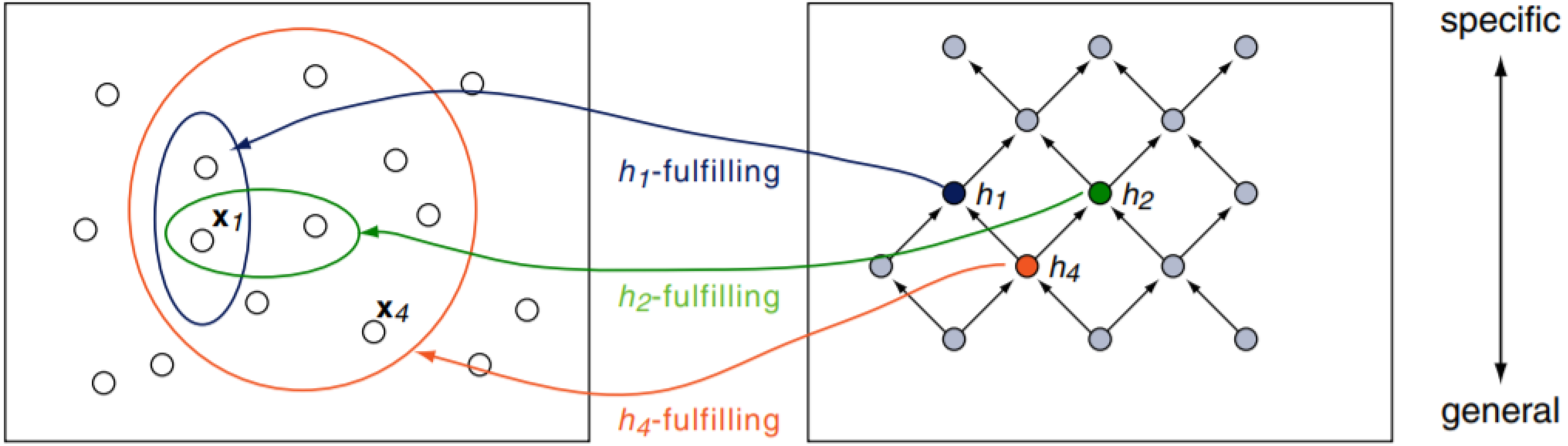
Definition 3 (Maximally Specific / General Hypothesis)

The hypotheses $s_0(\mathbf{x}) \equiv 0$ and $g_0(\mathbf{x}) \equiv 1$ are called maximally specific and maximally general hypothesis respectively. No $\mathbf{x} \in X$ fulfills s_0, and all $\mathbf{x} \in X$ fulfill g_0.

Maximally specific / general hypothesis in the example :

- $s_0 = \langle \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0} \rangle$ (never enjoy sport)
- $g_0 = \langle ?, ?, ?, ?, ?, ? \rangle$ (always enjoy sport)

Order of Hypotheses



$\mathbf{x}_1 =$ (*sunny, warm, normal, strong, warm, same*) $h_1 = \langle$ *sunny, ?, normal, ?, ?, ?* $\rangle$

$h_2 = \langle$ *sunny, ?, ?, ?, warm, ?* $\rangle$

$\mathbf{x}_4 =$ (*sunny, warm, high, strong, cool, change*) $h_4 = \langle$ *sunny, ?, ?, ?, ?, ?* $\rangle$

Definition 4 **(More General Relation)**

Let X be a feature space and let h_1 and h_2 be two boolean-valued functions with domain X. Then function h_1 is called more general than function h_2, denoted as $h_1 \geq_g h_2$, iff:

$$\forall \mathbf{x} \in X : \ (\ h_2(\mathbf{x}) = 1 \ \text{ implies } \ h_1(\mathbf{x}) = 1 \ )$$

h_1 is called stricly more general than h_2, denoted as $h_1 >_g h_2$, iff:

$$(h_1 \geq_g h_2) \ \text{ and } \ (h_2 \not\geq_g h_1)$$

About the maximally specific / general hypothesis:

- s_0 is minimum and g_0 is maximum with regard to $\geq_g$: no hypothesis is more specific wrt. s_0, and no hypothesis is more general wrt. g_0.
- We will consider only hypothesis spaces that contain s_0 and g_0.

