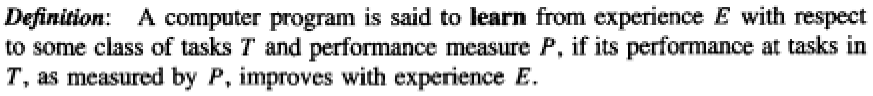
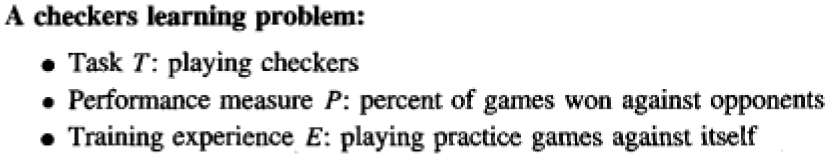
# Unit-1

1. Define well posed learning problem



***Definition***: A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

Example

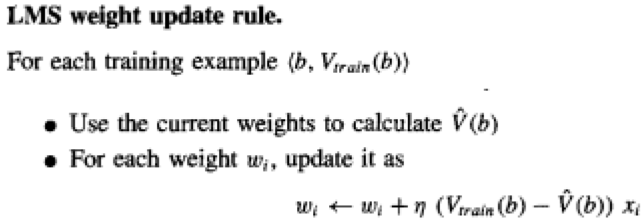


**A checkers learning problem:**

- Task *T*: playing checkers
- Performance measure *P*: percent of games won against opponents
- Training experience *E*: playing practice games against itself

2. List the steps in designing a learning system.
We should consider the performance measure in designing a learning system. The basic steps are
   - ➢ Choosing the Training Experience
   - ➢ Choosing the Target Function
   - ➢ Choosing a Representation for the Target Function
   - ➢ Choosing a Function Approximation Algorithm

3. Give LMS(least mean squares) weight update rule



**LMS weight update rule.**

For each training example $\langle b, V_{train}(b)\rangle$

- Use the current weights to calculate $\hat{V}(b)$
- For each weight $w_i$, update it as

$$w_i \leftarrow w_i + \eta\ (V_{train}(b) - \hat{V}(b))\ x_i$$

4. What can be the performance measure of the checker's learning program
The performance measure of a checker's learning program could be based on several factors such as accuracy, winning rate, loss rate, draw rate, average game length, computational efficiency, learning speed and adaptability.
It also depends on the specific objectives and requirements of the program.

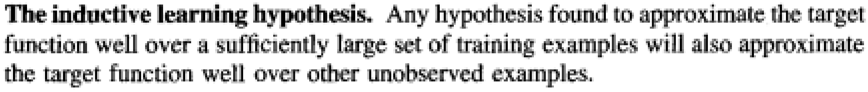
5. Define learning
Learning refers to algorithms or models that improve their performance on a task based on experience, data, or feedback. It can occur through various mechanisms, including supervised learning, unsupervised learning, reinforcement learning, and semi-supervised learning.

6. Give the applications of machine learning
Autonomous vehicle
Image recognition and computer vision
Recommendation systems
Health care
Fraud detection
Robotics

7. Define concept learning.
In machine learning, concept is more formally defined as "inferring a boolean-valued function from training examples of its inputs and outputs"

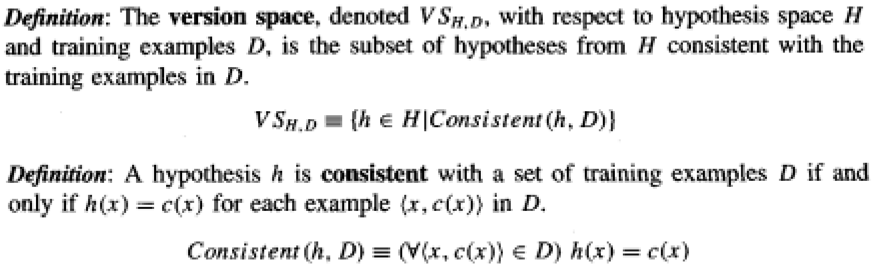
**<u>Example</u>**:
One possible target concept may be to **<u>find the day when my friend Ramesh enjoys his favorite sport.</u>** We have some attributes/features of the day like, Sky, Air Temperature, Humidity, Wind, Water, Forecast and based on this we have a target Concept named EnjoySport.

