Statistical Methods for High Throughput Screening Drug Discovery Data

by

Yuanyuan (Marcia) Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Statistics

Waterloo, Ontario, June 2005

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Abstract

High Throughput Screening (HTS) is used in drug discovery to screen large numbers of compounds against a biological target. Data on activity against the target are collected for a representative sample of compounds selected from a large library. The goal of drug discovery is to relate the activity of a compound to its chemical structure, which is quantified by various explanatory variables, and hence to identify further active compounds. Often, this application has a very unbalanced class distribution, with a rare active class.

Classification methods are commonly proposed as solutions to this problem. However, regarding drug discovery, researchers are more interested in ranking compounds by predicted activity than in the classification itself. This feature makes my approach distinct from common classification techniques.

In this thesis, two AIDS data sets from the National Cancer Institute (NCI) are mainly used. Local methods, namely K-nearest neighbours (KNN) and classification and regression trees (CART), perform very well on these data in comparison with linear/logistic regression, neural networks, and Multivariate Adaptive Regression Splines (MARS) models, which assume more smoothness. One reason for the superiority of local methods is the local behaviour of the data. Indeed, I argue that conventional classification criteria such as misclassification rate or deviance tend to select too small a tree or too large a value of k (the number of nearest neighbours). A more local model (bigger tree or smaller k) gives a better performance in terms of drug discovery.

Because off-the-shelf KNN works relatively well, this thesis takes this promising method and makes several novel modifications, which further improve its performance. The choice of k is optimized for each test point to be predicted. The empirically observed superiority

of allowing k to vary is investigated. The nature of the problem, ranking of objects rather than estimating the probability of activity, enables the k-varying algorithm to stand out. Similarly, KNN combined with a kernel weight function (weighted KNN) is proposed and demonstrated to be superior to the regular KNN method.

High dimensionality of the explanatory variables is known to cause problems for KNN and many other classifiers. I propose a novel method (subset KNN) of averaging across multiple classifiers based on building classifiers on subspaces (subsets of variables). It improves the performance of KNN for HTS data. When applied to CART, it also performs as well as or even better than the popular methods of bagging and boosting. Part of this improvement is due to the discovery that classifiers based on irrelevant subspaces (unimportant explanatory variables) do little damage when averaged with good classifiers based on relevant subspaces (important variables). This result is particular to the ranking of objects rather than estimating the probability of activity. A theoretical justification is proposed. The thesis also suggests diagnostics for identifying important subsets of variables and hence further reducing the impact of the curse of dimensionality.

In order to have a broader evaluation of these methods, subset KNN and weighted KNN are applied to three other data sets: the NCI AIDS data with Constitutional descriptors, Mutagenicity data with BCUT descriptors and Mutagenicity data with Constitutional descriptors. The k-varying algorithm as a method for unbalanced data is also applied to NCI AIDS data with Constitutional descriptors. As a baseline, the performance of KNN on such data sets is reported. Although different methods are best for the different data sets, some of the proposed methods are always amongst the best.

Finally, methods are described for estimating activity rates and error rates in HTS data. By combining auxiliary information about repeat tests of the same compound, likelihood

methods can extract interesting information about the magnitudes of the measurement errors made in the assay process. These estimates can be used to assess model performance, which sheds new light on how various models handle the large random or systematic assay errors often present in HTS data.

Acknowledgements

First, I would like to express my deepest gratitude to both of my supervisors: Dr. Hugh A. Chipman and Dr. William J. Welch. They have provided me with their thoughtful guidance, remarkable insights, and continuous encouragement. Without them, this thesis could not have become a reality.

Second, I wish to thank Dr S. Stanley Young for highlighting many interesting problems and drawing a big picture of drug discovery for me. I always gain a lot of knowledge by working with him. I also wish to thank Dr. R.W. Oldford, one of my committee members, for the time he dedicated to reviewing my thesis, and for his encouragement and helpful suggestions.

Next, thank my supervisor, Dr. Laurent Briollais, for his generosity and his support. I highly value them as they helped me through the last year of my thesis, the toughest year I have ever had. Thank you, Laurent!

I greatly appreciate my dear family for their steadfast support and unconditional help. They are always there to take care of me and motivate me.

My special thanks also go to many special individuals:

Amy for her endless encouragement and continuous confidence in me.

Wenyu and her Husband, Eric, for introducing me to this wonderful profession and offering me a lot of help.

Marc, my previous officemate, for creating a very friendly environment for the first three years of my PHD and always offering me strong support when it was needed.

My teammates: Hui Shen, Yan Yuan, Xu Wang, Longyang Wu, and Xianlin Ma for their helpful collaboration, understanding, and encouragement.

The faculty, students, and staff in the department of Statistics and Actuarial Science for providing a supportive and stimulating environment.

Last, but definitely not least, I would like to dedicate this thesis to my son, Kaiwen. He gave me the strongest support through this journey. He makes me brave, mature, and independent. He has accompanied me in completing this chapter of my life, and he will accompany me in the new one. I am so lucky to have him. *God bless us!*

Contents

List of Tables

List of Figures

Abbreviation

AHR	Average Hit Rate
CART	Classification and Regression Trees
HTS	High Throughput Screening
KNN	K-nearest Neighbours
MARS	Multivariate Adaptive Regression Splines
MART	Multiple Additive Regression Trees
MSE	Mean Squared Error
NCI	National Cancer Institute
SAR	Structure-activity Relationship

Chapter 1

Drug Selection and Data Mining

1.1 Drug research

Tremendous progress has been made in the field of drug research and development (R&D) (e.g. Levy 2000). The development of computer-based technology, and advances in human genetics, robotics, and miniaturization have spurred dramatic changes in drug R&D. Drug discovery is much faster than before. Modern drug discovery is not based exclusively on chemistry. It is the fruit of integrated interdisciplinary knowledge from biologists, chemists, engineers, and statisticians.

Target selection and drug selection are two important related phases of drug R&D. A target is often defined as a disease-linked protein which plays a fundamental role in the onset or progression of a disease. Due to advances in gene sequencing and the increasing availability of high-throughput methods for studying genes, the proteins they encode, and the pathways in which they are involved, huge volumes of data are collected to examine various genomic effects. The number of available biological targets is being vastly expanded

because of the large genomic variability. For a fixed biological target, the drug selection process aims to identify a compound or a few compounds with the desired properties for further consideration. Two data sets described in Section 2.3, for instance, concern the compounds' potency against the HIV-1 virus.

Novel technologies are revolutionizing drug selection. Combinatorial chemical synthesis is used to build up massive libraries of millions of compounds as a drug-candidate base. High Throughput Screening (HTS) is a process to screen large libraries often composed of hundreds of thousands of compounds (drug candidates) at a rate that may exceed 20,000 compounds per week. The HTS technology is the result of automation and miniaturization of the assay process: Robots are used to automatically select compounds from libraries, combine them with a biological target, and measure outcomes related to activity. This technique makes it possible to assay a large number of potential effectors of biological activity against targets and thus accelerate drug discovery. However, screening a huge collection of compounds against every available biological target is still time consuming, expensive, and thus no longer feasible with rapidly increasing compound collections (Service 1996).

One promising approach is to study the structure-activity relationship (SAR) of the compounds. From the screened data, we can find out which types of small molecules, shapes, and chemical attributes tend to have the desired effect on a target. The explanatory variables used in such models are chemical descriptors, which describe the chemical structure of a compound. Descriptor variables are computed, not physically measured, and hence are easy and extremely cheap to generate even for enormous chemical libraries. Modelling the relationship between the descriptors and activities of sampled compounds provides insight into the SAR, gives predictions for untested compounds of interest, and guides future screening. The SAR for secondary assays can identify the best drug candidates for future toxicity and

clinical experiments. Statistical modelling makes the drug selection process more efficient and productive. Drug selection is now an active learning process: make, test, fail, and try again.

SAR data sets from HTS are usually very large due to the numbers of both compounds and descriptors. The analysis of such large data sets is often called data mining. In this thesis, I first use several methods commonly used in data mining to model two typical HTS data sets and then focus on two local methods (trees and K-nearest neighbours (KNN)), which turn out to be superior, modify them and combine them with other techniques to further improve predictive performance.

1.2 Overview of data mining

With increasing automation and advances in information technology, vast amounts of data are being collected and stored by computer. The great need to discover useful patterns from a huge data set cannot always be met by classical statistical methods. Moreover, improvements in computer speed and capacity now make feasible statistical techniques which ten years ago could not have been considered. Therefore, data mining has become a very important research direction in recent years. It can be applied to many fields such as industrial statistics, marketing, biometrics, and so on. Identifying and developing data mining tools for drug discovery is the first step of our approach.

1.2.1 Definition of data mining

Data mining is a "nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in the data" (Fayyad, Piatesky-Shapiro, and Smyth 1996).

In a broad sense, data mining is a sequential process including problem definition, data preparation, running mining algorithms, and understanding and interpreting results. In a narrow sense, data mining refers to a particular phase of the whole process — using algorithms for mining valuable patterns. For example, the book of Hastie, Tibshirani and Friedman (2001) has this focus.

Data mining tasks can be divided into supervised learning, such as prediction or classification and un-supervised learning, such as clustering. Here, I am interested in supervised learning since the goal is to predict activity using molecular descriptors. Section 1.2.2 reviews some basic supervised learning tools in data mining.

1.2.2 Overview of some basic tools

Linear Regression

Linear regression is simple but basic to many statistical models. It predicts a numeric response by a linear combination of predictors. For observation i, let y_i be the value of the response variable in the data and let $x_{i1}, \ldots, x_{ip}$ be the corresponding predictor variables. A statistical model for the data generation process is

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i \qquad (i = 1, \ldots, n),$$

where the β_j's are unknown parameters and the ϵ_i's are random errors. The least squares estimates, $\hat{\beta}_j$'s, are calculated by minimizing the function

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}))^2.$$

On the other hand, suppose the ϵ_i's are independent normal random variables with mean

0 and variance σ^2. Maximizing the log-likelihood function

$$l(\beta_0, \beta_1, \ldots, \beta_p, \sigma^2) = c - n \log \sigma - \sum_{i=1}^{n} (y_i - (\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}))^2 / 2\sigma^2$$

leads to the maximum likelihood estimation of β, which is identical to the least squares estimates in this instance.

Generalized Linear Model (GLM)

The generalized linear model is a generalization of the linear regression model. A linear model specifies the linear relationship between the expected value of the dependent (or response) variable Y, and a set of predictor variables, the x_j's. However, there are many relationships that cannot be summarized by a simple linear equation. For example, the expected value of a binary response (0/1) is bounded, while a simple linear regression is unbounded. GLMs can be used to model a response variable that has a discrete distribution, and the mean function can be related to the predictors by a nonlinear link function. For example, they can model binary or more generally binomial data in the form of a logistic regression model; GLMs also can model count data using a log-linear model. Here, I outline logistic regression since it is a popular model widely used in classification problems. The simple logistic regression model assumes Y_i takes values 0 or 1 and has a Bernoulli distribution with parameter p_i; i.e. $P(Y_i = 1) = p_i$, and $P(Y_i = 0) = 1 - p_i$. It is expressed as

$$\log \left(\frac{p_i}{1 - p_i} \right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \qquad i = 1, 2, \cdots, n.$$

Assuming all the observations are independent, The estimates of the $\hat{\beta}$'s are obtained by maximizing the log-likelihood function

$$l(\beta_0, \beta_1, \ldots, \beta_p) = c + \sum_{i=1}^{n} (y_i \log p_i + (1 - y_i) \log(1 - p_i)) \tag{1.1}$$

Closed-form expressions for the maximum likelihood estimates are intractable. Iterative methods such as the Newton-Raphson algorithm are used to obtain estimates numerically.

Generalized Additive Model (GAM)

In both logistic regression and linear regression, the linear function of each predictor can be replaced by a smooth function of that predictor, yielding a generalized additive model (Hastie and Tibshirani 1990). The GAM function in the S programming language is used in this thesis. By default, the smooth function for each variable has four degrees of freedom which is defined as

$$df = tr(S) - 1 \tag{1.2}$$

where S is the implicit smoother matrix. Values for df should be greater than 1, with 1 implying a linear fit.

Multivariate Adaptive Regression Splines (MARS)

MARS is an adaptive procedure for regression, originally proposed by Friedman (1991). For regression data, it assumes

$$Y_i = g_0 + \sum_{j_1} g_{j_1}(x_{i,j_1}) + \sum_{j_1<j_2} g_{j_1,j_2}(x_{i,j_1}, x_{i,j_2}) + \epsilon_i \tag{1.3}$$

The functions $g_{j_1}(x_{i,j_1})$ in (1.3) are defined by

$$g_{j_1}(x_{i,j_1}) = b_{0,j_1} x_{i,j_1} + \sum_{k=1}^{K_{j_1}} b_{k,j_1}(x_{i,j_1} - t_{k,j_1})_+ \tag{1.4}$$

and the functions $g_{j_1,j_2}(x_{i,j_1}, x_{i,j_2})$ are a sum of products of two basis functions:

$$g_{j_1,j_2}(x_{i,j_1}, x_{i,j_2}) = \sum_{k_1,k_2} b_{k_1,k_2,j_1,j_2}(x_{i,j_1} - t_{k_1,j_1})_+(x_{i,j_2} - t_{k_2,j_2})_+$$

where the b_{0,j_1}'s, b_{k,j_1}'s and b_{j_1,j_2}'s are unknown parameters. The basis functions, $(x_{i,j_1} - t_{k,j_1})_+$, are defined as

$$(x_{i,j_1} - t_{k,j_1})_+ = \begin{cases} x_{i,j_1} - t_{k,j_1} & \text{if } x_{i,j_1} > t_{k,j_1} \\ 0 & \text{elsewhere} \end{cases} \tag{1.5}$$

The basis functions g_{j_1}, g_{j_1,j_2} and the locations of the knots t_k are selected by a stepwise selection procedure. The sums in (1.3) are over all variable indexes suggested by stepwise selection. Some variables may be excluded from the sums. A modified form of the cross-validation criterion originally proposed in Craven and Wahba (1979) is used to select the final model and it is defined as the residual sum of squares penalized by the size of the model:

$$GCV = \frac{\frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2}{(1 - \frac{cM}{n})^2}. \tag{1.6}$$

Here, M is the number of basis functions in (1.3) and c is a user-specified parameter. Larger values of c will give models with fewer basis functions. With the chosen basis functions, the estimates of the b_{0,j_1}'s, b_{k,j_1}'s and b_{j_1,j_2}'s are determined by the least squares criterion.

For classification, Kooperberg, Bose, and Stone (1997) have developed POLYMARS. Considering the response variable Y_i ranging over class labels $\kappa = \{1, ..., K\}$ and $P(Y_i = k) = p_{ik}$, they assume

$$\log(\frac{p_{ik}}{p_{iK}}) = g_0 + \sum_{j_1 k} g_{j_1 k}(x_{i,j_1 k}) + \sum_{j_1 k < j_2 k} g_{j_1 k, j_2 k}(x_{i,j_1 k}, x_{i,j_2 k}) + \epsilon_{ik}, \qquad k \in \kappa.$$

In order to reduce the intensive computation, a least squares approximation is applied in stepwise selection. Let Z be a response matrix such that its elements z_{ij} satisfy

$$z_{ij} = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{elsewhere} \end{cases} \qquad i = 1, 2, \cdots, n \qquad j = 1, 2, \cdots, K.$$

Similar to (1.6), the classification-based GCV is defined as

$$GCV = \frac{\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}[z_{ij} - \widehat{z_{ij}}]^2}{(1 - \frac{cM}{n})^2}$$

where $\widehat{z_{ij}}$ is an estimate of z_{ij} from the model.

Once a model has been selected, it is re-estimated via maximum likelihood. In the thesis, the R implementation of POLYMARS with the default options is applied.

Neural Networks

A neural network (NN) provides a way to approximate a general non-linear function. The motivation of NN goes back to Werbos (1974) and Parker (1985). A feed-forward NN with one hidden layer is the simplest but most common form (e.g. Ripley 1993 and 1996). The expectation of response variable, Y_i, is a function of the input information, x_{ij}'s (Venables and Ripley 1999):

$$\mathrm{E}(Y_i|x) = \phi_0(\alpha + \sum_h \omega_h \phi_h(\alpha_h + \sum_j \omega_{hj} x_{ij})).$$

Usually, the function ϕ_h in the hidden layers is a logistic function defined as

$$\phi(x) = \frac{\exp(x)}{1 + \exp(x)}$$

and ϕ_0 is a linear or logistic function. The weights α, ω_h, α_h, and ω_{hj} are to be estimated from data usually by nonlinear optimization techniques. NN models often have very good predictions but are uninterpretable. Some experts argue that NN can be seen as a benchmark of data analysis. That is, in practice, our final models should provide better predictions than NN and be interpretable as well.

If the response is categorical, an appropriate classification network is constructed. For

example, for binary data, ϕ_0 is taken to be the logistic function

$$\phi(x) = \frac{\exp(x)}{1 + \exp(x)}.$$

The estimate of the weights is obtained by maximizing the log-likelihood function.

Specifically, in this thesis a feed-forward neural network with one hidden layer (9 hidden units) is implemented (Section 2.4). The decay is chosen to be 0.0005 for the categorical data and 0.001 for the continuous data.

Tree Models

Decision trees date back to the early 1960s. Some early applications in social science can be found in Morgan and Sonquist (1963). Hawkins et al. (1982) and Breiman et al. (1984) made statisticians familiar with decision trees; in machine learning Quinlan (1993) developed the C4.5 algorithm. As a powerful data analysis method, trees are applied to many fields, such as credit scoring and medical diagnosis. New tree-based methods are being developed and have many implementations, such as classification and regression trees (CART) and C4.5. The tree I describe here is the S implementation, based on the work of Clark and Pregibon (1992).

Through binary recursive partitioning, a tree successively splits the data along the coordinate axes of the predictors such that at each division, the resulting two subsets of data are as homogeneous as possible with respect to the response of interest. At each step in the construction algorithm, an optimal split is identified. This local optimality does not guarantee that the globally optimal tree will be found. The subsets which are not further split are terminal nodes.

