UNIT III

Bayes Theorem in Bayesian Learning:

Machine learning aims to determine the best hypothesis from a space H given the observed training data D.

- Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

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

- P(h) denotes the initial probability that hypothesis h holds, and P(D) denotes the prior probability that training data D will be observed.
- P(h|D) is the posterior probability of h, which reflects our confidence that h holds after we have seen the training data D.
- Bayes theorem provides a way to calculate the posterior probability P(h|D) from the prior probability P(h), together with P(D) and P(D|h).
- The most probable hypothesis h in a set H given the observed data D is the maximum a posteriori (MAP) hypothesis.

$$\begin{aligned} h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\ &= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\ &= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) \end{aligned}$$

- The maximum likelihood (ML) hypothesis hML maximizes P(D|h).

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D|h)$$

- Bayes theorem can be applied to any set of mutually exclusive propositions whose probabilities sum to one.

Bayes theorem and concept learning:

Bayes theorem calculates the posterior probability of hypotheses given training data to determine the most probable hypothesis.

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

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

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

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

- *Bayes theorem*: the posterior probability $P(h|D)$ of h given D

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

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

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Brute-Force Bayes Concept Learning:
The concept learning problem involves learning a target concept c from a finite hypothesis space H over instance space X.

- The learner is given a sequence of training examples (xi, di) where xi is an instance from X and di is the target value of xi.
- The training data D can be written as the sequence of target values (d1, d2, ..., dm).
- A concept learning algorithm can be designed based on Bayes theorem to output the maximum a posteriori hypothesis.

Brute-Force Map Learning Algorithm:
BRUTE-FORCE MAP LEARNING algorithm is a concept learning algorithm that uses Bayes theorem to calculate the posterior probability of each hypothesis given the training data.

- The algorithm considers all possible hypotheses in the hypothesis space H and assigns equal prior probability to each of them.
- The algorithm assumes that the training data is noise-free and that the target concept is contained in the hypothesis space H.
- To specify a learning problem for the algorithm, we need to define the prior probabilities P(h) and the likelihood probabilities P(D|h).
- Under the assumption of noise-free training data, P(D|h) is 1 if the hypothesis h is consistent with the data D, and 0 otherwise.
- The algorithm calculates the posterior probability P(h|D) for each hypothesis h by applying Bayes theorem, and selects the most probable hypothesis as the maximum a posteriori (MAP) hypothesis.
- In the case of inconsistent hypotheses, the algorithm assigns a posterior probability of 0.
- The algorithm maintains a version space of hypotheses that are consistent with the training data, and assigns an equal posterior probability to each of them.
- The performance of the algorithm depends on the size of the hypothesis space H, as it needs to calculate the posterior probability for each hypothesis.
- BRUTE-FORCE MAP LEARNING algorithm serves as a standard against which we can compare the performance of other concept learning algorithms.

Bayes optimal classifier:

Bayes optimal classifier is a probabilistic approach for solving classification problems. The goal of a classifier is to predict the class label of a new instance based on its feature values.

- Bayes optimal classifier is based on Bayes theorem, which describes how to update probabilities based on new evidence.
- The Bayes optimal classifier assumes that we have access to the true probability distribution of the feature values for each class.
- Given this assumption, we can calculate the likelihood of observing the feature values given each class, as well as the prior probability of each class.
- Using Bayes theorem, we can compute the posterior probability of each class given the observed feature values.
- The class with the highest posterior probability is chosen as the predicted class label.

- The Bayes optimal classifier is the optimal classifier in the sense that it minimizes the misclassification rate, assuming that the true probability distributions are known.
- However, in practice, we do not have access to the true probability distributions, and we must estimate them from the training data.
- The estimation of the probability distributions can be done using parametric or non-parametric methods.
- The Bayes optimal classifier is sensitive to the quality of the probability estimates, and it can be affected by the curse of dimensionality when the number of features is large.

Naïve Bayes classifier:

Naïve Bayes is a probabilistic algorithm that is widely used for classification tasks.

