Scheme of Evaluation
Internal Assessment Test 2 – Dec.2021

Sub:	Artificial Intelligence & Machine Learning							**Code:**	18CS71
Date**:**	20/12/2021	Duration:	90mins	Max Marks:	50	**Sem:**	VII	**Branch:**	ISE

Note: Answer Any Five Questions

Question #	**Description**	**Marks Distribution**		**Max Marks**
1	• Finding the overall probabilities • Applying Naïve Bayes to calculate the conditional probabilities • Predicting the result	2 M 6 M 2 M	10 M	10 M
2	• Finding the euclidean distance • Updating the centroid • Applications of kmeans clustering	4 M 4 M 2 M	10 M	10 M
3	• Back propagation algorithm • Deriving the derivatives rule	4 M 6 M	10 M	10 M
4	• Finding the Euclidean distance • Choosing the neighbors based on k value and predicting the result for test data • Application of kNN	6M 2M 2M	10M	10 M
5a)	• Bayesian belief networks explanation • Example	2 M 4 M	6 M	10 M
5b)	• EM algorithm explanation	4 M	4 M	
6a)	• Defining perceptron • Discussing the Training rule	2 M 3 M	5 M	10 M
6b)	- Use of Bayesian learning - Features of Bayesian learning - Difficulties	1 M 2 M 2 M	5 M	

Internal Assessment Test 2 Solutions– Dec.2021

Sub:	Artificial Intelligence & Machine Learning							Code:	18CS71
Date:	20/12/2021	Duration:	90mins	Max Marks:	50	Sem:	VII	Branch:	ISE

Note: Answer Any Five Questions

1. Classify the test data {**Red, SUV, Domestic**} using NAÏVE Bayes classifier for the dataset shown below.

Color	Type	Origin	Stolen
Red	Sports	Domestic	Yes
Red	Sports	Domestic	No
Red	Sports	Domestic	Yes
Yellow	Sports	Domestic	No
Yellow	Sports	Imported	Yes
Yellow	SUV	Imported	No
Yellow	SUV	Imported	Yes
Yellow	SUV	Domestic	No
Red	SUV	Imported	No
Red	Sports	Imported	Yes

Solution:

The probabilities of the different target values can easily be estimated based on their frequencies over the 10 training examples

- P(Yes) = 5/10 = 0.5
- P(No) = 5/10 = 0.5

For new data {Red, SUV, Domestic} we need to classify the result

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

= argmax P (Vj) * P (Red|Vj) * P (SUV|Vj) * P (Domestic|Vj), **Vj ={Yes, No}**

Now we need to find the conditional probabilities for the test data as mentioned below

- P(Red|Vj=Yes) = 3/5 = 0.6
- P(SUV|Vj=Yes) = 1/5 = 0.2
- P(Domestic|Vj=Yes) = 2/5 = 0.4

Now we need to find the conditional probabilities for the test data w.r.t 'No'as mentioned below

- P(Red|Vj=No) = 2/5 = 0.4
- P(SUV|Vj=No) = 3/5 = 0.6
- P(Domestic|Vj=No) = 3/5 = 0.6

Finally for the test data we have the formula as below

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

$V_{NB}\{Yes\}$ = P (Yes)*P (Red|Yes)*P(SUV|Yes)*P(Domestic|Yes) = 0.5*0.6*0.2*0.4 = 0.024

$V_{NB}\{No\}$ = P (No)* P(Red|No)*P(SUV|No)*P(Domestic|No) = 0.5*0.4*0.6*0.6 = 0.072

So for new data {Red, SUV, Domestic} the result is No

---

2. Consider the following iris dataset. Using the k-Means Clustering approach, classify the below examples into k clusters by taking k value as 2.Also mention the applications of k-Means clustering approach.(Can consider 2 initial values for the first step as No.3 and No.6)

No	sepal.length	sepal.width
1	5.1	3.5
2	4.9	3
3	7	3.2
4	6.4	3.2
5	6.3	3.3
6	5.8	2.7

Solution:

