|| Jai Sri Gurudev |

Sri AdichunchanagiriShikshana Trust (R)

SJB INSTITUTE OF TECHNOLOGY

SJBIT

Study Material

Subject Name: Machine Learning

Subject Code: BCS602

By

Faculty Name: Mrs. Pavithra

Designation: Assistant Professor

Semester: VI

Department of Information Science & Engineering

Aca. Year: Even Sem /2024-25

MODULE 4

BAYESIAN LEARNING

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

INTRODUCTION

Bayesian learning methods are relevant to study of machine learning for two different reasons.

1. First, Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learningproblems
2. The second reason is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulateprobabilities.

Features of Bayesian Learning Methods

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any singleexample
- Prior knowledge can be combined with observed data to determine the finalprobability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting (1) a prior probability for each candidate hypothesis, and (2) a probability distribution over observed data for each possiblehypothesis.
- Bayesian methods can accommodate hypotheses that make probabilisticpredictions
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by theirprobabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Practical difficulty in applying Bayesian methods

1. One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advancetheyareoftenestimatedbasedonbackgroundknowledge,previouslyavailable data, and assumptions about the form of the underlyingdistributions.
2. A second practical difficulty is the significant computational cost required todetermine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantlyreduced.

BAYES THEOREM

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.

Notations

- P(h) prior probability of h, reflects any background knowledge about the chance that h iscorrect
- P(D) prior probability of D, probability that D will beobserved
- P(D|h) probability of observing D given a world in which hholds
- P(h|D) posterior probability of h, reflects confidence that h holds after D has been observed

Bayes theorem is the cornerstone of Bayesian learning methods because it 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).

Bayes Theorem:

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

- P(h|D) increases with P(h) and with P(D|h) according to Bayestheorem.
- 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 ofh.

Maximum a Posteriori (MAP) Hypothesis

- In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis h ∈ H given the observeddata D. Any such maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis.
- Bayestheoremtocalculatetheposteriorprobabilityofeachcandidatehypothesisish$_{MAP}$ is a MAP hypothesisprovided

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

- P(D) can be dropped, because it is a constant independent ofh

Maximum Likelihood (ML) Hypothesis

- In some cases, it is assumed that every hypothesis in H is equally probable apriori ($P(h_i) = P(h_j)$ for all h_i and h_j inH).
- Inthiscasethebelowequationcanbesimplifiedandneedonlyconsiderthetermp(D|h) to find the most probablehypothesis.

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

the equation can be simplified

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

P(D|h) is often called the likelihood of the data D given h, and any hypothesis that maximizes P(D|h) is called a maximum likelihood (ML)hypothesis

Example

- Consideramedicaldiagnosisprobleminwhichtherearetwoalternativehypotheses: (1) that the patient has particular form of cancer, and (2) that the patient does not. The available data is from a particular laboratory test with two possible outcomes: + (positive) and - (negative).

- We have prior knowledge that over the entire population of people only .008 have this disease. Furthermore, the lab test is only an imperfect indicator of thedisease.
- Thetestreturnsacorrectpositiveresultinonly98%ofthecasesinwhichthediseaseis actually present and a correct negative result in only 97% of the cases in which the disease is not present. In other cases, the test returns the oppositeresult.
- The above situation can be summarized by the followingprobabilities:

$$P(cancer) = .008 \qquad P(\neg cancer) = 0.992$$
$$P(\oplus|cancer) = .98 \qquad P(\ominus|cancer) = .02$$
$$P(\oplus|\neg cancer) = .03 \qquad P(\ominus|\neg cancer) = .97$$

Suppose a new patient is observed for whom the lab test returns a positive (+) result. Should we diagnose the patient as having cancer or not?

$$P(\oplus|cancer)P(cancer) = (.98).008 = .0078$$
$$P(\oplus|\neg cancer)P(\neg cancer) = (.03).992 = .0298$$
$$\Rightarrow h_{MAP} = \neg cancer$$

The exact posterior probabilities can also be determined by normalizing the above quantities so that they sum to 1

$$P(cancer|\oplus) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

$$P(\neg cancer|\oplus) = \frac{0.0298}{0.0078 + 0.0298} = 0.79$$

Basic formulas for calculating probabilities are summarized in Table

---

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

BAYES THEOREM AND CONCEPT LEARNING

What is the relationship between Bayes theorem and the problem of concept learning?

Since Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data, and can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable.

Brute-Force Bayes Concept Learning

Consider the concept learning problem

- Assume the learner considers some finite hypothesis space H defined over the instance space X, in which the task is to learn some target concept $c : X \rightarrow \{0,1\}$.
- Learner is given some sequence of training examples $((x_1, d_1) \ldots (x_m, d_m))$ where x_i is some instance from X and where d_i is the target value of x_i (i.e., $d_i = c(x_i)$).
- The sequence of target values are written as $D = (d_1 \ldots d_m)$.

Wecandesignastraightforwardconceptlearningalgorithmtooutputthemaximumaposteriori hypothesis, based on Bayes theorem, asfollows:

BRUTE-FORCE MAP LEARNING algorithm:

1. For each hypothesis h in H, calculate the posteriorprobability

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

2. Output the hypothesis h_{MAP} with the highest posteriorprobability

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

In order specify a learning problem for the BRUTE-FORCE MAP LEARNING algorithm we must specify what values are to be used for P(h) and for P(D|h) ?

Let's choose P(h) and for P(D|h) to be consistent with the following assumptions:

- The training data D is noise free (i.e., $d_i = c(x_i)$)
- The target concept c is contained in the hypothesis spaceH
- Do not have a priori reason to believe that any hypothesis is more probable than any other.

What values should we specify for P(h)?

- Given no prior knowledge that one hypothesis is more likely than another, it is reasonable to assign the same prior probability to every hypothesis h inH.
- Assume the target concept is contained in H and require that these prior probabilities sum to1.

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

What choice shall we make for P(D|h)?

