

Agenda

Day2

1. Types of Machine Learning Algorithms

2. **Bayesian Learning:** Bayes Theorem and Concept Learning, ML and LS error Hypothesis, Naïve Bayes Classifier, Bayesian belief networks, EM Algorithm

- **Source Code**

- <https://github.com/profthyagu>

1. Classification of Machine Learning Algorithms

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

1.1 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.

1.2 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

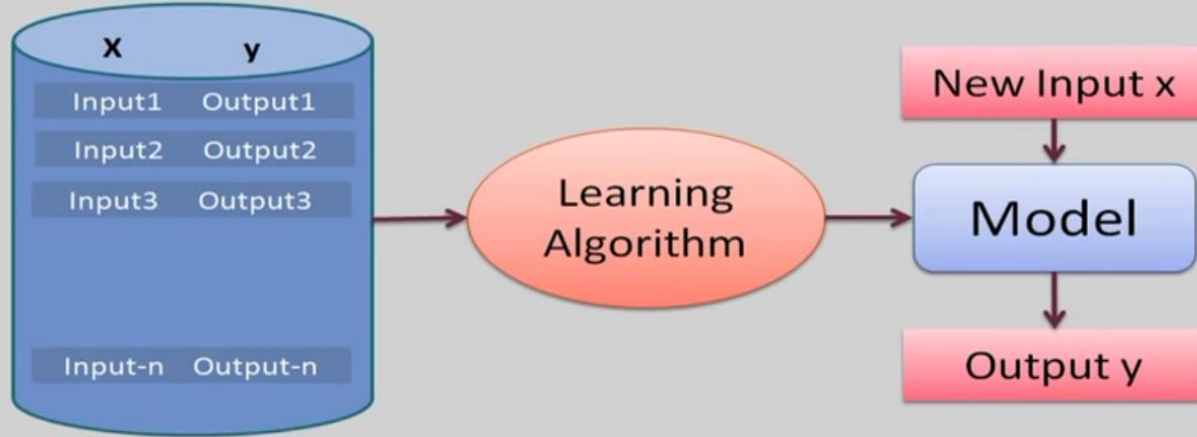
3. Reinforcement Learning

- Determine what to do based on Rewards and punishments

1.2.1. 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**

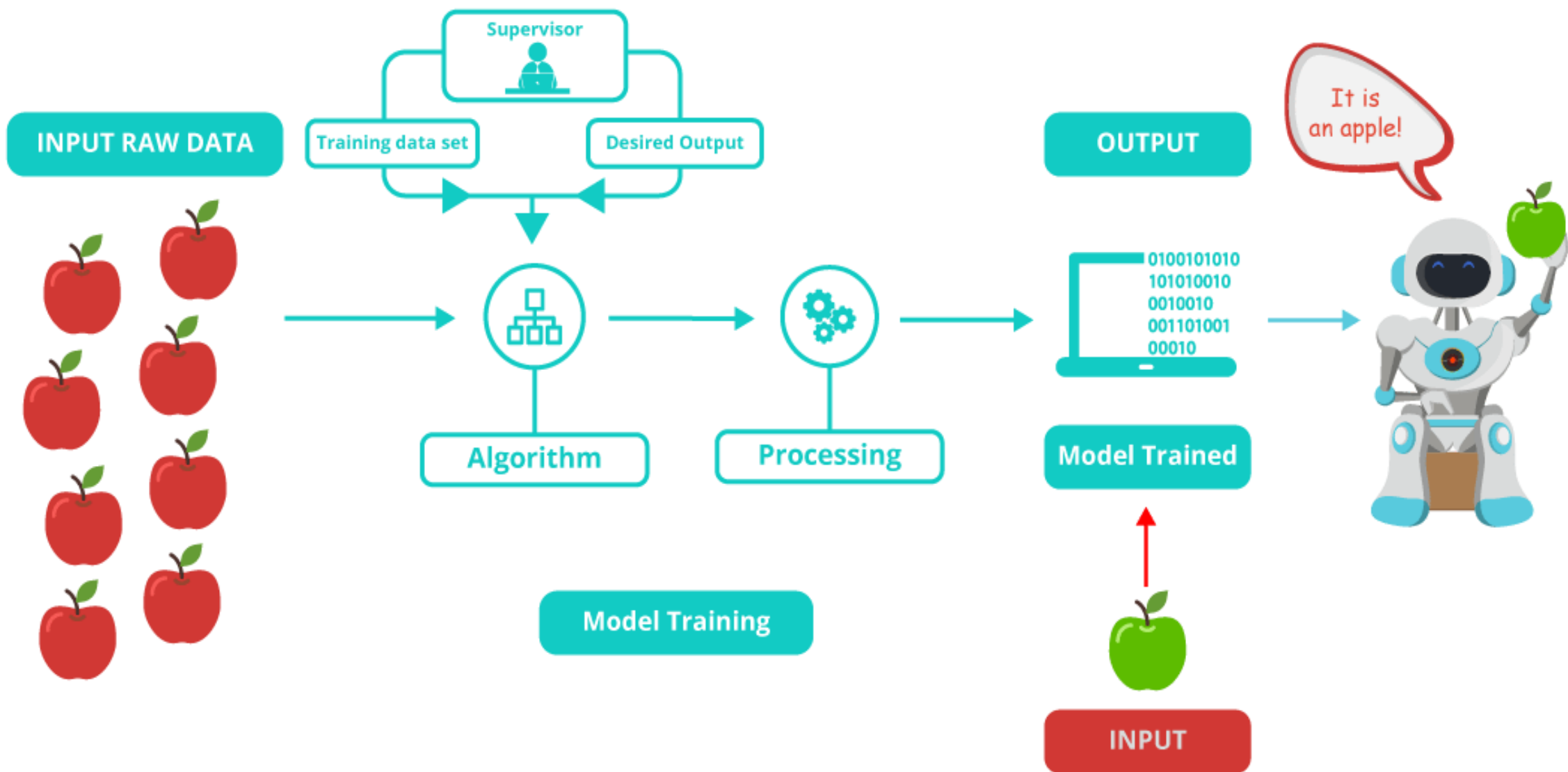
Supervised Learning



Supervised Learning
(Train Me)

1.2.2: Understanding Supervised Learning

- Let's see the mathematical definition of Supervised Learning.
- Supervised learning is the one where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. it,
- **$Y = f(X)$**
- The goal is to approximate the mapping function so well that whenever you get some new input data (x), the machine can easily predict the output variables (Y) for that data.



Supervised Learning Use cases



Cortana

Cortana or any speech automated system in your mobile phone trains your voice and then starts working based on this training. This is an application of Supervised Learning

Supervised Learning Use cases

Weather Apps

Predicts the upcoming weather by analyzing the parameters for a given time on some prior knowledge (*when its sunny, temperature is higher; when its cloudy, humidity is higher, etc.*).



Supervised Learning Use cases



Biometric Attendance

In **Biometric Attendance** you can train the machine with inputs of your biometric identity – it can be your thumb, iris or ear-lobe, etc. Once the machine is trained it can validate your future input and can easily identify you.

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

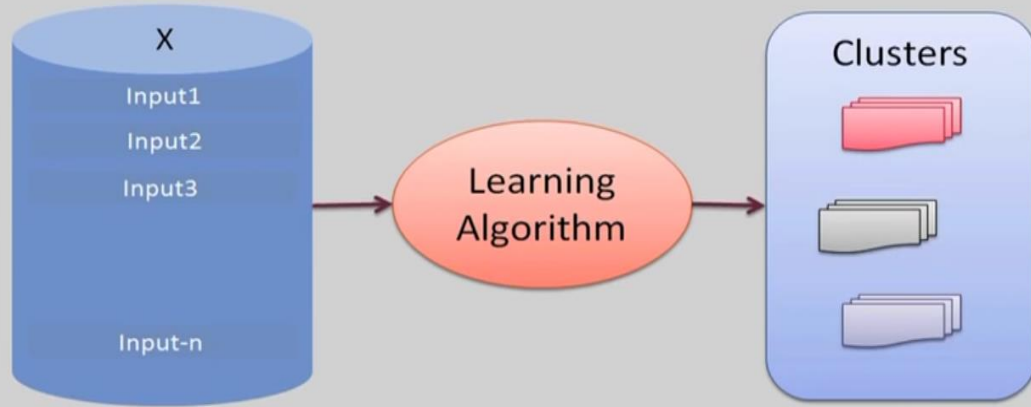
2. Regression (Continuous)

- a) Linear Regression
- b) SVR
- c) GPR
- d) Ensemble Methods

1.2.2: 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



Unsupervised Learning
(I am Self Learner)

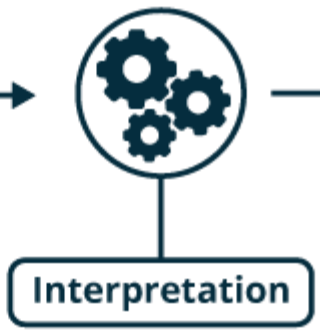
Understanding Unsupervised Learning

- **So What is Unsupervised Learning?**
- **Mathematically**, Unsupervised learning is where **you only have input data (X)** and no corresponding output variables.
- The goal for unsupervised learning is to model the *underlying structure or distribution in the data in order to learn more about the data.*

INPUT RAW DATA



- Unknown Output
- No Training Data Set

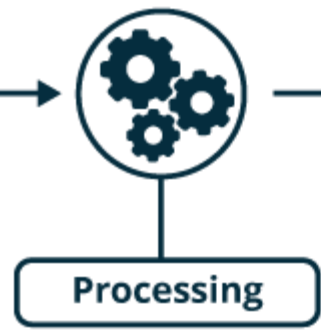


Interpretation

Algorithm

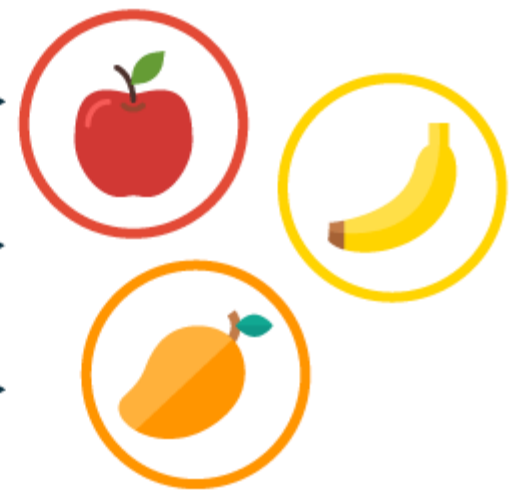


Model Training



Processing

OUTPUT



Model Trained

Unsupervised Learning Use cases



A friend invites you to his party where you meet totally strangers. Now you will classify them using unsupervised learning (no prior knowledge) and this classification can be on the basis of *gender, age group, dressing, educational qualification or whatever way you would like*. Since you didn't have any prior knowledge about people and so you just classified them "on-the-go".

Unsupervised Learning Use cases



Let's suppose you have never seen a Football match before and by chance watch a video on internet, now you can classify players on the basis of different criterion like Players wearing the same sort of kits are in one class, Players of one style are in one class (players, goal keeper, referee), or on the basis of playing style(attacker or defender) or whatever way you would observe, you can classify it.

Types of Unsupervised Learning

1. Clustering

- K Means Clustering
- Hierarchical Clustering
- Gaussian Mixture Models
- Genetic Algorithms
- Artificial Neural Networks

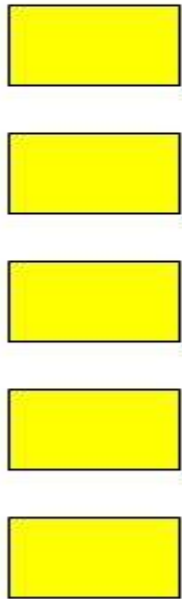
2. Dimensionality Reduction

- Tensor Decomposition
- Principal Component Analysis
- Multidimensional statistics
- Random Projection
- Artificial Neural Networks

3. Association Rules

Unsupervised learning: clustering

Raw data



extract
features

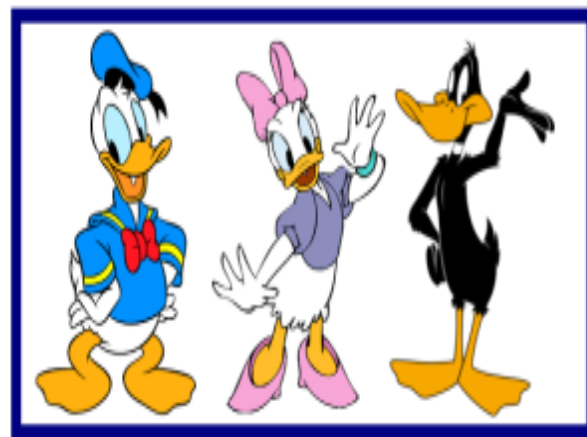
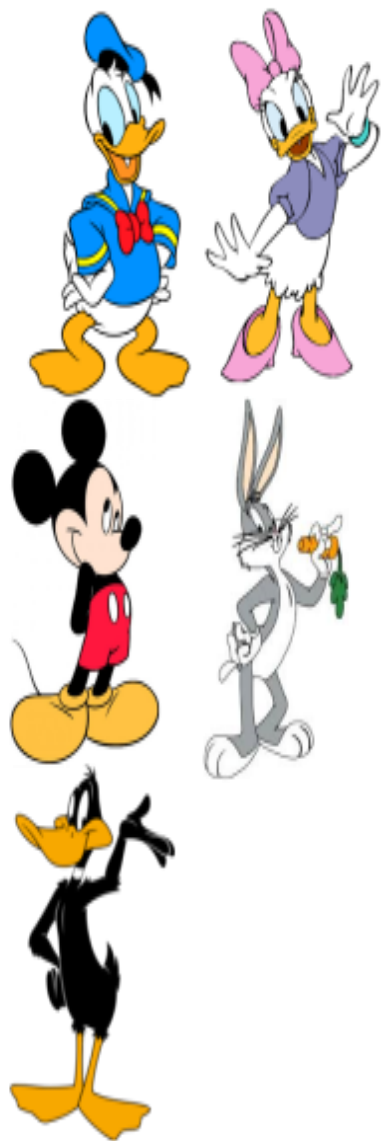
features

$f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$

group into
classes/clust
ers

Clusters

No “supervision”, we’re only given data and want to find natural groupings



1.3 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.