8. Define inductive learning hypothesis



**The inductive learning hypothesis.** Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

9. Define version space and consistent hypothesis



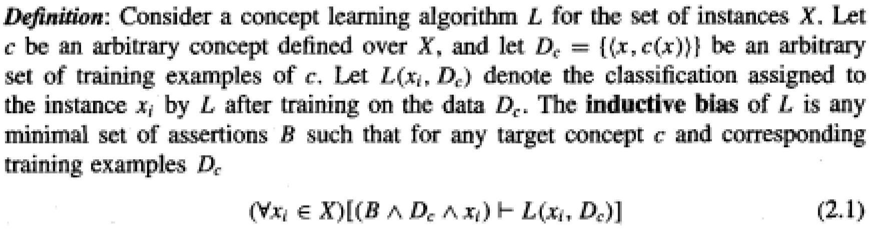
***Definition***: The **version space**, denoted $VS_{H,D}$, with respect to hypothesis space $H$ and training examples $D$, is the subset of hypotheses from $H$ consistent with the training examples in $D$.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

***Definition***: A hypothesis $h$ is **consistent** with a set of training examples $D$ if and only if $h(x) = c(x)$ for each example $\langle x, c(x)\rangle$ in $D$.

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x)$$

10. Define inductive bias

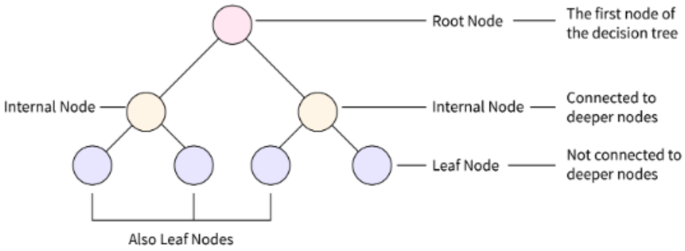


***Definition***: Consider a concept learning algorithm $L$ for the set of instances $X$. Let $c$ be an arbitrary concept defined over $X$, and let $D_c = \{\langle x, c(x)\rangle\}$ be an arbitrary set of training examples of $c$. Let $L(x_i, D_c)$ denote the classification assigned to the instance $x_i$ by $L$ after training on the data $D_c$. The **inductive bias** of $L$ is any minimal set of assertions $B$ such that for any target concept $c$ and corresponding training examples $D_c$

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)] \qquad (2.1)$$

11. What is a decision tree and how it works?
A decision tree is a hierarchical structure that represents a sequence of decisions and their possible consequences. It is a supervised learning algorithm used for classification and regression tasks.

A decision tree recursively splits the training data into subsets based on the values of input features. It selects the **best feature to split** on at each node **using a criterion** such as **information gain or Gini impurity**, aiming to maximize the homogeneity of the resulting subsets.


Root Node
The first node of the decision tree
Internal Node
Internal Node
Connected to deeper nodes
Leaf Node
Not connected to deeper nodes
Also Leaf Nodes

12. Define entropy

Entropy is a measure of impurity or disorder in a dataset. In decision trees, entropy is used to select the best splitting attribute

If the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

where $p_i$ is the proportion of S belonging to class *i*.

**Note:** The entropy is 0 if all members of S belong to the same class.
The entropy is 1 when the collection contains an equal number of positive and negative examples.
If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and 1.

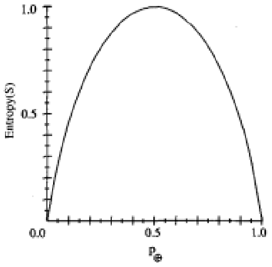


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

13. State the principle of Occams razor

A short hypothesis in machine learning refers to a simple, concise model or explanation that captures the underlying patterns or relationships in the data with minimal complexity.

The principle of Occam's Razor suggests that simpler explanations are generally preferred unless there is strong evidence to support more complex ones.

**Occam's razor:** Prefer the simplest hypothesis that fits the data.

14. What is overfitting in decision tree learning

Overfitting in machine learning refers to a model that models the training data too well. It captures the noise and the details in the training data to the extent that it negatively impacts the performance of the model on new data.

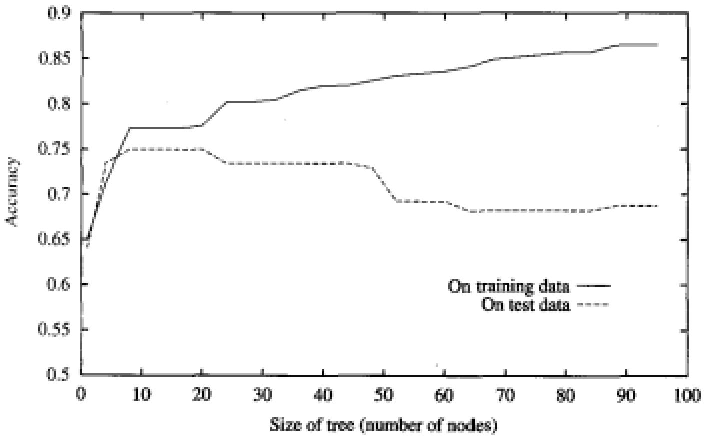


**FIGURE 3.6**
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases.

15. Define underfitting

    Underfitting in machine learning occurs when a model is too simple to capture the underlying pattern of the data or when the model has not been trained enough.