- P(D|h) is the probability of observing the target values D = ($d_1 \ldots d_m$) for the fixed set of instances ($x_1 \ldots x_m$), given a world in which hypothesis hholds
- Since we assume noise-free training data, the probability of observing classification d_i given h is just 1 if $d_i = h(x_i)$ and 0 if $d_i \neq h(x_i)$.Therefore,

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

GiventhesechoicesforP(h)andforP(D|h)wenowhaveafully-definedproblemfortheabove BRUTE-FORCE MAP LEARNINGalgorithm.

Recalling Bayes theorem, we have

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

Consider the case where h is inconsistent with the training data D

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

The posterior probability of a hypothesis inconsistent with D is zero

Consider the case where h is consistent with D

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

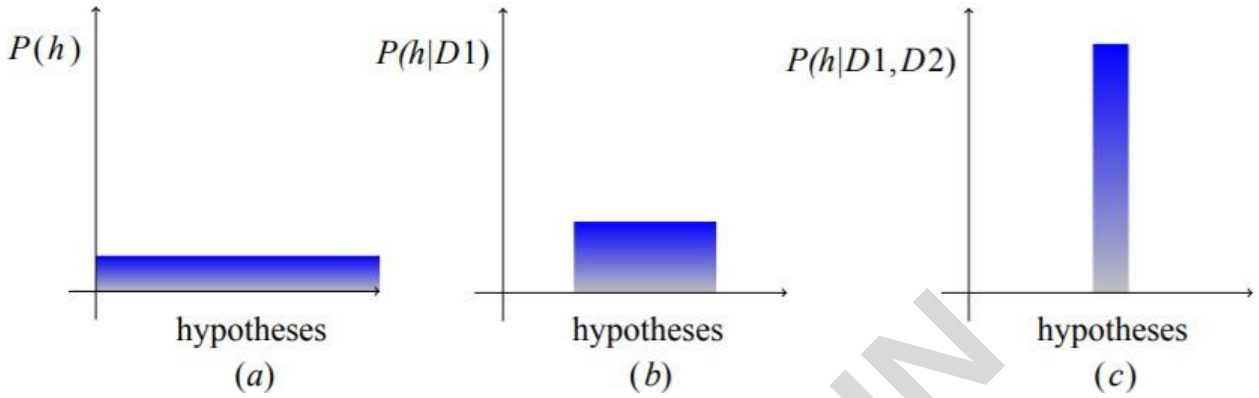
Where, $VS_{H,D}$ is the subset of hypotheses from H that are consistent with D

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

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

The Evolution of Probabilities Associated withHypotheses

- Figure (a) all hypotheses have the sameprobability.
- Figures (b) and (c), As training data accumulates, the posterior probability for inconsistent hypotheses becomes zero while the total probability summing to 1 is shared equally among the remaining consistenthypotheses.



MAP Hypotheses and Consistent Learners

- A learning algorithm is a consistent learner if it outputs a hypothesis that commits zero errors over the trainingexamples.
- Every consistent learner outputs a MAP hypothesis, if we assume a uniform prior probability distribution over H ($P(h_i) = P(h_j)$ for all i, j), and deterministic, noise free training data (P(D|h) =1 if D and h are consistent, and 0otherwise).

Example:

- FIND-S outputs a consistent hypothesis, it will output a MAP hypothesis under the probability distributions P(h) and P(D|h) definedabove.
- Are there other probability distributions for P(h) and P(D|h) under which FIND-S outputs MAP hypotheses?Yes.
- Because FIND-S outputs a maximally specific hypothesis from the version space, its outputhypothesiswillbeaMAPhypothesisrelativetoanypriorprobabilitydistribution that favours more specifichypotheses.

Note

- Bayesian framework is a way to characterize the behaviour of learningalgorithms
- By identifying probability distributions P(h) and P(D|h) under which the output is a optimal hypothesis, implicit assumptions of the algorithm can be characterized (Inductive Bias)
- Inductive inference is modelled by an equivalent probabilistic reasoning system based on Bayestheorem

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Consider the problem of learning a *continuous-valued target function* such as neural network learning, linear regression, and polynomial curve fitting

A straightforward 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 (ML) hypothesis*

- Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X, i.e., (∀ h ∈ H)[h : X → R] and training examples of the form<x_i,d_i>
- The problem faced by L is to learn an unknown target function f : X →R
- A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution with zero mean ($d_i = f(x_i) + e_i$)
- Each training example is a pair of the form (x_i ,d_i) where $d_i = f(x_i) + e_i$.
  - Here$f(x_i)$isthenoise-freevalueofthetargetfunctionand$e_i$isarandomvariable representing thenoise.
  - It is assumed that the values of the e_i are drawn independently and that they are distributed according to a Normal distribution with zero mean.
- The task of the learner is to *output a maximum likelihood hypothesis* or a*MAP hypothesis assuming all hypotheses are equally probable apriori*.

Using the definition of h_{ML} we have

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

Assuming training examples are mutually independent given h, we can write P(D|h) as the product of the various (d_i|h)

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

Given the noise e_i obeys a Normal distribution with zero mean and unknown variance σ^2, each d_i must also obey a Normal distribution around the true target value $f(x_i)$. Because we are writing the expression for P(D|h), we assume h is the correct description off.

Hence, $\mu = f(x_i) = h(x_i)$

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

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

Maximize the less complicated logarithm, which is justified because of the monotonicity of function p

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

The first term in this expression is a constant independent of h, and can therefore be discarded, yielding

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

Maximizing this negative quantity is equivalent to minimizing the corresponding positive quantity

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

Finally, discard constants that are independent of h.

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

Thus, above equation shows that the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between the observed training values d_i and the hypothesis predictions $h(x_i)$

Note:

Why is it reasonable to choose the Normal distribution to characterize noise?

- Good approximation of many types of noise in physicalsystems
- Central Limit Theorem shows that the sum of a sufficiently large number of independent,identicallydistributedrandomvariablesitselfobeysaNormaldistribution

Only noise in the target value is considered, not in the attributes describing the instances themselves

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- Consider the setting in which we wish to learn a nondeterministic(probabilistic) function f : X → {0, 1}, which has two discrete outputvalues.
- We want a function approximator whose output is the probability that f(x) = 1. Inother words, learn the target function f ` : X → [0, 1] such that f ` (x) = P(f(x) = 1)

How can we learn f `using a neural network?

