ML

Unit - 3

1) Naive Bayes Classifier Algorithm

↳ Supervised learning algorithm
↳ Based on Bayes theorem
↳ Used to classify samples (classification problems)
↳ It is a probabilistic classifier

Bayes' theorem:
↳ It is used to find the probability of hypothesis with prior knowledge

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

P(A/B) is posterior probability : Probability of hypothesis A on the observed event B

P(B/A) is likelihood probability : Probability of evidence given that probability of hypothesis is true

P(A) is prior probability : Probability of hypothesis A before observing evidence

P(B) is marginal probability : Probability of Evidence

Working:

1. Convert the given dataset into frequency tables
2. Generate likelihood table by finding probabilities of given features
3. Use Bayes theorem to calculate posterior probability.

ML

1) Types of ML

- Machine learning
  - Based on supervision
    - Supervised
    - Unsupervised
    - Semisupervised
    - Reinforcement
  - Based on increment learning
    - Online learning
    - Batch learning
  - Based on detection of patterns
    - Instance based
    - Model based

Based on Supervision

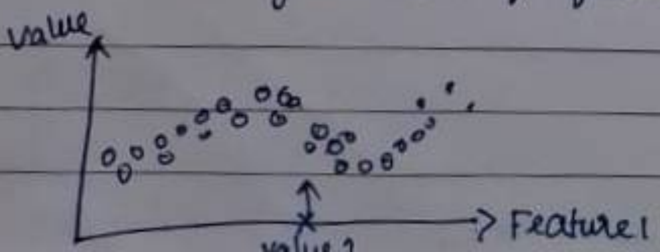
1) Supervised learning

The training data we provide to the algorithm Includes desired solutions, called labels

Eg: spam classification

Tasks in supervised learning

* Classification
* Regression - Prediction of target numerical value with given set of features



Overfitting

* Model doesn't make accurate prediction
* when a model gets trained with much data, it starts learning from noise in datasets
* It cannot categorise correctly because of noise
* Techniques to reduce:
    1) Increase training data
    2) Reduce model complexity
    3) Ridge regularization & Lasso Regularization
    4) Early stopping during training phase
    5) Use dropouts for neural networks

2)

Some algorithms:

* k-nearest neighbours
* Neural networks
* Linear Regression
* Decision trees & random forests
* Logistic Regression

2) Unsupervised learning:

The system tries to learn without teacher, the training dataset is unlabelled.

Tasks:

* Clustering → try to detect a group of similar i/ps
* Visualization → Preserves as much structure as algorithm can
* Dimensionality reduction → simplify the data/ feature extraction
* Anomaly detection → Removing outliers

Some algorithms:

* Clustering → k-means, DBSCAN
* Visualization & dimensionality reduction → PCA, kernel PCA
* Anomaly detection → Isolation Forest

3) Semi-supervised

The system is provided with little labeled data and a lot of unlabelled data.

Eg: Google photos

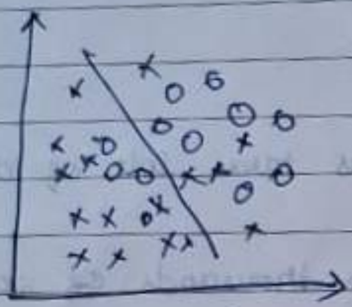
Working:

→ Cluster similar data using algorithm and use existing labelled data to label the remaining unlabelled data.

Overfitting and underfitting

Terms to be known:

1) Bias: Assumptions made by the model to make function easier (Actually the training error)
2) Variance: Difference between error rate of training data and testing data. It should be low



underfitting

overfitting

Appropriate fitting

Underfitting

* A machine learning algorithm is said to be underfitting when it works well on training data and performs poor on testing data.
* Reasons:
  1) High bias low variance
  2) Data not enough
  3) Too simple model
  4) Training data contain noise
* How to reduce
  1) Increase model complexity
  2) Increase no. of features
  3) Remove noise from data
  4) Increase no. of epochs

4) Reinforcement

* It is different from supervised learning, in reinforcement learning there is no answer key but the reinforcement agent decides what to do to perform task
* In the absence of training data it learns by experience

Based on increment learning

1) Batch learning / offline learning

First the system is trained, and launched into testing and runs without learning anymore. It just apply what it has learned.

2) Online learning

We train system incrementally by providing data sequentially either individually or small groups.

