2004-04-30

A Study of Several Statistical Methods for Classification with Application to Microbial Source Tracking

Xiao Zhong
Worcester Polytechnic Institute

A Study of Several Statistical Methods for Classificaiton with Application to Microbial Source Tracking

by

Xiao Zhong

A Project Report

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Applied Statistics

by

May 2004

APPROVED:

Jayson D. Wilbur, Advisor

William W. Farr, Associate Department Head

To My Parents

Abstract

With the advent of computers and the information age, vast amounts of data generated in a great deal of science and industry fields require the statisticians to explore further. In particular, statistical and computational problems in biology and medicine have created a new field of bioinformatics, which is attracting more and more statisticians, computer scientists, and biologists.

Several procedures have been developed for tracing the source of fecal pollution in water resources based on certain characteristics of certain microorganisms. Use of this collection of techniques has been termed microbial source tracking (MST). Most of the current methods for MST are based on patterns of either phenotypic or genotypic variation in indicator organisms. Studies also suggested that patterns of genotypic variation might be more reliable due to their less association with environmental factors than those of phenotypic variation. Among the genotypic methods for source tracking, fingerprinting via rep-PCR is most common. Thus, identifying the specific pollution sources in contaminated waters based on rep-PCR fingerprinting techniques, viewed as a classification problem, has become an increasingly popular research topic in bioinformatics.

In the project, several statistical methods for classification were studied, including linear discriminant analysis, quadratic discriminant analysis, logistic regression, and

k-nearest-neighbor rules, neural networks and support vector machine. This project report summaries each of these methods and relevant statistical theory. In addition, an application of these methods to a particular set of MST data is presented and comparisons are made.

Acknowledgements

Frankly, there are many reasons for which I am not willing to leave WPI although I will be able to work in an international company after my graduation. Recently, each time when I thought I am leaving, I felt sad... During the past two years, I have had both wonderful and sorrowful experiences in WPI, from which I learned much knowledge about statistics, mathematics, and life itself, with the priceless help from those kind, patient, generous, professional, and excellent people at WPI, especially in Mathematical Sciences Department.

First and foremost, I would like to express my deep thanks and appreciation to my project advisor, Dr. Jayson Wilbur, for his great help and support with me all the time, especially on this project and job searching. Besides leading me into the field of bioinformatics, he has become my example in many aspects with his essential natures and characters. I also would like to thank Dr. Joseph Petruccelli, Dr. Bogdan Vernescu, Dr. Carlos Morales, Dr. Balgobin Nandram, Dr. Andrew Swift, Dr. Dewon Shon, Dr. John Goulet, Dr. Bill Farr, Dr. Bill Martin, Dr. Homer Walker, and Dr. Luis Roman for your kindness, support, terrific teaching, and cherish memory left me.

Thanks also goes to our department secretaries, Colleen, Ellen, and Debbie. All of them are nice ladies.

In addition, thanks to my colleagues and classmates, Erik, Yan, Flora, Gang, Li, Raj, Didem, Rajesh, ... I will recall the days together with you in TA's office and Straton Hall and pray for your happiness from time to time in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivating Application: Microbial Source Tracking

Although the Clean Water Act (CWA) was enacted in 1972, the water quality of many of the nation's lakes, rivers and streams still do not meet the CWA's goal of "fishable and swimmable waters." A wide array of pollutant classes including temperature (i.e., thermal pollution), sediment, pathogens, nutrients, metals, dissolved oxygen, pH, pesticides and other organic chemicals can result in water quality impairment. Among the numerous ways in which waterways can be damaged, contamination from pathogenic microorganisms is the most serious for waters used for human recreation, drinking water and aquaculture. Moreover, waters contaminated with human feces are generally regarded as posing a greater risk to human health because they are more likely to contain human—specific enteric pathogens than are waters contaminated with animal feces [1].

The Total Maximum Daily Load (TMDL) for each pollutant class in impaired waters has been established to determine the maximum pollutant load that a water body can receive and still meet water quality standards. TMDLs provide the basis for establishing water quality controls and establish waste load allocations among point and non-point pollutant sources. Non-point sources are continuous sources of pollution to water quality, such as agricultural runoff after a rain event or unrestricted access of livestock and wildlife to rivers and streams. Other sources such as sewage treatment plants with a leak problem are considered point sources.

Most methods currently used to monitor microbiological TMDLs in watersheds depend on culturing bacterial indicator organisms such as fecal coliforms, *Escherichia coli* (*E. coli*) or fecal enterococci, because they tend to occur in the same sources as pathogenic organisms (e.g., fecal material), are present in greater densities, and are usually easier to identify than the microbial pollutants. Recently, several procedures for tracing the source of fecal pollution based on certain characteristics of these indicator organisms have been developed. Use of this collection of techniques has been termed microbial source tracking (MST). Most of the current methods for MST are based on patterns of either phenotypic or genotypic variation in indicator organisms.

Phenotypic methods focus on morphological differences between different lineages of bacteria and traits that may have been acquired from exposure to different host species or environments. These methods traditionally target multiple antibiotic resistance (MAR) patterns, cell surface or flagella antigens, or biochemical tests designed to identify variations in the utilization of various substrates that may be found within a particular host environment. However, some studies have suggested that phenotypic methods may be unreliable due to the fact that the organisms adapt to their environment. Thus, it is possible that patterns of phenotypic variation might be associated more with environmental factors than with the pollutant source [2].

Since genotypic profiles may be more stable than phenotypic profiles and are capable of discriminating between different animal sources, DNA-based fingerprinting techniques are increasingly being applied to MST. Particularly, genetic methodologies can be used to differentiate lineages of bacteria found within animal hosts with two assumptions. One is that within a species of bacteria, there are members or subgroups that have become more adapted to a particular host or environment for various reasons, including differences in pH, availability of nutrients, and receptor specificity. The other is that once these organisms become adapted to a particular environment and establish residency, the progeny produced by subsequent replications will be genetically identical. As a result, over time a group of organisms within a particular host or environment should possess a similar or identical genetic fingerprint, which will differ from those organisms adapted to a different host or environment. Specific genotypic methods used include ribotyping, length heterogeneity–PCR (LH–PCR) and terminal-restriction fragment length polymorphism (T–RFLP), repetitive PCR (rep–PCR), denaturing gradient gel electrophoresis (DGGE), pulsed–field gel electrophoresis (PFGE) and amplified fragment length polymorphism (AFLP) [3].

By now, all these approaches have been used with different levels of success in the United States. Most of them have only been tested in a limited number of watersheds, and many require further development before they can be considered appropriate for source tracking of fecal contamination. However, among the phenotypic methods for source tracking antibiotic resistance analysis (ARA) appears to be the most practical approach in small watersheds primarily because it is relatively inexpensive and simple to execute. Among the genotypic methods for source tracking, fingerprinting via rep-PCR is most common.

1.2 Statistical Methodology for Classification

The challenge arising from MST is to identify the specific pollution sources in contaminated waters. This task is viewed as a classification problem with various categorical predictor variables and with the response variable as well as the class variable. Thus, in this section, we introduce statistical methodology for classification to solve the problem.

With the advent of computers and the information age, statistical problems have exploded both in size and complexity. Vast amounts of data generated in many science and industry fields require the statistician to make sense of all. Particularly, challenges in the areas of data storage, organization and searching have led to the new field of data mining. In addition, statistical and computational problems in biology and medicine have created another new field, bioinformatics.

1.2.1 Clustering and Classification

Statistical learning involves extracting important patterns and trends from data for the purpose of understanding what the data say. It plays a critical role in the fields of statistics, data mining and artificial intelligence, intersecting with areas of engineering and other disciplines. Actually, the challenges in learning from data have led to a revolution in the statistical sciences. These problems can be roughly categorized as either clustering (unsupervised) or classification (supervised). Clustering, or grouping, is distinct from classification. Cluster analysis is a more primitive technique since no assumptions are made allowing for the number of groups or the group structure. Its task is to describe the associations and patterns among a set of input measures and there is no measurement of the outcome. This can be done on the basis of similarities

or distances (dissimilarities) and the inputs required are similarity measures or data from which similarities can be computed. However, classification methods focus on a known number of groups, and the operational objective is to predict the value of an outcome measure and assign new observations to one of these groups based on a number of input measures. The whole training stage is guided by the presence of the outcome variable. The following classification examples are extracted from various application fields including biology and medicine:

1. Given some demographic, diet and clinical measurements for a patient with a coronary heart disease, predict whether he or she will have a heart attack in half a year.

2. From some digitized face images, identify a special criminal being captured by police.

3. On the basis of some supermarket sales performance measures and economic data in the passing twenty years, predict the sales potential of it in1 month from now.

4. From the infrared absorption spectrum of a diabetic patient's blood, estimate the amount of glucose in the blood of the patient.

5. identify the criminal in the special database of potential people from multiple features, such as face image, height, weight, accent, left-handed, and so on [4].