16. What is pruning

    Pruning is a technique used in machine learning, particularly in decision tree algorithms, to reduce the size of a tree by removing certain branches or nodes. The goal of pruning is to prevent overfitting and improve the generalization ability of the model.

    Pruning can be done in various ways, including:

    **Pre pruning:** Stopping the growth of the tree early, before it becomes too complex, based on certain criteria such as the maximum depth of the tree or the minimum number of samples required to split a node.

    **Post pruning:** Allowing the tree to grow to its full size and then removing branches that do not improve performance during a pruning phase.

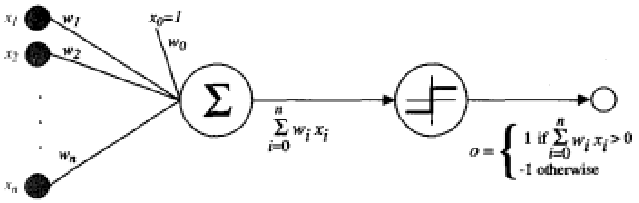
## Unit-2

1. Define **Artificial Neural Networks (ANNs)** and explain their basic structure.

   **ANNs** are computational models inspired by the structure and function of the human brain. They consist of interconnected nodes called neurons organized into layers. Each neuron receives input signals, processes them using an activation function, and produces an output signal. In a basic feedforward neural network, information flows in one direction from input to output layer through hidden layers.

2. What is a perceptron

   Perceptrons are the simplest form of neural networks, consisting of a single layer of input nodes connected to a single output node. Each connection is associated with a weight representing the strength of the connection. The output of a perceptron is calculated by applying an activation function (usually a step function) to the weighted sum of inputs.

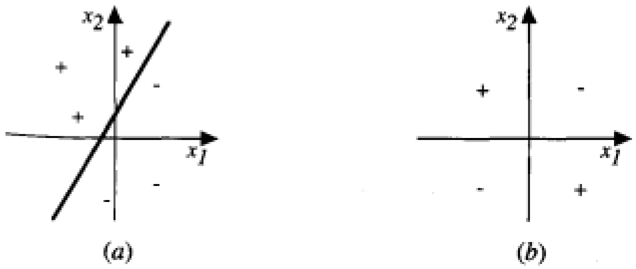


$x_1$
$w_1$
$x_0=1$
$w_0$
$x_2$
$w_2$
$w_n$
$x_n$
$\Sigma$
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

**FIGURE 4.2**
A perceptron.

3. What is a multi layer neural network
   Multilayer neural networks consist of multiple layers of neurons, including input, hidden, and output layers. Information flows from the input layer through one or more hidden layers to the output layer. Each neuron in the hidden layers and the output layer applies an activation function to the weighted sum of inputs from the previous layer.

4. Represent the decision surface of linearly seperable and non linearly seperable data



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 (i.e., that cannot be correctly classified by any straight line). $x_1$ 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 NOR. Unfortunately, however, some boolean functions cannot be represented by a single perceptron, such as the XOR function whose value is 1 if and only if xl !=x2. Note the set of linearly nonseparable training examples corresponds to this XOR function.

5. Define perceptron training rule
   It is basically used for binary classification tasks. The perceptron training rule is a learning algorithm used to train a single-layer perceptron, a type of artificial neural network.

   **Example:** Imagine we have a binary classification problem at hand, and we want to use a perceptron to learn this task. Perceptron can produce 2 values: +1 / -1 where +1 means that the input example belongs to the + class, and -1 means the input example belongs to the – class.
   As we have 2 classes, we would want to learn the weight vector of our perceptron in such a way that, for every training example the perceptron would produce the correct +1 / -1.
   **In the perceptron training rule,** we would initialize the weights at random and then feed the training examples into our perceptron and look at the produced output that

can be either +1 or -1! So, we would want the perceptron to produce +1 for one class and -1 for the other. After observing the output for a given training example, we will NOT modify the weights unless the produced output was wrong!

weight $w_i$ associated with input $x_i$ according to the rule

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

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, and $\eta$ 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 each step.

6. Define delta rule

   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.

   If the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the target concept.

   The delta rule is a simple learning algorithm used in the training of artificial neural networks, specifically for single-layer perceptrons. It adjusts the weights of the perceptron based on the difference between the actual output and the target output for each training example.

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

   Here *t* is the target output for the current training example, *o* is the output generated by the perceptron, and $\eta$ 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 each step.

   The delta rule uses an error function to perform gradient descent learning

   $$E(\vec{w}) \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

   where D is the set of training examples, $t_d$ is the target output for training example d, and $o_d$ is the output of the linear unit for training example d.