In case of bad data, performance gradually decrease

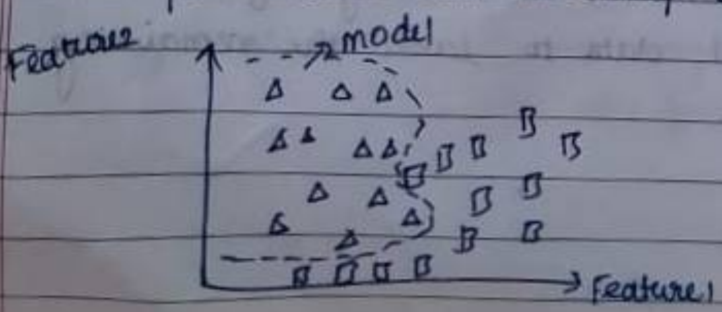
Based on (detection of patterns) generalization

1) Instance based learning

The system learns examples by heart, the generalizes new cases by comparing them to learned examples

2) Model-based

Building a model with examples and make predictions with the help of it.



2) Challenges in Machine learning
* Insufficient Quantity of data
* Non reprensative training data
* Poor Quality of data
* Irrelevant features
* Overfitting
* Underfitting

Insufficient Quantity of data:

* Machine learning algorithms takes lots of data to work.
* Even simple problem needs thousands of examples, complex problems like image & Sound needs millions of examples.

Non representative training data:

* If sample is too small sampling noise will occur, this might also occur in case of large data due to sampling method is flawed. This is called Sampling bias
* The samples might tend to select a portion but that is not accurate. This is called skewness
* Absence of data in some surveys. This is called nonresponse bias
* Transfer learning is used in Deep learning & neural networks where a pre-built model is used on our dataset.

Irrelevant features:
* Feature selection: Selection more useful feature among existing features to train the mode
* Feature extraction: Combing existing features to create new more useful feature to train model

Applying Bayes theorem

$$P(yes/Sunny) = \frac{P(Sunny/yes)\ P(yes)}{P(Sunny)}$$

$$= \frac{3/10 \times 10/14}{5/14}$$

$$P(yes/Sunny) = 3/5$$

$$P(No/Sunny) = \frac{P(Sunny/No)\ P(No)}{P(Sunny)}$$

$$= \frac{2/4 \times 4/14}{5/14}$$

$$P(No/Sunny) = 2/5$$

$$P(Yes/Sunny) > P(No/Sunny)$$

∴ The player should play

Example 2:

For multiple attributes

Say, today = (Sunny, Hot, Normal, False)

$$P(Yes/today) = \frac{P(Sunny/yes) \cdot P(Hot/yes) \cdot P(Normal/yes) \cdot P(False/yes) \cdot P(Yes)}{P(today)}$$

$$P(No/today) = \frac{P(Sunny/No) \cdot P(Hot/No) \cdot P(Normal/No) \cdot P(False/No) \cdot P(No)}{P(today)}$$

Advantages

* Fast and easy ML algorithm to classify
* Used for binary as well as multi-class classification
* Most popular choice for text classification problems

Disadvantages

* Cannot learn relationship between features, it assumes all features are independent

Applications

* Credit scoring
* medical data classification
* Real-time predictions
* Spam filtering & Sentiment analysis.

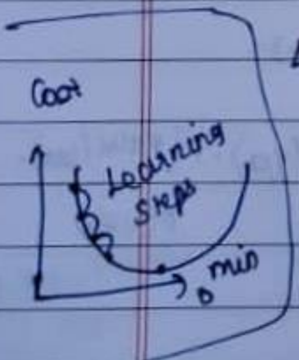
Advantages
* Fast and easy ML algorithm to classify
* Used for binary as well as multi-class classification
* Most popular choice for text classification problems

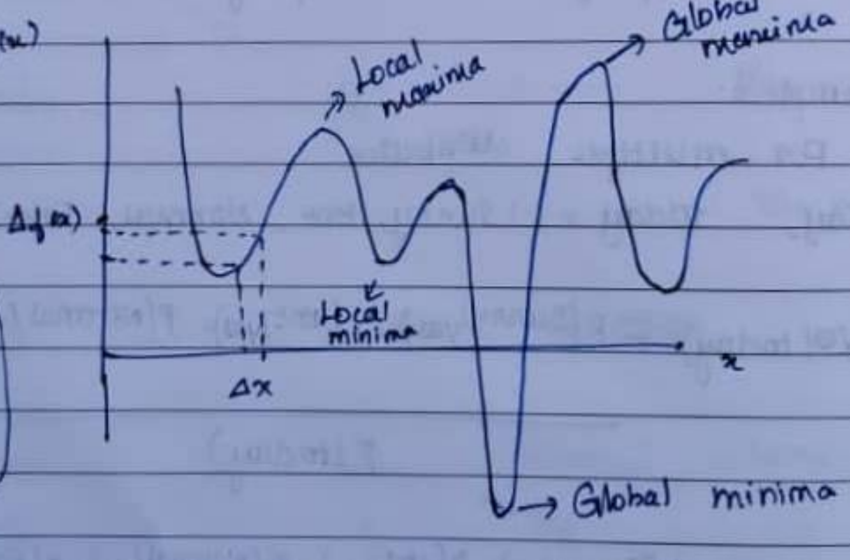
Disadvantages
* Cannot learn relationship between features, it assumes all features are independent