Typically, for such problems, we have an outcome measurement, usually quantitative (like sales of a supermarket) or categorical (like heart attack/no heart attack), to predict based on a set of features (like diet and clinical measurements). We also have a set of training data, where we observe the outcome and feature measurements for

a set of objects (such as people). With this data we build a prediction model, or classifier, which will be employed to predict the outcome for new unseen objects. In general, we wish to find an optimal classifier that accurately predicts an outcome.

1.2.2 Discrimination and Classification

As previously mentioned, the problems of learning from data can be roughly divided into two categories: clustering and classification. Sometimes, researchers in this field also propose some ideas based on discrimination (separation) and classification (allocation). In particular, discrimination and classification are multivariate techniques concerned with separating distinct sets of objects or observations and allocating new objects or observations to previously defined groups.

Discriminant analysis is rather exploratory in nature. It is a separation procedure that employs discrimination techniques on a one-time basis in order to investigate observed differences when casual relationships are not well understood. More clearly, the goal of discrimination is to describe, either graphically or algebraically, the differential features of objects or observations from several known populations, and find discriminants functions whose numerical values are such that the populations are separated as much as possible.

Classification procedures are less exploratory in the sense that they ordinarily require more problem structure than discrimination does, and lead to well-defined rules by which new objects can be assigned correctly. Thus, the goal of classification is to sort objects or observations into two or more labeled classes. The emphasis is to derive a rule that can be used for optimally assigning new objects or observations to the labeled classes.

However, in practice, discrimination and classification frequently overlap, and the distinction between them becomes blurred. For example, a function that separates objects may serve as a classifier, and, conversely, a rule that classifies objects may suggest a discriminatory procedure.

1.2.3 Overview of Classification

The examples of classification described in Section 1.2.1 have several common characteristics. For each, there is a set of variables defined as inputs which have some influence on one or more outputs; the objective is to use the inputs to predict the values of the outputs. In the statistical literature, the inputs are often called the predictors, or more classically, the independent variables, while the outputs are called the responses, or more classically, the dependent variables.

In these examples, we have qualitative and quantitative input variables. As a result, the types of methods used for prediction can also be categorized into three classes based upon the types of input variables: quantitative, qualitative, or both. The outputs also vary in nature among these examples in the same way.

Specifically, there are three kinds of outputs for them: quantitative, qualitative, or ordered categorical. For quantitative outputs where some measurements are bigger than others and measurements close in value and similar in nature, regression is conventionally used to denote the prediction procedure. However, for qualitative outputs where there is no explicit ordering in the classes and often descriptive labels rather than numbers are used to denote the classes, classification is conventionally employed for the prediction procedure. The third variable type is ordered categorical, such as small, medium and large, where there is an ordering between the values, but no quantitative measurement is appropriate. In particular, qualitative variables

also referred to as categorical or discrete variables as well as factors are represented numerically by codes, especially when there are only two classes like "survived" or "died", which are represented by a single binary digit or bit as 0 or 1, or else by -1 and 1, sometimes referred to as targets. For more than two classes case, the most useful and commonly used coding is via dummy variables: a p-level qualitative variable is represented by a vector of p binary variables or bits, only one of which is "on" at a time.

Typically, we denote an input variable by the symbol $\mathbf{X}$. If $\mathbf{X}$ is a vector, its components can be accessed by subscripts $\mathbf{X}_i$. Quantitative outputs are denoted by $\mathbf{Y}$, and qualitative outputs by $\mathbf{G}$. The generic aspects of a variable are referred to uppercase letters such as $\mathbf{X}$, $\mathbf{Y}$ or $\mathbf{G}$ and observed values are written in lowercase. For example, the jth observed value of $\mathbf{X}$ is written as x_j, where x_j is again a scalar or vector. Generally, vectors will not be bold except when they have N components, and matrices are represented by bold uppercase letters. For example, a set of N input k–vectors x_j, $j = 1, ..., N$ is represented by the $N \times k$ matrix $\mathbf{X}$. In addition, all vectors are assumed to be column vectors, the jth row of $\mathbf{X}$ is x_j^T, the transpose of x_j.

Now, we can loosely state classification as follows: given an input vector $\mathbf{X}$, make a good prediction of the output $\mathbf{G}$ (or $\mathbf{Y}$), denoted by $\hat{\mathbf{G}}$ (or $\hat{\mathbf{Y}}$), which should take values in the same set $\mathcal{G}$ (or $\mathcal{R}$) associated with $\mathbf{G}$ (or $\mathbf{Y}$). For a two–class $\mathbf{G}$, we can denote the binary coded target as $\mathbf{Y}$, and then treat it as a quantitative output. For example, if $\hat{\mathbf{Y}}$ lies in $[0, 1]$, then we can assign to $\hat{\mathbf{G}}$ the class label according to whether $\hat{y} > 0.5$. This idea may also generalize to K-level qualitative outputs case.

One of the key steps in classification analysis is to construct prediction rules based on training data which are typically denoted by (x_i, y_i) or $(x_i, g_i), i = 1, ..., N$. There are many methods developed for this goal, including linear methods for regression,

linear methods for classification, kernel methods, boosting methods, neural networks, support vector machines, nearest–neighbors, prototype methods, etc. Among them, the linear model fit by least squares and the k-nearest–neighbor prediction rule are simple but powerful.

1.3 Data Description

As mentioned previously, elevated fecal coliform levels are found in many watersheds due to sources that include inadequate septic systems, run–off from pastures and manure treated agricultural land, and wildlife. And rep–PCR DNA fingerprinting has been shown effective for identifying sources of fecal contamination by DNA fingerprints generated using the polymerase chain reaction (PCR) and whole *E. coli* cells. The motivation of this method is based on such consideration below: fingerprints from *E. coli* strains isolated from local streams or lakes may be identified by comparison to our fingerprint database of *E. coli* strains isolated from known human and animal sources.

The data analyzed in the project are rep–PCR finger printings with BOX primers (i.e., BOX–PCR fingerprinting). They come from the Nakatsu Lab at the Department of Agronomy of Purdue University. To get the *E. coli*, manure samples were taken from several different animals. Most of the animals were from Indiana, but a few samples from California. Then rep–PCR fingerprinting was done for all the *E. coli*, using the Box primer. It's also referred to as Box–PCR fingerprinting. Specifically, the original data involves 40 bands (variables) and 680 samples. The corresponding categories, pollution sources, to theses samples are: pig, cow, human, chicken, turkey, dog, deer, quail, raccoon, carnivore, rabbit, coyote, chipmunk, squirrel, rat, duck,

goose, cat, and sewage. For the convenience of making an efficient comparison among those methods described in the project, the data eventually analyzed are achieved by eliminating the samples corresponding to minor categories. The 441 samples involved in the data belong to 5 categories: chicken (58), cow (151), human (33), pig (62), and sewage (137).

In order to get a better idea about the data, several plots are created based on them. One is the histogram plot of sample means for each of the 40 bands (Figure 1.1). The plot displays a bimodal distribution indicating that most bands are either present in most samples or absent in most samples, with relatively few bands being present in between 20 and 80% of samples. A second plot (Figure 1.2) displays the correlation structure. It denotes the correlation coefficients between each pair of the 40 bands with different darkness levels at points—dark corresponds to -1 and light corresponds to 1. The plot shows that there is no special correlation structure in the data. The last plot is a multidimensional scaling (MDS) plot (Figure 1.3) representing 40-dimensional data in 2-dimensional space, in which 5 hosts: chicken (58), cow (151), human (33), pig (62), and sewage (137). are indicated by the 5 colors: black, red, green, blue, and purple, respectively.

After describing the statistical methods for classification in Chapter 2 and Chapter 3, we apply them to these data and make a comparison of the classification results in Chapter 4.

