$$
\begin{aligned}
P(X_R = 1) &= 0.2 \\
P(X_S = 1) &= 0.1 \\
P(X_A = 1 | X_R = 1) &= 1 \\
P(X_A = 1 | X_R = 0) &= 0.2 \\
P(X_T = 1 | X_R = 1, X_S = 0) &= 1 \\
P(X_T = 1 | X_R = 1, X_S = 1) &= 1 \\
P(X_T = 1 | X_R = 0, X_S = 1) &= 0.9 \\
P(X_T = 1 | X_R = 0, X_S = 0) &= 0
\end{aligned}
$$

Given that Anisa's grass is wet, what is the probability that the sprinkler was on overnight? That is, what is the value of $P(X_S = 1 | X_A = 1)$? The calculations are shown below.

$$
\begin{aligned}
P(X_S = 1 | X_A = 1) &= \frac{P(X_S = 1, X_A = 1)}{P(X_A = 1)} \\
&= \frac{\sum_{X_J, X_R} P(X_S = 1, X_A = 1)}{\sum_{X_J, X_R, X_S} P(X_A = 1)} \\
&= \frac{\sum_{X_J, X_R} P(X_J | X_R) * P(X_A = 1 | X_R, X_S = 1) * P(X_R) * P(X_S = 1)}{\sum_{X_J, X_R, X_S} P(X_J | X_R) * P(X_A = 1 | X_R, X_S) * P(X_R) * P(X_S)} \\
&= \frac{\sum_{X_R} P(X_A = 1 | X_R, X_S = 1) * P(X_R) * P(X_S = 1)}{\sum_{X_R, X_S} P(X_A = 1 | X_R, X_S) * P(X_R) * P(X_S)} \\
&= \frac{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1}{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1 + 0 * 0.8 * 0.9 + 1 * 0.2 * 0.9} \\
&= 0.3382
\end{aligned}
$$

$$
= \frac{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1}{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1 + 0 * 0.8 * 0.9 + 1 * 0.2 * 0.9}
$$

$$
= 0.3382
$$

18.3 Hidden Markov Models

A class of models known as Hidden Markov Models (HMM) assumes that there are nodes in the model that are not observable. What is accessible is a set of observations, which are dependent upon the unobservable

state of the system. HMMs are Markov models because one assumes that the state that the system is in is not dependent on the history of the past states, and a given state has well defined successor states, one of which will be next state with a given probability. HHMs are popular where temporal patterns are involved, for example in Speech Recognition.

In many applications, there is a natural order in the data that imposes label dependency between points. The label of a point depends on points prior to it as specified by the label dependency. For instance, (a) part of speech (POS) tag of a word in English statement depends on the tag of the previous word; (b) the address segmentation problem that automatically tags locality, city, and state information in the address exploits relative placements of these entities in the address. Thus, there are two sequences per example: *observation sequence* and *label sequence*. The observation sequence refers to the sequence that we observe as the name suggests, while the label sequence refers to the sequence of states that are hidden. Each hidden state emits a symbol, thus forming an observation sequence. Let X be the sequence of n observations $<x_1, x_2, -, x_n>$ and let Y be the corresponding label sequence $Y = <y_1, y_2,\ldots, y_n>$, which is often unknown or hidden from the user. Note that there is an order information associated with the sequence $x_1 < x_2 < \ldots < x_n$, where < denotes "occurs before" relationship. The i^{th} label y_i outputs i^{th}observation x_i.

The Wikipedia defines a Hidden Markov Model as follows: An HMM is a statistical Markov model that is used to model a system that is assumed to a Markov process with hidden states. HMM models a sequence of labels, which is hidden and is not directly observable. For example,

1. In POS tagger, HMM models relationship between different parts of speech and is used in tagging a new sentence with appropriate POS tags. These tags are inferred from the observation sequence.
2. In an address segmentation problem, HMM models relationship between different portions of an address. Note that the portions of address are not directly available and needs to be inferred from the address based on prior experience.

We are interested in learning a joint model between observation and label sequences. The HMM assumes that the observations are independent of each other and they depend only on the corresponding label, while labels are dependent on each other, as per the HMM specification. For example, i^{th} observation x_i is independent of all other data points and is dependent only on i^{th} label of the point. That is y_i, which in turn depends on the label of the immediate predecessor y_{i-1}. In the simplest case, the label of a point depends only on the label of its immediate predecessor. Such models are called first order Markov model.