Reinforcement Learning
(My life, My rules!
Rewards
(Hit & Trial))

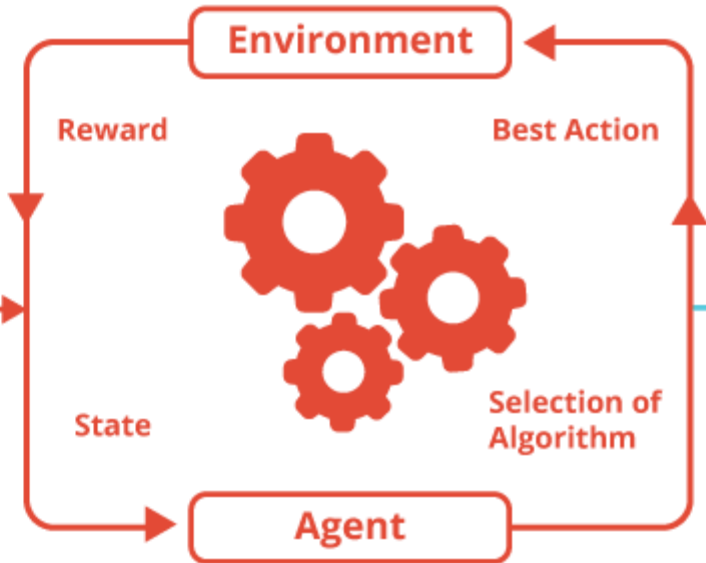
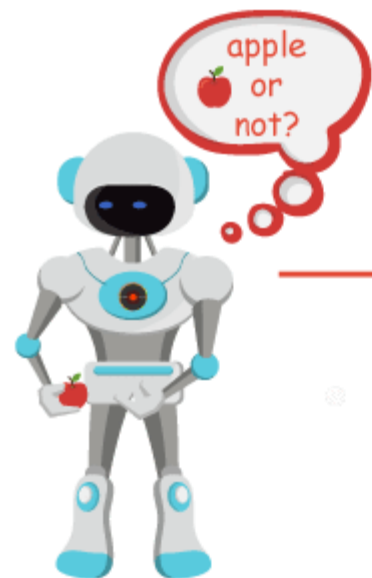
Understanding Reinforcement Learning

Reinforcement learning can be thought of as a hit and trial method of learning. The machine gets a Reward or Penalty point for each action it performs. If the option is correct, the machine gains the reward point or gets a penalty point in case of a wrong response.

The reinforcement learning algorithm is all about the interaction between the environment and the learning agent. The learning agent is based on exploration and exploitation.

Exploration is when the learning agent acts on trial and error and Exploitation is when it performs an action based on the knowledge gained from the environment. The environment rewards the agent for every correct action, which is the reinforcement signal. With the aim of collecting more rewards obtained, the agent improves its environment knowledge to choose or perform the next action.

INPUT RAW DATA



Model Training

OUTPUT



Model Trained

INPUT



It is an apple!



Let see how Pavlov trained his dog using reinforcement training?

Pavlov divided the training of his dog into four stages.

In the first part, *Pavlov gave meat to the dog*, and in response to the meat, the dog started salivating.

In the next stage he created a sound with a bell, but this time the dogs did not respond anything.

In the third stage, he tried to train his dog by using the bell and then giving them food. Seeing the food the dog started salivating.

Eventually, the dogs started salivating just after hearing the bell, even if the food was not given as the dog was reinforced that whenever the master will ring the bell, he will get the food.

1. Before Conditioning



Food

**Unconditioned
Stimulus**



Response



Salivation

**Unconditioned
Response**

2. Before Conditioning



Bell

Neutral stimulus



Response



No Salivation

**No Conditioned
Response**

3. During Conditioning



Bell

Food



Response



Salivation

**Unconditioned
Response**

4. After Conditioning



Bell

**Conditioned
Stimulus**

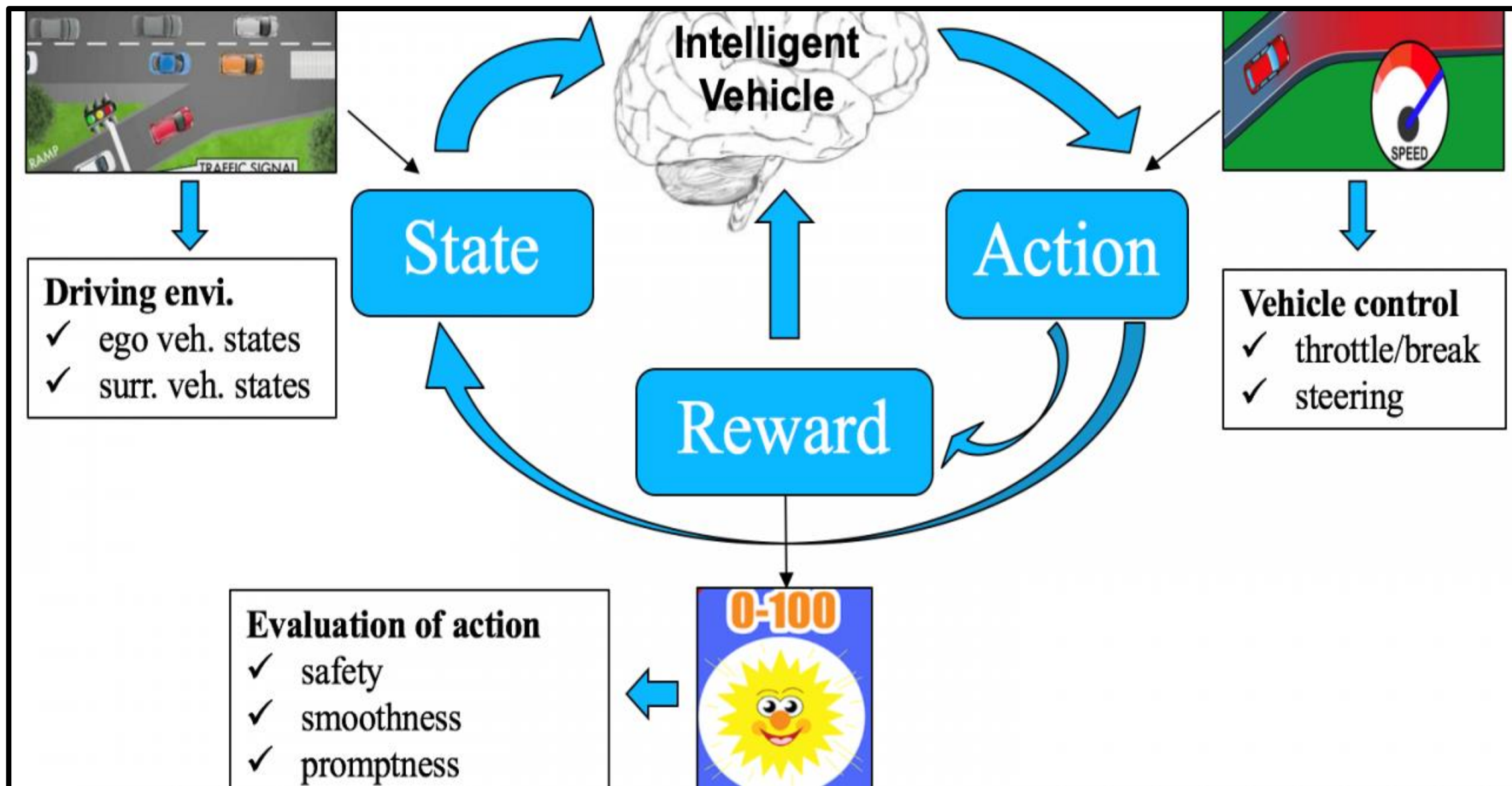


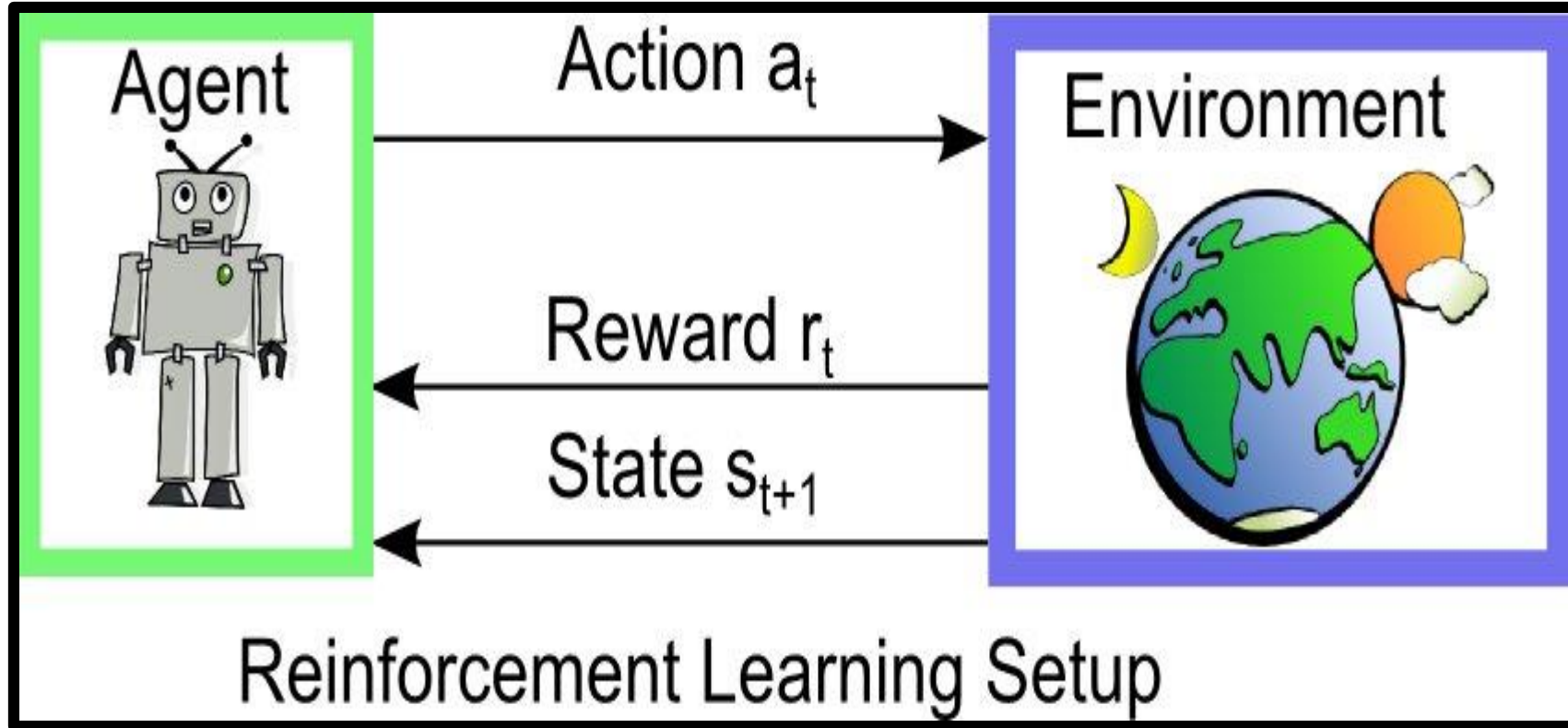
Response



Salivation

**Conditioned
Response**





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	<ul style="list-style-type: none"> ■ Predicting financial results ■ Fraud detection
Supervised	Classification and/or Regression	<ul style="list-style-type: none"> ■ Spam filtering ■ Fraud detection
Supervised	Decision tree	<ul style="list-style-type: none"> ■ 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	<ul style="list-style-type: none">■ Financial transactions■ Streaming analytics in IoT■ Underwriting in insurance
Unsupervised	Pattern recognition	<ul style="list-style-type: none">■ Spam detection■ Biometrics■ Identity management
Unsupervised	Association rule learning	<ul style="list-style-type: none">■ Security and intrusion detection■ Bioinformatics■ Manufacturing and Assembly

2. Bayesian Learning

1. Bayes Theorem
2. MAP, ML hypotheses
- 3. Relation to Concept Learning**
4. MAP learners
- 5. Maximum likelihood and Least Squared Error hypothesis**
6. Naive Bayes learner
7. Example: Learning over text data
8. Bayesian belief networks
9. Expectation Maximization algorithm

Two Roles for Bayesian Methods

- Provides practical approach for most of learning algorithms:
- Provides useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities .

Features of Bayesian Learning Methods

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Bayesian methods can accommodate hypothesis that make probabilities predictions.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities
- Provides a standard of optimal decision making against which other practical methods can be measured.

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



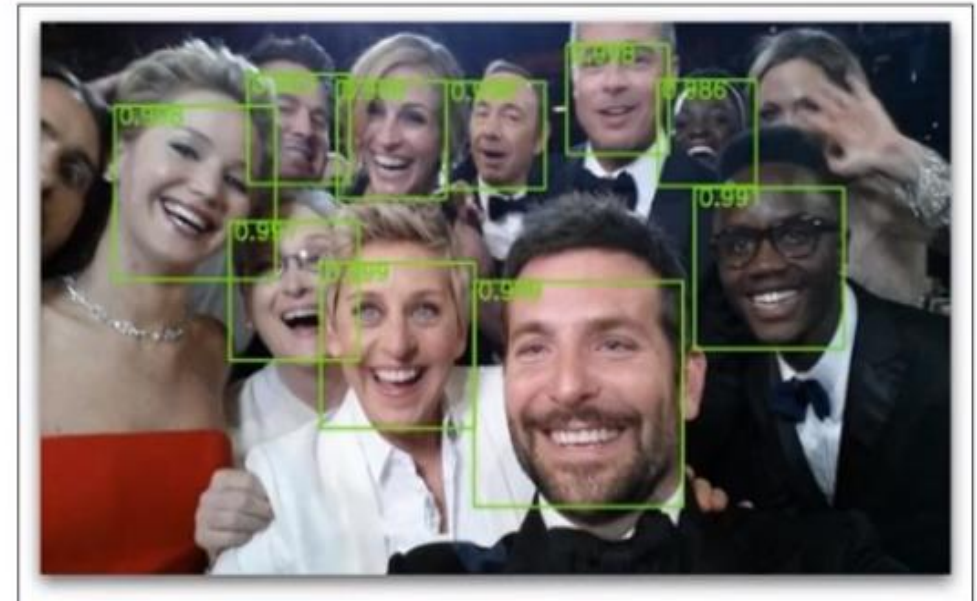
INDIA

Why is Aadhaar mandatory for PAN? SC asks Centre

Email Spam Detection



Face Recognition



Sentiment Analysis



Use cases

Medical Diagnosis



Digit Recognition



Weather Prediction

<p>TODAY</p> <p>62 37</p> <p>morning fog, partly cloudy</p> 	<p>TOMORROW</p> <p>58 41</p> <p>rain showers, cloudy</p> 	
---	--	---

Basic Formulas for Probabilities

- **Product Rule:** probability $P(A \wedge B)$ of a conjunction of two events A and B:

$$P(A \wedge B) = P(A \mid B) P(B) = P(B \mid A) P(A)$$

- **Sum Rule:** probability of a disjunction of two events A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- **Theorem of total probability:** if events A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n P(A_i) = 1$, then $P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$

Conditional Probability

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

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

Example :

$$P(A) = 4/52$$

$$P(B) = 4/51$$

$$P(A \text{ and } B) = 4/52 * 4/51 = 0.006$$

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

2.1 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;

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 $P(A | B) = P(B | A)P(A) / P(B)$

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?