- Use of brute force way would be to first collect the observed frequencies of 1's and 0's for each possible value of x and to then train the neural network to output the target frequency for eachx.

What criterion should we optimize in order to find a maximum likelihood hypothesis for f' in this setting?

- First obtain an expression forP(D|h)
- Assume the training data D is of the form D = {(x_1, d_1) . . . (x_m, d_m)}, where d_i is the observed 0 or 1 value for f(x_i).
- Both x_i and d_i as random variables, and assuming that each training example is drawn independently, we can write P(D|h)as

$$P(D \mid h) = \prod_{i=1}^{m} P(x_i, d_i \mid h) \qquad \text{equ (1)}$$

Applying the productrule

$$P(D \mid h) = \prod_{i=1}^{m} P(d_i \mid h, x_i) P(x_i) \qquad \text{equ (2)}$$

The probability $P(d_i|h,x_i)$

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

Re-express it in a more mathematically manipulable form, as

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

Equation (4) to substitute for P(di |h, xi) in Equation (5) to obtain

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \qquad \text{equ (5)}$$

We write an expression for the maximum likelihood hypothesis

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

The last term is a constant independent of h, so it can be dropped

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \qquad \text{equ (6)}$$

It easier to work with the log of the likelihood, yielding

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \qquad \text{equ (7)}$$

Equation (7) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting

Gradient Search to Maximize Likelihood in a Neural Net

- Deriveaweight-trainingruleforneuralnetworklearningthatseekstomaximizeG(h,D) using gradientascent
- The gradient of G(h,D) is given by the vector of partial derivatives of G(h,D) with respect to the various network weights that define the hypothesis h represented by the learnednetwork
- In this case, the partial derivative of G(h, D) with respect to weight w_{jk} from input k to unit jis

$$\begin{aligned} \frac{\partial G(h,D)}{\partial w_{jk}} &= \sum_{i=1}^{m} \frac{\partial G(h, D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}} \end{aligned} \qquad \text{equ (1)}$$

- Suppose our neural network is constructed from a single layer of sigmoid units.Then,

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i) x_{ijk} = h(x_i)(1 - h(x_i)) x_{ijk}$$

where x_{ijk} is the k^{th} input to unit j for the i^{th} training example, and d(x) is the derivative of the sigmoid squashing function.

- Finally,substitutingthisexpressionintoEquation(1),weobtainasimpleexpressionfor the derivatives that constitute thegradient

$$\frac{\partial G(h, D)}{\partial w_{jk}} = \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk}$$

Because we seek to maximize rather than minimize P(D|h), we perform gradient ascent rather than gradient descent search. On each iteration of the search the weight vector is adjusted in the direction of the gradient, using the weight update rule

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

Where,

$$\Delta w_{jk} = \eta \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk} \qquad \text{equ (2)}$$

Where, η is a small positive constant that determines the step size of the i gradient ascent search

MINIMUM DESCRIPTION LENGTH PRINCIPLE

- A Bayesian perspective on Occam'srazor
- Motivated by interpreting the definition of h_{MAP} in the light of basic concepts from informationtheory.

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

which can be equivalently expressed in terms of maximizing the $\log_2$

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

or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{argmin}\ -\log_2 P(D|h) - \log_2 P(h) \qquad \text{equ (1)}$$

This equation (1) can be interpreted as a statement that short hypotheses are preferred, assuming a particular representation scheme for encoding hypotheses and data

- $-\log_2 P(h)$: the description length of h under the optimal encoding for the hypothesis space H, $L_{CH}(h) = -\log_2 P(h)$, where C_H is the optimal code for hypothesis spaceH.
- $-\log_2 P(D|h)$:thedescriptionlengthofthetrainingdataDgivenhypothesish,underthe optimal encoding from the hypothesis space H: $L_{CH}(D|h) = -\log_2 P(D|h)$, where $C_{D|h}$ is the optimal code for describing data D assuming that both the sender and receiver know the hypothesish.
- RewriteEquation(1)toshowthath_{MAP}isthehypothesishthatminimizesthesumgiven by the description length of the hypothesis plus the description length of the data given thehypothesis.

$$h_{MAP} = \underset{h \in H}{argmin}\ L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

Where, C_H and $C_{D|h}$ are the optimal encodings for H and for D given h

The Minimum Description Length (MDL) principle recommends choosing the hypothesis that minimizes the sum of these two description lengths of equ.

$$h_{MAP} = \underset{h \in H}{argmin}\ L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

Minimum Description Length principle:

$$h_{MDL} = \underset{h \in H}{\text{argmin}}\ L_{C_1}(h) + L_{C_2}(D \mid h)$$

Where, codes C_1 and C_2 to represent the hypothesis and the data given the hypothesis

The above analysis shows that if we choose C_1 to be the optimal encoding of hypotheses C_H, and if we choose C_2 to be the optimal encoding $C_{D|h}$, then $h_{MDL} = h_{MAP}$

Application to Decision Tree Learning

Apply the MDL principle to the problem of learning decision trees from some training data. *What should we choose for the representations C1 and C2 of hypotheses and data?*

- For C_1: C_1 might be some obvious encoding, in which the description length grows with the number of nodes and with the number of edges
- For C_2: Suppose that the sequence of instances (x_1 . . . x_m) is already known to both the transmitter and receiver, so that we need only transmit the classifications (f (x_1) . . . f (x_m)).
- Now if the training classifications (f (x_1) . . . f(x_m)) are identical to the predictions of the hypothesis, then there is no need to transmit any information about these examples. The description length of the classifications given the hypothesis ZERO
- If examples are misclassified by h, then for each misclassification we need to transmit a message that identifies which example is misclassified as well as its correct classification
- The hypothesis h_{MDL} under the encoding C_1 and C_2 is just the one that minimizes the sum of these description lengths.