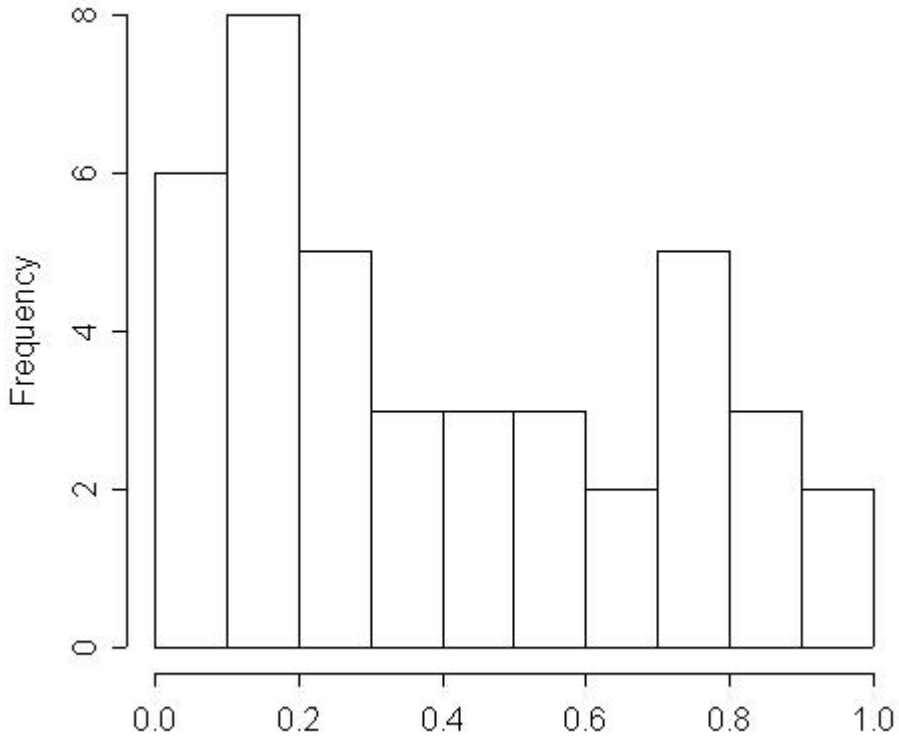

Figure 1.1: Histogram of the proportion of samples, each of the forty bands is present

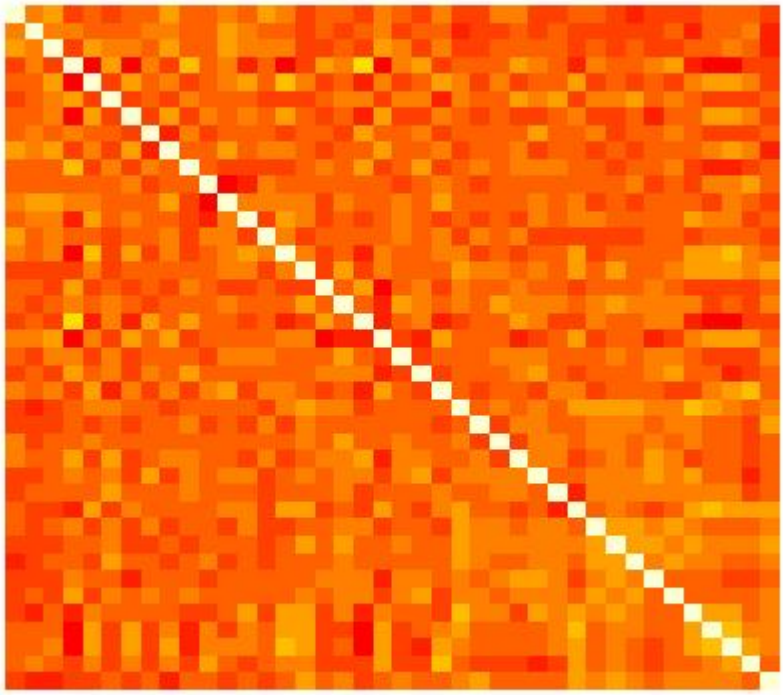

Figure 1.2: Plot of correlation structure

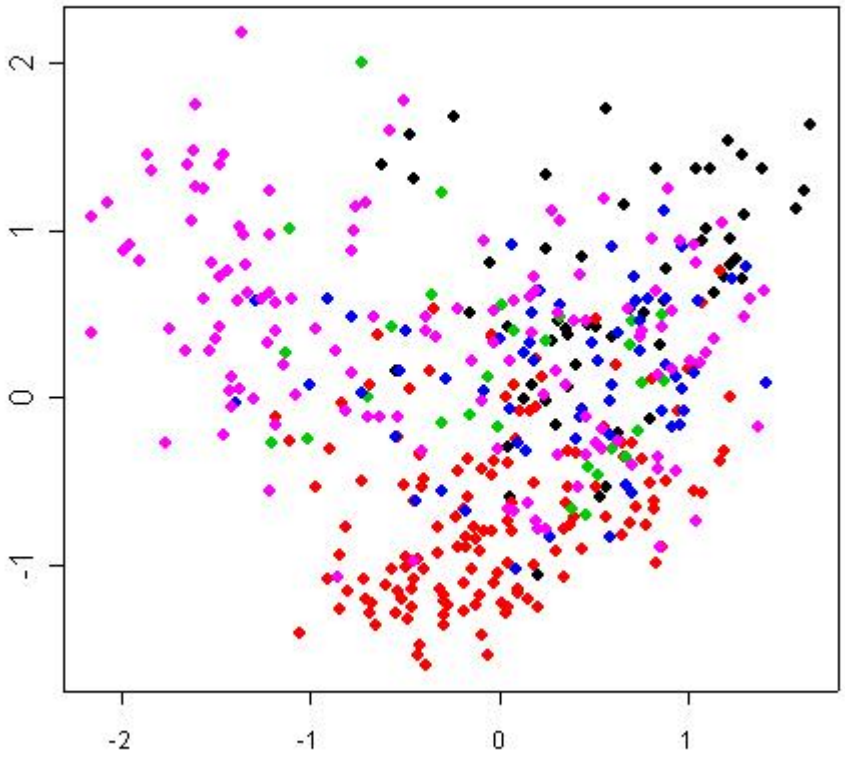

Figure 1.3: MDS plot (Euclidean distance)

Chapter 2

Classical Statistical Methods for Classification

2.1 Overview

Several statistical models for prediction and classification have been developed and applied in many areas of science, finance and industry. Among them are linear models and k–nearest–neighbor methods. Since these are two quite different kinds of simple, but powerful, classical prediction methods, we discuss them in detail in this chapter.

2.2 Linear Models

In Chapter 1, it's easy to find that the predictor $G(x)$ takes values in a discrete space $\mathcal{G}$, thus we can divide it into a set of regions labeled according to the classes. If the boundaries of these regions are linear, the corresponding classification methods

are called linear methods. In general, there are several different ways in which linear decision boundaries can be obtained, including linear regression, logistic regression, and linear discriminant analysis.

2.2.1 Linear Regression

Given a vector of inputs $X = (X_1, X_2, ..., X_p)$, we predict the output Y via the model:

$$\hat{\mathbf{Y}} = \hat{\beta}_0 + \sum_{i=1}^{p} X_i \hat{\beta}_i.$$

For convenience of notation, include the constant variable $X_0 = 1$ in $\mathbf{X} = (X_0, X_1, ..., X_p)$, then the linear model can be written in vector form as

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}$$

where $\hat{\mathbf{Y}}$ is a M-vector, and $\hat{\beta}$ is a $M \times (p+1)$ matrix of coefficients. Then in the $(p+1)$-dimensional input-output space, $(\mathbf{X}, \hat{\mathbf{Y}})$ represents a hyperplane and the function $f(\mathbf{X}) = \mathbf{X}\hat{\beta}$ is linear.

Among many different methods of fitting the linear model to a set of training data, the most popular is the method of least squares: we try to minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i\beta)^2$$

to get the corresponding coefficient β. After differentiating with respect to β, we can obtain the unique solution

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y},$$

and the fitted value at the jth input x_j is $\hat{y}_j = \hat{f}(x_j)$. Now, we can fit the lin-

ear regression models to the class indicator variables and classify to the largest fit. For example, the decision boundary between class i and j is that set of points for which $\hat{y}_i(x) = \hat{y}_j(x)$, that is, the set $x : (\hat{\beta}_{i0} - \hat{\beta}_{j0}) + (\hat{\beta}_i - \hat{\beta}_j)^T x = 0$, an affine set or hyperplane. We classify x to the class with the largest value for its discriminant function.

2.2.2 Linear Discriminant Analysis

Assume $f_m(x)$ is the group-conditional density function of $\mathbf{X}$ in group $\mathbf{G} = m$, and π_m is the prior probability of group m with $\sum_{m=1}^{M} \pi_m = 1$. Then Bayes theorem can be applied to obtain the group posteriors $Pr(\mathbf{G}|\mathbf{X})$, which correspond to the optimal classification results with the lowest error rates among all classification techniques:

$$Pr(\mathbf{G} = m|\mathbf{X} = x) = \frac{f_m(x)\pi_m}{\sum_{l=1}^{M} f_l(x)\pi_l}.$$

Linear discriminant analysis (LDA) is the best classification method for Normal-distributed data

$$f_m(x) = \frac{1}{(2\pi)(p/2)|\Sigma_m|^{1/2}} e^{-1/2(x-\mu_m)^T \Sigma_m^{-1}(x-\mu_m)},$$

it assumes that the groups have a common covariance matrix $\Sigma_m = \Sigma \forall m$. In order to obtain the linear discriminant functions, or the decision boundaries between two different groups, we consider the two-group cases first.

