Bayesian Learning

provides a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability distributions and optimal decisions can be made by reasoning about these probabilities along with the training data.

* It provides a quantitative approach to weighing the evidence supporting alternative hypotheses.

They are relevant field of study for Machine Learning because

(1) They are a practical method for learning since hypothesis selection is based on probability.

(2) They provide a very useful perspective to understanding other learning algorithms that do not explicitly quantify hypothesis selection.

Features of Bayesian Learning

(1) Each observed training example can incrementally increase or decrease the estimated probability that a hypothesis is correct, unlike other learning algorithms that completely eliminate a hypothesis that is

inconsistent with the training data.

(2) Prior Knowledge can be combined with observed data to determine the final probability of a hypothesis.

Prior knowledge is provided by asserting

(1) a prior prob of each candidate hypothesis

(2) a probability distribution over observed data for each possible hypothesis.

(3) Bayesian methods can accomodate hypotheses that make probabilistic predictions.

(4) New instances can be classified by combining the predictions of multiple hypotheses, weighted by probabilities.

(5) Bayesian methods can provide a standard for optimal decision making against which other practical methods can be compared, even for computationally intractable problems.

Practical difficulty in Bayesian Learning:

(1) Requires initial knowledge of many probabilities.
If they are not known, they are often estimated on background knowledge, previously available data and assumptions about the form of underlying distributions.

(2) Significant computational cost required to determine the Bayes optimal hypothesis.

Bayes Theorem:-

Bayes Theorem is named after a famous mathematician, 'Thomas Bayes'. Also known as the Bayes' Rule is a simple formula to calculate conditional probability.

It is found to be very useful in probabilistic machine learning algorithms.

Consider a hypothesis space H, and training data D, the aim of any ML algorithm is to find the best hypothesis that fits the training data.

The best hypothesis is the <u>most probable hypothesis</u> given D and initial knowledge of any prior probabilities

of hypotheses in H.

Notation:-

$P(h)$ denotes the initial probability that hypothesis h holds, before the training data is observed. It is called the prior probability of h.

$P(D)$ denotes the prior probability of D (ie) probability of D given no knowledge about which hypothesis holds.

$P(D|h)$ denotes the probability of D given that the hypothesis h holds

(ie) $P(x|y)$ denotes Probability of x given y.

$P(h|D)$ is the probability of h (ie) prob that h holds given the training data. This is called the posterior probability of h. It reflects the confidence that h holds after seeing the training data D.

$P(h)$ - independent of D

$P(h|D)$ - dependent on D.

Bayes Theorem:-

$$\text{Posterior} \leftarrow P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)} \rightarrow \text{Prior}$$

Likelihood

$P(h|D)$ increases with $P(h)$ and $P(D|h)$.

$P(h|D)$ decreases as $P(D)$ increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

Maximum a Posteriori hypothesis (MAP)

During learning, the learner considers a set of candidate hypotheses in H and is interested in finding the most probable hypothesis $h \in H$ given the observed data D.

In other words, find the maximally probable hypothesis if there are several hypotheses. This is called a Maximum a Posteriori hypothesis (MAP).

The MAP hypotheses can be determined by using Bayes Theorem to calculate the posterior probability of each candidate hypothesis.

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

$$= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h) \cdot P(h)}{P(D)}$$

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

The term $P(D)$ is dropped, because it a constant independent of h.

$P(D|h)$ is called the likelihood of the data D given h, and any hypothesis that maximises $P(D|h)$ is called a maximum likelihood (ML) hypothesis h_{ML}.

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

Brute Force Bayes Concept Learning

Consider concept learning where a learner assumes a finite hypothesis H defined over X. Task is to learn $c: X \rightarrow \{0, 1\}$. Consider D as the training data, with a sequence of target values $D = \langle d_1 \dots d_m \rangle$. Bayes Theorem can be used to generate a MAP hypothesis.

Brute Force MAP learning algorithm.