Under these models, $P(y_i|y_1, y_2, \ldots, y_{i-1}) = P(y_i| y_{i-1})$ and the joint probability of X and Y is given by

$$P(X,Y) = P(y_1) * P(x_1 \mid y_1) * \prod_{k=2}^{n} P(x_k \mid y_k) * P(y_k \mid y_{k-1}) \quad (18.14)$$

It is possible to construct higher order HMMs. Their discussion is out of scope of this book. Interested readers can refer to (Bishop, 2006). Higher order HMMs are used for finding genes in the genome sequence and a bunch of other biological sequence mining applications. Their details can be obtained from (Durbin et al., 1998),

An HMM has the following components.

1. A set of observation symbols Σ.
2. A set of labels $L = \tilde{Y} \cup \{B\} \cup \{E\}$. $\tilde{Y}$ is a set of labels and *B* and *E* are special labels used for denoting *begin* and *end* state of the markov chain. We append special labels *B* and *E* at the start and the end of label sequence *Y*. Thus *Y* becomes <*B*, y_1, y_2, …, y_n, *E*> with $\forall_{1 \le i \le n} y_i \in \tilde{Y}$.
3. Markov Chain is used for modelling of the label dependencies. It has |*L*| states. Each state corresponds to a label in $\tilde{Y}$ and it emits a symbol. Thus, each symbol in *X* comes from the set of symbols S. That is $x_i \in \Sigma$. The states *B* and *E* are dummy states and do not emit any symbol.
4. Transition probability matrix stores probability of transition from state y_{i-1} to y_i for every y_{i-1} and y_i from the set of labels. We denote it as $P(y_i \mid y_{i-1})$ where y_i, $y_{i-1} \in \tilde{Y}$. There are |*L*|*|*L*| entries in the transition matrix. Let $T_{|L|*|L|}$ be the transition matrix. Note that each row of the transition matrix sums to 1.
5. Emission matrix stores probability of emitting a symbol $\sigma \in \Sigma$ state $y_i \in \tilde{Y}$. We denote it as $P(s|y_i)$. It stores this probability for every $\sigma \in \Sigma$ in every $y_i \in \tilde{Y}$. Thus, the emission matrix has $|\tilde{Y}|$ rows and |Σ|columns. Let $O_{|\tilde{Y}|*|\Sigma|}$ be the emission matrix. Note that the each row sums to 1.

A *Hidden Markov Model* is completely specified with the above components. We will use *q* to denote the parameters of HMM, which includes *T* and *O*. The set of labels, observation symbols and the Markov chain are provided as part of a specification. We also use *θ* to denote complete HMM specification along with the parameters. The meaning of *θ* will be clear from the context and we will state it explicitly whenever required.

The following three problems are associated with HMM, out of which, two are inference problems and the remaining one is the learning

problem.

Learning Given a set of training examples, determine parameters θ of the given HMM.

Inference Given a *Hidden Markov Model*, θ, find the probability that a given observation sequence X is generated by it: $P(X; \theta)$[1].

Inference Given a *Hidden Markov Model*, θ, find the most probable sequence of states that generates a given sequence X. Formally,

$$Y = \underset{Y}{\operatorname{argmax}}\, P(X, Y\,;\, \theta)$$

An Example: *Modelling a Casino*

An HMM can be used to model a sequence of coin tosses in a casino. The casino maintains two types of coins which show different sides with different probabilities. For instance, one type of coin lands up heads with probability 0.6, while the other type of coin lands up heads with probability 0.9. In practice, we do not have access to the model that the casino uses to select a particular type of a coin for the next throw. It is reasonable to assume that selection of the next type of coin depends only on the current type and is independent of all previously selected types. We only observe a sequence of outcomes after throwing a coin (heads or tails).

Here, the sequence of outcome corresponds to an observation sequence (X), while the sequence of coin types corresponds to a hidden sequence (Y). For a throw of a coin, the probability of the outcome depends only on the coin that is chosen and is independent of the coins chosen earlier and the previous outcomes.