Algorithm : Find-S, a Maximally Specific Hypothesis

1. Initialize **h** to the most specific hypothesis in **H**
2. **For** each positive training instance **x**
   - **For** each attribute constraint $\mathbf{a_i}$ in **h**

     If the constraint $\mathbf{a_i}$ in **h** is satisfied by **x** then do nothing

     else replace $\mathbf{a_i}$ in h by the next more general constraint that is satisfied by **x**
3. Output hypothesis **h**

Step1: Find S

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

1. Initialize h to the most specific hypothesis in H

h0 = **<Ø, Ø, Ø, Ø, Ø, Ø>**

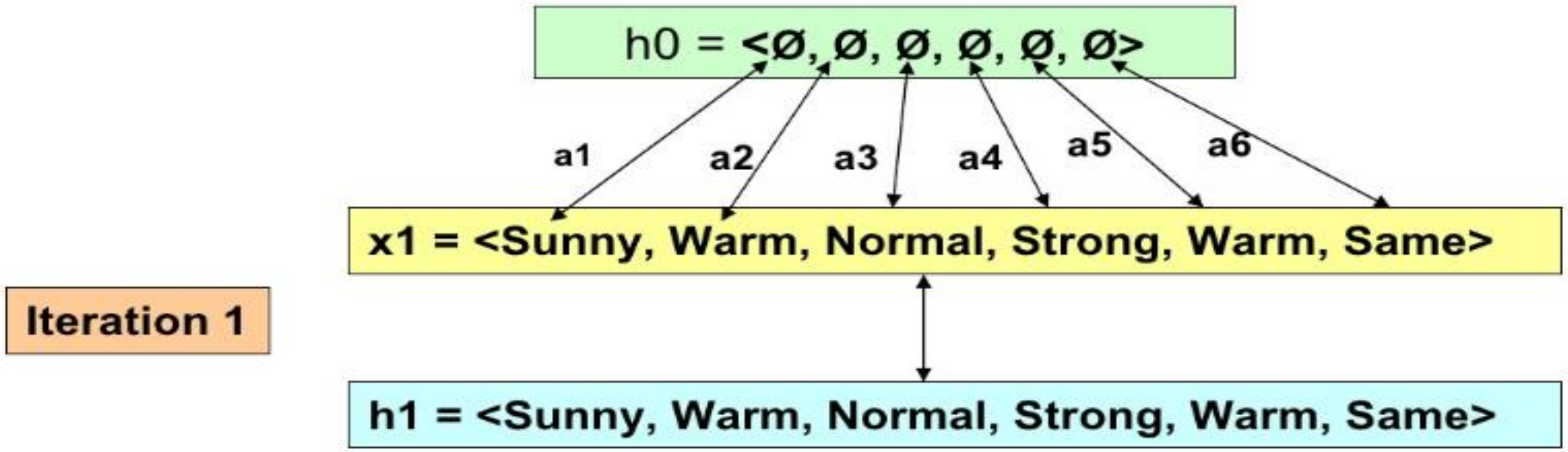
Step2 : Find S

2. For each positive training instance x
   - For each attribute constraint a_i in h

     If the constraint a_i is satisfied by x

     Then do nothing

     Else replace a_i in h by the next more general constraint that is satisfied by x



Step2 : Find S

Iteration 2

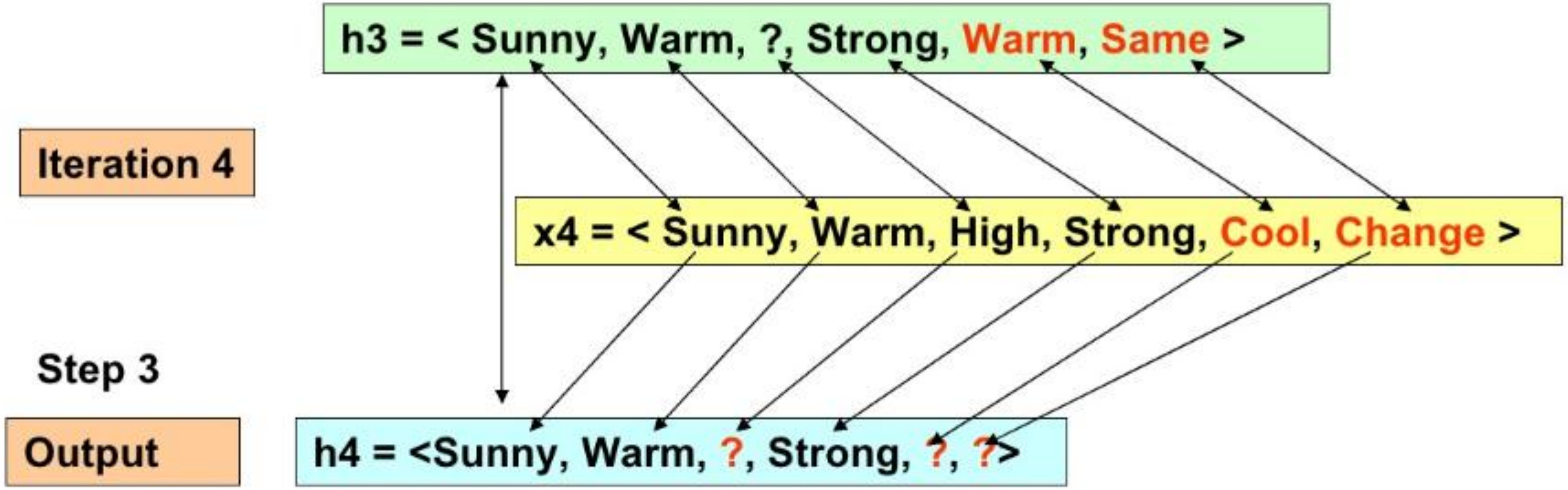
h1 = <Sunny, Warm, Normal, Strong, Warm, Same>