Applications
* Credit scoring
* medical data classification
* Real-time predictions
* Spam filtering & Sentiment analysis.

2) Gradient Descent

* Gradient descent is defined as one of the most commonly used iterative optimization algorithms to train deep learning and machine learning models.
* It helps in finding local minimum of a function





* Any function has one/more optima, finding optima requires gradients of the functions.

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

* Rules of Derivatives

1. Sum rule : $(f(x) + g(x))' = f'(x) + g'(x)$
2. Product rule : $(f(x) \cdot g(x))' = f'(x) \cdot g(x) + f(x) \cdot g'(x)$
3. Quotient rule : $(f(x) / g(x))' = \frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{(g(x))^2}$
4. Scaling rule : $(a \cdot f(x))' = a \cdot f'(x)$
5. Chain rule

* Types

1. Batch Gradient Descent:
   ↳ Processes all training examples for each iteration
   ↳ Very expensive for large training examples
   ↳ So stochastic or mini-batch Gradient descent is used for large examples

2. Stochastic Gradient Descent
   ↳ Processes one training example per iteration
   ↳ Faster than Batch GD
   ↳ When number of examples increase, number of iterations will be quite large

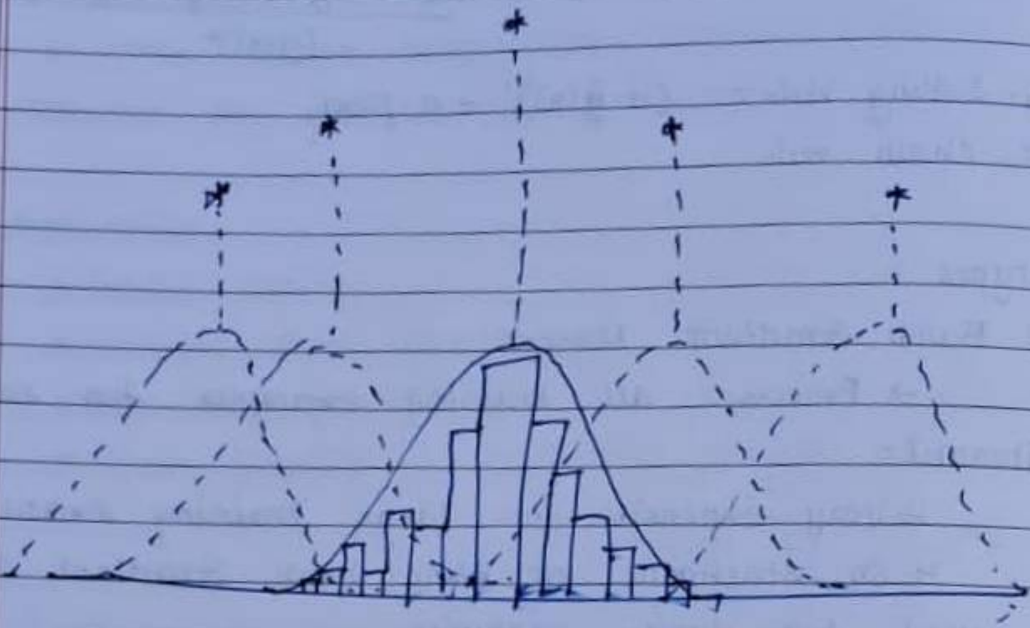
3. Mini-Batch GD
   ↳ Faster than both Batch and Stochastic GD
   ↳ It processes b examples per iteration $(b < m)$
   ↳ If the no. of training examples is large, it is processed in batches of b training examples.

* Requirements

A function has to be differentiable and convex to apply Gradient descent algorithm.