A classification tree models the response with a multinomial distribution. In the case of

binary data, this simplifies to a binomial distribution. The deviance is defined to be minus twice the log-likelihood; for a node η, this is

$$D(\eta) = -2(a_\eta \log(p_\eta) + (n_\eta - a_\eta) \log(1 - p_\eta))$$

where n_η indicates the number of training cases residing in this node. Of them, a_η cases are in the $Y = 1$ category. The probability of $Y = 1$, p_η, is estimated by a_η/n_η.

The deviance for a node η of a regression tree is calculated according to the normal distribution:

$$D(\eta) = \sum_{i=1}^{n_\eta} (y_i - \widehat{y}_i)^2 \tag{1.7}$$

where n_η is the same as in the classification case; $\widehat{y}_i$ is an average over the training cases residing in η. The deviance of the tree is the summation of the deviances over all the terminal nodes.

In the thesis, the `tree` implementation in S (Clark and Pregibon 1992), which is similar to CART (Breiman et al. 1984), is applied. For categorical data, classification trees are used; regression trees are built for continuous data. The deviance is used as a splitting criterion. Pruning the default tree is examined in Section 2.5.

C4.5

C4.5 is a popular classification method. Like CART, it uses recursive partitioning to generate a decision tree from a set of data. However, the key idea of C4.5 is to generate a rule set from the resulting tree. The rule-set generation process includes:

- Write a rule for each path in the decision tree from the root to a leaf.

- Simplify each rule by removing conditions that are not helpful for discriminating the corresponding class from other classes.
- Sift all the simplified rules for each class by removing the rules that do not contribute to the overall accuracy.
- Order the sets of rules to minimize false positive errors, and choose a default class.

K-Nearest Neighbours (KNN)

KNN is a very simple but powerful non-parametric method (see Dasarathy, 1990 for a review). It estimates the class probability or response value directly. The algorithm is as follows: for each observation in the test set, the k nearest points in the training set based on some proximity measure are found. The prediction is made by either the majority vote among the selected closest points (classification) or the mean response of the selected closest points (regression). Euclidean distance is a standard proximity measure. The parameter k stands for the number of neighbours examined and can be determined by leave-one-out cross-validation on the training data.

1.3 Structure of the thesis

Chapter 2 describes the motivating problems of drug discovery, then compares the performance of a few classification methods on a binary-response data set and corresponding methods for continuous data. Further investigation of the most successful approaches, namely trees and KNN, leads to an understanding of the special features of HTS data. Chapter 3 describes graphical and numerical performance criteria relevant to the objectives of drug

discovery data. These measures are used to build and assess models in later chapters. Chapter 4 uses some new approaches to modify KNN. The techniques include varying k in KNN and combining the KNN method with kernel weights, i.e. weighting the probability estimation by distance. In Chapter 5, I propose the novel method of averaging the classifiers built on different subspaces (subsets of variables) and then compare it to bagging and boosting. Chapter 6 presents further comparisons of k-varying, weighted KNN, and subset KNN methods on the NCI AIDS data with a second descriptor set, and on a toxicity data set with two different descriptor sets. Estimating activity rates and error rates for assessing model performance in Chapter 7 sheds light on how various models handle large random or systematic errors in drug screening data. A summary and suggestions for further research are given in Chapter 8.

Chapter 2

Mining HTS Data

2.1 Introduction

Scientists in biology and chemistry believe that the biological activity of a compound is a consequence of its chemical structure (e.g. Livingstone 1995, Section 1.2; King et al. 1992; Klopman 1984). Here, chemical structure is taken to mean all characteristics that uniquely define a molecule, including atoms in the molecule, bonds between atoms, and the three-dimensional configuration of the atoms. A drug-like molecule is a small three dimensional object and scientists have developed various numerical descriptors to characterize it. Examples include molecule weight and octanol-water partition coefficient ($\log P$). Furthermore, the size, atomic number, charge and the degree of connectivity between two heavy atoms, etc. are interesting properties of a molecule and they play an important role to determine a molecule's biological activity. Methods for uncovering the structure-activity relationship (SAR) have already been applied to the drug discovery process successfully.

This chapter first introduces characteristics of SAR for drug discovery data and reviews

some methods which have applied to those data (in Section 2.2). Section 2.3 then describes a couple of data sets I have used in this thesis. Section 2.4 gives more details about the modelling methods I am going to use in the thesis and compares their performance according to a criterion specific to the efficiency of the drug discovery process. Sections 2.5 and 2.6 investigate the most successful methods (trees and KNN) further and argue that typical goodness-of-fit criteria may be inappropriate for HTS data. Section 2.7 summarizes the findings.

2.2 Overview of SAR modelling for drug discovery data

Advances in biotechnology such as newly developed synthesis methods and better assay techniques make it possible to screen tens of thousands to hundreds of thousands of compounds at the early stages of drug discovery. In the main applications of this chapter, for instance, two measures of biological activity in protecting human cells from HIV infection were assayed for two databases of about 30,000 compounds each. To find the most promising drug candidates, biochemists would like to examine as many compounds as possible. However, it is expensive to test the huge number of compounds potentially available. Research pharmaceutical companies now have up to two million compounds in their databases, and combinatorial chemistry (Service 1996) can potentially generate similar numbers of new compounds. There is a great need to optimize the testing of compounds against targets. One important approach is to use the data from assayed compounds to relate biological activity (the response) to molecular descriptors of chemical structure (explanatory variables). Uncovering the SAR

helps biologists and chemists make decisions on which compounds are most likely to be highly active, so that they can speed up the searching process (e.g. Jones-Hertzog et al. 1999).

The structural properties of a molecule are often represented as a set of numerical values in drug discovery. There are many sets of descriptors based on different approaches to molecular structure-activity relationships, as well as for similarity analysis and high-throughput screening of molecule databases. For instance, twenty sets of descriptors can be computed by DRAGON, a software for the calculation of molecular descriptors (**http://www.talete.mi.it/dragon.htm**). There is no agreement on how to choose the descriptor set, a topic of on-going research (e.g. Feng et al. 2003).

In my thesis, two descriptor sets are used (for details see Section 2.2.1): BCUT numbers (throughout the thesis) and constitutional descriptors (in Chapter 6). The former is a class of molecular descriptors defined as eigenvalues of a modified connectivity matrix and the latter includes measures of the "constitution" of a compound (basic composition, e.g. the number of carbons, the number of oxygens, etc). These chemical descriptors were selected for the following reasons:

- They are relatively small sets of descriptors.
- They have been applied to SAR modeling successfully (e.g, Lam 2001, Feng et al. 2003).

The next section describes BCUT numbers and Constitutional descriptors in more details.

2.2.1 Chemical descriptors used in the thesis

1. BCUT numbers

The BCUT numbers I use in the thesis are six continuous variables. Three person(s) or research groups have contributed to their evolution. Burden (1989) originally suggested constructing a matrix to represent the hydrogen-suppressed connection table of the molecule. To build this connectivity matrix, the atomic numbers were put on the diagonal and values describing bond-type of each pair of atoms are put on the off-diagonal. The two lowest eigenvalue of this matrix were used as chemical descriptors of the molecule. Burden's seemingly far-fetched ideas were successfully confirmed by Rusinko and Lipkus (A. Rusinko III and A. H. Lipkus, unpublished result obtained at Chemical Abstract Service, Columbus OH) in 1993. They found structure searches based on Burden's suggestion were surprisingly comparable to the results of accepted similarity searching procedures. Pearlman and Smith (1998) built on this success by proposing constructing three classes of matrices: one class with atomic charge-related values on the diagonal, a second class with atomic polarizability-related values on the diagonal, and a third class with H-bond-abilities on the diagonal. Also they put a variety of additional information on the off-diagonal including functions of inter-atomic distance, overlaps, computed bond-orders, etc. In addition to the lowest eigenvalues (as Burden suggested) Pearlman and Smith also used the largest eigenvalues. Weighting factors for the diagonal to off-diagonal elements were also used. Over fifty BCUT numbers were derived. I have used six relatively uncorrelated BCUTs from the set.

The advantage of BCUT numbers over other descriptors is their low dimensionality, which allows many statistical methods to run. However, they are not easy to interpret in terms of chemical structure.

2. Constitutional descriptors

The Constitutional descriptors are the measurements of the 'constitution' of a compound (see Appendix A for a list of variable descriptions). The 47 descriptors include information such as molecular weight, atomic weight, atomic counts, etc. Basically, they depend essentially only on the atoms in a molecule but not specifically on the connections between the atoms in the molecule. In comparison, BCUT numbers are determined by the connectivity. For instance, the molecular weight of a compound, the first variable listed in Appendix A, is the sum of the atomic weight of all the atoms that make up one molecule of the compound without looking at their connectivity. Generally speaking, larger molecules will have higher values for all of these descriptors. Many different molecules might have the same values of constitutional descriptors, as there are many ways the same atoms can be connected to make a valid molecule.

Constitutional descriptors are quite attractive due to their simplicity and interpretability. However, most chemists would not consider them very discriminating in prediction of a complex phenomenon like the binding of a small molecule to a protein.

2.2.2 SAR models

Empirical modelling of SARs poses many challenges. First, although the data generated by HTS may have an enormous number of tested compounds, active compounds are often rare. Second, the compound structure is complicated and it is hard to quantify it. Third, SAR data inevitably involve threshold and nonlinear effects when the chemical structure is represented by a set of descriptors. Fourth, large random or systematic measurement errors may be present. Fifth, the screening process itself may produce chemical databases with strong local clustering in the descriptor space. Previous screens aimed at other biological

responses may have led to synthesis and hence the availability of compounds in concentrated regions of high activity. These regions may or may not be relevant to the current screening.

As an open research area, SAR modelling attracts many statistical methods and techniques, which include simple methods such as linear regression (e.g. Livingstone, 1995, chapter 6) and more advanced methods like NN, Partial Least Squares (PLS), recursive partition (RP, e.g. trees). For example, Feng et al. (2003) built NN, PLS and RP using four different sets of chemical descriptors and compared their performance. It is well accepted that more complicated methods (e.g. NN, PLS and RP) should have more prediction power than linear regression to model a SAR. For instance, Young and Hawkins (1998) applied a recursive partition procedure, FIRM, to a large, structure-activity data set and showed that different mechanisms of the data can be discovered.

Due to the difficulty of data and variety of the problems, SAR modelling continues to excite people to find a better solution (methods). For instance, Lam (2001) provides some novel approaches of design and analysis of large chemical data sets by looking at the compounds' SAR. Chapter 3 of Lam's thesis introduces a uniform coverage design aiming at identifying the clusters of active compounds with diverse chemical structures. A cell-based analysis method is proposed in Chapter 4 to find those clusters of active compounds.

Next, I am going to investigate our data and pursue a good interpretation of data.

2.3 The data and objectives

2.3.1 NCI AIDS data

For illustration, I use two data sets on AIDS anti-viral screening from the National Cancer Institute (NCI) chemical data base. Each data set includes about 30,000 compounds. One of them has an ordinal response measuring how a compound protects human CEM cells from HIV-1 infection; the other has a continuous response recording the concentration of the compound that provides 50% protection of infected cells. The two studies investigate different biological targets, but most compounds (over 75%) are common to the two data sets. The compound activities are in the public domain (`http://dtp.nci.nih.gov/docs/aids/aids_data.html`). The database is updated periodically; I am working on the 1999 release.

Six chemical descriptor variables were generated by GlaxoSmithKline chemists. They are continuous variables called BCUT numbers (Burden 1989, Pearlman and Smith 1998) which are determined by the connectivity , i.e., topological relations between different atoms within the molecule (details please see Section 2.2.1).

Pearlman and Smith (1999) showed that it may be possible to find fairly low-dimensional (2-D or 3-D) subsets of BCUT variables such that active compounds are clustered in the relevant subspace. Thus, for modelling the relationship between activity and structure, BCUT values are good candidates as explanatory variables.

Ordinal response data

For the ordinal response data, the activity measure for each compound has three levels: 0 (inactive), 1 (moderately active), and 2 (active). Below is a quantitative description of these

categories.

2 (Active): A compound provided at least 50% protection of cells from HIV-1 infection on the first test and provided 100% protection of cells from HIV-1 infection on the second test.

1 (Moderately active): A compound provided at least 50% protection of cells from HIV-1 infection on the first test and provided at least 50% but less than 100% protection of cells from HIV-1 infection on the second test.

0 (Inactive): All other cases.

The data are very unbalanced: 215 compounds are active, 393 are moderately active and the rest (29,204) are inactive. Figure 2.1 plots BCUT3 and BCUT4, two typical variables for NCI AIDS data. Figure 2.1(a) displays all the active compounds. Octagons and triangles indicate active and moderately active compounds, respectively. Figure 2.1(b) has the same scale and displays all inactive compounds. Both distributions are very complex. The active compounds are located in regions with many inactive compounds. Similar patterns are seen for any pair of descriptors. Thus, no obvious distinct clusters of active regions are apparent by looking at the descriptors two at a time.

However, there are also some promising signals. The density plots in Figure 2.2 compare the distributions of active (straight line), moderately active (dashed line), and inactive compounds (dotted line) in BCUT space. The default bandwidth is used and determined by $\log_2(length(x)) + 1$. For visual consideration, the points with extreme values of the BCUT numbers (the upper 1% and the lower 1%) are not included in each of the 6 pictures. The plots reveal that the highly active compounds have BCUT distributions that differ from

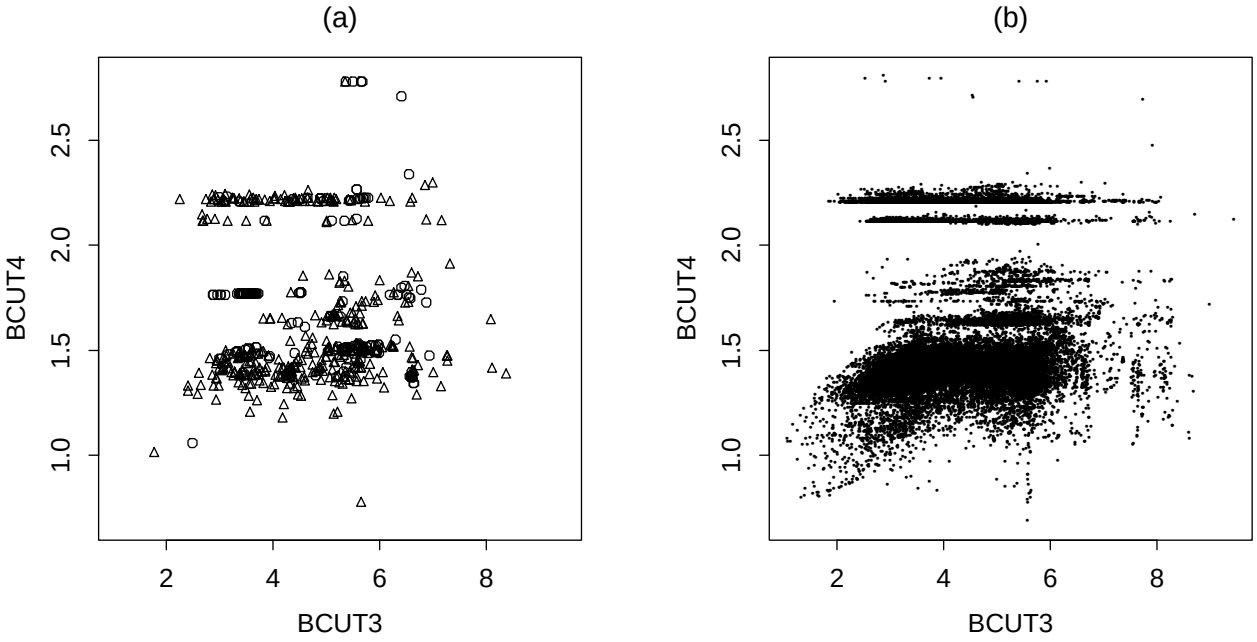

Figure 2.1: Plot of BCUT3 and BCUT4 values: (a) active compounds, with octagons and triangles indicating active and moderately active compounds, respectively, and (b) inactive compounds.

these for the inactive compounds; these differences are much smaller when comparing the moderately active and inactive compounds.

One issue with the densities in Figure 2.2 is the very much larger sample size of inactive compounds. To explore this, Figure 2.3 uses a sample of inactives. In each picture of Figure 2.3, a random sample of 215 inactive compounds is used to produce the density estimate. The density for the inactive compounds does not change much based on a much smaller sample and similar features on each BCUT predictor present.

As some of the methods I investigate are intended for a binary response, and there are relatively few compounds in the two active categories, I combine the active and moderately active cases into one group. The resultant data set has 98% inactive and 2% active (608) compounds.

Continuous response data

There are 28596 compounds in total. The response is defined as the negative logarithm of the compound concentration that protects infected cells by 50% ($-\log(EC50)$). In this setting, larger values indicate more potent compounds. The plots in Figure 2.4 are grey-scale 2-d density images of the response conditional on each predictor bin. Some obvious outliers in BCUT space are excluded, as in the ordinal case. The density of the points is indicated by the grey-level, darker being higher density, and the line is a fitted local regression curve. The flatness of the curves suggests that no single BCUT value is a good predictor.