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

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

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

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

Although the algorithm requires significant computation for calculating P(h|D) for each hypothesis, and may be impractical for large hypothesis spaces, it provides a standard for judging other concept learning algorithms

In order to specify a learning problem for Brute Force MAP learning, values for P(h) and P(D|h) needs to be specified. They are consistent with the following assumptions.

1. The training data D is noise free
2. The target concept c is contained in hypothesis space H
3. No reason to believe that any hypothesis is more probable than the other.

Given these assumptions:- What value for P(h)?

Every hypothesis is equally likely; hence same prior probability for all h in H.

Hence $P(h) = \frac{1}{|H|}$ for all h in H.

What value for P(D|h)?

P(D|h) is the probability of observing target values $D = \langle d_1, \ldots d_m \rangle$ for instances $\langle x_1 \ldots x_m \rangle$, given hypothesis h. Since training data is noise free,

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

(ie) Prob of data D given hypothesis h is 1 if D is consistent with h, and 0 otherwise.

Knowing values of P(h) and P(D|h), as per Bayes Theorem

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

Case 1: If h is inconsistent with D, from (1)

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

Hence $P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$ if h is inconsistent with D.

Case 2: If h is consistent with D, from (1)

$$P(D|h) = 1$$

Hence $P(h|D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)}$

Calculating $P(D)$ from total probability rule

Given n mutually exclusive events $A_1, A_2 \cdots A_n$, whose probabilities sum to unity, then

$$P(B) = P(B|A_1) \cdot P(A_1) + P(B|A_2) \cdot P(A_2) \cdots P(B|A_n) \cdot P(A_n),$$

where B is an arbitrary event

and $P(B|A_i)$ is the conditional prob of B assuming A_i

Hence $$P(D) = \sum_{h_i \in H} P(D|h_i) P(h_i)$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|}$$

$$P(D) = \frac{|VS_{H,D}|}{|H|}$$

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

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

where

$|VS_{H,D}|$ is the no of hypotheses from H consistent with D.

The above analysis shows that the choices for $P(h)$ and $P(D|h)$, makes every consistent hypothesis to have a posterior probability $\frac{1}{|VS_{H,D}|}$ and every inconsistent hypothesis to have posterior probability 0.

Therefore every consistent hypothesis is a MAP hypothesis.

MAP hypotheses and consistent learners.

A learning algorithm is a consistent learner provided it outputs a hypothesis that commits zero errors over the training examples.

* Every consistant learner outputs a MAP hypothesis, if the following assumption hold

① uniform prior probability distribution over H

(ie) $P(h_i) = P(h_j)$ for all i, j

② Deterministic noise-free training data.