By computing the log-ratio of two group posteriors, we see that

$$\begin{aligned} log\frac{Pr(\mathbf{G}=k|\mathbf{X}=x)}{Pr(\mathbf{G}=l|\mathbf{X}=x)} &= \log\frac{f_k(x)}{f_l(x)}+\log\frac{\pi_k}{\pi_l} \\ &= \log\frac{\pi_k}{\pi_l}-1/2(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l) \\ &\quad +x^T\Sigma^{-1}(\mu_k-\mu_l). \end{aligned}$$

Since the assumptions are the two groups have a common covariance matrix and data have a Normal distribution, it's easy for us to get the equation above that is linear in x and indicates the decision boundary between group k and l is linear in x (a hyperplane in high dimensional space). The conclusion can be generalized for any pair of groups, which means that the decision boundaries among different groups are all linear and correspond to the linear discriminant functions with the following format:

$$\delta_k(x)=x^T\Sigma^{-1}\mu_k-1/2\mu_k^T\Sigma^{-1}\mu_k+\log\pi_k \tag{2.1}$$

and

$$G(x)=\arg\max_k\delta_k(x)$$

Where three parameters π_k, μ_k, and Σ can be estimated by using the training data with a Normal distribution: $\hat{\pi}_k=N_k/N$, N_k is the number of group-k observations;

$$\begin{aligned} \hat{\mu}_k &= \sum_{g_i=k}x_i/N_k; \\ \hat{\Sigma} &= \sum_{k=1}^{K}\sum_{g_i=k}(x_i-\mu_k)(x_i-\mu_k)^T/(N-K). \end{aligned}$$

Where g_i denotes the group of observation i.

2.2.3 Quadratic Discriminant Analysis

Quadratic discriminant analysis (QDA) arises in the classification cases with fewer assumptions compared with linear discriminant analysis (LDA). If the covariance matrix for each group, Σ, is not the same one, then the convenient cancellations in (2.1) will not occur and the quadratic terms in x will remain. (2.2) may be changed into quadratic discriminant functions (QDA):

$$\delta_k(x) = -1/2 \log |\Sigma_k| - 1/2(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k. \tag{2.2}$$

It's quadratic in x and implies that the decision boundaries between each pair of groups are not linear.

The estimates for parameters in (2.3) are similar to those in (2.2) except that the covariance matrix should be separately estimated for each group. Although the number of parameters needed to be estimated dramatically increases in high dimensional space, both LDA and QDA performs well on various classification tasks. A known and widely cited example as a proof for this is the STATLOG project [5], in which LDA was among the top 3 classifiers for 7 of the 22 datasets, QDA was among the top 3 for 4 datasets, and one of the pair among the top 3 for 10 datasets. Actually, the performances of LDA and QDA are similar, but QDA is the preferred approach with more convenience than LDA. Some statisticians have proposed their explanation for the two simple classification tools' popularity: not because the data are almost Normal or for LDA the covariance matrices are almost equal, but the data can only support simple decision boundaries like linear or quadratic, and the estimates for the parameters in the models are stable. It's a bias variance trade-off: Maybe the liner decision boundary is not appropriate, but the bias can be estimated with much lower variance than more exotic ones. Of course, for QDA, which has many parameters to

be estimated,the explanation is less believable.

As for the computations for LDA and QDA, we can simplify them by diagonalizing $\hat{\Sigma}$ or $\hat{\Sigma}_k$, especially for the latter: first, compute the eigendecomposition for each $\hat{\Sigma}_k = \mathbf{U}_k \mathbf{D}_k \mathbf{U}_k^T$, here $\mathbf{U}_k$ is $p \times p$ orthonormal and $\mathbf{D}_k$ is a diagonal matrix of positive eigenvalues d_{kl}; then the components for $\delta_k(x)$ (2.3) are the following:

$$\begin{aligned} (x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) &= [\mathbf{U}_k^T (x - \mu_k)]^T \mathbf{D}_k^{-1} [\mathbf{U}_k^T (x - \mu_k)] \\ \log |\hat{\Sigma}_k| &= \Sigma_l \log d_{kl}. \end{aligned}$$

Specifically, the LDA classifier may be defined by:

1. Sphere the data according to the common covariance estimate $\hat{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, which means $\mathbf{x}$ can be transformed to $\mathbf{X}^\star$ in the following way:

$$\mathbf{X}^\star \leftarrow \mathbf{D}^{-1/2} \mathbf{U}^T \mathbf{X}.$$

2. Classify to the closest group and modulate the effect of the group prior probabilities π_k.

In summary, the data with Normal distribution and common covariance are optimally be classified by linear decision boundaries, and the procedure can be finished with the steps above. In addition, LDA is the optimal classification approach for data with a Normal distribution. If the data don't have a Normal distribution, then all the linear methods including LDA, linear regression, QDA, separating hyperplane, and logistic regression are very similar except for some specific requirements for applications, for example, LDA assumes a common covariance matrix exists, but QDA doesn't, and

so on. Particularly, for two-class classification problems, LDA has the same power as linear regression methods.

2.2.4 Logistic Regression

As for the logistic regression model, which arises from the desire to model the posterior probabilities of the M classes via linear functions in x, while at the same time ensuring that they sum to one and remain in [0,1]. The form of the model is

$$\begin{aligned}
\log \frac{Pr(\mathbf{G}=1|\mathbf{X}=x)}{Pr(\mathbf{G}=M|\mathbf{X}=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{Pr(\mathbf{G}=2|\mathbf{X}=x)}{Pr(\mathbf{G}=M|\mathbf{X}=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{Pr(\mathbf{G}=M-1|\mathbf{X}=x)}{Pr(\mathbf{G}=M|\mathbf{X}=x)} &= \beta_{(M-1)0} + \beta_{M-1}^T x.
\end{aligned}$$

Simple calculations lead to

$$\begin{aligned}
Pr(\mathbf{G}=m|\mathbf{X}=x) &= \frac{exp(\beta_{m0}+\beta_m^T x)}{1+\sum_{l=1}^{M-1} exp(\beta_{l0}+\beta_l^T x)}, \\
& \qquad\qquad m = 1, \ldots, M-1, \\
Pr(\mathbf{G}=M|\mathbf{X}=x) &= \frac{1}{1+\sum_{l=1}^{M-1} exp(\beta_{l0}+\beta_l^T x)},
\end{aligned}$$

and we can testify they sum to one. Particularly, we denote the probabilities $Pr(\mathbf{G} = m|\mathbf{X} = x) = p_m(x;\theta), \theta = \beta_{10}, \beta_1, \ldots, \beta_{(M-1)0}, \beta_{M-1}$ is the entire parameter set.

In the case of $M = 2$, there is a single linear function. The first two examples given in Chapter 1 (Section 1.2.1) can be solved with the simple model, which is widely employed in biological applications where binary responses, two classes, occur quite

often. It's also the reason why we select it to be one of methods for MST studies.

2.3 Nearest Neighbor Methods

Nearest-neighbor methods use those observations in the training set closest in input space to unlabeled sample x to obtain its category $\hat{Y}$. Among a variety of pattern recognition algorithms, k nearest neighbor classifier (k-NN) is a nonparametric analysis method which has been proved very successful in many fields. It is described in more detail in this section. Consider the two-group case (Human and Nonhuman) of microbial source tracking (MST) described in Chapter 1. For either linear classifiers or quadratic classifiers, in order to discriminate between the two groups a decision function is needed to build in such a way that the error is as small as possible. But for k nearest neighbor, that discrimination problem can be solved in a different way.

2.3.1 Distances for Pairs of Units

Before discussing k nearest neighbor methods in detail, it's necessary to introduce about the measure of "distances" or "closeness" for pairs of units. Generally, the Euclidean distance and the city-block distance metrics depicted below are employed to calculate distances between two p-dimensional units $\mathbf{X}' = [x_1, x_2, \ldots, x_p]$ and $\mathbf{Y}' = [y_1, y_2, \ldots, y_p]$:

(1) Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2} = \sqrt{(\mathbf{x} - \mathbf{y})'(\mathbf{x} - \mathbf{y})}$$

unit	BiologC1	BiologC3	BiologC4	BiologC5	BiologC7
a	1	1	1	1	0
b	1	1	1	1	0
c	1	1	1	1	0
d	1	1	1	1	0
e	1	1	0	1	0
f	1	1	0	1	1

Table 2.1: Carbon utilization data

(2) City-block distance

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} |x_i - y_i|$$

However, sometimes units can not be represented by meaningful p–dimensional measurements, but compared with each other on the basis of the presence or absence of certain features by introducing a binary variable, which assumes the value one if the feature is present and the value zero otherwise. To make the explanation easier, consider small subset of n=6 units possessing p=5 features selected from microbial source tracking (MST) data:

For unit e and f, there are four 1–1 matches, one 0-0 match, and one 0-1 mismatch. Assume x_{ej} be the binary value of the jth binary feature on the unit e and x_{fj} be the binary value of the jth binary feature on the unit $f, j = 1, 2, \ldots, 5$. Then the squared Euclidean distance

$$\sum_{j=1}^{5}(x_{ej} - x_{fj})^2 = (1-1)^2 + (1-1)^2 + (0-0)^2 + (1-1)^2 + (0-1)^2 = 1$$

corresponds to the number of mismatches between unit e and f. In order to use it reasonably to measure the similarity or closeness, several schemes for defining similarity coefficients have been proposed, which specify differential weights of the 1–1