7. What is the main purpose of the backpropagation algorithm?

   The main purpose of the backpropagation algorithm is to train artificial neural networks by adjusting the weights of connections between neurons to minimize the error between the predicted output and the actual output.

8. Describe the key steps involved in the backpropagation algorithm.

   The key steps involved in the backpropagation algorithm include forward propagation, backward propagation of errors, calculation of gradients using the chain rule, and updating the weights using gradient descent.

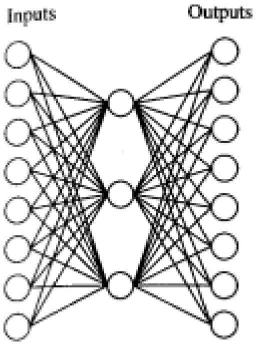
9. What is the role of the gradient descent optimization technique in backpropagation?
   Gradient descent optimization technique is used in backpropagation to adjust the weights of the neural network in the direction that minimizes the error between predicted and actual outputs.

10. How is the error calculated in the backpropagation algorithm?
    The error in backpropagation is typically calculated using a loss function such as mean squared error, which measures the difference between the predicted output and the actual output.

11. What is the significance of the activation function in backpropagation?
    The activation function in backpropagation introduces non-linearity into the network and helps in learning complex relationships between inputs and outputs.

12. Explain the concept of forward propagation in the context of backpropagation.
    Forward propagation involves passing the input data through the neural network to generate a predicted output, which is then compared to the actual output to compute the error.

13. Draw 8-3-8 artificial neural network **(Multi layer ANN)**
    It consists of 8 nodes in the input layer, 3 in the hidden layer and 8 in the output layer



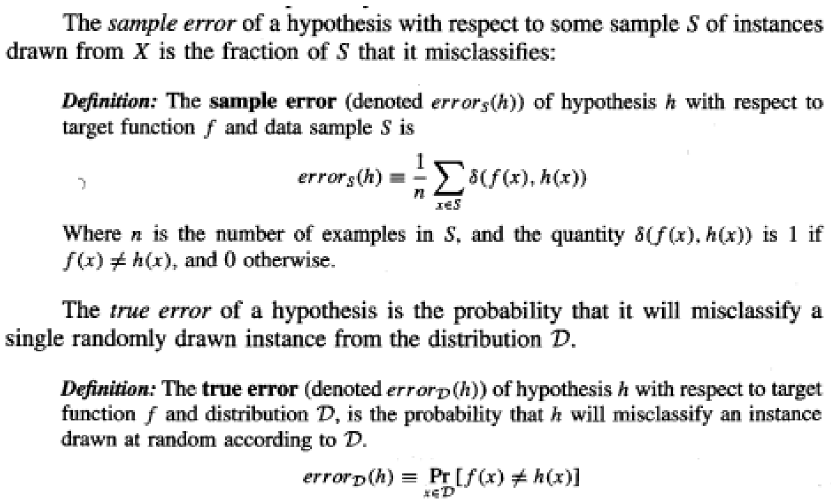
14. Define sample error, true error, confidence interval
    **Sample error** assesses the model's performance on the training data

    **True error** assesses its ability to generalize to new, unseen data (testing data).
    Achieving a low sample error is important, but minimizing the true error is the ultimate goal of machine learning model training.

    **Confidence interval**

    A confidence interval is a range of values derived from sample data that is likely to contain the true population parameter with a certain level of confidence

    For example, if we're estimating the average height of all people in a city based on a sample, a 95% confidence interval might be from 160 cm to 170 cm. This means that we're 95% confident that the true average height falls within this range.



The *sample error* of a hypothesis with respect to some sample $S$ of instances drawn from $X$ is the fraction of $S$ that it misclassifies:

***Definition:*** The **sample error** (denoted $error_S(h)$) of hypothesis $h$ with respect to target function $f$ and data sample $S$ is

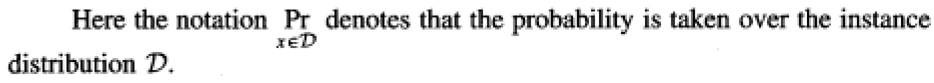
$$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$

Where $n$ is the number of examples in $S$, and the quantity $\delta(f(x), h(x))$ is 1 if $f(x) \neq h(x)$, and 0 otherwise.

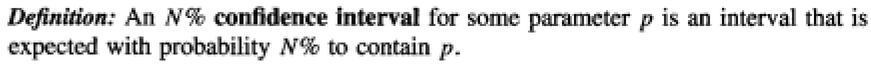
The *true error* of a hypothesis is the probability that it will misclassify a single randomly drawn instance from the distribution $\mathcal{D}$.