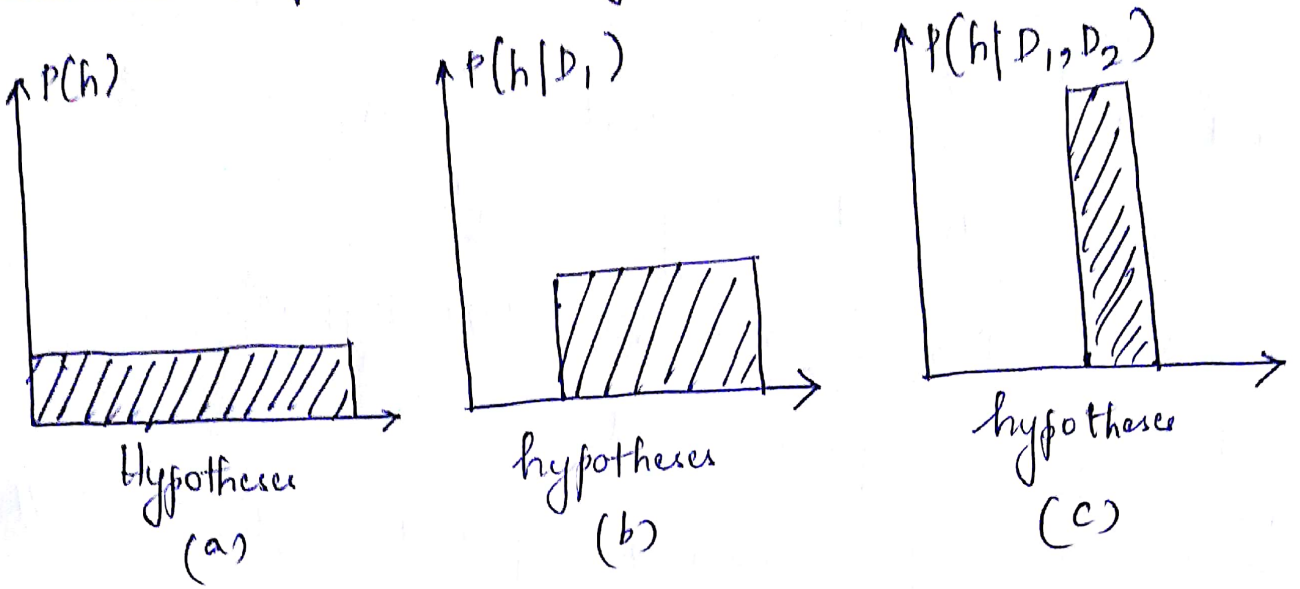
(ie) $P(D|h) = 1$ if D and h are consistant,
0 otherwise.

* Consider the FIND-S algorithm for concept learning. It outputs a hypothesis that is maximally specific [outputs a maximally specific member of the version space] and consistent with the training data.

* Since it generates a consistent hypothesis, it generates a MAP hypothesis [(ie) consistant has to be MAP] under the assumptions of $P(h)$ and $P(D|h)$.

* This MAP hypothesis is generated by not explicitly calculating probabilities.

Evolution of probabilities graph:



The above graph shows the evolution of posterior probabilities with increasing training data.

Graph(a) shows uniform prior assign equal probabilities to all hypotheses.

Graph (B) - As training data increases from D1 and then to D1 ∧ D2 in Graph(c), the posterior probability of inconsistent hypothesis becomes zero and posterior probability increases for hypotheses belonging to the version space.

To Summarise: Bayesian framework allows one to characterise the behaviour of learning algorithms (find-S and CE) even when they do not explicitly calculate probabilities.

The inductive bias associated with any learning algorithms is generally justified by converting it into in an equivalent deductive system.

The Bayesian interpretation provides an alternate method to characterise assumptions implicit in the learning algorithms. The learning can be modeled by an equivalent probabilistic reasoning system based on Bayes Theorem.

Maximum Likelihood and Least-Squared Error hypothesis

Many learning approaches such as Neural Network and linear regression learn a continuous valued function rather than a discrete function.

Bayesian Learning can be used to minimise the squared error between the o/p hypothesis prediction and the training data and hence will output a Maximum Likelihood hypothesis [(ML) hypothesis].

(ie) Any learning algorithm that minimises the ~~Least~~ squared error between the prediction and the training data will output a ML hypothesis under suitable assumptions.

Consider the following problem setting.

Learner L considers an instance space X, a hypothesis space H consisting of some class of real-valued functions defined over X.

(ie) each h in H is of the form

$$h : X \rightarrow R$$

Problem faced by L is to learn an unknown target function $f : X \rightarrow R$ drawn from H.

* A set of training examples m is provided, which is corrupted by random noise.

(ie) the target value is corrupted by random noise for each training sample.

(ie) Each training example is of the form $\langle x_i, d_i \rangle$ where $d_i = f(x_i) + e_i$

- $f(x_i)$: Noise-free value of target function.
- e_i : random variable representing noise.

