It favours for the attributes which has more value.

Disad:- Broader tree at one level.

Alternate measure is Gain Ratio.

↓

Split information

$$\text{Split info} = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}$$

where S_1 to S_c are 'c' subsets of examples resulting from partitioning 'S' by the 'c' valued attribute A.

$$\text{Gain Ratio}(S,A) = \frac{\text{Gain}(S,A)}{\text{Split info}(S,A)}$$

08/01/2024

1) Handle the training examples with missing attribute

Gain(S,A) A(x) = Temp / Humidity / Wind / Outlook

Training example $\langle x, c(x) \rangle$, A(x) => Null

2 strategies

1) whichever value is max in that attribute then fill it with that max value irrespective of c(x)

2) c(x) of that instance if it +ve or -ve check the target concept whichever is true or -ve then fill it correspondingly. acc to c(x) value

Handling attributes with diff costs

Some times - are associated with costs/ weight.

eg:- To predict medical disease

Temp / Pulse / BloodTest / Biopsy / Target

$$\frac{\text{Gain}^2(S,A)}{\text{Cost}(A)}$$

null

Based on value.

v imp Avoiding Overfitting in the Data.

less more

error in train data & error in test data.

when we apply hypothesis on training sample it show less error

" " " some " " testing sample it shows more error because it has learnt more than what is required.

Given an hypothesis h, it said to overfit the training examples if the hypothesis h is performing better over the training set than over the testing set. So Training error(h) < Testing error(h).

Error less Accuracy more

why is this overfitting causing in ID3

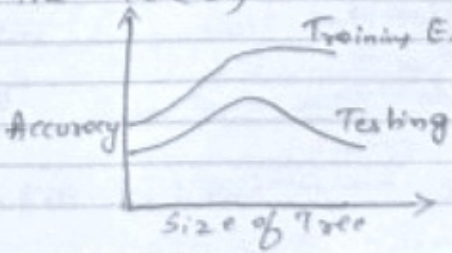
-> Noise.

-> If Size of data set is small they are not in a position to test.

eg:- 1 (Noise)

Outlook = sunny, Temp = hot, Humidity = Normal, wind = storm : Play Tennis = No

(for the Tree)



9t

Avoiding overfitting in the Data.

Strategy:-

① ↳ Stop growing the tree after reaching level

② ↳ Grow the tree completely, allow overfitting to occur, then we go for pruning the tree

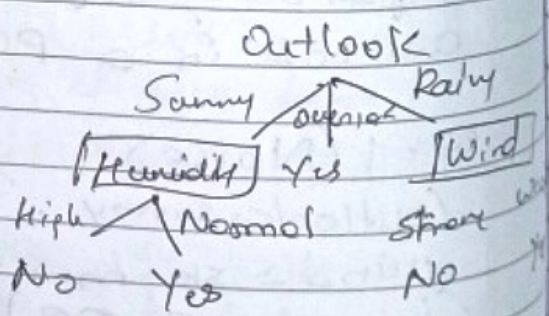
① Disadv:- 9t is not easy to determine where we stop growing the tree. (practical difficult)

② 9t is mostly used strategy.

② —— i reduced error pruning.
\ ii Rule-post pruning.

i In the tree that is already built, every Decision node becomes candidate to prune, pruning is made until we get high accuracy. or while pruning accuracy is checked if it is high it is chopped off and replaced with leaf node.

Humidity & wind



i) It Considers each of the Decision nodes in the tree to be Candidate for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node and assigning it to the most ~~complication~~ common

Nodes are removed only if the resulting pruned tree performs better that the original tree for

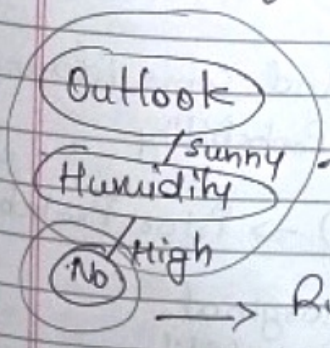
Pruning of tree continues until further pruning is harmful. The pruning is stopped one the accuracy ~~start~~ start is decreasing.

ii) Rule-post-Pruning

After the tree is built it is ~~set~~ converted into set of rules. For every leaf node there is a rule. Rule is in the form of path from the root node to leaf node.

and

Example: if (outlook = sunny) ∧ (Humidity = high) then (playtennis = No)



This path is known as Rule Antecedent (pre-condition)

→ Rule Consequent (post-condition)

- Only pre-condition is pruned.