x2 = <Sunny, Warm, High, Strong, Warm, Same>

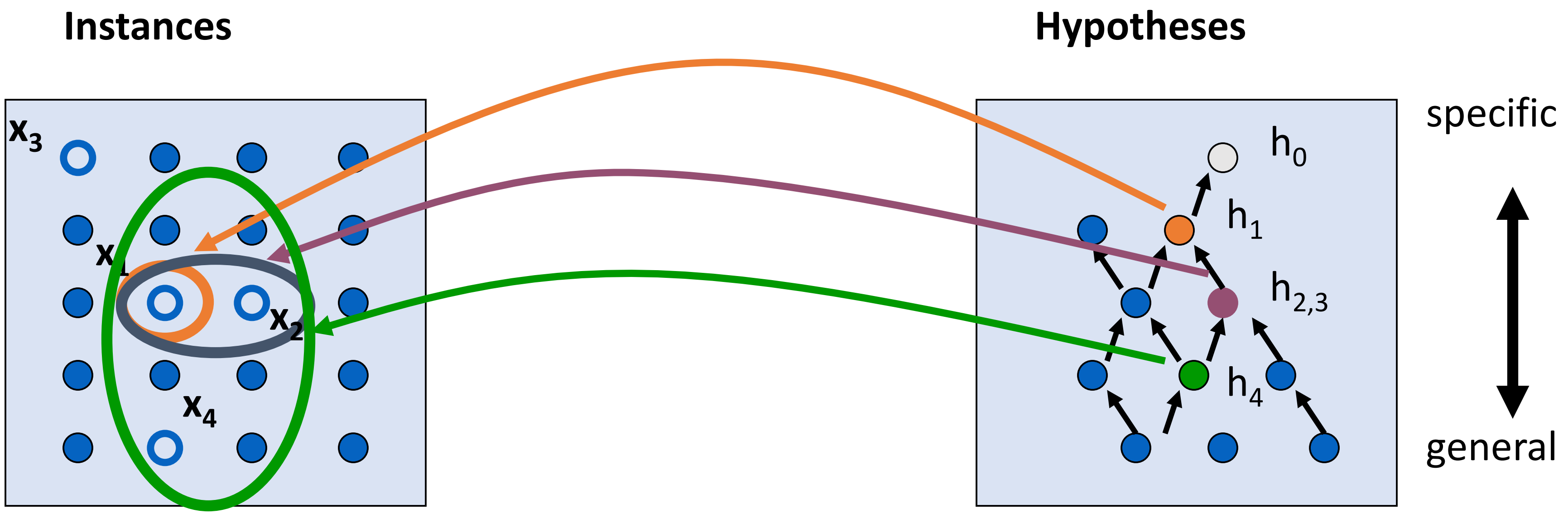
h2 = <Sunny, Warm, ?, Strong, Warm, Same>

Iteration 3 | Ignore | h3 = <Sunny, Warm, ?, Strong, Warm, Same>

Iteration 4 and Step 3 : Find S



Hypothesis Space Search by Find-S



x_1=<Sunny,Warm,Normal,Strong,Warm,Same>+

x_2=<Sunny,Warm,High,Strong,Warm,Same> +

x_3=<Rainy,Cold,High,Strong,Warm,Change> -

x_4=<Sunny,Warm,High,Strong,Cool,Change> +

h_0 = < Ø, Ø, Ø, Ø, Ø, Ø,>

h_1=< Sunny,Warm,Normal,Strong,Warm,Same>

$h_{2,3}$=< Sunny,Warm,?,Strong,Warm,Same>

h_4=< Sunny,Warm,?, Strong,?,?>

Source Code

https://github.com/profthyagu

Lab Program 10

For a given set of training data examples stored **in** a .CSV file, implement **and** demonstrate the Candid ate - Elimination algorithm to output a description of the set of all hypotheses consistent **with** the trai ning examples