The task of the learner is to output a ML hypothesis for the problem of learning a continuous valued function

Continuous valued target functions are learnt based on their probability densities

Basics of continuous random variable [only for Reference]

Random Variable:- In an experiment, any variable which assumes different numerical values based on the outcomes of the experiment and to which probability can be assigned is called a random variable.

Example:- In an experiment of Tossing 3 coins, number of heads (H) can take values 0, 1, 2, 3.

The resulting values and their probability representations is called Probability distribution of that event (H)

H	P(H)
0	0.125
1	0.375
2	0.375
3	0.125

Here H takes discrete values and hence called a discrete random variable, and its distribution → Discrete Prob Distribution

Continuous Random Variable

Any random variable that can take infinite no of values is a continuous random variable, and its distribution is called continuous probability distribution.

Example:- Consider a line segment 0–1. If the event is picking a point at random, say X, then X can take any infinite number of values.

if $X = 0.32 \Rightarrow$ Prob of occurence of $X = 0.32$ is so small, that a more meaningful

representation of probability distribution is required. This representation is called probability density function (pdf) that specifies the probability that X will lie in a certain range rather than X taking a particular value.

(contd)

Hence PDF represents the probability distribution of a continuous random variable and the summation of all probabilities for continuous variables is made possible through integration. In this context,

$d_i = f(x_i) + e$ and

Here e is a continuous random variable and hence impossible to assign a finite probability to the infinite set of values for the random variable. e. So we represent it by a PDF and the integral of this function limit the probability to a certain range or interval.

Probability density function

$$p(x_0) \equiv \lim_{\epsilon \to 0} \frac{1}{\epsilon} P(x_0 \leq x < x_0 + \epsilon)$$

where p - Pdf.

P - finite probability that is limited in the range $[x_0, x_0 + \epsilon]$.

Given the background of the probability density function for representing the distribution of probability for continuous random variables, the ML hypothesis can be used to prove that it reduces the sum of the squared error between the observed training data and predictions. (ie) ML hypotheses in fact generates Least squared error hypotheses..

We know that

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

Here p refers to the probability density function. [p - lower case].

We also know that $d_i = f(x_i) + e_i$

where d_i is the sequence of target values $\langle d_1, d_2 \cdots d_m \rangle$

Since the training examples are mutually independent given h,

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h)$$ [Note: small represents PDF]

Assuming that both $f(x_i)$ and e_i follows Normal distribution of data,

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2} \rightarrow (1)$$

where $p(d_i|h)$ the probability density function is represented by the PDF of Normal distribution given as

P.D.F of Normal Distribution

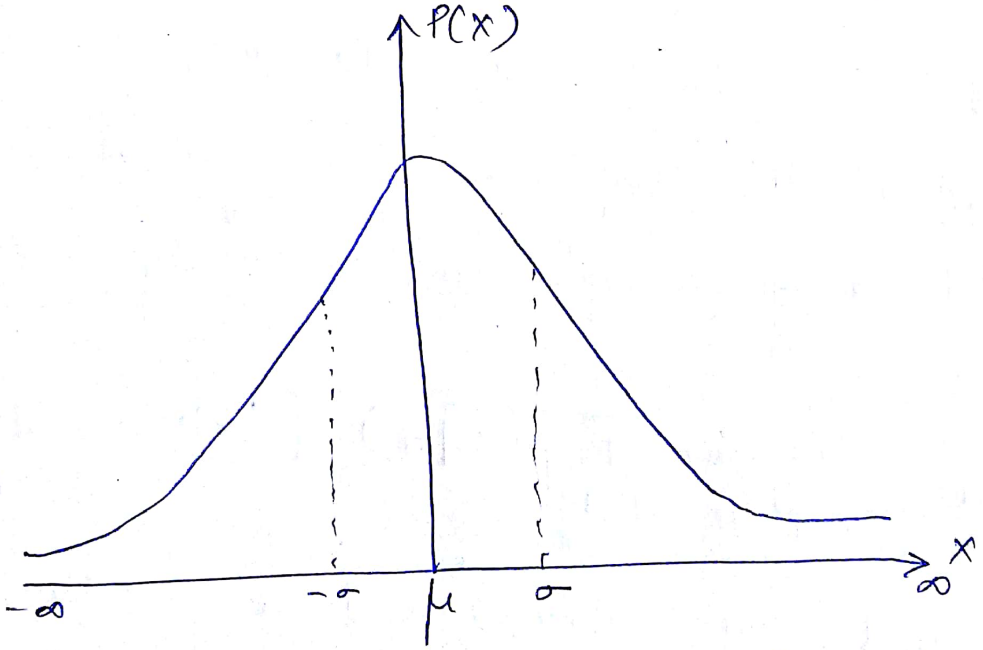
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Where σ^2 - variance