The HMM has the following components.

1. Set of observation symbols S = {H, T}, where H corresponds to heads and T corresponds to tails.
2. Set of labels $L = \tilde{Y} \cup \{B\} \cup \{E\}$ where $\tilde{Y} = \{T_1, T_2\}$.
3. The topology of the HMM that models coin tosses from the casino is shown in Figure 18.2.
4. Transition matrix T_{4*4}.
5. Emission matrix O_{2*2}

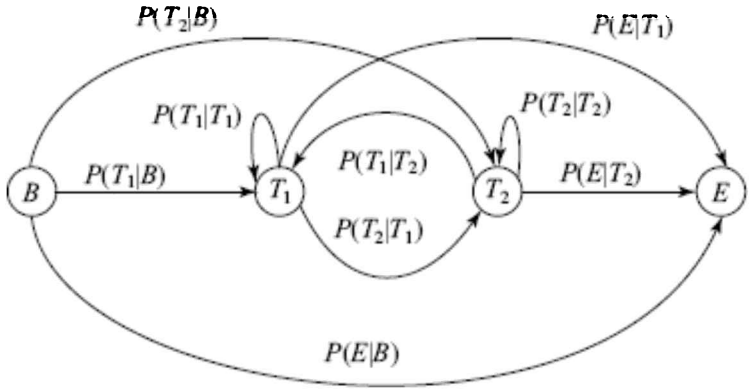

FIGURE 18.2 The transmission matrix of the HHM for modelling the Casino captures the dependencies shown here.

Application of HMMs in Biology

HMMs have been used extensively in Computational Biology to perform various tasks such as pairwise and multiple sequence alignments, motif finding, gene finding, RNA structure analysis, etc. The book on Biological Sequence Mining by Richard Durbin and coauthors provides detailed account of all these applications. Sequence alignment is a fundamental tools used by biologists for searching the existing databases of known genes or proteins. Such search enables identification of closely matching sequence or homolog of a new gene discovered in an experimental set-up. Pairwise alignment is used to align two sequences. The objective is to search for alignment that is optimal in terms of the cost defined by a certain cost matrix. *Pair-HMM* is used to align two sequences, where the actual alignment is obtained using *Viterbi* algorithm. *Profile-HMM* is used to perform alignment of multiple sequences at the same time.

18.3.1 HMM Training

The objective of HMM training is to learn parameters $\theta = (T,O)$ given a set of training examples and HMM specification in the form of Markov chain, sets of labels and symbols. The parameters can be determined through either maximum likelihood or Bayesian estimation techniques. The training data is specified as (i) an ordered pair of *observation symbol sequence* and the *corresponding label sequence*, or (ii) only the observation symbol sequences are given without specifying the corresponding label sequences. We will cover both these cases in this section.

Training Data Consisting of Observation and Label Sequences

The training data D consists of m ordered pairs of observation and label sequences $<(X_1, Y_1), (X_2, Y_2), \ldots, (X_m, Y_m)>$. In addition, the set of observation symbols S, the set of labels L and topology of HMM is provided as an input. With this set up, the objective is to determine transition probabilities in T and emission probabilities in O.

Each row in T can be modelled using multinomial distribution. Formally, $T_i \sim \text{Mult}(\varphi_i)$. There are $|L - 1|$ multinomial distributions and for each one $|L - 1|$ parameters need to be estimated. Since the state corresponding to label E does not have any connections to other states, we need not determine the corresponding parameters. Using maximum likelihood estimation (MLE), $T(i, j)$ entry, corresponding to transition probability $P(y_j \mid y_i)$, can be estimated as follows.

$$T(i, j) = \phi_{i,j} = P(y_j|y_i) = \frac{\sum_{i=1}^{m} \#(y_i - y_j \text{ transitions in } X_i \in D)}{\sum_{i=1}^{m} \sum_{y \in L} \#(y_i - y_j \text{ transitions in } X_i \in D)} \tag{18.15}$$

In case of limited amount of training data, we do not encounter transitions between certain states and hence MLE of respective φ_{ij}, is zero, which drives probability of certain sequences being generated from the HMM to zero. The problem can be avoided by adding a small fake or pseudocount δ in numerator and denominator of Eq. (18.15). Addition of pseudocount is nothing but Bayesian estimation using conjugate priors (Dirichlet distribution in this case).

