Decision Trees, Support Vector Machines (SVMs), and Logistic Regression are some examples of discriminative supervised learning algorithms. At a deeper level, both *discriminative* and *generative* models can be seen to be equivalent.

We will describe two examples of generative supervised learning algorithms: (i) *Naive Bayes* (*NB*) classifier, and (ii) *Hidden Markov Model* (HMM). And the Decision Tree (DT) classifier, which is an example of discriminative supervised learning algorithm. Then, we look at *K*-means clustering which is an example of unsupervised learning. Finally, we will have a brief look at Reinforcement Learning.

18.1 Naïve Bayes Classifiers

A Naïve Bayes Classifier (NB) is very simple yet very powerful classifier. NB assumes that the features are conditionally independent, given the class labels. With this assumption, class-conditional joint distribution between features becomes

$$P(x \mid y) = P(x^{(1)}, x^{(2)}, \ldots, x^{(d)} \mid y) \tag{18.3}$$

$$= P(x^{(1)} \mid y) * P(x^{(2)} \mid y) * \ldots * P(x^{(d)} \mid y) \tag{18.4}$$

$$= \prod_{k=1}^{d} P(x^{(k)} \mid y) \tag{18.5}$$

In the Naïve Bayes classifier, the learning problem is to determine class-conditional densities or mass functions for each feature, depending on its nature and a class prior, given a set of training examples. Class-conditional density estimations are obtained for continuous features, while class conditional mass function estimations are obtained for discrete features. The number of parameters to be learnt depends on the parametric form of class conditional densities or mass functions. For example, if we use Gaussian distribution to model a continuous feature, we need to learn parameters (i) mean μ, and (ii) standard deviation σ. Thus, the key point in learning of NB classifier is to carefully model each feature with appropriate distributions. The wisdom for modelling can be obtained from domain experts or from the training data, if available in abundance. The learning problem is then to estimate relevant parameters from the training data using either maximum likelihood or Bayesian estimation techniques. For a learning problem with q classes, we need to learn parameters of class-conditional distribution of each feature for each of the classes. In addition, we need to estimate the prior probability of each class. It is sufficient to estimate priors for $q - 1$ classes, since the prior for the remaining class can be obtained using the fact that $\Sigma_q P(q) = 1$.

The inference problem using an NB classifier is to predict the class label for a new example. The probability of each class label $y_i \in Y$ for new example x_{new} is predicted using Bayes' theorem

$$P(y_i \mid x_{new}) = \frac{P(x_{new} \mid y_i) * P(y_i)}{P(x_{new})} \tag{18.6}$$

The label with the highest probability among the set of labels for x_{new} is predicted as its label. Formally,

$$y_i = \underset{y_i \in Y}{\operatorname{argmax}}\, P(y_i \mid x_{new}) \tag{18.7}$$

Since the term $P(x_{new})$ in Eq. (18.6) is constant for all the classes, Eq. (18.7) becomes

$$y_i = \underset{y_i \in Y}{\operatorname{argmax}}\, P(x_{new} \mid y_i) * P(y_i) \tag{18.8}$$

$$= \underset{y_i \in Y}{\operatorname{argmax}}\, P(y_i) * \prod_{k=1}^{d} P(x^{(k)} \mid y) \tag{18.9}$$

Example 1: Text Classification

NB classifier is used extensively in text categorization. The text document is represented using a bag of word representation that contains a set of words, without specifying their order of occurrence. A tiny training data set containing 4 documents from two classes, namely "Bio" and "CS", is shown in Table 18.1.