μ - Mean

π and e are contants

and x ranges from $-\infty$ to ∞.



The above curve shows the typical bell curve that represents the probability density function represented by μ and σ.

* Substituting $\mu = f(x_i) = h(x_i)$ in equation (1)

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2}$$

* Applying a transformation common in likelihood calculations, log is applied to the above h_{ML}.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

where ln is the natural logarithm

* The first term is a constant independent of h, and discarded, yielding

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

Maximising this negative quantity is equivalent to minimising the equivalent positive quantity.

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

Discarding constants, independent of h

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

h_{ML} minimises the sum of the squared errors between the observed training data d_i and predictions $h(x_i)$.

Why we choose Normal distribution for the characterisation of data?

* N.D yields a mathematically straight forward distribution.
* The smooth, bell shaped curve is a good approximation to many types of noise in physical systems.

Maximum Likelihood hypothesis for predicting probabilities.

Consider the setting in which the aim is to learn a probabilistic function $f: X \rightarrow \{0, 1\}$ which has two discrete output values.

* Assume the training data is of the form

  $D = \{\langle x_1, d_1\rangle, \ldots \langle x_m, d_m\rangle\}$ where d_i is either 0 or 1.

* Treating both x_i and d_i as random variables,

$$P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h) \longrightarrow (1)$$

$$= \prod_{i=1}^{m} P(d_i|h, x_i) P(x_i) \longrightarrow (2)$$

[Applying product rule of probability]

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ 1 - h(x_i) & \text{if } d_i = 0 \end{cases} \longrightarrow (3)$$

Re-expressing it in a more mathematically manipulatable form:

$$P(d_i|h, x_i) = h(x_i)^{d_i} (1-h(x_i))^{1-d_i} \rightarrow (4)$$

[Follows Binomial Distribution]

If $d_i = 1$, the second term becomes 0
if $d_i = 0$, the first term becomes 0.

Substituting (4) in (2)

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i} (1-h(x_i))^{1-d_i} P(x_i)$$

Writing the ML hypothesis,

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1-h(x_i))^{1-d_i}$$

dropping $P(x_i)$ independent of h.

Applying log likelihood,

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

Minimum Description Length Principle:

is based on the information theoretic approach of Occam's Razor which states that the 'shortest and simplest hypothesis is the best hypothesis'.

MDL principle states that the shortest explanation of the training data also gives the greatest compression of data.

Hence a probabilistic model (hypothesis) that best describes the data can be optimally encoded so that the length of model and ~~model~~ data given the model is the most compressed representation.

For encoding hypothesis and data, the optimal code $\boxed{-\log_2 P_i}$ bits is used where P_i bits is used to encode a data i. [Proved from info theory]

Applying this optimal encoding to the MAP hypothesis;

$$h_{MAP} = \underset{h \in H}{\text{argmax}}\ P(D|h)\, P(h)$$

Taking log likelihood

$$h_{MAP} = \underset{h \in H}{\text{argmax}}\ \log_2 P(D|h) + \log_2 P(h)$$

Negating this term

$$h_{MAP} = \underset{}{\text{argmin}}\ -\log_2 P(D|h) - \log_2 P(h) \rightarrow (1)$$

The above equation (1) implicitly defines the optimal encoding of the probabilities $[-\log_2 P(h)$ & $-\log_2 P(D/h)]$ (2b)

* $\log_2 P(h)$ - description length of h
* $\log_2 P(D/h)$ - description length of training data.

