MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Autonomous Institution – UGC, Govt. of India

Department of COMPUTATIONAL INTELLIGENCE

B.TECH

(CSE-AI&ML, AI&ML)

B.TECH

(R-22 Regulation)

(III YEAR – I SEM)

2024-25

MACHINE LEARNING

(R22A6602)

LECTURE NOTES

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12(B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad–500100, Telangana State, India

Department of COMPUTATIONAL INTELLIGENCE

CSE-AIML, AIML

MACHINE LEARNING

(R22A6602)

LECTURE NOTES

Prepared by

M.N.S.Gangadhar

Assistant Professor

Department of Computational Intelligence

CSE-Artificial Intelligence & Machine Learning, Artificial Intelligence & Data Science

Vision

To be a premier centre for academic excellence and research through innovative interdisciplinary collaborations and making significant contributions to the community, organizations, and society as a whole.

Mission

- To impart cutting-edge Artificial Intelligence technology in accordance with industrynorms.
- To instill in students a desire to conduct research in order to tackle challenging technical problems for industry.
- To develop effective graduates who are responsible for their professional growth, leadership qualities and are committed to lifelong learning.

QUALITY POLICY

- To provide sophisticated technical infrastructure and to inspire students to reach their full potential.
- To provide students with a solid academic and research environment for a comprehensive learning experience.
- To provide research development, consulting, testing, and customized training to satisfy specific industrial demands, thereby encouraging self-employment and entrepreneurship among students.

For more information: www.mrcet.ac.in

3 - / - / - 3

(R20A0518) Machine Learning

Course Objectives:

1. Recognize the basic terminology and fundamental concepts of machine learning.
2. Understand the concepts of Supervised Learning models with a focus on recent advancements.
3. Relate the Concepts of Neural Networks Models of supervised Learning
4. Discover Unsupervised learning paradigms of machine learning
5. Understand the concepts of Reinforcement learning and Ensemble methods.

Expected Course Outcome:

1. Explain the concepts and able to prepare the dataset for different Machine learning models.
2. Identify and Apply appropriate Supervised Learning models.
3. Design Neural Network models for the given data.
4. Perform Evaluation of Machine Learning algorithms and Model Selection.
5. Devise un-supervised and Reinforcement learning models

UNIT – I

Introduction: Introduction to Machine learning, Supervised learning, Unsupervised learning, Reinforcement learning. Deep learning.
Feature Selection: Filter, Wrapper , Embedded methods.
Feature Normalization:- min-max normalization, z-score normalization, and constant factor normalization
Introduction to Dimensionality Reduction : Principal Component Analysis(PCA), Linear Discriminant Analysis(LDA)

UNIT-II

Supervised Learning – I (Regression/Classification)

Regression models: Simple Linear Regression, multiple linear Regression. Cost Function, Gradient Descent, Performance Metrics: Mean Absolute Error(MAE),Mean Squared Error(MSE) R-Squared error, Adjusted R Square.

Classification models: Decision Trees-ID3,CART, Naive Bayes, K-Nearest-Neighbours (KNN), Logistic Regression, Multinomial Logistic Regression Support Vector Machines (SVM) - Nonlinearity and Kernel Methods

UNIT – III

Supervised Learning – II (Neural Networks)

Neural Network Representation – Problems – Perceptrons , Activation Functions, Artificial Neural Networks (ANN) , Back Propagation Algorithm.

Convolutional Neural Networks - Convolution and Pooling layers, , Recurrent Neural Networks (RNN).

Classification Metrics: Confusion matrix, Precision, Recall, Accuracy, F-Score, ROC curves

UNIT - IV

Model Validation in Classification : Cross Validation - Holdout Method, K-Fold, Stratified K-Fold, Leave-One-Out Cross Validation. Bias-Variance tradeoff, Regularization , Overfitting, Underfitting. **Ensemble Methods:** Boosting, Bagging, Random Forest.

UNIT – V

Unsupervised Learning : Clustering-K-means, K-Modes, K-Prototypes, Gaussian Mixture Models, Expectation-Maximization.

Reinforcement Learning: Exploration and exploitation trade-offs, non-associative learning, Markov decision processes, Q-learning.

Text Book(s)

1. Machine Learning – Tom M. Mitchell, -MGH
2. Kevin Murphy, Machine Learning: A Probabilistic Perspective, MIT Press,2012
3. R. S. Sutton and A. G. Barto. Reinforcement Learning - An Introduction. MIT Press.1998

Reference Books

1. Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning, Springer2009
2. Christopher Bishop, Pattern Recognition and Machine Learning, Springer,2007.
3. Machine Learning Yearning, AndrewNg.
4. Data Mining–Concepts and Techniques -Jiawei Han and Micheline Kamber,Morgan Kaufmann

INDEX

UNIT-1

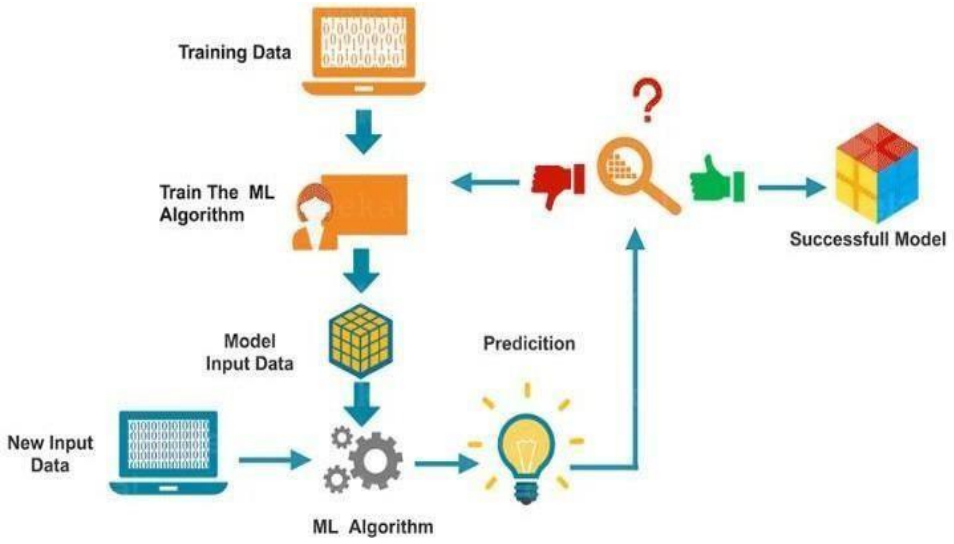
Introduction: Introduction to Machine learning , Supervised learning, Unsupervised learning, Reinforcementlearning. Deep learning.
Feature Selection: Filter, Wrapper , Embedded methods.
Feature Normalization:- min-max normalization, z-score normalization, and constant factor normalization
Introduction to Dimensionality Reduction : Principal Component Analysis(PCA), Linear Discriminant Analysis(LDA)

What is Machine Learning?

Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed. So instead of you writing the code, what you do is you feed data to the generic algorithm, and the algorithm/ machine builds the logic based on the given data.

How does Machine Learning Work?

Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the model.The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented raining data set



Types of Machine Learning

Machine learning is sub-categorized to three types:

- **Supervised Learning – Train Me!**
- **Unsupervised Learning – I am self sufficient in learning**
- **Reinforcement Learning – My life My rules! (Hit & Trial)**

Supervised Learning is the one, where you can consider the learning is guided by a teacher. We have a dataset which acts as a teacher and its role is to train the model or the machine. Once the model gets trained it can start making a prediction or decision when new data is given to it.

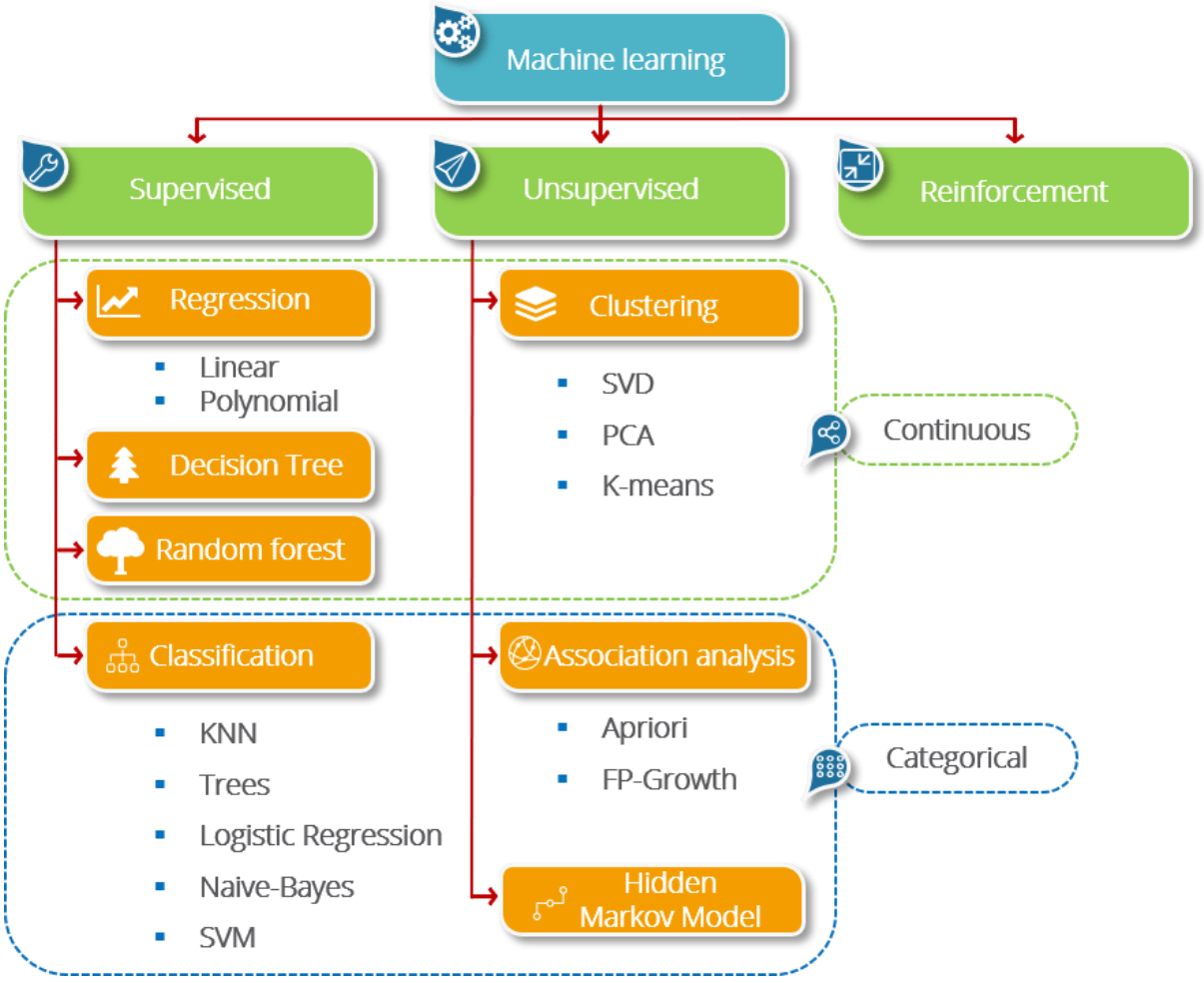
What is Unsupervised Learning?

The model learns through observation and finds structures in the data. Once the model is given a dataset, it automatically finds patterns and relationships in the dataset by creating clusters in it. What it cannot do is add labels to the cluster, like it cannot say this a group of apples or mangoes, but it will separate all the apples from mangoes.

Suppose we presented images of apples, bananas and mangoes to the model, so what it does, based on some patternsand relationships it creates clusters and divides the dataset into those clusters. Now if a new data is fed to the model, it adds it to one of the created clusters.

What is Reinforcement Learning?

It is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, andon the basis of the positive reward points gained the model trains itself. And again once trained it gets ready to predict the new data presented to it.



Classification of Machine Learning Algorithms Machine Learning algorithms can be classified into:

1. Supervised Algorithms – Linear Regression, Logistic Regression, Support Vector Machine (SVM), DecisionTrees, Random Forest
2. Unsupervised Algorithms – K Means Clustering.
3. Reinforcement Algorithm

1. Supervised Machine Learning Algorithms

In this type of algorithm, the data set on which the machine is trained consists of labelled data or simply said, consists both the input parameters as well as the required output. For example, classifying whether a person is a male or a female. Here male and female will be our labels and our training dataset will already be classified into the

given labels based on certain parameters through which the machine will learn these features and patterns andclassify some new input data based on the learning from this training data.

Supervised Learning Algorithms can be broadly divided into two types of algorithms, Classification and Regression.Classification Algorithms

Just as the name suggests, these algorithms are used to classify data into predefined classes or labels. Regression Algorithms

These algorithms are used to determine the mathematical relationship between two or more variables and the level of dependency between variables. These can be used for predicting an output based on the interdependency of two or more variables. For example, an increase in the price of a product will decrease its consumption, which means, in this case, the amount of consumption will depend on the price of the product. Here, the amount of consumption will be called as the dependent variable and price of the product will be called the independent variable. The level of dependency of the amount of consumption on the price of a product will help us predict the future value of the amount of consumption based on the change in prices of the product.

We have two types of regression algorithms: Linear Regression and Logistic Regression

(a) Linear Regression

Linear regression is used with continuously valued variables, like the previous example in which the price of the product and amount of consumption are continuous variables, which means that they can have an infinite number of possible values. Linear regression can also be represented as a graph known as scatter plot, where all the data points of the dependent and independent variables are plotted and a straight line is drawn through them such that the maximum number of points will lie on the line or at a smaller distance from the line. This line – also called the regression line, will then help us determine the relationship between the dependent and independent variables along with which the linear regression equation is formed.

(b) Logistic Regression

The difference between linear and logistic regression is that logistic regression is used with categorical dependent variables (eg: Yes/No, Male/Female, Sunny/Rainy/Cloudy, Red/Blue etc.), unlike the continuous valued variables used in linear regression. Logistic regression helps determine the probability of a certain variable to be in a certain group like whether it is night or day, or whether the colour is red or

blue etc. The graph of logistic regression consists of a non-linear sigmoid function which demonstrates the probabilities of the variables.

2. Unsupervised Machine Learning Algorithms

Unlike supervised learning algorithms, where we deal with labelled data for training, the training data will be unlabelled for Unsupervised Machine Learning Algorithms. The clustering of data into a specific group will be doneon the basis of the similarities between the variables. Some of the unsupervised machine learning algorithms are K- means clustering, neural networks.

Another machine learning concept which is extensively used in the field is Neural Networks..

3. Reinforcement Machine Learning Algorithms

Reinforcement Learning is a type of Machine Learning in which the machine is required to determine the ideal behaviour within a specific context, in order to maximize its rewards. It works on the rewards and punishment principle which means that for any decision which a machine takes, it will be either be rewarded or punished due to which it will understand whether or not the decision was correct. This is how the machine will learn to take the correct decisions to maximize the reward in the long run.

For reinforcement algorithm, a machine can be adjusted and programmed to focus more on either the long-term rewards or the short-term rewards. When the machine is in a particular state and has to be the action for the next state in order to achieve the reward, this process is called the Markov Decision Process.

What is Normalization in Machine Learning?

Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges.

Although **Normalization** is no mandate for all datasets available in machine learning, it is used whenever the attributes of the dataset have different ranges. It helps to enhance the performance and reliability of a machine learning model. In this article, we will discuss in brief various Normalization techniques in machine learning, why it is used, examples of normalization in an ML model, and much more. So, let's start with the definition of Normalization in Machine Learning.

Mathematically, we can calculate normalization with the below formula:

Xn = (X - Xminimum) / (Xmaximum - Xminimum)

- Xn = Value of Normalization
- Xmaximum = Maximum value of a feature
- Xminimum = Minimum value of a feature

Example: Let's assume we have a model dataset having maximum and minimum values of feature as mentioned above. To normalize the machine learning model, values are shifted and rescaled so their range can vary between 0 and 1. This technique is also known as **Min-Max scaling**. In this scaling

technique, we will change the feature values as follows:

Case1- If the value of X is minimum, the value of Numerator will be 0; hence Normalization will also be 0.
Xn = (X - Xminimum) / (Xmaximum - Xminimum)

Put X =Xminimum in above formula, we get;

Xn = Xminimum- Xminimum/ (Xmaximum - Xminimum)Xn = 0
Case2- If the value of X is maximum, then the value of the numerator is equal to the denominator; henceNormalization will be 1.

Xn = (X - Xminimum) / (Xmaximum - Xminimum)Put X =Xmaximum in above formula, we get;
Xn = Xmaximum - Xminimum/ (Xmaximum - Xminimum)

What is Dimensionality Reduction?
In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Why is Dimensionality Reduction important in Machine Learning and Predictive Modeling?
An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc. However, some of these features may overlap. In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems. A 3-D classification problem can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line. The below figure illustrates this concept, where a 3-D feature space is split into two 1-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further

Components of Dimensionality Reduction
There are two components of dimensionality reduction:

- **Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smallersubset which can be used to model the problem. It usually involves three ways:

1. Filter
2. Wrapper
3. Embedded

- **Feature extraction:** This reduces the data in a high dimensional space to a lower dimension space, i.e. a spacewith lesser no. of dimensions.

Methods of Dimensionality Reduction

The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used. The prime linearmethod, cled Principal Component Analysis, or PCA, is discussed

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.

It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of theoriginal data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

Advantages of Dimensionality Reduction

- It helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

- It may lead to some amount of data loss.
- PCA tends to find linear correlations between variables, which is sometimes undesirable.
- PCA fails in cases where mean and covariance are not enough to define datasets.
- We may not know how many principal components to keep- in practice, some thumb rules are applied.
- Explain fold
- based filtering

3. Kernel Principal Component Analysis

There are a lot of machine learning problems which a nonlinear, and the use of nonlinear feature mappings can help to produce new features which make prediction problems linear. In this section we will discuss the following idea: transformation of the dataset to a new higher-dimensional (in some cases infinite-dimensional) feature space and theuse of PCA in that space in order to produce uncorrelated features. Such a method is called **Kernel Principal Component Analysis** or **KPCA**.

Let us denote a covariance matrix in a new feature space as

$$\varphi\colon \mathbb{R}^D \to \mathbb{F},$$

$$\mathbf{C} = \tfrac{1}{N}\mathbf{\Phi}^\intercal\mathbf{\Phi},$$

where $\mathbf{\Phi} = \{\phi(\mathbf{x}_n)\}_{n=1}^{N}$. Will consider that the dimensionality of the feature space equals to . Eigendecompsition of is given by

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i,\ \forall i = 1, \dots M.$$

By the definition of

And therefore $\forall i = 1, \dots M$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^\intercal \mathbf{v}_i\} = \lambda_i \mathbf{v}_i.$$

It is obviously to see, that is a linear combination of $\phi(\mathbf{x}_n)$and thus can be written as

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{i_n}\phi(\mathbf{x}_n).$$

Substituting it to the equation above and writing it in a matrix notation, we get

$$\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i,$$

where is a Gram matrix in , $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^\intercal\phi(\mathbf{x}_j)$ andare column-vectors with elements Eigenvectorsof should be orthonormal, therefore, we get the following:

$$N\lambda_i \mathbf{a}_i^\intercal \mathbf{a}_i = 1.$$

Having eigenvectors of , we can get the projection of an item $\mathbf{x} \in \mathbb{R}^D$ on -th eigenvector:

$$z_i(\mathbf{x}) = \sum_{n=1}^{N} a_{i_n}\phi(\mathbf{x})^\intercal\phi(\mathbf{x}_n).$$

So far, we have assumed that the mapping $\varphi(\cdot)$ is known. From the equations above, we can see, that only a thing that we need for the data transformation is the eigendecomposition of a Gram matrix . Dot products, which are its elements can be defined without any definition of $\varphi(\cdot)$. The function defining such dot products in some Hilbert space is called kernel. Kernels are satisfied by the Mercer's theorem. There are many different types of kernels, there are several popular:

1. Linear: ;
2. Gaussian: ;
3. Polynomial: .

Using a kernel function we can write new equation for a projection of some data item onto -th eigenvector:

$$z_i(\mathbf{x}) = \sum_{n=1}^{N} a_{i_n} k(\mathbf{x}, \mathbf{x}_n)$$

.