NAIVE BAYES CLASSIFIER

- The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f (x) can take on any value from some finite setV.
- A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values (a_1, a_2...a_m).
- The learner is asked to predict the target value, or classification, for this newinstance.

The Bayesian approach to classifying the new instance is to assign the most probable target value, V_{MAP}, given the attribute values (a_1, a_2.. .a_m) that describe the instance

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

Use Bayes theorem to rewrite this expression as

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

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \qquad \textbf{equ (1)}$$

- The naive Bayes classifier is based on the assumption that the attribute values are conditionally independent given the target value. Means, the assumption is that given thetargetvalueoftheinstance,theprobabilityofobservingtheconjunction(a_1,a_2...a_m), is just the product of the probabilities for the individualattributes:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

Substituting this into Equation (1),

Naive Bayes classifier:

$$\mathbf{V_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)} \qquad \text{equ (2)}$$

Where, V_{NB} denotes the target value output by the naive Bayes classifier

An Illustrative Example

- Let us apply the naive Bayes classifier to a concept learning problem i.e., classifying days according to whether someone will playtennis.
- Thebelowtableprovidesasetof14trainingexamplesofthetargetconcept***PlayTennis***, where each day is described by the attributes Outlook, Temperature, Humidity, and Wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Use the naive Bayes classifier and the training data from this table to classify the following novelinstance:

  < Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong >

- Our task is to predict the target value ***(yes or no)*** of the target concept ***PlayTennis***for this newinstance

$$V_{NB} = \underset{v_j \in \{yes, no\}}{\text{argmax}}\ P(v_j) \prod_i P(a_i | v_j)$$

$$V_{NB} = \underset{v_j \in \{yes, no\}}{\text{argmax}}\ P(v_j)\ P(\text{Outlook=sunny}|v_j)\ P(\text{Temperature=cool}|v_j)\ P(\text{Humidity=high}|v_j)\ P(\text{Wind=strong}|v_j)$$

The probabilities of the different target values can easily be estimated based on their frequencies over the 14 training examples

- P(P1ayTennis = yes) = 9/14 =0.64
- P(P1ayTennis = no) = 5/14 =0.36

Similarly, estimate the conditional probabilities. For example, those for Wind = strong

- P(Wind = strong | PlayTennis = yes) = 3/9 =0.33
- P(Wind = strong | PlayTennis = no) = 3/5 = 0.60

Calculate V_{NB} according to Equation(1)

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$
$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Thus, the naive Bayes classifier assigns the target value ***PlayTennis*** = ***no*** to this new instance, based on the probability estimates learned from the training data.

By normalizing the above quantities to sum to one, calculate the conditional probability that the target value is ***no***, given the observed attribute values

$$\frac{.0206}{(.0206 + .0053)} = .795$$

Estimating Probabilities

- We have estimated probabilities by the fraction of times the event is observed to occur over the total number ofopportunities.
- For example, in the above case we estimated P(Wind = strong | Play Tennis = no) bythe fraction n_c /n where, n = 5 is the total number of training examples for which PlayTennis = no, and n_c = 3 is the number of these for which Wind =strong.
- Whenn_c=0,thenn_c/nwillbezeroandthisprobabilitytermwilldominatethequantity calculated in Equation (2) requires multiplying all the other probability terms by this zerovalue
- ToavoidthisdifficultywecanadoptaBayesianapproachtoestimatingtheprobability, using the ***m-estimate*** defined asfollows

m **-estimate of probability:**

$$\frac{n_c + mp}{n + m}$$

- p is our prior estimate of the probability we wish to determine, and m is a constant called the equivalent sample size, which determines how heavily to weight p relative to the observeddata
- Method for choosing p in the absence of other information is to assume uniform priors; that is, if an attribute has k possible values we set p = 1/k.

BAYESIAN BELIEF NETWORKS

- ThenaiveBayesclassifiermakessignificantuseoftheassumptionthatthevaluesofthe attributes $a_1 \ldots a_n$ are conditionally independent given the target valuev.
- This assumption dramatically reduces the complexity of learning the targetfunction

A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities
Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables

Notation

- Consider an arbitrary set of random variables $Y_1 \ldots Y_n$, where each variable Yi can take on the set of possible values$V(Y_i)$.
- The joint space of the set of variables Y to be the cross product $V(Y_1)$ x $V(Y_2)$ x. . . $V(Y_n)$.
- In other words, each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables ($Y_1 \ldots Y_n$). The probability distribution over this joint' space is called the joint probabilitydistribution.
- The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple ($Y_1 \ldots Y_n$).
- A Bayesian belief network describes the joint probability distribution for a set of variables.

Conditional Independence

Let X, Y, and Z be three discrete-valued random variables. X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given a value for Z, that is, if

$$(\forall x_i, y_j, z_k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

Where,

$$x_i \in V(X),\ y_j \in V(Y),\ \text{and}\ z_k \in V(Z).$$

The above expression is written in abbreviated form as

$$P(X \mid Y, Z) = P(X \mid Z)$$

Conditional independence can be extended to sets of variables. The set of variables $X_1 \ldots X_l$ is conditionally independent of the set of variables $Y_1 \ldots Y_m$ given the set of variables $Z_1 \ldots Z_n$ if

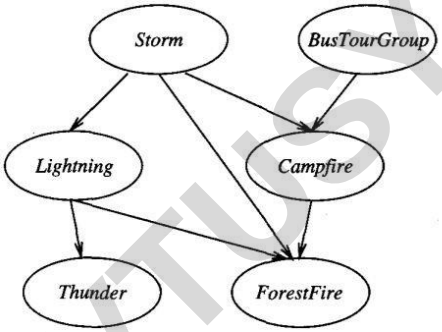
$$P(X_1 \ldots X_l | Y_1 \ldots Y_m, Z_1 \ldots Z_n) = P(X_1 \ldots X_l | Z_1 \ldots Z_n)$$

The naive Bayes classifier assumes that the instance attribute A_1 is conditionally independent of instance attribute A_2 given the target value V. This allows the naive Bayes classifier to calculate $P(A_l, A_2 \mid V)$ as follows,

$$\begin{aligned} P(A_1, A_2|V) &= P(A_1|A_2, V)P(A_2|V) \\ &= P(A_1|V)P(A_2|V) \end{aligned}$$

Representation

A Bayesian belief network represents the joint probability distribution for a set of variables. Bayesian networks (BN) are represented by directed acyclic graphs.



