Classification Methods

"No amount of sophistication is going to allay the fact that all your knowledge is about the past and all your decisions are about the future." - Ian E. Wilson

---

5.0 Introduction

Classification is at the heart of statistical pattern recognition. Assigning entities or objects based on a set of unique *features* is the primary task of any classification effort. Classification models are called *supervisory learning methods*. In this regime, we begin with a set of *training* data. Training data is merely a database of records pertaining to an entity. Each record consists of a set of features and an associated class label. If the records pertain to an individual, the features may be education, gender, income, marital status, number of children, location, credit history, citizenship, etc. The class label may be "high-credit risk" or "low credit risk". Such a classification may be pertinent to a mortgage company considering an applicant for a loan on a house. Data structures of this type are often alluded to as *labeled* data. The labeled data set is also known as *training data*. Classification begins under the assumption that a relationship exits between a feature vector and the corresponding class variable. The training data is used to determine that relationship. There are several approaches to deriving these relationships. Common techniques involve statistical methods, non linear modeling, and methods based on heuristics (a

commonsense rule or set of rules intended to increase the probability of solving some problem). We will discuss these approaches in detail in the ensuing sections.

Section 5.1 Statistical Classification

Statistical classification is a methodology that utilizes probabilistic ideas and concepts to derive *decision rules* also known as *classifiers* such that some optimality criterion is met. Consider the task of classifying dogs and cats into two groups. Since the features that distinguish adult dogs and cats are so unique and distinct it is a relatively simple exercise. More often than not features that distinguish objects are not so distinct and unique. Features may overlap to a degree that perfect classification may be is impossible. This may especially be true in the loan mortgage example we saw in the introductory section. In the scenario where features belonging to different classes are not 100% unique, classification schemes are prone to error. Two types of *error* are likely (1) Assign an observation that truly belongs to class A to class B, and (2) Assign an observation that truly belongs to Class B to Class A. The errors and correct assignments may be expressed as a 2x2 matrix

Notes:

known as the *confusion matrix* to be formally introduced later. It is therefore desirable that we derive procedures that minimize the combined error of miss-assignment to the two classes. Principal among probability based methods are Bayesian classifiers and Fisher discriminant analysis. Again, in this formulation, a *plug-in* decision rule is derived. Probability based methods begin with the training data and the feature vector is assumed to follow a probability distribution. The training data is used to estimate the unknown parameters which are then plugged into the decision rule to determine class membership of a new feature vector. Since assignment depends on statistics computed from the labeled data set, the training data is known as the supervisor. Artificial neural network (ANN) is a non-parametric, non linear modeling technique. In this approach, an architecture for the neural network is conceived, and features are classified based on a minimum sum-of-squares error criterion. The *K*-nearest neighbor classifier is a heuristics based method. Decision trees, support vector machines, and projection pursuit are other popular classification methods widely in use.

Common thread that runs across all these methods is a procedure that assigns an entity to a class based on a similarity measure with respect to some sort of an optimality criterion or a rule. The criterion may be maximize probability of a given class given an entity, minimize the distance between an entity and a given class (the class may be summarized in terms of its numerical characteristics). In this chapter

we will delve into the depths of these techniques pointing out merits, demerits, and other attendant caveats.

Section 5.2 Statistical discrimination

Let us revisit the classification problem related to adult cats and dogs. Imagine that we do not get to see the *canines* and *felines*. However we have a database of records consisting of the following features and species. The features include length of tongue, tail length, skull size, and number of claws. So the four features together for each animal constitutes a four dimensional vector. The class variable is "felines" or "canines." In this setting, assigning animals to the appropriate class is not so straight forward due to phylogenetic similarities and potentially due to lack of other distinguishable characteristics unique to each species. However the variations in the data can be described via statistical distributions, prior knowledge about the species, and use Bayesian methods to enable classification. Since the features vary from animal to animal within a class, we may describe the data mathematically by a probability distribution. Figure 5.1 shows distributions of measurements related to

Notes:

length of tail of canines and felines. Since the data is separated by class labels in the database, the within class proportions known variously as prior probabilities, mixing proportions, component densities can be easily computed.

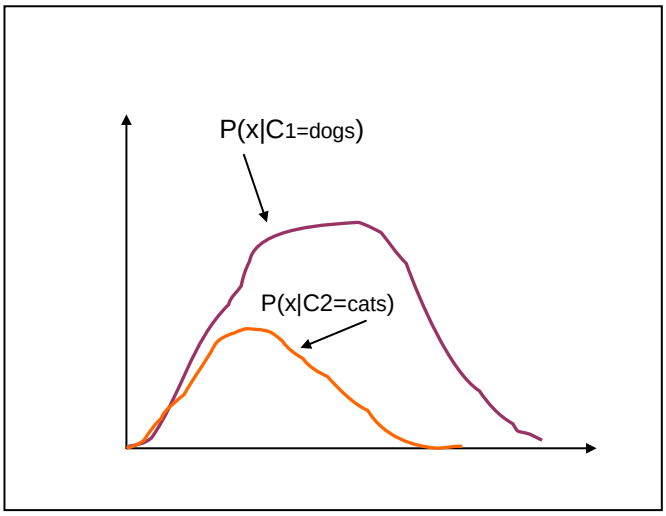

Figure 5.1 Probability distribution of tail lengths for felines and canines

Bayesian learning exploits the Bayesian paradigm. In this setting, the sampling distribution of data and prior probability of classes are combined to produce a probability distribution known as the posterior distribution. Consider the two-category problem. Let the classes be denoted by $C_i, (i = 1,2)$ and prior probabilities of

Notes:

the classes, π and $1-\pi$. Given an observation x, the posterior probabilities of the two classes C_1 and C_2 are:

$$\pi(C_1 \mid x) = \frac{\pi(x \mid C_1)\pi}{\sum_{i=1}^{2} f(x \mid C_1)\pi + f(x \mid C_1)(1-\pi)}$$

$$\text{and } \pi(C_2 \mid x) = \frac{f(x \mid C_2)(1-\pi)}{\sum_{i=1}^{2} f(x \mid C_1)\pi + f(x \mid C_1)(1-\pi)} \qquad (5.1)$$

Note that $\pi(x \mid C_i), i = 1,2$, is the sampling distribution of the observation vector given the class. If $\pi(C_1 \mid x) > \pi(C_2 \mid x)$, then it is reasonable to suspect that x belongs to class C_1 or C_2 otherwise. Since we need to make classification decisions for each x, it is desirable to minimize the average probability of error. Given an observation x, associated error probabilities are

$$P(error \mid x) = \begin{cases} P(C_1 \mid x), if & x \in C_2 \\ P(C_2 \mid x), if & x \in C_1 \end{cases} \qquad (5.2)$$

These error probabilities can be minimized by choosing C_1, if $P(C_1 \mid x) > P(C_1 \mid x)$, or C_2 otherwise. As minimizing the average probability of error is a more desirable, We compute the following integral:

$$P(error, x) = \int_{-\infty}^{\infty} P(errror \mid x) f(x) dx \qquad (5.3)$$

Small values of integrand for every value of x will achieve the desired objective of minimum average probability. Thus the notion of minimizing average probability of error justifies using posterior probabilities for classification. The rule based on minimum average probability is known as *Bayes decision rule* given by

Assign x to class C_1, if $P(C_1 \mid x) > P(C_2 \mid x)$, otherwise Assign x to class C_2.

Section 5.3 Fisher discriminant analysis

Statistical discriminant analysis is a classification methodology that computes a discriminant function for classifying observations into one of *k* categories on the basis of a set of quantitative variables. More formally, discriminant analysis partitions a *p*-dimensional vector space into regions R_j (j=1,2,…,k), where R_j is the sub space containing all *p*-dimensional vectors *x* such that $P(C_i \mid x)$ is a maximum.

Under the assumption of multivariate normality, one can easily compute $P(C_j \mid x)$, or some monotonic function $g\{P(C_j \mid x)\}$. The posterior probability density and its variants serve as instruments or *decision rules* for classification. As with any method that attempts to perform classification under uncertainty, misclassification errors are

Notes:

inevitable. Discriminant analysis provides a framework for classification such that the total probability of error is minimized. More formally stated, based on misclassification probabilities we can derive regions in the data space obtained by minimizing the *total probability of error*. The equivalence between minimum total error probability and maximum posterior probability of classification $P(C_j \mid x)$ can be easily shown under assumptions of normality of the observation vector.

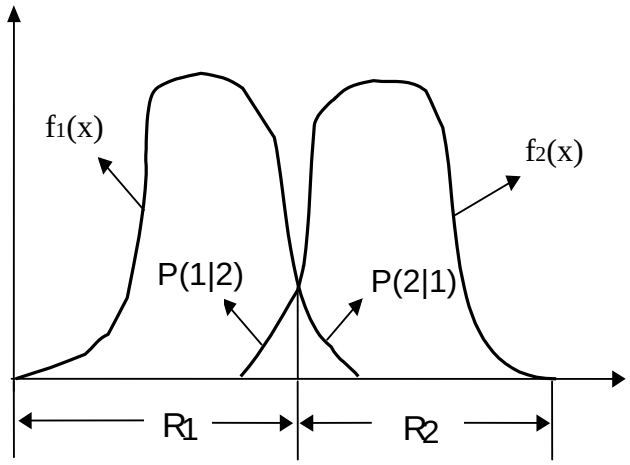

Figure 5.2. P(1|2) and P(2|1) show regions that correspond to error of misclassification. Computing areas under curves gives the total probability of error. the areas under regions, determines total probability of error.