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

Bayes Theorem

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

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

- $P(h)$ = prior (initial) probability that hypothesis h holds , before we observed any training data.
- $P(D)$ = prior probability of training data D
- $P(h | D)$ = posterior probability of h given D (*it holds after we have seen the training data D*)
- $P(D | h)$ = probability of observing data D given some world in which hypothesis h holds.

2.2.1 Maximum a posterior (MAP) hypothesis

- In many learning scenarios , the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypotheses $h \in H$ given the observed data D .
- Any such maximally probable hypothesis is called a maximum posteriori (MAP) hypothesis h_{MAP} :

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

2.2.2 Maximum Likelihood

- In some cases we will assume that every hypothesis in H is equally probable a priori ($P(h_i) = P(h_j)$ for all h_i in H) then can further simplify and need to consider the term $P(D|h)$ is often called the likelihood of the data D given h and hypothesis that maximizes $P(D|h)$ is called a **Maximum likelihood** (ML) hypothesis h_{ML}

$$h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$$

An Example : Cancer Patient Diagnosis

- To illustrate Bayes Rule , Consider a medical diagnosis problem in which there are two alternative hypotheses :
 1. That the patient has a particular form of cancer and
 2. That the patient does not.

The available data is from a particular laboratory test with two possible outcomes :

+ : positive

- : negative

Example : Medical Cancer Test Details of Patient

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$$P(cancer) = 0.008$$

$$P(\neg cancer) = 0.992$$

$$P(+ | cancer) = 0.98$$

$$P(- | cancer) = 0.02$$

$$P(+ | \neg cancer) = 0.03$$

$$P(- | \neg cancer) = 0.97$$

Example : Does patient have cancer or not?

The Maximum a posterior hypothesis for Patient having cancer/no cancer :

$$\mathbf{cancer}_{MAP} = P(+ | \text{Cancer}) P(\text{cancer}) = (0.98)(0.008) = \mathbf{0.0078}$$

$$\neg \mathbf{cancer}_{MAP} = P(\neg | \text{Cancer}) P(\text{cancer}) = (0.03)(0.992) = \mathbf{0.0298}$$

2.3 Relation to Concept Learning

- Consider our usual concept learning task
 - instance space \mathbf{X} , hypothesis space \mathbf{H} , training examples D
 - consider the **FindS** learning algorithm (outputs most specific hypothesis from the version space $V S_{H,D}$)
- What would Bayes rule produce as the MAP hypothesis?
- Does *FindS* output a MAP hypothesis??

Brute Force Bayes Concept Learning

- Assume that the learner considers some finite hypothesis space H defined over the instance space X , in which the task is to learn some target concept $c: X \rightarrow \{0,1\}$
- Assume fixed set of instances $\langle x_1, \dots, x_m \rangle$
- Assume D is the set of classifications: $D = \langle c(x_1), \dots, c(x_m) \rangle$
- Assume that the learner has given some sequence of training examples $\langle \langle x_1, d_1 \rangle \langle x_2, d_2 \rangle, \dots, \langle x_m, d_m \rangle \rangle$ where x_i is some instance from X and where d_i is the target value of x_i (i.e $d_i = c(x_i)$).

Brute Force MAP Learning Algorithm

1. For each hypothesis h in H , calculate the posterior probability

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

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h | D)$$

Assumptions

The probability distribution $P(h)$ and $P(D|h)$ is chosen to be consistent with the following assumptions :

1. The training data D is noise free(i.e. $d_i = c(x_i)$)
2. The target concept c is contained in the hypothesis space H
3. We have no *a priori reason* to believe that any hypothesis is more probable than any other.

The Values of $P(h)$ and $P(D|h)$

- Choose $P(h)$ to be *uniform* distribution
 - $P(h) = 1/|H|$ for all h in H
- Choose $P(D|h)$:

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \text{ (} h \text{ consistent with } D \text{)} \\ 0 & \text{otherwise} \end{cases}$$

Two cases

- By Applying Bayes theorem $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$
- **Case1** : When h is inconsistent with training data D :

$$P(h|D) = 0.P(h)/P(D) = 0$$

- **Case 2**: When h is inconsistent with D , we have

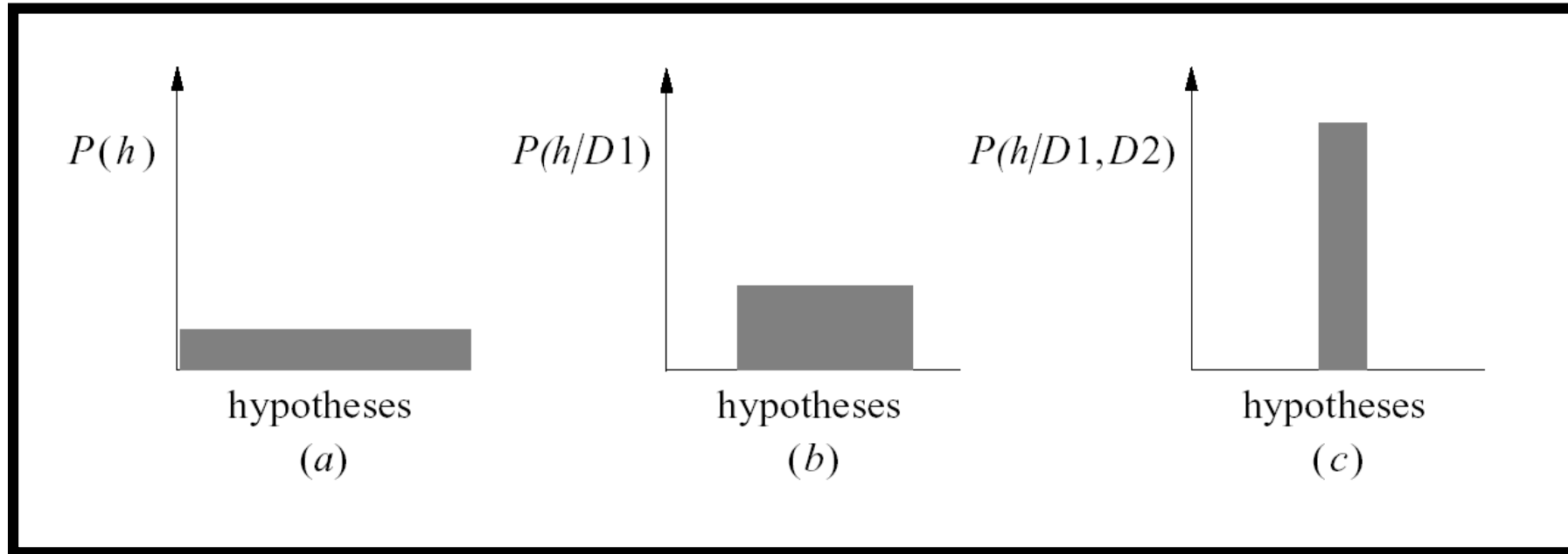
$$\begin{aligned} P(h|D) &= (1 * 1/|H|)/(|V_{SH,D}|/|H|) \\ &= 1/|V_{SH,D}| \end{aligned}$$

To Summarize

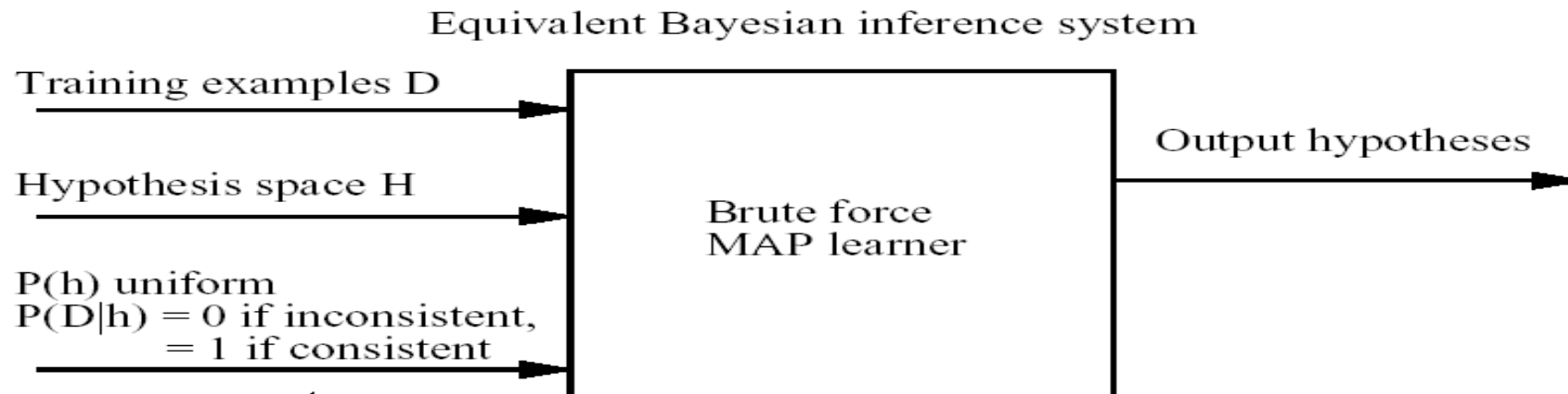
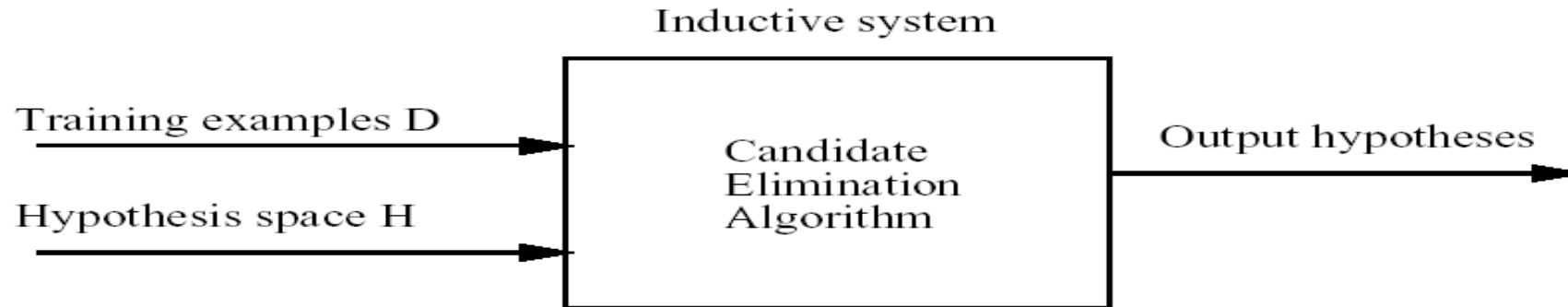
- To summarize , Bayes theorem implies that the posterior probability $P(h|D)$ under our assumed $P(h)$ and $P(D|h)$ is

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

2.4 MAP hypothesis and Consistent Learners



Characterizing Learning Algorithms by Equivalent MAP Learners

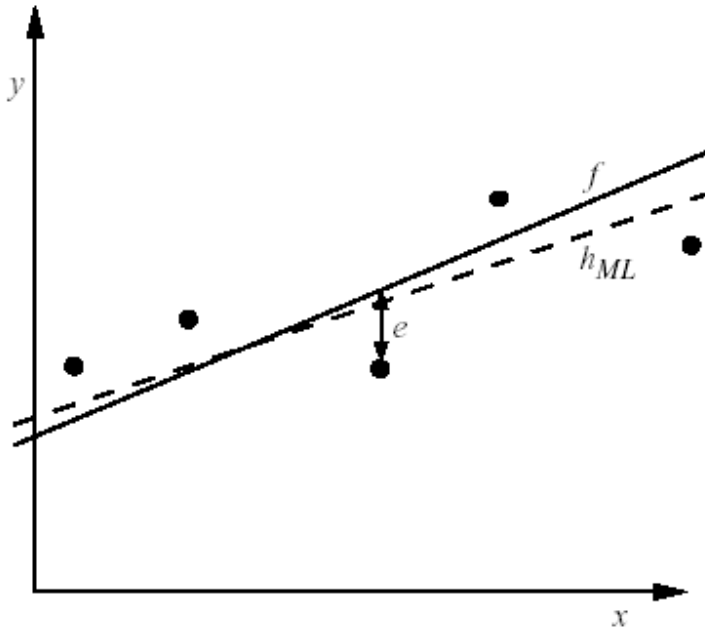


*Prior assumptions
made explicit*

2.5 Maximum likelihood and Least Squared Error hypothesis

- A straightforward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood hypothesis.

Learning A Real Valued Function



- Consider any real-valued target function f
Training examples $\langle x_i, d_i \rangle$, where d_i is noisy training value
 - $d_i = f(x_i) + e_i$
 - e_i is random variable (noise) drawn independently for each x_i according to some Gaussian distribution with mean=0
- Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors:

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$$

Learning A Real Valued Function

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} p(D|h) \\&= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i|h) \\&= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}\end{aligned}$$

- Maximize natural log of this instead...