So far, we have assumed that the columns of have zero mean. Using

,

and substituting it to the equation for , we get

$$\mathbf{K_c} = \mathbf{K} - \mathbf{1_N K} - \mathbf{K 1_N} + \mathbf{1_N K 1_N},$$

where is a matrix , where each element equals to $\frac{1}{N}$.

Summary: Now we are ready to write the whole sequence of steps to perform KPCA:

1. Calculate .
2. Calculate $\mathbf{K_c}$.
3. Find the eigenvectors of corresponding to nonzero eigenvalues and normalize them: $\mathbf{a_i} = \frac{1}{\sqrt{\lambda_i N}} \mathbf{a_i}$.
4. Sort found eigenvectors in the descending order of coresponding eigenvalues.
5. Perform projections onto the given subset of eigenvectors.

The method described above requires to define the number of components, the kernel and its parameters. It shouldbe noted, that the number of nonlinear principal components in the general case is infinite, but since we are computing the eigenvectors of a matrix , at maximum we can calculate nonlinear principal components.

UNIT-II

Supervised Learning – I (Regression/Classification)

Regression models: Simple Linear Regression, multiple linear Regression. Cost Function, Gradient Descent, Performance Metrics: Mean Absolute Error(MAE),Mean Squared Error(MSE) R-Squared error, Adjusted R Square.

Classification models: Decision Trees-ID3,CART, Naive Bayes, K-Nearest-Neighbours (KNN), Logistic Regression, Multinomial Logistic Regression Support Vector Machines (SVM) - Nonlinearity and Kernel Methods

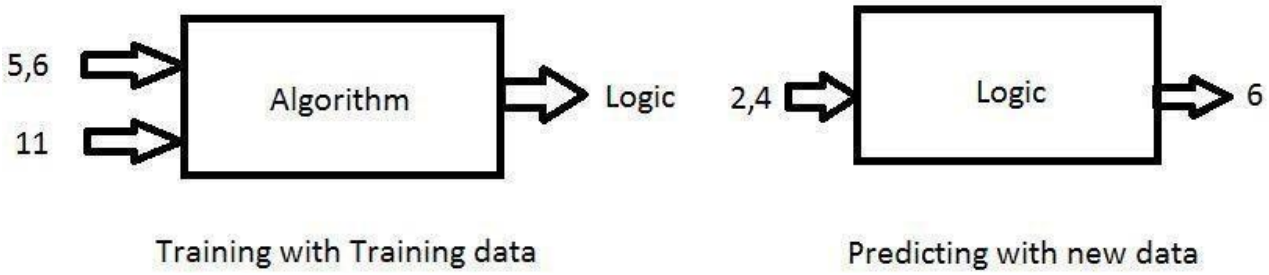
Supervised and unsupervised are mostly used by a lot machine learning engineers and data geeks. Reinforcementlearning is really powerful and complex to apply for problems.

Supervised learning

as we know machine learning takes data as input. lets call this data **Training data**

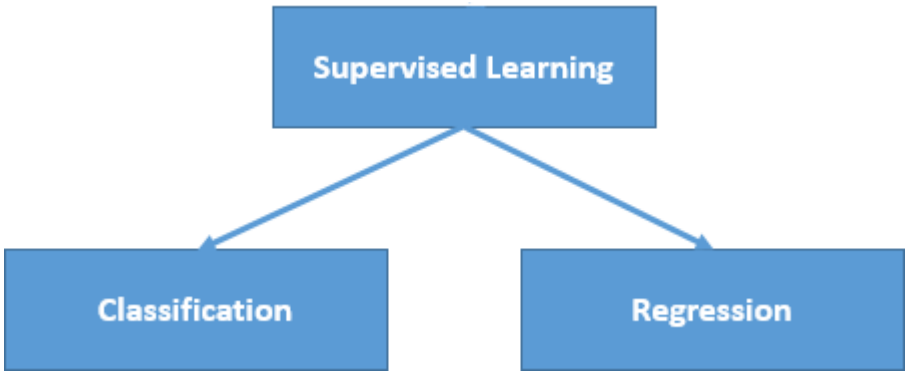
The training data includes both *Inputs* and L*abels(Targets)*

what are Inputs and Labels(Targets)?? for example addition of two numbers a=5,b=6 result =11, *Inputs* are 5,6and *Target* is 11



We first train the model with the lots of training data(inputs&targets)then with new data and the logic we got before we predict the output
(Note :We don't get exact 6 as answer we may get value which is close to 6 based on training data and algorithm)

This process is called *Supervised Learning* which is really fast and accurate.



Types of Supervised learning

Regression: This is a type of problem where we need to predict the *continuous-response* value (ex : above we predictnumber which can vary from -infinity to +infinity)

Some examples are

- what is the price of house in a specific city?
- what is the value of the stock?
- how many total runs can be on board in a cricket game?etc… there are tons of things we can predict if we wish.

Classification: This is a type of problem where we predict the *categorical response* value where the data can beseparated into specific "**classes**" (ex: we predict one of the values in a set of values).

Some examples are :

- this mail is spam or not?
- will it rain today or not?
- is this picture a cat or not?

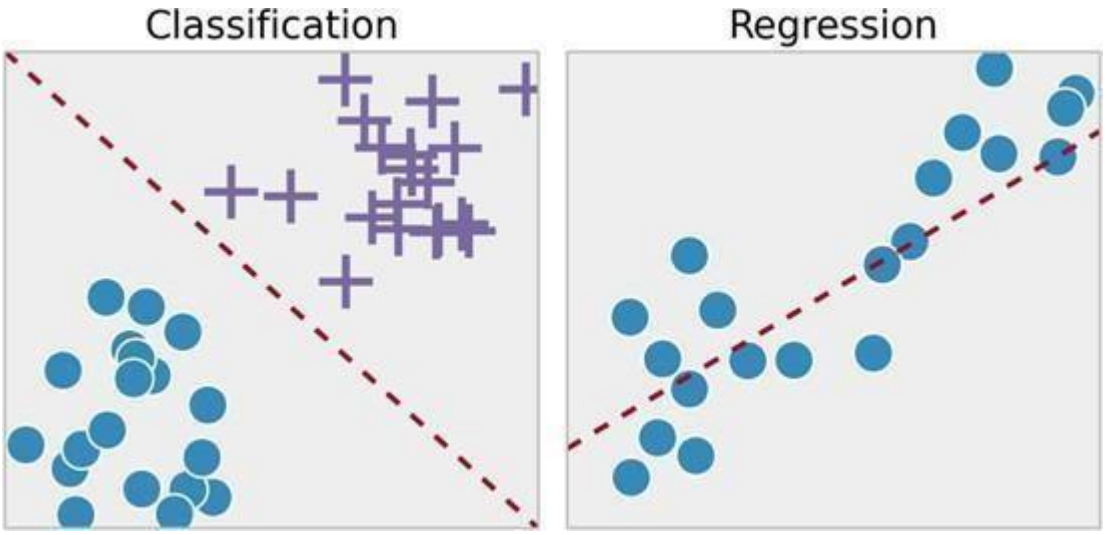
Basically 'Yes/No' type questions called **binary classification.**

Other examples are :

- this mail is spam or important or promotion?
- is this picture a cat or a dog or a tiger?

This type is called **multi-class classification.**

Here is the final picture



Classification separates the data, Regression fits the data

That's all for supervised learning.

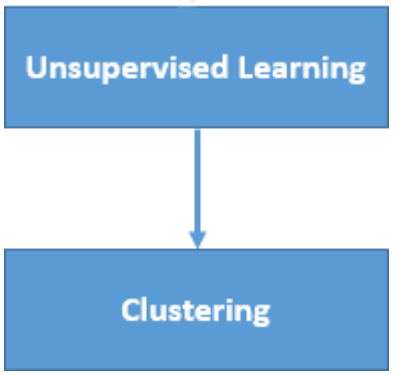
Unsupervised learning

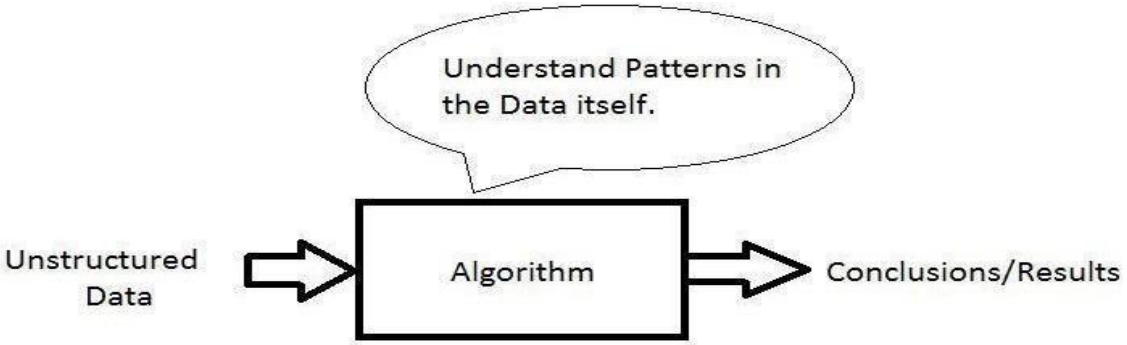
The training data does not include Targets here so we don't tell the system where to go , the system has to understanditself from the data we give.

Here training data is not structured (contains noisy data,unknown data and etc..)

Unsupervised process

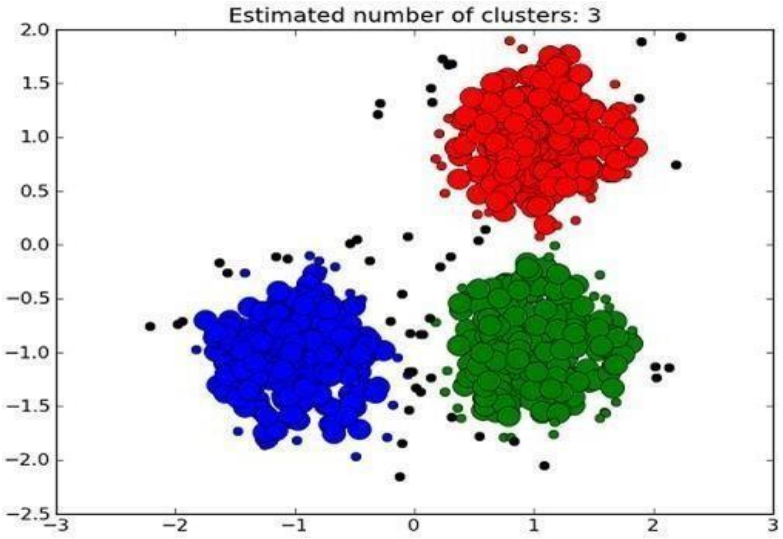
There are also different types for unsupervised learning like Clustering and anomaly detection (clustering is prettyfamous)





ands

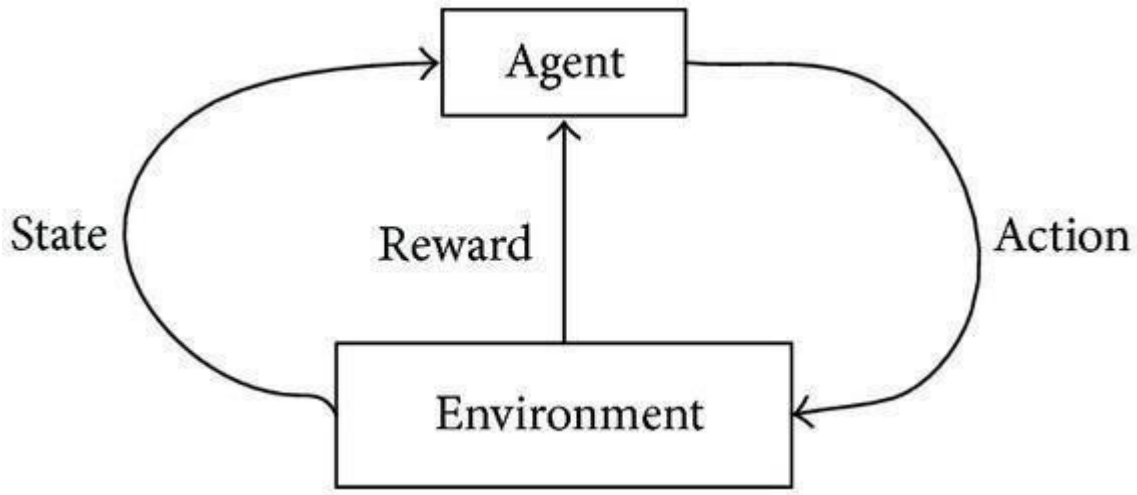
- given a set of images, cluster them into different objects



Clustering with 3 clusters

Unsupervised learning is bit difficult to implement and its not used as widely as supervised.

Reinforcement Learning is a type of *Machine Learning*, and thereby also a branch of *Artificial Intelligence*. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as thereinforcement signal.



There are many different algorithms that tackle this issue. As a matter of fact, Reinforcement Learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problemis known as a *Markov Decision Process*.

In order to produce intelligent programs (also called agents), reinforcement learning goes through the following steps:

1. Input state is observed by the agent.
2. Decision making function is used to make the agent perform an action.
3. After the action is performed, the agent receives reward or reinforcement from the environment.
4. The state-action pair information about the reward is stored.

List of Common Algorithms

- Q-Learning
- Temporal Difference (TD)

Deep Adversarial Networks

Use cases:

Some applications of the *reinforcement learning algorithms* are computer played board games (Chess, Go), robotic hands, and self-driving cars.

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.

We can understand the concept of regression analysis using the below example:

Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:

Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

Now, the company wants to do the advertisement of $200 in the year 2019 **and wants to know the prediction about the sales for this year**. So to solve such type of prediction problems in machine learning, we need regression analysis.

Regression is a supervised learning technique which helps in finding the correlation between variables and enablesus to predict the continuous output variable based on the one or more predictor variables. It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, ***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."*** The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Some examples of regression can be as:

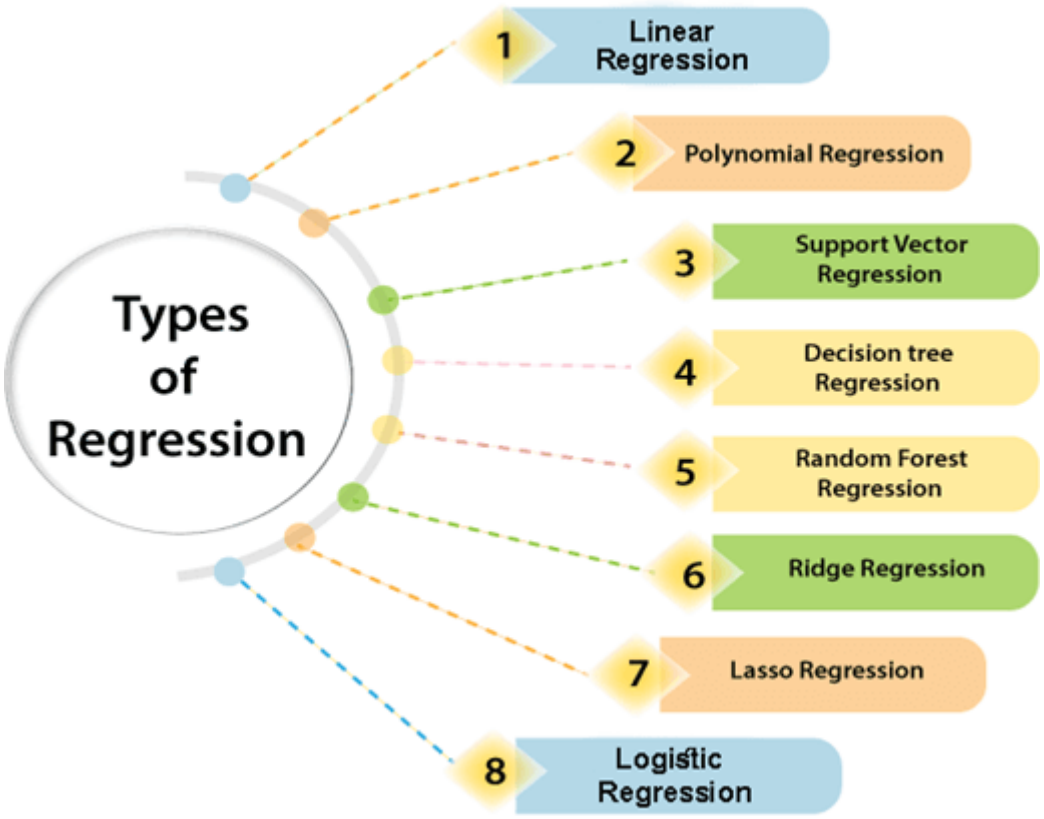
- Prediction of rain using temperature and other factors
- Determining Market trends

Prediction of road accidents due to rash driving.

Types of Regression

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

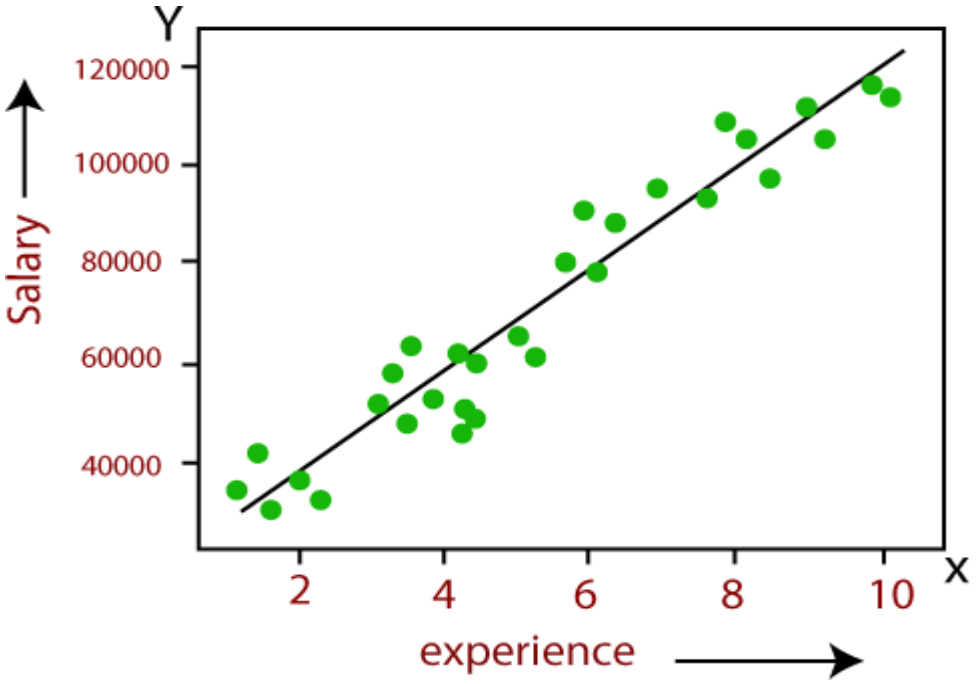
- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression:**



Linear Regression:

- Linear regression is a statistical regression method which is used for predictive analysis.
- It is one of the very simple and easy algorithms which works on regression and shows the relationship between thecontinuous variables.
- It is used for solving the regression problem in machine learning.

- Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable(Y-axis), hence called linear regression.
- If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there ismore than one input variable, then such linear regression is called **multiple linear regression**.
- The relationship between variables in the linear regression model can be explained using the below image. Here we arepredicting the salary of an employee on the basis of **the year of experience**.