	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

Campfire

The Bayesian network in above figure represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, ForestFire, Campfire,* and *BusTourGroup*

A Bayesian network (BN) represents the joint probability distribution by specifying a set of *conditional independence assumptions*

- BN represented by a directed acyclic graph, together with sets of local conditional probabilities
- Each variable in the joint space is represented by a node in the Bayesiannetwork
- The network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the network given its immediate predecessors in thenetwork.
- A ***conditional probability table*** **(CPT)** is given for each variable, describing the probability distribution for that variable given the values of its immediatepredecessors

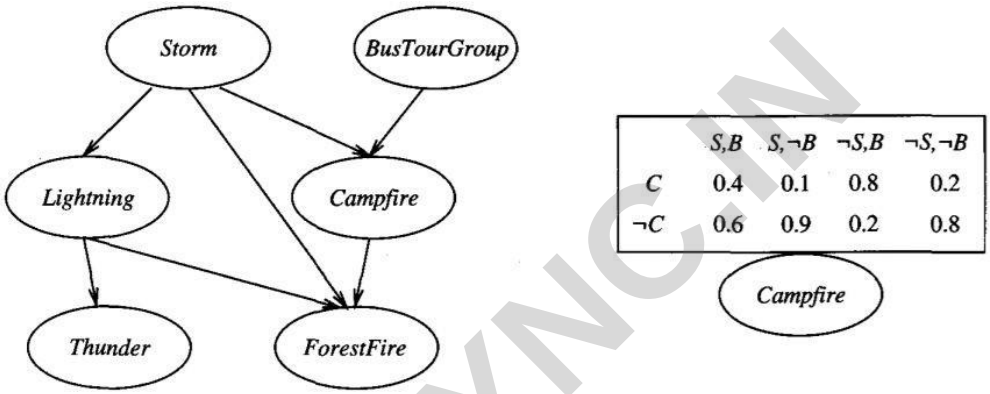
The joint probability for any desired assignment of values ($y_1, \ldots, y_n$) to the tuple of network variables ($Y_1 \ldots Y_m$) can be computed by the formula

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

Where, Parents(Y_i) denotes the set of immediate predecessors of Y_i in the network.

Example:

Consider the node ***Campfire***. The network nodes and arcs represent the assertion that ***Campfire*** is conditionally independent of its non-descendants ***Lightning*** and ***Thunder***, given its immediate parents Storm and ***BusTourGroup***.



This means that once we know the value of the variables ***Storm*** and ***BusTourGroup***, the variables ***Lightning*** and ***Thunder*** provide no additional information about ***Campfire***

The conditional probability table associated with the variable ***Campfire.*** The assertion is

P(Campfire = True | Storm = True, BusTourGroup = True) = 0.4

Inference

- Use a Bayesian network to infer the value of some target variable (e.g., ForestFire) given the observed values of the other variables.
- Inference can be straightforward if values for all of the other variables in the network are known exactly.
- A Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables.
- An arbitrary Bayesian network is known to be NP-hard

Learning Bayesian Belief Networks

Affective algorithms can be considered for learning Bayesian belief networks from training data by considering several different settings for learning problem

- First,thenetworkstructuremightbegiveninadvance,oritmighthavetobeinferredfrom the training data.
- Second, all the network variables might be directly observable in each training example, or some might be unobservable.
  - In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward and estimate the conditional probability tableentries
  - In the case where the network structure is given but only some of the variable values areobservableinthetrainingdata,thelearningproblemismoredifficult.Thelearning problem can be compared to learning weights for anANN.

Gradient Ascent Training of Bayesian Network

The gradient ascent rule which maximizes P(D|h) by following the gradient of ***ln P(D|h)*** with respecttotheparametersthatdefinetheconditionalprobabilitytablesoftheBayesiannetwork.

Let w_{ijk} denote a single entry in one of the conditional probability tables. In particular w_{ijk} denote the conditional probability that the network variable Y_i will take on the value y_i, given that its immediate parents U_i take on the values given by u_{ik}.

The gradient of ***ln P(D|h)*** is given by the derivatives $\frac{\partial \ln P(D|h)}{\partial w_{ijk}}$ for each of the w_{ijk}.
As shown below, each of these derivatives can be calculatedas

$$\frac{\partial \ln P(D|h)}{\partial w_{ij}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik}|d)}{w_{ijk}} \qquad \textbf{equ(1)}$$

Derive the gradient defined by the setofderivatives $\frac{\partial P_h(D)}{\partial w_{ijk}}$ for all ***i, j,*** and ***k***. Assuming the training examples ***d*** in the data set D are drawn independently, we write this derivativeas

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d)$$

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

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

We write the abbreviation $P_h(D)$ to represent $P(D|h)$.

This last step makes use of the general equality $\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$. We can now introduce the values of the variables Y_i and $U_i = Parents(Y_i)$, by summing over their possible values $y_{ij'}$ and $u_{ik'}$.