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\&= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\&= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -(d_i - h(x_i))^2 \\&= \operatorname{argmin}_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2\end{aligned}$$

Maximum Likelihood Hypothesis Learning to Predict Probabilities

- Consider predicting survival probability from patient data
- Training examples $\langle x_i, d_i \rangle$, where d_i is 1 or 0
- Want to train neural network to output a *probability* given x_i (not a 0 or 1)
- In this case can show

$$h_{ML} = \operatorname{argmax}_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))$$

- Weight update rule for a sigmoid unit:

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

where

$$\Delta w_{jk} = \eta \sum_{i=1}^m (d_i - h(x_i)) x_{ijk}$$

2.6 Naive Bayes classifier /Bayes Rule

2.6 Naive Bayes classifier /Bayes Rule

Using [Bayes' theorem](#), the conditional probability can be decomposed as

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

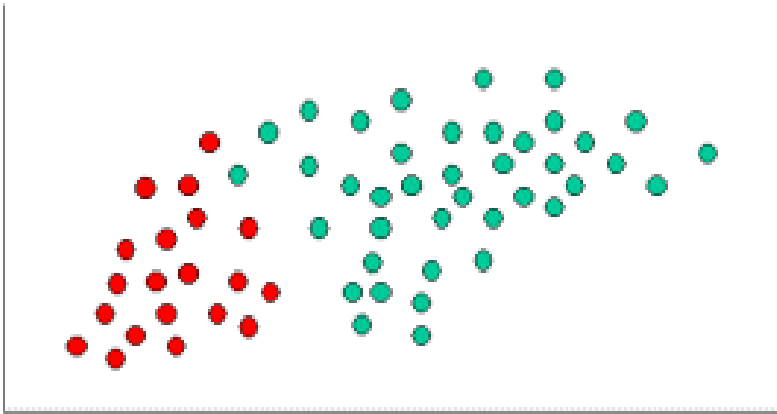
In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C

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

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

Naive Bayes classifier

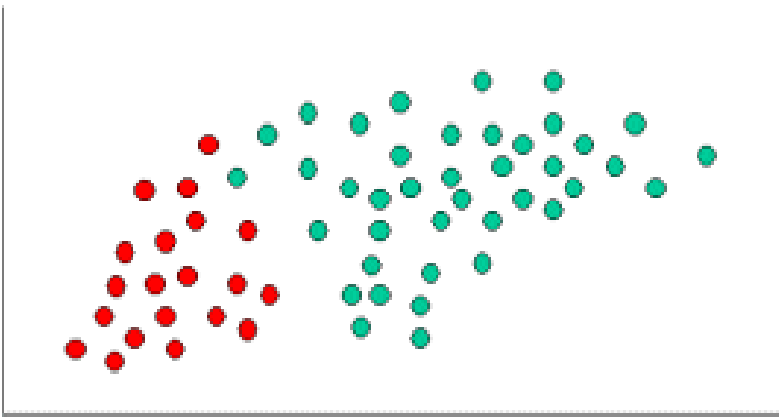
- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.



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

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

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



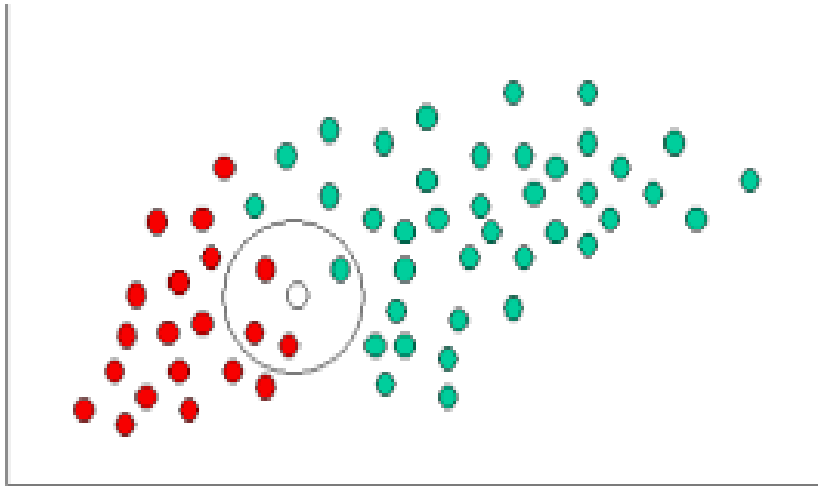
$$\text{Prior probability for GREEN} \propto \frac{40}{60}$$

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

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).

Posterior probability of X being GREEN \propto

Prior probability of GREEN \times Likelihood of X given GREEN

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

Posterior probability of X being RED \propto

Prior probability of RED \times Likelihood of X 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.

The Multinomial Naive Bayes's conditional distribution is:

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

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

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

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:

$$\hat{P}(x_i | \omega_j) = \frac{\sum t f(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega_j} + \alpha \cdot V}$$

where

- x_i : A word from the feature vector \mathbf{x} of a particular sample.
- $\sum t f(x_i, d \in \omega_j)$: 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 *naïve* assumption of conditional independence).

$$P(\mathbf{x} | \omega_j) = P(x_1 | \omega_j) \cdot P(x_2 | \omega_j) \cdot \dots \cdot P(x_n | \omega_j) = \prod_{i=1}^n P(x_i | \omega_j)$$

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" .

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

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" .

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

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 (*S1*) learns the probability terms $P(\mathbf{w}_k \mid \mathbf{v}_i)$, describing the probability that a randomly drawn word from a document in **class** \mathbf{v}_i will be the English word \mathbf{w}_k . It also learns the class prior probabilities $P(\mathbf{v}_i)$.

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 $P(w_k \mid v_j)$ 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*

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

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

ರಾಜ್ಯ ರಾಜ್ಯ ಬಿ.ಎಸ್

S2:CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

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

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

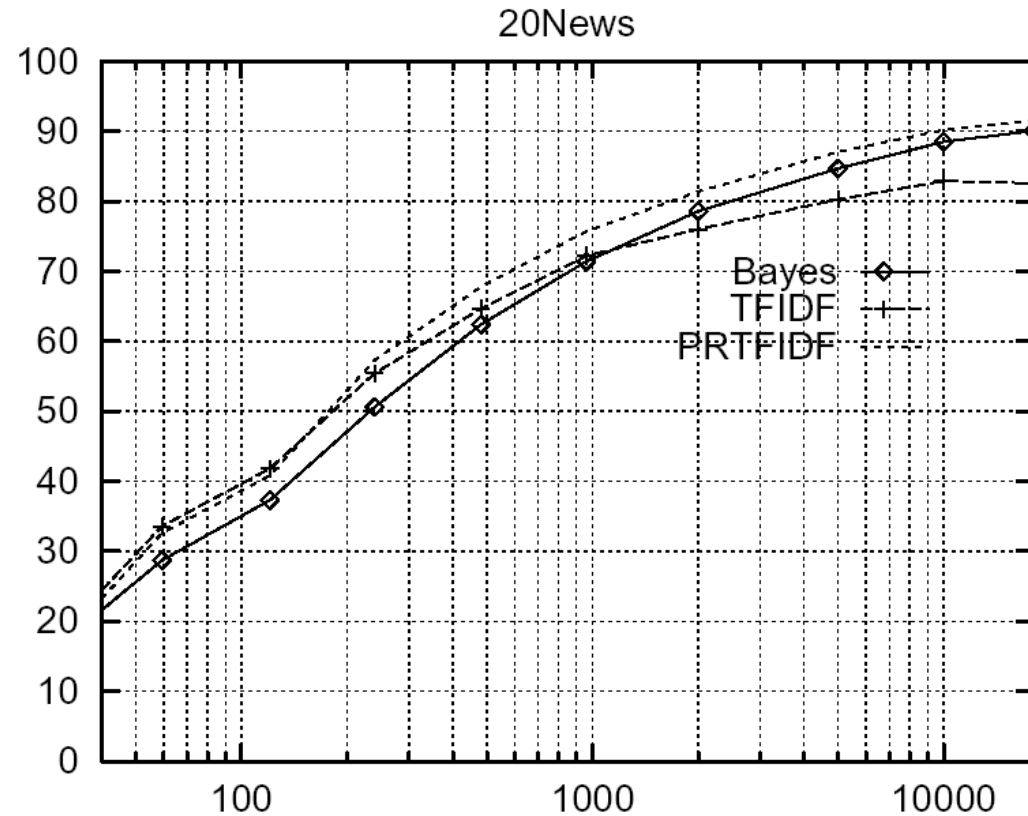
Twenty NewsGroups

- 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.
- **$P(a \text{ very close game}) = P(a) \times P(\text{very}) \times P(\text{close}) \times P(\text{game})$**
- **$P(a \text{ very close game} \mid \text{Sports}) = P(a \mid \text{Sports}) \times P(\text{Very} \mid \text{Sports}) \times P(\text{close} \mid \text{Sports}) \times P(\text{game} \mid \text{Sports})$**
- **$P(a \text{ very close game} \mid \text{Not Sports}) = P(a \mid \text{Not Sports}) \times P(\text{very} \mid \text{Not Sports}) \times P(\text{close} \mid \text{Not Sports}) \times P(\text{game} \mid \text{Not Sports})$**

Step 2: Calculating the probabilities

- Here , the word “close” does not exist in the category Sports, thus $P(\text{close} \mid \text{Sports}) = 0$, leading to $P(\text{a very close game} \mid \text{Sports})=0$.
- Given an observation $x = (x_1, \dots, x_d)$ from a **multinomial distribution** with N trials and parameter vector $\theta = (\theta_1, \dots, \theta_d)$, a "smoothed" version of the data gives the estimator.

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

- where the pseudo count $\alpha > 0$ is the smoothing parameter ($\alpha = 0$ 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	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

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

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

$$\begin{aligned}
& P(a \text{ --- Not Sports}) \times P(very|Not Sports) \times P(close|Not Sports) \times P(game|Not Sports) \times \\
& P(Not Sports) \\
& = 1.43 \times 10^{-5} \\
& = 0.0000143
\end{aligned}$$

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:

$$\hat{P}(x_i | \omega_j) = \frac{\sum t f(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega_j} + \alpha \cdot V}$$

where

- x_i : A word from the feature vector \mathbf{x} of a particular sample.
- $\sum t f(x_i, d \in \omega_j)$: 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 *naïve* assumption of conditional independence).

$$P(\mathbf{x} | \omega_j) = P(x_1 | \omega_j) \cdot P(x_2 | \omega_j) \cdot \dots \cdot P(x_n | \omega_j) = \prod_{i=1}^n P(x_i | \omega_j)$$

Source Code

2.7 Bayesian Network (BAYESIAN BELIEF NETWORKS)

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

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

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

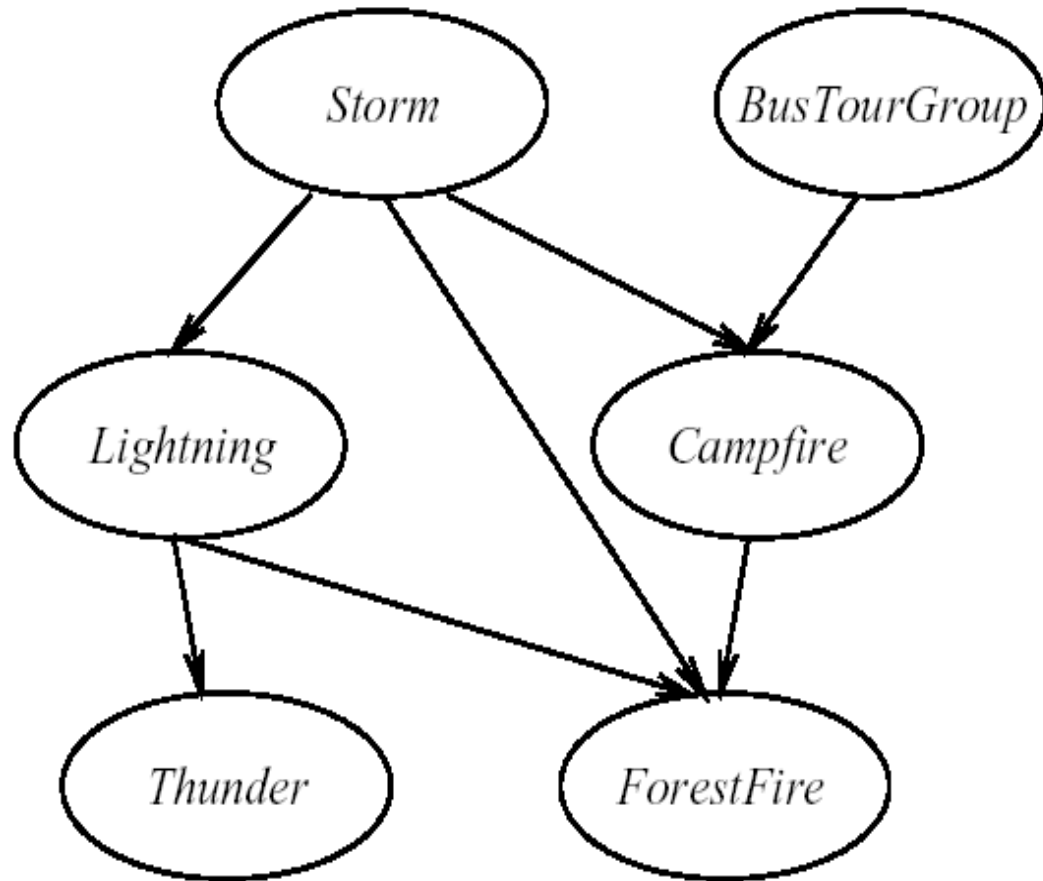
- Example: *Thunder* is conditionally independent of *Rain*, given *Lightning*

$$P(\text{Thunder} | \text{Rain}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})$$

- 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., $P(\text{Storm}, \text{BusTourGroup}, \dots, \text{ForestFire})$

- in general,
$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Parents}(Y_i))$$