Below is the mathematical equation for Linear regression:Y= aX+b

Here, Y=dependentvariables(target variables),X= Indepenen

Some popular applications of linear regression are:

- **Analyzing trends and sales estimates**
- **Salary forecasting**
- **Real estate prediction**
- **Arriving at ETAs in traffic.**
-

K Nearest Neighbors– Classification

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique

ALGORITHM

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor

Distance functions

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

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

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

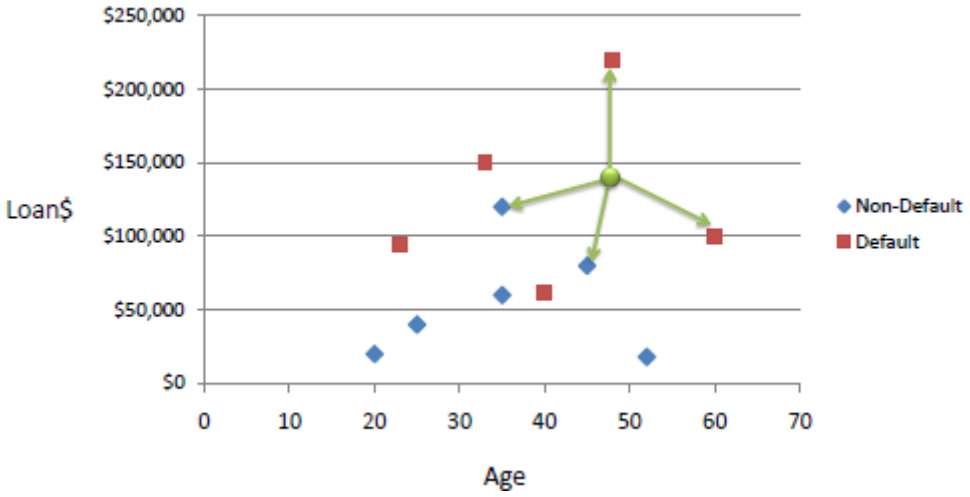
Hamming Distance

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

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1



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

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

what is a classifier?

A classifier is a machine learning model that is used to discriminate different objects based on certain features.

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of theclassifier is based on the Bayes theorem.

Bayes Theorem:

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

Using Bayes theorem, we can find the probability of **A** happening, given that **B**has occurred. Here, **B** is the evidenceand **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Example:

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as below.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe

that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make twoassumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to playgolf or not.

According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable **y** is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions. Variable **X** represent the parameters/features.

X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

Here x_1,x_2….x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remain static. Therefore, the denominator can be removed and a proportionality can be introduced.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

In our case, the class variable(**y**) has only two outcomes, yes or no. There could be cases where the classificationcould be multivariate. Therefore, we need to find the class **y** with maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Using the above function, we can obtain the class, given the predictors.Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use topredict the class variable take up only values yes or no, for example if a word occurs in the text or not.

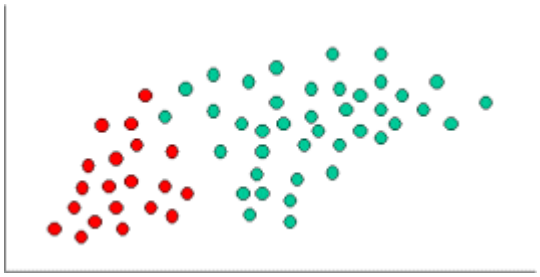
Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from agaussian distribution.

Conclusion:

Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They arefast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent. In most of the real life cases, the predictors are dependent, this hinders the performance of the classifier.

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.



To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief isknown as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.
Thus, we can write:

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). 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).

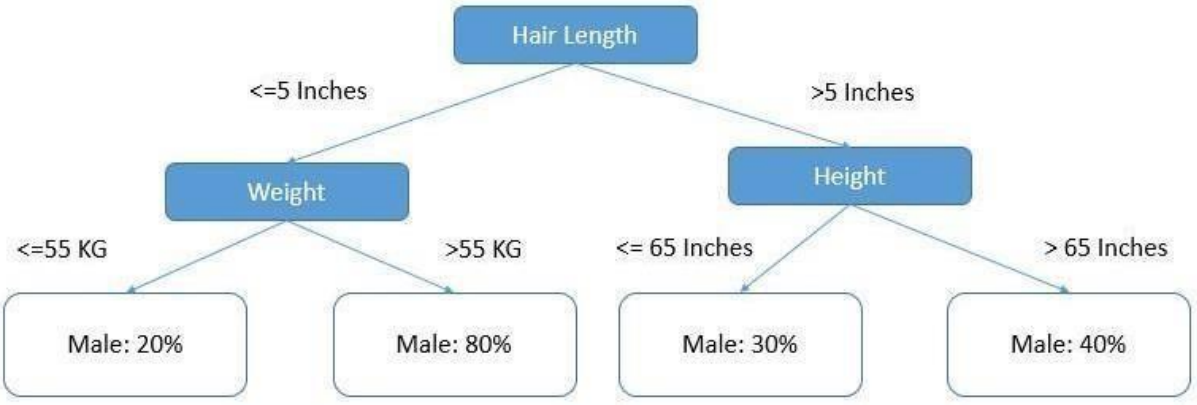
$$Posterior\ probability\ of\ X\ being\ GREEN \propto$$
$$Prior\ probability\ of\ GREEN \times Likelihood\ of\ X\ given\ GREEN$$
$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$
$$Posterior\ probability\ of\ X\ being\ RED \propto$$
$$Prior\ probability\ of\ RED \times Likelihood\ of\ X\ given\ RED$$
$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Assume that you are given a characteristic information of 10,000 people living in your town. You are asked to study them and come up with the algorithm which should be able to tell whether a new person coming to the town is male or a female.

Primarily you are given information about:

Skin colourHair lengthWeight Height
Based on the information you can divide the information in such a way that it somehow indicates the characteristicsof Males vs. Females.

Below is a hypothetical tree designed out of this data:



The tree shown above divides the data in such a way that we gain the maximum information, to understand the tree
– If a person's hair length is less than 5 Inches, weight greater than 55 KGs then there are 80% chances for thatperson being a Male.

If you are familiar with Predictive Modelling e.g., Logistic Regression, Random Forest etc. – You might be wondering what is the difference between a Logistic Model and Decision Tree! Because in both the algorithms we are trying to predict a categorical variable.

There are a few fundamental differences between both but ideally both the approaches should give you the same results. The best use of Decision Trees is when your solution requires a representation. For example, you are working for a Telecom Operator and building a solution using which a call center agent can take a

decision whether to pitch for an upsell or not!

There are very less chances that a call center executive will understand the Logistic Regression or the equations, but using a more visually appealing solution you might gain a better adoption from your call center team.

How does Decision Tree work?

There are multiple algorithms written to build a decision tree, which can be used according to the problem characteristics you are trying to solve. Few of the commonly used algorithms are listed below:

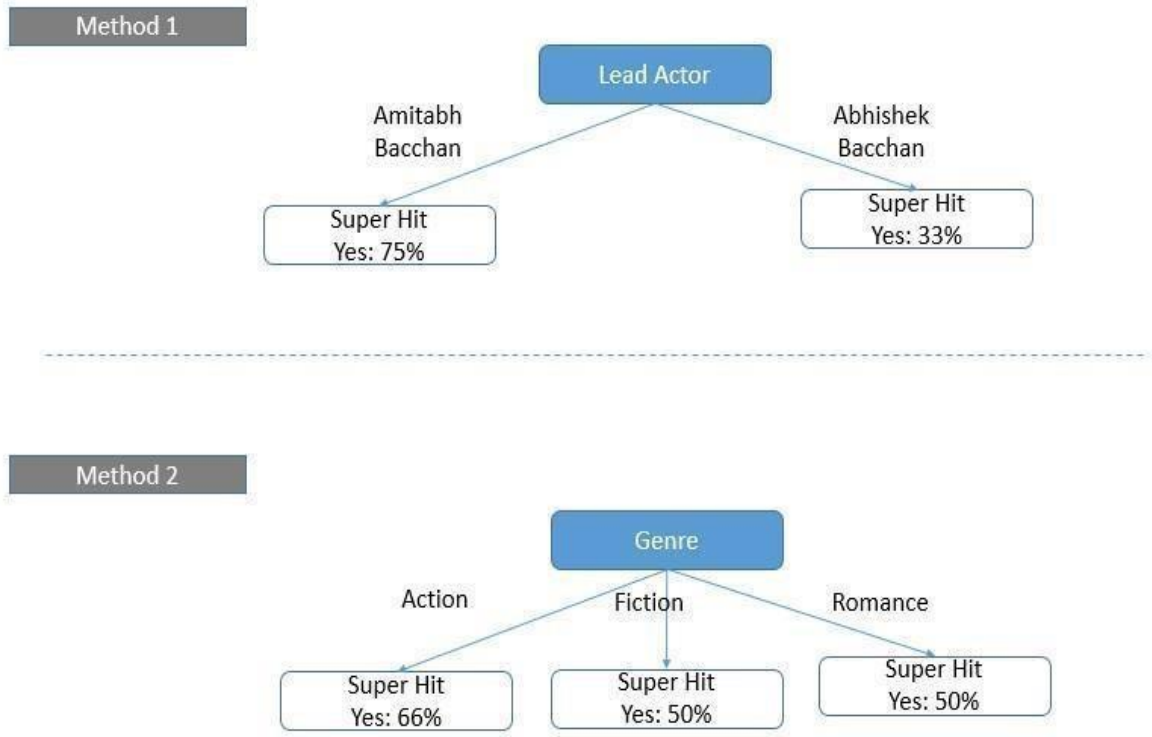
ID3 C4.5 CART
CHAID (CHi-squared Automatic Interaction Detector)MARS
Conditional Inference Trees

Though the methods are different for different decision tree building algorithms but all of them works on the principle of Greediness. Algorithms try to search for a variable which give the maximum information gain or divides the data in the most homogenous way.

For an example, consider the following hypothetical dataset which contains Lead Actor and Genre of a movie alongwith the success on box office:

Lead Actor	**Genre**	**Hit(Y/N)**
Amitabh Bacchan	Action	Yes
Amitabh Bacchan	Fiction	Yes
Amitabh Bacchan	Romance	No
Amitabh Bacchan	Action	Yes
Abhishek Bacchan	*Action*	*No*
Abhishek Bacchan	*Fiction*	*No*
Abhishek Bacchan	*Romance*	*Yes*

Let say, you want to identify the success of the movie but you can use only one variable – There are the followingtwo ways in which this can be done:



You can clearly observe that Method 1 (Based on lead actor) splits the data best while the second method (Based on Genre) have produced mixed results. Decision Tree algorithms do similar things when it comes to select variables.

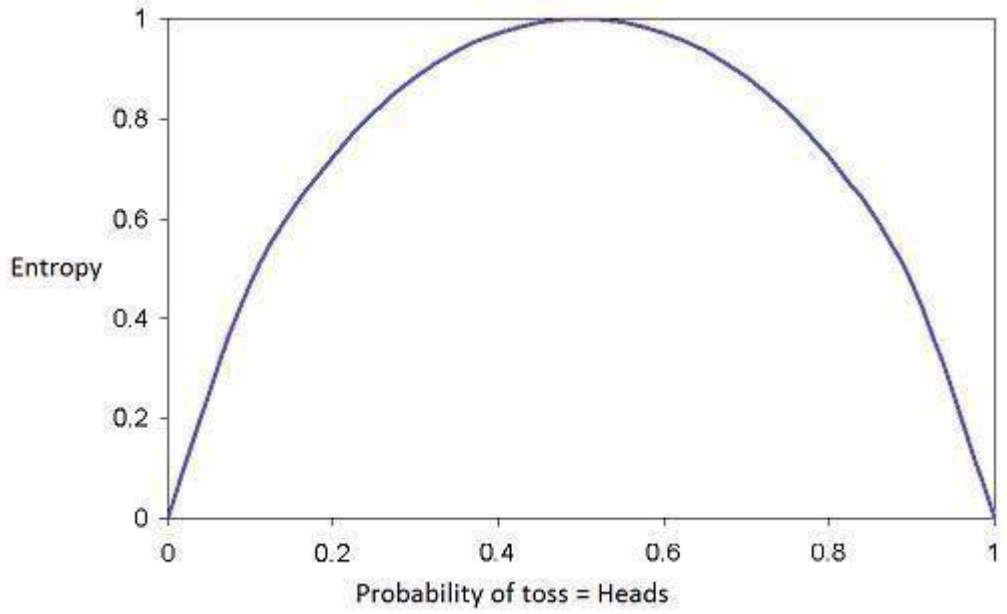
There are various metrics which decision trees use in order to find out the best split variables. We'll go through them one by one and try to understand, what do they mean?

Entropy & Information Gain

The word Entropy is borrowed from Thermodynamics which is a measure of variability or chaos or randomness. Shannon extended the thermodynamic entropy concept in 1948 and introduced it into statistical studies and suggested the following formula for statistical entropy:

$$H = -\sum_{j=1}^{n} P_j ln P_j$$

Where, H is the entropy in the system which is a measure of randomness.
Assuming you are rolling a fair coin and want to know the Entropy of the system. As per the formula given by Shann – Entropy would be equals to *-[0.5 ln(0.5) + 0.5 ln(0.5)]*.
Which is equal to -0.69; which is the maximum entropy which can occur in the system. In other words, there will be maximum randomness in our dataset if the probable outcomes have same probability of occurrence.



Graph shown above shows the variation of Entropy with the probability of a class, we can clearly see that Entropy ismaximum when probability of either of the classes is equal. Now, you can understand that when a decision algorithm tries to split the data, it selects the variable which will give us maximum reduction in system Entropy.

For the example of movie success rate – Initial Entropy in the system was:

*EntropyParent = -(0.57*ln(0.57) + 0.43*ln(0.43));* Which is 0.68

Entropy after Method 1 Split

Entropy$_{left}$ = -(.75*ln(0.75) + 0.25*ln(0.25)) =

Entropy$_{right}$ = -(.33*ln(0.33) + 0.67*ln(0.67)) = 0.63

Captured impurity or entropy after splitting data using Method 1 can be calculated using the followingformula: "*Entropy (Parent) – Weighted Average of Children Entropy*"
Which is,

*0.68 – (4*0.56 + 3*0.63)/7 = 0.09*
This number 0.09 is generally known as "Information Gain"

Entropy after Method 2 Split

*Entropy$_{left}$ = -(.67*ln(0.67) + 0.33*ln(0.33)) =*

*Entropy$_{middle}$ = -(.5*ln(0.5) + 0.5*ln(0.5)) =*

*Entropy$_{right}$ = -(.5*ln(0.5) + 0.5*ln(0.5)) = 0.69*

Now using the method used above, we can calculate the Information Gain as:

*Information Gain = 0.68 – (3*0.63 + 2*0.69 + 2*0.69)/7 = 0.02*

Hence, we can clearly see that Method 1 gives us more than 4 times information gain compared to Method 2 and hence Method 1 is the best split variable.

Gain Ratio

Soon after the development of entropy mathematicians realized that Information gain is biased toward multi-valued attributes and to conquer this issue, "Gain Ratio" came into picture which is more reliable than Information gain. The gain ratio can be defined as:

$$Gini_S(K) = \frac{T_1}{T} Gini\ (T_1) + \frac{T_2}{T} Gini\ (T_2)$$

Where Split info can be defined as:

$$-\sum_{i=1}^{n} Di\ log_2 Di$$

Assuming we are dividing our variable into 'n' child nodes and Di represents the number of records going into various child nodes. Hence gain ratio takes care of distribution bias while building a decision tree.

For the example discussed above, for Method 1

*Split Info = - ((4/7)*log2(4/7)) - ((3/7)*log2(3/7)) = 0.98*

And Hence,

Gain Ratio = 0.09/0.98 = 0.092

Gini Index

There is one more metric which can be used while building a decision tree is Gini Index (Gini Index is mostly used in CART). Gini index measures the impurity of a data partition K, formula for Gini Index can be written down as:

$$Gini(K) = 1 - \sum_{i=1}^{n} P_i^2$$

Where m is the number of classes, and P_i is the probability that an observation in K belongs to the class. Gini Index assumes a binary split for each of the attribute in S, let say T1 & T2. The Gini index of K given this partitioning is given by:

Which is nothing but a weighted sum of each of the impurities in split nodes. The reduction in impurity is given by:

$$Gini(K) - Gini_S(K)$$

Similar to Information Gain & Gain Ratio, split which gives us maximum reduction in impurity is considered fordividing our data.

Coming back to our movie example,If we want to calculate *Gini(K)*-

$$Gini\ (K) = 1 - \left(\left[\frac{4}{7}\right]^2 + \left[\frac{3}{7}\right]^2\right)$$

= 0.49

Now as per our Method 1, we can get $Gini_S(K)$ as,

$$Gini_S(K) = \frac{T_1}{T}\ Gini\ (T_1) + \frac{T_2}{T}\ Gini\ (T_2)$$

$$= \frac{4}{7}\left\{1 - \left(\left[\frac{3}{4}\right]^2 + \left[\frac{1}{4}\right]^2\right)\right\} + \frac{3}{7}\left\{1 - \left(\left[\frac{1}{3}\right]^2 + \left[\frac{2}{3}\right]^2\right)\right\}$$

= 0.24 + 0.19

= 0.43

LINEAR REGRESSION

Linear regression is a statistical approach for modelling relationship between a dependent variable with a given setof independent variables.

an approach for predicting a **response** using a **single feature**.

It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts theresponse value(y) as accurately as possible as a function of the feature or independent variable(x).

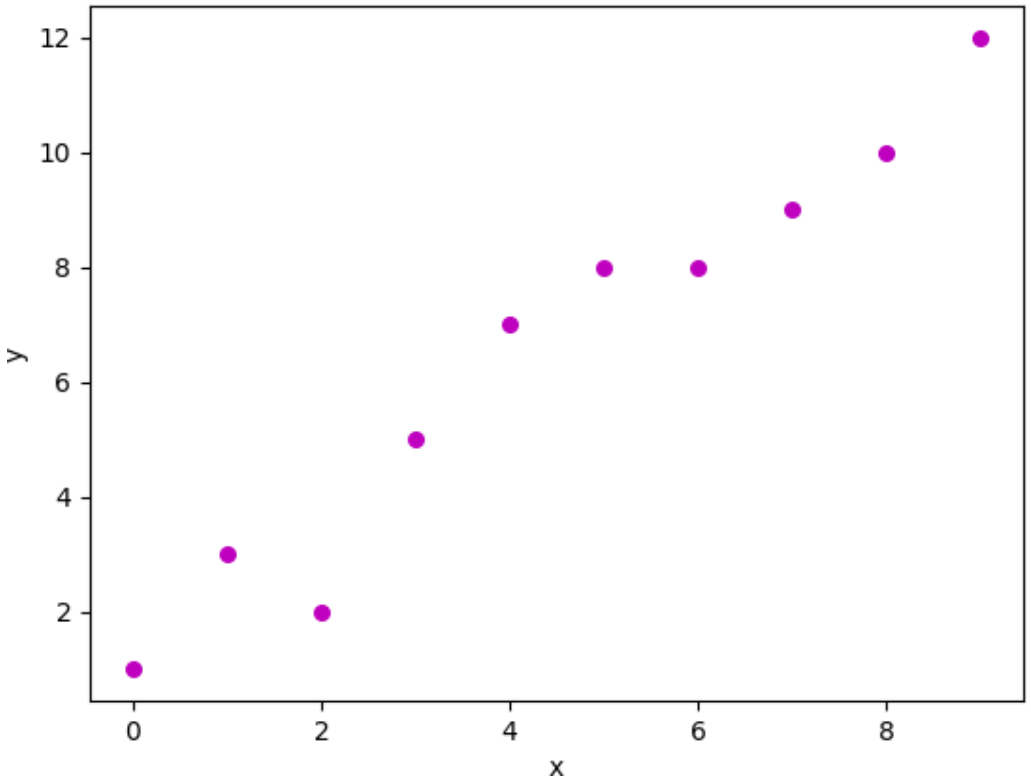
Let us consider a dataset where we have a value of response y for every feature x:

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

For generality, we define:

x as **feature vector**, i.e x = [x_1, x_2, …., x_n], y as **response vector**, i.e y = [y_1, y_2, …., y_n]for **n** observations (in above example, n=10).