Consider Figure 5.2. The quantities $P(1|2)$ and $P(2|1)$ correspond to areas of probabilities of misclassification. It is desirable to derive a classification procedure that minimizes the total misclassification error and/or expected cost of total error. In the following we will derive regions that address classification error.

First let us consider misclassification error relative to Figure 5.2.

$P(error)$ = P(x falls in R_1 and indeed $x \in C_2$) + P(x falls in R_2 and indeed $x \in C_1$)

Using fundamental rules of probability, the above equation decomposes to

$$P(error) = \int_{x \in R_1} P(x|C_2)\pi_2 dx + \int_{x \in R_2} P(x|C_1)\pi_1 dx \qquad (5.4)$$

Mathematically, this translates to evaluating the term:

$$P(error) = \pi_2 \int_{x \in R_1} f_2(x)dx + \pi_1 \int_{x \in R_2} f_1(x)dx \qquad (5.5)$$

Notes:

Using the fact that $\int_{x \in R_1} f_1(x)dx + \pi_1 \int_{x \in R_2} f_1(x)dx = 1$, we have

$$P(error) = \pi_1 + \int_{x \in R_1} [\pi_2 f_2(x) - \pi_1 f_1(x)]dx \qquad (5.6)$$

The error is minimized when the integrand is as small as possible, i.e.,

$\pi_2 f_2(x) - \pi_1 f_1(x) < 0$, i.e., assign the feature vector x to region R_1 if $\frac{f_1(x)}{f_2(x)} \geq \frac{\pi_2}{\pi_1}$

, otherwise to R_2 (5.7)

We started the discussion talking about the expected cost incurred due to misclassification. However we minimized total probability of error. To minimize the expected loss due to misclassification, we simply had to consider;

$$P(error) = \int_{x \in R_1} P(x \mid C_2)\pi_2 e_{12} dx + \int_{x \in R_2} P(x \mid C_1)\pi_1 e_{21} dx$$

where e_{12} and e_{21} are the costs due to classifying an observation to class 1 when it truly belongs to class 2, and vice-versa. In our derivation for the regions, we assumed equal costs for the two types of error.

It may be informative to determine the separating line that determines regions R_1 and R_2 using a univariate normal distribution as the sampling distribution of the data. The data is a set of *n* observations of a single feature related to an entity. Let $x_1, x_2, ..., x_n$ be a random sample from a normal population. In particular given an $x_i = x$, we have; $f_1(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2\sigma^2}(x-\mu_1)^2}$ and $f_2(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2\sigma^2}(x-\mu_2)^2}$. Note that we are assuming equal variances $\sigma_1^2 = \sigma_2^2 = \sigma^2$. The ratio; $\frac{f_1(x)}{f_2(x)} = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2\sigma^2}(x-\mu_1)^2}}{\frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2\sigma^2}(x-\mu_2)^2}}$ resolves to a simpler expression upon taking logarithms and some algebra. Thus,

$$\log\left(\frac{f_1(x)}{f_2(x)}\right) = \frac{\mu_2^2 - \mu_1^2}{2} + x(\mu_1 - \mu_2) \qquad (5.8)$$

Assuming $\pi_1 = \pi_2 = \pi$ in 5.7 and taking logarithm;

$$\log\left(\frac{f_1(x)}{f_2(x)}\right) \geq \log\left(\frac{\pi_2}{\pi_1}\right) = \log(1) = 0 \Rightarrow \log\left(\frac{f_1(x)}{f_2(x)}\right) = \frac{\mu_2^2 - \mu_1^2}{2} + x(\mu_1 - \mu_2) \geq 0$$

Notes:

or equivalently:

$$\frac{\mu_2^2 - \mu_1^2}{2} + x(\mu_1 - \mu_2) \geq 0 = x \geq \frac{\mu_1 + \mu_2}{2} \qquad (5.9)$$

Since μ_1 and μ_2 are unknown, we replace them by their sample estimates, $\bar{x}_1$ and $\bar{x}_2$.

Clearly the boundary between regions R_1 and R_2 is linear given by

$$x \geq \frac{\bar{x}_1 + \bar{x}_2}{2} \qquad (5.10)$$

Linear Discriminant analysis gives us precisely this linear boundary.

In the ensuing sections, we will discuss linear Discriminant analysis and its cousin quadratic Discriminant analysis in substantial depth in a multivariate setting.

Section 5.4 The Linear Discriminant function

Consider the Multivariate Gaussian density function

$$f\left(\underset{\sim}{x} : \underset{\sim}{\mu}, \underset{\sim}{\Sigma}\right) = \frac{1}{2\pi^{p/2}|\underset{\sim}{\Sigma}|^{1/2}} exp^{-\frac{1}{2}\left(\underset{\sim}{x} - \underset{\sim}{\mu}\right)^T \underset{\sim}{\Sigma}^{-1}\left(\underset{\sim}{x} - \underset{\sim}{\mu}\right)}.$$

This function is a multivariate extension of the univariate normal density we saw

Notes:

earlier. Assuming a normal probability model as the underlying stochastic process for the feature vector $\underset{\sim}{x}$, we derive the classical Fisher discriminant also known as Fisher discriminant analysis.

Let $\underset{\sim}{x} = (x_1, x_2, \ldots\ldots, x_p)$ be a p dimensional vector of attributes or features. Let C_i, $i = 1,2,....,k$ represent a set of k classes. Each vector $\underset{\sim}{x}$ belongs to one of the k classes. At the outset each of the k classes are populated by the vectors $\underset{\sim}{x}$. Their assignment to each class is determined by subject matter expertise and strong technical judgment. The schematic below describes the pre-defined library of features in the k classes.

	class1						class2									classk					
	X111	X121	X131	.	.	X1p1	X112	X122	X132	.	.	X1p2				X11k	X12k	X13k	.	.	X1pk
	X211	X221	X231	.	.	X2p1	X212	X222	X232	.	.	X2p2				X21k	X22k	X23k	.	.	X2pk
	X311	X321	X331	.	.	X3p1	X312	X322	X332	.	.	X3p2	.	.	.	X31k	X32k	X33k	.	.	X3pk
	.	.	.	.	.	.	.	.	.	.	.	.				.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.				.	.	.	.	.	.
	Xn11	Xn21	Xn21	.	.	Xnp1	Xn12	Xn22	Xn22	.	.	Xnp2				Xn1k	Xn2k	Xn2k	.	.	Xnpk

Given the classes populated by feature vectors $\underset{\sim}{x}$, we can easily compute the prior

Notes:

probability of each of the k classes. The prior probabilities are given by:

$P(C_i) = \frac{n_i}{\sum_{i=1}^{k} n_i}$, $(i = 1,2....,k)$, where n_i equals number of feature vectors in class C_i, and $\sum_{i=1}^{k} n_i$ is the totality of all features in the k classes.

Discriminant analysis attempts to revise the prior probabilities $P(C_i)$ in the light of the new vector. In other words, Discriminant Analysis tries to assign a new vector to that group for which the posterior probability of classification given by $P(C_i \mid x_{new})$ is a maximum. The evaluation of the posterior probability is achieved by invoking the Bayes theorem.

- *Bayes theorem*

Let $A_i (i = 1,2,....,k)$ be set of k mutually exclusive events such that $P\left(\bigcup_{i=1}^{k} A_i\right) = 1$. Let the conditional probabilities $P(B|A_i), i = 1,2,....,k$ be given as well, where B is another event that occurs along with $A_i (i = 1,2,....,k)$. The Bayes theorem postulates that the posterior probability is given by:

Notes:

$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{\sum_{i=1}^{k} P(B \mid A_i)P(A_i)},$$

where the denominator is due to the law of total probability.

By the application of the Bayes theorem, the posterior probability of $P(C_i \mid x_{new})$ is given by:

$$P(C_i \mid x_{new}) = \frac{P(x_{new} \mid C_i)P(C_i)}{\sum_{i=1}^{k} P(x_{new} \mid C_i)P(C_i)}.$$

Since the denominator is common across all classes, it is dropped from computation. Critical to the computation of the posterior probability is the term $P(x_{new} \mid C_i)$, which is simply the probability distribution of the observation vector. It simply encodes the underlying stochastic process that is producing the data in each class.