Since k=2, initial values are 3 and 6

Initial centroid	X	Y
c1	7	3.2
c2	5.8	2.7

2)Calculate the euclidean distance of the given equation

Distance(X,Y)(a,b) = Sqrt(X-a)2+(X-b)2

Initial centroid	X	Y	Distance from cluster1	Distance from cluster2
1	5.1	3.5	sqrt(7-5.1)2+(3.2-3.5)2 = 1.92	sqrt(5.8-5.1)2+(2.7-3.5)2 = 1.02
2	4.9	3	sqrt(7-4.9)2+(3.2-3)2 = 2.10	sqrt(5.8-4.9)2+(2.7-3)2 = 0.94
3	7	3.2	sqrt(7-7)2+(3.2-3.2)2 = 0	sqrt(5.8-7)2+(2.7-3.2)2 = 1.30
4	6.4	3.2	sqrt(7-6.4)2+(3.2-3.2)2 = 0.6	sqrt(5.8-6.4)2+(2.7-3.2)2 = 0.94
5	6.3	3.3	sqrt(7-6.3)2+(3.2-3.3)2 = 0.70	sqrt(5.8-6.3)2+(2.7-3.3)2 = 0.81
6	5.8	2.7	sqrt(7-5.8)2+(3.2-2.7)2 = 1.30	sqrt(5.8-5.8)2+(2.7-2.7)2 = 0

1st iteration

	C1	C2	assigned to
1	1.92	1.02	c2
2	2.1	0.94	c2
3	0	1.3	c1
4	0.6	0.94	c1
5	0.7	0.81	c1
6	1.3	0	c2

values 3,4,5 belongs to c1 and 1,2,6 belongs to c2

Now we need to calculate the new centroids

c1= (7+6.4+6.3)/3 = 6.56, (3.2+3.2+3.3)/3 = 3.23 = (6.6,3.2)

c2=(5.1+4.9+5.8)/3 = 5.26, (3.5+3+2.7)/3 =3.06 = (5.3,3.1)

2nd iteration

Find the distance w.r.t the updated centroid 6,6, 3.2 and 5.3,3.1

Initial centroid	X	Y	Distance from cluster1	Distance from cluster2
1	7	3.2	sqrt(6.6-5.1)2+(3.2-3.5)2 = 1.52	sqrt(5.3-5.1)2+(3.1-3.5)2 = 0.44
2	5.8	2.7	sqrt(6.6-4.9)2+(3.2-3)2 = 1.71	sqrt(5.3-4.9)2+(3.1-3)2 = 0.41
3	7	3.2	sqrt(6.6-7)2+(3.2-3.2)2 = 0.4	sqrt(5.3-7)2+(3.1-3.2)2 = 1.70
4	6.4	3.2	sqrt(6.6-6.4)2+(3.2-3.2)2 = 0.2	sqrt(5.3-6.4)2+(3.1-3.2)2 = 1.10
5	6.3	3.3	sqrt(6.6-6.3)2+(3.2-3.3)2 = 0.31	sqrt(5.3-6.3)2+(3.1-3.3)2 = 1.07
6	5.8	2.7	sqrt(6.6-5.8)2+(3.2-2.7)2 = 0.94	sqrt(5.3-5.8)2+(3.1-2.7)2 = 0.78

	C1	C2	assigned to
1	1.52	0.44	c2
2	1.71	0.41	c2
3	0.4	1.7	c1
4	0.2	1.1	c1
5	0.31	1.07	c1
6	0.94	0.78	c2

Values 3,4,5 belongs to c1 and 1,2,6 belongs to c2

Since there is no change in the previous cluster values, we will stop here and the final clusters are as mentioned below.

	C1	C2	assigned to
1	1.52	0.44	c2
2	1.71	0.41	c2
6	0.94	0.78	c2
3	0.4	1.7	c1
4	0.2	1.1	c1
5	0.31	1.07	c1

---

3. Write an algorithm for back propagation which uses stochastic gradient descent method. Derive the back propagation rule considering the output layer and training rule for output unit weights.

Solution:

BACKPROPAGATION (*training_example*, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $(\vec{x}, \vec{t})$, *where* $(\vec{x})$ *is the vector of network input values,* $(\vec{t})$ *and is the vector of target network output values.*
η *is the learning rate (e.g., .05).* n_i *is the number of network inputs,* n_{hidden} *the number of units in the hidden layer, and* n_{out} *the number of output units.*
The input from unit i into unit j is denoted x_{ji}, *and the weight from unit i to unit j is denoted* w_{ji}

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do

  - For each $(\vec{x}, \vec{t})$, in training examples, Do

    Propagate the input forward through the network:
    1. Input the instance $\vec{x}$, to the network and compute the output o_u of every unit u in the network.