A scatter plot of above dataset looks like:-



Now, the task is to find a **line which fits best** in above scatter plot so that we can predict the response for any newfeature values. (i.e a value of x not present in dataset)
This line is called **regression line**.
The equation of regression line is represented as:

Here,

- h(x_i) represents the **predicted response value** for ith observation.
- b_0 and b_1 are regression coefficients and represent **y-intercept** and **slope** of regression line respectively.

To create our model, we must "learn" or estimate the values of regression coefficients b_0 and b_1. And once we've estimated these coefficients, we can use the model to predict responses!

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i = h(x_i) + \varepsilon_i \Rightarrow \varepsilon_i = y_i - h(x_i)$$

Here, e_i is **residual error** in ith observation.

So, our aim is to minimize the total residual error.

We define the squared error or cost function, J as:

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

and our task is to find the value of b_0 and b_1 for which J(b_0,b_1) is minimum!

Without going into the mathematical details, we present the result here:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where SS_xy is the sum of cross-deviations of y and x:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

and SS_xx is the sum of squared deviations of x:

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

LOGISTIC REGRESSION

Consider an example dataset which maps the number of hours of study with the result of an exam. The result cantake only two values, namely passed(1) or failed(0):

HOURS(X)	0.50	0.75	1.00	1.25	1.50	1.75	2.00	2.25	2.50	2.75	3.00
PASS(Y)	0	0	0	0	0	0	1	0	1	0	1

i.e. y is a categorical target variable which can take only two possible type:"0" or "1".In order to generalize our model, we assume that:

- The dataset has 'p' feature variables and 'n' observations.
- The feature matrix is represented as:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$

Here, x_{ij} denotes the values of j^{th} feature for i^{th} observation.
Here, we are keeping the convention of letting x_{i0} = 1. (Keep reading, you will understand the logic in a few moments).

- The i^{th} observation, x_i, can be represented as:

$$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ . \\ . \\ x_{ip} \end{bmatrix}$$

- $h(x_i)$ represents the predicted response for i^{th} observation, i.e. x_i. The formula we use for calculating $h(x_i)$ is called **hypothesis**.

Differences between Linear Regression and Logistic Regression:-

LINEAR REGRESSION	**LOGISTIC REGRESSION**
Linear Regression is a supervised regression model.	Logistic Regression is a supervised classification model.
In Linear Regression, we predict the value by an integer number.	In Logistic Regression, we predict the value by 1 or 0.
Here no activation function is used.	Here activation function is used to convert a linear regression

Genearized Linear Models:-

The Generalized Linear Model (GLZ) is a generalization of the general linear model .In its simplest form, a linear model specifies the (linear) relationship between a dependent (or response) variable *Y*, and a set of predictor variables, the *X*'s, so that

$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_kX_k$

In this equation *b0* is the regression coefficient for the intercept and the *bi* values are the regression coefficients (for variables 1 through *k*) computed from the data.

So for example, we could estimate (i.e., predict) a person's weight as a function of the person's height and gender. You could use linear regression to estimate the respective regression coefficients from a sample of data, measuring height, weight, and observing the subjects' gender. For many data analysis problems, estimates of the linear relationships between variables are adequate to describe the observed data, and to make reasonable predictions for new observations..

However, there are many relationships that cannot adequately be summarized by a simple linear equation, for two major reasons:

Distribution of dependent variable. First, the dependent variable of interest may have a non-continuous distribution, and thus, the predicted values should also follow the respective distribution; any other predicted values are not logically possible. For example, a researcher may be interested in predicting one of three possible discrete outcomes (e.g., a consumer's choice of one of three alternative products). In that case, the dependent variable can only take on 3 distinct values, and the distribution of the dependent variable is said to be multinomial. Or suppose you are trying to predict people's family planning choices, specifically, how many children families will have, as a function of income and various other socioeconomic indicators. The dependent variable - number of children - is discrete (i.e., afamily may have 1, 2, or 3 children and so on, but cannot have 2.4 children), and most likely the distribution of that variable is highly skewed (i.e., most families have 1, 2, or 3 children, fewer will have 4 or 5, very few will have 6

SUPPORT VECTOR MACHINES

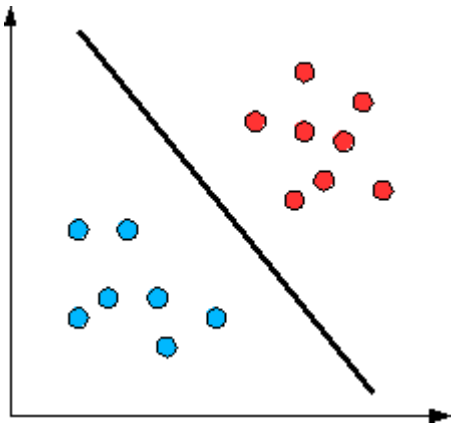
Support Vector Machine or SVM are supervised learning models with associated learning algorithms that analyze data for classification(clasifications means knowing what belong to what e.g ‘apple’ belongs to class ‘fruit’ while ‘dog’ to class ‘animals’ -see fig.1)

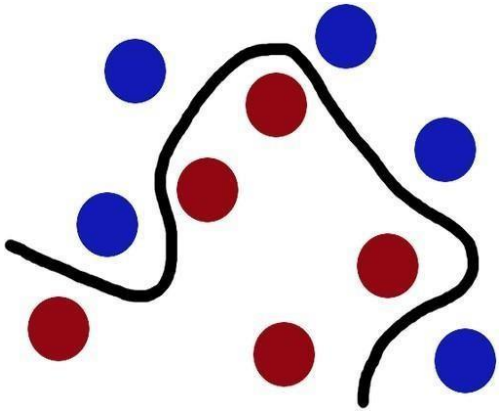
In support vector machines, it looks somewhat like which separates the blue balls from red.

SVM is a **classifier** formally **defined** by a separating hyperplane. An hyperplane is a subspace of one **dimension** lessthan its **ambient space**. The **dimension** of a mathematical space (or object) is informally defined as the minimum

number of coordinates (x,y,z axis) needed to specify any point (like each blue and red point) within it while anambient space is the space surrounding a mathematical object.

Therefore the hyperplane of a two dimensional space below (fig.2) is a one dimensional line dividing the red and bluedots.

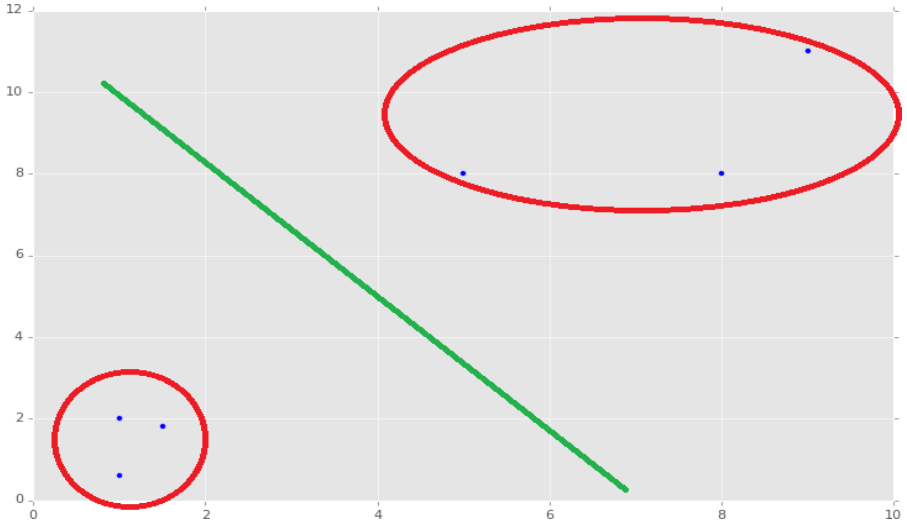




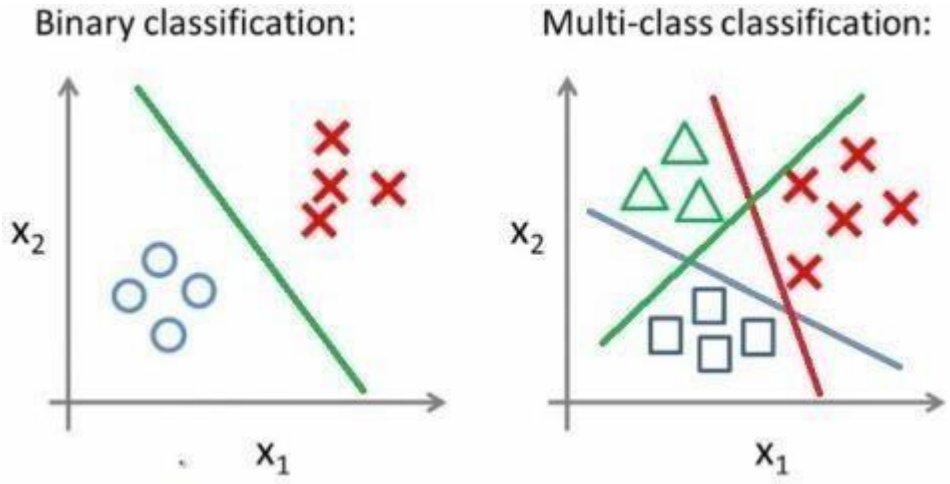
Can you try to solve the above problem linearly like we did with Fig. 2?NO!
The red and blue balls cannot be separated by a straight line as they are randomly distributed and this, in reality, is how most real life problem data are -**randomly distributed.**

In machine learning, a "kernel" is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem. It entails transforming linearly inseparable data like (Fig. 3) to linearly separable ones (Fig. 2). The kernel function is what is applied on each data instance to map the original non-linear observations intoa higher-dimensional space in which they become separable.

Using the dog breed prediction example again, kernels offer a better alternative. Instead of defining a slew of features, you define a single kernel function to compute **similarity** between breeds of dog. You provide this kernel, together with the data and labels to the learning algorithm, and out comes a classifier.



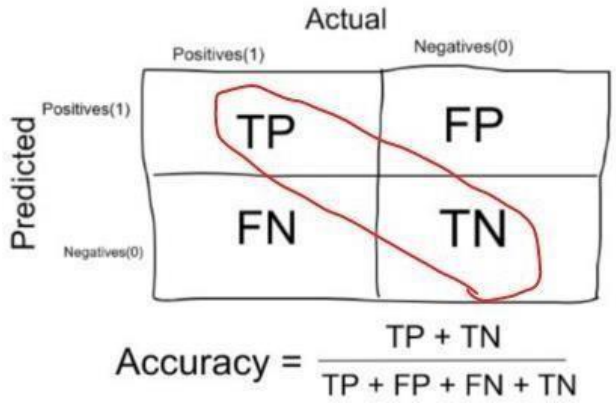
So this is with two features, and we see we have a 2D graph. If we had three features, we could have a 3D graph. The 3D graph would be a little more challenging for us to visually group and divide, but still do-able. The problem occurs when we have four features, or four-thousand features. Now you can start to understand the power of machine learning, seeing and analyzing a number of dimensions imperceptible to us.



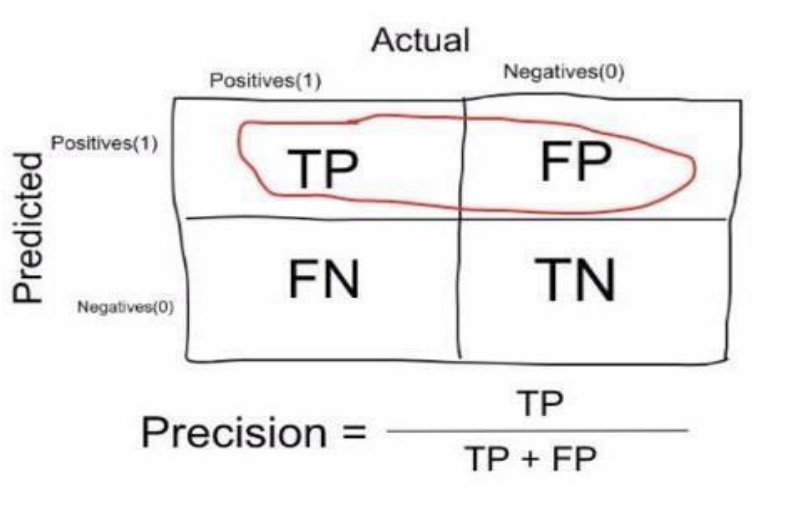
Common examples include **image classification** (is it a cat, dog, human, etc)or **handwritten digitrecognition** (classifying an image of a handwritten number into a digit from 0 to 9).

Performance Metrics

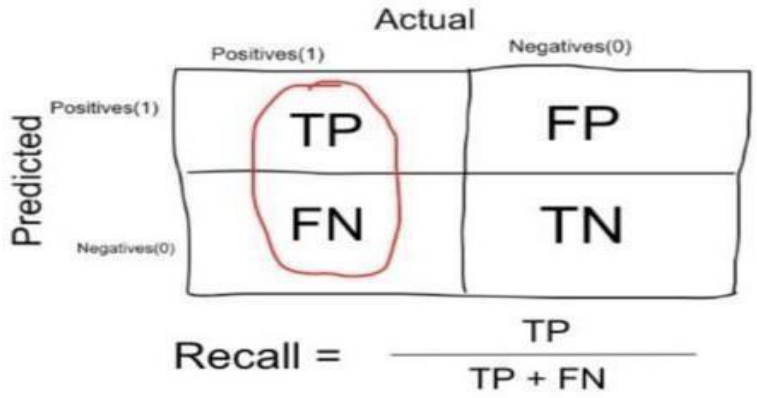
- **Accuracy**can be calculated by taking average of the values lying across the "main diagonal" i.e Accuracy = (True Positives+False Negatives)/Total Number of Samples



Precision:-It is the number of correct positive results divided by the number of positive results predicted by classifier.



- Recall :- It is the number of correct positive results divided by the number of all relevant samples



Structured prediction or structured (output) learning :-

It is an umbrella term for supervised machine learning techniques that involves predicting structured objects, rather than scalar discrete or real values.

Similar to commonly used supervised learning techniques, structured prediction models are typically trained by means of observed data in which the true prediction value is used to adjust model parameters. Due to the complexityof the model and the interrelations of predicted variables the process of prediction using a trained model and of training itself is often computationally infeasible and approximate inference and learning methods are used.

For example, the problem of translating a natural language sentence into a syntactic representation such as a parse tree can be seen as a structured prediction problem in which the structured output domain is the set of all possible parse trees. Structured prediction is also used in a wide variety of application domains including bioinformatics, natural language processing, speech recognition, and computer vision.

Example: sequence tagging

Sequence tagging is a class of problems prevalent in natural language processing, where input data are often sequences (e.g. sentences of text). The sequence tagging problem appears in several guises, e.g. part-of-speech tagging and named entity recognition. In POS tagging, for example, each word in a sequence must receive a "tag" (class label) that expresses its "type" of word:

DT-DeterminerVB-Verb
JJ-AdjectiveNN-Noun

Ranking :-

Learning to Rank (LTR) is a class of techniques that apply supervised machine learning (ML) to solve **ranking problems**. The main difference between LTR and traditional supervised ML is this:

- Traditional ML solves a prediction problem (classification or regression) on a single instance at a time.

E.g. if you are doing spam detection on email, you will look at all the features associated with that email and classify it as spam or not. The aim of traditional ML is to come up with a class (spam or no-spam) or asingle numerical score for that instance.
LTR solves a ranking problem on a list of items. The aim of LTR is to come up with optimal ordering of those items. As such, LTR doesn't care much about the exact score that each item gets, but cares more about the relative ordering among all the items.

The most common application of LTR is search engine ranking, but it's useful anywhere you need to produce a ranked list of items.

The training data for a LTR model consists of a list of items and a "ground truth" score for each of those items. For search engine ranking, this translates to a list of results for a query and a relevance rating for each of those results with respect to the query. The most common way used by major search engines to generate these relevance ratingsis to ask human raters to rate results for a set of queries

Learning to rank algorithms have been applied in areas other than information retrieval:

- In machine translation for ranking a set of hypothesized translations
- In computational biology for ranking candidate 3-D structures in protein structure prediction problem
- In recommender systems for identifying a ranked list of related news articles to recommend to a user after heor she has read a current news article
- In software engineering, learning-to-rank methods have been used for fault localization kernel k-means clustering algorithm

This algorithm applies the same trick as k-means but with one difference that here in the calculation of distance,kernel method is used instead of the Euclidean distance.

Algorithmic steps for Kernel k-means clustering

Let X = {a1, a2, a3, ..., an} be the set of data points and 'c' be the number of clusters.

1) Randomly initialize 'c' cluster center.

2) Compute the distance of each data point and the cluster center in the transformed space using:

$$\mathcal{D}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{a}_i \in \pi_c} \phi(\mathbf{a}_i)}{|\pi_c|}.$$

$$\phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_i) - \frac{2\sum_{\mathbf{a}_j \in \pi_c} \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_j)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_l \in \pi_c} \phi(\mathbf{a}_j) \cdot \phi(\mathbf{a}_l)}{|\pi_c|^2}.$$

where,

Matrix Factorization:
matrix factorization is to, obviously, factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix.

Matrix factorization can be used to discover latent features underlying the interactions between two different kinds of entities. (Of course, you can consider more than two kinds of entities and you will be dealing with tensor factorization, which would be more complicated.) And one obvious application is to predict ratings in collaborative filtering.

In a recommendation system such as Netflix or MovieLens, there is a group of users and a set of items (movies for the above two systems). Given that each users have rated some items in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users. In this case, all the information we have about the existing ratings can be represented in a matrix. Assume now we have 5 users and 10 items, and ratings are integers ranging from 1 to 5, the matrix may look something like this (a
hyphen means that the user has not yet rated the movie):

	D1	**D2**	**D3**
U1	5	3	-
U2	4	-	-
U3	1	1	-

U4	1	-	-
U5	-	1	5

The mathematics of matrix factorization

Having discussed the intuition behind matrix factorization, we can now go on to work on the mathematics. Firstly, we have a set of users, and a set of items. Let of size $|U| \times |D|$ be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times K$ matrix) such that their product approximates :

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

In this way, each row of would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item d_j by , we can calculate the dot product of the two vectors corresponding to and d_j:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

Now, we have to find a way to obtain and $\mathbf{Q}$. One way to approach this problem is the first intialize the two matrices with some values, calculate how `different' their product is to , and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of and In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

Having obtained the gradient, we can now formulate the update rules for both and q_{kj}:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

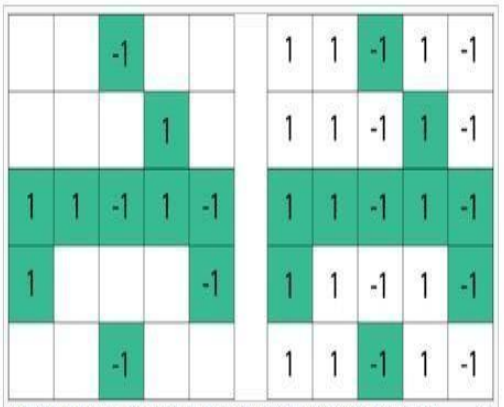
Here, is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for , say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

A question might have come to your mind by now: if we find two matrices and $\mathbf{Q}$ such that approximates , isn't that our predictions of all the unseen ratings will all be zeros? In fact, we are not really trying to come up with and $\mathbf{Q}$ such that we can reproduce exactly. Instead, we will only try to minimise the errors ofthe observed user-item pairs. In other words, if we let be a set of tuples, each of which is in the formof (u_i, d_j, r_{ij}), such that contains all the observed user-item pairs together with the associated ratings, we are only trying to minimise every for $(u_i, d_j, r_{ij}) \in T$. (In other words, is our set of training data.) As for the restof the unknowns, we will be able to determine their values once the associations between the users, items andfeatures have been learnt.