3) Maximum Likelihood Estimation

* It is the method for determining the parameters (mean, SD) of normally distributed random sample data (or) method of finding best fitting PDF of random sample data



Maximum likelihood estimation plot

* The figure shows multiple attempts to fit the probability density function (PDF) bell curve over sample data
* The dotted bell curve indicate poorly fitted PDF and lined bell curve indicated perfectly fitted PDF.
* The lined bell curve has maximum likelihood estimator.
* This is how maximum likelihood estimation works

$$L = F([X_1 = x_1], [X_2 = x_2], \dots [X_n = x_n] P)$$

L → likelihood value

F → PDF

$X_1, X_2 \dots X_n$ → Random samples

$x_1, x_2 \dots x_n$ → Value of random samples

P → Probability

K means Clustering – Introduction

K-Means Clustering is an Unsupervised Machine Learning algorithm, which groups the unlabeled dataset into different clusters.

K means Clustering

Unsupervised Machine Learning is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine's job in this case is to organize unsorted data according to parallels, patterns, and variations.

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm; an unsupervised learning algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

(It will help if you think of items as points in an n-dimensional space). The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The "points" mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature *x*, the items have values in [0,3], we will initialize the means with values for x at [0,3]).

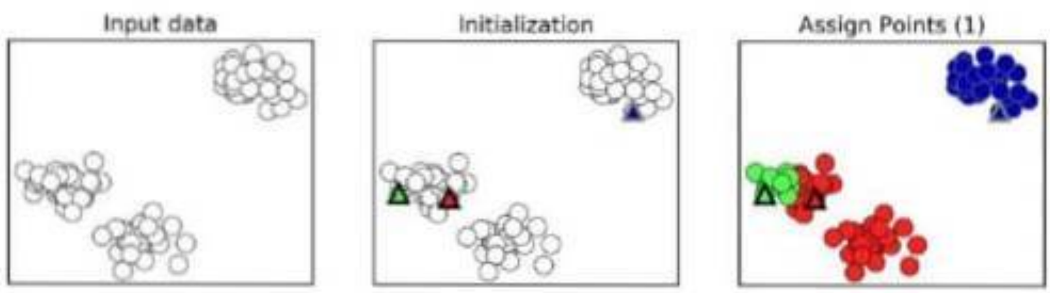
k-Means Clustering

It tries to find cluster centers that are representative of certain regions of the data. The algorithm alternates between two steps:

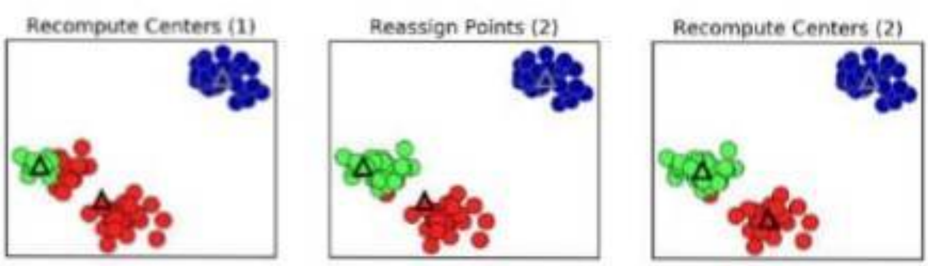
- assigning each data point to the closest cluster center, and then
- setting each cluster center as the mean of the data points that are assigned to it.

The algorithm is finished when the assignment of instances to clusters no longer changes.

k-Means Clustering

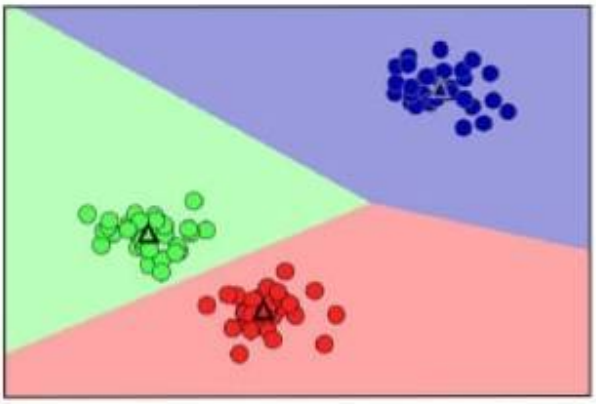


k-Means Clustering



The cluster centers are updated to be the mean of the assigned points

Cluster Boundaries



k-means with scikit-learn

In[49]:

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# generate synthetic two-dimensional data
X, y = make_blobs(random_state=1)

# build the clustering model
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

Failure cases of k-means

Each cluster is defined solely by its center, which means that each cluster is a convex shape.

As a result of this, *k*-means can only capture relatively simple shapes. *k*-means also assumes that all clusters have the same "diameter" in some sense;

it always draws the boundary between clusters to be exactly in the middle between the cluster centers.