- It is based on Bayes' theorem, which is a formula that calculates the probability of a hypothesis given some observed evidence.
- Naïve Bayes classifier assumes that the features are independent of each other, given the class variable.
- The algorithm estimates the probability of each class based on the observed feature values, and then selects the class with the highest probability as the predicted class.

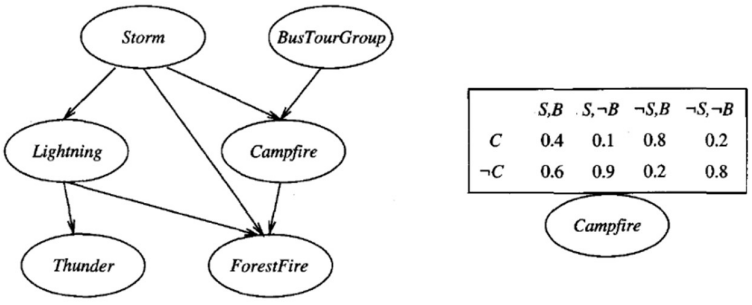
$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

- Naïve Bayes classifier is simple, efficient, and can be used for both binary and multiclass classification tasks.
- It works well with high-dimensional data and can handle missing values and irrelevant features.
- There are three types of Naïve Bayes classifiers: Gaussian Naïve Bayes for continuous data, Multinomial Naïve Bayes for discrete count data, and Bernoulli Naïve Bayes for binary data.
- Naïve Bayes classifier has some limitations, including the assumption of independence between features, and it can be sensitive to irrelevant features.
- Despite its limitations, Naïve Bayes classifier is a powerful and popular algorithm for classification tasks in various fields, including natural language processing, document classification, spam filtering, and image classification.

Bayesian Belief Networks:

Bayesian belief networks (BBNs) are graphical models that use probability theory to represent and reason about uncertainty in a system.

- A BBN consists of a directed acyclic graph (DAG) where nodes represent variables of interest and edges represent probabilistic dependencies between them.



- Each node has a conditional probability distribution (CPD) that specifies the probability of the node given its parents in the DAG.
- BBNs can be used for a variety of tasks, including prediction, diagnosis, and decision making.
- BBNs allow for efficient inference using the chain rule of probability, which allows one to calculate the probability of a particular set of variables given evidence.
- BBNs can be learned from data using various methods, including maximum likelihood estimation and Bayesian learning.
- BBNs can also be used to perform causal inference, allowing one to answer "what if" questions about a system and predict the outcome of interventions.
- BBNs have been successfully applied in a wide range of domains, including medicine, finance, and engineering.
- One limitation of BBNs is that they assume a static model of the system, which may not be appropriate for systems with complex dynamics.
- BBNs also assume that variables are conditionally independent given their parents, which may not hold in all cases. However, this assumption can often be relaxed by adding additional edges to the DAG or using more complex models.

Conditional independence:

In BBN, conditional independence is a key concept that is used to simplify the computation of joint probabilities.

- Conditional independence is the notion that the occurrence of one event does not affect the probability of another event occurring, given the occurrence of a third event.
- In BBN, conditional independence is used to reduce the number of parameters that need to be estimated for a large network.
- Two variables in a BBN are conditionally independent if the probability of one variable given another variable is the same as the probability of the first variable given the other variable and a third variable.
- This can be expressed mathematically as $P(X|Y,Z) = P(X|Z)$, where X and Y are two variables in the network, and Z is a set of other variables in the network.
- If X and Y are conditionally independent given Z, we can simplify the network by removing the direct link between X and Y, and introducing a new node that represents Z.
- The new node represents the shared factors that influence both X and Y, and its probability distribution can be learned from data.
- If there is no direct causal relationship between two nodes in the graph, they are conditionally independent given the nodes that lie on the unique path between them.

Learning Bayesian Belief Networks:

Learning BBNs from training data can be done using effective algorithms.

- The network structure can be given in advance or inferred from the training data.
- All network variables can be directly observable or some might be unobservable in each training example.