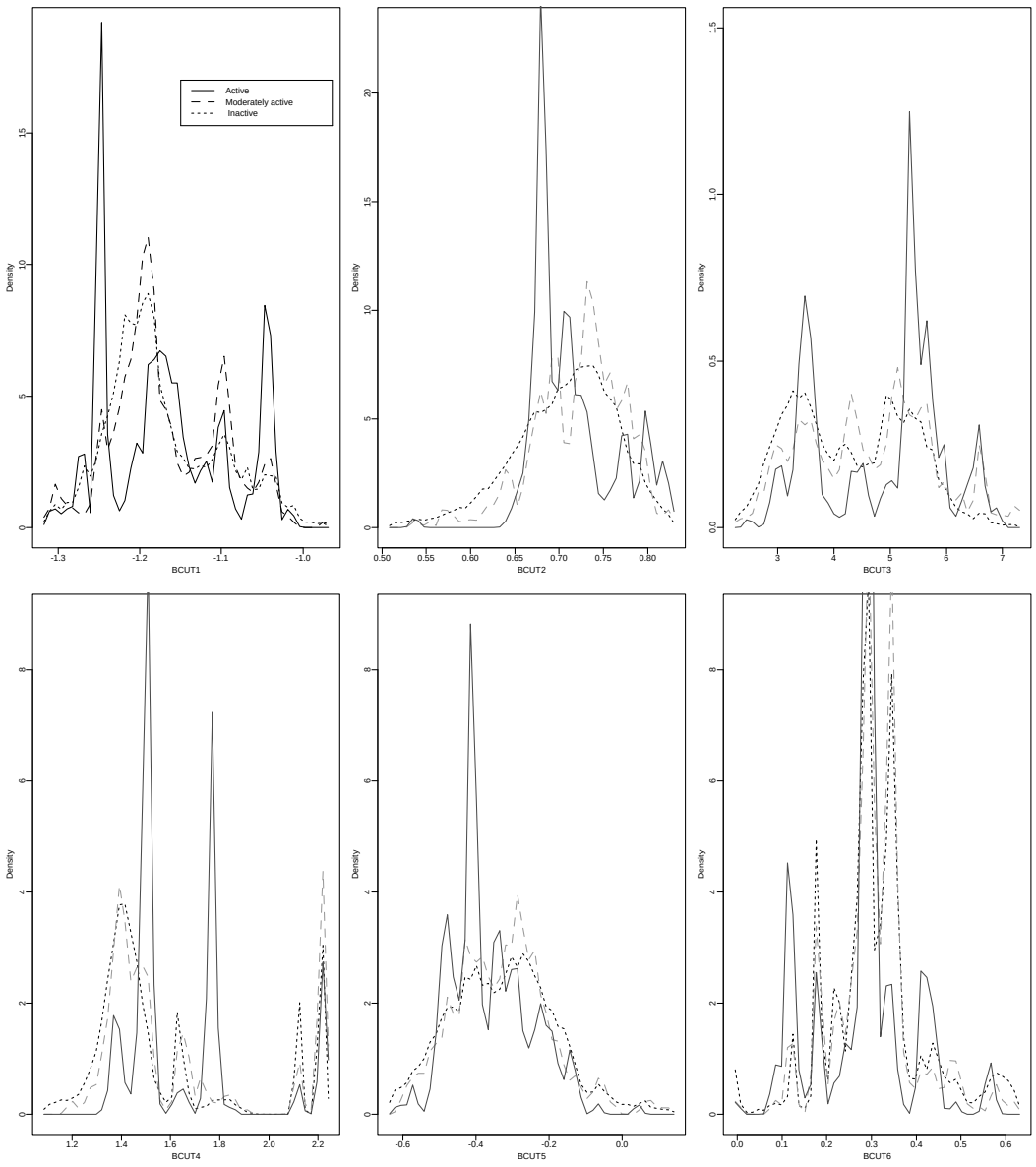

Figure 2.2: Density of BCUT1 to BCUT6 by activity. In each picture, some possible outliers are removed (the extreme 1% of data on both sides).

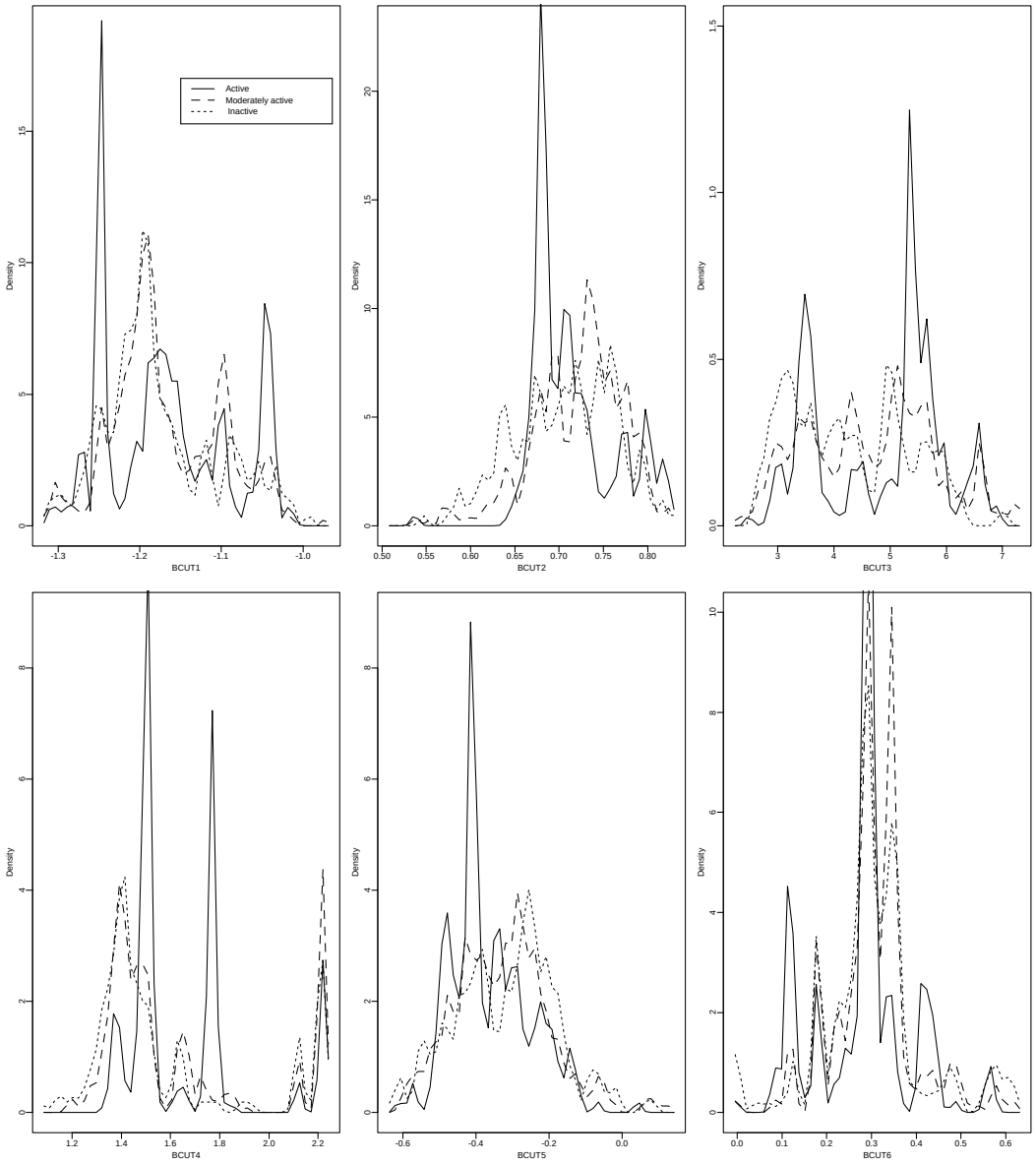

Figure 2.3: Density of BCUT1 to BCUT6 by activity. In each picture, some possible outliers (the extreme 1% of data on both sides) and a random sample of 215 inactive compounds are used to produce its density line (compared to the total 29204 inactive compounds in Figure 2.2).

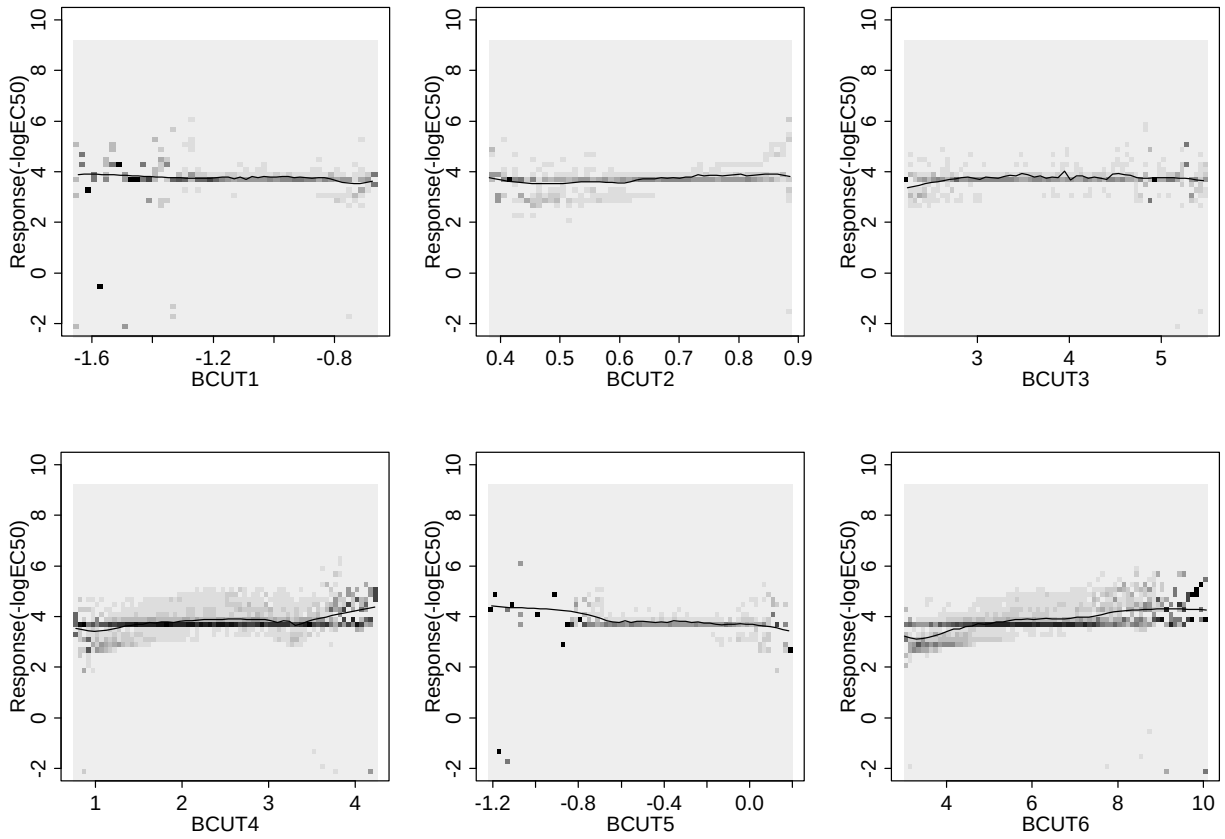

Figure 2.4: Grey-scale density plots of the response versus predictors. The curve in each plot represents a fitted local regression.

2.3.2 Objectives

The object of analyzing such data is to understand the SAR. Specifically, scientists want to use a fitted SAR from a relatively small screen (thousands to tens of thousands of compounds) to guide the selection of further compounds from a database. These databases are often huge, but the active compounds are rare, and screening all possible compounds is economically prohibitive. Thus, chemists and biologists want to select and assay a relatively small number of compounds. Further compounds will be chosen according to the fitted SAR. Ideally, they will include many potent molecules, which will be passed on to the next stage of drug development.

To compare the modelling methods, I randomly divide the data into training (model building) and test (validation) sets of equal size. For categorical data, separate splits are made of the active and inactive compounds, so the training and test sets are both comprised of 304 active compounds and 14602 inactive compounds. For the continuous data, a random split is applied such that the training and test set each has 14298 observations. The training/test split simulates the situation in which a limited number of assays are used for model fitting (the training data) and in which activity is to be predicted for the rest of the collection (the test data).

The hit rate (proportion of active compounds or "hits" amongst those selected) is a popular measure (e.g., Tatsuoka, Gu, Sacks, and Young 1998) for evaluating the predictive performance of classification models. Furthermore, the hit rate needs to be high only for a few hundred compounds ranked highest by any method and hence chosen. For instance, a 50% hit rate for 200 compounds selected will generate 100 active compounds. These compounds are usually examined for their chemical structures. It is desirable to have several "leads"

from different chemical classes for the further optimization of activity, toxicity analysis, etc. The hit curve of the highest-ranked compounds selected by a model depicts the number of active compounds, or hits, versus the number of compounds selected. In this chapter, hit curves will be compared visually to assess performance of models. Figure 2.5 gives an example of a hit curve.

Similarly, for continuous data, I use a statistical method to predict activity and select the compounds with the highest predicted activities. I then compare the distributions of measured activity for the compounds selected by the methods. An example of the graphical method of comparison is given later in Figure 2.7.

2.4 Comparison of the methods

First I divided both data into the training and test sets with equal size. For the categorical data, the training and test set have equal number of active and inactive compounds. After dividing the data, I constructed the models listed in Section 1.2.2 based on the training set and evaluated them on the test set. The estimated probability of activity (categorical data) or the estimated response (continuous data) is used as a score for the selection of compounds from the test set. For some methods, such as trees, C4.5, and KNN, a large number of ties may be present when the compounds in the test set are ranked.

2.4.1 Comparison of methods for the categorical data

For drug discovery data, I am interested in a few highest-ranked compounds selected by a model and would like to know their "target-hit" ability. Since the selection of compounds by models is no better than random after several hundred "lead" compounds are identified,

the hit curves I use go only up to 500 compounds selected by the models.

Figure 2.5 displays the hit curves for one random training/test data split. The horizontal axis represents the number of compounds selected; the vertical axis represents the number of actives actually obtained. For example, to select 100 compounds using one of the classifiers, every compound in the test data is scored, and the 100 highest-ranked compounds are chosen. The tree, KNN, and C4.5 methods give about 50 actives out of the 100 selected. Since the rankings provided by these three methods have many ties, their hit curves consist of points connected by straight lines. The points indicate the actual hits I can obtain when groups of tied compounds are simultaneously selected, and the lines provide the expected number of hits in between. Figure 2.5 shows that the tree model and KNN are the most successful techniques, dominating all the other methods. These methods are more local, and more flexible, and they make minimal assumptions about the underlying relationship. They are able to focus on very local regions containing concentrations of active compounds (e.g. Hawkins, Young, and Rusinko 1997) and capture interactions and thresholds which often exist in HTS data (e.g. Young and Hawkins 1998; Pearlman and Smith 1999). The performance of a technique as simple as KNN is impressive. Unlike GLM, GAM, MARS, and NN, the tree model and KNN do not assume that the relationship between the probability of biological activity (the response) and the measurable features of the chemical structure (BCUTs) can be approximated by a continuous function.

C4.5 is a method similar to classification trees. It grows the decision tree to produce rules and then simplifies them. Figure 2.5 shows it is among the best models at the very beginning (selecting fewer than 100 compounds) but finds few further actives after that. This may be due to the way that C4.5 deletes rules: it leaves a large number of compounds grouped together with tied scores. This is evident in Figure 2.5: from 100 to 500 compounds,

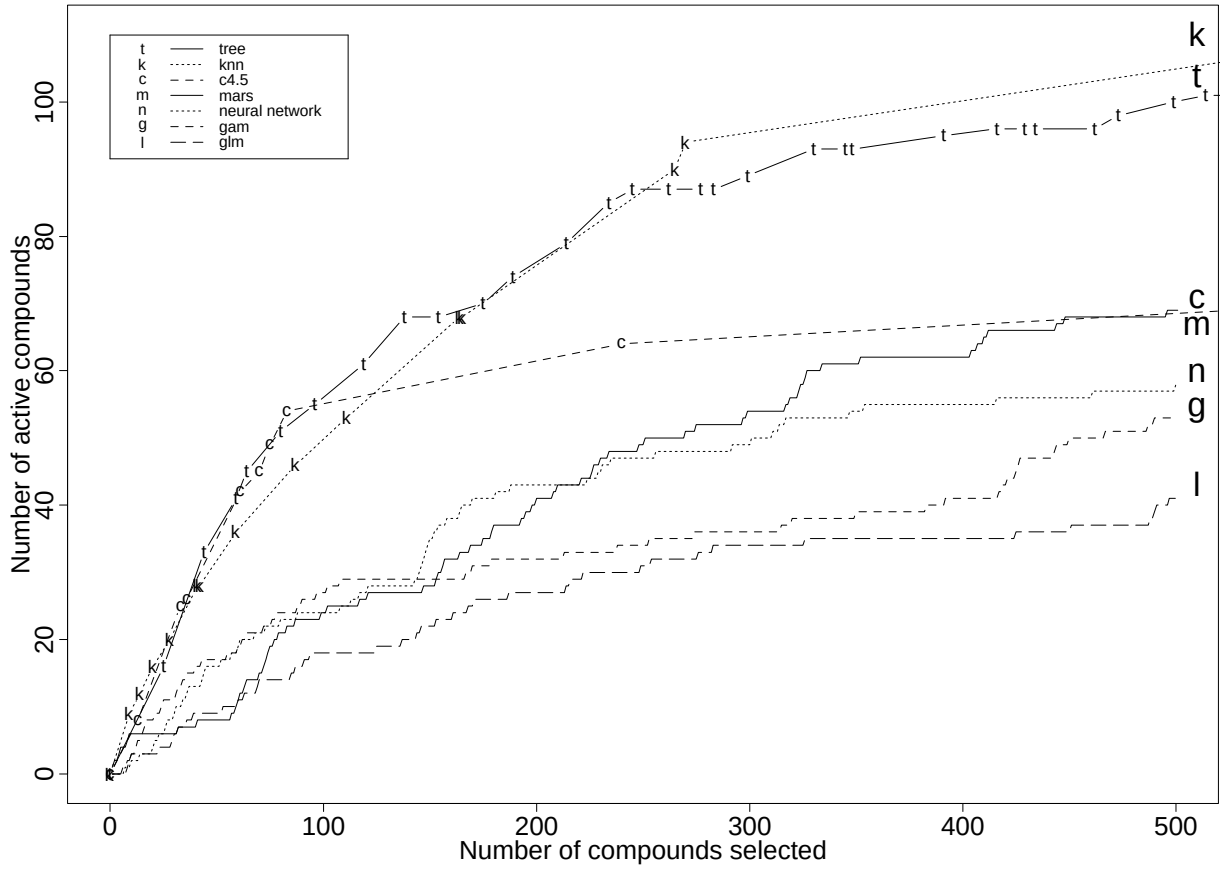

Figure 2.5: The number of active compounds found by the above methods in the test data versus the number of compounds selected.