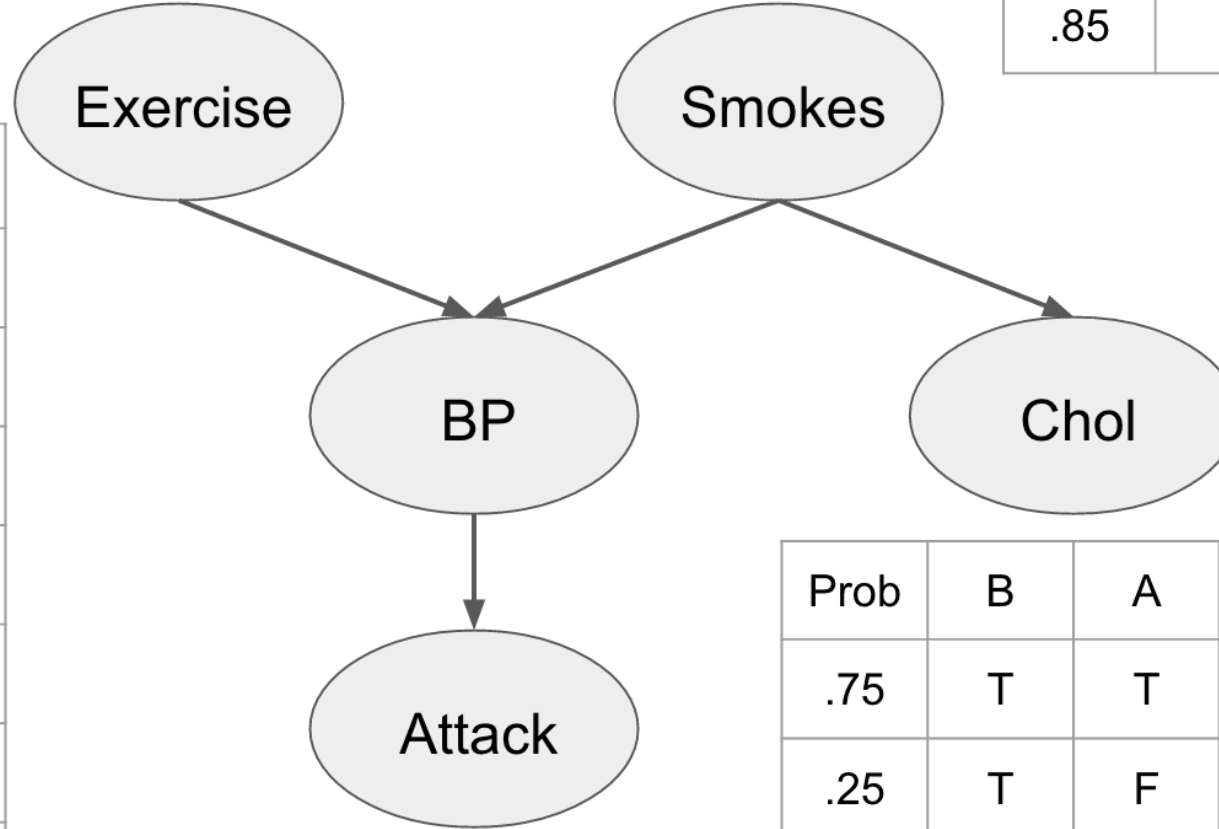
where $\text{Parents}(Y_i)$ denotes immediate predecessors of Y_i in graph

- so, joint distribution is fully defined by graph, plus the $P(y_i | \text{Parents}(Y_i))$

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

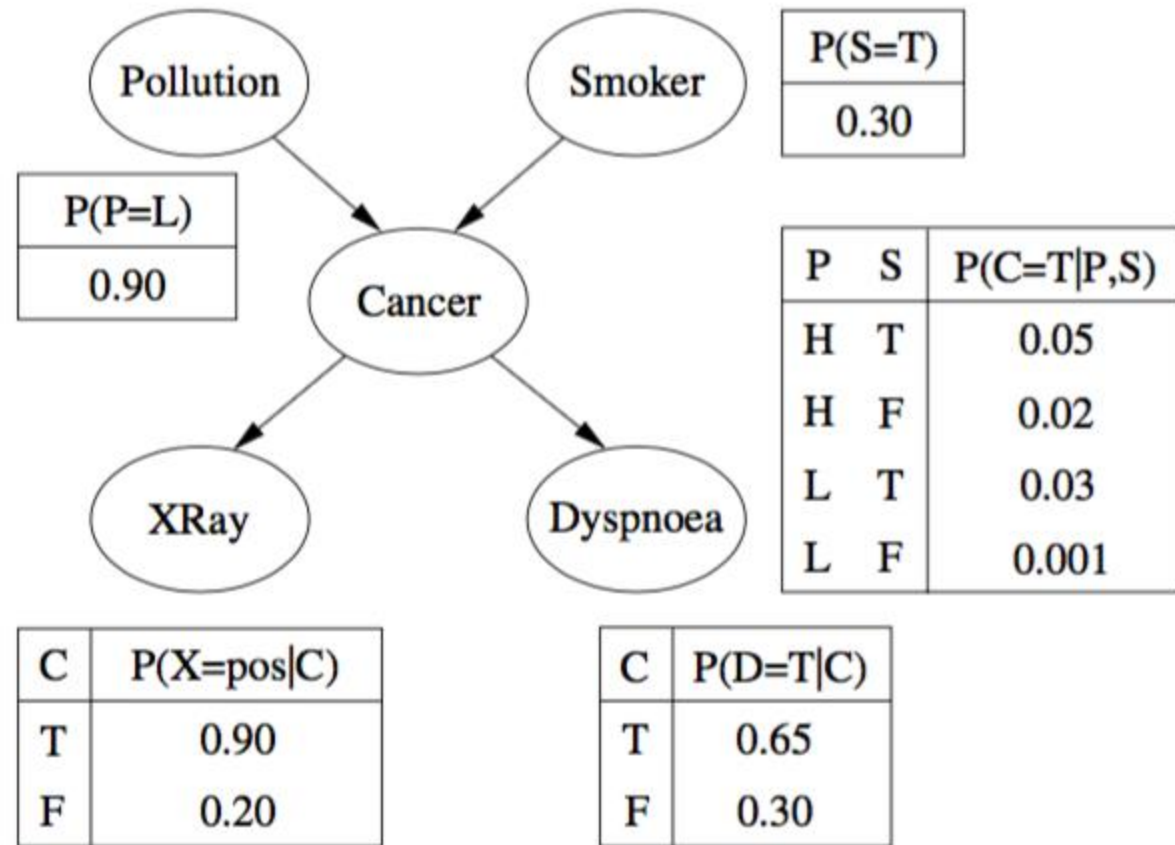


FIGURE 2.1

A BN for the lung cancer problem.

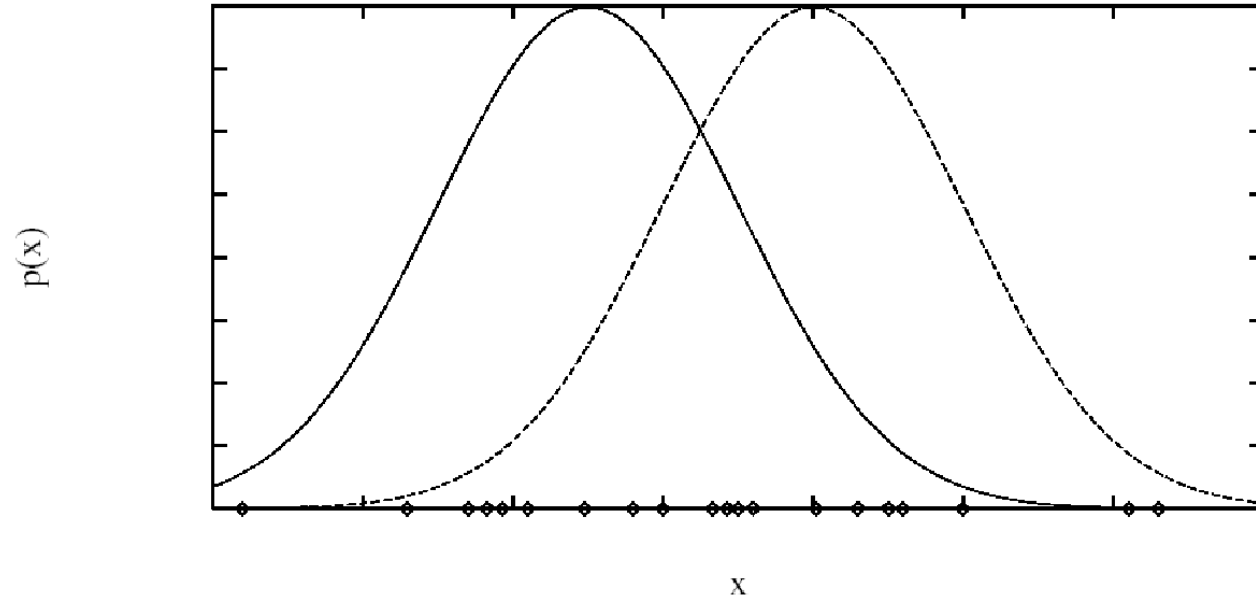
Lab Program5

- **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.**

Source Code

2.8 EM Algorithm

Generating Data from Mixture of k Gaussians

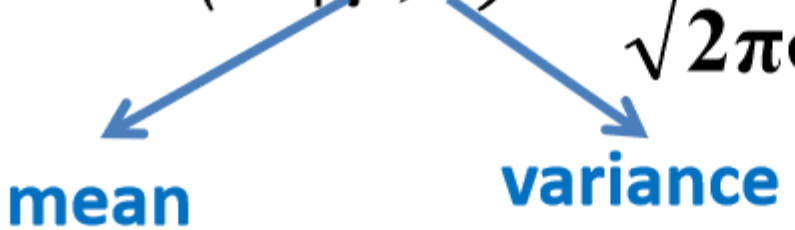


- Each instance x generated by
 1. Choosing one of the k Gaussians with uniform probability
 2. Generating an instance at random according to that Gaussian

Gaussian Distribution

□ Univariate Gaussian Distribution


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



mean variance

□ Multi-Variate Gaussian Distribution

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



mean covariance

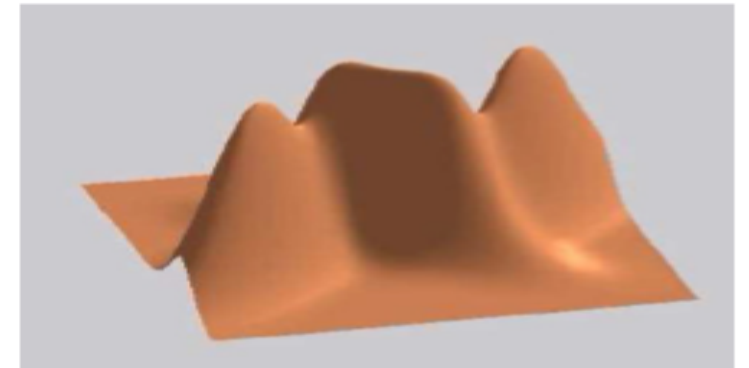
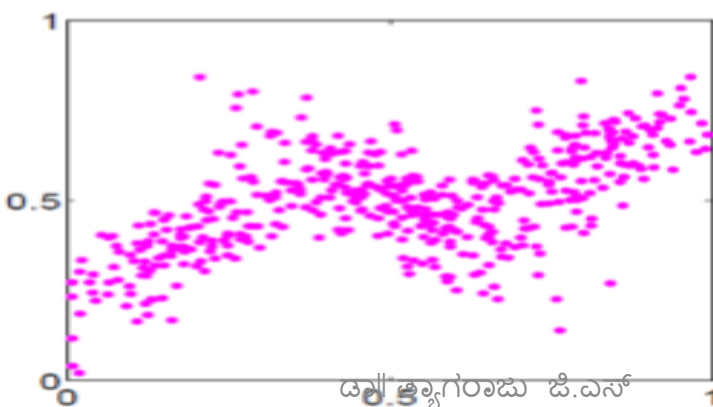
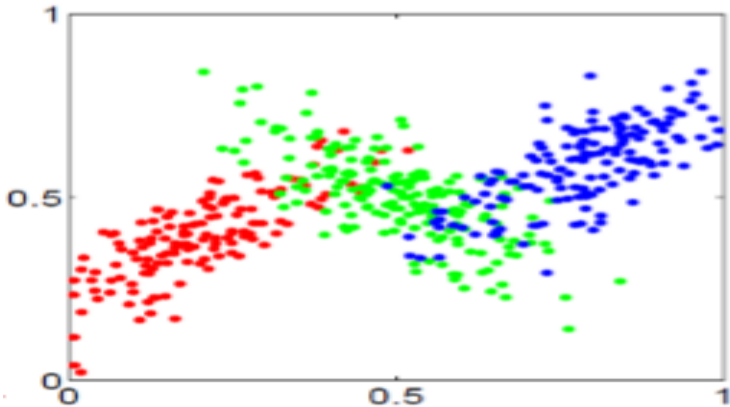
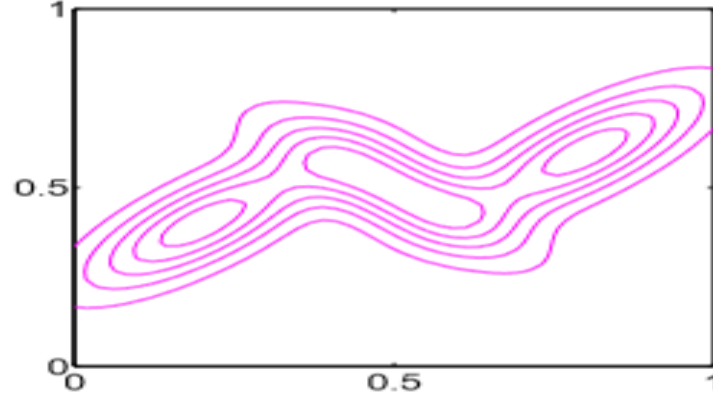
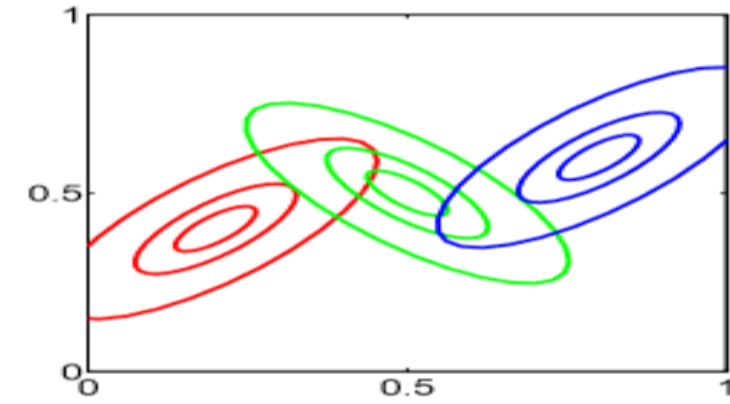
Gaussian Mixtures

- Linear super-position of Gaussians

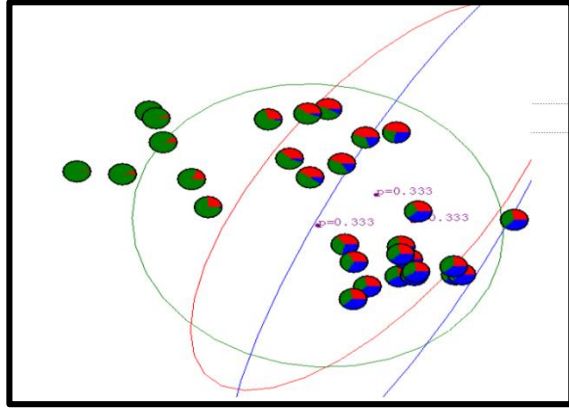
$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

Number of Gaussians

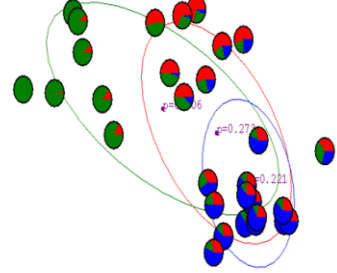
Mixing coefficient: weightage
for each Gaussian dist.