Our decision rule denoted by $d_i(x)$ is given as:

$$d_i(x) = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log\left|\Sigma_{pooled}\right| - \frac{1}{2}(x-\mu_i)'\Sigma_{pooled}^{-1}(x-\mu_i) + \log P(C_i) \qquad (5.11)$$

Since the first and second terms in the expression for $d_i(x)$ are common across all classes, they can be dropped from the decision rule. The third term in $d_i(x)$ is called the *generalized Mahalanobis distance*. Therefore, the decision rule for classifying an observation into one of *k* classes is given by:

$$\arg\max_i \left(d_i(x)\right) \qquad (5.12)$$

The expression for the pooled covariance matrix is given by:

$$\Sigma_p = \frac{(n_1 - 1)\Sigma_1 + (n_2 - 1)\Sigma_2 ..... + (n_k - 1)\Sigma_k}{n_1 + n_2 + ... + n_k - k} \quad (5.13)$$

In the actual computation of the decision rule, Σ_p is replaced by its sample analog:

$$S_p = \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2 ..... + (n_k - 1)S_k}{n_1 + n_2 + ... + n_k - k} \quad (5.14)$$

In effect, Discriminant analysis measures proximities between a new observation $x_{(new)}$ and the k classes and assigns it to that class for which $d(x_{(new)}, C_j), j = 1,2,....,k$ is the smallest. The Mahalanobis distance metric can be based on either the full covariance matrix or simply the diagonal matrix of variances (correlation matrix).

Section 5.5 Discriminant analysis as a classifier

In the application of discriminant analysis as a classifier, the labeled data is split into two sets; *training* data set and *validation* data set. The proportion of samples in each data set is arbitrary. In practice, 60-70% of the samples are utilized for training and the reminder for validation. The training data set is used to estimate the prior probabilities of the classes, the class conditional mean vectors and covariance

Notes:

matrices. If the decision rule is based on the linear discriminant function, the class conditional covariance matrices are replaced by the pooled covariance matrix. The parameter estimates replace the unknown parameters in $d_i(x)$. Next each observation in the validation set is plugged into $d_i(x), i = 1,2,....,k$ and is classified into that group for which $d_i(x)$ is a maximum. Performance of the classifier is evaluated by computing the apparent error rate (AER). Figure 5.3 is the confusion matrix associated with performance.

Predicted class / Actual Class	C_1	C_2
C_1	n_{11}	n_{12}
C_2	n_{21}	n_{22}

Figure 5.3. Confusion matrix that is often constructed to compute the apparent error rate (AER) relative to a classifier

Mathematically, apparent error rate is simply;

$$AER = \frac{n_{12} + n_{21}}{n_{11} + n_{12} + n_{21} + n_{22}} \qquad (5.15)$$

Classifier training and subsequent evaluation may be improved by using resampling techniques such as jackknife, Bootstrap and cross validation to generate testing-sets known as *hold-out* samples. Broadly, these schemes involve holding out a percentage of the sample (say 10%) for validation, and the remaining is used for training. The procedure is repeated by sampling hold-out samples of sizes 10%.

Section 5.6 Cross validation

The cross validation methodology, also known as *k-fold* cross validation is a techniques wherein the labeled data set is divided into *k* sets of equal size. The classifier is trained k times, such that one of the k sets is held out as a validation data set at each training step. The performance of the classifier is simply the average apparent error rate across the k training data sets. A similar approach to estimating classifier accuracy is the jackknife method. Jackknife estimation of classifier accuracy entails training the classifier *n* times, using samples $n_{(i)}, i = 1,2,....,n$, where $n_{(i)} = n-1$, with the i^{th} observation deleted at each iteration. Again performance is evaluated by averaging the accuracies over the *n*

Notes:

training data sets. Another variant of cross-validation is the bootstrap. In this approach simple random samples of size *n* with replacement are drawn repeatedly, and the classifier is trained at sampling step. An estimate of classifier accuracy is simply the average of training error over the *n* samples.

Example 5.1

The *iris* data published by Dr. Ronald A. Fisher in 1936 is widely used to benchmark performance of classifiers. Iris data consists of a total of 150 observations on sepal length, sepal width, petal length, and petal width measured in millimeters on fifty iris specimens from each of three species, iris *setosa, iris versicolor,* and *iris virginica*. We will apply linear and quadratic discriminant types of classifiers to iris data and evaluate their performance.

The data set is given in Table 5.1. Each block of five measurements corresponds to one sample. The five measurements are Sepal Length Sepal Width Petal Length Petal Width and Species. Sepal length, sepal width, Petal length, and petal width are the input attributes peculiar to each species. The species variable is the class

Notes:

variable. Since the data set contains input attributes data as well as the corresponding class label, it is labeled and lends itself to application of supervisory learning methods.

```
 50 33 14 02 1 64 28 56 22 3 65 28 46 15 2 67 31 56 24 3
63 28 51 15 3 46 34 14 03 1 69 31 51 23 3 62 22 45 15 2
59 32 48 18 2 46 36 10 02 1 61 30 46 14 2 60 27 51 16 2
65 30 52 20 3 56 25 39 11 2 65 30 55 18 3 58 27 51 19 3
68 32 59 23 3 51 33 17 05 1 57 28 45 13 2 62 34 54 23 3
77 38 67 22 3 63 33 47 16 2 67 33 57 25 3 76 30 66 21 3
49 25 45 17 3 55 35 13 02 1 67 30 52 23 3 70 32 47 14 2
64 32 45 15 2 61 28 40 13 2 48 31 16 02 1 59 30 51 18 3
55 24 38 11 2 63 25 50 19 3 64 32 53 23 3 52 34 14 02 1
49 36 14 01 1 54 30 45 15 2 79 38 64 20 3 44 32 13 02 1
67 33 57 21 3 50 35 16 06 1 58 26 40 12 2 44 30 13 02 1
77 28 67 20 3 63 27 49 18 3 47 32 16 02 1 55 26 44 12 2
50 23 33 10 2 72 32 60 18 3 48 30 14 03 1 51 38 16 02 1
61 30 49 18 3 48 34 19 02 1 50 30 16 02 1 50 32 12 02 1
61 26 56 14 3 64 28 56 21 3 43 30 11 01 1 58 40 12 02 1
51 38 19 04 1 67 31 44 14 2 62 28 48 18 3 49 30 14 02 1
51 35 14 02 1 56 30 45 15 2 58 27 41 10 2 50 34 16 04 1
46 32 14 02 1 60 29 45 15 2 57 26 35 10 2 57 44 15 04 1
50 36 14 02 1 77 30 61 23 3 63 34 56 24 3 58 27 51 19 3
57 29 42 13 2 72 30 58 16 3 54 34 15 04 1 52 41 15 01 1
71 30 59 21 3 64 31 55 18 3 60 30 48 18 3 63 29 56 18 3
49 24 33 10 2 56 27 42 13 2 57 30 42 12 2 55 42 14 02 1
49 31 15 02 1 77 26 69 23 3 60 22 50 15 3 54 39 17 04 1
66 29 46 13 2 52 27 39 14 2 60 34 45 16 2 50 34 15 02 1
44 29 14 02 1 50 20 35 10 2 55 24 37 10 2 58 27 39 12 2
47 32 13 02 1 46 31 15 02 1 69 32 57 23 3 62 29 43 13 2
74 28 61 19 3 59 30 42 15 2 51 34 15 02 1 50 35 13 03 1
56 28 49 20 3 60 22 40 10 2 73 29 63 18 3 67 25 58 18 3
49 31 15 01 1 67 31 47 15 2 63 23 44 13 2 54 37 15 02 1
56 30 41 13 2 63 25 49 15 2 61 28 47 12 2 64 29 43 13 2
51 25 30 11 2 57 28 41 13 2 65 30 58 22 3 69 31 54 21 3
54 39 13 04 1 51 35 14 03 1 72 36 61 25 3 65 32 51 20 3
61 29 47 14 2 56 29 36 13 2 69 31 49 15 2 64 27 53 19 3
68 30 55 21 3 55 25 40 13 2 48 34 16 02 1 48 30 14 01 1
45 23 13 03 1 57 25 50 20 3 57 38 17 03 1 51 38 15 03 1
55 23 40 13 2 66 30 44 14 2 68 28 48 14 2 54 34 17 02 1
51 37 15 04 1 52 35 15 02 1 58 28 51 24 3 67 30 50 17 2
 63 33 60 25 3 53 37 15 02 1
```

Table 5.1. 150 observations on the four attributes of three species of flowers

We will use the *proc discrim* procedure with the linear discriminant analysis option available in SAS® to produce the results. The data set consisting 50 observations is set aside as a validation set to evaluate how well the method generalizes. The 50 observations were randomly sampled from the three species types. Fragments of output from the SAS program is reproduced in Output Tables 5.2(a)-5.2(e).

Class Level Information

spec_no	Variable Name	Frequency	Weight	Proportion	Prior Probability
1	_1	26	26.0000	0.262626	0.333333
2	_2	40	40.0000	0.404040	0.333333
3	_3	33	33.0000	0.333333	0.333333

Table 5.2(a) Summary of data in the three classes. Note that spec_no is species number. 1,2, and 3 map to setosa, versicula and virginica respectively. Since there are 50 observations in each species type, the prior probability of classes is 0.33 each.