1) * Build a Decision Tree from the Training s
grow the tree completely until training da
is fit allow the overfitting to occur.
* Convert the tree into set of rules by
creating a rule for each path from the root
node to leaf node.
* Prune each rule by removing any tree
conditions so that the accuracy is
improved.
* we generate one rule for each leaf node i
tree.
* The path from the root to leaf becomes Rul
Antecedent or pre-condition.
* A classification at leaf node becomes the rule
consequent or the the post-condition.

* Each rule is pruned by removing the
Antecedant or the precondition.
* This process is continued until there is
improvement in the estimated accuracy.
* No pruning is performed if there is no
improvement in the accuracy.

16/01/2024 3rd Module.

Bayesian learning.

computing probabilities to find some inferences
prediction

Bayes Theorem

Likelihood

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

$P(A|B)$ → Posterior Probability

$P(A)$ → Prior Probability

$P(B)$ → Marginal Probability.

Ex:- What is the probability of finding the king Given that it is a facecard in a pack of card.

facecards = J & K $4 \times 3 = 12$

$$P(\text{King} | \text{facecard}) = \frac{P(\text{face} | \text{king}) \cdot P(\text{King})}{P(\text{face})}$$

$$= \frac{4/4 \times 4/52}{12/52}$$

$$1 \times \frac{4}{52} \times \frac{52}{12} \quad \text{or} \quad \frac{4}{12} = \frac{1}{3}$$

$$P(K/F) = \frac{1}{3}$$

→ Bayesian learning provides a probabilistic approach for inferences.

→ Bayesian learning Algorithms calculate explicit probabilities for hypothesis and are among the most practical approaches for certain types of learning problems

imp Features of Bayesian learning

* Each training example can incrementally increase or decrease the estimated probability that the hypothesis is correct. This provides more flexibility than Algorithms that completely eliminate hypothesis if it is found to be inconsistent.

* Prior information can be combined with observed data to determine the final probability of the hypothesis

* Bayesian method can accomodate hypothesis that makes probabilistic predictions.

* New instances can be classified by combining predictions of multiple hypothesis weighted by their probabilities

Bayes Theorem:-

It provides a way to calculate the probab of hypothesis based on its prior probability, The probability of observing various data given the hypothesis, and the observed data itself

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

$P(h)$ → Prior P of h

$P(D)$ → marginal P

$P(D)$ = prior probability that training Data D will be observed, i.e the probability of D given no knowledge about which hypothesis holds.

$P(D/h)$ = probability of observing data D in which the hypothesis h, holds. We are interested in $P(h/D)$ ie the prob that h holds given the observed training data D. It is also called posterior prob of h

Many a times we have set of hypothesis h and we are interesting in finding the most probable hypothesis Data D. Any such maximally probable hypothesis is called maximum a posterious hypothesis (MAP)

$$h_{map} = \arg\max_{h \in H} P(h|D)$$

$$\Rightarrow \arg\max_{h \in H} \frac{P(D|h) \times P(h)}{P(D)}$$

maximum a posterior ~~probability~~ Hypothesis.

$$h_{map} = \arg\max P(D|h) \times P(h)$$

Consider a medical diagnosis problem in which ~~there~~ there are 2 alternative hypothesis

→ That the patient has particular form of cancer
→ That ~~the~~ ~~doesn't~~ doesn't have.

→ The ~~Available~~ data is from a ~~pro~~ particular with 2 possible outcome +ve or -ve. There is prior knowledge a
also a lab ~~test~~.

The test returns a possible result in only 98%.