Table 18.1 A small collection of tiny documents

DocID	*Words*	*Label*
1	Algorithms, Tree, Graph	CS
2	Tree, Life, Gene, Algorithms	Bio
3	Graphs, NP, Algorithms, Tree	CS
4	Protein, Assay, Cell	Bio

The class priors can be estimated as follows:

$$P(y = \text{CS}) = \frac{\sum_i \mathbb{I}(y_i = \text{CS})}{\sum_j \sum_i \mathbb{I}(y_i = j)} = \frac{\#(\text{Docs with label "CS"})}{\#(\text{Total docs})} = 2/4 = 0.5$$

$$P(y = \text{Bio}) = \frac{\#(\text{Docs with label "Bio"})}{\#(\text{Total docs})} = 2/4 = 0.5$$

The class-conditional probability of each feature (word in this case) is computed as follows:

$$P(\text{"Algorithms"} \mid y = \text{"CS"}) = \frac{P(\text{"Algorithms"}, y = \text{"CS"})}{\sum_w P(w, y = \text{"CS"})}$$

$$= \frac{\#(\text{"Algorithms" in "CS" docs})}{\#(\text{Total words in "CS" docs})} = \frac{2}{7} \approx 0.3$$

Similarly,

$$P(\text{“Tree”}|y=\text{“CS”}) = \frac{2}{7},\ P(\text{“Graph”}|y=\text{“CS”}) = \frac{2}{7},\ P(\text{“NP”}|y=\text{“CS”}) = \frac{1}{7}$$

$$P(\text{“Algorithms”}|y=\text{“Bio”}) = P(\text{“Tree”}|y=\text{“Bio”}) = P(\text{“Gene”}|y=\text{“Bio”}) = P(\text{“Assay”}|y=\text{“Bio”}) = \frac{1}{7}$$

$$P(\text{“Cell”}|y=\text{“Bio”}) = P(\text{“Life”}|y=\text{“Bio”}) = P(\text{“Protein”}|y=\text{“Bio”}) = \frac{1}{7}$$

Here, we assumed that *P*(word | label) is distributed according to multinomial distribution.

For a new document x_{new} containing the words (Algorithms, Tree), the class label is assigned based on the highest probability of belonging to the class. The probability can be calculated using the estimated prior and class-conditional probabilities of each word.

$$\begin{aligned} P(y=\text{“CS”}|x_{new}) &\propto P(y=\text{“CS”}\prod_w P(w|y=\text{“CS”}) \\ &\propto P(y=\text{“CS”}) * P(\text{“Algorithms”}|y=\text{“CS”}) * P(\text{“Tree”}|y=\text{“CS”}) \\ &\propto \frac{1}{2} * \frac{2}{7} * \frac{2}{7} = \frac{4}{98} \end{aligned}$$

Similarly,

$$P(y=\text{“Bio”}|x_{new}) \propto \frac{1}{98}.$$

Now

$$\begin{aligned} P(y=\text{“CS”}|x_{new}) &= \frac{P(y=\text{“CS”}) * \prod_w P(w|y=\text{“CS”})}{P(x_{new})} \\ &= \frac{P(y=\text{“CS”}) * \prod_w P(w|y=\text{“CS”})}{P(y=\text{“CS”}) * \prod_w P(w|y=\text{“CS”}) + P(y=\text{“Bio”}) * \prod_w P(w|y=\text{“Bio”})} \\ &= \frac{4/98}{4/98 = 1/98} = \frac{4}{5} = 0.8 \end{aligned}$$

and

$$\begin{aligned} P(y=\text{“Bio”}|x_{new}) &= \frac{P(y=\text{“Bio”}) * \prod_w P(w|y=\text{“Bio”})}{P(x_{new})} \\ &= \frac{P(y=\text{“Bio”}) * \prod_w P(w|y=\text{“Bio”})}{P(y=\text{“CS”}) * \prod_w P(w|y=\text{“CS”}) + P(y=\text{“Bio”}) * \prod_w P(w|y=\text{“Bio”})} \\ &= \frac{1/98}{4/98 + 1/98} = \frac{1}{5} = 0.2. \end{aligned}$$

Since $P(y=\text{”CS”}|\ x_{new}) > P(y=\text{”Bio”}|\ x_{new})$, we assign the label “CS” to the new document.