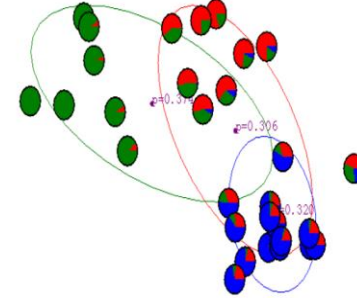
GMM : Example



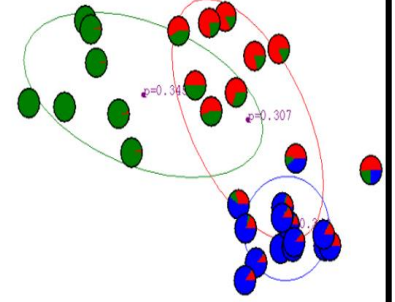
After first iteration



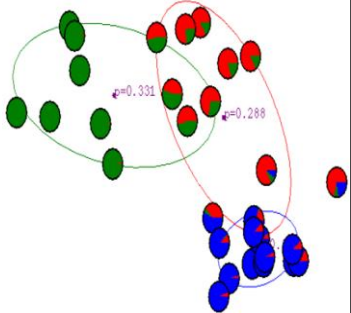
After 2nd iteration



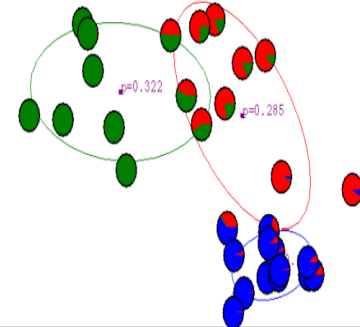
After 3rd iteration



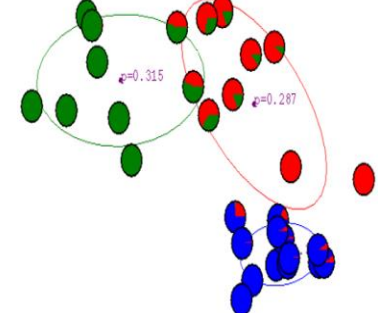
After 4th iteration



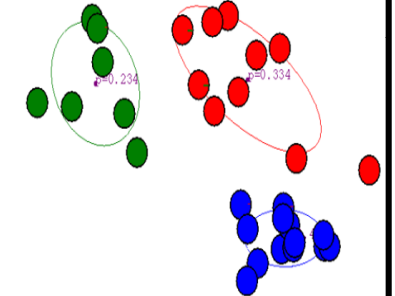
After 5th iteration



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 $\langle \mu_1, \dots, \mu_k \rangle$ of the k Gaussians
 - Don't know which instance x_i was generated by which Gaussian
- Determine:
 - Maximum likelihood estimates of $\langle \mu_1, \dots, \mu_k \rangle$

- EM Algorithm:

- Pick random initial $h = \langle \mu_1, \mu_2 \rangle$ then iterate

E step: Calculate the expected value $E[z_{ij}]$ of each **hidden variable** z_{ij} , assuming the current hypothesis

$h = \langle \mu_1, \mu_2 \rangle$ holds.

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

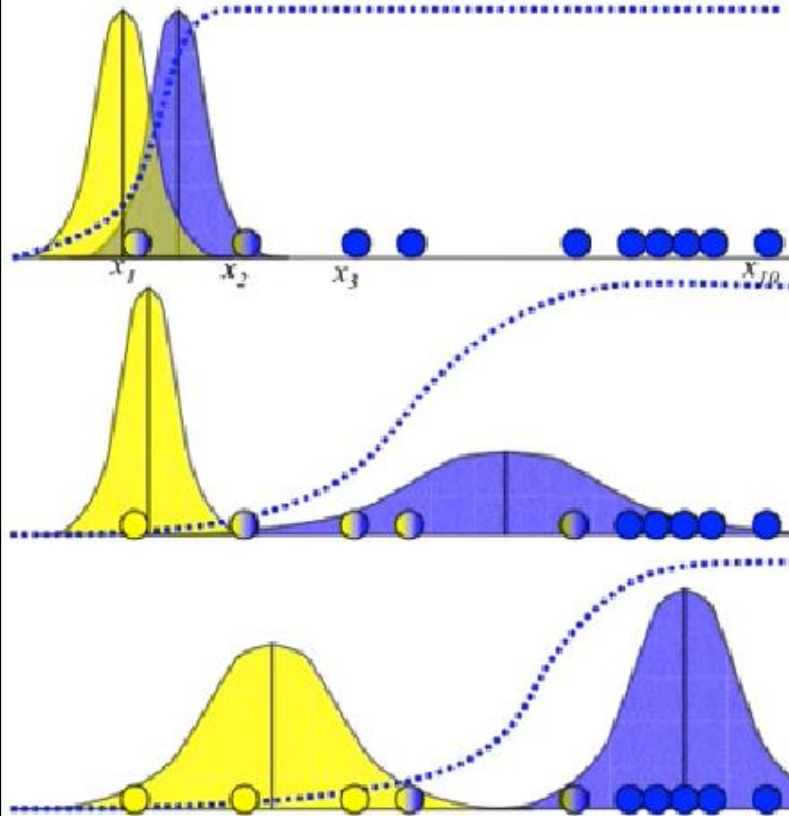
M step: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} its expected value $E[z_{ij}]$ calculated above. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

ಡಾ|| ಅ್ಯಾಗರಾಜು ಬಿ.ಎಸ್

GMM :

EM:



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

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

$$a_i = P(a | x_i) = 1 - b_i$$

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

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

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

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

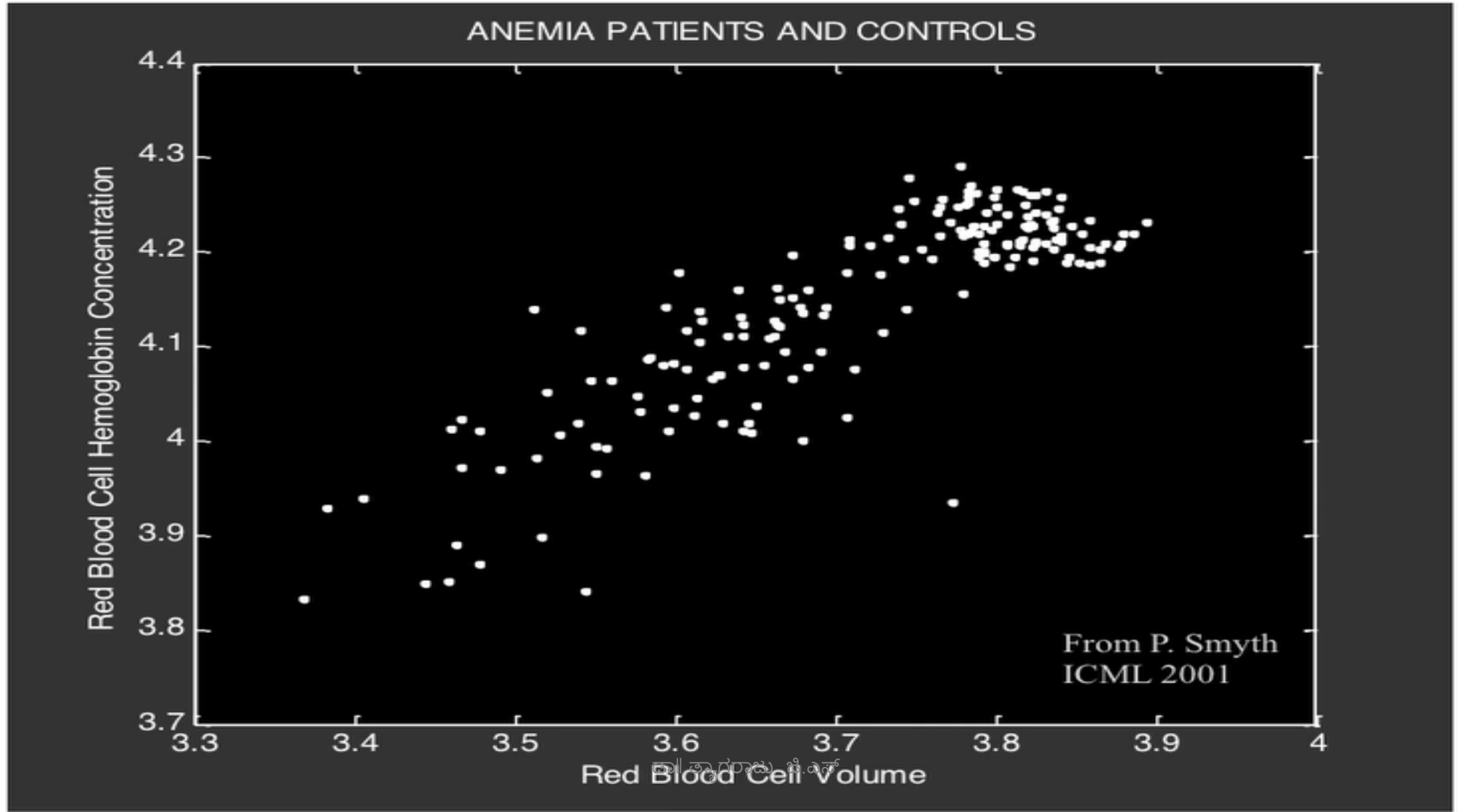
could also estimate priors:

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

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

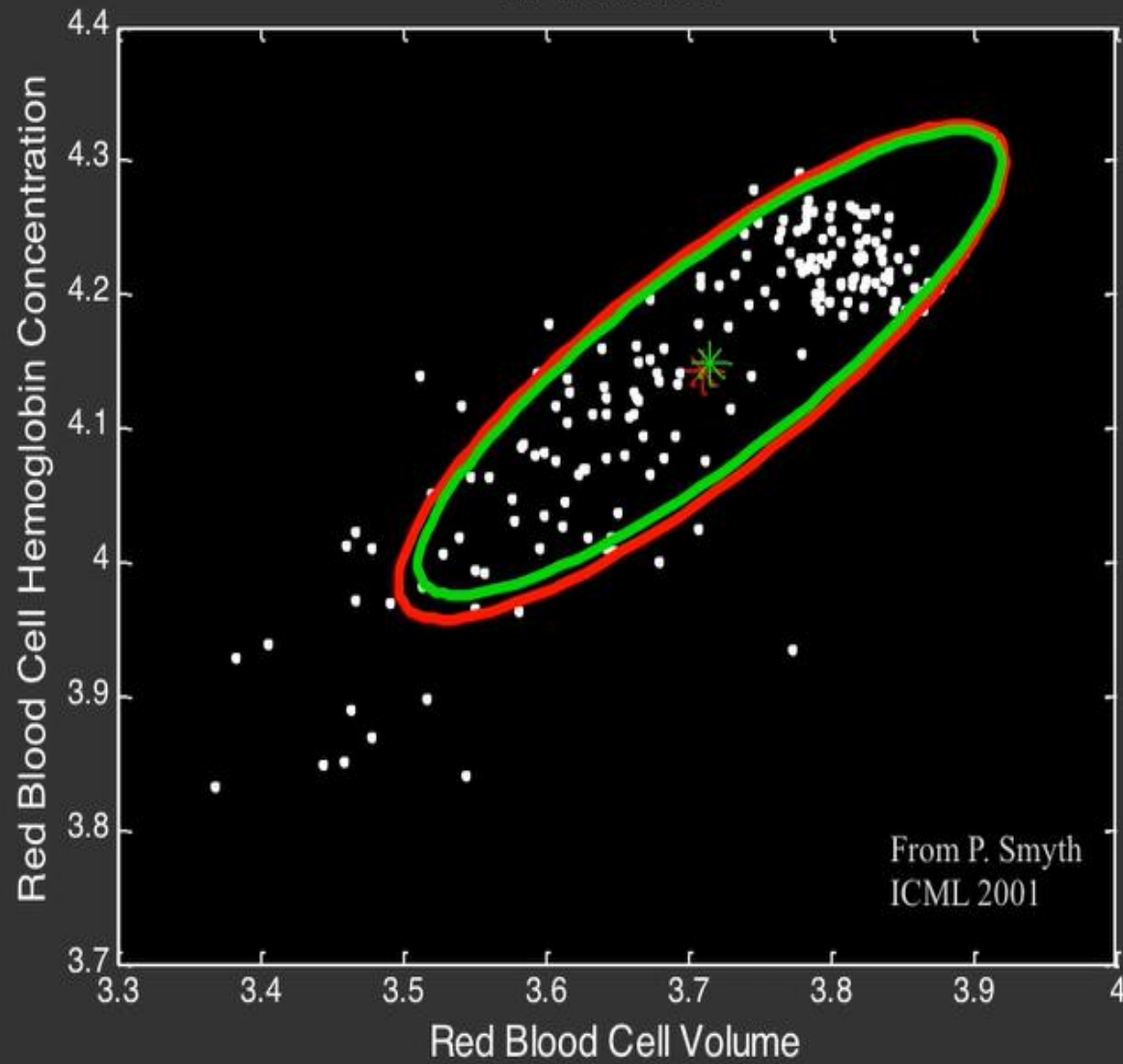
Copyright © 2014 Victor Lavrenko

GMM : Example2

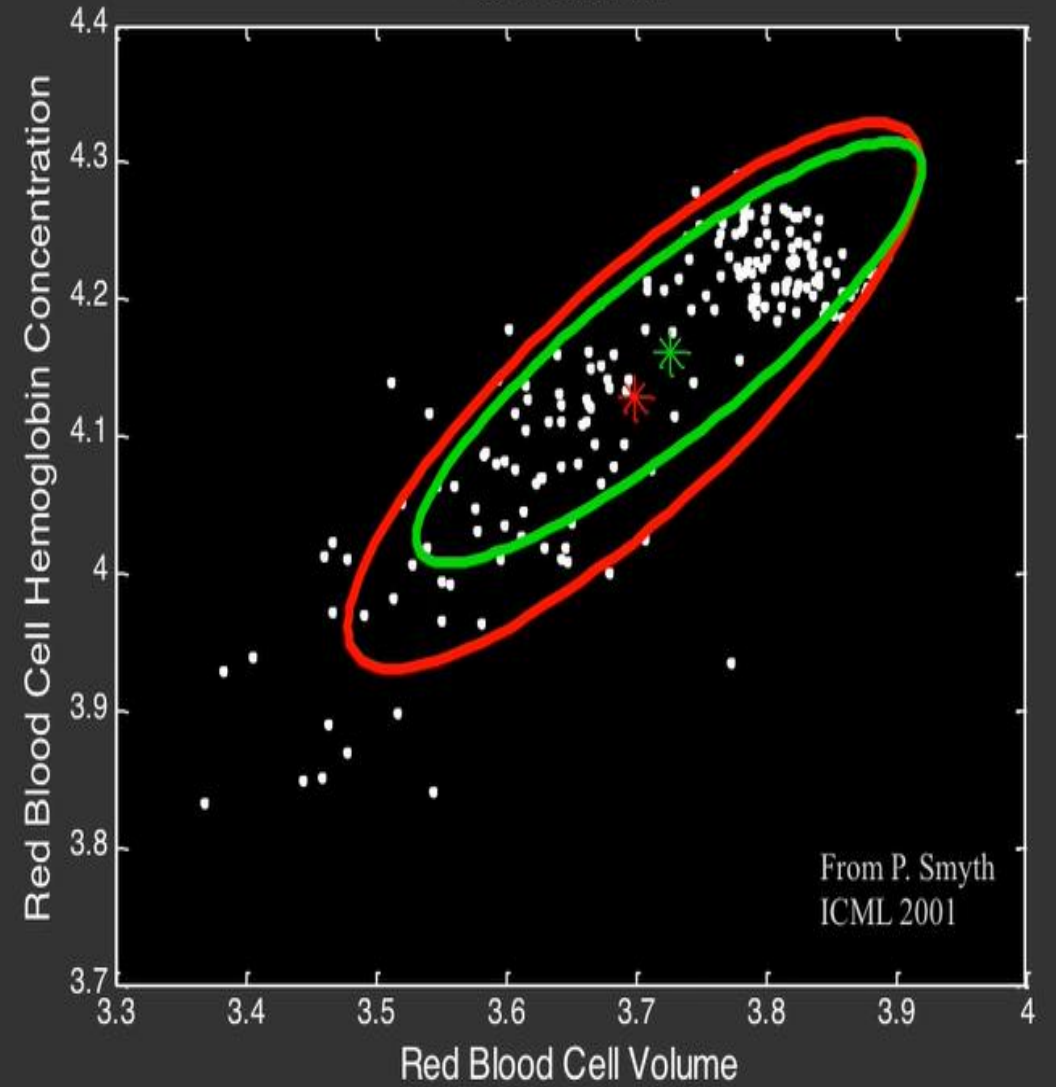


GMM : Example2

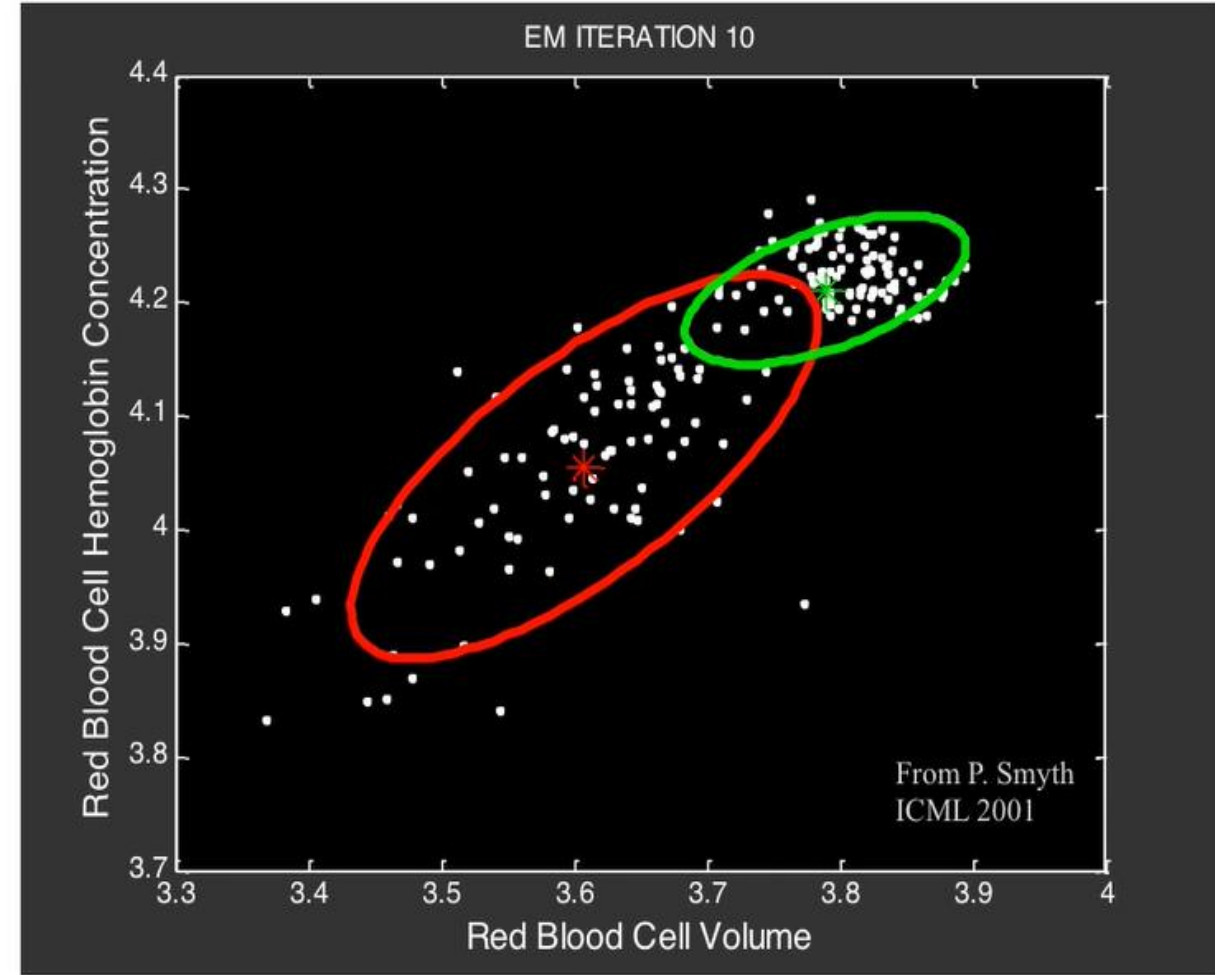
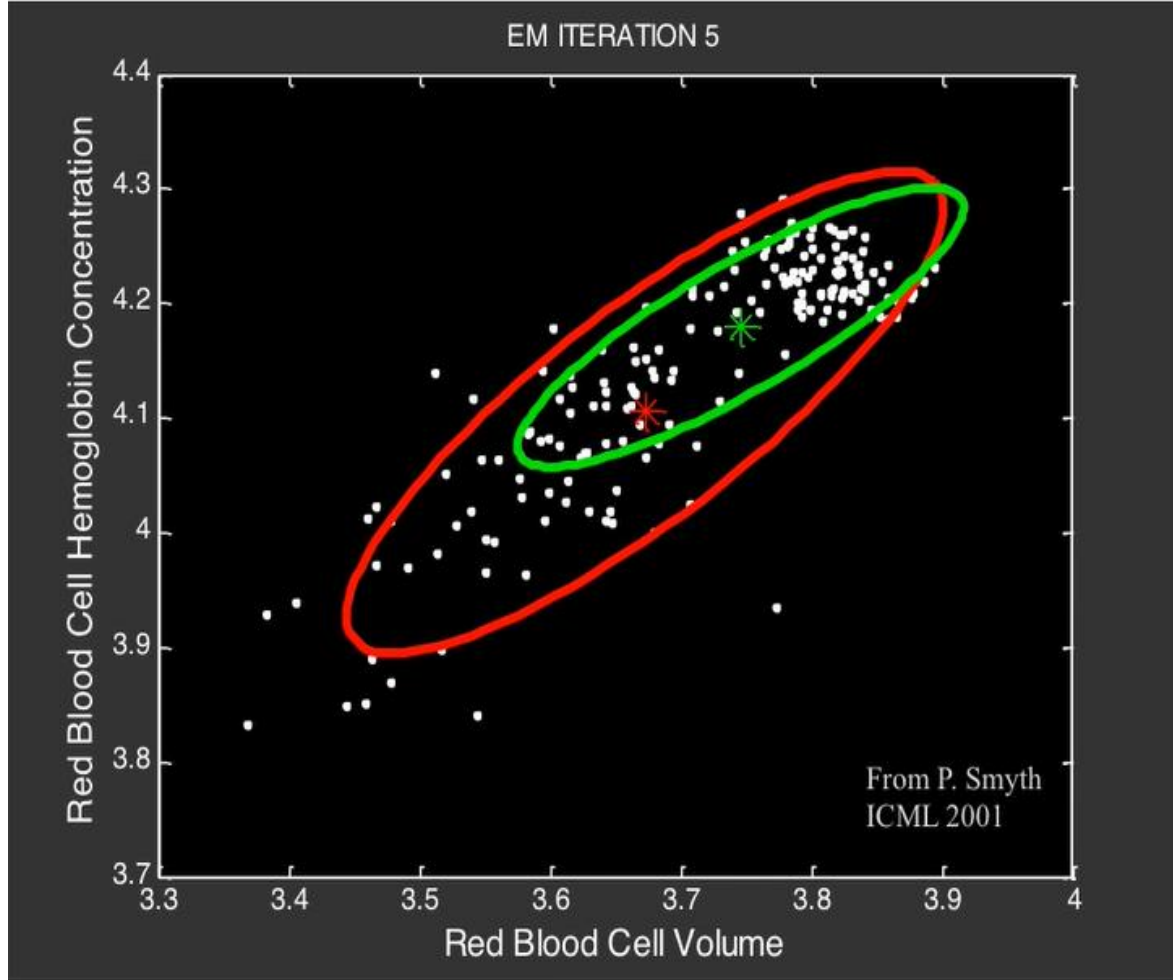
EM ITERATION 1