$$T(i, j) = \phi_{i,j} = P(y_i|y_i) = \frac{\sum_{i=1}^{m} \#(y_i - y_j \text{ transitions in } X_i \in D) + \delta}{\sum_{i=1}^{m} \sum_{y \in L} \#(y_i - y_j \text{ transitions in } X_i \in D) + \delta} \tag{18.16}$$

Similarly each row in O can be modelled appropriately depending on the nature of X. The X can be either discrete or continuous as described earlier. The discrete observations can be modelled using appropriate discrete distributions such as binomial, multinomial, Poisson, etc. On the other hand, the continuous or real observations can be modelled using distributions like Gaussian. The parameter estimation can be performed via MLE or Bayesian estimation techniques. As an example, we explain parameter estimation when observations are discrete in nature. Let each observation s come from a set of finite symbols Σ. Each label $y \in \tilde{Y}$ emits these symbols according to a fixed distribution with unknown parameters. Let us assume that

$$O_{y,\sigma} = P(\sigma \in \Sigma \mid y \in L) \neg \text{Mult}(\phi_y)$$

The estimates for O_s can be obtained using MLE as follows.

$$O_{y,\sigma} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{\text{length}(X_i)}\mathbb{I}[X_{i,j}=\sigma, Y_{i,j}=y]}{\sum_{i=1}^{m}\sum_{j=1}^{\text{length}(X_i)}\sum_{\forall\sigma\in\Sigma}\mathbb{I}[X_{i,j}=\sigma, Y_{i,j}=y]} \tag{18.17}$$

Here, I(.) is an indicator function that returns 1 if that argument is true. The *length* (X_i) returns number of observations in sequence X_i. The Bayesian estimates are obtained by using appropriate prior distribution. It is equivalent to adding a small pseudocount *d* to the actual counts.

$$O_{y,\sigma} = \frac{\left[\sum_{i=1}^{m}\sum_{j=1}^{\text{lenght}(X_i)}\mathbb{I}[X_{i,j}=\sigma, Y_{i,j}=y]\right]+\delta}{\sum_{\forall\sigma\in\Sigma}\left[\sum_{i=1}^{m}\sum_{j=1}^{\text{length}(X_i)}\mathbb{I}[X_{i,j}=\sigma, Y_{i,j}=y]\right]+\delta} \tag{18.18}$$

Training Data Consisting only Observation Sequences

Unlike the first case, here the label sequences are not available with observation sequences. Therefore we cannot directly apply the procedure described in the previous section. This is the case of missing information, where label sequences are not available. In such cases, the *Expectation Maximization* (EM) technique is used to counter the missing information problem. EM is a powerful technique that operates in two steps repetitively until convergence.

1. *E-step*: Assigns values to the missing variable.
2. *M-step*: Maximizes the objective function with respect to the assignments made in *E*-step.

HMM training with EM algorithm can be performed via two different random initializations of either
label sequence or parameters such as transition and emission matrix. These techniques are known as *Viterbi* and *Baum-Welsch* training respectively. Their algorithms are given in Figures 18.3 and 18.4 respectively.

```
HMM-Viterbi(HMM: H(θ), Observations: X)
1  for each o_i ∈ X
2        label_i ← Randomly choose a label from L
3  repeat
4         Estimate θ = (T,O) using Bayesian or Maximum Likelihood
5         Compute most likely labels for the observations using θ
6  until convergence in label
7  return θ
```

FIGURE 18.3 The Algorithm *HMM-Viterbi* is used for training of the HMM when the label sequence is not provided as part of the training data. Here, we show an iterative procedure to estimate parameters of *HMM*, θ, by randomly initializing hidden sequence or label sequence. The subsequent iterations ensure that the process will converge to a local minimum.