	$1(\text{unit}_f)$	$0(\text{unit}_f)$	Totals
$1(\text{unit}_e)$	a	b	a+b
$0(\text{unit}_e)$	c	d	c+d
Totals	a+c	b+d	p=a+b+c+d

Table 2.2: Contingency table for Carbon utilization measurement

matches and the 0–0 matches. Actually, the frequencies of matches and mismatches for two units can be shown in the following contingency table: In this table, a denotes the frequency of 1–1 matches, b is the frequency of 1–0 matches, c is the frequency of 0–1 matches, and d is the frequency of 0-0 matches. It's easy to find that for unit e and unit f in our sample, $a = 3, b = 0, c = 1, d = 1$. With an application to microbial source tracking (MST), several common similarity coefficients S_{ij} =Similarity Between Objects i and j, defined in terms of the frequencies in the contingency table above are [6]:

1. Dice [7]: $\frac{2a}{2a+b+c}$
2. Jaccard [8]: $\frac{a}{a+b+c}$
3. Matching Coefficient [9]: $\frac{a+d}{p}$
4. Ochiai [10]: $\frac{a}{\sqrt{(a+b)(a+c)}}$
5. Jeffrey's X [11]: $\frac{a}{2}\left(\frac{1}{a+b}+\frac{1}{a+c}\right)$

Where p is the number of binary features. For instance, if the Dice similarity coefficient is employed and the equal weights are given to all the matches, the similarity number for unit e and f can be calculated as:

$$\frac{a+d}{p} = (3+1)/(3+0+1+1) = 4/5.$$

In the same way, the similarity numbers for pairs of units in the sample selected at the beginning of this subsection can be computed and displayed in the 6×6 symmetric matrix

$$\begin{pmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & & \\ 4/5 & 4/5 & 4/5 & 4/5 & 1 & \\ 3/5 & 3/5 & 3/5 & 3/5 & 4/5 & 1 \end{pmatrix}$$

Thus, this matrix indicates such conclusion as unit e is least similar to any other unit and unit a, b, c, and d are the same with each other [6].

2.3.2 K Nearest Neighbor Classifier

Based on the computation of distances for pairs of units above, let's go back the k nearest neighbor classifier (k–NN). Assume each unit in the selected Carbon sample except for unit c has been labeled H (Human) or N (Nonhuman)—unit a, b, and d are labeled H and unit e and f are labeled N—and we need to classify unit c into H or N: What the k nearest neighbor classifier does is to select the k nearest neighbors around the unit c and use them to assign a label to unit c. First, an appropriate k needs to be chosen for the specific problem, which is a tough task because too large (or too small) k may result in non generalizing classifiers. In general, the optimal k can be found by employing the leave–one–out method on the training set with independent test sets for accurate error estimation and comparison of different k nearest neighbor classifiers required. Just for instance, let's assume $k = 3$ for this case. Second, the key idea of k nearest neighbor method is that determining the group of the unlabeled

unit can be done according to a majority voting rule which states that the label to be assigned should be the one that occurs the most among the neighbors. Here, from the 6×6 symmetric matrix of the similarity coefficient, the three nearest neighbors of unit c is unit a, b and d, which are all labeled H, thus unit c is assigned by k nearest neighbor approach into group H, the majority label among its three nearest neighbor.

k–nearest–neighbor classifier [12] takes much time to get final classification results and has some computational considerations. However, it is still a good tool with some improvements, like invariant metrics and tangent distance [13], and adaptive nearest–neighbor selection [14].

2.4 Summary

We have described two techniques for classification in Section 2.2 and 2.3: the stable, but biased, linear models and less stable, but often less biased, class of k–nearest–neighbor rules. It seems that with a reasonably large set of training data we should be able to find a fairly large neighborhood of observations close to any x and average them, thus we could always approximate the theoretically optimal conditional expectation by k–nearest–neighbor averaging. However, our intuition is not correct in high dimensions with the phenomenon commonly referred to as the curse of dimensionality [15]. In particular, the class of nearest–neighbor methods can fail in at least two ways:

1. If the dimension of the input space is high, the nearest neighbors need not be close to the target point, which can result in large errors;

2. If special structure is known to exist, it can be used to reduce both the bias and the variance of the estimates.

These are also the reasons why we anticipate using other classes of models for $f(x)$, specifically designed to overcome the dimensionality problems. In particular, support vector machines methods described in Chapter 3 are developed specifically for this purpose.

Chapter 3

Neural Networks and Support Vector Machine

3.1 Introduction

Chapter 2 describes linear models and k-nearest-neighbor procedures, two simple but important procedures for classification. Many variants of the two methods have been developed separately and have been the most popular techniques used in the areas of statistics, data mining, and artificial intelligence. Neural networks and support vector machines are among them.

3.2 Neural Networks

Neural networks have been shown to compete well with the best learning methods on many problems, and are especially effective in problems where prediction instead

of interpretation is the goal and a high signal-to-noise ratio and settings exist. The central idea of neural networks techniques is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features. That is, neural network models consist of sums of nonlinearly transformed linear models.

3.2.1 Overview

The original idea of neural networks came from psychologists and neurobiologists [14] who tried to explore neurons' computational analogues and the name "neural network" derived from the fact that they were developed as models for the human brain first. It has been developed separately in statistics [16] and artificial intelligence [17] fields based on essentially identical models. Researchers in these fields defined a neural network as a set of connected input/output units where a flexible weight is given to each connection. The weights are repeatedly adjusted by training the neural network, a phase is generally called neural network learning or connectionist learning, so that the input samples can be classified into a correct category. Each unit represents a neuron, and the connections represent synapses.

Particularly, from the views of statisticians, the neural network is a useful tool for nonlinear statistical model. The central idea of neural networks is to extract linear combinations of the inputs as derived features, and then model the outputs as nonlinear functions of these features. As a powerful learning method, neural networks have various applications in many fields. The most important advantage of neural networks is their high tolerance to noisy data and ability to predict new (unlabeled) observations. However, they require long training time and are not suitable for some applications where speed is emphasized. They also require more empirical knowledge

or experiences to determine the network topology or "structure" where lots of parameters, like the number of the layers, the number of units in the input layer, the number of hidden layers, the number of units in each hidden layer, and the number of units in the output layer. The other disadvantage is their poor interpretability for the reason that it's hard for people to explain the symbolic meaning behind the learned weights. Fortunately, several algorithms have developed to extract rules from the neural networks. A graph corresponding to the content above is shown in Figure 3.1.

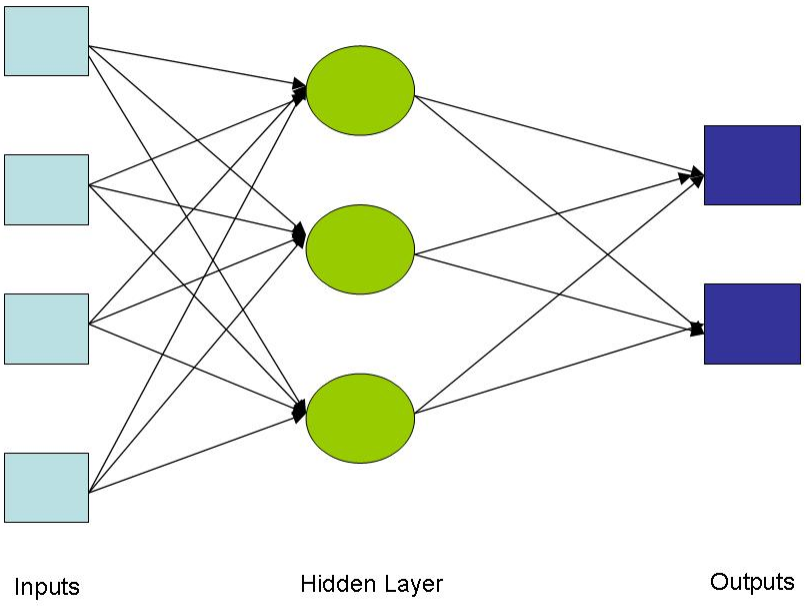

Figure 3.1: An example of multilayer feed-forward neural networks with one hidden layer, seven input units, and three output units

It has been known that normalizing the input values to make them fall between 0.0 and 1.0 will speed up the learning phase. Especially discrete-valued features may be encoded in such way below.

Figure 3.1 An example of multilayer feed-forward neural networks with one hidden layer, seven input units, and three output units

1) If $\mathbf{X} = (x_0, x_1, x_2)$, then three input units will be used to represent $\mathbf{X}$ in the neural networks.

Input Units	x_0	x_1	x_2
I_0	1	0	0
I_1	0	1	0
I_2	0	0	1

2) If $\mathbf{O} = O_1, O_2$, then one output unit will be enough to represent $\mathbf{O}$ in the neural networks.