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

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

This last step follows from the product rule of probability . Now consider the rightmost sum in the final expression above. Given that $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$, the only term in this sum for which $\frac{\partial}{\partial w_{ijk}}$ is nonzero is the term for which $j' = j$ and $i' = i$. Therefore

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

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

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

Applying Bayes theorem to rewrite $P_h(d|y_{ij}, u_{ik})$, we have

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

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

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

$$= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}} \quad \text{equ (2)}$$

Thus, we have derived the gradient given in Equation (1). There is one more item that must be considered before we can state the gradient ascent training procedure. In particular, we require that as the weights w_{ijk} are updated they must remain valid probabilities in the interval [0,1]. We also require that the sum $\sum_j w_{ijk}$ remains 1 for all i, k. These constraints can be satisfied by updating weights in a two-step process. First we update each w_{ijk} by gradient ascent

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}}$$

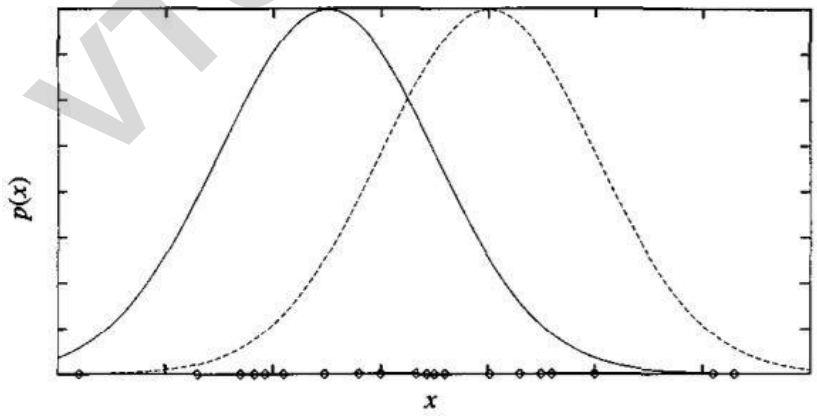
where η is a small constant called the learning rate. Second, we renormalize the weights w_{ijk} to assure that the above constraints are satisfied.
this process will converge to a locally maximum likelihood hypothesis for the conditional probabilities in the Bayesian network.

THE EM ALGORITHM

The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.

Estimating Means of k Gaussians

- Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normaldistributions.



- This problem setting is illustrated in Figure for the case where k = 2 and where the instances are the points shown along the xaxis.
- Each instance is generated using a two-stepprocess.
  - First, one of the k Normal distributions is selected atrandom.
  - Second, a single random instance x_i is generated according to this selected distribution.
- This process is repeated to generate a set of data points as shown in thefigure.

- To simplify, consider the specialcase
  - The selection of the single Normal distribution at each step is based onchoosing each with uniformprobability
  - Each of the k Normal distributions has the same variance σ^2, knownvalue.
- The learning task is to output a hypothesis h = (μ_1 , . . . ,μ_k) that describes the means of each of the kdistributions.
- We would like to find a maximum likelihood hypothesis for these means; that is, a hypothesis ***h*** that maximizes p(D|h).

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2 \qquad (1)$$

In this case, the sum of squared errors is minimized by the sample mean

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad (2)$$

- Ourproblemhere,however,involvesamixtureofkdifferentNormaldistributions,and we cannot observe which instances were generated by whichdistribution.
- Consider full description of each instance as the triple (x_i, z_{i1},z_{i2}),
  - where x_i is the observed value of the ith instanceand
  - where z_{i1} and z_{i2} indicate which of the two Normal distributions was used to generate the valuex_i
- In particular, z_{ij} has the value 1 if x_i was created by the j^{th} Normal distribution and 0 otherwise.
- Here x_i is the observed variable in the description of the instance, and z_{il} and z_{i2} are hiddenvariables.
- If the values of z_{il} and z_{i2} were observed, we could use following Equation to solve for the means p_1 andp_2
- Because they are not, we will instead use the EMalgorithm

EM algorithm

Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

Step 2: Calculate a new maximum likelihood hypothesis $h' = \langle \mu_1', \mu_2' \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu_1', \mu_2' \rangle$ and iterate.

Let us examine how both of these steps can be implemented in practice. Step 1 must calculate the expected value of each z_{ij}. This $E[z_{ij}]$ is just the probability that instance x_i was generated by the jth Normal distribution

$$E[z_{ij}] = \frac{p(x = x_i|\mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i|\mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i-\mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i-\mu_n)^2}}$$

Thus the first step is implemented by substituting the current values $\langle \mu_1, \mu_2 \rangle$ and the observed x_i into the above expression.

In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$. maximum likelihood hypothesis in this case is given by

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}]\ x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

ARTIFICIAL NEURAL NETWORKS

INTRODUCTION

Artificialneuralnetworks(ANNs)provideageneral,practicalmethodforlearningreal-valued, discrete-valued, and vector-valued targetfunctions.

Biological Motivation

- Thestudyofartificialneuralnetworks(ANNs)hasbeeninspiredbytheobservationthat biological learning systems are built of very complex webs of interconnected***Neurons***
- Human information processing system consists of brain ***neuron***: basic building block cell that communicates information to and from various parts ofbody

Facts of Human Neurobiology

- Number of neurons ~ 10^{11}
- Connection per neuron ~ 10^{4-5}
- Neuron switching time ~ 0.001 second or 10^{-3}
- Scene recognition time ~ 0.1second
- 100 inference steps doesn't seem likeenough
- Highly parallel computation based on distributedrepresentation

Properties of Neural Networks

- Many neuron-like threshold switchingunits
- Many weighted interconnections amongunits
- Highly parallel, distributedprocess
- Emphasis on tuning weightsautomatically
- Input is a high-dimensional discrete or real-valued (e.g, sensor input)

NEURAL NETWORK REPRESENTATIONS

- A prototypical example of ANN learning is provided by Pomerleau's systemALVINN, which uses a learned ANN to steer an autonomous vehicle driving at normal speeds on publichighways
- The input to the neural network is a 30x32 grid of pixel intensities obtained from a forward-pointed camera mounted on thevehicle.
- The network output is the direction in which the vehicle issteered