Note that both Viterbi and Baum-Welsch algorithm converge to locally optimal estimates, based on the initialization. This is due to convergence properties of EM in general. Hence, these algorithms should be run with

different initializations and selecting the parameters that lead to the best log-likelihood.

```
HMM-Baum-Welch(HMM: H(θ ), Observations: X)
1  Randomly initialize the transition matrix T
2  Randomly initialize the emission matrix E
3  repeat t ∈ C
4        Compute most likely labels for the observations using θ
5        Estimate θ = (T,O) using Bayesian or Maximum Likelihood
6
7  until convergence in θ
8  return θ
```

FIGURE 18.4 The algorithm HMM-Baum-Welch is also used for training of the HMM when label sequence is not provided as part of the training data. Here, we show an iterative procedure to estimate parameters of HMM, θ, by randomly initializing model parameters. The subsequent iterations ensure that the process will converge to a local minimum.

18.3.2 Finding $P(X; \theta)$

Find probability of a given sequence, $X = <x_1, x_2, -, x_n>$ where each $x_i \in \Sigma$, being generated from a given HMM θ. The sequence X can be generated by traversing various paths in the Markov chain. Each path generates a sequence of labels with the corresponding probability of generating X. The probability is given by

$$P(X,Y; \theta) = \prod_{k=1}^{n} P(y_k \mid y_{k-1}; \theta) * P(x_i \mid y_k; \theta) \tag{18.19}$$

Note that $y_0 = B$ and $y_{n+1} = E$ and these states do not emit any symbol. The quantity of interest can then be obtained by summing over all such probabilities. Let S be the set of all possible label sequences that generate X.

$$P(X; \theta) = \Sigma_{Y \in S} P(X,Y; \theta) \tag{18.20}$$

$$= \Sigma_{Y \in S} \prod_{k=1}^{n} P(y_k \mid y_{k-1}; \theta) * P(x_i \mid y_k; \theta) \tag{18.21}$$

The key computational step here involves enumeration of all paths in the Markov chain. Clearly, such an operation is computationally inefficient and highly expensive. It can be efficiently carried out by storing intermediate results in order to avoid repetitive computations. The intermediate result at position i in the sequence can be calculated as follows.

$$P(y_i, x_1, x_2, \ldots, x_i) = \Sigma\, P(y_i, x_1, x_2, \ldots, x_i) * y_{i-1} \tag{18.22}$$

Using chain rule of probability,

$$= \sum_{y_{i-1}} P(x_i \mid x_1, x_2, \ldots, x_{i-1}, y_i, y_{i-1}) * P(y_i \mid x_1, x_2, \ldots, x_{i-1}, y_{i-1}) * P(x_1, x_2, \ldots, x_{i-1}, y_{i-1}) \quad (18.23)$$

Note that the symbol x_i only depends on the label y_i and hence,

$$P(x_i \mid x_1, x_2, \ldots, x_{i-1}, y_i, y_{i-1}) = P(x_i \mid y_i)$$

Also, the label y_i depends only on the label of previous state y_{i-1} and hence,

$$P(y_i \mid x_1, x_2, \ldots, x_{i-1}, y_{i-1}) = P(y_i \mid y_{i-1})$$

In addition, we will define a forward variable $\alpha_{i-1}(y_{i-1})$ that gives a total probability of observed sequence, up to i–1 symbol ending in the state $y_{i-1} \in Y$. Let

$$\alpha_{i-1}(y_{i-1}) = P(x_1, x_2, \ldots, x_{i-1}, y_{i-1})$$

Equation (18.23) becomes

$$P(y_i, x_1, x_2, \ldots, x_i) = \sum_{y_{i-1}} P(x_i \mid y_i) * P(y_i \mid y_{i-1}) * \alpha_{i-1}(y_{i-1}) \quad (18.24)$$

$$= P(x_i \mid y_i) * \sum_{y_{i-1}} P(y_i \mid y_{i-1}) * \alpha_{i-1}(y_{i-1}) \quad (18.25)$$

$$= \alpha_i(y_i) \quad (18.26)$$

At $y_{n+1} = E$, we can simply compute $P(X; \theta)$ as follows.

$$P(X; \theta) = \sum_{y_n} P(E \mid y_n) * \alpha_n(y_n) \quad (18.27)$$

We use $\alpha_0(B) = 1$.

```
ForwardAlgorithm(HMM: H(θ), Observations: X, Parameters: θ)
1  α0(B) ← 1
2  repeat
3  for i ←1 to n
4        for each yi∈Y
5              αi(yi) ← P(xi|yi) * Σ_{yi-1} P(yi|yi-1)*αi-1(yi-1)
6  return P(X) ← Σ_{yn} P(E|yn)*αn(yn)
```

FIGURE 18.5 The *ForwardAlgorithm* is used to find the probability of a sequence X being generated from a given HMM.