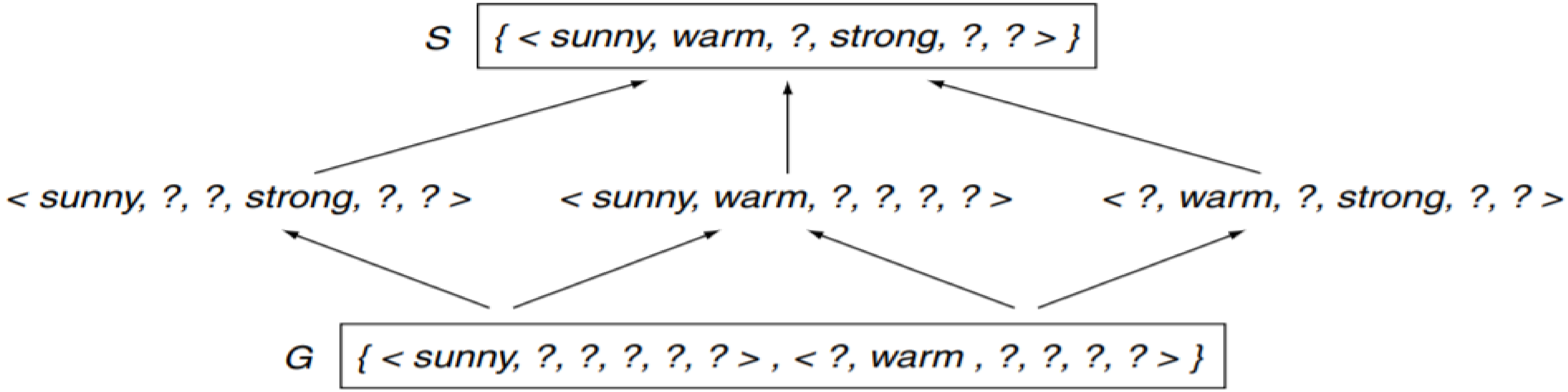
Version Spaces

Definition 5 (Version Space)

The version space $V_{H,D}$ of an hypothesis space H and a example set D is comprised of all hypotheses $h \in H$ that are consistent with a set D of examples:

$$V_{H,D} = \{h \mid h \in H \;\wedge\; (\; \forall(\mathbf{x}, c(\mathbf{x})) \in D: \;\; h(\mathbf{x}) = c(\mathbf{x}) \;)\;\}$$

Illustration of $V_{H,D}$ for the example set D:



Representing Version Spaces

- The **general boundary**, G, of version space $VS_{H,D}$ is the set of maximally general members.
- The **specific boundary**, S, of version space $VS_{H,D}$ is the set of maximally specific members.
- Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

where $x \geq y$ means x is more general or equal than y

Definition 6 (Boundary Sets of a Version Space)

Let H be hypothesis space and let D be set of examples. Then, based on the $\geq_g$-relation, the set of maximally general hypotheses, G, is defined as follows:

$$\{g \mid g \in H \ \wedge \ \textit{consistent}(g, D) \ \wedge \ (\nexists g' : g' \in \ H \ \wedge \ g' >_g \ g \ \wedge \ \textit{consistent}(g', D)) \ \}$$

Similarly, the set of maximally specific (i.e., minimally general) hypotheses, S, is defined as follows:

$$\{s \mid s \in H \ \wedge \ \textit{consistent}(s, D) \ \wedge \ (\nexists s' : s' \in \ H \ \wedge \ s >_g \ s' \ \wedge \ \textit{consistent}(s', D)) \ \}$$

Theorem 7 (Version Space Representation)

Let X be a feature space and let H be a set of boolean-valued functions with domain X. Moreover, let $c : X \to \{0, 1\}$ be a target concept and let D be a set of examples of the form $(\mathbf{x}, c(\mathbf{x}))$. Then, based on the $\geq_g$-relation, each member of the version space $V_{H,D}$ lies in between two members of G and S respectively:

$$V_{H,D} = \{h \mid h \in H \ \wedge \ (\ \exists g \in G \ \ \exists s \in S : \ \ g \geq_g h \geq_g s \ ) \ \}$$