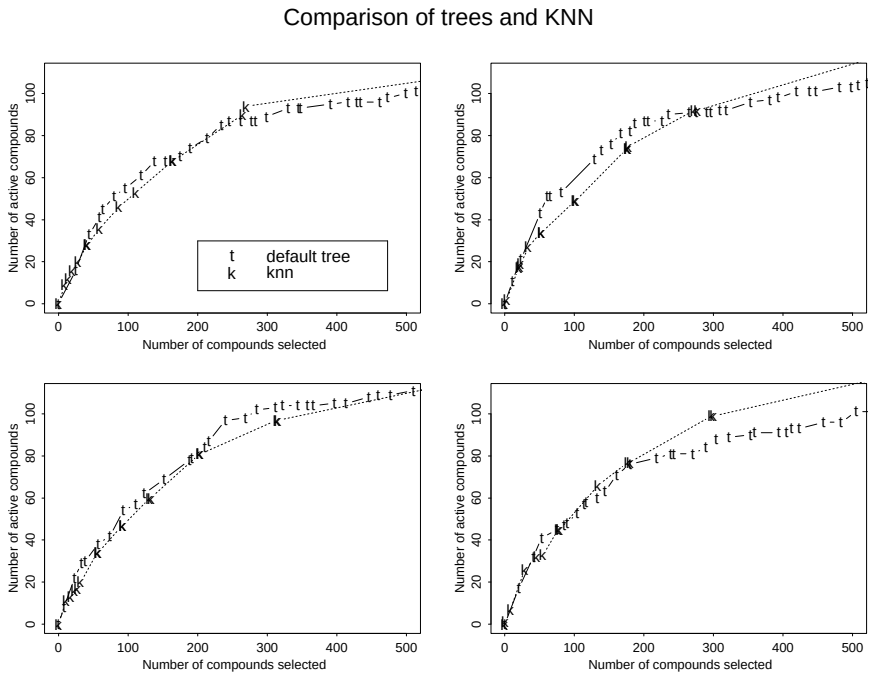

Figure 2.6: The number of active compounds found by the default tree and KNN, respectively, in the test data versus the number of compounds selected based on four random splits.

only two large groups are selected. In Section 2.5, I demonstrate why larger trees with more nodes (rules) perform better in terms of hit curves.

To check whether these results depend on the training/test data split, I randomly split our data four times. All splits have the same balance of active/inactive in training and test sets. Figure 2.6 gives hit curves for KNN (with the symbol "k") and the default tree model (with the symbol "t") for each split. The first split (upper right plot in Figure 2.6) is the same split as in Figure 2.5. In terms of the hit curves, trees and KNNs are competitive with each other. One deficiency of KNN is that it gives very little usable information regarding how each predictor (the BCUTs here) relates to activity. Ideally, a good classifier provides

not only accurate predictions but also some insight into important chemical features. In this respect, classification trees are good candidates: the important BCUT numbers are used to generate splits during the tree-growing procedure, and the most promising terminal nodes of the final tree correspond to small regions of high activity in the BCUT metric space (e.g. Rusinko et al. 1999).

2.4.2 Comparison of methods for the continuous data

Unlike categorical data, continuous data do not have a hit curve. However, I still want to capture a model's ability to find the best compounds. The five pictures in Figure 2.7 summarize the activity distributions of the compounds in the test set that are ranked highest by the various methods. The number of compounds selected is 100, 200, 300, 400, and 500, respectively. As a baseline, a quantile plot of the whole test set is drawn at the left of each plot. The symbols $\times$, $+$, $\triangle$, and $\circ$ indicate the 90%, 75%, 50%, and 25% quantile, respectively. The points above the quantile curves are the upper 10% of measured activities for the selected compounds. The 90% quantile and the plotted points are two important criteria for evaluating the model performance. The higher the quantile and the extreme values, the better the performance. Figure 2.7 shows that the distributions of $-\log(EC50)$ values of selected compounds vary considerably by model type. In general, LM, GAM, MARS, and NN are not very different from each other. However, the compounds identified by the trees and KNN have higher activity since the median and upper quantiles are higher and there are more extreme values. Moreover, those two methods pick the very best compounds in the test set. Strong local behavior in the continuous data are also indicated by the fact that the optimal k equals 7 in KNN and the optimal tree size is over 150 (Section 2.5).

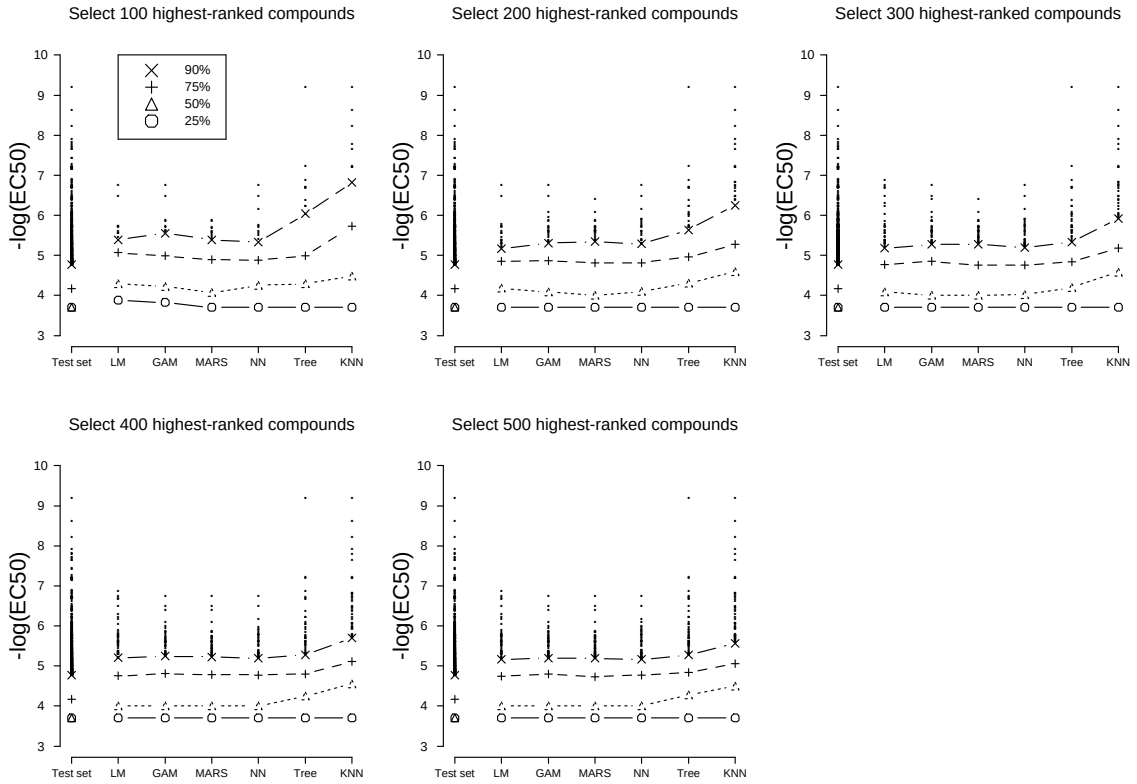

Figure 2.7: Quantile plots of the response variable for the selected compounds (the symbols $\times$, $+$, $\triangle$ and $\circ$ indicate the 90%, 75%, 50%, and 25% quantiles, respectively). The points above the quantile curves display the upper 10% of measured activities.

Even the best models select some compounds which are very inactive. One reason why the classification rankings are more striking is that they are treating the bottom 98% of the data as equal, whereas, in a regression setting, there is a substantial difference between the relatively low score of 2 and a score of 5. The regression models for continuous data may be modelling these unimportant distinctions. Weighting the more active compounds more heavily is one way to avoid this. Preliminary unpublished work by Lam shows that weighting improves matters, but unresolved issues remain, such as how to choose the proper weights and whether the weights can also apply to other methods (NN, MARS). The present study does not explore the weighting strategy further. Instead, it analyzes classification trees in detail and explains why tree pruning is ineffective in these contexts (Section 2.5).

2.5 Classification and regression trees

Being among the best performing models for the NCI AIDS data, trees are investigated in detail in this section. Classification and regression trees are conceptually similar. Both of them recursively make binary splits of the data along coordinate axes of the predictors such that at each division, the resulting two subsets of data are as homogeneous as possible with respect to the response of interest. The default trees giving the hit curves displayed in Figures 2.5 and Figure 2.6 and the regression tree displayed in Figure 2.7 are constructed using two constraints to stop further splitting:

- There must be at least 10 observations in a node; and
- The node deviance must be at least 1% of the root node deviance.

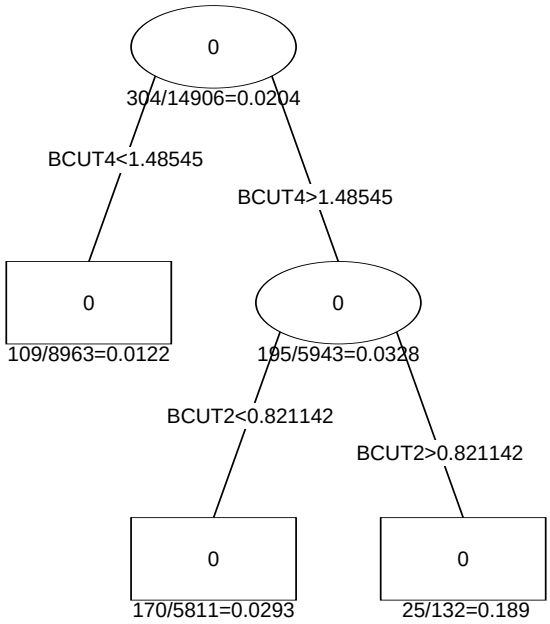

Figure 2.8: Part of the classification tree fitted to the NCI AIDS anti-viral binary data. The edges connecting the nodes are labeled by the left and right splitting rules. Interior nodes are denoted by ellipses and terminal nodes by rectangles, with the predicted class centered in the node. The fraction under each node is the number of compounds misclassified relative to the number of compounds in the node.

These constraints are the default options of tree models defined in S-Plus (Clark and Pregibon 1991). Since these settings are used regularly throughout the thesis, the resultant model will be referred to as the "default tree".

Figure 2.8 depicts the first few nodes of a default tree and thus, enables us to see how a tree makes predictions. This tree was built on the binary-response training set that leads to the hit curve in Figure 2.5. The whole training set (14906 compounds with 304 actives) in the top node (root) is divided into two subsets using BCUT4. The number inside each node is the predicted class. The fraction below each node indicates the number of cases misclassified

relative to the number of cases falling in the node. The misclassification rate,combined with the predicted label,can be used to determine the estimated hit rate of each node. For instance, 132 compounds in the training set reside in the right-bottom node of this tree. Of them, 25 are active. The estimated hit rate is $\hat{p}$= 25/132 =0.189, and, because this is less than 0.5, the compounds in this node are predicted to be inactive.

2.5.1 Classification trees

A better criterion for ranking nodes

It is common to take $\hat{p}$ as a score to rank the terminal nodes and prioritize compounds in the test set. However, these scores do not always provide a reliable ranking. For example, if I have two terminal nodes, one having 100 compounds with 99 active ($\hat{p} = 0.99$) and the other having only one compound which is active ($\hat{p} = 1$), the score for the first node is a much more reliable estimate of the true activity rate, although, according to $\hat{p}$, the second node should be ranked first.

To account for uncertainty in $\hat{p}$, I assume a binomial model for responses in each terminal node and conduct a one-sided hypothesis test. Note that a confidence set can be obtained from a hypothesis test by identifying all parameter values not rejected by the current data. Here, $\hat{p}$ is replaced by the test statistic, $p^{-}_{0.95}$, the smallest value of the null hypothesis value of p that is not rejected at 0.05 significance level.

Lam (2001, Chapter 4) originally proposed this method in the context of cell-based analysis methods for SAR modelling. Suppose that the compounds falling in the region defined by a terminal node of a tree have a probability of p to be active and that n compounds are in that terminal node. The number of actives, A, is a random variable following a

binomial distribution with n trials and probability p. Given the observed number of active compounds, a, in the node, $p^-_{0.95}$ is the solution to the function

$$P(A < a|p) = \sum_{i=0}^{a-1} \binom{n}{i} p^i (1-p)^{n-i} = 0.95 \tag{2.1}$$

Similarly, the upper bound of the one-sided 95% confidence interval of p ($p^+_{0.95}$) can be calculated by

$$P(A > a|p) = \sum_{i=a+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} = 0.95 \tag{2.2}$$

By definition, $p^-_{0.95} \leq \hat{p} \leq p^+_{0.95}$. I do not use $p^+_{0.95}$ because $p^+_{0.95}$ has difficulty handling the case when all the compounds in a terminal node are active. That is, $p^+_{0.95} = 1$ if $\hat{p} = 1$. It is not very convincing when the size of a terminal node is small. A similar argument could be made against $p^-_{0.95}$ when all the compounds in a terminal node are inactive. However, for such terminal nodes, we are not interested in an accurate estimate as much as identifying that the node should not be selected near the start of the hit curve. Therefore, $p^-_{0.95}$ is used in my thesis since it penalizes small terminal nodes. For instance, in the node having 100 compounds with 99 active, the lower bound of the 95% confidence interval is equal to 0.95, which is much larger than 0.05 for the node having only one compound which is active. Figure 2.9 displays $p^-_{0.95}$ as a function of a (number of hits) under different values of trials ($n = 10$, $n = 20$ and $n = 100$, respectively). They are all monotone increasing functions. The highest point for each curve corresponds to a perfectly estimated hit rate ($\hat{p} = 1$). However, the values of $p^-_{0.95}$ differ and are the largest (0.97) when $n = 100$ and the smallest (0.74) when $n = 10$.

Following Lam (2001) I use the bisection algorithm to compute $p^-_{0.95}$:

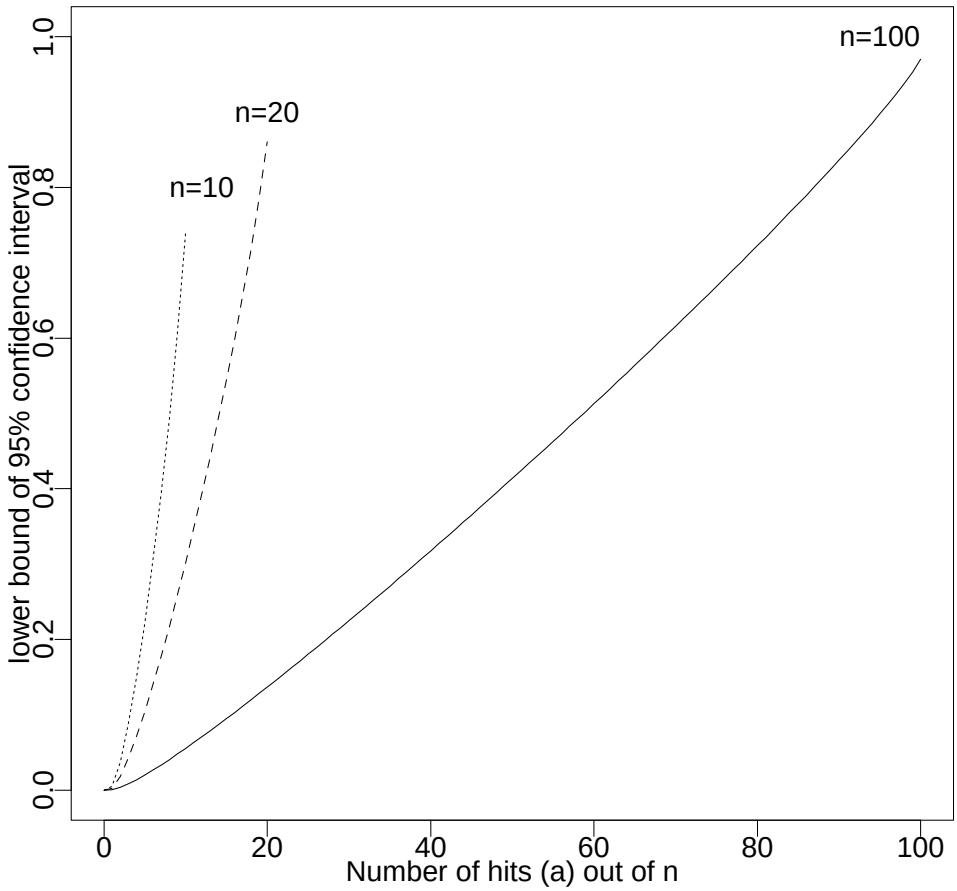

Figure 2.9: $p^{-}_{0.95}$ as a function of the number of hits (a) under various n values: $n = 10$, $n = 20$, $n = 100$.

1. *Initialization:*
   - *lower*= 0
   - *upper*= 0
   - *threshold*= 1
2. *Iterate while threshold*> $1e-4$.
   - $test = (lower + upper\ )/2$
   - $prob = P(A < a|test)$
   - *if prob*< 0.95
     - $upper = test$
   - *else*
     - $lower = test$
   - *Replace threshold by absolute value of* $(prob - 0.95)$
3. *Return test.*

The above algorithm gives the approximate $p^-_{0.95}$ in most cases. However, several special considerations should be pointed out.

1. When $a = 0$,

   $p^-_{0.95}$ does not exist, so I take $p^-_{0.95}$ =0.

2. When $a = n$,

   $p^-_{0.95} = (0.05)^{1/n}$.

The $p^-_{0.95}$ score will be large if $\hat{p}$ is large and if there are many compounds in the node. The hit curves for the default trees in Figure 2.5 and Figure 2.6 make use of such scores. They improve the trees' performances considerably. For example, for the data split used in Figure 2.5, when 100 compounds are selected, the expected number of active compounds (i.e., obtained by linear interpolation) increases from 52.0 to 56.0.