Using the above update rules, we can then iteratively perform the operation until the error converges to its minimum. We can check the overall error as calculated using the following equation and determine when we shouldstop the process.

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} \left(r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj}\right)^2$$

Matrix completion is the task of filling in the missing entries of a partially observed matrix. A wide range of datasets are naturally organized in matrix form. One example is the movie-ratings matrix, as appears in the Netflix problem: Given a ratings matrix in which each entry (i, j) represents the rating of movie j by customer i if customer i has watched movie j and is otherwise missing, we would like to predict the remaining entries in order to make good recommendations to customers on what to watch next. Another example is the term-document matrix: The frequencies of words used in a collection of documents can be represented as a matrix, where each entry corresponds to the number of times the associated term appears in the indicated document.



Matrix completion of a partially revealed 5 by 5 matrix with rank-1. Left: observed incomplete matrix; Right: matrix completion result.

Without any restrictions on the number of degrees of freedom in the completed matrix this problem is underdetermined since the hidden entries could be assigned arbitrary values. Thus matrix completion often seeks to find the lowest rank matrix or, if the rank of the completed matrix is known, a matrix of rank r that matches the known entries. The illustration shows that a partially revealed rank-1 matrix (on the left) can be completed with zero-error (on the right) since all the rows with missing entries should be the same as the third row. In the case of the Netflix problem the ratings matrix is expected to be low-rank since user preferences can often be described by a few factors, such as the movie genre and time of release. Other applications include computer vision, where missing pixels in images need to be reconstructed, detecting the global positioning of sensors in a network from partial distance information, and multiclass learning. The matrix completion problem is in general NP-hard, but there are tractable algorithms that achieve exact reconstruction with high probability.

In statistical learning point of view, the matrix completion problem is an application of matrix regularization which is a generalization of vector regularization. For example, in the low-rank matrix completion problem one may apply the regularization penalty taking the form of a nuclear norm $R(X) = \lambda \|X\|_*$

UNIT-III

Supervised Learning – II (Neural Networks)

Neural Network Representation – Problems – Perceptrons , Activation Functions, Artificial Neural Networks(ANN) , Back Propagation Algorithm.

Convolutional Neural Networks - Convolution and Pooling layers, , Recurrent Neural Networks (RNN).

Classification Metrics: Confusion matrix, Precision, Recall, Accuracy, F-Score, ROC curves

Online Learning:-

Online machine learning is a method of machine learning in which data becomes available in a sequential order and is used to update our best predictor for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once. Online learning is a common technique used in areas of machine learning where it is computationally infeasible to train over the entire dataset, requiring the need of out-of-core algorithms. It is also used in situations where it is necessary for the algorithm to dynamically adapt to new patterns in the data, or when the data itself is generated as a function of time, e.g., stock price prediction. Online learning algorithms may be prone to catastrophic interference, a problem that can be addressed by incremental learning approaches.

Online or incremental machine learning is a subfield of machine learning wherein we focus on incrementally updating a model as soon as we see a new training example. This differs from the more traditional batch learning approach where we train on all of our data in one go.

Online learning has several upsides, the two most important of which relate to space complexity and speed. Traininga model online means you don't need to keep your entire training data in memory, which is great news if you are dealing with massive datasets. The incremental nature also means that your model can quickly react to changes in thedistribution of the data coming in, provided the algorithm you use is tweaked properly.

The second point makes online learning especially attractive: implemented correctly, it can do near real- time *learning* as well as inference. It's a good choice in situations where you want to react to unforseen changes as fast as possible, making it a viable option for implementing dynamic pricing systems, recommendation engines, decision engines and much more besides. It's also an excellent choice for immediate-reward reinforcement learning (e.g. contextual bandits).

is Semi-Supervised Machine Learning important?

When you don't have enough labeled data to produce an accurate model and you don't have the ability or resources to get more, you can use semi-supervised techniques to increase the size of your training data. For example, imagineyou are developing a model for a large bank intended to detect fraud. Some fraud you know about, but other instances of fraud slipped by without your knowledge. You can label the dataset with the fraud instances you're aware of, but the rest of your data will remain unlabelled:

Name	Loan Amount	Loan Repaid	Fraud
Ashley	100000	1	1
Chuck	25000	0	0
Tim	4000	1	1
Mike	150000	1	1
Colin	200000000	0	
Libby	400400	1	0
Sheila	3200	1	1
Mandi	34850	1	
Gareth	6570	0	0

You can use a semi-supervised learning algorithm to label the data, and retrain the model with the newly labeled dataset:

Name	Loan Amount	Loan Repaid	Fraud
Ashley	100000	1	1
Chuck	25000	0	0
Tim	4000	1	1
Mike	150000	1	1
Colin	200000000	0	0
Libby	400400	1	0
Sheila	3200	1	1
Mandi	34850	1	1
Gareth	6570	0	0

Then, you apply the re-trained model to new data, more accurately identifying fraud using supervised machine learning techniques. However, there is no way to verify that the algorithm produced labels that are 100% accurate, resulting in less trustworthy outcomes than traditional supervised techniques.

Classification methods for IOT applications:-

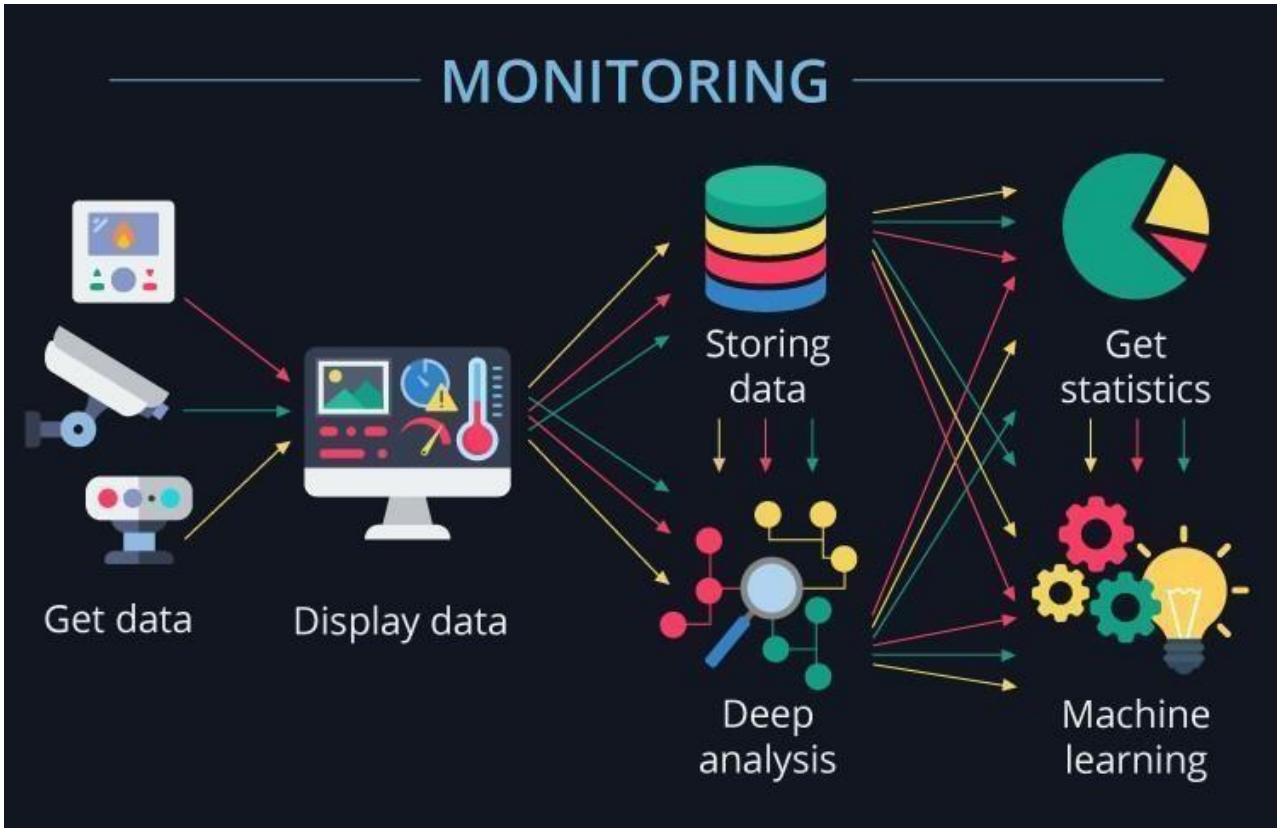
Considering the variety of IoT systems, it can be difficult for business players to decide what they really need. The number of IoT solutions is increasing at quite an impressive pace, and these systems are designed to perform variousfunctions.

Basing on our experience in IoT development, we want to bring more clarity to this variety and introduce ourapproach to the IoT systems classification.

IoT solutions for monitoring

With the help of sensor data, a user of a smart, connected thing can monitor its real-time state and environment. In a longer-term perspective, the results of monitoring can be gathered and applied for advanced insights.

Accumulated sensor data helps to get detailed and meaningful statistics, assess the performance of equipment from different perspectives, uncover new patterns and tendencies and more. Monitoring is important assistance in proactive maintenance as IoT system users get an opportunity to identify the problems before the damage is done and take necessary measures.



Storing data and showing to users

Connected devices can give an expanded picture of patients' health, environmental conditions, equipment state in factories and power plants, help users monitor their pets, cars, homes and more. Remote monitoring of facilities, processes and events brings better operational insights: sensors can gather the data that helps to see and assess real- time state of smart connected things.

Let's take a smart railway as an example. Trains can be equipped with sensors that take the data about real-time status of the parts of a train and enable the IoT system users to monitor the state of breaks, wheels and engines. The results can be viewed by a train driver, traffic superintendent or any other responsible person as well as collected for further analysis.
In subway trains, big data can be used to measure passenger flow and identify the hours and days when additional trains are needed. Another case (when the solutions are designed not only for the monitoring) is adding a new train

to the route if many passengers are waiting (for example, trains come every 20 minutes, but when the platform is getting overcrowded, additional trains are provided).

Deep analysis and detection of specific situations

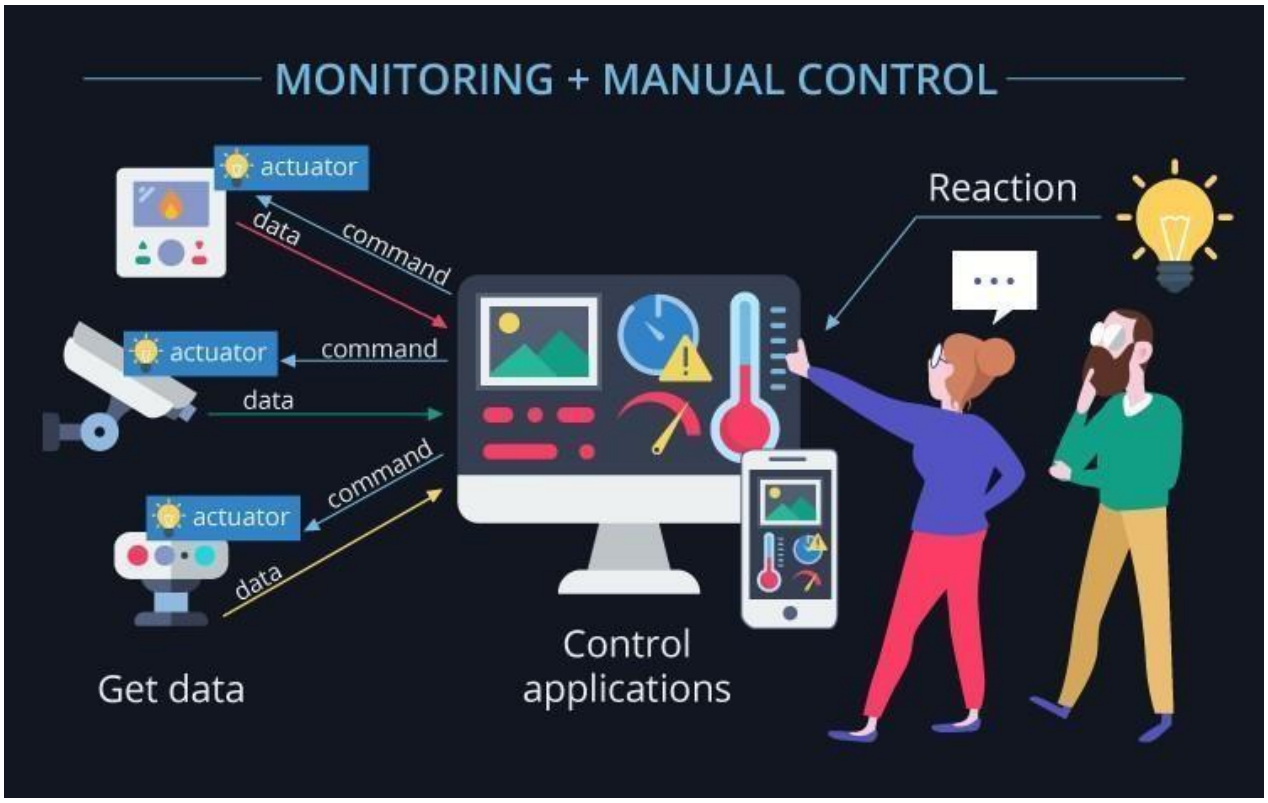
An IoT system can not only take and display sensor data (for example, temperature or humidity), but also make conclusions about what certain values of data could mean.

Processing the data coming from sensors, an IoT system can not only detect anomalies, but also predict

operational malfunctioning and point at the root causes of problems. Thus, comparing current data coming from sensors and the data stored in the cloud as acceptable values, an IoT solution monitoring trains and railways can show that a certain part is about to be out of order. The historical data about the conditions of using trains and the breakdowns that happen can help identify (and then – predict) the conditions leading to failures.

IoT solutions for monitoring with manual control

In many cases, the potential of the Internet of things can bring far more value than just improved monitoring. Thus, a user of an IoT system can give commands to the connected things and enable them to perform certain operations.



Let's take a case when a user receives the results of monitoring and some response is needed. Theoretically, a smart system can tackle certain issues even without human participation, but not always. In some cases, machines are

incapable to perform required operations, in the others – the issues cannot be trusted to machines (for example, replacing a broken detail).

Manual control is a good solution to be on the safe side in the situations an IoT system didn't face before. With unfamiliar incoming data, it can be better to let humans make final decisions. An IoT system can, for example, give recommendations – and people decide whether to follow them or not. In the longer term, the data about user actions in response to certain sensor data can be used by machine learning module to make models for control applications and contribute to further automation of a system.

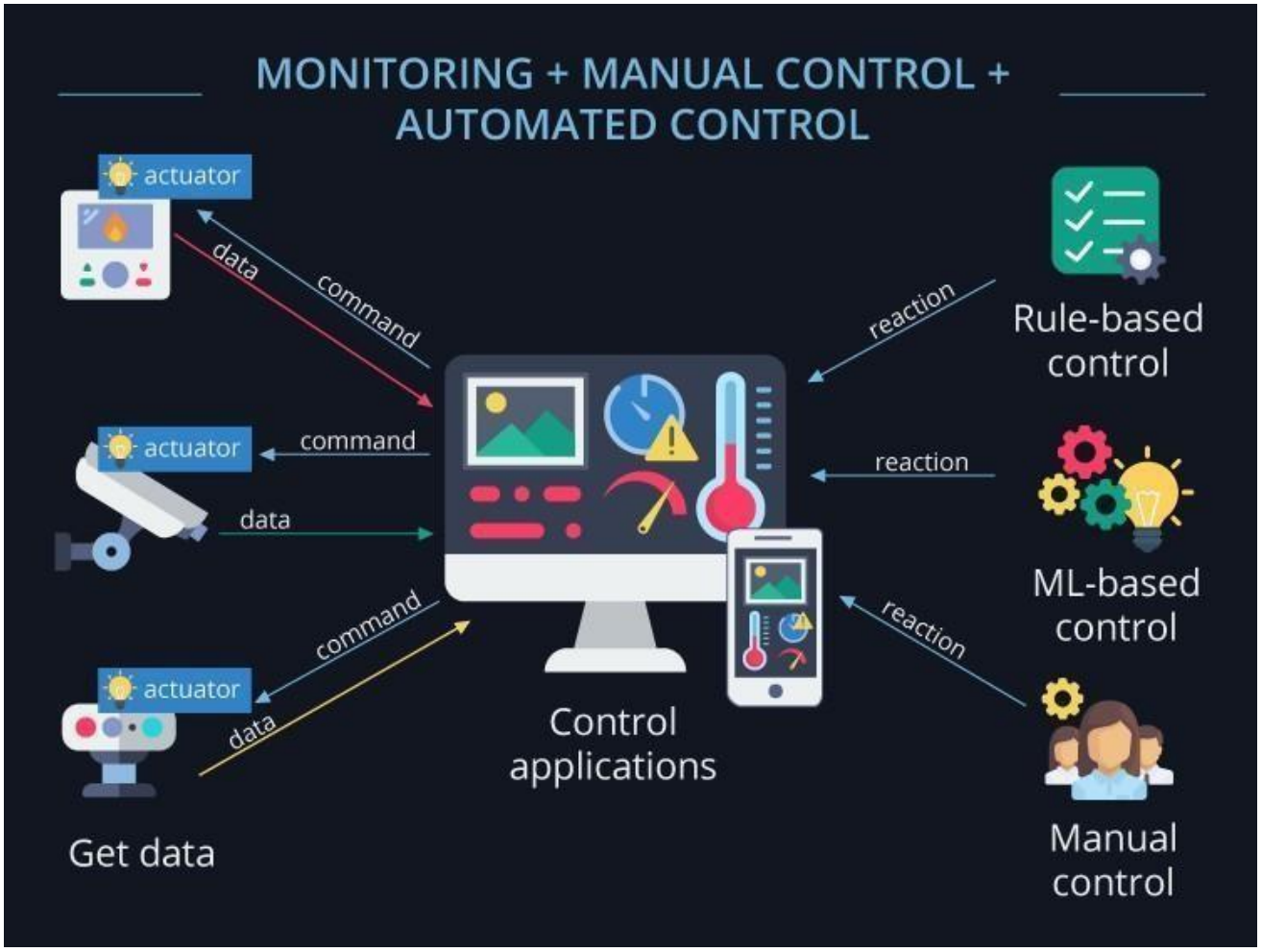
IoT solutions for monitoring with automated control

In IoT systems with automated control, control applications send commands to actuators. The choice of the commands to be sent depends on the data coming from sensors and/or previously defined schedules.

Rule-based control

An IoT system with rule-based control is designed to act in accordance with the algorithms featuring what should bedone in response to certain data coming from sensors. Rules are stated before the system is put into action.

In freight trains, sensors can measure temperature, vibrations and other parameters critical for cargo. This big data goes to the cloud, and, when the smart system identifies that some parameters differ from acceptable values, control applications send commands to adjust these parameters (for example, increase or decrease refrigerating).
In such freight trains, it's also possible to set acceptable values in different carriages before the trip begins (as soon as various goods can be transported in different cars and each type of transported goods requires corresponding conditions).



Machine learning-based control

The next stage of IoT systems evolution is the systems with machine-learning based control when IoT potential is used to its fullest extent. In machine learning, sensor data is continuously collected and regularly used in standard machine learning algorithms. New models are generated, and their applicability is then tested by analysts and / or data scientists. When the models are approved, they can be used by an IoT system.

In a smart railway, such learning can be performed with analyzing human commands. The responses of humans to certain sensor data are accumulated in a big data warehouse, and then the models of how to act are built accordingly (considering the actions of humans in certain situations).
Cameras can take photos of potential problems (suspicions that there are some problems) and send them for further analysis (either manually or with computer assistance). As soon as various images are collected in the cloud (and theproblems are identified), smart systems "learn" to determine the types of problems without human participation and send corresponding notifications

Machine learning potential can contribute to optimizing subway trains' schedules. Smart system accommodates the data about the passenger flow on different days and at different times of the day. Then, it defines the days and the time slots when additional trains should be put on the line, and, thus, offers schedule optimizations.