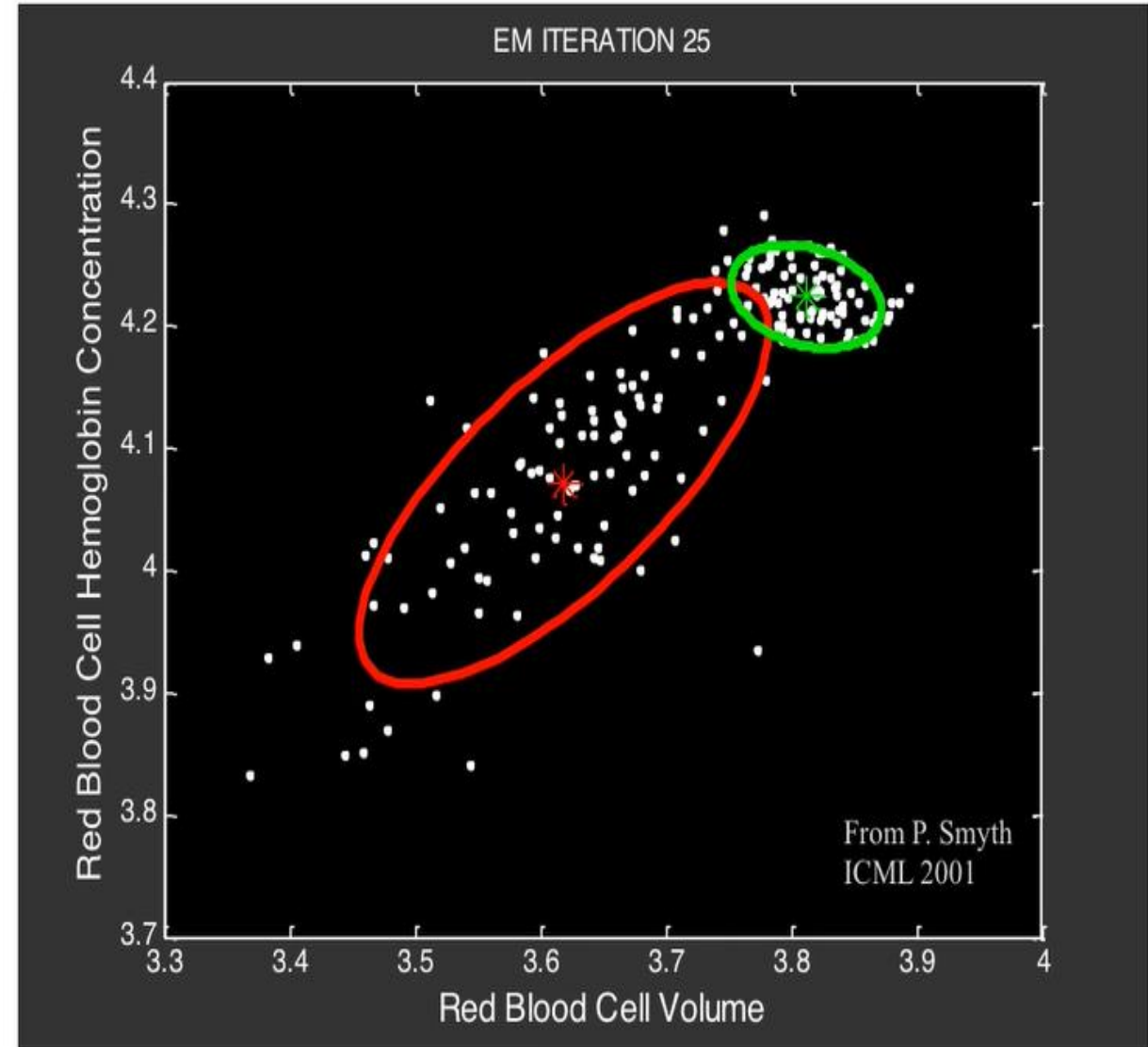
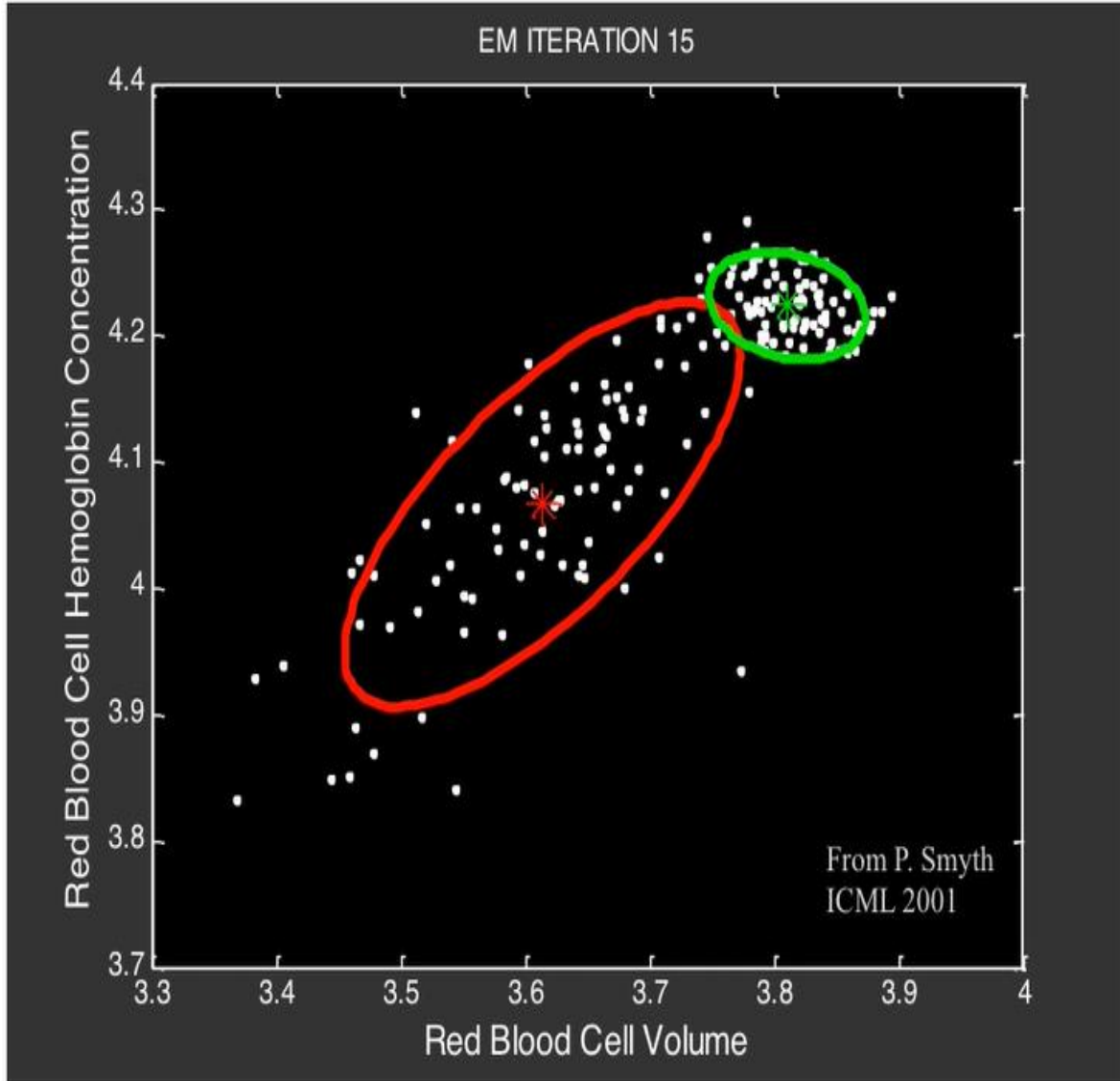
EM ITERATION 3



GMM : Example2



GMM : Example2



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

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

Distance functions

Euclidean

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

Manhattan

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

Minkowski

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

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

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2}$$

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

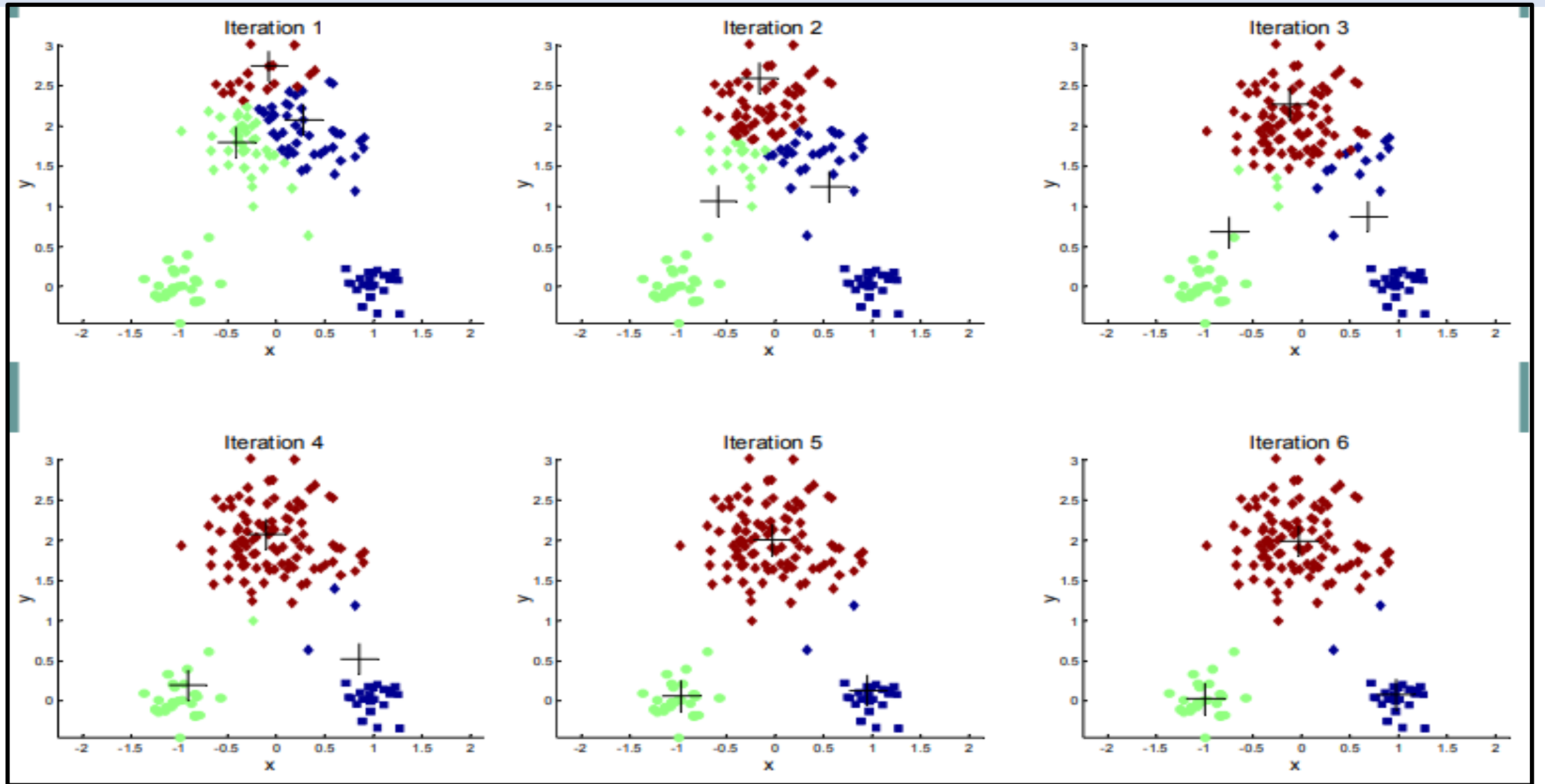
We use the following equation to calculate the n dimensional centroid point amid k n-dimensional points

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

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

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

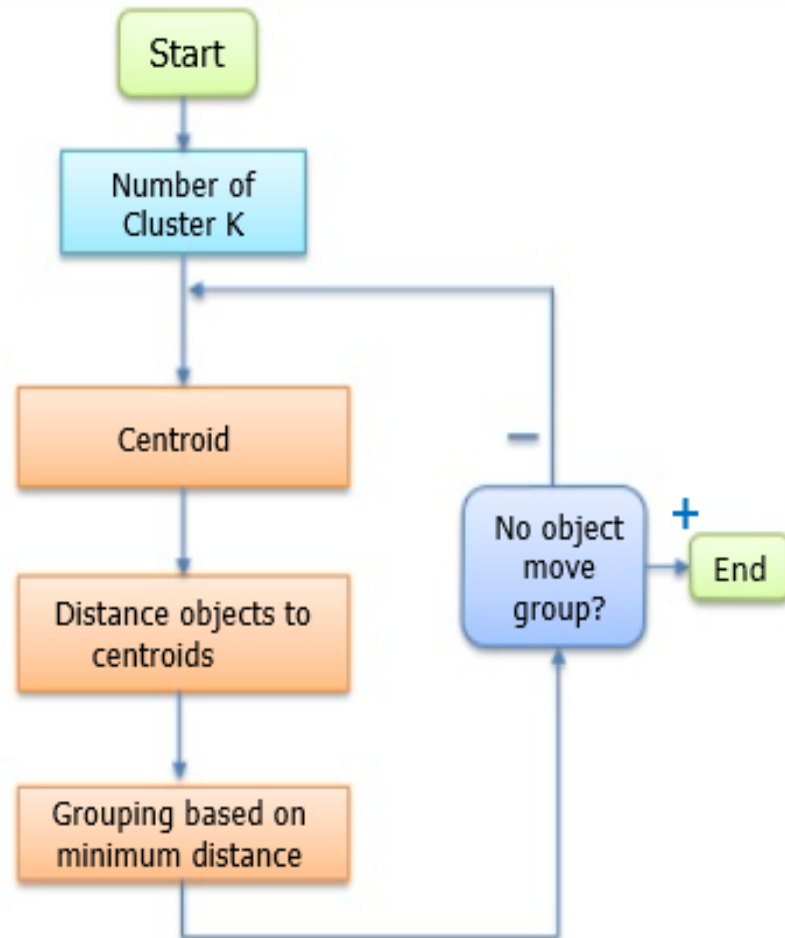
Examples of K Means



How the K-Mean Clustering algorithm works?

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

Process Flow of K-means

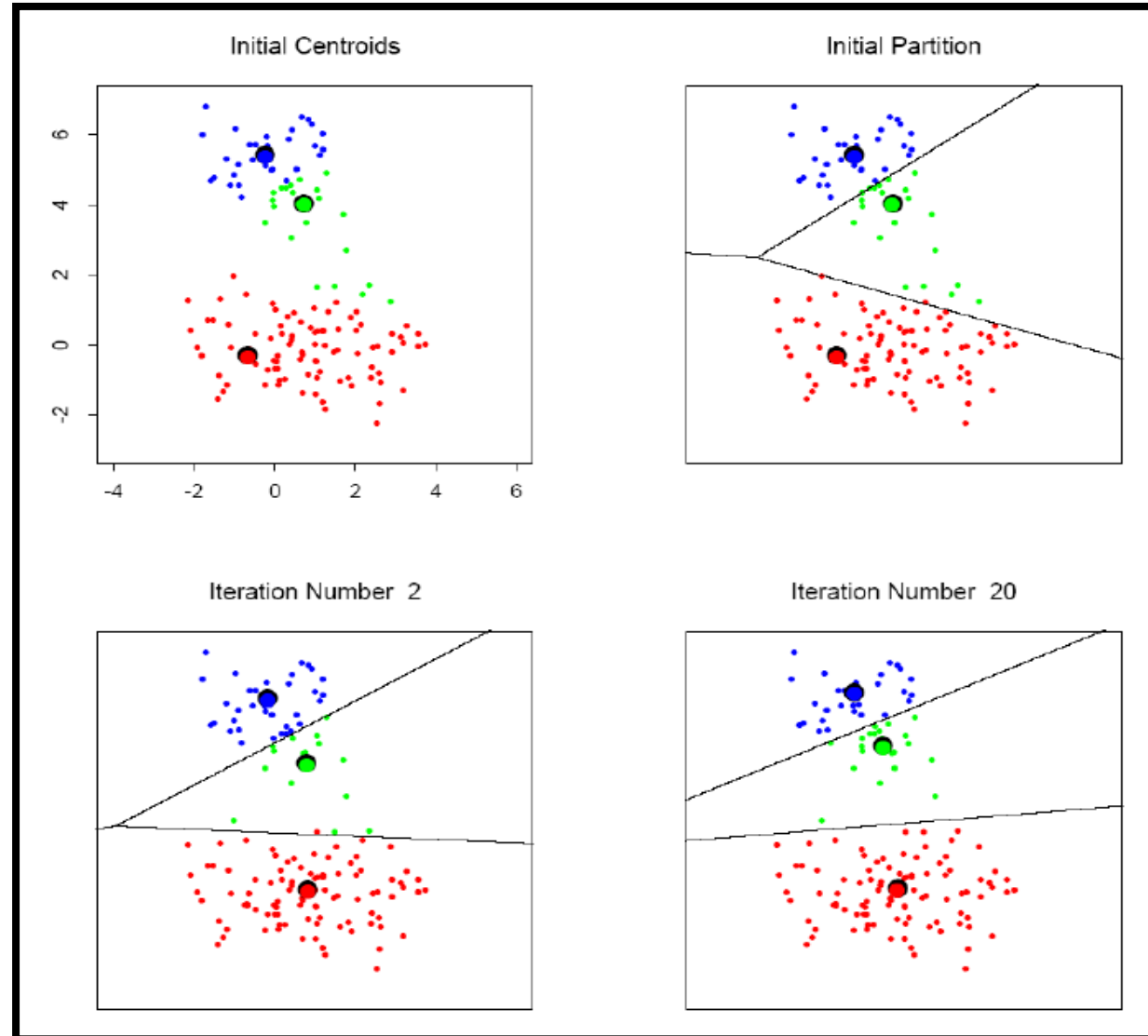


Iterate until *stable* (cluster centers converge):

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)

www.edureka.io/data-science

K-means clustering example



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.

Source Code