Output Units	Class I	Class II
O_1	0	1
O_2	1	0

In addition, the initial values of the weights may also affect the resulting accuracy. Once a network has been trained and its accuracy is not considered acceptable. It's common to repeat the training process with different network topology or a different set of initial weights. About the learning phase, the back propagation algorithm proposed in the 1980s is the most popular neural network algorithm, which is performed on multi-layered feed-forward networks. Specifically, the back propagation algorithm can be performed by iteratively processing a set of training samples, and then comparing the network's prediction for each sample with the actual known class label. For each training sample, the weights are modified for the purpose of minimizing the mean squared error between the two class labels. Such modifications are made in the "backwards" direction: from the output layer, through each hidden layer down to the

first hidden layer. In general, although not guaranteed, it is supposed that the weight will converge and the training phase stops at that point.

3.2.2 Vanilla Neural Net

The most widely used neural network model is the "Vanilla" neural net, sometimes called the single hidden layer back–propagation network, or single layer perceptron. It is a two–stage classification model and generally can handle multiple quantitative responses in a seamless fashion. Let's consider K–class classification, then there are K units at the top of the network diagram shown in Figure 3.1 and K target measurements Y_k, $k = 1, \ldots, K$, each being coded as a 0–1 variable for the kth class. And the target Y_k is modeled as a function of linear combinations of the derived features Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K, \\ f_k(X) &= g_k(T), k = 1, \ldots, K, \end{aligned}$$

where $Z = (Z_1, Z_2, \ldots, Z_M)$, and $T = (T_1, T_2, \ldots, T_k)$. Usually, the activation function $\sigma(v)$ is chosen to be the sigmoid $\sigma(v) = 1/(1 + e^{-v})$. In addition, we denote the complete set of unknown parameters in the model, often called weights, by θ:

$$\begin{aligned} \alpha_{0m}, \alpha_m; m &= 1, 2, \ldots, MM(p+1)weights, \\ \beta_{0k}, \beta_k; k &= 1, 2, \ldots, KK(M+1)weights. \end{aligned}$$

and the corresponding classifier is $G(x) = \arg\max_k f_k(x)$. In order to make the model fit the training data well, we use either squared error or cross–entropy (deviance):

$$R(\theta) = -\sigma_{i=1}^{N}\sigma_{k=1}^{K} y_{ik} \log f_k(x_i),$$

to seek values of the weights.

Back–propagation algorithm is the generic approach to minimizing $R(\theta)$. It's performed by a forward and backward sweep over the network, keeping track only of quantities local to each unit—typically it will be stopped before getting the global minimizer of $R(\theta)$ to avoid overfitting. The two–pass procedure has also been called the delta rule [18]. It can be implemented efficiently on a parallel architecture computer because each hidden unit passes and receives information only to and from units that share a connection. As a result, the advantages of back–propagation are its simple and local nature although it can be very slow. Moreover, there are several issues that should be considered in training neural networks, such as starting values, overfitting, scaling of the inputs, number of hidden units and layers, and so on.

3.3 Support Vector Machines

If two classes are linearly separable, the optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class. It can be extended to the non-separable case, where the classes overlap. The techniques employed to determine the optimal separating hyperplanes when two groups are linearly non-separable, the support vector machines (SVMs), are developed by Vapnik [19]. Support vector machines produce nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. They

rely on preprocessing the data to represent patterns in a high dimension-typically much higher than the original feature space. With an appropriate nonlinear mapping function to a sufficiently high dimension, data from two groups can be separated by a hyperplane, that is, the one with the maximum distance from the nearest training patterns, and the support vectors are those nearest patterns with the maximum distance b from the optimal hyperplane. [20]

3.3.1 Support Vector Classifier

The aim of support vector classification is to devise a computationally efficient way of learning good separating hyperplanes in a high dimensional feature space. The good hyperplane can be understood as optimizing the generalization boundaries. Different generalization boundaries and corresponding algorithms include: one can optimize the maximal margin, the margin distribution, the number of support vectors, and so on.

The simplest model of support vector machines, SVMs, is the maximal margin classifier. As a starting point for the analysis and construction of more complex SVMs, although it can not be used in many practical problems—for noisy data, there may be almost no linear separation in the feature space except for overfitting the data by employing very powerful kernels, it is the easiest algorithm to understand and forms the main building block for advanced SVMs. Thus the description for this classifier below is crucial for understanding SVM theory.

Assume the training data consist of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, with $x_i \in \Re^p$ and $y_i \in \{-1, 1\}$. Then a hyperplane can be defined by

$$\{x : f(x) = x^T\beta + \beta_0 = 0, \}$$

where β is a unit vector: $||\beta|| = 1$. And the corresponding classification rule is

$$G(x) = sign[x^T\beta + \beta_0].$$

It can be shown that $f(x)$ above gives the signed distance from x to the hyperplane

$$f(x) = x^T\beta + \beta_0 = 0$$

and there exists a function

$$f(x) = x^T\beta + \beta_0$$

with $y_i f(x_i) > 0 \forall i$. Now the maximal margin classifier problem can be treated as an optimization problem

$$\max_{\beta,\beta_0,||\beta||=1} C \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq C, \text{ for } i = 1, \ldots, N \tag{3.1}$$

Where, the distances between either of the two groups and the hyperplane are both C units. Hence the width of $2C$ units is the margin.

Actually, the usual way of describing the support vector criterion for separable data is

$$\min_{\beta,\beta_0} ||\beta|| \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq 1, \text{for } i = 1, \ldots, N. \tag{3.2}$$

And $C = 1/||\beta||$. For this convex optimization problem with quadratic criterion and linear inequality constraints, there are several ways to solve it.

Now, assume the groups overlap in feature space, then what we can do to deal with such cases is to maximize $||C||$ with allowing for some points to be on the wrong side

of the margin. The constraint in (3.1) can be modified in two ways:

$$y_i(x_i^T\beta + \beta_0) \geq C - \xi_i, \text{ or } y_i(x_i^T\beta + \beta_0) \geq C(1 - \xi_i), \tag{3.3}$$

Where the slack variables $\xi = (\xi_1, \ldots, \xi_n)$, and $\forall i, \xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq$constant. Since the second way results in the standard support vector classifier, we will use it solve our problem. In addition, an explanation for it is given here: The value ξ_i is the proportional amount by which the prediction $f(x_i) = x_i^T\beta + \beta_0$ is on the wrong side of its margin, by bounding the sum of $\xi_i, i = 1, \ldots, N$, the total proportional amount of predictions fall on the wrong side of their margin can also be bounded. For example, bounding $\Sigma\xi_i$ at a constant C means bounding the total number of training misclassifications at C since misclassifications occur when $\xi_i > 1$. Similarly, we define the support vector classifier for non-separable cases in the usual way

$$min||\beta|| + \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq C(1 - \xi_i) \quad \forall i, \\ \xi_i \geq 0, \Sigma\xi_i \leq \text{constant}. \end{cases} \tag{3.4}$$

From the definition, we can find an attractive property of support vector classifiers: the points well inside their group boundaries do not contribute much to the boundaries' building. It is obviously different from linear discriminant analysis (LDA).

3.3.2 Discussion about Support Vector Machine

We have introduced how to find linear boundaries in the input feature space with the support vector classifier. As mentioned at the beginning of this section, with more advanced support vector classifiers, we can make the classification process more flexible by enlarging the feature space using mapping functions such as polynomials,

Gaussians, splines, or other basis expansions. The choice of the mapping functions is often determined by the designers' knowledge of the problem domain, but the rule of thumb here is: with the transformation, data that are not linearly separable in original feature space can be well separated by linear boundaries in the enlarged space. Once the mapping functions $h_m(x), m = 1, \ldots, M$ are selected, we just need to employ the same process as before to fit the support vector classifier with transformed input feature $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i)), i = 1, \ldots, N$, and produce the nonlinear boundary function in the original feature space

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0.$$

Then we can obtain the corresponding classifier, it is

$$\hat{G}(x) = sign(\hat{f}(x)).$$

There is one additional point that should be made. The dimensionality of the mapped feature space can be arbitrarily high, infinite in some cases, which may lead to two disadvantageous situations: on one hand, the computations can become prohibitive by computational resources; on the other hand, with sufficient mapping functions, the data can be separable, but overfitting may occur at the same time. Essentially, what the SVM classifier is solving is a function–fitting problem associated with a particular criterion and regularization form, similar to smoothing splines techniques [21].

Chapter 4

Application to Microbial Source Tracking

4.1 Overview

In this chapter, we apply linear discriminant analysis, k-nearest-neighbor, logistic regression, neural networks and support vector machine approaches to the BOX-PCR data described in Chapter 1. This is done for both a two group and a five group case and we conclude with discussion.