The DISCRIM Procedure
Classification Summary for Calibration Data: WORK.TRAIN
Resubstitution Summary using Linear Discriminant Function

Generalized Squared Distance Function

$$D_j^2(X) = (X-\bar{X}_j)' \, COV^{-1} \, (X-\bar{X}_j)$$

Notes:

Posterior Probability of Membership in Each spec_no

$$Pr(j|X) = \exp(-.5\ D_j^2(X)) \,/\, \sum_k \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into spec_no

From spec_no	1	2	3	Total
1	26	0	0	26
	100.00	0.00	0.00	100.00
2	0	39	1	40
	0.00	97.50	2.50	100.00
3	0	1	32	33
	0.00	3.03	96.97	100.00
Total	26	40	33	99
	26.26	40.40	33.33	100.00
Priors	0.33333	0.33333	0.33333	

Table 5.2(b) Output shows classification of observations from one species class into another species class. Notice that none of the samples belonging to species setosa is misclassified. However, 1 out 40 from versicula, and 1 out 33 from virginica are misclassified. These results produced using the training data.

Error Count Estimates for spec_no

	1	2	3	Total
Rate	0.0000	0.0250	0.0303	0.0184
Priors	0.3333	0.3333	0.3333	

Table 5.2(c). This Table is simply a compendium of error rates/misclassification rates and prior probabilities.

The DISCRIM Procedure
Classification Summary for Test Data: WORK.VALID
Classification Summary using Linear Discriminant Function

Generalized Squared Distance Function

$$D_j^2(X) = (X-\bar{X}_j)'\ COV^{-1}\ (X-\bar{X}_j)$$

Posterior Probability of Membership in Each spec_no

$$Pr(j|X) = \exp(-.5\ D_j^2(X)) \,/\, \sum_k \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into spec_no

From spec_no	1	2	3	Total
1	24	0	0	24
	100.00	0.00	0.00	100.00

2	0	10	0	10
	0.00	100.00	0.00	100.00
3	0	0	17	17
	0.00	0.00	100.00	100.00
Total	24	10	17	51
	47.06	19.61	33.33	100.00
Priors	0.33333	0.33333	0.33333	

Table 5.2 (d) Output shows classification of observations from one species class into another species class. Notice that none of the samples belonging to any of the three species is misclassified. These results produced using the validation data set consisting of 51 observations. .

Error Count Estimates for spec_no

	1	2	3	Total
Rate	0.0000	0.0000	0.0000	0.0000
Priors	0.3333	0.3333	0.3333	

Table 5.2(e) This Table is simply a compendium of error rates/misclassification rates and prior probabilities.

Applying quadratic discriminant analysis produce results identical to linear discriminant analysis.

Example 5.2: Predicting profit or loss based on financial ratios of publicly traded Indian companies

The study attempts to determine the financial health of publicly listed Indian companies by making use of various financial ratios. The data used consists of 14 ratios and a profit/loss indicator for about 100 companies, for the financial year 2001-02. The objective is to select a small number of ratios from the chosen set, which taken together can best model the financial health (measured as profit/loss) of a company. The companies chosen span across multiple industries, and are of diverse financial status.

The fourteen features chosen are, *Working Capital/Total Assets, EBIT/Total Assets (ROI),Sales/Total Assets (total asset turnover ratio),EBIT/Total Interest Payments (Interest Coverage Ratio),Current Assets / Current Liabilities (Current Ratio),Debt/Equity (Debt-Equity*

Ratio),Sales/fixed assets (Fixed Assets Turnover Ratio,Debt/Assets (Debt Asset Turnover Ratio),Net Sales / Inventory (Inv. Turnover Ratio),Gross profit / net sales (gpm ratio),Net profit / net sales (NPM),Net profit / total assets (Net Income to Asset Ratio),Return on Equity (Net Profit / Shareholder's Equity),Net Profit / Debt. The classification variable is a coded one or zero based on profit or loss.

After talking to domain experts, and applying step-wise discriminant analysis, the input feature vector is whittled down to Return on Equity (ROE) and Net profit / Total Assets.

German credit data set

This data set is available in the SAS permanent library, SAMPSIO.DMLGECR and is related to scoring a set of individuals based on a set of demographic and other transactional and finance related variables. Table below is the list of variables used. Note that the target variable is given as good_bad.

DMLGECR data does not contain the good_bad variable and can be used as a score data.

Variable	Model Role	Measurement	Description
age	input	interval	age in years
amount	input	interval	credit amount
checking	input	nominal or ordinal	status of existing checking account 1: ... < 0 DM 2: 0 <= ... < 200 DM 3: ... >= 200 DM 4: no checking account

Notes:

coapp	input	nominal	other debtors/guarantors 1: none 2: co-applicant 3: guarantor
depends	input	interval	number of dependents
durations	input	interval	duration in months
employed	input	ordinal	present employment since 1: unemployed 2: ... < 1 year 3: 1 <= ... < 4 years 4: 4 <= ... < 7 years 5: ... >= 7 years
existcr	input	interval	number of existing credits at this bank
foreign	input	binary	foreign worker 1: yes 2: no
good_bad	target	binary	credit rating
history	input	ordinal	credit history 0: no credits taken / all credits paid back duly 1: all credits at this bank paid back duly 2: existing credits paid back duly till now 3: delay in paying off in the past 4: critical account / other credits existing (not at this bank)
housing	input	nominal	housing 1: rent

			2: own 3: for free
installp	input	interval	installment rate in percentage of disposable income
job	input	ordinal	job 1: unemployed / unskilled non-resident 2: unskilled resident 3: skilled employee / official 4: management / self-employed / highly qualified employee / officer
martial	input	nominall	personal status and sex 1: male -- divorced / separated 2: female -- divorced / separated / married 3: male -- single 4: male -- married / windowed 5: female -- single
other	input	nominal	other installment plans 1: bank 2: stores 3: none
property	input	nominal or ordinal	property 1: real estate 2: if not 1, building society savings agreement / life insurance 3: if not 1 or 2, car or others 4: unknown / no property
purpose	input	nominal	purpose

			0: new car 1: used car 2: furniture / equipment 3: radio / television 4: domestic appliances 5: repairs 6: education 7: vacation 8: retraining 9: business x: others
resident	input	interval	present residence since
savings	input	nominal or ordinal	status of existing saving account or bonds 1: ... < 100 DM 2: 100 <= ... < 500 DM 3: 500 <= ... < 1,000 DM 3: ... >= 1,000 DM 4: unknown / no saving account
telephon	input	binary	telephone 1: none 2: yes, registered under the customer's name

Section 5.7 Other non-parametric methods based classifiers

When no assumptions can be made about the underlying distribution of the data vector $\underset{\sim}{x}$ nonparametric methods can be used to estimate the class specific densities.

Some methods that come to mind are Parzen windows and the k nearest neighbor

methods. Some popular windows utilized are uniform, normal, Epanechnikov, biweight, and triweight kernels in the density estimation.

- *K-Nearest neighbor Classification*

This is another simple, yet intuitive non-parametric classification technique quite popular in many applications. The procedure begins with a training data set of observations, $\ell = \left(\underset{\sim}{X} * C \right)$, where $\underset{\sim}{X}$ is a nxp matrix of features, and C is a vector denoting the k classes. The distances $d\left(X_0, X_j, j = 1,2,....,n\right)$ between an unlabelled observation vector $X_0 \in \Re^p$ from the testing set, and k observations in the training set are computed, and X_o is assigned to the most frequently occurring class within the set of k observations. Figure 5.4 is a visual illustration of the technique using three classes.

Notes:

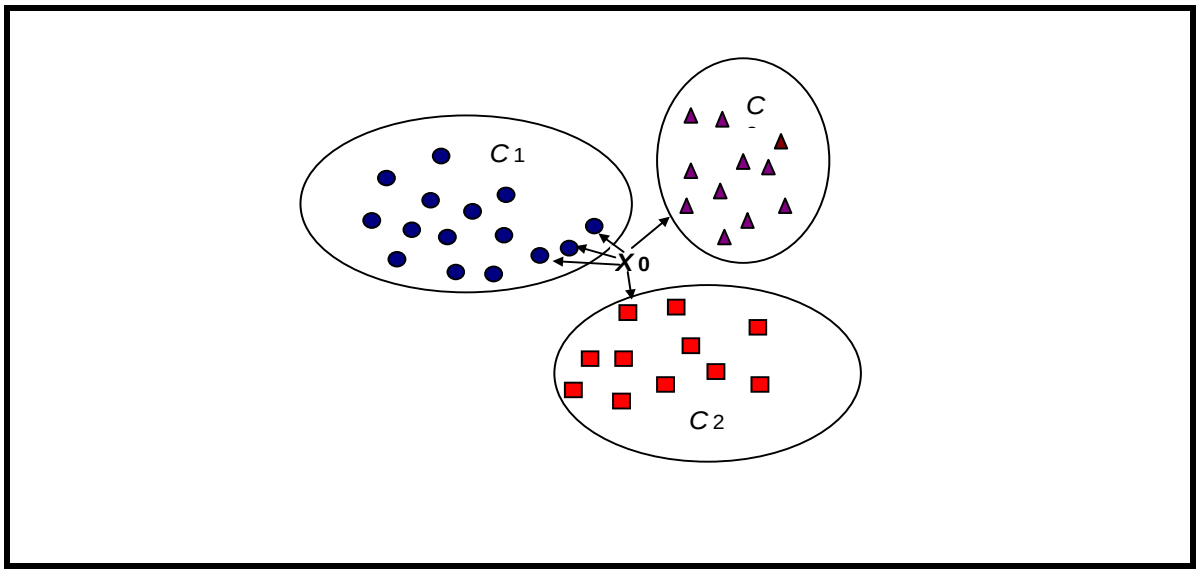

Figure 5.4. Classification by k-nearest neighbor algorithm. Notice that the observation X_0 is classified into category C_1 because the majority of the observations it is near to belong to class C_1.