Rewriting equation (1) using the encoding

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}} \; L_{C_H}(h) + L_{C_{D/h}} D/h.$$

Finally

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} \; L_{C_1}(h) + L_{C_2}(D/h) \rightarrow (2)$$

where C_1 and C_2 are the optimal encoding bits for model and data

(2) shows that the 'h' is chosen that minimizes the sum of the description length of hypothesis plus the description length of data given the hypothesis.

MDL principle provides a way of trading off hypothesis complexity for the no of errors committed by the hypothesis. It selects a shorter hypothesis that makes fewer errors over longer hypotheses that perfectly classifies datas, hence avoiding overfitting of data.

Naive Bayes Classifier

* is a popular classification algorithm based on Bayes Theorem.
* The NBayes algorithm follows class-conditional independence which assumes that the attributes of instances in training data are independent of each other.
* The word 'Naive' follows from this basic assumption.
* It applies to learning tasks where each instance x is described by a conjunction of attributes and the target function $f(x)$ can take on any value from a finite set V.
* A new instance x_i defined by the attribute values $\langle a_1, a_2 \ldots a_n \rangle$ is classified by the NB learner to one of the classes $v_j \in V$.
* The Bayesian Learning selects the most probable hypothesis

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j \mid a_1, a_2 \ldots a_n)$$

Using Bayes Theorem

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \; \frac{P(a_1, a_2 \ldots a_n \mid v_j) \, P(v_j)}{}$$

$$V_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

Since NB assumes attribute values are independent, the probability of observing the conjunction $(a_1, a_2 \ldots a_n)$ is just the product of the probabilities of the individual attributes. (ie) $P(a_1, a_2 \ldots a_n) = \prod_{i=1}^{m} P(a_i | v_j)$

NB Classifier

$$V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i=1}^{m} P(a_i | v_j)$$

↓

Target value output.

Estimating probabilities:

The observed V_{NB} is a good estimate of the probability as long as the probabilities do not go to tiny values (close to zero).

The non-zero probability term dominates the classification, and the resultant classification is a biased estimate.

In such cases,

$$m\text{-estimate of Prob} = \frac{n_c + mp}{n + m}$$

p - prior probability

m - constant that determines how heavily to weight

Bayesian Belief Network

A Bayesian Belief Network or Bayesian N/w is a statistical model used to describe the conditional dependencies among the different random variables.

* Unlike Naive Bayes Classifier which assumes that the attributes are all conditionally independent, given the target attribute, the BBN describes conditional independence among subsets of variables.

* It describes the probability distribution governing a set of variables by specifying conditional independence assumptions over a subset of variables along with their conditional probabilities.

Bayesian Network

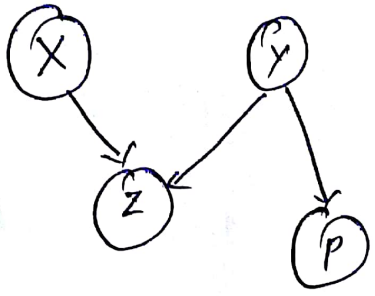
* is a simple graphical notation for conditional independence assumptions.
* Each variable is represented by a node.
* The edges connecting the nodes show the dependency relation between the variables.
* The resultant graph is a DAG that represents the probability distribution of data over a set of

More precisely;

* if $Y_1, Y_2 \ldots Y_n$ are the random variables, where each variable Y_i can take a set of values $V(Y_i)$.
* Each item corresponds to one of the possible assignments of values to the tuple of variables $\langle Y_1, Y_2 \ldots Y_N \rangle$.
* The probability distribution over this joint space is called the joint probability distribution

Hence a BBN describes the joint probability distribution over the set of variables. It is a graphical model of causal relationship [cause and effect].