→ The available data is from a particular lab test with 2 possible outcomes i.e +ve and -ve. There is prior knowledge that over entire population of people only 0.008 have this Disease also lab test is only an imperfect indicator of this disease. The test returns a correct +ve result in only 98% of the cases in which the disease is actually present and a correct -ve result is only 97% of cases in which the disease is not present. In other cases the test returns the opposite result. Suppose we have a new patient for whom the lab test returns a +ve result should be diagnosed the patient has having cancer or not.

conditional probability. cancer is there but it is +ve in 98%.

18/01/2024

Consider ⊖

Solu^{n} :- $P(\text{cancer}) = 0.008$ $P(\oplus/\text{cancer}) = 0.98$ $P(\ominus/\sim\text{cancer}) = 0.9$

+ve already when they have cancer $P(\oplus/\sim\text{cancer}) = 0.0$

$P(\sim\text{cancer}) = 0.992$ $1-0.008$ $1-0.98$

They have cancer & give -ve result, $P(\ominus/\text{cancer}) = 0.02$

$h_1 = \text{cancer}$ ~~$h_2 = \text{not cancer}$~~ $h_2 = \sim\text{cancer}$

$h_{map} = \underset{h \in H}{\arg\max}\ P(h/D)$

$h_{map} = P(D/h) \times P(h)$

$= P(\oplus/\text{cancer}) \cdot P(\text{cancer}) = 0.98 \times 0.00$

$= 0.0078.$

$h_2 map = P(\oplus/\sim\text{cancer}) * P(\sim\text{cancer})$

$= 0.03 * 0.992$

$= 0.0298$

$\boxed{h_{map} = \sim\text{Cancer}}$

19/01/2024 * Bayes Theorem and Concept learning.

$c: X \to \{0,1\}$

$\langle x_1 d_1 \rangle \langle x_2 d_2 \rangle \ldots \langle x_m d_m \rangle$

x_i = instances target = d_i

x_i	d_i
<Sun warm hot ...>	Yes
<Sun hot cold ...>	no

$d_i = c(x_i)$ based on the instance concept is learnt target is found

$\langle x_1 x_2 \ldots x_m \rangle$ - set of instances fixed.

$D = \{d_1, d_2 \ldots d_m\}$ last column of table mode as set which belongs to

Bruteforce map learning Algorithm.

Bayes Theorem and Concept learning

~~Concept~~ Consider a finite Hypothesis space H defined over an instance space X. Task is to learn some target concept $c: X \to \{0,1\}$

Let the sequence of training examples be $\{\langle x_1, d_1\rangle, \langle x_2 d_2 \rangle \ldots \langle x_m d_m \rangle\}$ where x_i is some instance from X and d_i is the target value of x_i, i.e $d_i = c(x_i)$

We assume that the sequences of instances $\langle x_1, x_2 \ldots x_m \rangle$ is held fixed so that the training data ~~set~~ D can be written just as a sequence of target values $D = \{d_1, d_2 \ldots d_m\}$

we can design a concept learning algorithm to output the map hypothesis based on bayes Theorem which we call it as. Brute force MAP learning Algorithm.

* Brute force MAP learning Algorithm

imp.

Step1:- for each hypothesis h in H calculate the posterior probability $P(h|D) = \frac{P(D|h) * P(h)}{P(D)}$

Step2:- Output the hypothesis h_{map} with the highest posterior probability i.e $h_{map} = \underset{h \in H}{\text{argmax}}\, P(h|D)$

$$h_{map} = \underset{h \in H}{\text{argmax}} \frac{P(D|h) * P(h)}{P(D)}$$

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

$h_1 = \ldots$ $h_2 \ldots$ $h_3 = \ldots$

$h_1 + h_2 + h_3 \ldots = 1$

when $d_i = h(x_i) \Rightarrow 1$ (Consistent)

$\neq \quad " \Rightarrow 0$

Assumptions.

Here we must specify what values are to be used for $P(h)$ and $P(D|h)$. In order to choose the probability distribution $P(h)$ & $P(D|h)$ we make the following assumptions.