***Definition:*** The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis $h$ with respect to target function $f$ and distribution $\mathcal{D}$, is the probability that $h$ will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$



Here the notation $\Pr_{x \in \mathcal{D}}$ denotes that the probability is taken over the instance distribution $\mathcal{D}$.

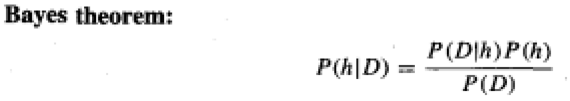


***Definition:*** An $N\%$ **confidence interval** for some parameter $p$ is an interval that is expected with probability $N\%$ to contain $p$.

## Unit-3

1. Give the importance of bayes theorem

   It is used to update the probability of a hypothesis based on new evidence.



**Bayes theorem:**

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

2. What is MAP hypothesis

   The learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis h Є H given the observed data D. Any such maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis.

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

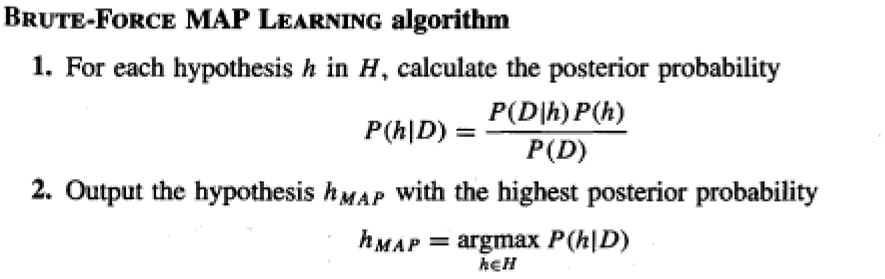
   $P(D|h)$ is often called the likelihood of the data D given h, and any hypothesis that maximizes $P(D|h)$ is called a **maximum likelihood (ML) hypothesis, hML**.

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

   In the final step above we dropped the term P(D) because it is a constant independent of h.

3. Define brute force bayes concept learning

   Brute-Force Bayes Concept Learning is a method used to learn a concept from training data by exhaustively considering and evaluating all possible hypotheses based on the Bayes' theorem.



BRUTE-FORCE MAP LEARNING algorithm

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

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

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

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

The process involves calculating the posterior probability of each hypothesis given the training data and selecting the hypothesis with the highest posterior probability as the learned concept.

4. Write two differences between lazy and eager learner’s

| **Lazy learners** | **Eager learners** |
|---|---|
| Lazy learners have a faster training time because they delay processing the training data until prediction time. They store the training instances and perform computations only when a prediction is needed. | Eager learners require more time during the training phase as they process the entire dataset upfront to construct a model. Once the model is built, predictions are generally faster compared to lazy learners. |
| Lazy learners typically consume less memory during the training phase since they only need to store the training instances. However, during prediction, they may require more memory to store the entire training dataset for comparison. | Eager learners often use more memory during the training phase because they need to store the constructed model, which may involve storing parameters, decision trees, or other representations of the learned knowledge. However, during prediction, they usually require less memory as they do not need to store the training instances. |

5. Define maximum likelihood principle
Any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood hypothesis

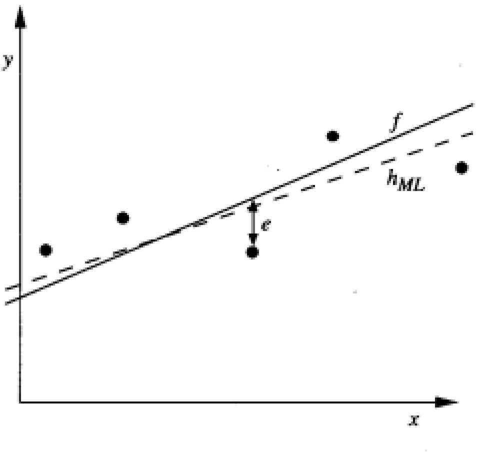


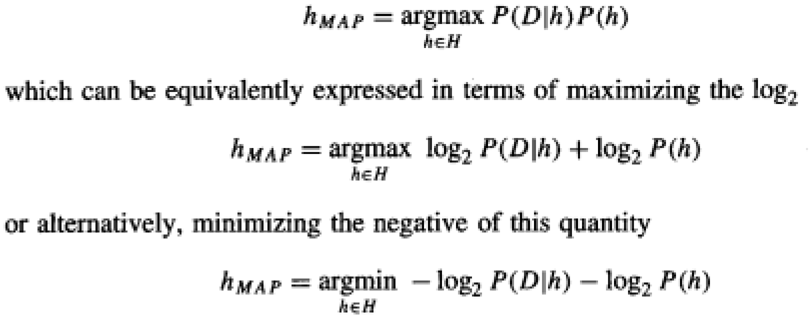
**FIGURE 6.2**
Learning a real-valued function. The target function $f$ corresponds to the solid line. The training examples $\langle x_i, d_i \rangle$ are assumed to have Normally distributed noise $e_i$ with zero mean added to the true target value $f(x_i)$. The dashed line corresponds to the linear function that minimizes the sum of squared errors. Therefore, it is the maximum likelihood hypothesis $h_{ML}$, given these five training examples.