Example 2: Protein Sequence Classification

Proteins are important biomolecules that participate in a majority of cellular functions in the living organism. They are made up of amino acids. There exist twenty different amino acids. For a computer scientist, proteins can be thought of as strings constructed from language, with an alphabet set of twenty amino acids. Proteins exist as a three dimensional structure in its physical form and the instructions for the same is believed to be coded in the sequence information. The structure in turn provides valuable information about protein function. Proteins are classified into various groups by biologists, based on their structure and function. Protein sequence can be obtained quickly once the genome sequence is known. On the contrary, obtaining protein structure requires rigorous experimental set-up and hence a relatively small number of structures is available, resulting in a wide gap between number of proteins with known sequence and those with known structure. The sequence information provides vital clues for protein classification. In this problem, we will design a learning algorithm based on NB classifier to classify proteins into one of the k groups. Thus, this is an instance of a multiclass classification problem.

Let D be the database of known protein sequences. Let $f_1, f_2, \ldots, f_k$ be k classes or groups of proteins. Note that the proteins are not uniformly distributed across these k classes, i.e. some classes contain more proteins than other. They represent selection bias that the biologists have for certain classes of proteins. The first step is to represent each protein with a set of appropriate features. Let each protein $d \in D$ be represented with the following attributes or features:

1. Length of protein l_d.
2. Percentage of each amino acid in d. There is one such feature per amino acid and hence in all there are twenty such features.

Thus, each protein is represented with 21 features. All these features are continuous in nature, that is, their values are real numbers between 0 and 1.

Modelling Let $x_d^{(1)}, x_d^{(2)}, \ldots, x_d^{(20)}$ be the features corresponding to the frequencies of amino acids of protein d. These features sum up to 1. $\Sigma_{j=1}^{20} x_d^{(j)}=1$. Let $x_d^{(21)}$ be the feature corresponding to the length of the protein d. Clearly, the length feature is independent of all other features.

1. We model distribution of amino acids in a protein sequence, using a multinomial distribution.
2. We model length of proteins using a Gaussian distribution with parameters mean μ and standard deviation σ.

Learning Estimate class priors and parameters of Multinomial and Gaussian distribution using the training data D.

1. The parameter corresponding to feature $x(i)$ of multinomial distribution can be obtained for class f_j can be obtained as follows:

$$P(x^{(i)}|y = f_j) = \frac{\text{number of } x^{(i)} \text{ in sequences of } f_j}{\text{total number of observation sequences in } f_j} \quad (18.10)$$

The process needs to be repeated for all features across all *k* classes to obtain all the parameters of multinomial distribution.

2. The mean of protein length μ needs to be estimated for each protein class. For class f_j, it can be estimated as follows:

$$\mu_j = \frac{1}{\text{number of sequences of class } f_j} * \sum_{\text{all sequences of class } f_j} \text{length(sequence)} \quad (18.11)$$

The standard deviation \sigma can be obtained as follows:

$$\text{var}_j = \frac{1}{\text{number of sequences of class } f_j} * \sum_{\text{all sequences of class } f_j} (\mu_j - \text{length})^2 \quad (18.12)$$

$$\sigma_j = \sqrt{\text{var}_j} \quad (18.13)$$

Inference Given the estimated model parameters, label a new sequence with an appropriate class label.

18.2 Inference in Bayesian Networks

The Naive Bayes classifier assumes that the features are conditionally independent, given the class label. This assumption is often violated in practice. On the other hand, assuming that features are dependent on one another poses challenges in reliable estimation of parameters since enough data is not available. The Bayes' network enables specification of dependencies between variables, thus offering a middle path solution. The fundamental concepts of a Bayesian network were covered in Chapter 17. In this chapter, we will study inference in Bayesian network. The simplest form of inference is using variable elimination, which we will study in detail in this chapter. General purpose inference schemes include junction tree algorithm and message passing over factor graph constructed from the belief network.

Let us study a famous example, due to Judea Pearl, that is often cited in Bayesian network literature. One fine morning, Anisa noticed that the grass in her lawn was wet. She was puzzled: is it due to overnight rain or it is due to a sprinkler that remained switched on by mistake? She looked around and found that the grass in her neighbour's, Malala's, backyard was also wet. Our model of this situation is shown in Figure 18.1.