In many applications, large trees lead to over-fitting and thus hinder the model's performance. However, over-fitting does not appear to be a problem for the NCI AIDS data.

Table 2.1: Comparison of the first 3 terminal nodes for two ranking methods using training set frequencies. Pure tree nodes are contained in corresponding default tree nodes.

Node	Default tree						Pure tree		
	Training set					Test set	Training set		Test set
	$\frac{\#activecomp.}{\#total}$	$\hat{p}$	Rank	$p^-_{0.95}$	Rank	$\frac{\#activecomp.}{\#total}$	$\frac{\#activecomp.}{\#comp.}$	Rank	$\frac{\#activecomp.}{\#total}$
A	$\frac{13}{14}$	0.93	1	0.70	3	$\frac{8}{15}$	$\frac{13}{13}$	3	$\frac{8}{14}$
B	$\frac{20}{22}$	0.91	2	0.74	1	$\frac{16}{25}$	$\frac{16}{16}$	2	$\frac{14}{21}$
C	$\frac{22}{25}$	0.88	3	0.71	2	$\frac{17}{19}$	$\frac{22}{22}$	1	$\frac{17}{19}$

For investigation, a "pure" tree is constructed by removing the constraints on node size and deviance for the default tree so that all the terminal nodes have compounds of one class. In plots analogous to those in Figure 2.6, the pure tree performs as well as the default tree for this data set.

Pure trees also perform well because they are able to identify highly localized regions of high activity (nuggets). To illustrate, I select the best three terminal nodes from both the default tree and the pure tree built on the first training/test split as in Figure 2.5. Table 2.1 shows the top-ranked nodes (training data) from the two trees. Because the nodes selected from the pure tree are just subsets of those selected from the default tree, I use A, B, and C to represent those nodes. In the default tree, for example, 25 cases reside in node C with 22 hits. These same 22 hits are all in node C of the pure tree. In the pure tree, the $p^-_{0.95}$ criterion is reduced to choosing the largest pure active node first. This is reflected in the "Rank" column on the pure tree side of the table.

The three nodes displayed in Table 2.1 have very good hit frequencies. They define almost the same regions of the BCUT space although they may come from different trees (the default or the pure tree). In the test set, for instance, there are 19 compounds in node C for both the default and pure trees. Of them, 17 are active: a hit rate of 89%.

Examining the binary splits defining these nodes can indicate the role played by predictor variables. For example, Figure 2.10 displays a subset of the default tree which leads to the terminal node with predicted class label "1" and 3/25 misclassified (the third bottom terminal node in the picture and node C in Table 2.1). There are many splits on BCUT4 (v4 in Figure 2.10) for this default tree with 8 splits leading to node C. Of these, 5 split on BCUT4. The BCUT metric space corresponding to these nodes is highly localized. The range of BCUT4 in node C, for instance, covers only about 0.3% of the whole range. Therefore, BCUT4 is a very important predictor leading to the highly active region, node C, and also activity is highly localized in this variable.

Examining the size of the classification tree

The tree models leading to Figure 2.5 and Figure 2.6 are all very large. The default tree in Figure 2.5, for instance, has 131 terminal nodes. The corresponding pure tree has 393 terminal nodes.

Breiman et al. (1984) suggest a two-step procedure for choosing the size of a tree:

1. Pruning. A common approach is to grow the largest possible tree and then prune it back.

2. Searching amongst the many possible pruned trees using cross-validation to determine the best tree size. The most popular criteria are misclassification rate and deviance.

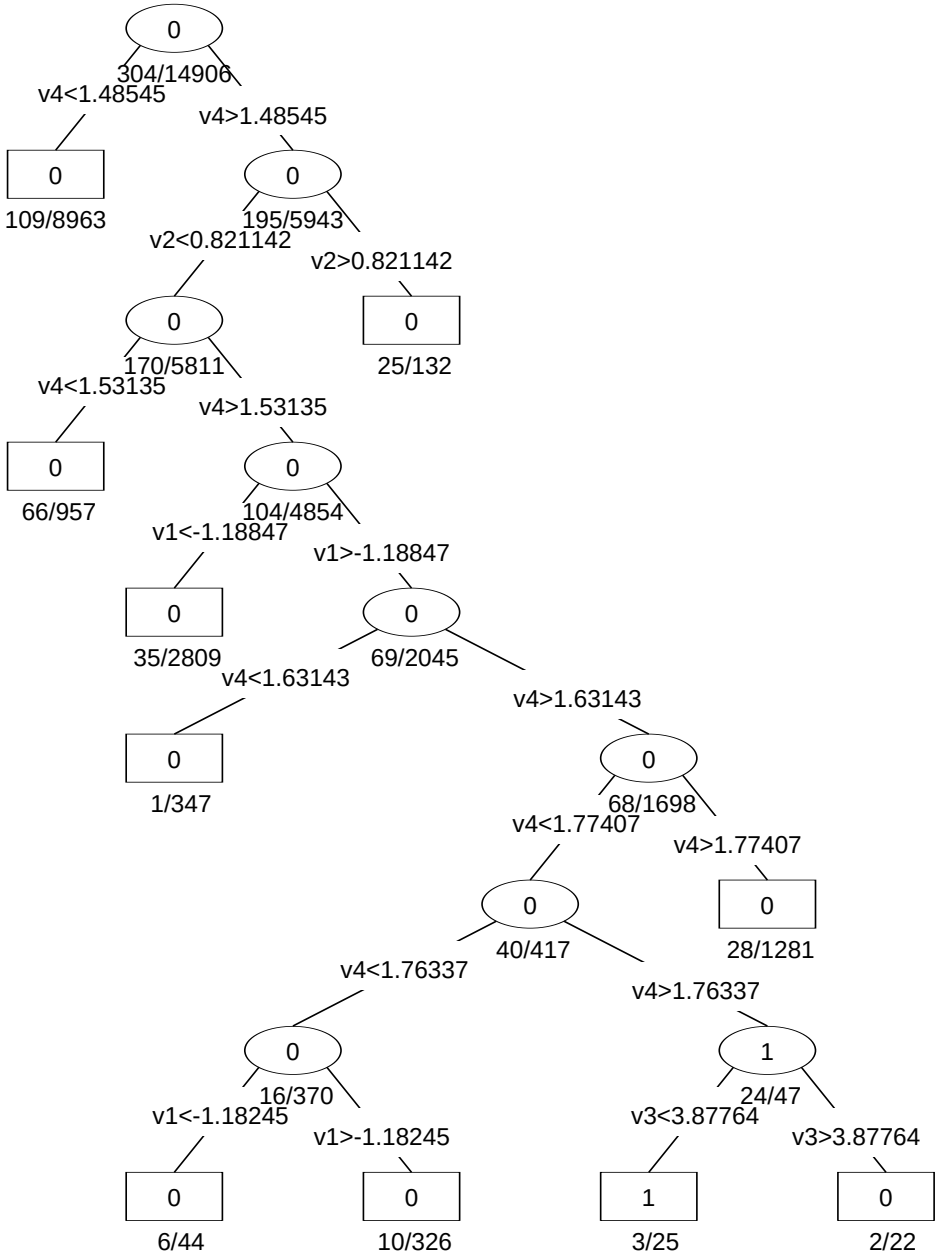

Figure 2.10: The subtree of default tree which leads to the terminal node with predicted class label "1" and 3/25 misclassified (the third bottom terminal node in the picture and node C in Table 2.1).

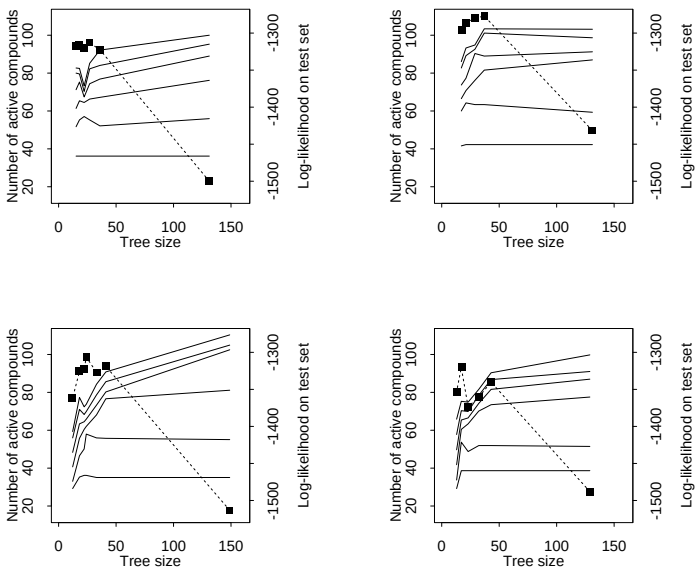

Figure 2.11: Each plot represents one training/test data split and displays the number of hits in the test set versus the size of the tree for selecting 50, 100, 200, 300, 400, 500 compounds (six solid lines), respectively, and the log-likelihood (calculated based on test sets) versus tree size (dotted line).

I explore the impact of pruning by computing the hit rate for a fixed number of compounds selected as trees are pruned back. In Figure 2.11, the default tree is pruned using "misclassification" as a pruning criterion. The four pictures illustrate the results for four random splits (those in Figure 2.6). In each picture, the six solid lines depict how the number of hits in the test set varies with the tree size when 50, 100, 200, 300, 400, or 500 compounds are selected, respectively, and the dotted line displays how the log-likelihood evaluated on the test set relates to the tree size. Most of the solid lines are monotone increasing, which indicates that larger trees seem to perform as well as or better than smaller trees. This phenomenon is more obvious when more compounds are selected (say 300 instead of 50). However, the

log-likelihood (a larger-the-better criterion), indicated by the dotted lines, suggests small trees (say 20–40 terminal nodes). Similar analysis of the misclassification rate also suggests small trees are optimal. Here, the deviance and misclassification rate are based on the test set, which would not be possible in practice. Thus, even when these classical criteria for choosing a tree size are evaluated on the compounds I am predicting, they lead to trees that identify fewer active compounds.

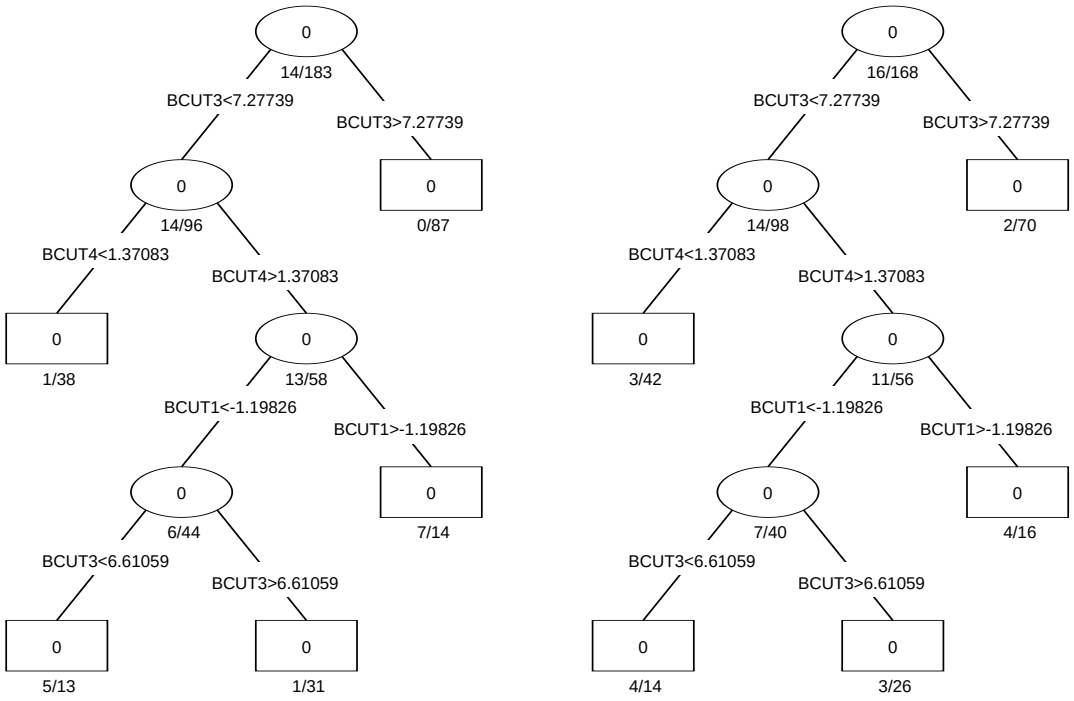

Figure 2.12: Part of a classification tree fitted to the NCI AIDS anti-viral data. The left one is a subset of the tree built on the training set, and the right one is the same tree evaluated on the test set.

I found that some very good terminal nodes may be lost when a large tree is pruned back. I now look at a particular node to see why pruning may lead to lower hit rates. Figure 2.12

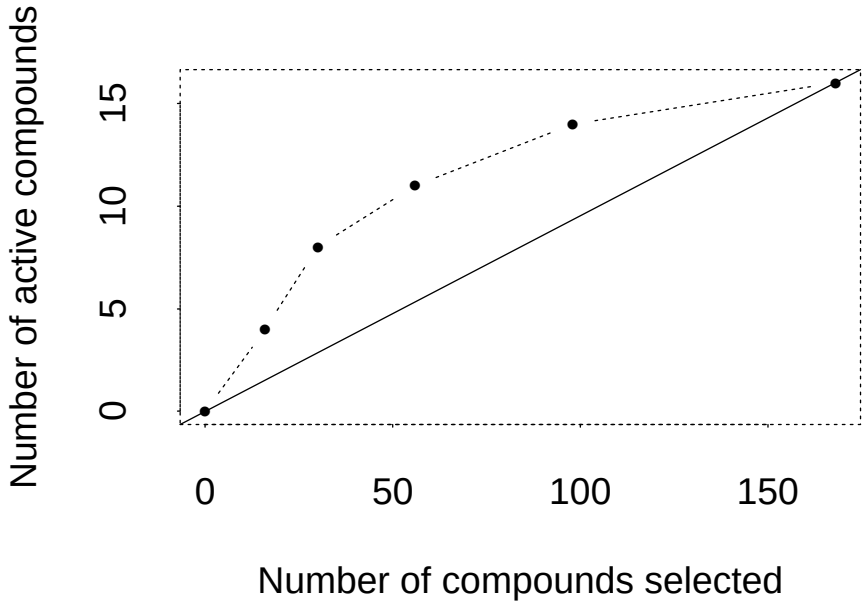

Figure 2.13: The number of active compounds versus the number of compounds selected in the test set for the subtree in Figure 2.12 (dotted line) and after the subtree is pruned back to the top node (solid line).

displays a subset of a default tree fitted to the training set (left tree) and evaluated on the test set (right tree). The terminal nodes of the trees in Figure 2.12 are also terminal nodes in the default tree. For example, in the training set, there are 183 compounds residing in the top node with 14 actives; in the test set, there are 168 compounds with 16 actives. During the pruning of the default tree, this subtree is reduced to the top node, giving a test hit rate of $16/168 = 0.095$, indicated by the straight line as the hit curve in Figure 2.13. When the subtree in Figure 2.12 is not pruned back to the top node, one can identify some sub-regions with higher hits.

For example, the child node having 13 compounds with 5 actives in the training set has 14 compounds with 4 actives in the test set. Its hit rate ($4/14 = 0.29$) is much higher than the top node (0.095). Therefore, the hit curve of this subtree indicated by the dotted line in Figure 2.13 is above the straight line.

The improvement shown in Figure 2.13 is for a test set and is thus not guaranteed to happen. A larger tree could over-fit the training data and have a worse test set hit curve. However, this example shows us the over-fitting in the traditional sense ("deviance" or "misclassification") may improve hit curves by subdividing large nodes into smaller ones, yielding fewer tied predictions.

2.5.2 Regression trees

In Figure 2.14, the default regression tree of the first split of data is pruned. It shows a pruned sequence of trees, starting from the largest tree (default tree). Cost-complexity pruning is used, with residual sum of squares on the training set as the pruning criterion.

The first five pictures of Figure 2.14 show the quantiles of the measured activities as they vary with tree size when 100, 200, 300, 400 and 500 highest-ranked compounds in the test set are selected. Again, it seems that large trees are at least as good as small trees. In contrast, the curve of the deviance on the test set against tree size in the sixth plot reaches the minimum at around 10 terminal nodes. Preliminary experiments show that very large trees (over 300 terminal nodes) are also competitive. Similar patterns also appear in the other three splits of data.

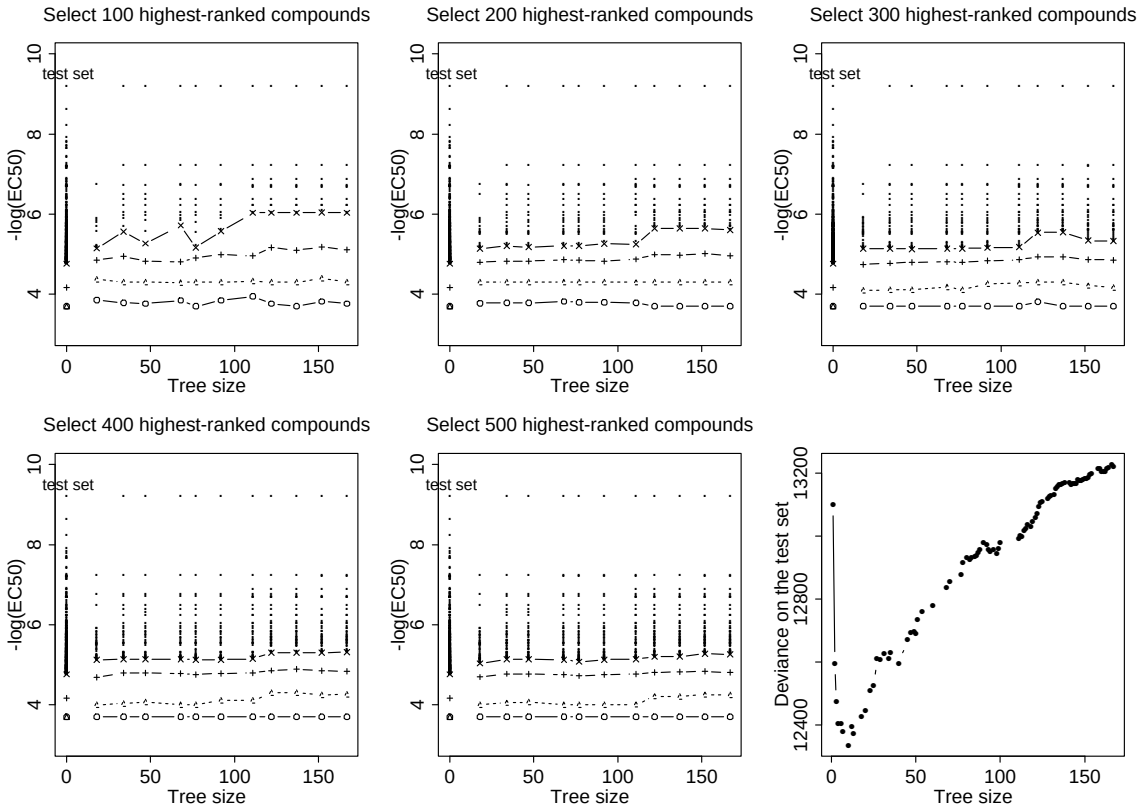

Figure 2.14: Quantiles of measured activity and deviance versus tree size (test set). In the first five pictures, the symbols $\times$, $+$, $\triangle$, and $\circ$ indicate 90%, 75%, 50%, and 25% quantiles of measured activity, respectively, and the points above the quantile curves display the upper 10% lead compounds.