- In the case where only some variable values are observable, the learning problem is more difficult, analogous to learning weights for hidden units in an artificial neural network.
- Gradient ascent procedure can be used to search through a hypothesis space that corresponds to all possible entries for the conditional probability tables.
- The objective function that is maximized during gradient ascent is the probability P(D/h) of the observed training data D given the hypothesis h.
- This corresponds to searching for the maximum likelihood hypothesis for the table entries.

Parametric Methods: Maximum Likelihood Estimation:

Parametric methods are a class of machine learning algorithms that assume a certain functional form for the probability distribution of the data. Maximum likelihood estimation (MLE) is a common technique used in parametric methods to estimate the parameters of a probability distribution based on a set of observations.

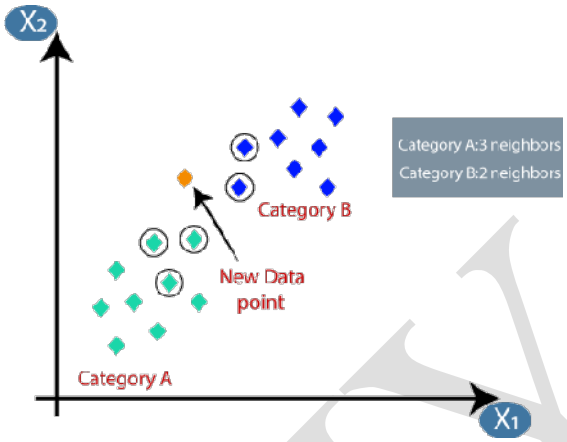
- MLE assumes that the data follows a specific probability distribution, such as Gaussian or Poisson.
- The goal of MLE is to find the parameter values that maximize the likelihood function, which measures how well the observed data fits the assumed distribution.
- The likelihood function is a function of the parameters and the observed data.
- The MLE estimates for the parameters are obtained by finding the values that maximize the likelihood function.
- MLE can be used for both univariate and multivariate data.
- MLE can be used for both supervised and unsupervised learning problems.
- In supervised learning, MLE is used to estimate the parameters of a model that predicts the label or class of a data point based on its features.
- In unsupervised learning, MLE is used to estimate the parameters of a model that describes the underlying structure of the data, such as a clustering model or a latent variable model.
- MLE is often used in combination with regularization techniques, such as L1 or L2 regularization, to prevent overfitting.
- MLE can be sensitive to outliers and the choice of the assumed distribution.

Non parametric methods: K nearest neighbour:

Non-parametric methods are statistical methods that do not make any assumptions about the underlying distribution of the population from which the sample is drawn. K-nearest neighbor (K-NN) is a non-parametric method used for classification and regression.

- The basic idea behind K-NN is to classify or predict the target value of an input sample by finding its nearest neighbors in the training dataset and taking a majority vote or averaging their target values.
- The value of K in K-NN is a hyperparameter that determines the number of nearest neighbors to consider.
- A larger K value results in a smoother decision boundary, but may lead to over-generalization and loss of accuracy, while a smaller K value may result in overfitting.
- The distance metric used to calculate the distance between samples can vary, with commonly used metrics including Euclidean distance, Manhattan distance.

- K-NN can handle datasets with any number of features and is not sensitive to the distribution of the data. However, it is computationally expensive and may require a large amount of memory for large datasets.
- Variations of K-NN include weighted K-NN, where the contribution of each nearest neighbor is weighted by their distance from the input sample.



Introduction to Support Vector Machine:

SVM is a type of supervised learning algorithm that can be used for classification or regression tasks.