It makes sense to notice that, even if an IoT solution can, in most cases, successfully operate without human participation, there should be an option of manual control.

Endnote

To sum it up, IoT systems can be classified in the following way:

Solutions for monitoring: sensor data helps monitor the state and environment of smart connected things. In this case, IoT solutions can perform **storing data and showing it to users**. Also, the data gathered with sensors can be analyzed and used for **detecting specific situations**.
Monitoring + manual control: with user apps, users are empowered to give the commands to connected things' actuators and control the processes in an IoT system.
Monitoring + automated control: control apps automatically send the commands to actuators, and human participation in controlling an IoT system is significantly reduced. Automated control can be performed on the basis of the previously defined rules (**rule-based control**). With **machine learning**, IoT systems can adapt to user behavior and changing environment and "learn" how to perform operations in more productive ways. However, it's reasonable to enable the shift from automated to manual control over IoT solution's operation as no IoT system is immune to breakdowns and unpredicted situations.
An organization needs to clearly realize what it expects from the Internet of Things and which types of IoT solution will help to cover current and future business needs. The exploration of the IoT path should begin with the great dealof business and IT strategy planning.

Various models for IOT applications:-

1. Smart home

Smart Home clearly stands out, ranking as highest Internet of Things application on all measured channels. More than 60,000 people currently search for the term "Smart Home" each month. This is not a surprise. The IoT Analytics company database for Smart Home includes 256 companies and startups. More companies are active in smart home than any other application in the field of IoT. The total amount of funding for Smart Home startups currently exceeds $2.5bn. This list includes prominent startup names such as Nest or AlertMe as well as a number ofmultinational corporations like Philips, Haier, or Belkin.

2. Wearables

Wearables remains a hot topic too. As consumers await the release of Apple's new smart watch in April 2015, there are plenty of other wearable innovations to be excited about: like the Sony Smart B Trainer, the Myo gesture control, or LookSee bracelet. Of all the IoT startups, wearables maker Jawbone is probably the one with the biggest funding to date. It stands at more than half a billion dollars!

3. Smart City

Smart city spans a wide variety of use cases, from traffic management to water distribution, to waste management, urban security and environmental monitoring. Its popularity is fueled by the fact that many Smart City solutions promise to alleviate real pains of people living in cities these days. IoT solutions in the area of Smart City solve traffic congestion problems, reduce noise and pollution and help make cities safer.

4. Smart grids

Smart grids is a special one. A future smart grid promises to use information about the behaviors of electricity suppliers and consumers in an automated fashion to improve the efficiency, reliability, and economics of electricity. 41,000 monthly Google searches highlights the concept's popularity. However, the lack of tweets (Just 100 per month) shows that people don't have much to say about it.

5. Industrial internet

The industrial internet is also one of the special Internet of Things applications. While many market researches such as Gartner or Cisco see the industrial internet as the IoT concept with the highest overall potential, its popularity currently doesn't reach the masses like smart home or wearables do. The industrial internet however has a lot going for it. The industrial internet gets the biggest push of people on Twitter (~1,700 tweets per month) compared to other non-consumer-oriented IoT concepts.

6. Connected car

The connected car is coming up slowly. Owing to the fact that the development cycles in the automotive industry typically take 2-4 years, we haven't seen much buzz around the connected car yet. But it seems we are getting there. Most large auto makers as well as some brave startups are working on connected car solutions. And if the BMWs and Fords of this world don't present the next generation internet connected car soon, other well-known giants will: Google, Microsoft, and Apple have all announced connected car platforms.

7. Connected Health (Digital health/Telehealth/Telemedicine)

Connected health remains the sleeping giant of the Internet of Things applications. The concept of a connectedhealth care system and smart medical devices bears enormous potential (see our analysis of market segments), not just for companies also for the well-being of people in general. Yet, Connected Health has not reached the masses yet. Prominent use cases and large-scale startup successes are still to be seen. Might 2015 bring the breakthrough?

8. Smart retail

Proximity-based advertising as a subset of smart retail is starting to take off. But the popularity ranking shows that it is still a niche segment. One LinkedIn post per month is nothing compared to 430 for smart home.

9. Smart supply chain

Supply chains have been getting smarter for some years already. Solutions for tracking goods while they are on the road, or getting suppliers to exchange inventory information have been on the market for years. So while it is perfectly logic that the topic will get a new push with the Internet of Things, it seems that so far its popularity remains limited.

10. Smart farming

Smart farming is an often overlooked business-case for the internet of Things because it does not really fit into the well-known categories such as health, mobility, or industrial. However, due to the remoteness of farming operations and the large number of livestock that could be monitored the Internet of Things could revolutionize the way farmers work. But this idea has not yet reached large-scale attention. Nevertheless, one of the Internet of Things applications that should not be underestimated. Smart farming will become the important application field in the predominantly agricultural-product exporting countries.

Here are some examples of IoT Applications in different industries:

Airline – An equipment tracking app provides an airline's engineers with a live view of the locations of each piece of maintenance equipment. By increasing the efficiency of engineers, this IoT application is not only generating significant cost savings and process improvements, but also impacting the customer experience in the end through more reliable, on-time flights.

Pharmaceutical – A medication temperature monitoring app uses sensors to detect if the medication's temperature has gone outside of the acceptable range and ensures medical supplies still meet quality standards upon delivery. The handling temperatures are medications, vaccines for examples, is critical to their effectiveness. IoT based smartapplications can be used to not monitor that medications are kept within the proper handling temperature range, but also to remind patients when it is time to take their medication.

Manufacturing – A lighting manufacturer for the horticultural industry built a Smart App that leverages IoT sensors and predictive analytics to perform predictive maintenance and optimize lighting, power consumption and plant photosynthesis. The IoT application transformed their business from a lighting systems manufacturer to a greenhouse optimization as-a-service business.

Insurance – An insurance company offers policyholders discounts for wearing Internet-connected Fitbit wristbands.The fitness tracking service is part of the insurer's Vitality program aimed at integrating wellness benefits with life insurance. Through this IoT application, this insurer is creating smart life insurance products and rewarding customers for their positive actions.

Business Services – A facility services company uses their multi-device IoT software to enable support personnel to receive alerts about service issues and take immediate action. By aggregating data from thousands of sensors in devices like coffee machines, soap dispensers, paper towel dispensers and mouse traps rather than doing manual checks, the application has significantly cut costs and improved service levels.

Media & Entertainment – An entertainment design and production firm uses sensors in turnstiles of venues to understand the foot traffic of people at events. Their IoT application visualizes the attendee traffic patterns in real time to help sponsors understand the best places to advertise, and to ensure the attendee count stays within the fire code compliance of the venue.

What is sparse in machine learning?

A common problem in **machine learning** is **sparse** data, which alters the performance of **machine learning** algorithms and their ability to calculate accurate predictions. Data is considered **sparse** when certain expected values in a dataset are missing, which is a common phenomenon in general large scaled data analysis.

What is sparse model?

Sparse modeling is a rapidly developing area at the intersection of statistical learning and signal processing, motivated by the age-old statistical problem of selecting a small number of predictive variables in high-dimensional datasets.

Sparse models contain fewer features and hence are easier to train on limited data. Fewer features also means less chance of over fitting. Fewer features also means it is easier to explain to users, as only most meaningful features remain in face recognition, sparse models provide a unique way to recognize a face from a database of profiles takenunder different orientations in MRI, sparse models promise faster image acquisition.

Sparse models – models where only a small fraction of parameters are non-zero – arise frequently in machine learning. Sparsity is beneficial in several ways: sparse models are more easily interpretable by humans, and sparsity can yield statistical benefits – such as reducing the number of examples that have to be observed to learn the model. In a sense, we can think of sparsity as an antidote to the oft-maligned curse of dimensionality.

More formally, suppose we're given data X on which we want to run an optimization algorithm F to obtain a model $W = F(X)$. We might think of F as stochastic gradient descent for learning a logistic regressor, or alternating least-squares for non-negative matrix factorization (NMF). If X is high-dimensional, this procedure can be expensive both in time and memory. But if our target model W is sparse, we can instead run our optimization routine F on a compressed, lower-dimensional version of the data $X' = PX$, where P is a random projection matrix. This yields a compressed solution $W' = F(X')$. We can then leverage

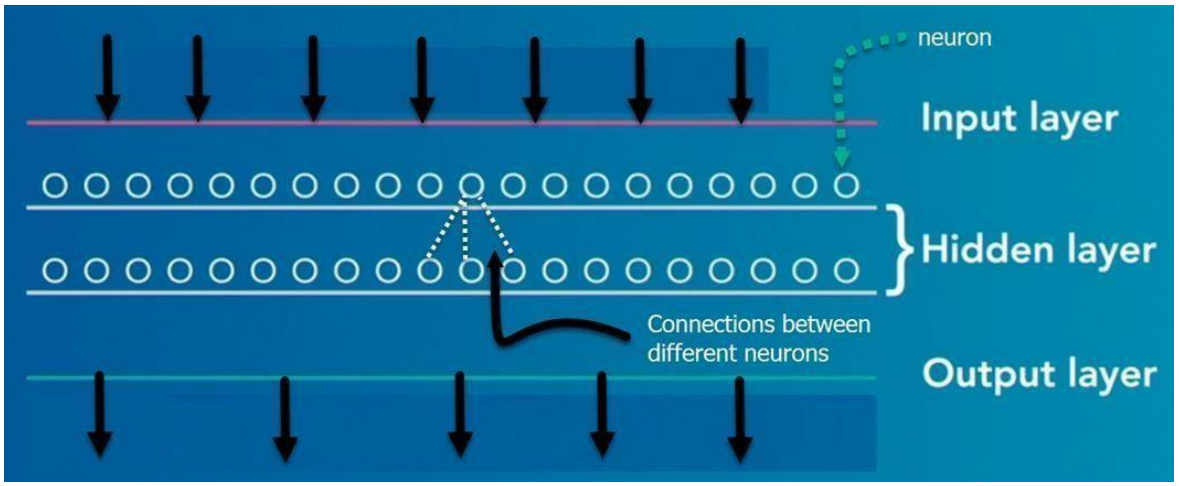
classic methods from compressive sensing to recover an approximate solution W_{approx} in the original, high-dimensional space such that $W_{approx} \approx W$.

What is Deep Learning?

Deep learning is a computer software that **mimics the network of neurons in a brain**. It is a subset of machine learning and is called deep learning because it makes use of deep **neural networks.**

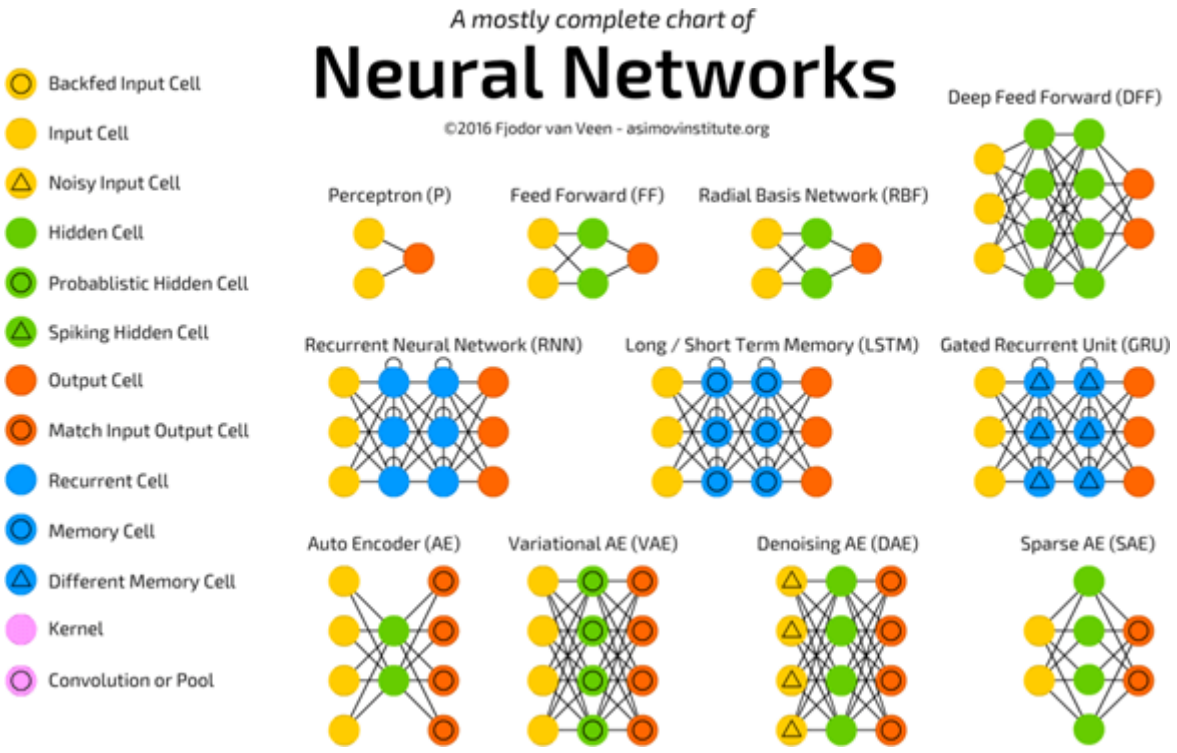
Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more thantwo layers.



Each Hidden layer is composed of neurons. The neurons are connected to each other. The neuron will process and then propagate the input signal it receives the layer above it. The strength of the signal given the neuron in the next layer depends on the weight, bias and activation function.

The network consumes large amounts of input data and operates them through multiple layers; the network can learnincreasingly complex features of the data at each layer.



Why is Deep Learning Important?

Deep learning is a powerful tool to make prediction an actionable result. Deep learning excels in pattern discovery (unsupervised learning) and knowledge-based prediction. Big data is the fuel for deep learning. When both are combined, an organization can reap unprecedented results in term of productivity, sales, management, and innovation.

Deep learning can outperform traditional method. For instance, deep learning algorithms are 41% more accurate than machine learning algorithm in image classification, 27 % more accurate in facial recognition and 25% in voice recognition.

Limitations of deep learningData labelling

Most current AI models are trained through "supervised learning." It means that humans must label and categorize the underlying data, which can be a sizable and error-prone chore. For example, companies developing self-driving- car technologies are hiring hundreds of people to manually annotate hours of video feeds from prototype vehicles to help train these systems.

Obtain huge training datasets

It has been shown that simple deep learning techniques like CNN can, in some cases, imitate the knowledge of experts in medicine and other fields. The current wave of machine learning, however, requires training data sets that are not only labeled but also sufficiently broad and universal.

Deep-learning methods required thousands of observation for models to become relatively good at classification tasks and, in some cases, millions for them to perform at the level of humans. Without surprise, deep learning is famous in giant tech companies; they are using big data to accumulate petabytes of data. It

allows them to create an impressive and highly accurate deep learning model.

Modeling Time Series Data:-

A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series isa sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

Time series are very frequently plotted via line charts. Time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, earthquake prediction, and largely in any domain of applied science and engineering which involves temporal measurements.

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data

Methods for time series analysis may be divided into two classes: frequency-domain methods and time- domain methods. The former include spectral analysis and wavelet analysis; the latter include auto- correlation and cross-correlation analysis. In the time domain, correlation and analysis can be made in a filter-like manner using scaled correlation, thereby mitigating the need to operate in the frequency domain.

Additionally, time series analysis techniques may be divided into parametric and non-parametric methods. The parametric approachesassume that the underlying stationary stochastic process has a certain structure which can be described using a small number of parameters (for example, using an autoregressive or moving average model).In these approaches, the task is to estimate the parameters of the model that describes the stochastic process. By

network. HMM models are widely used in speech recognition, for translating a time series of spoken words into text.

Feature Learning:-

In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in pattern recognition, classification and regression.

In machine learning, feature learning or representation learning[1] is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data. This replaces manual feature engineeringand allows a machine to both learn the features and use them to perform a specific task.

Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensor data has not yielded to attempts to algorithmically define specific features. An alternative is to discover such features or representations through examination, without relying on explicit algorithms.

Feature learning can be either supervised or unsupervised.

Supervised feature learning is learning features from labeled data. The data label allows the system to compute an error term, the degree to which the system fails to produce the label, which can then be used as feedback to correct the learning process (reduce/minimize the error). Approaches include:

Supervised dictionary learning

Dictionary learning develops a set (dictionary) of representative elements from the input data such that each data point can be represented as a weighted sum of the representative elements.

In supervised feature learning, features are learned using labeled input data. Examples include supervised neural networks, multilayer perceptron and (supervised) dictionary learning.

In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering

Neural networks

Neural networks are a family of learning algorithms that use a "network" consisting of multiple layers of inter- connected nodes. It is inspired by the animal nervous system, where the nodes are viewed as neurons and edges are viewed as synapses. Each edge has an associated weight, and the network defines computational rules for passing input data from the network's input layer to the output layer. A network function associated with a neural network characterizes the relationship between input and output layers, which is parameterized by the weights. With appropriately defined network functions, various learning tasks can be performed by minimizing a cost function over the network function (weights).

Multilayer neural networks can be used to perform feature learning, since they learn a representation of their input atthe hidden layer(s) which is subsequently used for classification or regression at the output layer. The most popular network architecture of this type is Siamese networks.

Unsupervised

Unsupervised feature learning is learning features from unlabeled data. The goal of unsupervised feature learning is often to discover low-dimensional features that captures some structure underlying the high-dimensional input data. When the feature learning is performed in an unsupervised way, it enables a form of semisupervised learning where features learned from an unlabeled dataset are then employed to improve performance in a supervised setting with labeled data. Several approaches are introduced in the following.

K-means clustering

K-means clustering is an approach for vector quantization. In particular, given a set of n vectors, k-means clustering groups them into k clusters (i.e., subsets) in such a way that each vector belongs to the cluster with the closest mean.The problem is computationally NP-hard, although suboptimal greedy algorithms have been developed.

K-means clustering can be used to group an unlabeled set of inputs into k clusters, and then use the centroids of these clusters to produce features. These features can be produced in several ways. The simplest is to add k binary features to each sample, where each feature j has value one iff the jth centroid learned by k-means is the closest tothe sample under consideration. It is also possible to use the distances to the clustersas features, perhaps after transforming them through a radial basis function (a technique that has been used to train RBF networks.

Principal component analysis

Principal component analysis (PCA) is often used for dimension reduction. Given an unlabeled set of n input data vectors, PCA generates p (which is much smaller than the dimension of the input data) right singular vectors corresponding to the p largest singular values of the data matrix, where the kth row of the data matrix isthe kth input data vector shifted by the sample mean of the input (i.e., subtracting the sample mean from the data vector). Equivalently, these singular vectors are the eigenvectors corresponding to the plargest eigenvalues of the sample covariance matrix of the input vectors. These p singular vectors are the feature vectors learned from the input data, and they represent directions along which the data has the largest variations.

Unsupervised dictionary learning

Unsupervised dictionary learning does not utilize data labels and exploits the structure underlying the data for optimizing dictionary elements. An example of unsupervised dictionary learning is sparse coding, which aims to learn basis functions (dictionary elements) for data representation from unlabeled input data.

Multilayer/deep architectures

The hierarchical architecture of the biological neural system inspires deep learning architectures for feature learning by stacking multiple layers of learning nodes. These architectures are often designed based on the assumption of distributed representation: observed data is generated by the interactions of many different factors on multiple levels. In a deep learning architecture, the output of each intermediate layer can be viewed as a representation of the original input data. Each level uses the representation produced by previous level as input, and produces new representations as output, which is then fed to higher levels. The input at the bottom layer is raw data, and the output of the final layer is the final low-dimensional feature or representation.