2.6 KNN

For KNN on the continuous response data, the dependence of performance on model complexity is similar to that for trees. Figure 2.15 displays the quantiles and deviance versus k. The quantile plots pick very small values for k (3–7) as optimal. However, the deviance (the smaller, the better) suggests a quite large value for k (k=27 in this case). An important difference from earlier plots for trees is that small k values on the left of the plot represent

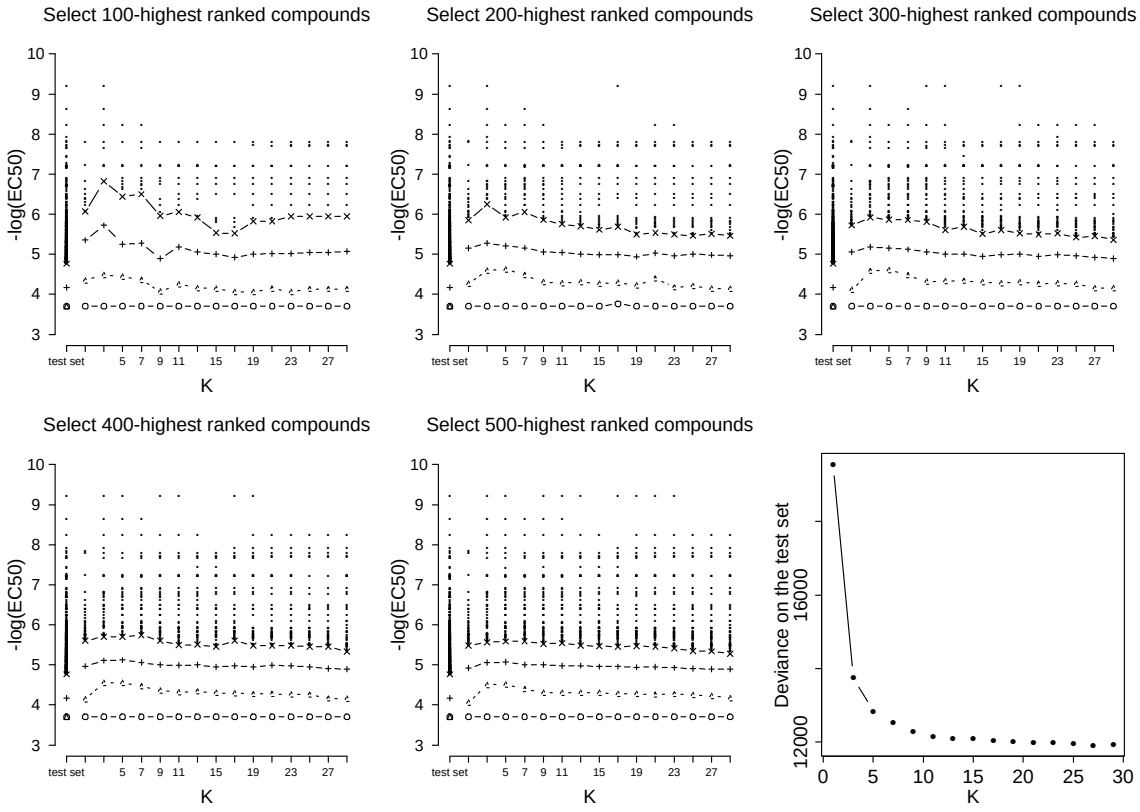

Figure 2.15: Quantiles of measured activity and deviance versus K (test set). In the first five pictures, the symbols $\times$, $+$, $\triangle$, and $\circ$ indicate the 90%, 75%, 50%, and 25% quantiles of measured activity, respectively, and the points above the quantile curves display the upper 10% lead compounds.

the most local models, while large tree sizes represent local models.

I have conducted a similar analysis for KNN for the categorical data. However, there is no apparent discrepancy between the value of k chosen by the misclassification rate and by the hit rate. Various random splits also suggest that the optimal k suggested by both criteria is quite small (about 3–7), although a very small k (1–2) performs poorly. Using a very large k does not identify as many actives, but the difference versus a small k is minor.

2.7 Conclusion

These data sets illustrate that HTS data are very complex, calling for specific methods to model the SAR. Local methods such as trees and KNN are competitive in these examples. In fact, they are good candidates for SAR modelling, in which localization is often present.

The appropriate ranking of terminal nodes is very important in improving a tree's performance. For instance, if I have two pure active nodes, then the lower-bound criterion will choose the node with the most compounds.

It is surprising that large trees outperform small trees in terms of number of hits and larger quantiles since conventional criteria (deviance or misclassification rate) suggest much smaller tree sizes. For binary data, it is probably because conventional accuracy measurements often assume that the response classes are of equal interest. For imbalanced HTS data, in which the aim is to predict the rare active compounds, the standard criteria are dominated by the majority of the inactive compounds. For continuous data, I am more interested in the few most potent compounds. However, deviance treats all the compounds equally. Therefore, the optimal tree size and the value of k for KNN suggested by the hit curves and quantile plots are far from those picked by deviance.

The deviance or the misclassification rate are not good performance measures for both data sets. In Chapter 3, alternative measures are introduced. Since the classification problems are more interesting to me, most investigation for the later chapters is based on the categorical data, which is referred to as the "NCI AIDS data".

Chapter 3

A Criterion for Assessing Methods When the Objective Is Ranking

3.1 Introduction

Consider the common classification problem with 2 classes $y \in \{0, 1\}$. Any classification method classifies each object, using some explanatory variables. Often, the misclassification rate is used as criterion for model building (with training data) or for model assessment (from test data). The misclassification rate is simply the proportion of objects assigned to the wrong class.

The approximation function $\widehat{p}(\mathbf{x})$, an estimate of the true probability of activity function $p(\mathbf{x})$, is chosen to minimize some specified loss function $L(y, p(\mathbf{x}))$.

$$\widehat{p}(\mathbf{x}) = argmin_p L(y, p(\mathbf{x}))$$

The training sample Z_1 enables us to estimate $\widehat{p}(\mathbf{x})$. For the classification problem, the

loss function can be the misclassification rate, and the misclassification rate is measured on the test set Z_2:

$$\frac{\sum_{(\mathbf{x}_i,y_i)\in Z_2} \mathrm{I}(\mathrm{I}(\widehat{p}(\mathbf{x}_i) > \frac{1}{2}) \neq y_i)}{n_2}.$$

Pursuing a low misclassification rate is a common strategy for solving the classification problem. However, the misclassification rate is not always an appropriate standard. When the proportion, π, of active compounds in the test data is small, the trivial classifier, which classifies all test compounds as inactive, has a small misclassification error (i.e., $= \pi$). For example, in the NCI AIDS data, the misclassification error of the trivial classifier is 2.04%. Since overall misclassification error is dominated by the majority of inactive compounds and interest lies in the hit rates for only a few lead compounds, the misclassification rate is an inappropriate loss function to identify the model with a high hit curve. This will be clear after I investigate the relationship between the misclassification error and hit curve (Section 3.2).

3.2 Investigation of the relationship between misclassification rate and hit curve

In drug discovery, one aim is to improve the hit curve. However, the minimization of the misclassification rate, a common criterion for classification, will not necessarily lead to the optimal hit curve. In some situations, these two goals might be far away from each other, and thus the chosen methods may perform poorly.

For the classification problem with a binary response, it is very common to provide the predicted class label by thresholding a classifier at $\hat{p} = 0.5$. Furthermore, in evaluating a

Table 3.1: Classification table.

		Predicted	
		active	inactive
Real	active	n_{tp}	n_{fn}
	inactive	n_{fp}	n_{tn}

classification method, scientists are very interested in four numbers on the test set related to the method: the number of false positives (n_{fp}), the number of false negatives (n_{fn}), the number of true positives (n_{tp}), and the number of true negatives (n_{tn}), which are the four elements of the classification table (Table 3.1). In detail,

- n_{fp}: the number of inactive compounds in the test set which are classified to be active (false positives);
- n_{fn}: the number of active compounds in the test set which are classified to be inactive (false negatives);
- n_{tp}: the number of active compounds in the test set which are classified to be active (true positives); and
- n_{tn}: the number of inactive compounds in the test set which are classified to be inactive (true negatives).

The misclassification error is a function of n_{tp}, n_{fn}, n_{fp}, and n_{tn}:

$$\text{misclassification error} = \frac{n_{fp} + n_{fn}}{n_{fn} + n_{fp} + n_{tp} + n_{tn}}$$

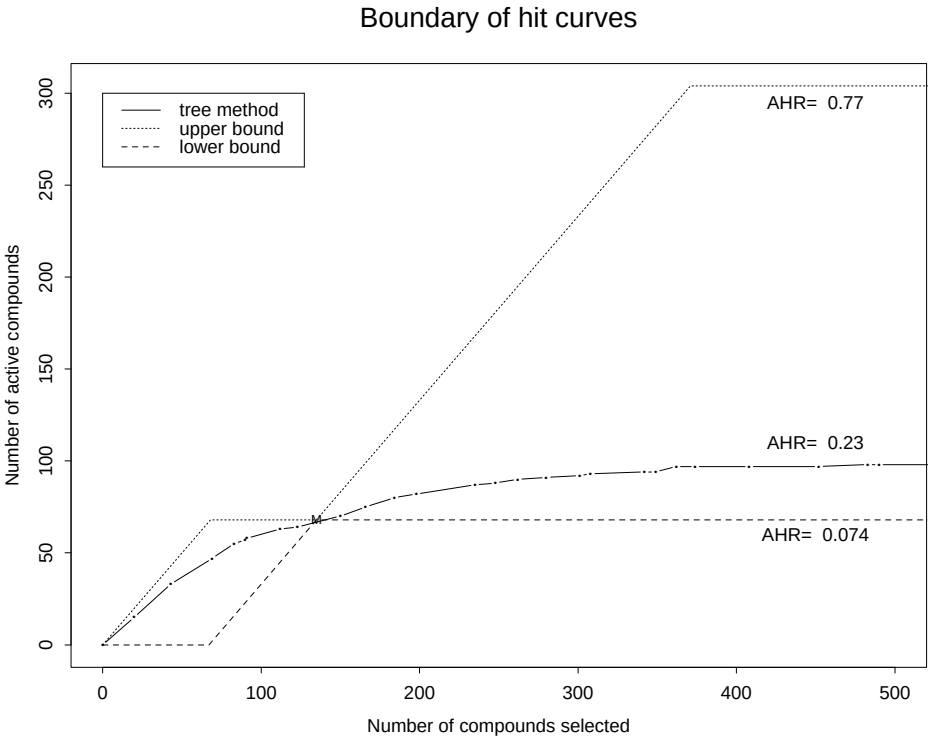

Figure 3.1: Bounds on the hit curve consistent with the tree method.

Hit curves could be very different with the same classification table when n_{tp}, n_{fn}, n_{fp} and n_{tn} are fixed. For the NCI AIDS data, the default tree method on the first training/test split of data has a misclassification rate of 2.07% on the test set (with the hit curve shown in Figure 2.5 of Section 2.4.1). Its hit curve is redisplayed here in Figure 3.1 (the curve connecting the dots) as well as in Figure 2.5. Based on its classification table (Table 3.2), we only know the hit curve must pass one point with the coordinate $(n_{tp} + n_{fp}, n_{tp})$ indicated by symbol "M" in Figure 3.1. For the given values of n_{tp}, n_{fp}, n_{fn} produced by the method, the best curve will have the method rank the true positive compounds ahead of the false positives, and the false negatives ahead of the true negatives. Hence, its hit curve (the dotted

Table 3.2: Classification table of the tree method evaluated on the test set based on the first split of data.

		Predicted	
		active	inactive
Real	active	$n_{tp} = 65$	$n_{fn} = 239$
	inactive	$n_{fp} = 69$	$n_{tn} = 14533$

line in Figure 3.1) connects five points:

$$(0,0),\ (n_{tp}, n_{tp}),\ (n_{tp} + n_{fp}, n_{tp}),$$
$$(n_{tp} + n_{fp} + n_{fn}, n_{fn} + n_{tp}),\ (n_{tp} + n_{fp} + n_{fn} + n_{tn}, n_{fn} + n_{tp}).$$

In contrast, the worst method puts the false positives in front of the true positives and the true negatives in front of the false negatives. Its hit curve (the dashed line in Figure 3.1) passes through

$$(0,0),\ (n_{fp}, 0),\ (n_{tp} + n_{fp}, n_{tp}),$$
$$(n_{tp} + n_{fp} + n_{tn}, n_{tp}), (n_{tp} + n_{fp} + n_{fn} + n_{tn}, n_{fn} + n_{tp}).$$

Therefore, with the fixed n_{tp}, n_{fn}, n_{fp} and n_{tn}, the hit curve can be as high as the dotted line, as low as the dashed line, or in between. That is, although different methods might misclassify the same number of compounds, their hit curves can vary substantially.

One goal of drug discovery is to improve the hit curve. The standard criteria such as the misclassification rate do not characterize the optimal hit curve and hence fail to reach this goal. On the other hand, as indicated by Tatsuoka et al. (1998), hit curves, commonly

used in model comparison, are not as clear-cut as some single-number criteria. In the next section the average hit rate (AHR) is discussed as a simple numeric summary of a hit curve.

3.3 Quantifying the hit curves

The definition of AHR is equivalent to "average precision" (in the appendix of Harman 1995), which is a common measurement in information retrieval. The "precision" is defined as the ability of a system to present only relevant items:

$$\text{precision} = \frac{number \quad of \quad relevant \quad items \quad retrieved}{total \quad number \quad of \quad items \quad retrieved}$$

Finding active compounds in a large collection of compounds is similar to finding relevant documents in a large collection of documents. Moreover, I want active compounds to appear in the ranked list as early as possible in the same way that I want relevant documents to appear as early as possible in the text retrieval process. Both tasks commonly have a highly unbalanced class distribution and output a ranked list of objects. I refer to precision as the *hit rate* in the remainder of this thesis.

A simple example illustrates the calculation of AHR. Suppose there are only five compounds in the test data, of which three are active. Table 3.3 shows a particular method which ranks the compounds and selects them in the following order: A, I, A, I, A, where A and I denote active and inactive, respectively. The hit rate of each active compound is computed. Hits (active compounds) are found at the first, third, and fifth selections. At the points on the hit curve where each active is found, the hit rates (proportions of hits) are 1/1, 2/3, and 3/5, respectively. The AHR criterion averages the hit rates at these points on the curve:

Table 3.3: Calculation of average hit rates.

rank	status (A: Active I: inactive)	hit rate	AHR
1	A	$\frac{1}{1}$	
2	I		$(\frac{1}{1} + \frac{2}{3} + \frac{3}{5})/3 =$
3	A	$\frac{2}{3}$	
4	I		0.76
5	A	$\frac{3}{5}$	

$$\text{AHR} = \frac{1}{3}\left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5}\right) = 0.76.$$

This is identical to the calculation of average precision. Depending on the different positions of the 3 hits of the 5 selected compounds, there are $\binom{5}{3} = 10$ differently ranked lists. For comparison, Table 3.4 illustrates three lists, corresponding to best, worst, and "in between" scenarios. According to the hits, list 1 is the best-possible ranking, list 3 is the worst-possible ranking, and list 2 is in the middle. Their AHRs are displayed in the last row of the table. Figure 3.2 illustrates their hit curves. The numbers in the curves indicate the corresponding lists. Hit curves with the most rapid early rise correspond to larger AHR. Therefore, the AHR is a good summary of the hit curve and much simpler to use for optimization.

Although the concept of AHR is very simple, the calculation is not as trivial as it looks especially when one has a list with many tied scores. Since the responses do not all take identical values within the tied group, there are multiple ways to calculate the AHR by reordering ties. Under the assumption of random ordering of compounds within tied groups,

Table 3.4: Comparison of 3 ranked lists.

Rank	List 1	List 2	List 3
1	A	A	I
2	A	I	I
3	A	A	A
4	I	I	A
5	I	A	A
AHR	1	0.76	0.48

exact closed-form expressions for the AHR can be derived. These are given after some initial development.