Candidate Elimination Algorithm

- The CANDIDATE-ELIMINATION algorithm computes the version space containing all hypotheses from **H** that are consistent with an observed sequence of training examples.
- It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the **G** boundary set to contain the ***most general hypothesis in H***

$$G_0 \leftarrow \{(?, ?, ?, ?, ?, ?)\}$$

- and initializing the S boundary set to contain the most specific (least general) hypothesis

$$S_0 \leftarrow \{(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\}$$

2.5 Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H

S $\leftarrow$ maximally specific hypotheses in H

For each training example d=<x,c(x)>

Case 1 : If d is a positive example

Remove from G any hypothesis that is inconsistent with d

For each hypothesis s in S that is not consistent with d

- *Remove s from S.*
- *Add to S all minimal generalizations h of s such that*
  - *h consistent with d*
  - *Some member of G is more general than h*
- *Remove from S any hypothesis that is more general than another hypothesis in S*

Case 2: If d is a negative example

Remove from S any hypothesis that is inconsistent with d

For each hypothesis g in G that is not consistent with d

- *remove g from G.*
- *Add to G all minimal specializations h of g such that*
  - *h consistent with d*
  - *Some member of S is more specific than h*
- *Remove from G any hypothesis that is less general than another hypothesis in G*

Candidate Elimination Algorithm [Mitchell 1997]

1. Initialization: $G = \{g_0\},\ S = \{s_0\}$
2. If $\mathbf{x}$ is a positive example
   - Remove from G any hypothesis that is not consistent with $\mathbf{x}$
   - For each hypothesis s in S that is not consistent with $\mathbf{x}$
     - Remove s from S
     - Add to S all minimal generalizations h of s such that
       1. h is consistent with $\mathbf{x}$ and
       2. some member of G is more general than h
     - Remove from S any hypothesis that is less specific than another hypothesis in S
3. If $\mathbf{x}$ is a negative example
   - Remove from S any hypothesis that is not consistent with $\mathbf{x}$
   - For each hypothesis g in G that is not consistent with $\mathbf{x}$
     - Remove g from G
     - Add to G all minimal specializations h of g such that
       1. h is consistent with $\mathbf{x}$ and
       2. some member of S is more specific than h
     - Remove from G any hypothesis that is less general than another hypothesis in G

An Illustrative Example

Figure traces the CANDIDATE-ELIMINATION algorithm applied to the first two training examples from table

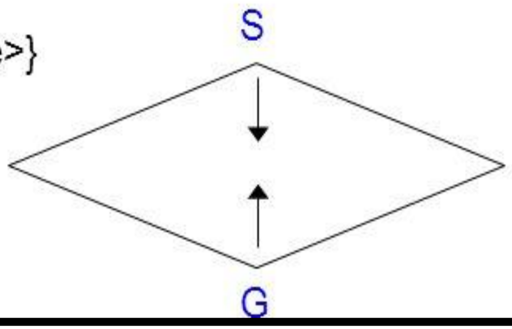
Candidate-Elimination Algorithm

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

$S_0 = \{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}$
$G_0 = \{<?, ?, ?, ?, ?, ?>\}$

$S_1 = \{<\text{Sunny, Warm, Normal, Strong, Warm, Same}>\}$
$G_1 = \{<?, ?, ?, ?, ?, ?>\}$

$S_2 = \{<\text{Sunny, Warm, ?, Strong, Warm, Same}>\}$
$G_2 = \{<?, ?, ?, ?, ?, ?>\}$