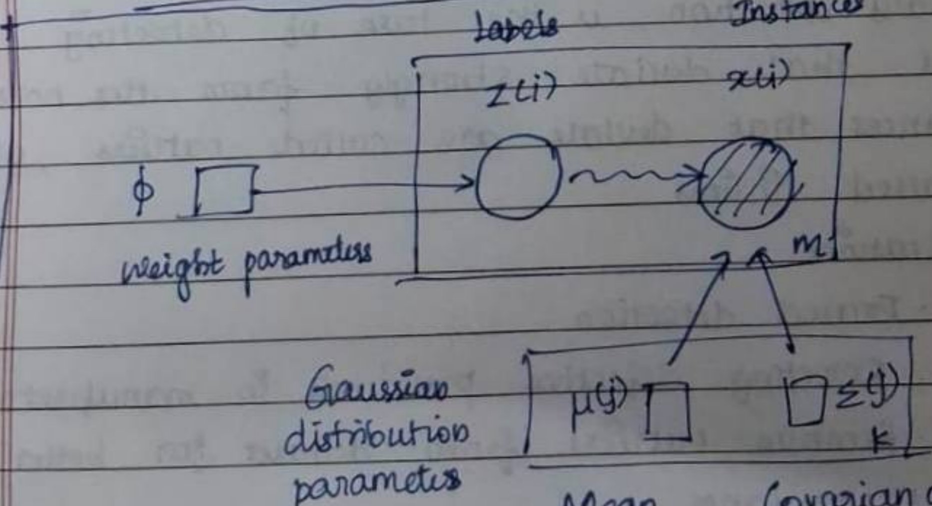
Gaussian mixtures

* Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the instances are generated from several Gaussian distributions whose parameters are unknown
* All the instances generated from a Gaussian distribution form a cluster that typically look like an ellipsoid.
* The dataset x is assumed to have generated by following probabilistic process.

1. For each instance, a cluster is choosen randomly from k clusters. The probability of choosing j^{th} cluster depends on its weight $\phi(j)$. The index of cluster chosen for i^{th} index is denoted as $z(i)$.
2. If $z(i) = j$, meaning i^{th} instance is assigned to j^{th} cluster, the location $x(i)$ of this index is sampled randomly from the Gaussian distribution with mean $\mu(j)$ & covariance $\Sigma(j)$.

$$x(i) \sim \mathcal{N}(\mu(j), \Sigma(j))$$

Gaussian mixture model:



$z^{(i)} \sim \text{Categorical}(\phi)$

$x(i) \sim \mathcal{N}(\mu_z(j), \Sigma_z(j))$

classmate

Date
Page

Interpretation

* The circles represent random variables
* Squares represent fixed values
* Large rectangles are called plates, they indicate that their content is repeated several times
* The number at bottom right of each plate indicate how many times its content is repeated (m & k)
* Each variable z(i) is drawn from categorical distribution with weight ϕ. Each variable x(i) is drawn from normal distribution with mean and covariance defined.
* Solid arrow represents conditional dependencies
* The squiggly arrow from z(i) to x(i) represents a switch: depending on z(i), the instance x(i) will be sampled from different Gaussian distribution
* Shaded nodes indicate that value is known [only variables x(i) have known values], they are called observed variables. The unknown random variables z(i) are called latent variables.

Anomaly detection using Gaussian mixtures

* Anomaly detection is the task of detecting instances that deviate strongly from the norm
* Instances that deviate are called outliers, others are called inliers
* Applications
    1. Fraud detection
    2. Detecting defective products in manufacturing
    3. Remove outliers from dataset for better performance

	OR			
12. (b)	i) For the dataset set collected on a population as shown in Table.1, the problem statement is to predict what range car a person will buy. Identify whether this is a regression or classification problem supervised / unsupervised / Semi supervised learning Multivariate/univariate Online or offline learning Instance based or model based learning **(7 Marks)** ii) Illustrate how the data is prepared for machine learning algorithm **(8 Marks)**	CO2	Ap	Ap

Gender	Profession	Income level	Marital status	No. of children	Owner ship of four wheeler *
Male	Yes	Low	Y	2	0
Male	No	Medium	N	0	1
Female	Yes	High	N	0	2
Female	Yes	Low	Y	1	0
Male	No	Low	Y	2	0
Female	No	Medium	Y	2	1
Male	Yes	High	N	0	3

0- owns no car

1- owns car value less than 5 lakhs

2- owns car value between 5 -10 lakhs

3- owns car value between 10-20 lakhs

Machine

earning

arning models.

e model is working

→ AUC

re