Suppose n compounds are arranged in descending order according to their estimated probabilities of activity, and the compounds in any group having the same probability of activity are in an arbitrary order within the group. Denote the response values for this ordered list of compounds by $y_{(1)}, \ldots, y_{(n)}$. The AHR of the list is defined as

$$AHR = \frac{\sum_{i=1}^{n} (\frac{y_{(i)} \sum_{j=1}^{i} y_{(j)}}{i})}{A}, \tag{3.1}$$

where

$$A = \sum_{i=1}^{n} y_{(i)} \text{ (the number of actives in the list)}$$

$$y_{(i)} = \begin{cases} 1 & \text{if the } (i)\text{-th compound is active} \\ 0 & \text{if the } (i)\text{-th compound is inactive.} \end{cases}$$

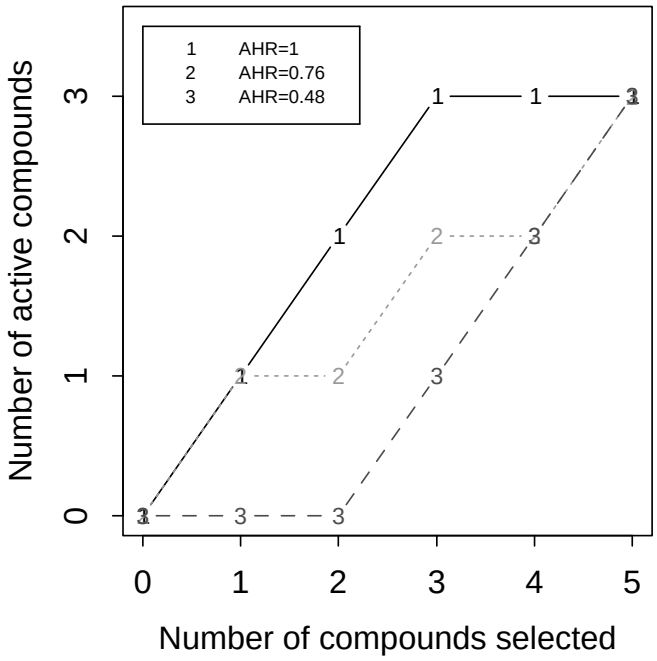

Figure 3.2: Comparison of the hit curves and their AHRs of the 3 ranked lists.

Because of ties the expected AHR over the random permutation of compounds with the same rankings is calculated:

$$\begin{aligned} \mathrm{E}(\mathrm{AHR}) &= \mathrm{E}\left(\frac{\sum_{i=1}^{n}\left(\frac{y_{(i)}\sum_{j=1}^{i} y_{(j)}}{i}\right)}{A}\right) \\ &= \frac{\sum_{i=1}^{n}\left(\frac{1}{i}\sum_{j=1}^{i}\mathrm{E}(y_{(i)}y_{(j)})\right)}{A} \\ &= \frac{\sum_{i=1}^{n}\left(\frac{1}{i}\sum_{j=1}^{i} p_{ij}\right)}{A}, \end{aligned} \tag{3.2}$$

where

$$p_{ij} = \mathrm{E}(y_{(i)}y_{(j)}) = P(y_{(i)} = 1 \quad and \quad y_{(j)} = 1). \tag{3.3}$$

p_{ij} can be calculated numerically and easily implemented on computer. Suppose all the compounds can be divided into M groups so that the compounds of each group have the same estimated probability of being active. For the m-th group, where $m = 1, 2, \ldots, M$, there is a total of C_m compounds with A_m actives, and Θ_m is a set of indices of those compounds:

If $(i) \in \Theta_m$ and $(j) \in \Theta_m$,

$$p_{ij} = \begin{cases} \frac{A_m}{C_m} & \text{if } (i) = (j) \\ \frac{A_m}{C_m}\frac{A_m - 1}{C_m - 1} & \text{if (i)} \neq \text{(j) and } A_m \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, if $(i) \in \Theta_{m_1}$ and $(j) \in \Theta_{m_2}$, where $m_1 \neq m_2$,

$$p_{ij} = \frac{A_{m_1}}{C_{m_1}}\frac{A_{m_2}}{C_{m_2}}. \tag{3.4}$$

The algorithm for computing AHR is simple:

1. *Order the list according to the scores; assign the group index to the list; compute A_m and C_m; sum = 0*
2. *Loop through the list. For $i = 1, 2, \ldots, n$*
   - *sum1=0*
   - *For $j = 1, 2, \ldots, i$*
     - *Calculate p_{ij}*
     - *sum1= sum1+ p_{ij}*
   - *sum=sum +$\frac{sum1}{i}$*
3. *Return $\frac{sum}{A}$*

The algorithm produces very useful results and enables much more efficient calculations of AHR than simulating random permutations and taking an average across each permutation.

With simulation, for instance, it took days to do cross-validation by means of simulation but only hours with the above algorithm.

The variance of AHR is important when many tied rankings are considered in both tree and KNN methods. In order to calculate this variance, $E(AHR^2)$ is required. As the square of a double sum will involve complicated calculation, I instead obtain AHR via simulating random permutations and then computing the standard deviation of the AHRs from permutations. In the next section, the standard deviation of AHRs (indicated by the number with an underline in Table 3.5) is calculated for both tree and KNN methods on the four random splits of NCI AIDS data along with AHR. Apparently, the variability of the AHR due to the compounds with tied scores is small.

Other measures for ranked lists are summarized in Appendix A of Harman (2000). One alternative is the average hit rate over *all* compounds (active and inactive):

$$\mathrm{AHR}_{\mathrm{all}} = \frac{\sum_{i=1}^{n}(\frac{\sum_{j=1}^{i} y_{(j)}}{i})}{n} = \frac{\sum_{i=1}^{n} h_i}{n} \tag{3.5}$$

where

$$h_i = \frac{\sum_{j=1}^{i} y_{(j)}}{i} \tag{3.6}$$

It is actually an extreme case of the measure called "Document Level Average" (Appendix A of Harman 2000). There, the precision is calculated at several document cut-off values and presented as a table (not a one-number summary as AHR proposed here).

For the NCI AIDS data, most of compounds are inactive and many methods are selecting actives at a rate similar to random selection after several hundred compounds have been selected. Therefore, most h_i's measured on the majority of the inactive compounds are not very different for different methods but contribute significantly to $\mathrm{AHR}_{\mathrm{all}}$. This character-

istic will make it difficult for AHR_{all} to discriminate between methods on the basis of the number of actives near the start of the hit curve.

Another view of the distinction between AHR and AHR_{all} is in terms of rewarding the identification of actives. The reward reflects the desire to identify actives as early as possible in the hit curve, by making the contribution to AHR of each active equal to the hit rate among all compounds identified so far. The AHR_{all} criterion, on the other hand, gives a reward nearly as large for the identification of an inactive as for an active. For example, suppose the 11th highest ranked compound is considered, and 10 out of the first 10 compounds are active. If the 11th compound is active, a "reward" of 11/11 is given by both AHR and AHR_{all}. If the 11th compound is inactive, the reward is 0 for AHR, and 10/11 for AHR_{all}. If the goal is to reward identification of actives, AHR seems more reasonable than AHR_{all}.

3.4 Evaluation of the model performance by AHR

AHR has several nice properties:

- The value is between 0 and 1. The higher value indicates the better ranking scheme. 1 corresponds to identification of all actives before any inactives, and the smallest possible value of AHR depends on the data and is always larger than 0.

- AHR is sensitive to the entire ranking but insensitive to small changes. Changing a single rank will change the final score; a small change in ranking makes a relatively small change in the score.

- The expectation of AHR can be easily calculated even when many tied ranks are present in the list.

- AHR rewards the methods that find the interesting points quickly (highly ranked). It results from two facts: ranks closest to 1 receive the largest weight and then the weights gradually decrease along the ranking list; the hit rates on only active compounds contribute to the calculation of AHR.

- The inactives behind the last active compound in the list are ignored when computing AHR but the ones in the front contribute to the AHR in an indirect way.

In fact, the calculation of AHR only counts the hit rates of actives and in a very unbalanced data set, AHR is dominated by the highest-ranked hits, although it is measured on the whole list. As a result, for the NCI AIDS data, AHR is a good measure of the hit curve, which displays only the first 500 selected compounds.

For instance, in Figure 3.1, the upper bound of the hit curves has an AHR of 0.77 and the lower bound 0.074. The AHR of the tree is in between (0.23). The order of the AHRs matches the visual ranking of hit curves. Therefore, in practice, the AHRs are calculated to compare the methods instead of drawing hit curves. It is very convenient, especially when the good candidates from a myriad of methods need to be picked up. For example, instead of drawing Figure 2.5 for the NCI AIDS data, I have a table of the AHRs (Table 3.5) of the methods on four random splits of the data.

The worst-to-best ordering of models by AHR in Table 3.5 is mostly consistent across the four splits of the data. The ordering is GLM, GAM, NN, MARS, trees to KNN. The only exceptions are Tree and KNN in the first split and MARS and NN in the third split. The performance order by AHR also corroborates the conclusion drawn in Chapter 2. Also, the standard deviation of AHR for both trees and KNN is calculated by simulation (the numbers with underlines). These standard deviations describe the variability of AHR due

Table 3.5: AHRs of GLM, GAM, NN, MARS, trees, and KNN in the test set of four random splits of the data. For both tree and KNN methods which produce many tied rankings, the variability of AHR (standard deviation) is also calculated by simulation (the numbers with underlines).

	1st split	2nd split	3rd split	4th split	average(sd)
GLM	0.0515	0.0479	0.0592	0.0512	0.0524(0.0048)
GAM	0.0807	0.0692	0.0755	0.0727	0.0745(0.0049)
NN	0.0955	0.0957	0.1151	0.0854	0.0979(0.0124)
MARS	0.1352	0.1000	0.1005	0.1104	0.1115(0.0165)
Trees(sd)	0.2282(0.0041)	0.1833(0.0045)	0.1920(0.0050)	0.2018(0.0026)	0.2013(0.0194)
KNN(sd)	0.2279(0.0049)	0.2121(0.0033)	0.2313(0.0029)	0.2377(0.0028)	0.2273(0.0109)

Table 3.6: AHRs in the test set of four random splits of data provided by trees and KNN methods.

ith split	AHR(trees)	AHR(KNN)	Difference (d_i)
1	0.2282	0.2279	-0.0003
2	0.1833	0.2121	0.0288
3	0.1920	0.2313	0.0393
4	0.2018	0.2377	0.0359

Mean $\overline{d} = 0.0259$

Standard deviation $s_d = 0.018$

to the compounds with tied scores in a ranking list. Since the standard deviations are small, the comparison among the methods using AHR is fair. The AHR values for trees and KNN in Table 3.5 are expectations with respect to permutations of ties.

In addition, the AHR calculation provides a way to do hypothesis testing. Both Figure 2.5 and Table 3.5 show that KNN and trees are better than the other methods with KNN slightly better than trees. Is KNN statistically significantly better than trees? Table 3.6 lists the AHRs in the test set for four random splits of data provided by both methods and their differences (d_i's). Under the assumption that d_i's are from a normal distribution with mean μ_d and variance σ^2, I can use a paired-sample t-test:

$$H_0: \qquad \mu_d = 0$$

$$H_a: \qquad \mu_d > 0$$

$$t = \frac{\bar{d}-0}{s_d/\sqrt{4}} = 2.88 \text{ (p-value= 0.032)}$$

Since the p-value is small (0.032), there is some evidence in the data that the AHR of KNN is statistically significantly larger than the AHR of trees.

I also did the pairwise t-test among the other methods listed in Table 3.5. Those statistical tests further confirm the conclusions at the end of Chapter 2. In summary, trees and KNN are significantly better than NN, GAM, and GLM. KNN is superior to trees, but not by much. Based on the fact that KNN and trees are good at modelling drug discovery data, I modify them in later chapters to further improve the model performance.

3.5 Conclusion

Misclassification error is not an effective criterion in terms of hits for a very unbalanced data set like the NCI AIDS data. In contrast, AHR is a single-number summary of a hit curve and is easy to calculate. It will be used extensively throughout the rest of the thesis since I focus on models for selecting a rare class from unbalanced data such as the NCI AIDS data. The only exception is that in Chapter 6 the misclassification error is a criterion for Mutagenicity data (in this problem, the response classes are balanced.).

Chapter 4

Investigation and Modification of KNN

The model comparison in both Chapter 2 and Chapter 3 reveals that the KNN rule is among the best performers, even though it is simple and only an "off-the-shelf" implementation. This chapter identifies several characteristics of KNN which constrain its performance. Two modifications of KNN are introduced, and shown to improve performance in the drug discovery problem.

Section 4.1 discusses the possible drawbacks of the regular KNN method. The k-varying algorithm is introduced in Section 4.2, and Section 4.3 studies the characteristics of the method. Section 4.4 applies the k-varying algorithm to both a simulated data set and the NCI AIDS data set. These two examples give insight into the improved performance of the k-varying algorithm. Section 4.5 introduces a second modification of KNN by combining regular KNN with kernel weights and hence improves the smoothness of the method. Section 4.6 presents some conclusions.

4.1 Problems with the regular KNN method

Under the general assumption that

$$p(\mathbf{x}') \approx p(\mathbf{x}),$$

where $\mathbf{x}'$ is close to $\mathbf{x}$, the corresponding KNN rule typically uses

$$\hat{p}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i \tag{4.1}$$

to estimate the probability of activity. $N_k(\mathbf{x})$ is a neighbourhood of $\mathbf{x}$ and is determined by the k points nearest $\mathbf{x}$ ($\mathbf{x}_i$, $i = 1, 2, \ldots, k$) in the training set. In general, I hope to average as many nearest points as possible to obtain a small variance of the estimate, but the assumption above might no longer hold. A large bias may be introduced with the increasing value of k and thus with the enlarged neighbourhood of a test compound. The k chosen by means of cross-validation aims to balance decreasing variance with increasing bias.

Even though KNN is very effective in many complicated real-data problems, it is a rather simple method. Following are some of its drawbacks:

1. k is constant.

   Choosing a single value of k for all data points may not be optimal. For example, if the density of points varies across the predictor space, a constant value of k implies neighbourhoods of differing sizes. Choosing k adaptively by examining local areas can be helpful. This is explored in Sections 4.2–4.4.

2. Points are in or out.

The estimated probability of activity function (4.1) is rough, and is sensitive to the observed y values especially when k is small. Exactly k nearest points contribute to the prediction: the k chosen points receive the same weight $(1/k)$ and the rest 0. It is a discrete process: points are either retained or discarded. Thus, high variance might be introduced when k is chosen to be small. Applying smooth weights subject to the distance makes the point-selecting process continuous and might greatly reduce the variance. In Section 4.5, for example, a kernel function is combined with KNN to improve the smoothness of the method.

3. Curse of dimensionality.

   Similar to that of many models, the intuition behind the KNN method breaks down in high dimensions (e.g. Hastie, Tibshirani, and Friedman 2001, Section 2.5). Therefore, one must identify the important subspaces (dimension reduction). Once a small set of relevant subspaces is found, the resulting KNN can be very powerful. Chapter 5 provides a technique for identifying good subsets of variables and combining predictions across subsets into an ensemble model.

4. The choice of metric can be important.

   Selecting the right metric is always the first concern but very difficult to resolve for distance-oriented methods like KNN. For convenience, Euclidean distance is used here.

While the problems of KNN are obvious, the solution is not easy to find. In this chapter and the next chapter, I attempt to improve upon the KNN method by tackling these problems.

4.2 The k-varying algorithm

4.2.1 A motivating illustration

Some interesting facts arise when the regular KNN method is applied to the NCI AIDS data. In Chapter 2, leave-one-out cross-validation on the training sets suggests 3–7 nearest neighbours of interest. A smaller value of k allows us to pay attention to more local regions. Thus, KNN with a small k appears desirable in the NCI AIDS data.

However, investigation of the terminal nodes of the trees in Section 2.5 suggests some active regions have many active compounds. For example, the smallest size of the best three nodes (Table 2.1) identified by the default and pure trees (both having very large tree sizes) is 13, and each terminal node has a very high active ratio (≥ 0.88). Considering the fact that the probability estimation from more data points is more reliable (with smaller variance), the large k in KNN is preferred if possible (with small bias).

Selecting k adaptively may give us more accurate and reliable probability estimation. For example, Figure 4.1 illustrates simulated data with two active regions. Squares and solid diamonds indicate inactive compounds and active compounds, respectively. Two circles with the same radius represent boundaries of two possible high-activity regions. Region "a" should be ranked higher than region "b" since region "a" has more data points and the probability estimation in region "a" could be more reliable. However, if I use a very small k (from 1 to 5), the two regions cannot be distinguished in terms of ranking (by probability estimates) when the test compounds are located in the center of the regions (indicated by "?").