Propagate the errors backward through the network:

2. For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

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

Where

$$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

Derivation of the BACKPROPAGATION Rule

Deriving the stochastic gradient descent rule: Stochastic gradient descent involves iterating through the training examples one at a time, for each training example d descending the gradient of the error E_d with respect to this single example

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \quad \text{.....equ. (1)}$$

where, E_d is the error on training example d, summed over all output units in the network

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in output} (t_k - o_k)^2$$

For each training example d every weight w_{ji} is updated by adding to it Δw_{ji}

Here outputs is the set of output units in the network, t_k is the target value of unit k for training example d, and o_k is the output of unit k given training example d.

The derivation of the stochastic gradient descent rule is conceptually straightforward, but requires keeping track of a number of subscripts and variables

- x_{ji} = the i^{th} input to unit j
- w_{ji} = the weight associated with the i^{th} input to unit j
- $net_j = \Sigma_i w_{ji}x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output computed by unit j
- t_j = the target output for unit j
- σ = the sigmoid function
- outputs = the set of units in the final layer of the network
- Downstream(j) = the set of units whose immediate inputs include the output of unit j

---

Use chain rule to write

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

$$= \frac{\partial E_d}{\partial net_j} x_{ji} \qquad \text{.....equ(2)}$$

Derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

Consider two cases: The case where unit j is an output unit for the network, and the case where j is an internal unit (hidden unit).

Case 1: Training Rule for Output Unit Weights.

w_{ji} can influence the rest of the network only through net_j , net_j can influence the network only through o_j. Therefore, we can invoke the chain rule again to write

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j} \qquad \text{.....equ(3)}$$

To begin, consider just the first term in Equation (3)

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

The derivatives $\frac{\partial}{\partial o_j}(t_k - o_k)^2$ will be zero for all output units k except when $k = j$. We therefore drop the summation over output units and simply set $k = j$.

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2$$

$$= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j) \qquad \text{.....equ (4)}$$

Next consider the second term in Equation (3). Since $o_j = \sigma(net_j)$, the derivative $\frac{\partial o_j}{\partial net_j}$ is just the derivative of the sigmoid function, which we have already noted is equal to $\sigma(net_j)(1 - \sigma(net_j))$. Therefore,

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j}$$
$$= o_j(1 - o_j) \quad \text{.....equ(5)}$$

Substituting expressions (4) and (5) into (3), we obtain

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j) \quad \text{.....equ(6)}$$

and combining this with Equations (1) and (2), we have the stochastic gradient descent rule for output units

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta\, (t_j - o_j)\, o_j(1 - o_j) x_{ji} \quad \text{.....equ(7)}$$

---

4. Consider the training examples shown in the following table. The table shows a training set for a problem of predicting whether a loan applicant will repay his/her loan obligation or defaulting on his/her loan.

Tid	House Owner	Marital Status	Annual income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Using the kNN approach, predict the class label for this test example
House Owner=No, Marital Status=Married, Annual income=120K.Assume that k=3, single=0, married=1 and divorced=2

Solution:

- Find the Euclidean distance between all the values
  Sqrt((x2-x1)2+(y2-y1)2)

- Convert the categorical values into numericals. i.e assume Yes=1, No=0

Tid	Distance
1	Sqrt(0-1)2+(1-0)2+(120-125)2 = 5.196
2	Sqrt(0-0)2+(1-1)2+(120-100)2 = 20
3	Sqrt(0-0)2+(1-0)2+(120-70)2 = 50
4	Sqrt(0-1)2+(1-1)2+(120-120)2 = 1
5	Sqrt(0-0)2+(1-2)2+(120-95)2 = 25
6	Sqrt(0-0)2+(1-1)2+(120-60)2 = 60

7	Sqrt(0-1)2+(1-2)2+(120-220)2 =100
8	Sqrt(0-0)2+(1-0)2+(120-85)2 = 35
9	Sqrt(0-0)2+(1-1)2+(120-75)2 =45
10	Sqrt(0-0)2+(1-0)2+(120-90)2 = 30

Since k=3, we will choose 3values based on minimum distance
i.e.