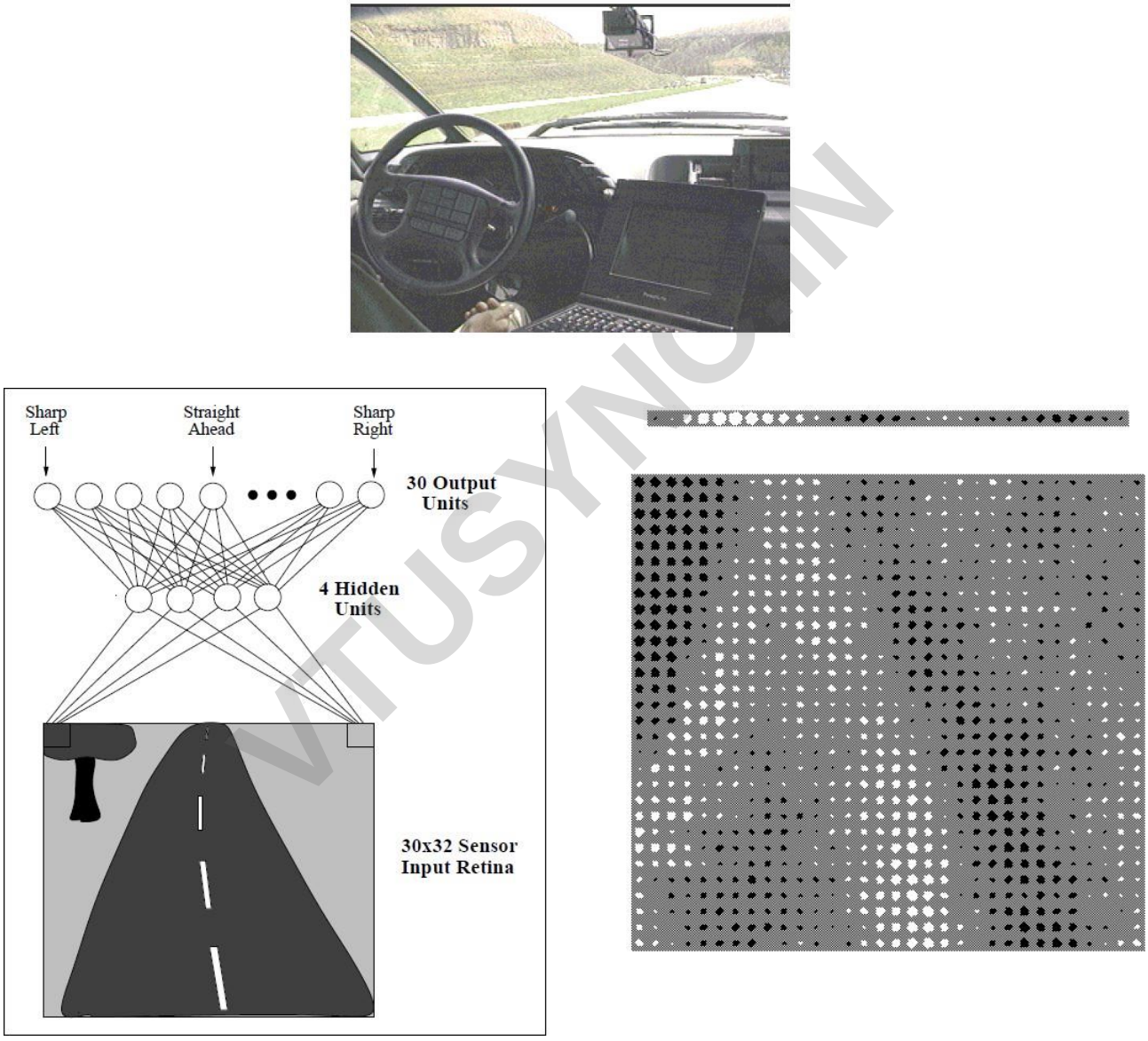


Figure: Neural network learning to steer an autonomous vehicle.

- Figure illustrates the neural networkrepresentation.
- Thenetworkisshownontheleftsideofthefigure,withtheinputcameraimagedepicted belowit.
- Each node (i.e., circle) in the network diagram corresponds to the output of a single network unit, and the lines entering the node from below are itsinputs.
- There are four units that receive inputs directly from all of the 30 x 32 pixels in the image. These are called "hidden" units because their output is available only within the network and is not available as part of the global network output. Each of these four hidden units computes a single real-valued output based on a weighted combination of its 960inputs
- Thesehiddenunitoutputsarethenusedasinputstoasecondlayerof30"output"units.
- Each output unit corresponds to a particular steering direction, and the output values of these units determine which steering direction is recommended moststrongly.
- The diagrams on the right side of the figure depict the learned weight valuesassociated with one of the four hidden units in thisANN.
- The large matrix of black and white boxes on the lower right depicts the weights from the 30 x 32 pixel inputs into the hidden unit. Here, a white box indicates a positive weight, a black box a negative weight, and the size of the box indicates the weight magnitude.
- Thesmallerrectangulardiagramdirectlyabovethelargematrixshowstheweightsfrom this hidden unit to each of the 30 outputunits.

APPROPRIATE PROBLEMS FOR NEURAL NETWORK LEARNING

ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones.

ANN is appropriate for problems with the following characteristics:

1. Instances are represented by many attribute-valuepairs.
2. The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valuedattributes.
3. The training examples may containerrors.
4. Long training times areacceptable.
5. Fast evaluation of the learned target function may berequired
6. The ability of humans to understand the learned target function is notimportant

PERCEPTRON

- One type of ANN system is based on a unit called a perceptron. Perceptron is a single layer neural network.