Section 5.8 Artificial Neural Networks

By definition a neural network is a massively distributed parallel processor that acquires knowledge by optimizing the inter-neuron synaptic strengths by a learning process. The learning process consists of an algorithm that systematically modifies the synaptic strengths to achieve the desired objective. The architecture of a neural network is similar to the network of neurons within a human brain responsible for memory, pattern recognition and a host of other activities. Figure 5.5 is a pictorial

Notes:

representation of a neuron.

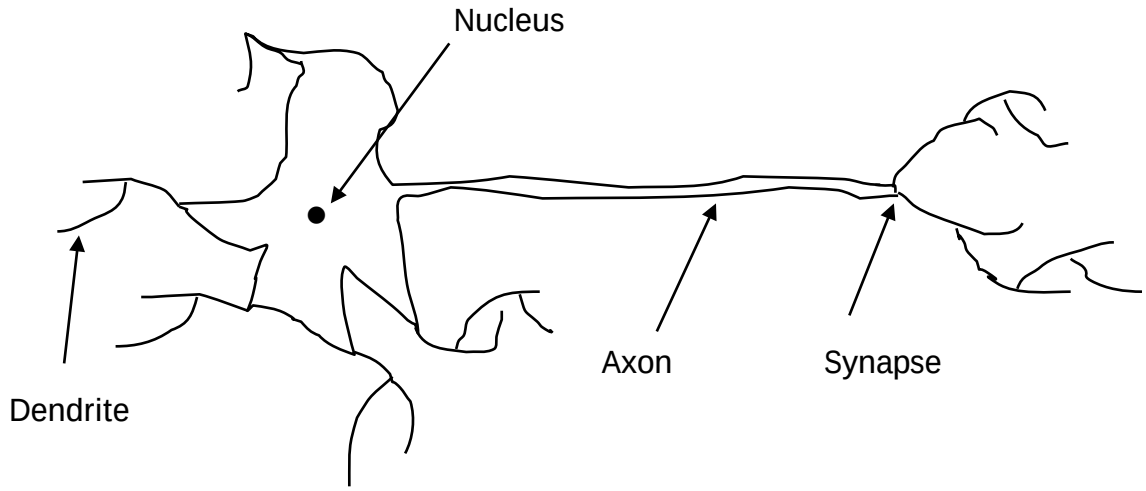

Figure 5.5 Pictorial representation of a neuron

The computational elements that comprise a neural network, akin to the neurons are known variously as nodes, units, or processing elements. A neural network is an arrangement of neurons organized in layers. The simplest neural network is a single layer network consisting of neurons in the input layer feeding into an output layer. A

Notes:

schematic of a single layer *feed-forward* network is given in Figure 5.6. The cumulative effect of the input layer nodes at the output node is a sum of the product of input values and their corresponding weights (synaptic strengths) plus a bias term.

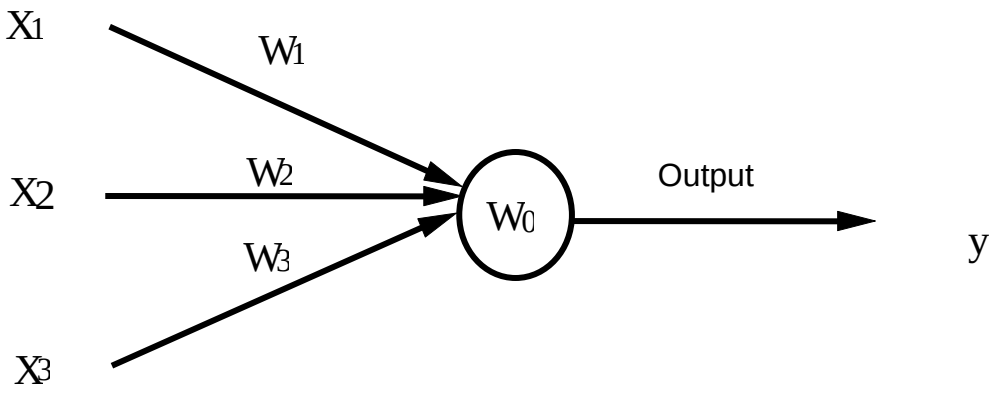

Fig. 5.6 Single layered feed-forward network

It is feed-forward in that the connections between inputs and outputs are in one (forward) direction only.

Here, $X_j, (j = 1,2,3)$ Xj are the inputs and $W_j, (j = 1,2,3)$ are the weights (synaptic strengths) and W_0 is called the bias term and finally $y = \sum_{j=1}^{3} w_j x_j + w_0$ is the output.

Output at a receiving neuron (node) is attenuated or amplified by an activation function. When an incoming signal, y exceeds a threshold, the neuron fires. Incoming signals y to a receiving neuron are of two types: (1) inhibitory, and (2) excitatory. Excitatory inputs cause a neuron to fire, inhibitory inputs cause a neuron to remain passive. Some common activation functions are:

1. $f(x) = \text{sgn}(x)$. This activation function is also called hard limiter nonlinearity that produces a binary output (± 1)

2. $f(x) = [\text{sgn}(x) + 1]/2$. This activation function produces a binary output

3. $f(x) = \tanh(x)$ is also an activation function popular among practitioners.

4. $f(x) = 1 + \exp^{-x}$. This activation function is called the sigmoid (logistic) nonlinearity and produces an output in the interval $(0,1)$. The sigmoid transformation is extensively applied to the data in applications involving feed-forward neural networks using the *back-propagation* algorithm. Figure 5.7 is the sigmoid activation function.

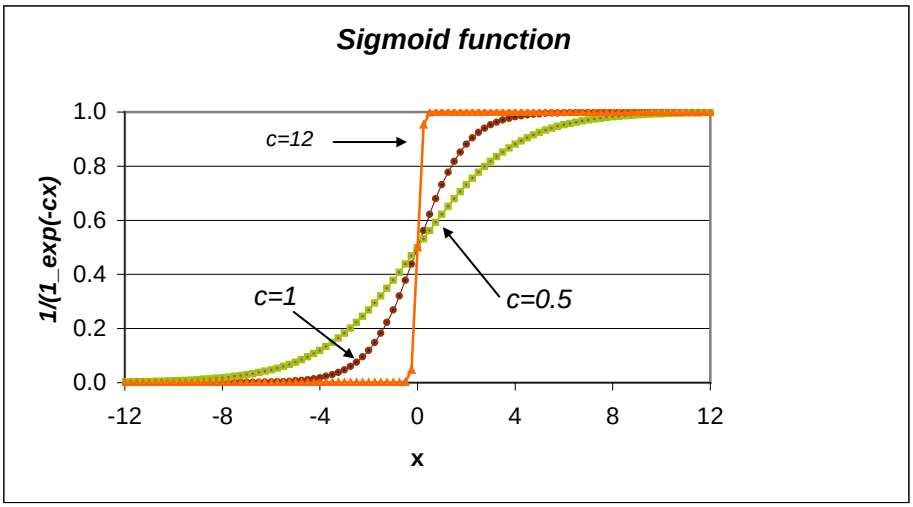

Figure 5.7 The sigmoid activation function for the adjustable parameter c equal to 12, 0.5, 1. Notice that for large "c", the function is approximately hard limiter, for c=0.5,1 it is largely linear.

Applying sigmoid activation to the output at any node "*j*" is given by

$$\frac{1}{1 + exp-(w_0 + \sum_{i=1}^{N_i} w_i x_i)} \qquad (5.16)$$

Where N_i is the number of input layer nodes (the size of the feature vector).

A natural extension is a *multi-layered feed-forward* network which consists of an input layer of neurons, a hidden layer of neurons feeding into an output layer of neurons. The number of hidden layers can be greater than one. A schematic of a fully connected, multi-layered feed-forward neural network with one hidden layer is illustrated in Figure 5.8.