Trace1

S_0: $\{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}$

S_1: {<*Sunny, Warm, Normal, Strong, Warm, Same*>}

S_2: {<*Sunny, Warm, ?, Strong, Warm, Same*>}

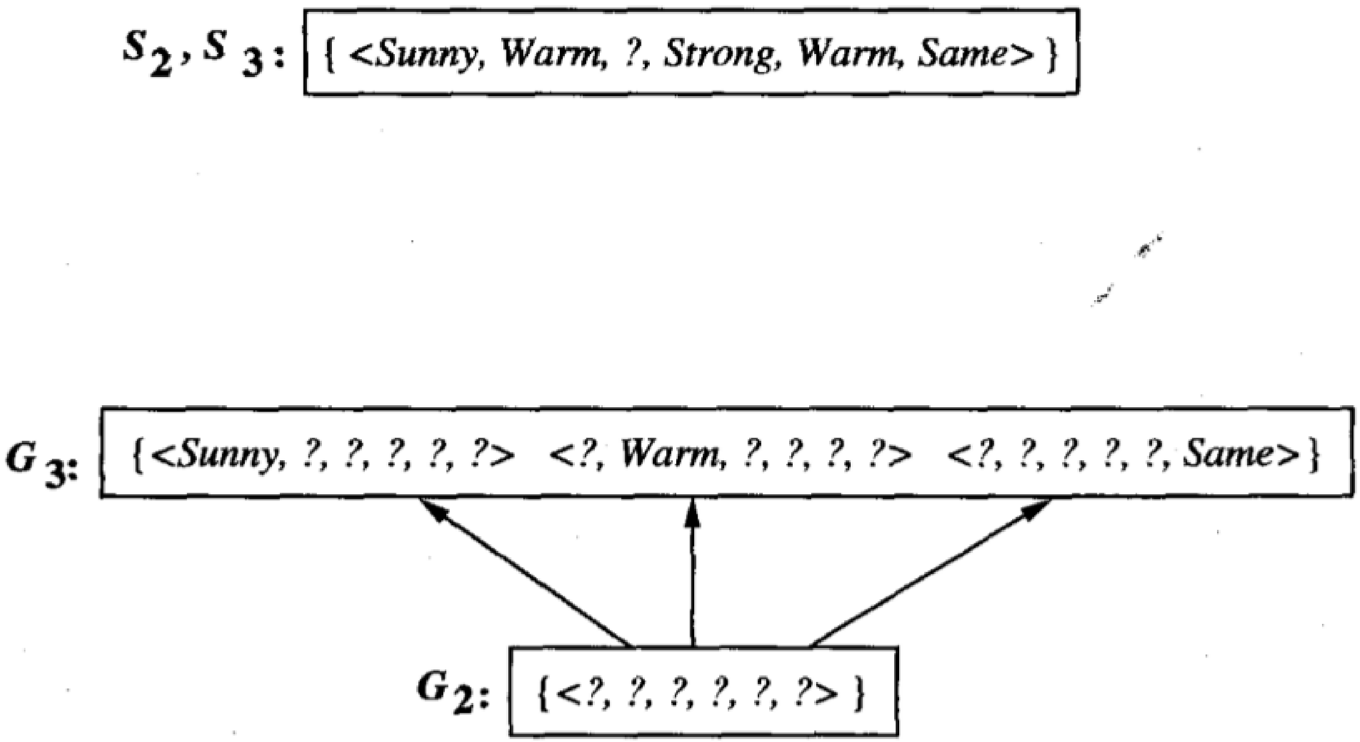
G_0, G_1, G_2: {<*?, ?, ?, ?, ?, ?*>}

Training examples:

1. <*Sunny, Warm, Normal, Strong, Warm, Same*>, *Enjoy Sport = Yes*
2. <*Sunny, Warm, High, Strong, Warm, Same*>, *Enjoy Sport = Yes*

CANDIDATE-ELIMINATION Trace 1. S_0 and G_0 are the initial boundary sets corresponding to the most specific and most general hypotheses. Training examples 1 and 2 force the S boundary to become more general, as in the FIND-S algorithm. They have no effect on the G boundary.