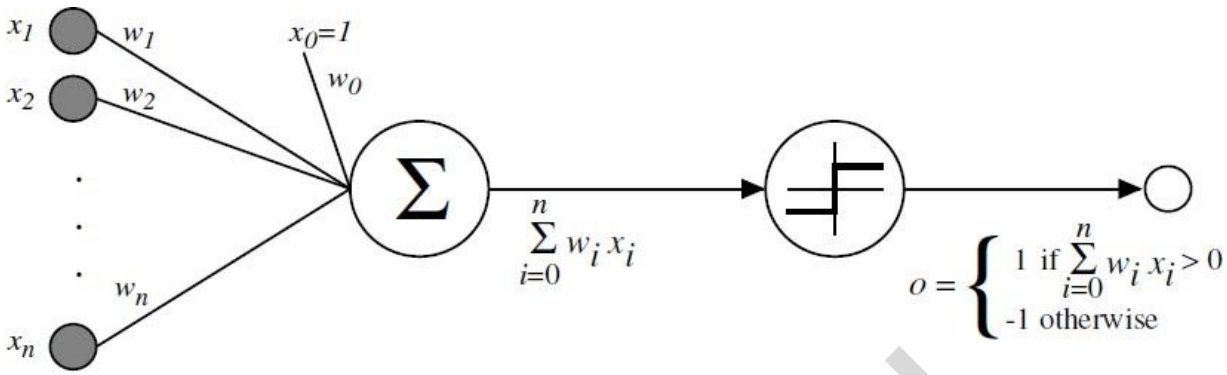


Figure: A perceptron

- A perceptron takes a vector of real-valued inputs, calculates a linear combination of theseinputs,thenoutputsa1iftheresultisgreaterthansomethresholdand-1otherwise.
- Given inputs ***x*** through ***x,*** the output **O*(x₁, . . . , xₙ)*** computed by the perceptronis

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise.} \end{cases}$$

- Where, each $w_i$is a real-valued constant, or weight, that determines the contribution of input $x_i$to the perceptronoutput.
- $-w_0$isathresholdthattheweightedcombinationofinputs$w_1x_1+...+w_nx_n$mustsurpass in order for the perceptron to output a1.

Sometimes, the perceptron function is written as,

$$O(\vec{x}) = \text{sgn}\ (\vec{w} \cdot \vec{x})$$

Where,

$$\text{sgn}(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise.} \end{cases}$$

Learning a perceptron involves choosing values for the weights $w_0, \ldots, w_n$. Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors

$$H = \{\vec{w} \mid \vec{w} \in \Re^{(n+1)}\}$$

Representational Power of Perceptrons

- The perceptron can be viewed as representing a hyperplane decision surface in the n-dimensional space of instances (i.e., points)
- Theperceptronoutputsa1forinstanceslyingononesideofthehyperplaneandoutputs a -1 for instances lying on the other side, as illustrated in belowfigure

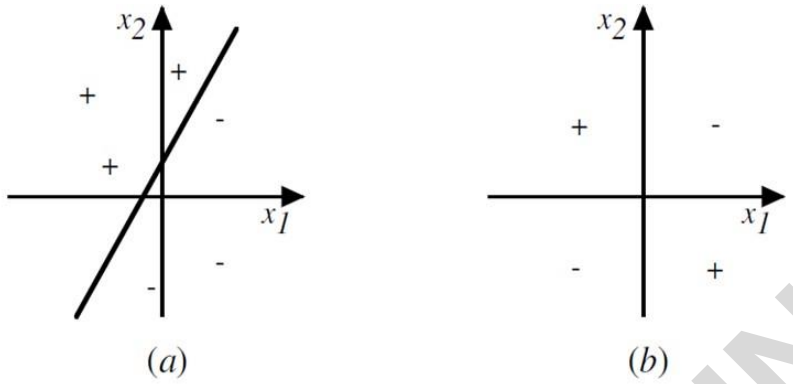
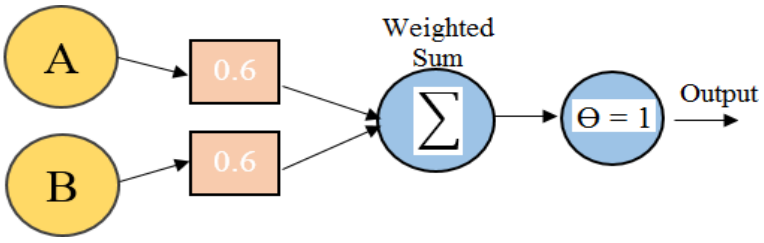


Figure : The decision surface represented by a two-input perceptron.
(a) A set of training examples and the decision surface of a perceptron that classifies them correctly. *(b)* A set of training examples that is not linearly separable.
x_1 and x_2 are the Perceptron inputs. Positive examples are indicated by "+", negative by "-".

Perceptrons can represent all of the primitive Boolean functions AND, OR, NAND (~ AND), and NOR (~OR)
Some Boolean functions cannot be represented by a single perceptron, such as the XOR function whose value is 1 if and only if $x_1 \neq x_2$

Example: Representation of AND functions

A	B	A ^ B
0	0	0
0	1	0
1	0	0
1	1	1



If A=0 & B=0 → 0*0.6 + 0*0.6 = 0.
This is not greater than the threshold of 1, so the output = 0.
If A=0 & B=1 → 0*0.6 + 1*0.6 = 0.6.
This is not greater than the threshold, so the output = 0.
If A=1 & B=0 → 1*0.6 + 0*0.6 = 0.6.
This is not greater than the threshold, so the output = 0.
If A=1 & B=1 → 1*0.6 + 1*0.6 = 1.2.
This exceeds the threshold, so the output = 1.

Drawback of perceptron

- The perceptron rule finds a successful weight vector when the training examples are linearly separable, it can fail to converge if the examples are not linearlyseparable

The Perceptron Training Rule

The learning problem is to determine a weight vector that causes the perceptron to produce the correct + 1 or - 1 output for each of the given training examples.

To learn an acceptable weight vector

- Begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies anexample.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examplescorrectly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight $w_i$associated with input xi according to therule.

$$w_i \leftarrow w_i + \Delta w_i$$

Where,

$$\Delta w_i = \eta(t - o)x_i$$

Here,
t is the target output for the current training example
o is the output generated by the perceptron
η is a positive constant called the ***learning rate***

- The role of the learning rate is to moderate the degree to which weights are changed at eachstep.Itisusuallysettosomesmallvalue(e.g.,0.1)andissometimesmadetodecay as the number of weight-tuning iterationsincreases

Drawback:
The perceptron rule finds a successful weight vector when the training examples are linearly separable, it can fail to converge if the examples are not linearly separable.