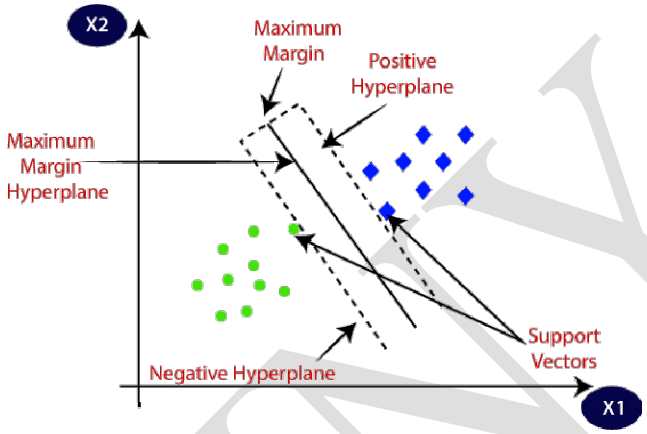
- The main goal of SVM is to find the best possible line or hyperplane that separates the data into different classes.
- SVM is particularly useful when dealing with datasets that have a lot of features, as it can handle high-dimensional data efficiently.
- SVM works by mapping the input data to a higher-dimensional feature space and then finding the hyperplane that best separates the data in that space.
- SVM tries to find the hyperplane that maximizes the margin.
- SVM uses a kernel function to transform the input data into the higher-dimensional space, where it is easier to find the hyperplane that separates the data.
- SVM can handle both linearly separable and non-linearly separable data using different types of kernel functions.
- SVM is a powerful and widely used algorithm in machine learning, and has been successfully applied in various domains, such as image classification, natural language processing, and bioinformatics.

Optimal Separating Hyperplane:

The Optimal Separating Hyperplane is defined as the hyperplane that maximizes the margin between the two classes.

- The margin is defined as the distance between the hyperplane and the closest data points from each class.
- The distance between a data point and the hyperplane is calculated using the equation of the hyperplane and the coordinates of the data point.
- The hyperplane is defined by the weights assigned to each feature in the dataset.

- The SVM algorithm finds the optimal weights for the hyperplane by solving a constrained optimization problem.
- The optimization problem involves minimizing the norm of the weight vector subject to the constraint that all data points are correctly classified by the hyperplane.
- The optimization problem can be solved using quadratic programming techniques.
- In some cases where the classes are not linearly separable, the SVM algorithm can use a kernel function to map the data into a higher-dimensional space where the classes are separable.
- The optimal separating hyperplane in the higher-dimensional space corresponds to a nonlinear decision boundary in the original feature space.



The Nonseparable Case: Soft Margin Hyperplane:

In many practical situations, it is not possible to find a hyperplane that perfectly separates the two classes. This is known as the non-separable case.

- In such cases, we need to allow for some errors to be made in the classification.
- Soft Margin Hyperplane is an extension of the hard margin hyperplane for the non-separable case.
- It introduces slack variables which allows some points to be misclassified, but penalizes the degree of misclassification.
- The optimization objective of Soft Margin Hyperplane is to maximize the margin while minimizing the classification errors, measured by the slack variables.
- A trade-off parameter, C, controls the balance between maximizing the margin and minimizing the classification errors.
- Larger values of C lead to a smaller margin but fewer errors and vice versa.
- The soft margin hyperplane can be formulated as a quadratic optimization problem with linear constraints, similar to the hard margin hyperplane.
- The solution of the optimization problem gives us the optimal hyperplane that maximizes the margin while minimizing the classification errors.
- In the case where the classes are not linearly separable in the original feature space, a kernel function can be used to transform the feature space into a higher dimensional space where the classes become separable.
- The soft margin hyperplane can then be found in the transformed feature space, allowing for non-linear classification.

Defining Kernels:

In SVM, a kernel is a function that takes two data points as input and outputs their similarity.

- The kernel function is used to transform the data into a higher-dimensional space where it is easier to find a separating hyperplane.
- A kernel function should satisfy Mercer's condition, which means it should be positive semi-definite.
- There are different types of kernel functions that can be used in SVM, such as linear kernel, polynomial kernel, radial basis function (RBF) kernel, sigmoid kernel, etc.
- The linear kernel is the simplest kernel, which calculates the dot product between two data points in the original space.
- The polynomial kernel is used to handle nonlinearly separable data by mapping it into a higher-dimensional space.
- The RBF kernel is the most commonly used kernel in SVM, which maps the data into an infinite-dimensional space.
- The sigmoid kernel is used to model neural networks and logistic regression.
- The choice of kernel function depends on the problem domain and the characteristics of the data.
- The parameters of the kernel function, such as the degree of the polynomial kernel or the width of the RBF kernel, should be tuned using cross-validation to achieve optimal performance of the SVM classifier.