Restricted Boltzmann machine

Restricted Boltzmann machines (RBMs) are often used as a building block for multilayer learning architectures. An RBM can be represented by an undirected bipartite graph consisting of a group of binary hidden variables, a group

of visible variables, and edges connecting the hidden and visible nodes .

Autoencoder

An autoencoder consisting of an encoder and a decoder is a paradigm for deep learning architectures. An

example is provided by Hinton and Salakhutdinov[18] where the encoder uses raw data (e.g., image) as input and produces feature or representation as output and the decoder uses the extracted feature from the encoder as input and reconstructs the original input raw data as output.

Modeling Sequence Data:-

Applications of sequence modeling are plentiful in day-to-day business practice. Some of them emerged to meet today's challenges in terms of quality of service and customer engagement. Here some examples:

Speech Recognition to listen to the voice of customers.
Machine Language Translation from diverse source languages to more common languages.Topic Extraction to find the main subject of customer's translated query.
Speech Generation to have conversational ability and engage with customers in a human like manner. Text Summarization of customer feedback to work on key challenges and pain points.

In the auto industry, self-parking is also a sequence modeling task. In fact, parking could be seen as a sequence of mouvements where the next movement depends on the previous ones.

Other applications cover text classification, translating videos to natural language, image caption generation, hand writing recognition/generation, anomaly detection, and many more in the future…which none of us can think (or aware) at the moment.

However, before we go any further in the applications of Sequence Modeling, let us understand what we are dealingwith when we talk about sequences.

Introduction to Sequence Modelling

Sequences are a data structure where each example could be seen as a series of data points. This sentence: "I am currently reading an article about sequence modeling with Neural Networks" is an example that consists of multiple words and words depend on each other. The same applies to medical records. One single medical record consists in many measurments across time. It is the same for speech waveforms.
So why we need a different learning framework to model sequences and what are the special features that we are looking for in this framework?
For illustration purposes and with no loss of generality, let us focus on text as a sequence of words to motivate this need for a different learning framework.

In fact, machine learning algorithms typically require the text input to be represented as a **fixed-length** vector. Manyoperations needed to train the model (network) can be expressed through algebraic operations on the matrix of input feature values and the matrix of weights (think about a n-by-p design matrix, where n is the number of samples observed, and p is the number of variables measured in all samples).
Perhaps the most common fixed-length vector representation for texts is the **bag-of-words** or bag-of-n-grams due to its simplicity, efficiency and often surprising accuracy. However, the bag-of-words (BOW) representation has many disadvantages:
First, the word order is lost, and thus different sentences can have exactly the same representation, as long as the same words are used. Example: "The food was good, not bad at all." vs "The food was bad, not good at all.". Even though bag-of-n-grams considers the word order in short context, it suffers from data sparsity and high dimensionality.

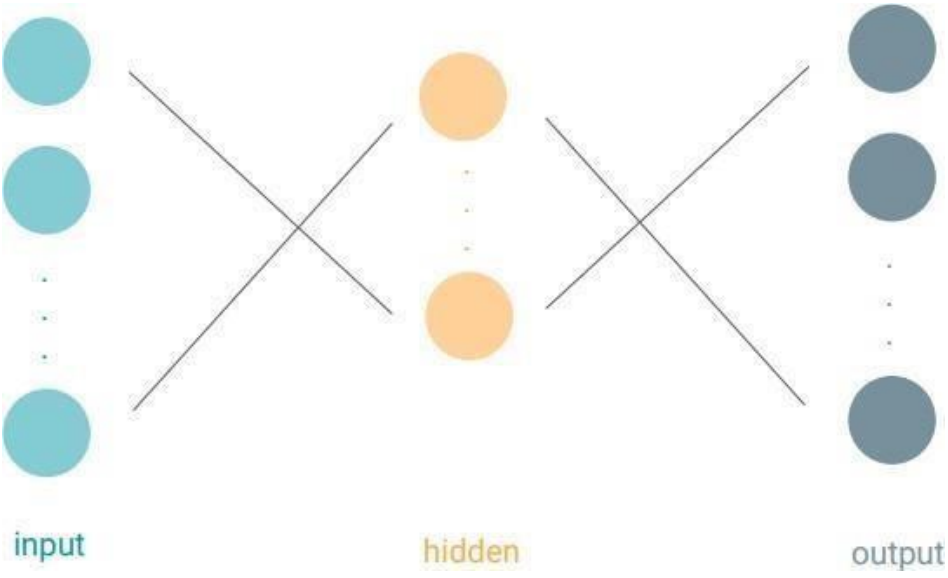
In addition, Bag-of-words and bag-of-n-grams have very little knowledge about the semantics of the words or more formally the distances between the words. This means that words “powerful”, “strong” and “Paris” are equally distant despite the fact that semantically, “powerful” should be closer to “strong” than “Paris”.
Humans don’t start their thinking from scratch every second. As you read this article, **you understand each word based on your understanding of previous words**. Traditional neural networks can’t do this, and it seems like a major shortcoming. Bag-of-words and bag-of-n-grams as text representations do not allow to keep track of long-term dependencies inside the same sentence or paragraph.

Another disadvantage of modeling sequences with traditional Neural Networks (e.g. Feedforward Neural Networks) is the fact of not sharing parameters across time. Let us take for example these two sentences : “On Monday, it was snowing” and “It was snowing on Monday”. These sentences mean the same thing, though the details are in different parts of the sequence. Actually, when we feed these two sentences into a Feedforward Neural Network for a prediction task, the model will assign different weights to “On Monday” at each moment in time. **Things we learn about the sequence won’t transfer if they appear at different points in the sequence.** Sharing parameters gives the network the ability to look for a given feature everywhere in the sequence, rather than in just a certain area.
Thus, to model sequences, we need a specific learning framework able to:

deal with variable-length sequencesmaintain sequence order
keep track of long-term dependencies rather than cutting input data too short share parameters across the sequence (so not re-learn things across the sequence)
Recurrent neural networks (RNNs) could address this issue. They are networks with loops in them, allowinginformation to persist.

So, let us find out more about RNNs! How a Recurrent Neural Network works?

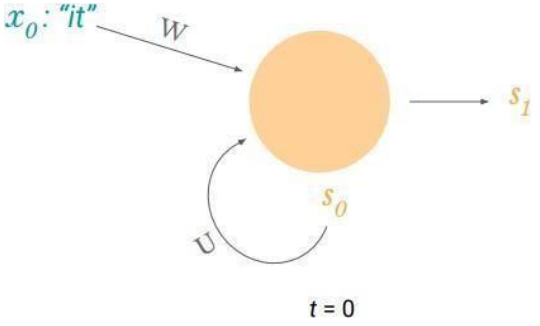
A Recurrent Neural Network is architected in the same way as a “traditional” Neural Network. We have someinputs, we have some hidden layers and we have some outputs.



The only difference is that each hidden unit is doing a slightly different function. So, let’s explore how this hiddenunit works.

A recurrent hidden unit computes a function of an input and its own previous output, also known as the cell

state. For textual data, an input could be a vector representing a word *x(i)* in a sentence of *n* words (also known as word embedding).



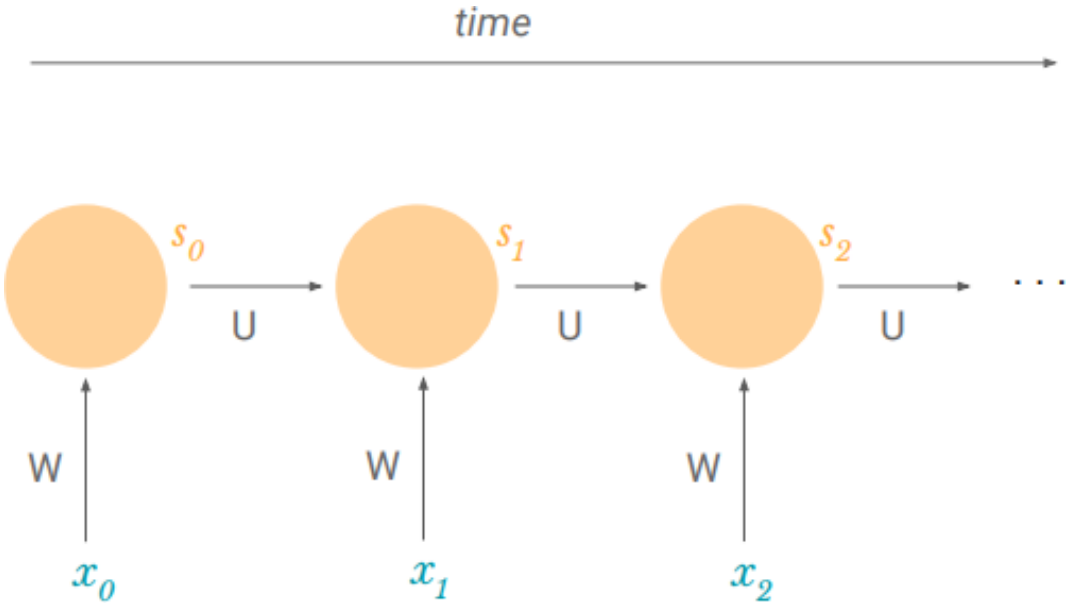
x_0 : vector representing first word
s_0 : cell state at $t = 0$ (some initialization)
s_1 : cell state at $t = 1$

$s_1 = tanh(Wx_0 + Us_0)$

W and *U* are weight matrices and *tanh* is the hyperbolic tangent function.

Similarly, at the next step, it computes a function of the new input and its previous cell state: ***s2*** = ***tanh*(*Wx1*+ *Us1*** . This behavior is similar to a hidden unit in a feed-forward Network. The difference, proper to sequences, is that we are adding an additional term to incorporate its own previous state.

A common way of viewing recurrent neural networks is by unfolding them across time. We can notice that **we are using the same weight matrices *W* and *U* throughout the sequence. This solves our problem of parameter sharing**. We don't have new parameters for every point of the sequence. Thus, once we learn something, it can apply at any point in the sequence.



The fact of not having new parameters for every point of the sequence also helps us **deal with variable-length sequences**. In case of a sequence that has a length of 4, we could unroll this RNN to four timesteps. In other cases, we can unroll it to ten timesteps since the length of the sequence is not prespecified in the algorithm. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word.

Introduction to Convolution Neural Network

It is assumed that the reader knows the concept of Neural networks. When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block regular Neural Network there are three types of

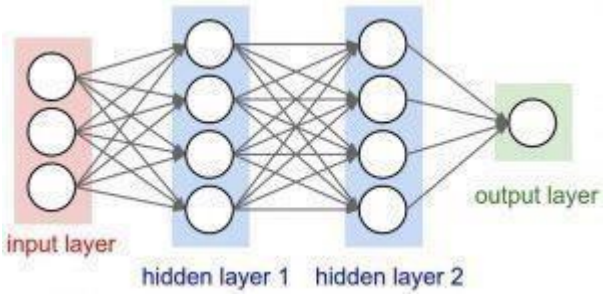
1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equalto the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation whichbasically isused tominimize the loss. Here's the basic python code for a neural network with random inputs and two hidden layers.

- Python

```
activation = lambda x: 1.0/(1.0 + np.exp(-x)) # sigmoid functioninput = np.random.randn(3, 1)
hidden_1 = activation(np.dot(W1, input) + b1)
hidden_2 = activation(np.dot(W2, hidden_1) + b2)output = np.dot(W3, hidden_2) + b3
```

1,W2,W3,b1,b2,b3 are learnable parameterofthmodel.

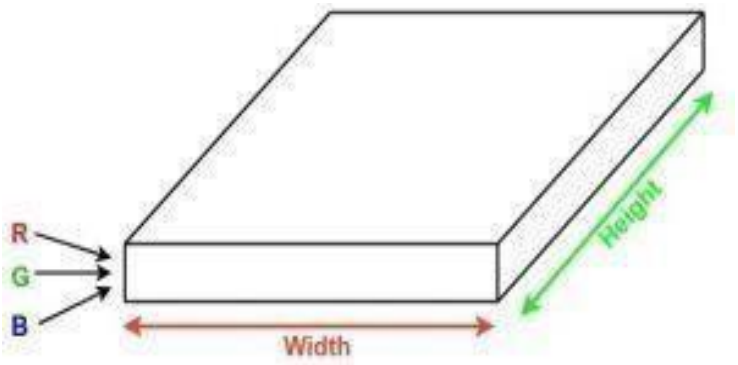


Convolution Neural Network

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as

images generally havered, green,andblue channels).

Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension 34x34x3. The possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

Layers used to build ConvNets

A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiablefunction.

Types oflayers:

Let's take an example by running a covnets on ofimage of dimension 32 x 3 x

1. **Input Layer:** This layer holds the raw input of the image with width 32, height 32, and depth 3.
2. **Convolution Layer:** This layer computes the output volume by computing the dot product between all filters and image patches. Suppose we use a total of 12 filters for this layer we'll get output volume of

dimension 32x 32 x 12.

3. **Activation Function Layer:** This layer will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.

4. **Pool Layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2,the resultanvolume will be of dimension .

UNIT - IV

Model Validation in Classification : Cross Validation - Holdout Method, K-Fold, Stratified K-Fold, Leave-One-Out Cross Validation. Bias-Variance tradeoff, Regularization , Overfitting, Underfitting. **Ensemble**

Methods: Boosting, Bagging, Random Forest.

Supervised Machine Learning: Model Validation, a Step by Step Approach

Model validation is the process of evaluating a trained model on test data set. This provides the generalization ability of a trained model. Here I provide a step by step approach to complete first iteration of model validation in minutes.

The basic recipe for applying a supervised machine learning model are:

Choose a class of model

Choose model hyper parametersFit the model to the training data
Use the model to predict labels for new data

*From Python Data Science Handbook by **Jake VanderPlas***

Jake VanderPlas, gives the process of model validation in four simple and clear steps. There is also a whole process needed before we even get to his first step. Like fetching all the information we need from the data to make a good judgement for choosing a class model. Also providing finishing touches to confirm the results after. I will get into depth about these steps and break it down further.

- Data cleansing and wrangling.
- Split the data into training and test data sets.
- Define the metrics for which model is getting optimized.
- Get quick initial metrics estimate.
- Feature engineering to optimize the metrics. (*Skip this during first pass).*
- Data pre-processing.
- Feature selection.
- Model selection.
- Model validation.
- Interpret the results.
- Get the best model and check it against test data set.

I will be using data set from **UCI Machine Learning Repository**. Data set is from the *Blood Transfusion Service Center* in Hsin-Chu City in Taiwan. This is a classification problem. The idea behind this extends to regression problem as well

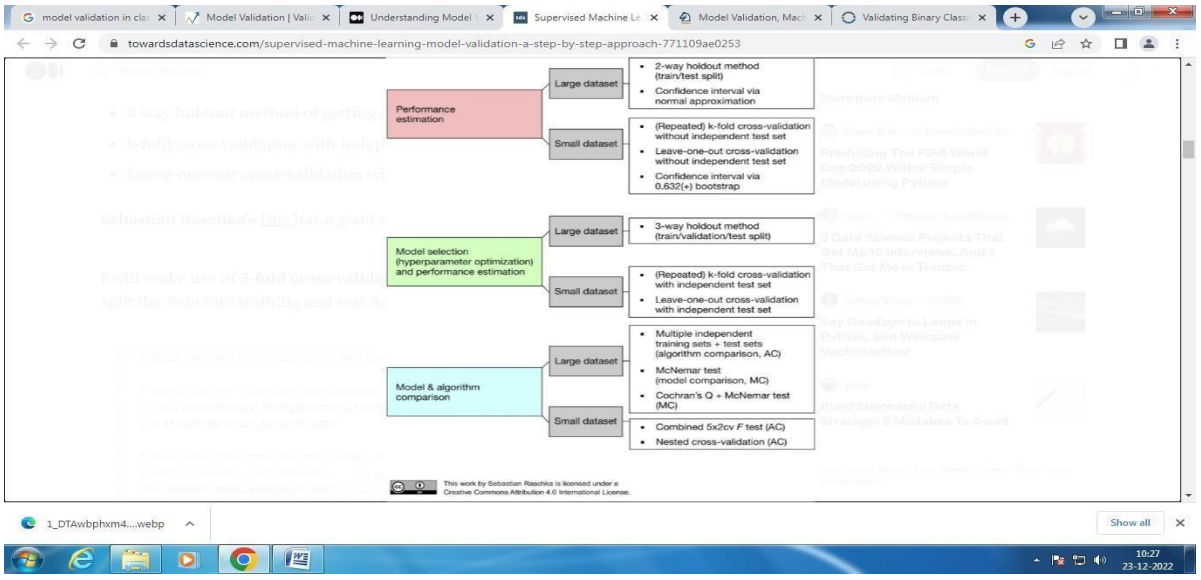
Data cleansing and wrangling.

Blood Transfusion Service Center Data Set is a clean data set. This will not be the case for most other data sets. So this is the step to inspect and clean up the data for example handling missing values…

Split the data into training and test data sets.

There are many ways to get the training and test data sets for model validation like:

- 3-way holdout method of getting training, validation and test data sets.
- k-fold cross-validation with independent test data set.
- Leave-one-out cross-validation with independent test data set.



The main idea behind this step is to get the baseline estimate of metrics which is being optimized. This baseline will work as reference in further steps of model validation. There are several ways to get the baseline estimate for classification problem. I am using the majority class for prediction. The baseline accuracy score is approximately 77%.

```
# Get
quick initial metrics
estimate.
# Using simple pandas value counts method
print(y_train.value_counts(normalize=True))
# Using sklearn accuracy_score
import numpy as np
from sklearn.metrics import accuracy_score
```

```
majority_class = y_train.mode()[0]
prediction = np.full(shape=y_train.shape,
fill_value=majority_class)
accuracy_score(y_train, prediction)
```

Feature engineering to optimize the metrics.

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.

From Wikipedia

This means identifying the relationships between independent and dependent features. This is with the help of graphs like pair plots or correlation matrix. Then the identified relationships we can add as polynomial or interaction features.

Feature engineering step is the point of entry for successive iterations. This is a critical step and plays a greater rolein predictions as compared to model validation.

As a quick solution we can throw in some polynomial features using *PolynomialFeatures in sci-kit learn*.

Domain knowledge on the problem in hand will be of great use for feature engineering. This is a bigger topic in itselfand requires extensive investment of time and resource.

Data pre-processing.

Data pre-processing converts features into format that is more suitable for the estimators. In general, machine learning model prefer standardization of the data set. I will make use of **RobustScaler** for our example.

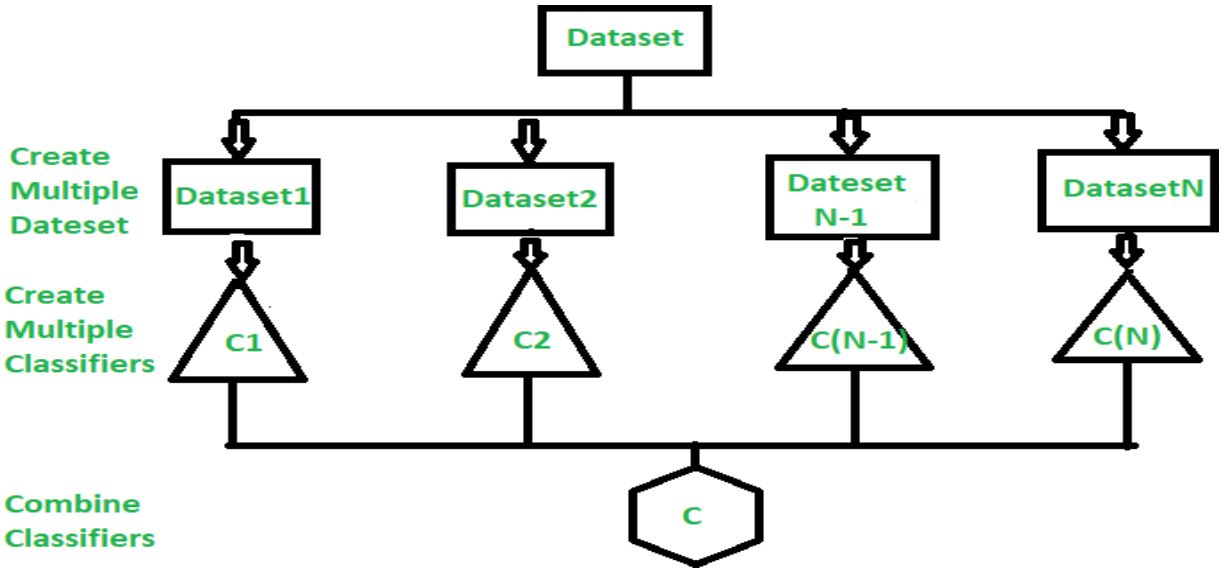
Refer to sci-kit learn's *Preprocessing data* section for detailed information.