Tid	Distance	Defaulted Borrower
1	Sqrt(0-1)2+(1-0)2+(120-125)2 = 5.196	No
2	Sqrt(0-0)2+(1-1)2+(120-100)2 = 20	No
4	Sqrt(0-1)2+(1-1)2+(120-120)2 = 1	No

For all these 3 neighbors the default class is **No.** Hence for the test data
House Owner=No, Marital Status=Married, Annual income=120K – No

Applications of kNN:

- Text mining
- Agriculture
- Finance
- Medical
- Facial recognition
- Recommendation systems (Amazon, Hulu, Netflix, etc)

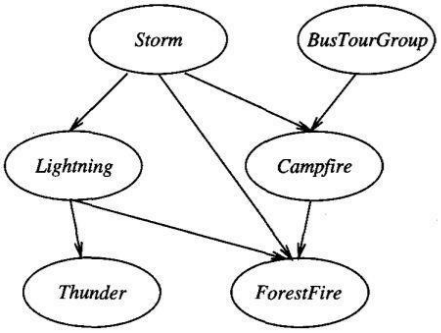
5.a) Explain Bayesian Belief Networks and conditional independence with example
Solution:

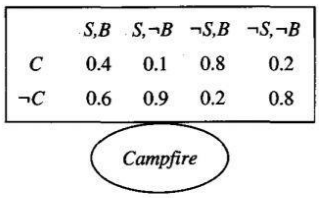
BAYESIAN BELIEF NETWORKS

- The naive Bayes classifier makes significant use of the assumption that the values of the attributes $a_1 \ldots a_n$ are conditionally independent given the target value v.
- This assumption dramatically reduces the complexity of learning the target function

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

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





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*

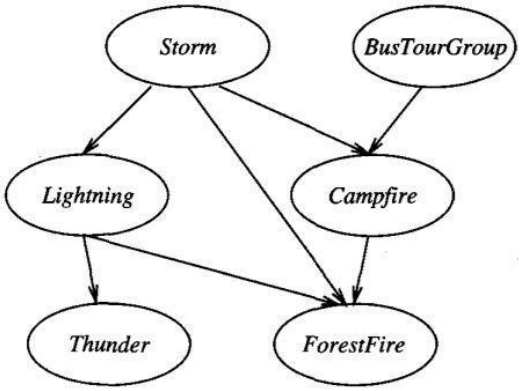
- BN represented by a directed acyclic graph, together with sets of local conditional probabilities
- Each variable in the joint space is represented by a node in the Bayesian network
- The network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the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

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***.



	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

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

5.b) Explain the EM Algorithm in detail.

Solution:

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.

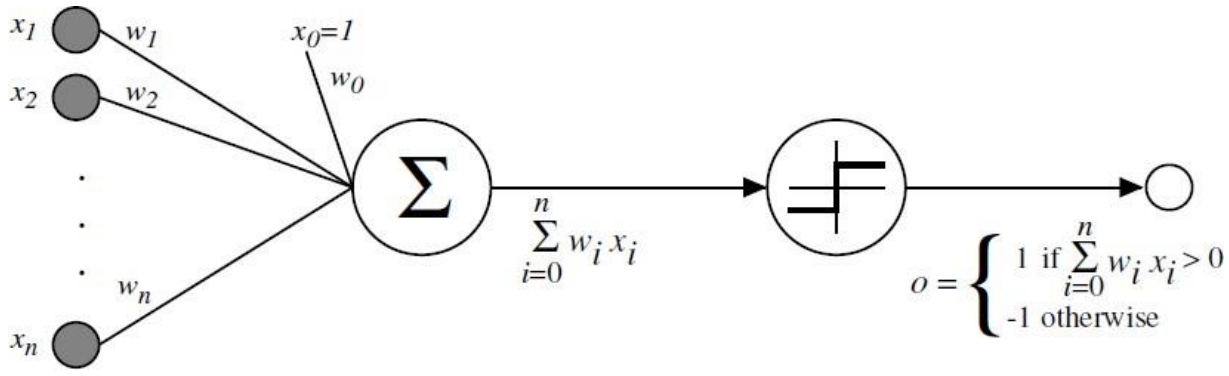
In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$.
maximum likelihood hypothesis in this case is given by

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

6.a) Define perceptron and discuss its training rule.

Solution:

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



A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.
Given inputs ***x*** through ***x,*** the output $\mathbf{O}(x_1, \ldots, x_n)$ computed by the perceptron is

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

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 an example.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight w_i associated with input xi 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
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 each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases

---

6.b) How Bayesian learning is useful in a machine learning context? Explain the features of Bayesian learning methods and difficulties?

Solution:

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 learning problems

2. The second reason is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

Features of Bayesian Learning Methods

1. 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 single example

2. Prior knowledge can be combined with observed data to determine the final probability 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 possible hypothesis.

3. Bayesian methods can accommodate hypotheses that make probabilistic predictions

4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
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 advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
2. A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantly reduced.