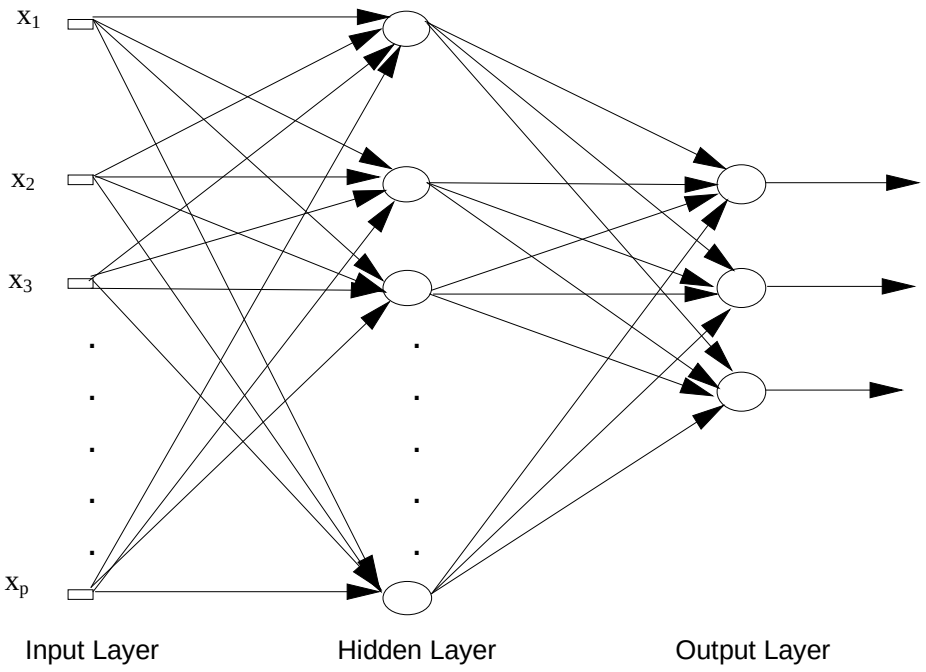

Figure 5.8 Architecture of a multi-layer feed forward neural network

Section 5.9 The back propagation algorithm

The goal of a neural network algorithm is to establish a relationship between the inputs and their corresponding responses. Neural networks are chosen as a resort because it is difficult to mathematically express a relationship between the inputs (vector of features) and the outputs (classes). We will in the following, formally define the constituents and the mechanics of the back propagation network which is usually effective in establishing a relationship or a mapping between the input signatures and the output classes.

Schematically, the back-propagation algorithm is described in Figure 5.9.

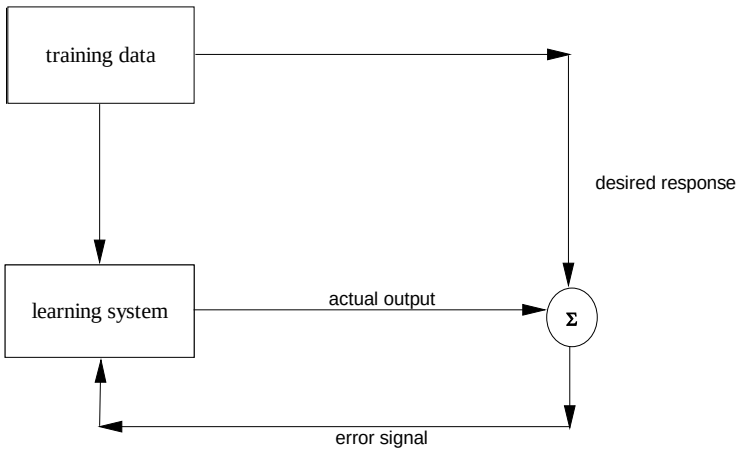

Figure 5.9 Supervisory Learning using back-propagation

Notes:

Suppose we have a set of data of n input-output pairs (the input-output pairs correspond to a vector of input values and corresponding class labels). The pairs are denoted by $(x_1, c_1), (x_2, c_2), ..., (x_n, c_k)$. The input-output pairs are such that $x \in \Re^P$, and $c \in \Re^K$. This is typically the case when the problem under consideration is classification. ANN may be used for a prediction problem, where the input-output pairs, $(x_i, y_i), i = 1,2,.....,n$ are quantitative in x_i and y_i. Back propagation algorithm is used to train the neural networks to develop an approximate relationship y ≅ f(x).

Section 5.10 Training with back propagation

The back propagation algorithm uses an iterative optimization technique to determine the best set of hidden-layer and output layer weights. Given an initial set of hidden-layer weights $(w_{ji}^h, (j = 1,,...., N_h))$, and an initial set of output-layer weights $(w_{kj}^o, (k = 1,,...., N_o))$, generated from the Gaussian or the uniform distribution, the method updates the weights by minimizing the sums of square of error. At each iteration,

- A multi layered feed-forward neural network with one hidden layer is preferable
- The elements of the input vector are chosen after a careful review of important input features.
- The data is scaled such that the domain of any given input variable is the unit interval (0,1).
- Sufficient training data in each labeled class is available prior to training.
- The number of hidden layer units is typically a third or half of the sum of input and output layer units.
- The hidden layer and output layer weights (synaptic strengths) are generated from a Gaussian(0,1)/Uniform(0,1) distribution. It is recommended a small value for the variance is chosen. The leaning rate η is usually chosen between (0.1-0.3).

- All the bias units may be set to zero in the entire network.
- The training data is split into two sets, namely the training set, and validation set.
- The *k-fold* cross-validation may be applied for training and validation.
- The training set is used to train several preliminary network architectures
- The validation set is used to identify the network with the least sums of squares of error.
- The network is trained using the pattern by pattern approach. The patterns are selected randomly from each class to eliminate any biases during learning.

Section 5.11 Some practical considerations

When the *sigmoidal* activation function is utilized, it seldom reaches zero or one. So, we set the output values zero and one to 0.1 and 0.9 respectively. There is always the nagging question about sample sizes needed to for good performance. In truth there is no hard and fast rule available. "As much data as possible," is a common refrain in the trade. Another popular rule of thumb is that the number of samples is at least 10 times the number of input variables.

The back-propagation algorithm is good at learning from the training data. It is able to grasp well within class variation in the exemplars. On the other hand, the

Notes:

algorithm does not forecast well if it is not adequately trained. In other words, the network performance is poor in the testing set. It should be ensured that the network is exposed to the whole input space of vector patterns.

During the learning phase, it should be ensured that inputs to the network is presented in a random order. If the order of presentation is by class, the network will tend to lose learning from earlier patterns belonging to later classes.

In many applications, limiting the number of layers to three appears to work well. Adding more layers degrades network performance in many cases. Unless performance increases considerably using higher order networks, three-layer networks are preferable.

Since the initial random weights are small, the sigmoid function tends to be linear during early training. Recall that Figure 5.7 showed that the sigmoid is mostly linear for small values of the adjustable parameter "*c*". The network gradually becomes non linear as the weights grow. When the weights are exactly set to zero, the derivatives are all zero, and the algorithm is stalled. Large weights lead to poor

Notes:

solutions more often than not.

Another issue that requires care is *overfitting*. Overfitting is a condition where a neural network with several layers and therefore many weight parameters will fit the data perfectly. This means that the error function is minimized at the global optimum. Such a network does not *generalize* well. Generalization is the ability of the network to predict well over the entire range of the input space of features. One strategy to avoid overfitting is the application of early stopping rules. Early stopping involves training the network partially and testing the ability of the network to predict on the testing data set. Since the network starts out with small weights generated from the uniform or standard normal distributions, the model tends to be linear as we observed earlier. In effect early stopping may yield a model that is eventually linear. If the final model is linear anyway, there does not seem to be much justification for neural network modeling. Remember neural network is our recourse when the relationship between the output variable and the input variables cannot be described by a functional relationship. Yet another way of dealing with overfitting is *weight*

decay. In this approach a penalty term is added to the objective function.

The debate over the number of hidden layer units is never ending. We provided some guidelines based on our experience and recommendations proposed in the literature. Some researchers argue that too few hidden layer units fail to capture inherent non-linearity in the data. Too many though tend to overfit. A way around this problem is using an objective function with a penalty term discussed earlier. Eventually selection of number of hidden layer units comes down to extensive experimentation using various network configurations.

The neural network optimizer is a non linear function with multiple local minima. Due to this condition, the final solution is sensitive to starting values for the weights. It is recommended that the network is trained using various settings of the starting values and select the one that yields the lowest penalized error. Another approach known as *bagging* is commonly used in applications. An *ensemble* of networks is trained using several bootstrap versions of the training data. The average predicted value from each bootstrap iteration is used as a final predicted value. See chapter three

for a deeper understanding of the method of bootstrap. As a cautionary note, it has been reported in the literature that bagging is useful when the classifier/predictor under consideration is a weak one. A weak classifier is one whose performance is slightly better than random guessing. See section 5.18 for more on bagging methods.

Section 5.12 Tree based classifiers: Decision Trees

Tree based classifiers known as *decision trees* are quite popular in applications. It may be visualized as a segmentation method which eventually assigns observations into categories based on a set of attributes. For example, a non-partisan organization gathers data relating to current hot-button issues to divine political leanings of a group of individuals. A variety of information regarding positions on various social, economic, and political issues is collected. The goal of the study is to determine party affiliation (democrat, republican or independent) on the basis of their responses. Figure 5.10 depicts a decision tree based classification rule derived using the organization's database of records.