Trace 2

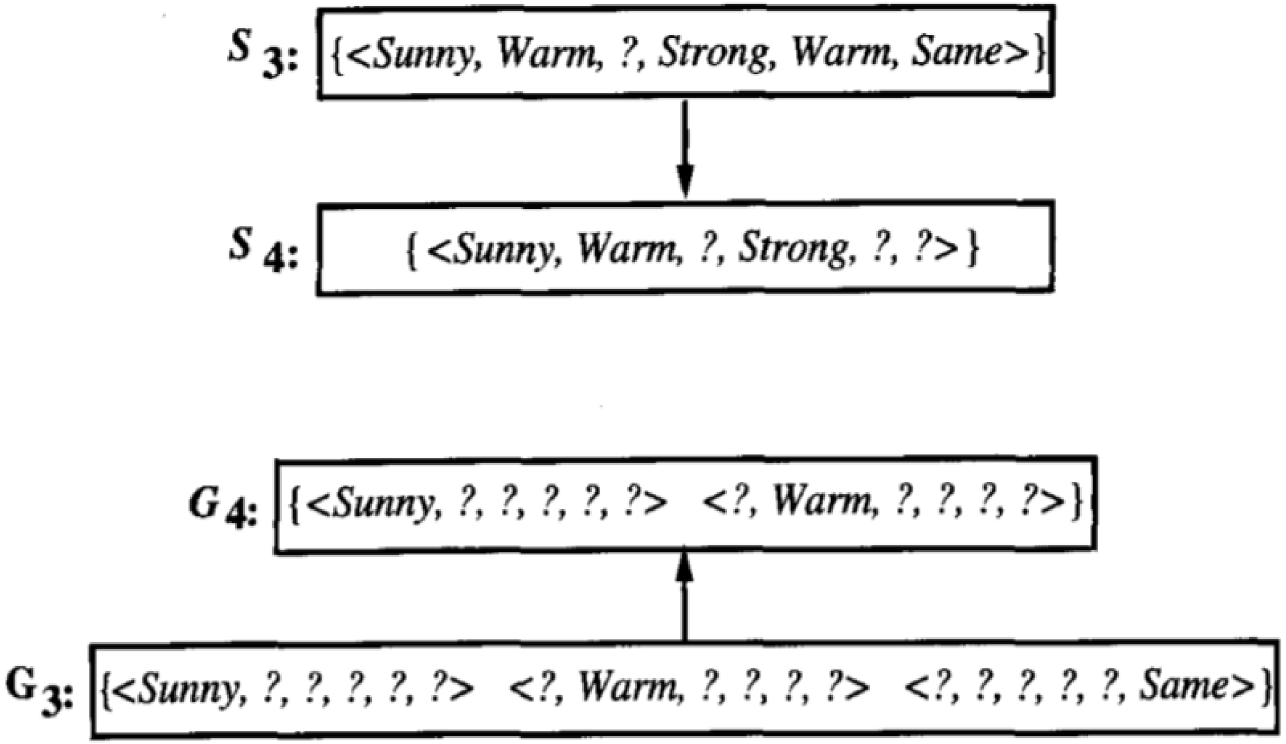


Training Example:

3. *<Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No*

CANDIDATE-ELIMINATION Trace 2. Training example 3 is a negative example that forces the G_2 boundary to be specialized to G_3. Note several alternative maximally general hypotheses are included in G_3.

Trace3

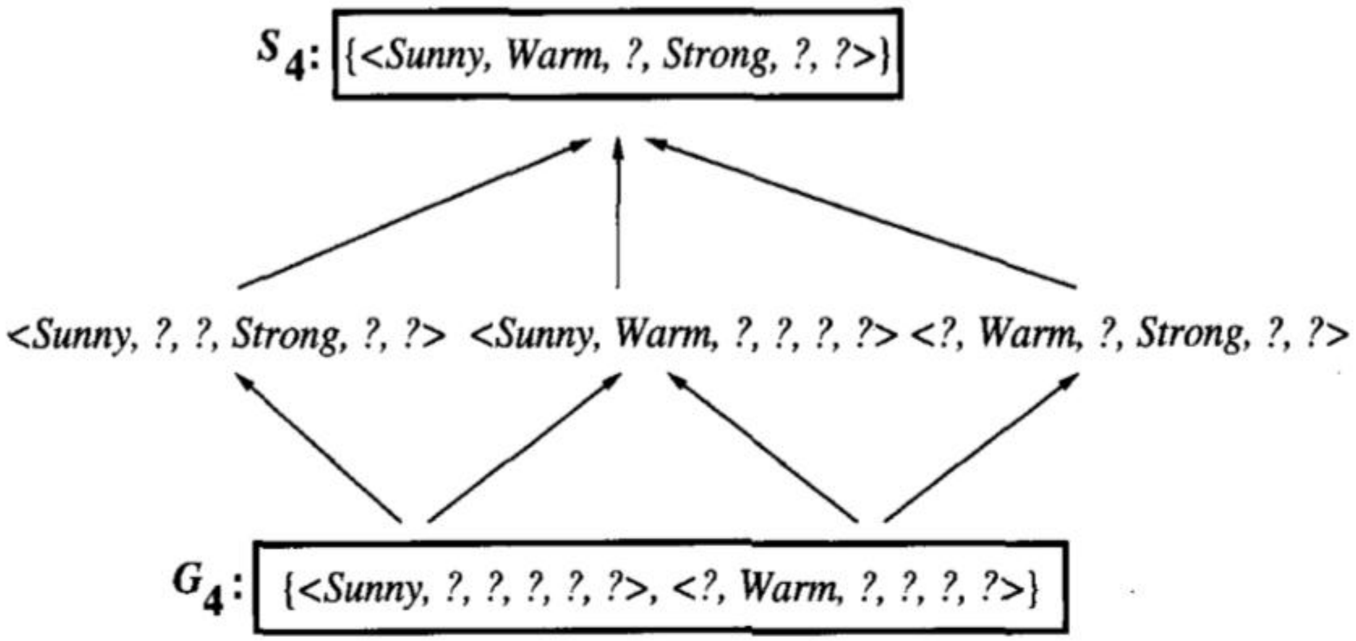


Training Example:

4. <*Sunny, Warm, High, Strong, Cool, Change*>, *EnjoySport* = *Yes*

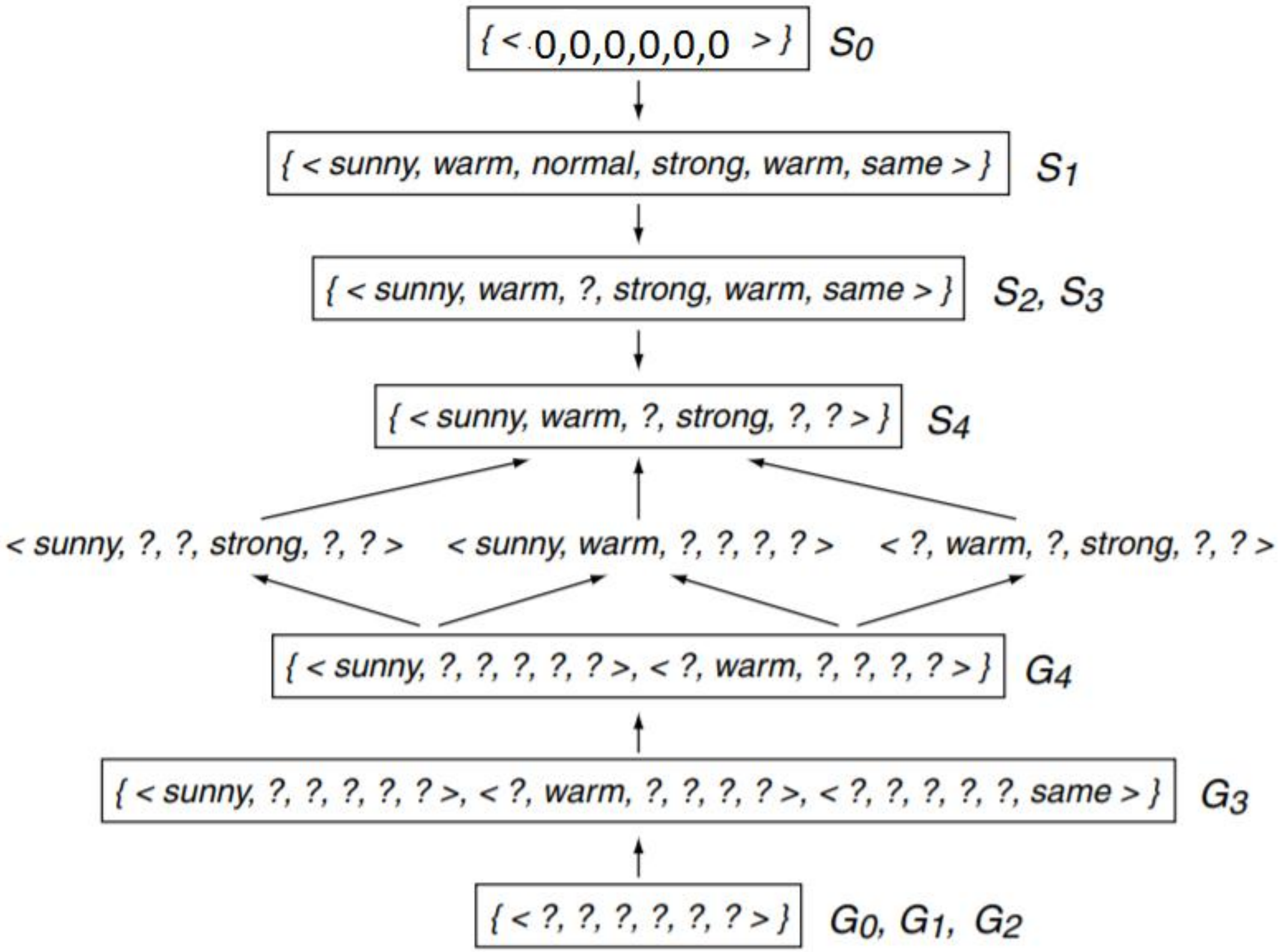
CANDIDATE-ELIMINATION Trace 3. The positive training example generalizes the S boundary, from S_3 to S_4. One member of G_3 must also be deleted, because it is no longer more general than the S_4 boundary.

Final Version Space



The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

Candidate Elimination Algorithm (illustration)



$\mathbf{x}_1 =$ (*sunny, warm, normal, strong, warm, same*) *EnjoySport*$(\mathbf{x}_1) = 1$

$\mathbf{x}_2 =$ (*sunny, warm, high, strong, warm, same*) *EnjoySport*$(\mathbf{x}_2) = 1$

$\mathbf{x}_3 =$ (*rainy, cold, high, strong, warm, change*) *EnjoySport*$(\mathbf{x}_3) = 0$

$\mathbf{x}_4 =$ (*sunny, warm, high, strong, cool, change*) *EnjoySport*$(\mathbf{x}_4) = 1$

Source Code

https://github.com/profthyagu

Github Repository

- https://github.com/profthyagu