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

It 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 $h(x_i)$.

6. What is the Minimum Description Length (MDL) principle
The best model for a given dataset is the one that minimizes the total length required to encode both the model and the data. In other words, it seeks to find the model that provides the most concise description of the data without sacrificing predictive accuracy.

Consider the definition of



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

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

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

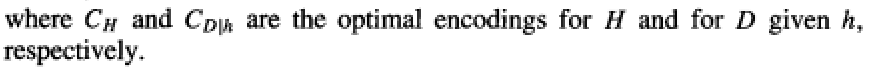
or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}}\ -\log_2 P(D|h) - \log_2 P(h)$$

The above equation can be interpreted as a statement that short hypotheses are preferred, assuming a particular representation scheme for encoding hypotheses and data.

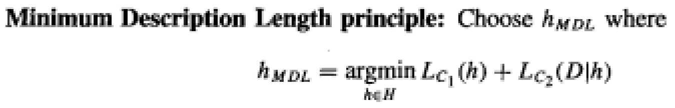
Rewrite the above equation to show that $h_{MAP}$ is the hypothesis h that minimizes the sum given by the description length of the hypothesis plus the description length of the data given the hypothesis.

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



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

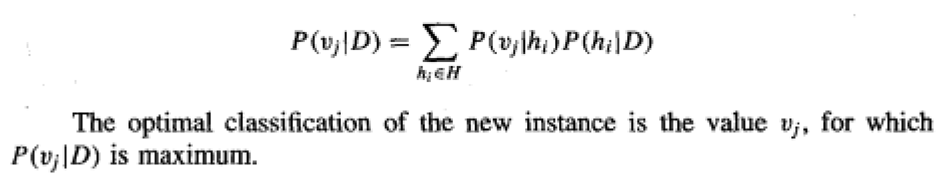
Assuming we use the codes C1 and CZ to represent the hypothesis and the data given the hypothesis, we can state the MDL principle as



**Minimum Description Length principle:** Choose $h_{MDL}$ where

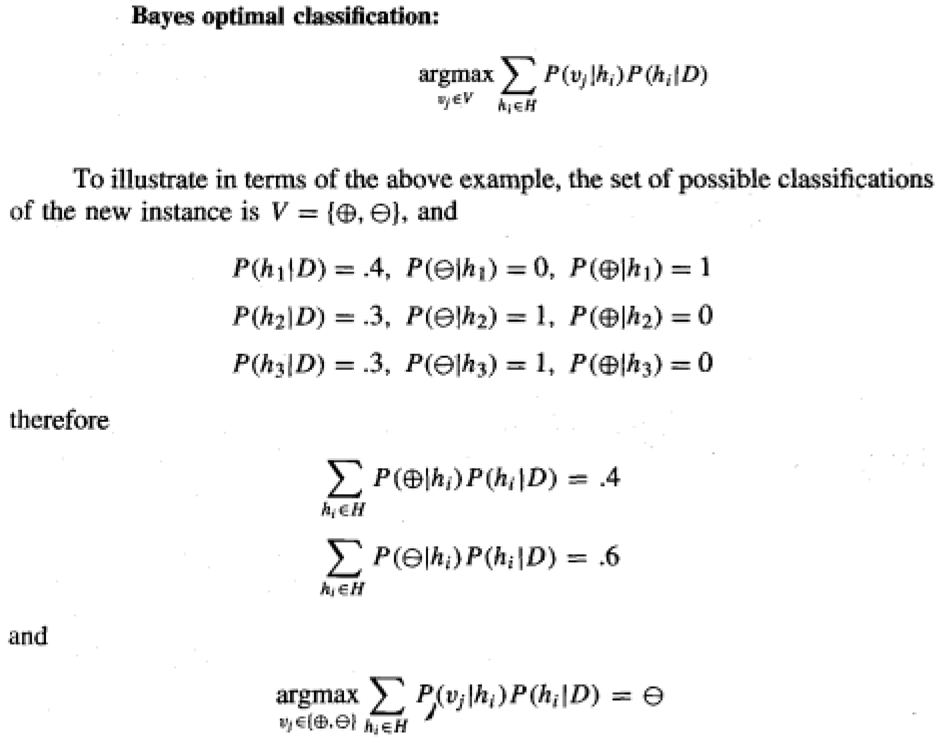
$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}}\, L_{C_1}(h) + L_{C_2}(D|h)$$

7. What is the Bayes optimal classifier?
The Bayes optimal classifier is a probabilistic classifier that assigns the most probable class label to a given instance based on Bayes' theorem. It calculates the posterior probability of each class label given the instance's features and selects the class label with the highest posterior probability.