4.2 Two Group Classification

It is easier for statisticians to do two-group classification than three or more groups. On the other hand, in Chapter 1, we mentioned that for MST, waters contaminated with human feces are generally thought as putting greater risk to human health than

Host	Human	Nonhuman	Total	Error rate
Human	14	19	33	0.5758
Nonhuman	106	302	408	0.2598
Total	120	321	441	
Error rate	0.8833	0.0592		0.2834

Table 4.1: Confusion matrix for linear discriminant analysis with equal priors. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	4	29	33	0.8788
Nonhuman	7	401	408	0.0172
Total	11	430	441	
Error rate	0.6364	0.0674		0.0816

Table 4.2: Confusion matrix for linear discriminant analysis with proportional priors. Row labels indicate true host and column labels indicate predicted host.

the other pollutants. As a result, in this section, We only consider about two categories, human and nonhuman, in the data. The classification results corresponding to each method mentioned above are shown with the confusion matrix below. For each of them, row labels denote true host and column labels denote predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	11	22	33	0.6667
Nonhuman	14	394	408	0.0343
Total	25	416	441	
Error rate	0.5600	0.0529		0.0816

Table 4.3: Confusion matrix for 1-nearest-neighbor method based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	1	32	33	0.9697
Nonhuman	1	407	408	0.0025
Total	2	439	441	
Error rate	0.5000	0.0729		0.0748

Table 4.4: Confusion matrix for 5-nearest-neighbor method based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	0	33	33	1
Nonhuman	0	408	408	0
Total	0	441	441	
Error rate		0.0748		0.0748

Table 4.5: Confusion matrix for 10-nearest-neighbor method based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	7	9	16	0.5625
Nonhuman	26	399	425	0.0612
Total	33	408	441	
Error rate				0.0794

Table 4.6: Confusion matrix for logistic regression analysis based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	0	33	33	1
Nonhuman	0	408	408	0
Total	0	441	441	
Error rate		0.0748		0.0748

Table 4.7: Confusion matrix for two group classification by support vector machines based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	27	6	33	0.1818
Nonhuman	0	408	408	0.0000
Total	27	414	441	
Error rate	0.0000	0.0145		0.0136

Table 4.8: Confusion matrix for two group classification by a neural network with 2 nodes. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	31	2	33	0.0606
Nonhuman	0	408	408	0.0000
Total	31 410	441		
Error rate	0.0000	0.0049		0.0045

Table 4.9: Confusion matrix for two group classification by a neural network with 3 nodes. Row labels indicate true host and column labels indicate predicted host.

Host	Human	Nonhuman	Total	Error rate
Human	32	1	33	0.0303
Nonhuman	0	408	408	0.0000
Total	32	409	441	
Error rate	0.0000	0.0024		0.0023

Table 4.10: Confusion matrix for two group classification by a neural network with 4 nodes. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	33	3	8	9	5	58	0.4310
Cow	2	107	12	24	6	151	0.2914
Human	3	7	15	3	5	33	0.5455
Pig	5	6	12	33	6	62	0.4677
Sewage	7	12	25	13	80	137	0.4161
Total	50	135	72	82	102	441	
Error rate	0.3400	0.2074	0.7917	0.5976	0.2157		0.3923

Table 4.11: Confusion matrix for linear discriminant analysis based on hold-one-out cross-validation with equal priors. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	34	4	3	9	8	58	0.4138
Cow	1	127	5	10	8	151	0.1589
Human	6	8	8	2	9	33	0.7576
Pig	5	17	4	29	7	62	0.5323
Sewage	6	20	8	10	93	137	0.3212
Total	52	176	28	60	125	441	
Error rate	0.3462	0.2784	0.7143	0.5167	0.2560		0.3401

Table 4.12: Confusion matrix for linear discriminant analysis based on hold-one-out cross-validation with priors proportional to host representation in training data. Row labels indicate true host and column labels indicate predicted host.

4.3 Five Group Classification

Since in the data described in Chapter 1, we totally have five specific categories, not only human and nonhuman. In this section, we consider about the classification of these five hosts: chicken, cow, human, pig and sewage, with the same approaches as Section 4.2. The classification results corresponding to each method are also shown by the confusion matrix below. For each of them, row labels denote true host and column labels denote predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	36	8	3	6	5	58	0.3793
Cow	7	124	4	8	8	151	0.1788
Human	5	5	12	1	10	33	0.6364
Pig	3	17	5	28	9	62	0.5484
Sewage	7	17	11	16	86	137	0.3723
Total	58	171	35	59	118	441	
Error rate	0.3793	0.2749	0.6571	0.5254	0.2712		0.3500

Table 4.13: Confusion matrix for 1-nearest-neighbor method based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	31	11	0	8	8	58	0.4655
Cow	0	133	2	8	8	151	0.1192
Human	2	14	2	3	12	33	0.9394
Pig	4	25	2	23	8	62	0.6290
Sewage	8	28	7	10	84	137	0.3869
Total	45	211	13	52	120	441	
Error rate	0.3111	0.3697	0.8462	0.5577	0.3000		0.3800

Table 4.14: Confusion matrix for 5-nearest-neighbor method based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	27	12	1	6	12	58	0.5345
Cow	0	136	0	7	8	151	0.0993
Human	1	18	1	2	11	33	0.9697
Pig	5	29	0	21	7	62	0.6613
Sewage	4	34	3	9	87	137	0.3650
Total	37	229	5	45	125	441	
Error rate	0.2703	0.4061	0.8000	0.5333	0.3040		0.3800

Table 4.15: Confusion matrix for k-nearest-neighbor method based on hold-one-out cross-validation with k equaling 10. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	34	8	4	4	5	55	0.5000
Cow	4	121	13	17	17	172	0.0397
Human	2	3	6	5	3	19	1.0000
Pig	8	8	2	25	9	52	0.7258
Sewage	10	11	8	11	103	143	0.3504
Total	58	151	33	62	137	441	
Error rate	0.414	0.1987	0.8182	0.5968	0.2482		0.3447

Table 4.16: Confusion matrix for logistic regression analysis based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	29	7	0	9	13	58	0.5000
Cow	1	127	2	5	16	151	0.1589
Human	3	10	5	1	14	33	0.8485
Pig	2	17	0	32	11	62	0.4839
Sewage	2	20	2	5	108	137	0.2117
Total	37	181	9	52	162	441	
Error rate	0.2162	0.2983	0.4444	0.3846	0.3333		0.3200

Table 4.17: Confusion matrix for five group classification by support vector machines based on hold-one-out cross-validation. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	55	1	0	1	1	58	0.0172
Cow	12	134	0	1	4	151	0.0000
Human	14	0	0	12	7	33	0.0303
Pig	35	1	0	22	4	62	0.0000
Sewage	11	7	0	6	113	137	0.0073
Total	127	143	0	42	129	441	
Error rate	0.0000	0.0070	0.0000	0.0000	0.0078		0.0023

Table 4.18: Confusion matrix for five group classification by neural network with 2 nodes. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	50	2	0	5	1	58	0.1379
Cow	0	146	1	2	2	151	0.0331
Human	14	0	3	8	8	33	0.9091
Pig	1	8	0	43	10	62	0.3065
Sewage	1	4	0	28	104	137	0.2409
Total	66	160	4	86	125	441	
Error rate	0.2424	0.0875	0.2500	0.5000	0.1680		0.2154

Table 4.19: Confusion matrix for five group classification by neural network with 3 nodes. Row labels indicate true host and column labels indicate predicted host.

Host	Chicken	Cow	Human	Pig	Sewage	Total	Error rate
Chicken	52	4	0	1	1	58	0.1034
Cow	0	146	3	1	1	151	0.0331
Human	3	3	14	5	8	33	0.5758
Pig	1	8	2	49	2	62	0.2097
Sewage	1	2	3	3	128	137	0.0657
Total	57	163	22	59	140	441	
Error rate	0.0877	0.1043	0.3636	0.1695	0.0857		0.1179

Table 4.20: Confusion matrix for five group classification by neural network with 4 nodes. Row labels indicate true host and column labels indicate predicted host.

4.3.1 Discussion

For this particular data, none of the classifiers evaluated via cross-validation was clearly superior. The one method that was not evaluated via cross-validation was the neural network. Therefore, results in those tables should be viewed with suspicion.

With additional time, this project could have pursued various other topics and detailed analyses. In particular, detailed simulation studies and theoretical comparisons between methods would be valuable. In addition, classifiers which specifically target data structures commonly found in microbial source tracking data could be explored.

Appendix: R code

R (version 1.9.0) was used extensively in this project to perform calculations and generate plots. *R* is a language and environment for statistical computing and graphics, which is available as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. *R* can be downloaded at: `http://www.r-project.org`.

R can be extended via packages available through the Comprehensive *R* Archive Network (CRAN) family of Internet sites covering a very wide range of modern statistics. The following is a list of the CRAN packages used in this project.