Feature selection.

Feature selection or dimensionality reduction on data sets helps to

- Either to improve models' accuracy scoresor
- To boost their performance on very high-dimensional data sets.

I will use **SelectKBest**, univariate feature selection method. The scoring function used for classification andregression problems will vary.



Why do ensembles work?

Dietterich(2002) showed that ensembles overcome three problems –

- **Statistical Problem –**

The Statistical Problem arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!

- **Computational Problem –**

The Computational Problem arises when the learning algorithm cannot guarantees finding the best hypothesis.

- **Representational Problem –**

The Representational Problem arises when the hypothesis space does not contain any good approximation of thetarget class(es).

Main Challenge for Developing Ensemble Models?

The main challenge is not to obtain highly accurate base models, but rather to obtain base models which make different kinds of errors. For example, if ensembles are used for classification, high accuracies can be accomplished if different base models misclassify different training examples, even if the base classifier accuracy is low.

Methods for Independently Constructing Ensembles –

- *Majority Vote*
- *Bagging and Random Forest*

- *Randomness Injection*
- *Feature-Selection Ensembles*
- *Error-Correcting Output Coding*

Methods for Coordinated Construction of Ensembles

UNIT – V

Unsupervised Learning : Clustering-K-means, K-Modes, K-Prototypes, Gaussian

Mixture Models,Expectation-Maximization.

Reinforcement Learning: Exploration and exploitation trade-offs, non-associative learning, Markov decision processes, Q-learning.

Unsupervised Machine Learning:

Introduction to clustering

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

"*Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.*"

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervisedlearning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their ownexperiences, which makes itcloser to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which makeunsupervised learning moreimportant.

- In real-world, we do not always have input data with the corresponding output so tosolve such cases, weneed unsupervised learning.

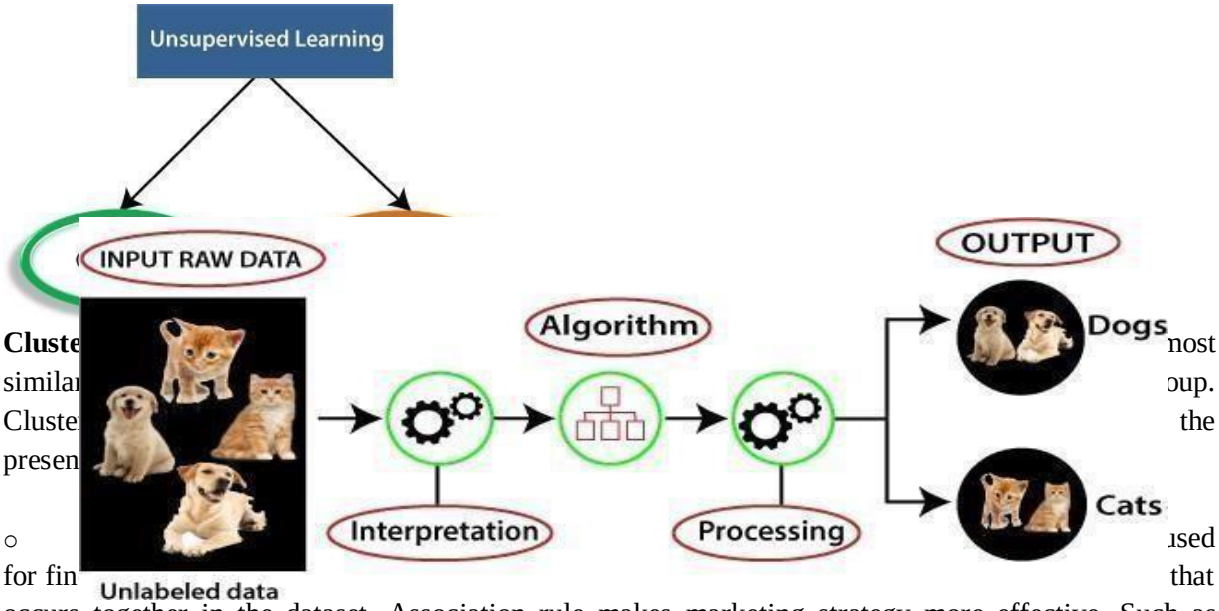
Working of Unsupervised Learning

WorkHere, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k- means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groupsaccording to the similaritiesand difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



Cluste nost similar oup. Cluste the presen

- ısed for fin that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

K-means clustering of unsupervised learning can be understood by the below diagram:

- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Unsupervised learning is used for more complex tasks as compared to supervisedlearning because, inunsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning asit does not havecorresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, andalgorithms do not know the exact output in advance.

k-means clustering algorithm

One of the most used clustering algorithm is *k-means*. It allows to group the data according to the existingsimilarities among them in *k* clusters, given as input to the algorithm. I‟ll start with asimple example.

Let‟s imagine we have 5 objects (say 5 people) and for each of them we know two features (height and weight). Wewant to group them into *k=2* clusters.

Our dataset will look like this:

How to apply k-means?

As you probably already know, I‟m using Python libraries to analyze my data. The *k-means* algorithm is implemented in the *scikit-learn* package. To use it, you will just need the following line in your script:

What if our data is… non-numerical?

At this point, you will maybe have noticed something. The basic concept of *k-means* stands on mathematical calculations (means, euclidian distances). But what if our data is non-numerical or, in other words, *categorical*? Imagine, for instance, to have the ID code and date of birth of the five people of the previous example, instead of their heights and weights.

We could think of transforming our categorical values in numerical values and eventually apply *k-means*. But beware: *k-means* uses numerical distances, so it could consider close two really distant objects that merely have been assigned two close numbers.

k-modes is an extension of *k-means*. Instead of distances it uses *dissimilarities* (that is, quantification of the total mismatches between two objects: the smaller this number, the more similar the two objects). And instead of means, it uses *modes*. A mode is a vector of elements that minimizes the dissimilarities between the vector itself and each object of the data. We will have as many modes as the number of clusters we required, since they act as centroids.

Reinforcement learning

Reinforcement learning addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals

Introduction

- Consider building a **learning robot**. The robot, or ***agent***, has a set of sensors to observe the state of itsenvironment, and a set of actions it can perform to alter this state.
- Its task is to learn a control strategy, or ***policy***, for choosing actions that achieve its goals.
- The goals of the agent can be defined by a ***reward function*** that assigns a numericalvalue to each distinctaction the agent may take from each distinct state.
- This reward function may be built into the robot, or known only to an external teacher whoprovides thereward value for each action performed by the robot.
- The **task** of the robot is to perform sequences of actions, observe their consequences,and learn a controlpolicy.
- The control policy is one that, from any initial state, chooses actions that maximize thereward accumulatedover time by the agent.

Example:

- A mobile robot may have sensors such as a camera and sonars, and actions such as "move forward"and "turn."
- The robot may have a goal of docking onto its battery charger whenever its battery level is

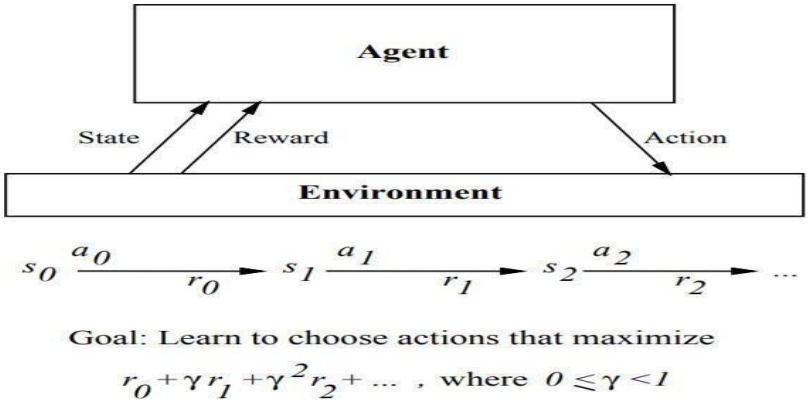
low.

- The goal of docking to the battery charger can be captured by assigning a positive reward (Eg., +100) to state- action transitions that immediately result in a connection to the charger and a reward of zero to every other state-action transition.

Reinforcement Learning Problem

- An agent interacting with its environment. The agent exists in an environment described by some set of possible states S.
- Agent perform any of a set of possible actions A. Each time it performs an action a, in some state st the agent receives a real-valued reward r, that indicates the immediate value of this state-action transition. This produces a sequence of states si, actions ai, and immediate rewards ri as shown in the figure.

The agent's task is to learn a control policy, $\boldsymbol{\pi}$**: S** $\rightarrow$ **A**, that maximizes the expected sum of these rewards, withfuture rewards discounted exponentially by their delay.



Reinforcement learning problem characteristics

1. **Delayed reward**: The task of the agent is to learn a target function π that maps from the current state s to the optimal action a = π (s). In reinforcement learning, training information is not available in (s, π (s)). Instead, the trainer provides only a sequence of immediate reward values as the agent executes its sequence of actions. The agent, therefore, faces the problem of ***temporal credit assignment***: determining which of the actions in its sequence are to be credited with producing the eventual rewards.

2. **Exploration:** In reinforcement learning, the agent influences the distribution of training examples by the action sequence it chooses. This raises the question of which experimentation strategy produces most effective learning. The learner faces a trade-off in choosing whether to favor exploration of unknown states and actions, or exploitation of states and actions that it has already learned will yield high reward.

3. **Partially observable states:** The agent's sensors can perceive the entire state of the environment at each time step, in many practical situations sensors provide only partial information. In such cases, the agent needs to consider its previous observations together with its current sensor data when choosing actions, and the best policy may be onethat chooses actions specifically to

improve the observability of the environment.

4. **Life-long learning:** Robot requires to learn several related tasks within the same environment,using the same sensors. For example, a mobile robot may need to learn how to dock on its battery charger, how to navigate through narrow corridors, and how to pick up output from laser printers. This setting raises the possibility of using previously obtained experience or knowledge to reduce sample complexity when learning new tasks.

Learning Task

Consider Markov decision process (MDP) where the agent can perceive a set S of distinct states of itsenvironment and has a set A of actions that it can perform

- At each discrete time step t, the agent senses the current state st, chooses a current action at, andperforms it.
- The environment responds by giving the agent a reward rt = r(st, at) and by producing the succeedingstate st+l

= δ(st, at). Here the functions δ(st, at) and r(st, at) depend only on the current state and action, and not onearlier states or actions.

The task of the agent is to learn a policy, $\boldsymbol{\pi: S \rightarrow A}$, for selecting its nextaction a, based on the current observedstate st; that is, $\boldsymbol{\pi(st) = at}$.

How shall we specify precisely which policy π we would like the agent to learn?

1. One approach is to require the policy that produces the greatest possible ***cumulative reward*** for the robot overtime.

[To state this requirement more precisely, define the cumulative valueVπ (st) achieved by following an arbitrary policy π from an arbitrary

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \qquad \textbf{equ (1)}$$

initial state st as follows:

[Where, the sequence of rewards rt+i is generated by beginning atstate st and by repeatedly using the policy π to select actions.

[Here $0 \leq \gamma \leq 1$ is a constant that determines the relative value of delayed versusimmediate rewards. if we set γ

= 0, only the immediate reward is considered. As we set γ closer to 1, future rewards are given greater emphasis relative to the immediate reward.

[The quantity Vπ (st) is called the ***discounted cumulative reward*** achieved by policy π

from initial state ***s***. It is reasonable to discount future rewards relative to immediate rewards because, in many cases, we prefer to obtain the reward sooner rather than later.

2. Other definitions of total reward is ***finite horizon reward,***

$$\sum_{i=0}^{h} r_{t+i}$$

Considers the undiscounted sum of rewards over a finite number ***h*** of steps

3. Another approach is ***average reward***

$$\lim_{h\to\infty} \frac{1}{h} \sum_{i=0}^{h} r_{t+i}$$

Considers the average reward per time step over the entire lifetime of the agent.

We require that the agent learn a policy π that maximizes Vπ (st) for allstates s. such a policy is called an ***optimalpolicy*** and denote it by π*

Refer the value function Vπ*(s) an optimal policy as V*(s). V*(s) gives the maximum discounted cumulative rewardthat the agent can obtain starting from state ***s***.

Example:

A simple grid-world environment is depicted in the diagram

- The six grid squares in this diagram represent six possible states, or locations,for the agent.
- Each arrow in the diagram represents a possible action the agent can take tomove from one state to another.
- The number associated with each arrow represents the immediate reward ***r(s, a)*** the agent receives if it executesthe corresponding state-action transition
- The immediate reward in this environment is defined to be zero forall state-action transitions except for those leading into the state labelled G. The state G as the goal state, and the agent can receive reward by entering thisstate.

Once the states, actions, and immediate rewards are defined, choose a value for the discount factor γ, determine theoptimal policy π * and itsvalue function V*(s).

Let's choose γ = 0.9. The diagram at the bottom of the figure shows one optimal

policy for this setting.

Values of V*(s) and Q(s, a) follow from r(s, a), and the discount factor γ = 0.9. An optimal policy, corresponding toactions with maximal Q values,is also shown.

The discounted future reward from the bottom centre state is
0+ γ 100+ γ2 0+ γ3 0+... = 90

Q LEARNING

*How can an agent learn an optimal policy π * for an arbitrary environment?*

The training information available to the learner is the sequence of immediate rewards r(si,ai) for i = 0, 1,2, ...
Given this kind of training information it is easier to learn a numerical evaluation function defined over states andactions, then implement the optimal policy in terms of this evaluation function.
What evaluation function should the agent attempt to learn?

One obvious choice is V*. The agent should prefer state sl over state s2 whenever V*(sl) > V*(s2), because thecumulative future reward will begreater from sl
The optimal action in state ***s*** is the action ***a*** that maximizes the sum of the immediate reward r(s, a) plus the value V*of the immediate successor state, discounted by γ.

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))] \qquad \textbf{equ (3)}$$

The Q Function

The value of Evaluation function ***Q(s, a)*** is the reward receivedimmediately upon executing action a from state ***s***,plus the value (discounted by ***γ***) of following the optimal policy thereafter

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \qquad \textbf{equ (4)}$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) \qquad \textbf{equ (5)}$$

Rewrite Equation (3) in terms of ***Q(s, a)*** as
Equation (5) makes clear, it need only consider each available action ***a***

in its current state ***s*** and choose the actionthat maximizes ***Q(s, a)***.

An Algorithm for Learning *Q*

- Learning the Q function corresponds to learning the **optimal policy**.

- The key problem is finding a reliable way to estimate training valuesfor ***Q***, given only a sequence of immediaterewards ***r*** spread out over

$$V^*(s) = \max_{a'} Q(s, a')$$

time. This can be accomplished through ***iterative approximation***

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$$

Rewriting Equation

- Q learning algorithm:

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

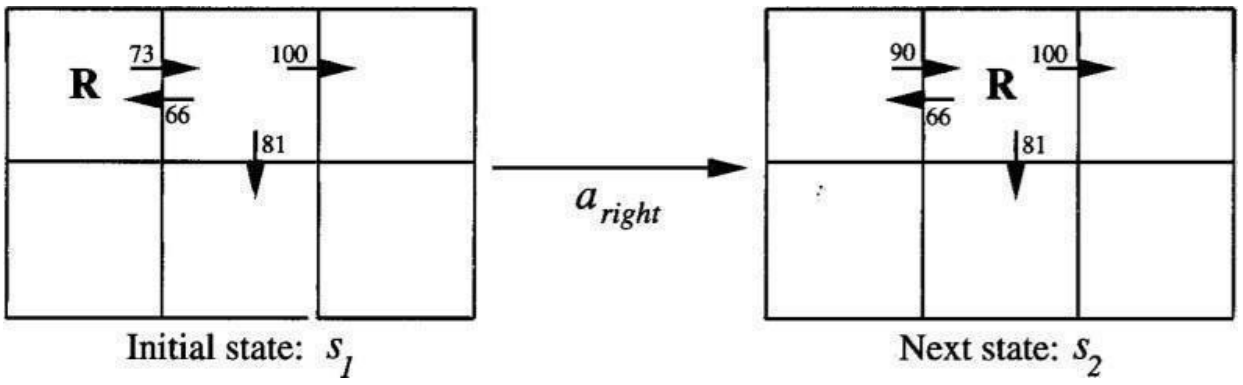
Q learning algorithm assuming deterministic rewards and actions.
The discount factor γ may be any constantsuch that $0 \leq \gamma < 1$

- Q to refer to the learner's estimate, or hypothesis, of the actual Q function

An Illustrative Example

- To illustrate the operation of the Q learning algorithm, consider a single action taken by an agent, and thecorresponding refinement to

Q shown in below figure



- The agent moves one cell to the right in its grid world and receives an immediate reward of zero for thistransition.
- Apply the training rule of Equation

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

to refine its estimate Q for the state-action transition it just executed.

$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$
$$\leftarrow 0 + 0.9 \max\{66, 81, 100\}$$
$$\leftarrow 90$$

According to the training rule, the new Q estimate for this transitionis the sum of the received reward (zero)and the highest Q value associated with the resulting state (100), discounted byγ (.9).

Convergence

Will the Q Learning Algorithm converge toward a Q equal to the true Q function?

Yes, under certain conditions.

1. Assume the system is a deterministic MDP.

2. Assume the immediate reward values are bounded; that is, there exists some positive constant c such that for allstates s and actions a, **| r(s, a)| < c**

3. Assume the agent selects actions in such a fashion that it visits every possible state-action pair infinitely often

Here are four machine learning trends that could become a reality in the near future:

1) Intelligence on the Cloud

Algorithms can help companies unearth insights about their business, but this proposition can be expensive with no guarantees of a bottom-line increase. Companies often deal with havingto collect data, hire data scientists and train them to deal with changing databases. Now that more data metrics are becoming available, the cost to store it is dropping thanks to the cloud. There will no longer be the need to manage infrastructure as cloud systems can generate new models as the scale of an operation increases, while also delivering more accurate results. More open-source ML frameworks are coming to the fold, obtaining pre-trained platforms thatcan tag images, recommend products and perform natural language processing tasks.

2) Quantum Computing Capabilities

Some of the tasks that ML can help companies deal with is the manipulation and classification of large quantities of vectors in high-dimensional spaces. Current algorithms take a large chunk of time to solve these problems, costing companies more to complete their business processes. Quantum computers are slated to become all the rage soon as they can manipulate high-dimensional vectors at a fraction of the time. These will be able to increase the number of vectors and dimensions that are processed when compared to traditional algorithms in a quicker period of time.

3) Improved Personalization

Retailers are already making waves in developing recommendation engines that reach their target audience more accurately. Taking this a step further, ML will be able to improve the personalization techniques of these engines in more precise ways. The technology will offer more specific data that they can then use on ads to improve the shopping experience for consumers.

4) Data on Data

As the amount of data available increases, the cost of storing this data decreases at roughly thesame rate. ML has great potential in generating data of the highest quality that will lead to better models, an improved user experience and more data that helps repeat but improve uponthis cycle. Companies such as Tesla add a million miles of driving data to enhance its self- driving capabilities every hour. Its Autopilot feature learns from this data and improves the software that propels these self-driving vehicles forward as the company gathers more data onthe possible pitfalls of autonomous driving technology.