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

The optimal classification of the new instance is the value $v_j$, for which $P(v_j|D)$ is maximum.



**Bayes optimal classification:**

$$\underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

To illustrate in terms of the above example, the set of possible classifications of the new instance is $V = \{\oplus, \ominus\}$, and

$$P(h_1|D) = .4,\ P(\ominus|h_1) = 0,\ P(\oplus|h_1) = 1$$
$$P(h_2|D) = .3,\ P(\ominus|h_2) = 1,\ P(\oplus|h_2) = 0$$
$$P(h_3|D) = .3,\ P(\ominus|h_3) = 1,\ P(\oplus|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(\oplus|h_i)P(h_i|D) = .4$$
$$\sum_{h_i \in H} P(\ominus|h_i)P(h_i|D) = .6$$

and

$$\underset{v_j \in \{\oplus, \ominus\}}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = \ominus$$

Max is 0.6. therefor the target concept is -

8. Define radial basis function
   A radial basis function is a real-valued function whose value depends only on the distance from a central point.

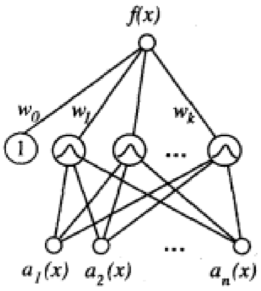


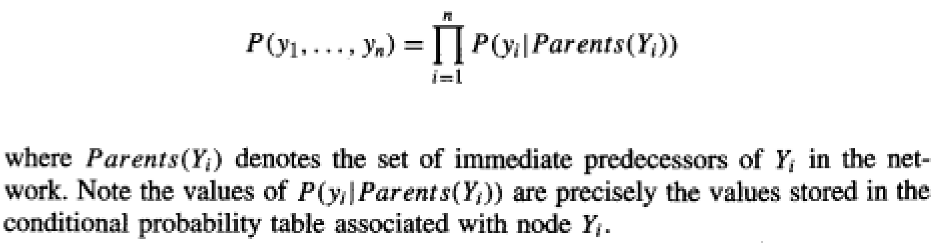
**FIGURE 8.2**
A radial basis function network. Each hidden unit produces an activation determined by a Gaussian function centered at some instance $x_u$. Therefore, its activation will be close to zero unless the input $x$ is near $x_u$. The output unit produces a linear combination of the hidden unit activations. Although the network shown here has just one output, multiple output units can also be included.

9. What is Case-Based Reasoning (CBR)
   In Case-Based Reasoning, a case refers to a specific instance or example from past experience that contains a problem description, solution, and context.
   It is a problem-solving approach that relies on past experiences stored as cases to solve new problems.

10. what is a bayesian belief network

A Bayesian belief network (Bayesian network for short) represents the joint probability distribution for a set of variables.

The joint probability for any desired assignment of values (y1, . . . , yn,) to the tuple of network variables (YI . . . Yn,) 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. Note the values of $P(y_i|Parents(Y_i))$ are precisely the values stored in the conditional probability table associated with node $Y_i$.

## Unit-4

1. What are Genetic Algorithms?

   Genetic Algorithms are optimization algorithms that simulate the process of natural selection to find solutions to complex problems. They work by evolving a population of candidate solutions using techniques such as selection, crossover, and mutation.

2. Describe an illustrative example of Genetic Algorithms.

   An illustrative example could involve optimizing the design of a car chassis using Genetic Algorithms. The algorithm would generate a population of potential designs, evaluate their performance (e.g., aerodynamics, stability), select the best-performing designs, and then use crossover and mutation operators to create new designs based on the selected ones.

3. What is hypothesis space search in the context of Genetic Algorithms?

   Hypothesis space search refers to the process of exploring and searching through the space of possible solutions (hypotheses) to find the best solution or set of solutions using Genetic Algorithms.

4. Define Reinforcement learning

   Reinforcement learning is a branch of machine learning concerned with how an agent can learn to make decisions in an environment in order to maximize some notion of cumulative reward. In reinforcement learning, an agent interacts with an environment by taking actions, receiving feedback in the form of rewards or penalties, and learning from this feedback to improve its decision-making process over time.