Ex:-



Nodes : Random Variables.

Links : Dependency

X, Y are parents of Z & P.

DAG : Hence No loops.

Conditional Independence:-

If X, Y, Z are three discrete-valued random variables, X is conditionally independent of Y given Z if probability distribution governing X is independent of Y given a value for Z.

$\therefore$ i.e) $(\forall x_i\ y_j\ z_k)$ $\boxed{P(X=x_i \mid Y=y_j, Z=z_k) = P(X=x_i \mid Z=z_k)}$

Generalising

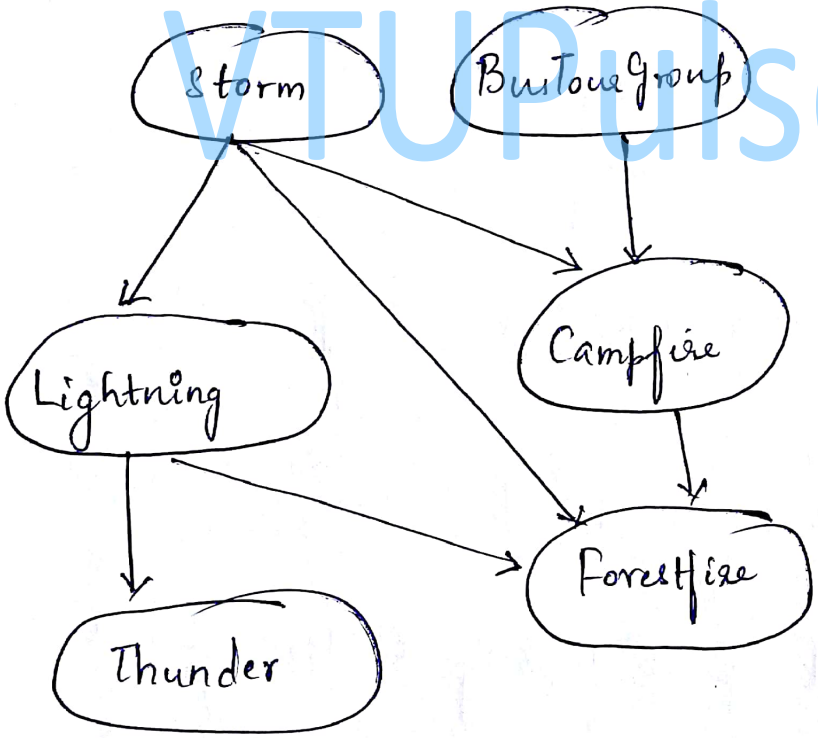
$$P(X_1 \ldots X_l \mid Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l \mid Z_1 \ldots Z_n)$$

X is independent of Y given Z.

Representation of BBN:

represents the joint probability distribution of a set of variables.

Example:



Each random variable is represented by a node. For each variable, two types of information are specified. (*) The N/w arcs represent the assertion that the variable is conditionally independent of its

non-descendants in the N/w, given its immediate predecessors.

(*) A CPT [conditional prob table] is given for each variable that describes the probability distribution of the variable given its parents. [immediate predecessors]

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

where $P(y_1 \ldots y_n)$ is the joint prob distribution of the assignment of values $(y_1 \ldots y_n)$ to $(Y_1 \ldots Y_N)$.

Example:-

	S B	S ~B	~S B	~S ~B
C	0.4	0.1	0.8	0.2
~C	0.6	0.9	0.2	0.8

In the above table, the top left entry expresses the assertion that

$$P(\text{Campfire} = T \mid \text{Storm} = T, \text{BusTourGroup} = T)$$

The set of conditional probability tables for all the variables, together with the set of conditional independence assumptions described by the N/w describe the full joint probability distribution of the Network.

Inference.