18.3.3 Finding Most Probable Label Sequence

The second inference problem is concerned about finding the most probable sequence of labels Y, corresponding to a given observation sequence X, for a given HMM with parameters θ. Formally,

$$Y = \underset{Y}{\operatorname{argmax}}\ P(X, Y; \theta) \tag{18.28}$$

The most probable label sequence can be obtained by enumerating all possible label sequences generating X and then calculating joint probability between observation symbol and each label sequence. As described in the previous section, such an approach is computationally expensive. An efficient algorithm can be designed for this task using dynamic programming technique, similar to forward algorithm (Figure 18.5), by storing results of intermediate computations. This algorithm (Figure 18.6) is known as *Viterbi algorithm* or *max sum algorithm*.

The most likely label sequence for observation sequence till position i can be computed as follows.

$$P(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_n) = P(x_i|y_i) \max_{y_{i-1} \in y} [P(y_i|y_{i-1}) * P(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1})] \tag{18.29}$$

Let us define viterbi variable γ as follows:

$$\gamma(y_{i-1}) = P(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}) \tag{18.30}$$

$$\gamma(B) = 1 \tag{18.31}$$

Now Eq. (18.32) becomes

$$P(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_i) = P(x_i|y_i) * \max_{y_{i-1} \in y} [P(y_i|y_{i-1}) * \gamma(y_{i-1})] \tag{18.32}$$

In addition, we store pointer to y_{i-1} in $\zeta(i-1)$ for constructing the most probable label sequence.

$$\zeta(i-1) = \underset{y_{i-1} \in y}{\operatorname{argmax}}\ P(y_i|y_{i-1}) * \gamma(y_{i-1}) \tag{18.33}$$

$$\zeta(1) = B \tag{18.34}$$

Thus, the most probable label sequence for the observation sequence and the corresponding probability can be obtained at label E by the following recursions.

$$\gamma(E) = \max_{y_n \in y} P(E|y_n) * \gamma(y_n) \tag{18.35}$$

$$\zeta(n) = \underset{y_n \in y}{\text{argmax}}\ P(E|y_n) * \gamma(y_n) \tag{18.36}$$

The (E) is the probability of the most likely label sequence that generates X and $\zeta(n)$ gives the corresponding label sequence.

$$Y = \underset{Y}{\text{argmax}}\ P(X, Y; \theta) \tag{18.37}$$

$$Y = \zeta(n) \tag{18.38}$$

```
Viterbi(HMM: H(θ), Observations: X, Parameters: θ)
  1 γ0(B) ← 1
  2 repeat
  3 for i ← 1 to n
  4       for each yi∈Y
  5            γ(E) ← Max_{yn∈Y} P(E|yn)*γ(yn)
  6            ζ(E) ← Argmax_{yn∈Y} P(E|yn)*γ(yn)
  7 return  Y ← ζ(n)
```

FIGURE 18.6 The *Viterbi* algorithm for the HMM is used to obtain the most probable sequence of label for a given observation sequence.

18.4 Concept Learning

In Chapters 13 and 14, we discussed in detail the representation and reasoning with categories or concepts. A concept is a subset of the universe of discourse or the domain. We started by defining atomic concepts like *brother and parent* and used them to define other concepts like *uncle*.

In this section, we explore how concepts may be formed or learnt from instances of the concept. We assume that each individual in the domain is described by a set of attributes, and based on values of these attributes, certain concepts may be defined. For example, in the romantic fiction of the last century an "eligible bachelor" could be "described" as tall, dark[2] and handsome, and perhaps some other features like being sensitive, intelligent, having a sense of humour, being rich, etc. The task of concept learning is as follows.

Given that one has a set of training instances for which it is known whether they belong to a concept or not, to learn a general concept in terms of the attributes of the instances. The idea is that once such a concept has been learnt then one can use the knowledge to classify previously unseen instances correctly. This process of learning from examples is known as *Inductive Learning*.