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

Given these assumptions what values do we have to

Given that no prior knowledge is available that one hypothesis is more likely than an other. It is reasonable to assign the same prior probability to every hypothesis h in H. Also these prior probability sum to one. Hence we can write

$P(h) = \frac{1}{|H|} \quad \forall\ h \text{ in } H$

$P(D|h)$ is the probability of observing the target values $D = \{d_1, d_2 \ldots d_m\}$ for the fixed set of instances $\langle x_1, x_2 \ldots x_m \rangle$ given that hypothesis h holds. Since we assume noise free data the ~~following~~ Probability of observing the classification $d_i = h(x_i)$ and 0 if $d_i \neq h(x_i)$ $\therefore P(D|h)$

$\begin{cases} 1, & \text{if } d_i = h(x_i)\ \forall\ d_i \text{ in } D \\ 0, & \text{otherwise.} \end{cases}$

In other words the probability of data Given hypothesis h is 1 if d is ~~in~~ consistent

with h and 0 otherwise

Given these choices for $P(h)$ & $P(D/h)$ we now have a fully defined bruteforce map learning Algorithm.

Compute the posterior probability $P(h/D)$ of every hypothesis using Baye's Theorem

Case 1:- h is inconsistent with the Training data D, $P(h|D) = \frac{0 \times P(h)}{P(D)} = 0$ $P(D/h) = 0$.

The Posterior probability of hypothesis inconsistent with D is always 0.

Case 2: h is consistent with D.

$$P(h|D) = \frac{1 * 1/|H|}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

where $VS_{H,D}$ is a subset of Hypothesis from H that are consistent with D, ie it is the version space of with respect to D.

To summarize we can write

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

* with this we can say <u>every consistent hypothesis is a map hypothesis</u>

Note:- $h_{map} = \underset{h \in H}{\operatorname{argmax}}\ P(D|h) * P(h)$ when Priod P is not give we take some value & hence ignore it and write it as

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

In some cases we assume that every hypothesis in H is equally probable that is $P(h_i) = P(h_j)$ for all h_i & h_j belonging to H.

$P(h_i) = P(h_j) \; \forall \; h_i, h_j \in H$.

In this case further we simplify:

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

and need to consider only the term $P(D|h)$ to find the most probable hypothesis.

$P(D|h)$ is often called as the Likelihood of the $P(D|h)$ ——→ and any hypothesis that maximises $P(D|h)$ is called maximum likelihood hypothesis denoted by h_{ML}

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

22/01/2024

v.imp ML and LS Error Hypothesis

Maximum Likelihood & least squared Error

$f: X \rightarrow R$ (Continous valued target)

In the target by associating noise we can make it Continous valued.

we consider a problem of learning a continous value target function. Consider an instance space X and the hypothesis space H consisting of some class of Real valued functions over X i.e each h in H is of the form of function $h: X \rightarrow R$ where R is the set of Real numbers.

The problem faced by the learner L is to learn an unknown target function $f: X \rightarrow R$. A set of 'm' training examples are provided

p = probability density function

where the target f.value of each example is corrupted by a random noise drawn according to a normal probability distribution.

In other words each training ex is of the form $\langle x_i, d_i \rangle$ where $d_i = f(x_i) + e_i$, where $f(x_i)$ is noise free value of target func^n & e_i is the random variable representing the same

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

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

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

∴ we are now dealing with continous value

$(\text{Error})^2 = (\text{Actual} - \text{Predicted})^2$ OR $(\text{Predicted} - \text{Actual})^2$

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

$\therefore h_{ML} = LS$

We show this by considering ML hypothesis and we start with the equation.

→ we write the same equ^n with p to refer to the probability density

→ we are taking every instance as independent hence we are taking product. eg:- $P(A) \cdot P(B)$

probability Density of Normal Distribution $= p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

$\log e^x = x$

Assuming that training examples are mutual independent, give h we can write $P(D|h)$ as the product of various $p(d_i|h)$ that can be shown as

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

$p(D_i|h)$ can be written as a normal distribution with variance σ^2 and mean $\mu = f(x_i) \approx h(x_i)$

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

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

$\log(ab) = \log a + \log b$

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

constant we can discard.

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

$\Rightarrow$ independent of h

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

// maximizing -ve term same minimize +ve term

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

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

ML = least squared error