Using a Bayesian Network to infer the value of some target variable, given the observed values of other variables is straightforward, if values for all variables in the Network are known exactly. In general BBN can be used to compute the probability distribution for any subset of n/w variables given the values or distributions for any subset of the remaining variables.

In order to learn a BBN, the n/w structure may be given in advance or it may have to be inferred from the training data. In addition, all variables may be directly observable from the training data or it may be unobservable.

If the variables are directly not observable from the training data, the learning of the BBN is analogus to the learning of hidden weights in Neural Network.

The conditional probability table entries which are hidden is calculated using a

Gradient Ascent Procedure; that searches through a space of hypotheses that corresponds to the set of all possible entries in CPT.

The aim is to maximise $P(D|h)$ that searches for the maximum likelihood hypothesis for the table entries.

Gradient Ascent Training of Bayesian Network.

The gradient ascent rule maximises $P(D|h)$ by calculating the gradient of $\ln P(D|h)$.

Let w_{ijk} denote a single entry in CPT.

(ie) the conditional probability that N/w variable Y_i will take on the value y_{ij} given that its immediate parents U_i take on the values given by u_{ik}.

Example:- if Y_i is the variable campfire, U_i is the tuple of its parents $\langle$Storm, BusTourGroup$\rangle$,

Y_{ij} = True and u_{ik} = $\langle$False, False$\rangle$.

The gradient of $\ln P(D|h)$ is given by the derivatives of $\frac{\partial}{\partial w_{ijk}} \ln P(D|h)$.

$P(D|h)$ is represented as $P_h(D)$

$$\frac{\partial \ln P_h(D)}{\partial \omega_{ijk}} = \frac{\partial}{\partial \omega_{ijk}}\left[\ln \prod_{d \in D} P_h(d)\right]$$

$$= \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial \omega_{ijk}}$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial \omega_{ijk}}$$

$$\left[\because \frac{\partial}{\partial x} \ln f(x) = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}\right]$$

$$\frac{\partial \ln P_h(D)}{\partial \omega_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial \omega_{ijk}} \sum_{j'k'} P_h(d \mid y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial \omega_{ijk}} \sum_{j'k'} P_h(d \mid y_{ij'}, u_{ik'}) P_h(y_{ij'} \mid u_{ik'}) P_h(u_{ik'})$$

[Product rule of Probability]

$$\frac{\partial \ln P_h(D)}{\partial \omega_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \cdot \frac{\partial}{\partial \omega_{ijk}} P_h(d \mid y_{ij}, u_{ik}) \cdot P_h(y_{ij} \mid u_{ik}) \cdot P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial \omega_{ijk}} P_h[d \mid y_{ij}, u_{ik}) \, \omega_{ijk} P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d \mid y_{ij}, u_{ik}) \cdot P_h(u_{ik})$$

Applying Bayes Rule; to $P_h(d \mid y_{ij}, u_{ik})$

$$\frac{\partial \ln P_h(D)}{\partial \omega_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \cdot \frac{P_h(y_{ij}, u_{ik} \mid d) P_h(d) \cdot P_h(u_{ik})}{P_h(y_{ij}, u_{ik})}$$

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})}$$

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{}$$

$$= \sum_{d \in D} \frac{P_h(y_j, u_{ik} | d)}{w_{ijk}}$$

Gradient Ascent Procedure:

$$w_{ijk} = w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_j, u_{ik} | d)}{w_{ijk}}.$$

As weights w_{ijk} are updated, they should sum to 1. [valid probability range $[0, 1]$].

The EM Algorithm.

EM algorithm [Expectation Maximisation] is useful in a number of learning scenarios where only a subset of the relevant instance features might be observable. In cases like these, EM algorithm is very useful to estimate the values of the unobserved variables, given the probability distribution of these variables.

It is a general method of finding the maximum likelihood estimate of the parameters of an underlying distribution from a given dataset with incomplete or missing values.