5. Give an example for mutation operator.

   In genetic algorithms, mutation is an operator used to introduce random changes in individuals (i.e., candidate solutions) in the population. Here's an example of a mutation operator applied to a binary string representation:

   Let's consider a binary string representation where each bit represents a gene in an individual's chromosome. Suppose we have an individual represented by the binary string "11010110". A mutation operator might randomly select one or more bits in the string and flip their values. For example, after applying mutation, the string might become "11110110" or "10010110".

6. What are the types of crossover operators

   Crossover operators are used to combine genetic information from two parent individuals to create new offspring individuals. There are several types of crossover operators, each with its own way of combining genetic material.

   ➢ Single-Point Crossover: In this type of crossover, a single crossover point is randomly selected along the length of the chromosomes of the parent individuals. The genetic material beyond this point is swapped between the parents to create two offspring.
   ➢ Two-Point Crossover: Similar to single-point crossover, but with two crossover points. Genetic material between the two points is swapped between the parents to create offspring.
   ➢ Uniform Crossover: In uniform crossover, each gene position in the offspring is randomly selected from one of the parent individuals with equal probability.
   ➢ Multi-Point Crossover: This is a generalization of single-point and two-point crossover, where multiple crossover points are randomly selected along the length of the chromosomes.

7. Define beam search

   Beam search is commonly used in various applications, including natural language processing, machine translation, speech recognition, and constraint satisfaction problems, where efficiently exploring a large search space is crucial for finding optimal or near-optimal solutions.

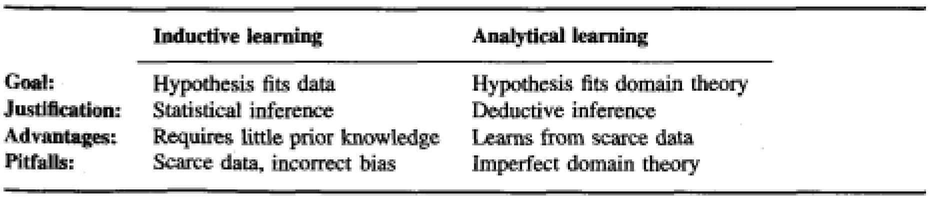
   It is a variant of the breadth-first search algorithm that explores a graph or state space by expanding only a fixed number of most promising nodes, called the "beam width," at each level of the search tree.

8. vornoi diagram

   The decision surface is a combination of convex polyhedra surrounding each of the training examples. For every training example, the polyhedron indicates the set of query points whose classification will be completely determined by that training example. Query points outside the polyhedron are closer to some other training example. This kind of diagram is often called the Voronoi diagram of the set of training examples.

## **Unit-5**

1. Differentiate between inductive learning and deductive learning.

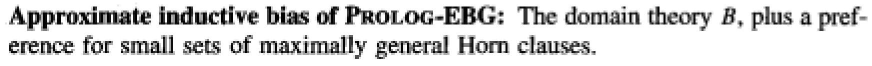


| | **Inductive learning** | **Analytical learning** |
|---|---|---|
| **Goal:** | Hypothesis fits data | Hypothesis fits domain theory |
| **Justification:** | Statistical inference | Deductive inference |
| **Advantages:** | Requires little prior knowledge | Learns from scarce data |
| **Pitfalls:** | Scarce data, incorrect bias | Imperfect domain theory |

2. Define explanation based learning
   Explanation-based learning (EBL) is a machine learning paradigm where new knowledge is acquired by generalizing from specific examples and explanations provided by an external source, often referred to as a teacher or domain expert.

3. List the steps in PROLOG EBG
   For each new positive training example that is not yet covered by a learned Horn clause, it forms a new Horn clause by:
   (1) explaining the new positive training example,
   (2) analyzing this explanation to determine an appropriate generalization, and
   (3) refining the current hypothesis by adding a new Horn clause rule to cover this positive example, as well as other similar instances.

4. What is the inductive bias of PROLOG-EBG



**Approximate inductive bias of PROLOG-EBG:** The domain theory *B*, plus a preference for small sets of maximally general Horn clauses.

5. List the methods for using prior knowledge to alter the search performed by purely inductive methods.
   - Use prior knowledge to derive an initial hypothesis from which to begin the search. **(KBANN)**
   - Use prior knowledge to alter the objective of the hypothesis space search. **( TANGENTPROP, EBNN)**
   - Use prior knowledge to alter the available search steps **(FOCL)**

6. Write two remarks on explanation based learning
   Unlike inductive methods, PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples.
   The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation.

7. What is augment search operator
   The term "augment search operator" typically refers to a method or technique used in search algorithms to enhance or augment the search process. It involves modifying the current state or solution in a way that helps explore the search space more effectively in pursuit of finding an optimal or satisfactory solution.
   Some common examples of augment search operators include
   - Mutation
   - Crossover