Notes:

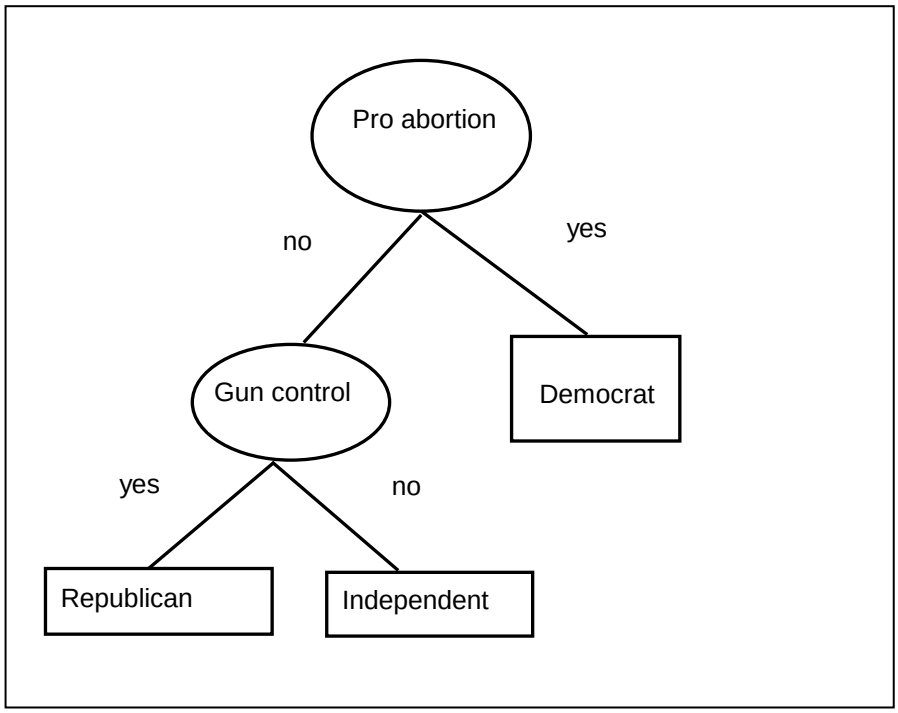

Figure 5.10. Rule generated by a decision to segment individuals into categories based on responses to questions about social, economic, and political issues

A decision tree is a supervisory learning method that relies on training data compiled *a priorily.* Training data is simply a set of k finite classes hosting the labeled samples. Each sample being a *p*-dimensional vector of attributes relative to an entity such as a person, corporation, or a geographical region. Recall that a labeled sample is a sample observation with associated class label. Decision tree classifiers are constructed by repetitively splitting the data set into subsets of two known as *binary splits* such that each descendent subset is purer than the antecedent set. The *purity* of the set pertains to amount of usable information contained in the set, commonly measured by the information theory based idea known as *entropy*.

[Breiman, 1984] argues that there is no convincing justification for entropy being the measure *de jure* for determining information content. Other measures parleyed in the literature will be discussed later. A decision tree classifier partitions a *p*-dimensional data space into sub regions such that each partition may isolate unique behavior or identity of a combination of attributes.

Section 5.13 Decision tree fundamentals

Let X denote the library of labeled data known as the training set. The training set X is referred to as the parent node. Each i^{th} element belonging to class k of the training set is an observation vector $x_i(k) = (x_{i1}, x_{i2}, ..., x_{ip})$. Each coordinate x_{ij} corresponds to a measured value of an attribute. Consider a multi-class classification problem involving four classes. The decision tree at the outset consists of a root node that is the entire data set. A good binary split is one that breaks the root into two nodes such that all observations belonging to classes one and two are assigned to the left child node (n_L) and those that belong to classes three and four are assigned to the right child node (n_R). The descendent children are further split to

Notes:

beget their *left* and *right* descendent nodes and so on. Figure 5.11 shows node splitting and proportions of observations assigned to the children nodes.

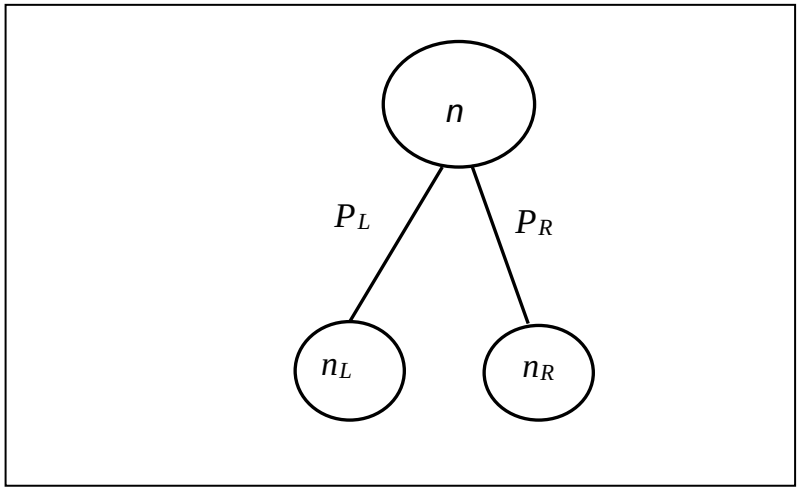

Figure 5.11. A binary split of a node into left and right children nodes

Section 5.14 Node impurity

To ensure that the derivative nodes from repeated splitting are purer than their antecedents, a measure of impurity $i(n)$ is computed at each node. The node impurity $i(n)$ has the property that it is maximal when all the classes occur in equal proportion and it is minimal when 100% of the node elements belong to one class. Popular among the impurity measures is entropy which is

$$i(n) = -\sum_{i=1}^{k} P(C_i \mid n) \log P(C_i \mid n) \qquad (5.17)$$

where $C_i, i = 1,2,....,k$ are the k classes.

Other measures of impurity often used are the *Gini index of diversity* and *the twoing rule*. The Gini index of impurity of a node *n* is,

$$i(n) = \sum_{i \neq j} P(i \mid n) P(j \mid n) \qquad (5.18)$$

The twoing rule employed at a node *n* chooses the split *s* that maximizes the expression,

$$\frac{P_L P_R}{4} [P(j|n_L) - P(j|n_R)]^2 \qquad (5.19)$$

Notes:

Although there is a plethora of splitting rules bandied around in the literature, the performance of a tree is quite insensitive to choice of the splitting rule. The performance is depended on obtaining the right sized tree. Right sized trees are discussed in the section on pruning trees.

One important question that remains to be addressed is the splitting criterion. If the attribute under consideration is interval-scaled, a candidate set of provisional splits is generated. Each split is an outcome dividing the domain of the attribute x into two regions, $x \le c, and \quad x > c$. Therefore for a set of cut points $c_1, c_2, ..., c_l$, we generate a corresponding list of candidate splits $S_1, S_2, ..., S_l$. In the case the attribute variable is categorical assuming values in the set $V = \{v_1, v_2, ..., v_K\}$, the splits are based on determining if x belongs to any of the possible sub sets generated by *V*.

Section 5.15 The number of candidate splits

It appears as if the number of possible candidate splits on an interval-scaled variable x is theoretically infinite. It is in fact far less than that. Given an attribute x, the number of distinct values of that variable is equal to *r*, say. The number of candidate

Notes:

splits for the attribute may be generated by taking the cut-points to be half-way between consecutive distinct values of x. For a categorical variable with K levels, the number of candidate splits is equal to $2^{K-1}-1$. The decision tree methodology may be viewed as recursive portioning of the data space into rectangles of smaller and smaller sizes obtained by searching through the list of attributes at every node and finding the attribute that provided the best split. Figures 5.12 and 5.13 are a demonstration of that understanding.

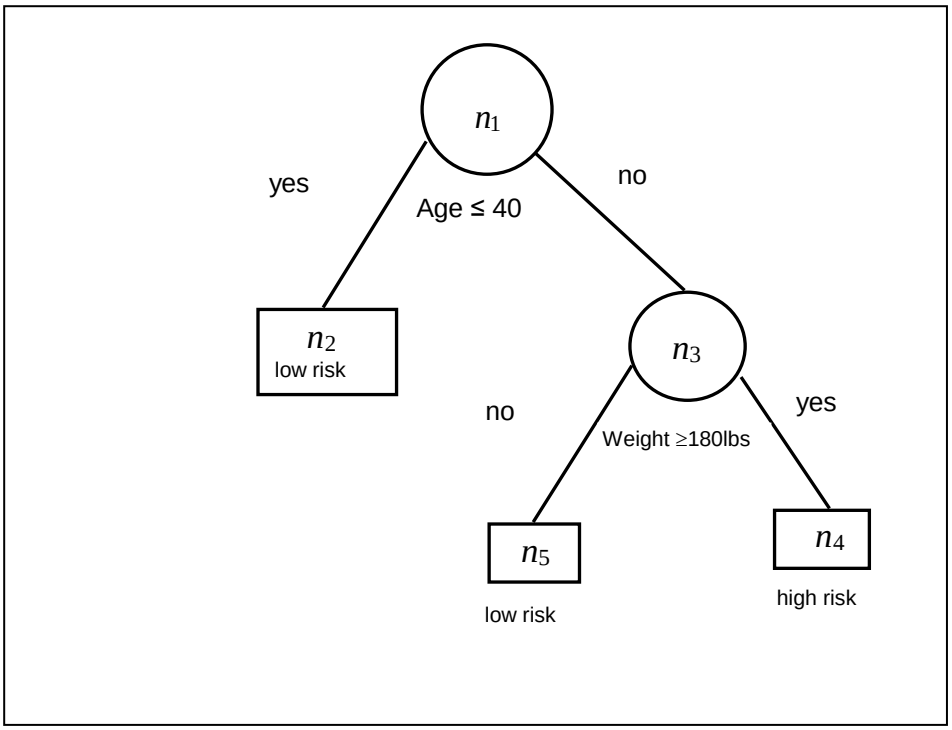

Figure 5.12. Tree due to partitioning a group of candidates into "low-risk" and "high-risk" groups

The partitioning of the data space into rectangular regions is illustrated in Figure 5.13.