CRAN Package	Statistical methods
class	k-nearest neighbor classification
e1071	Support vector machines
MASS	Linear discriminant analysis and Quadratic discriminant analysis
nnet	Neural networks
stats	Multidimensional scaling

The remaining pages in this appendix document the *R* code used in this project.

```
#####################################################
# Reading the data into R and defining variables #
#####################################################

MST<-read.csv("BNecoli.csv")

# n=441 by p=40 binary matrix of MST fingerprint data
x<-MST[,-c(1,2)]

# Class variable with five levels:
# (Chicken, Cow, Human, Pig and Sewage)
cl<-MST[,2]
cl<-as.factor(cl)

# Class variable with two levels: Human and Nonhuman
cl2<-array("Nonhuman",dim=length(cl))
cl2[cl=="Human          "]<-"Human"
cl2<-as.factor(cl2)


################
# Data summary #
################

# Histogram of variable means (proportions)
jpeg(filename = "hist.jpg", width = 480, height = 480,
pointsize = 12, quality = 75, bg = "white")
hist(apply(x,2,mean),main="",xlab="")
dev.off()

# Plot of correlation structure
jpeg(filename = "corr.jpg", width = 480, height = 480,
pointsize = 12, quality = 75, bg = "white")
image((cor(x))[40:1,],axes=FALSE)
dev.off()
```

```
# Multidimensional scaling plot (Euclidean distance)
colorvar<-array(6,dim=length(cl))
colorvar[cl=="Chicken"]<-1
colorvar[cl=="Cow"]<-2
colorvar[cl=="Human          "]<-3
colorvar[cl=="Pig"]<-4
mdsdata<-cmdscale(dist(x),k=2)

jpeg(filename = "mds.jpg", width = 480, height = 480,
pointsize = 12, quality = 75, bg = "white")
plot(mdsdata,col=colorvar,pch=16,xlab="",ylab="")
dev.off()




#######################################
# Linear discriminant analysis (LDA) #
#######################################

## Two-group case

z<-lda(x, cl2, prior=rep(0.5,2), CV=TRUE) # LDA with equal priors
z<-lda(x, cl2, CV= TRUE)    # LDA with proportional priors

# Confusion matrix and error rate computation
q<-table(cl2,z$class)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)
```

```
## Five-group case
z<-lda(x, cl, prior=rep(0.2,5), CV=TRUE)  # LDA with equal priors
z<-lda(x, cl, CV= TRUE)    # LDA with proportional priors

# Confusion matrix and error rate computation
q<-table(cl,z$class)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)
sum(errors)/sum(Total)

########################################
# Nearest Neighbor Classification Rules #
########################################

## Two-group case
z<-knn.cv(x, cl2, k=1)  #  1-NN classification rule
z<-knn.cv(x, cl2, k=5)  #  5-NN classification rule
z<-knn.cv(x, cl2, k=10) # 10-NN classification rule

# Confusion matrix and error rate computation
q<-table(cl,z)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)
```

```
## Five-group case

z<-knn.cv(x, cl, k=1)  #  1-NN classification rule
z<-knn.cv(x, cl, k=5)  #  5-NN classification rule
z<-knn.cv(x, cl, k=10) # 10-NN classification rule

# Confusion matrix and error rate computation
q<-table(cl,z)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)


####################################
# Logistic discriminant functions #
####################################

## Two-group case

x<-as.matrix(x)
phat<-array(dim=length(cl2))
for(i in 1:length(cl2)){
i<-1
y<-class.ind(cl2)[-i,1]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat[i]<-exp(yhat)/(1+exp(yhat))
}
```

```
## Five-group case
phat1<-array(dim=length(cl))
for(i in 1:length(cl)){
y<-class.ind(cl)[-i,1]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat1[i]<-exp(yhat)/(1+exp(yhat))
}

phat2<-array(dim=length(cl))
for(i in 1:length(cl)){
y<-class.ind(cl)[-i,2]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat2[i]<-exp(yhat)/(1+exp(yhat))
}

phat3<-array(dim=length(cl))
for(i in 1:length(cl)){
y<-class.ind(cl)[-i,3]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat3[i]<-exp(yhat)/(1+exp(yhat))
}

phat4<-array(dim=length(cl))
for(i in 1:length(cl)){
y<-class.ind(cl)[-i,4]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat4[i]<-exp(yhat)/(1+exp(yhat))
}
```

```
phat5<-array(dim=length(cl))
for(i in 1:length(cl)){
y<-class.ind(cl)[-i,5]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
temp<-glm(y~.,data=w,family="binomial")
yhat<-sum(temp$coeff*c(1,x[i,]))
phat5[i]<-exp(yhat)/(1+exp(yhat))
}

# Confusion matrix and error rate computation
phat<-cbind(phat1,phat2,phat3,phat4,phat5)
maxphat<-apply(phat,1,max)
maxphat5<-cbind(maxphat,maxphat,maxphat,maxphat,maxphat)
y<-class.ind(cl)
q<-t(maxphat5==phat)%*%y
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)

#################################
# Support vector machines (SVM) #
#################################

# Reinitialize data after logistic discrimination

x<-MST[,-c(1,2)]
cl<-as.factor(MST[,2])
cl2<-array("Nonhuman",dim=length(cl))
cl2[cl=="Human          "]<-"Human"
cl2<-as.factor(cl2)
```

```
## Two-group case

Ghat<-cl2
for(i in 1:length(cl2)){
y<-cl2[-i]
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
attach(w)
temp<-svm(y~.,data=w)
Ghat[i]<-predict(temp,as.data.frame(x[i,]))
}

# Confusion matrix and error rate computation
q<-table(cl2,Ghat)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)


## Five-group case

Ghat<-cl
for(i in 1:length(cl)){
#i<-1
y<-cl[-i]
#xalt<-x
xalt<-x[-i,]
w<-as.data.frame(cbind(y,xalt))
attach(w)
temp<-svm(y~.,data=w)
Ghat[i]<-predict(temp,as.data.frame(x[i,]))
}
```

```
# Confusion matrix and error rate computation
q<-table(cl,Ghat)
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)

####################
# Neural networks #
####################

## Two-group case
targets <- class.ind(cl2)
mst.nnet <- nnet(x, targets, size=2, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)
mst.nnet <- nnet(x, targets, size=3, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)
mst.nnet <- nnet(x, targets, size=4, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)

# Confusion matrix and error rate computation
q<-table(cl2,max.col(predict(mst.nnet, x)))
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)
sum(errors)/sum(Total)
```

```
## Five-group case

targets <- class.ind(cl)

mst.nnet <- nnet(x, targets, size=2, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)
mst.nnet <- nnet(x, targets, size=3, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)
mst.nnet <- nnet(x, targets, size=4, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)
mst.nnet <- nnet(x, targets, size=5, rang=0.5, decay=5e-4,
abstol=5e-200, maxit=10000)

# Confusion matrix and error rate computation
q<-table(cl,max.col(predict(mst.nnet, x)))
errors<-q-diag(diag(q))
Total<-apply(q,2,sum)
colerrors<-apply(errors,2,sum)
rowtotal<-apply(q,1,sum)
rowerrors<-apply(errors,1,sum)
q<-cbind(q,rowtotal,round(rowerrors/rowtotal,4))
q
Total
round(colerrors/Total,4)
sum(Total)
sum(errors)/sum(Total)
```

Bibliography

[1] J. M. Simpson, J. W. Santo Domingo, and D. J. Reasoner. Microbial source tracking: State of the science. *Environmental Science and Technology*, 36:5279–5288, 2002.

[2] D. M. Gordon, Bauer S., and Johnson J. R. The genetic structure of Escherichia coli populations in primary and secondary habitats. *Microbiology*, 148:1513–1522, 2002.

[3] T. M. Scott, J. B. Rose, T. M. Jenkins, S. R. Farrah, and J. Lukasik. Microbial source tracking: Current methodology and future directions. *Applied and Environmental Microbiology*, 68:5796–5803, 2002.

[4] X. Zhong. *Classification and Clustering Mining*. PhD thesis, Zhejiang University, Hangzhou, China, 2000.

[5] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[6] R. A. Johnson and Wichern D. W. *Applied Multivariate Statistical Analysis*. Prentice-Hall, 5th edition edition, 2002.

[7] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:297–302, 1945.

[8] P. Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques regions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

[9] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 38:1409–1438, 1958.

[10] A. Ochiai. Zoogeographical studies on the soleoid fishes found in japan and its neighboring regions ii. *Bulletin of the Japanese Society of Scientific Fisheries*, 22(9):526–530, 1957.

[11] S. Kulczynski. Die pflanzenassoziationen der pieninen. *Bull. Intern. Acad. Pol. Sci. Lett. Cl. Sci. Math. Nat., Ser. B*, 2:57–203, 1928.

[12] E. Fix and J. L. Hodges. Discriminatory analysis - nonparametric discrimination: consistency properties. Technical Report 21-49-004, United States Air Force School of Aviation Medicine, Randolph Field, Texas, 1951.

[13] P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 50–58. Morgan Kaufmann, 1993.

[14] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[15] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition*. John Wiley & Sons, 2001.

[17] V. Honavar and L. Uhr. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. Academic Press, 1994.

[18] B. Widrow and M. Hoff. Adaptive switching curcuits. In *Western Electronic Show and Convention (WESCON) Convention Record*, volume 4, pages 96–104. Institute of Radio Engineers, 1960.

[19] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[21] X. Zhong. Nonparametric regression analysis—spline smoothing. Class project for MA 542 (Applied Regresion Analysis) at Worcester Polytechnic Institute in Fall 2003, 2003.