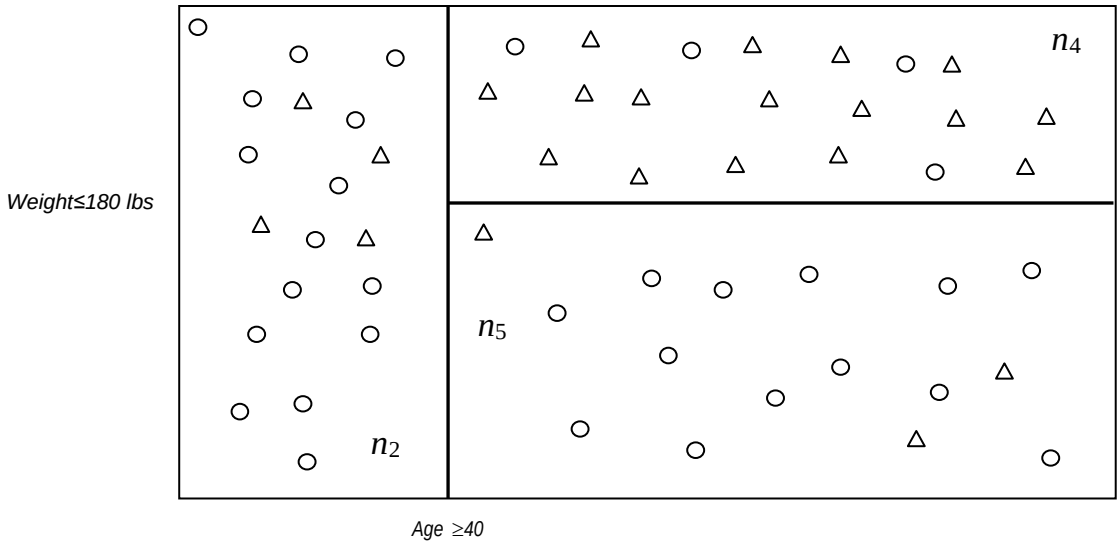

Figure 5.13. Partitioned rectangular sub-regions due to decision tree classification

Decision trees is a recursive partitioning procedure which admits both quantitative and qualitative responses and the computational algorithm which implements it is known as *classification and regression trees* (CART).

Section 5.16 Stopping rule and node to class mapping

In addition, the decision trees need two more steps to complete implementation. A stopping rule that prevents further branching, and a rule for mapping each terminal

Notes:

node to a class.

- *Node to class mapping*

To map each terminal node n_{ter} to one of k classes, a popular decision rule is

$$\max_{j\in\{1,2,\ldots,k\}} P(C_j \mid n_{ter}) \qquad (5.20)$$

where $P(C_j \mid n_{ter})$ is the proportion of observations from the j^{th} class represented in the terminal node n_{ter}. Note that the misclassification rate (MR) is computed using the formula

$$MR(n_{ter}) = \frac{\#\,of\ observations \quad \notin \quad C_j(n_{ter})}{\#\,of\ observations \quad in \quad n_{ter}} \qquad (5.21)$$

Decision tree modeling using classification and regression trees requires the following methodical steps:

- A method to split the tree into sub-nodes and a variable to split.
- A criterion to determine the goodness of a split "*s*" at any given node "*n*".
- A stopping rule to terminate further splitting
- A method to map the terminal nodes into classes based on a criterion such as

Notes:

minimum misclassification rate or equivalently maximum probability of class "j" at terminal node n_{ter}.

While tree building procedures appear to be relatively simple, there are a number of standard issues that one has to contend with such as missing values and disproportionate sample sizes due to splitting. Studies seem to suggest that decision trees are resistant to outliers. The good thing about tree based methods is that not only do they identify presence of outliers they minimize their effect by gathering all of them and segregating into a downstream node during tree building.

Section 5.17 More on missing values

[Breiman et al,1984] affirm that they are two aspects to the missing value problem in tree structured classifiers. One is that values may be missing on some attributes for some sample cases in the training set. The second problem is that one may want to determine the predicted class of an incoming "new" observation based on a fully grown tree. The problems are handled by the use of *surrogate splits*. The construction of a surrogate split simply requires comparing two splits $s_1 and \quad s_2$ at a

Notes:

node *n*. We search for two splits that are similar to one another, but the splitting variables are different. That is to say, while s_1 is split based on the attribute X_k, s_2 is split using attribute X_l. Thus split s_2 is determined to be the best surrogate split of s_1. For instance, if an observation is missing in a measurement on attribute X_k, decide if it goes to the right or left split based the best surrogate.

Another issue that afflicts tree based classifiers is the disproportionate apportioning of samples into left and right nodes. The treatment of this topic is beyond the scope of this textbook. The reader is referred to [Breiman et al, 1984] for details about techniques dealing with this problem. Many researchers argue that *tree pruning* methodologies are more effective than stop splitting rules discussed earlier. A more dependable approach to finding the right tree is to build a large sized tree, prune it back to the parent node and use the misclassification rate criterion to select among the pruned sub-trees. The idea of minimal cost complexity pruning is employed to determine the *right sized tree* for classification and regression modeling.

Section 5.18 Bagging learners

In the traditional approaches to learning, the training data set is used once to construct a learner. Bagging [Breiman et al, 1994] is a modern procedure that uses bootstrap(section 4.4) replications of the training data set to produce an aggregated predicted value. The aggregated prediction is an average of predictions over all the bootstrap replications if the output is numerical. If the predicted output is an

assignment to a class as in a classification problem, the bagged predictor is selected by a *plurality rule*.

Let $\ell = (x_i, y_i), i = 1,2,....., N$ be the training set and let $\ell_{\mathrm{B}}, \mathrm{B} = 1,2,...., K$ be K learning sets generated by bootstrap replications of the original training set ℓ. Recall that each bootstrap replication involves drawing a random sample of N, (x_i, y_i)pairs with replacement. From each of the K samples, a set of K predictors $\hat{f}(x_B), B = 1,2,..., K$ is produced. If the output variable Y is numerical, the bagged predictor of the output is simply given as;

$$\bar{f}(x_B) = \frac{\sum_{B=1}^{K} \hat{f}(x_B)}{K} \qquad (5.22)$$

If the predicted output is a class, then the procedure for bootstrap aggregation is by a plurality rule. Let $\psi_j = \sum_{j=1}^{K} I\left[\hat{f}_j(x_i) = y_i\right]$ and take;

$$\hat{f}(x_i) = \arg\max_j (\psi_j) \qquad (5.23)$$

- *Some comments about bagging predictors*

Notes:

1). Bagging is an effective procedure to use when the original classifier or predictive equation yields dramatically unstable results. An unstable learner is one whose results are erratic due to minor perturbations in the training set.

2). When the original learner produces stable estimates, implementing bagging does not yield significantly improved results.

3). Over some data sets, it may be that the learner produces predictions close to maximum achievable limits. In this case, no amount of bagging helps improve the performance.

Finally bagging is a term derived by combining the terms, “bootstrap” and “aggregating”.

Section 5.19 Final remarks

In this chapter we studied prediction methods involving categorical response variables. Majority of the techniques discussed apply to continuous responses known as predictive models. Chapter seven deals with regression analysis where the response variable y is almost always continuous. Classification methods are very useful in many applications. The area of statistical classification is still evolving

Notes:

and opportunities for further research abound. We tried to be as comprehensive as is possible in our coverage, but we had to leave out some topics due to limitations of time and space. The list of references is quite extensive, and the reader is encouraged to consult those references for further study and research.

Example Data set for Decision trees

Variable	Model Role	Measurement	Description
bad	target	binary	default or seriously delinquent
clage	input	interval	age of oldest trade line in months
clno	input	interval	number of trade (credit) lines
debtinc	input	interval	debt to income ratio
delinq	input	interval	number of delinquent trade lines
derog	input	interval	number of major derogatory reports
job	input	nominal	job category
loan	input	interval	amount of current loan request
mortdue	input	interval	amount due on existing mortgage
ninq	input	interval	number of recent credit inquiries
reason	input	binary	home improvement or debt consolidation
value	input	interval	value of current property
yoj	input	interval	years on current job