The two plots in Figure 4.2 display enlarged views of the region "a" and "b", respectively and the symbol "?" indicates the test compound in the center of each region. Table 4.1 depicts which region is ranked first by KNN with k from 1 to 5. In this simple example, 1

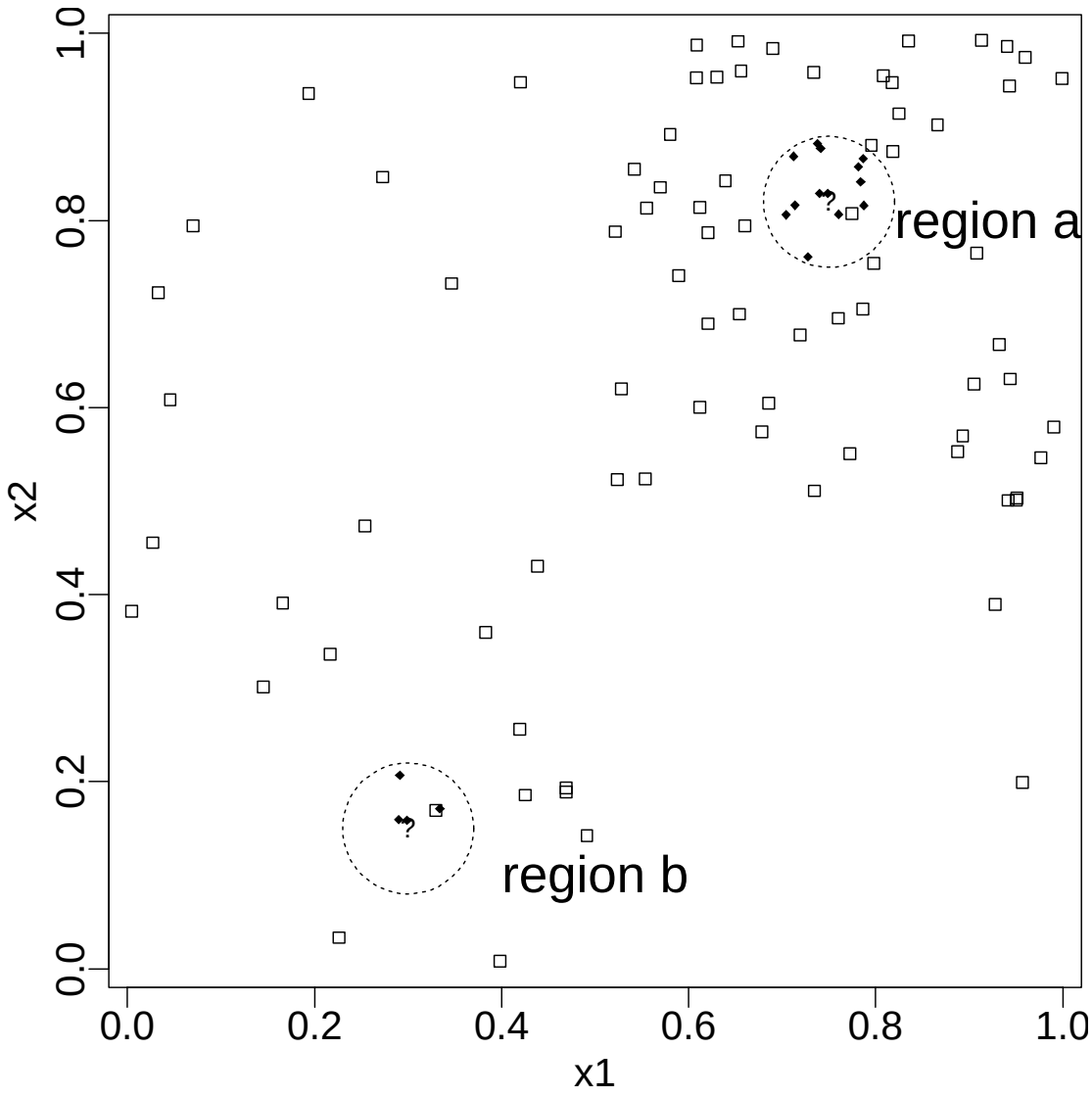

Figure 4.1: Simulated data with two active regions (a and b). Squares and solid diamonds indicate inactive compounds and active compounds, respectively. Two circles with the same radius represent the boundaries of the two active regions. The symbol "?" indicates the test compound in the center of each region.

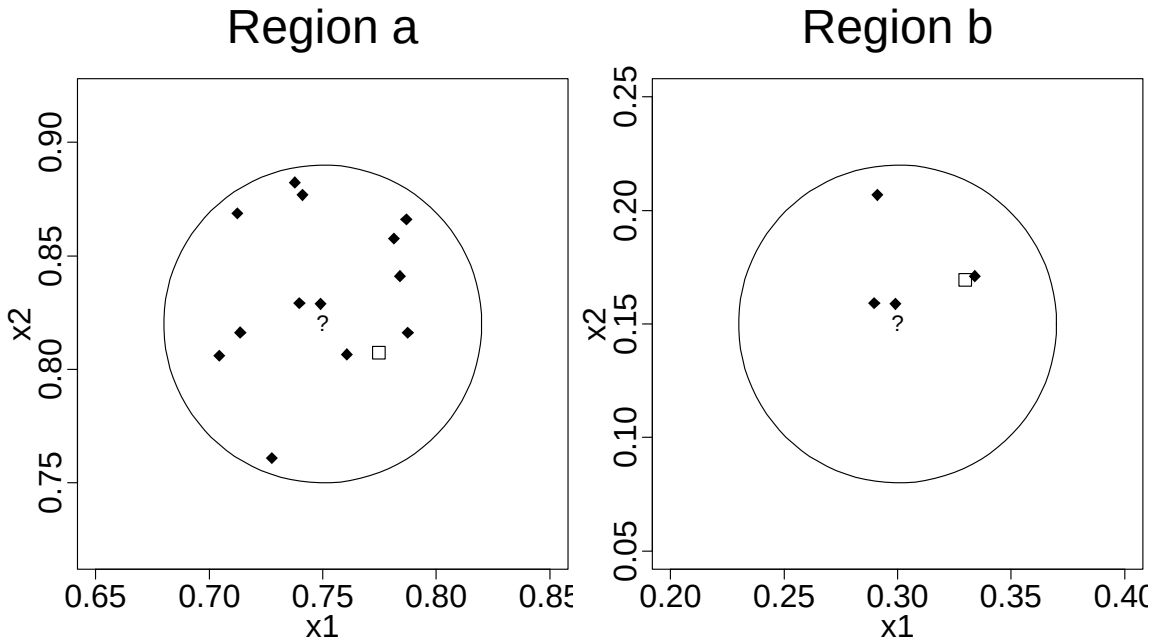

Figure 4.2: Active regions a and b, enlarged from Figure 4.1.

out of 5 times (when k=3) region "a" is ranked higher than region "b" while it is clear that region "a" is a more reliably active region than region "b". It would be ideal that all the compounds in the circles contribute to the prediction. Specifically, we would like $k = 14$ in region "a" and $k = 4$ in region "b" so as to include the maximal number of active compounds in each neighbourhood. This example suggests that allowing k to vary has advantages.

In addition, small k causes problems with hit curves, especially for large data sets. For example, for the NCI AIDS data, if $k = 3$, there are only 4 possible values of the estimated probability of activity (4.1) and thus 4 possible ranks (0/3, 1/3, 2/3, 3/3). Essentially, the whole test set (a total of 14906 compounds) is divided into 4 ranking groups. Within each group, further discrimination is not possible. The ranking scheme is too rough. This issue was discussed in Section 2.4.1 where in Figure 2.5 a few points of the hit curve of

Table 4.1: Comparison of the ranking by KNN of two active regions on the simulated data displayed in Figure 4.1.

k in KNN	# of actives in KNN		Best region(s)
	Region a	Region b	
1	1	1	a,b
2	2	2	a,b
3	3	2	a
4	3	3	a,b
5	4	4	a,b

KNN illustrate the places where the exact hit rates are present, and the connecting line is interpolative. This is caused by the many tied ranks. As a consequence, order within large groups of compounds is determined randomly. Selecting k adaptively alleviates this problem since it provides many more possible values of the score.

Of course, finding an algorithm to select good values of k is not easy. The intuition is to select the nearby compounds dynamically. For example, for an active test compound a good strategy could be to increase k until the active rate drops significantly. In Section 4.2.2, an algorithm which adaptively selects k is proposed.

4.2.2 The algorithm

Motivated by the intuition that a single value of k is not optimal for all scenarios, I now propose an algorithm that allows a different value of k for each test point. The values of k are chosen data adaptively and can depend on both $p(x)$ in a neighbourhood of a test point

x and the density of the training points.

Allowing k to vary raises the issue of ranking estimates of $p(x)$ based on different sample sizes. I follow the approach already used in Section 2.5.1 for comparing tree nodes of different sizes by computing a 95% one-sided confidence interval or lower bound, $p^-_{0.95}(k)$. For instance, the region with 9 actives out of 9 ($p^-_{0.95}(k) = 0.72$ when $k = 9$) is better than the one with 3 out of 3 ($p^-_{0.95}(k) = 0.37$ when $k = 3$), although they both have perfect hit rates ($= 1$).

The algorithm for choosing k adaptively is very simple:

- *Choose t to be the maximum k considered*
- *Loop through the test compounds. For $i = 1, 2, \ldots, n_{test}$*
  - *Find the t nearest neighbours in the training set to test point i.*
  - *Loop through the different k values. For $k = 1, 2, \ldots, t$*
    - *Find the number of actives among k nearest neighbours and calculate the corresponding lower bound of the 95% confidence interval $p^-_{0.95}(k)$*
    - *Return $\max_k p^-_{0.95}(k)$ and $k^*_i = argmax_k p^-_{0.95}(k)$ (the chosen k for test point i)*

The method is referred to as the "k-varying algorithm". Basically, for each test point i, the algorithm looks at from 1 to t nearest neighbours at the same time, and finds the value k^*_i that achieves the largest $p^-_{0.95}(k)$. k becomes a local factor and is determined by the activity of local observations. The central idea of the k-varying algorithm is to enlarge neighbourhoods as long as the estimated probability of activity is large or increasing.

The ceiling point t can be determined by cross-validation on the training set. That is, I choose different ceiling points, perform the algorithm using leave-one-out cross-validation on the training set, and t is selected to have the optimal AHR. Cross-validation can prevent t from getting too big such that some unnecessary positive bias is brought in (the 3rd property of the k-varying algorithm in Section 4.3). Usually, however, the effect of increasing

t further is to add more inactive compounds, which usually does not impact the rankings much. Therefore, the value of t is not critical and can be very robust in terms of AHR. For instance, in the NCI AIDS data, values of $t = 10 - 30$ work well and produce similar results. Alternative approaches to limiting k are possible. For example, it could stop increasing k once the lower confidence bound begins decreasing, or never moves significantly above the average p in the entire training set.

The situation in which multiple values of k give the same largest value of $p^-_{0.95}(k)$ only happens when all t of the nearest neighbours have $y = 0$. In this case, no matter what k is chosen $p^-_{0.95}(k)$ will be 0. The optimal k^*_i is set to 1. This is an arbitrary choice, but does not affect the predictions of the algorithm, since for all values of k^*_i the value of $p^-_{0.95}(k)$ will be 0.

4.2.3 The algorithm applied to the illustration

Before discussing characteristics of the algorithm, we give a simple illustration. Table 4.2 displays the result of applying the k-varying algorithm to the simulated data displayed in Figure 4.1. Suppose I have two test compounds: one in the center of region "a" and the other in the center of region "b". These are the points denoted by "?" in Figure 4.1 and Figure 4.2. For each test point, k is varied from 1 to 20 and corresponding $\hat{p}(k)$, hit rate at k nearest neighbours, and $p^-_{0.95}(k)$ are calculated. For the test compound in region "a", k is chosen to be 14, which gives a final score of 0.61 (the maximum $p^-_{0.95}(k)$). For region "b", $k = 5$ is chosen, and the corresponding $p^-_{0.95}(k)$ equals 0.34. Figure 4.3 displays $\hat{p}(k)$ and $p^-_{0.95}(k)$ as a function of k in active region "a" and "b". The picture shows up to 20 nearest neighbours. The selected k's (indicated by "+" in Figure 4.3) for both cases capture the active regions,

Table 4.2: Comparison of the ranking by the k-varying algorithm of two active regions on the simulated data.

Region	k	$\hat{p} = \frac{\#\ of\ actives}{k}$	$p^-_{0.95}$	Rank	Region	k	$\hat{p} = \frac{\#\ of\ actives}{k}$	$p^-_{0.95}$	Rank
	1	1/1	0.05			1	1/1	0.05	
	2	2/2	0.22			2	2/2	0.22	
	3	3/3	0.37			3	2/3	0.14	
	4	3/4	0.25			4	3/4	0.25	
	5	4/5	0.34			5	4/5	0.34	**2**
	6	5/6	0.42			6	4/6	0.27	
	7	6/7	0.48			7	4/7	0.23	
	8	7/8	0.53			8	4/8	0.19	
	9	8/9	0.57			9	4/9	0.17	
a	10	9/10	0.61		b	10	4/10	0.15	
	11	10/11	0.64			11	4/11	0.14	
	12	11/12	0.66			12	4/12	0.12	
	13	12/13	0.68			13	4/13	0.11	
	14	13/14	0.70	**1**		14	4/14	0.10	
	15	13/15	0.64			15	4/15	0.097	
	16	13/16	0.58			16	4/16	0.090	
	17	13/17	0.54			17	4/17	0.085	
	18	13/18	0.50			18	4/18	0.080	
	19	13/19	0.47			19	4/19	0.075	
	20	13/20	0.44			20	4/20	0.071	

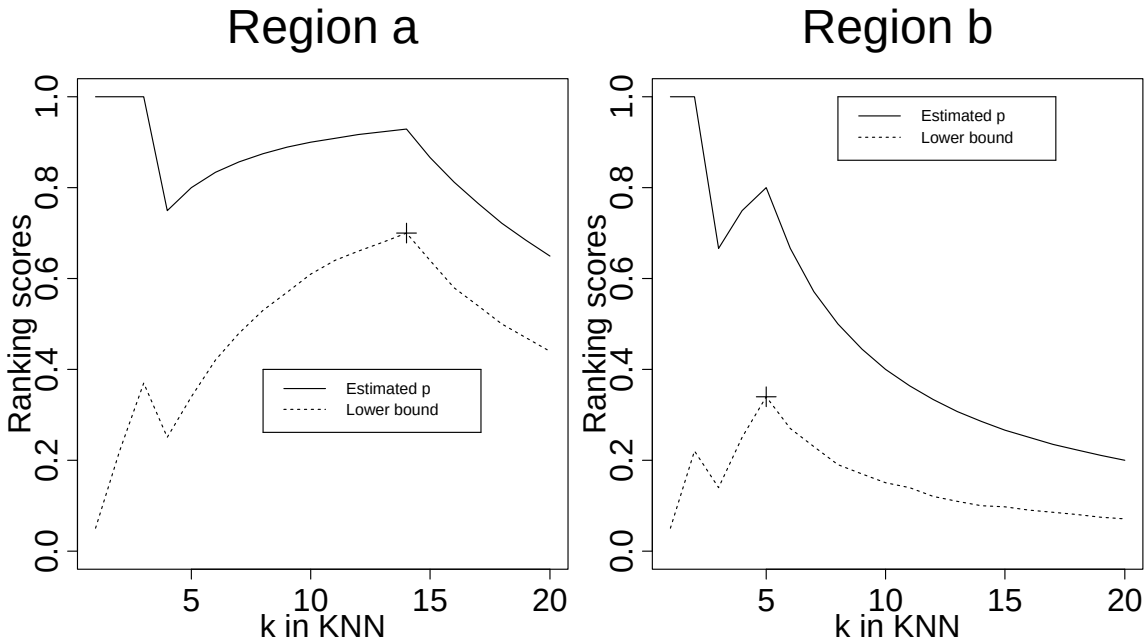

Figure 4.3: $\hat{p}$ and $p^{-}_{0.95}(k)$ versus k in the active region a and b. The solid line and dotted line indicate $\hat{p}$ and $p^{-}_{0.95}(k)$ as a function of k respectively. The "+" indicates the k chosen by maximizing $p^{-}_{0.95}(k)$.

and the final scores prioritize these two regions according to our preference. The choice of k_i^* is unchanged once the ceiling point t exceeds fourteen while the prioritization is right once t is larger than five.

4.3 Properties of the algorithm

Now let us look at the properties of the k-varying algorithm. Both the process of sequentially selecting k and AHR as a performance measurement make the algorithm hard to understand from a theoretical perspective while implementation is easy. Via several examples I will show how the algorithm is designed for unbalanced data sets to pursue the largest AHR.

Table 4.3: Table of $p^-_{0.95}(k)$.

		Number of trials									
		1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0	0
	1	0.05	0.025	0.017	0.013	0.010	0.009	0.007	0.006	0.006	0.005
	2		0.224	0.135	0.098	0.076	0.063	0.053	0.046	0.041	0.037
Number	3			0.368	0.249	0.189	0.153	0.129	0.111	0.098	0.087
	4				0.473	0.343	0.271	0.225	0.193	0.169	0.150
of	5					0.549	0.418	0.341	0.289	0.251	0.222
	6						0.607	0.479	0.400	0.345	0.303
actives	7							0.652	0.529	0.450	0.393
	8								0.688	0.571	0.493
	9									0.717	0.606
	10										0.741

In order to have more insight, Table 4.3 gives $p^-_{0.95}(k)$ values for all possible number of actives out of $1, 2, \ldots, 10$ nearest neighbours. Note that whenever there are no actives, $p^-_{0.95}(k) = 0$. Also, when moving from k to $k+1$ nearest neighbours, $p^-_{0.95}(k)$ will increase when the next nearest neighbour is active, and decrease when the next nearest neighbour is inactive.

The characteristics of the algorithm are described here.

1. It seeks low variance.

   This goal is reached by using $p^-_{0.95}(k)$ as a selection criterion. It takes account of the

uncertainty of the estimation and as a consequence, priority is given to those test compounds not only with high estimated probability but also with the estimation having little variability.

For example, Figure 4.4(a) shows $\hat{p}$ and $p^{-}_{0.95}$ for one particular realization of Y's for 10 nearest neighbours. It could be under the situation that $p(x)$ is constant among 10 nearest neighbours and equals about 0.5. Due to randomness, $\hat{p}$ changes a lot when k is small but is close to 0.5 after 4 nearest neighbours. The algorithm selects k=9 (a large value of k) giving the largest $p^{-}_{0.95}$ (indicated by "+") and it provides more confidence in prediction than a small value of k.

2. It avoids biasing $\hat{p}$ downwards.

   Figure 4.4(b) illustrates another example of the realization of Y's for 10 nearest neighbours. $\hat{p}$ is high when k is small but drops after k equals 4. It simulates the case that the test point has a very high probability of activity (about 1) but $p(x)$ drops off rapidly as the training points move away from the test point. The algorithm stops at a small value of k ($k = 4$ here) because the advantages in tightening the bound from larger sample size (less variance of $p(x)$) are outweighed by negative bias. The algorithm seeks a *maximum* value of $p^{-}_{0.95}$, hence negative bias is avoided.

3. It attempts to assign a higher score (i.e. $p^{-}_{0.95}(k)$) to "interesting" cases that have many active near neighbours. At the same time, it could bias the score $p^{-}_{0.95}(k)$ upwards.

   Figure 4.4(c) gives one plausible trajectory under the situation that the test compound has zero probability to be active but it is beside a peak of $p(x)$. Under this case, more active compounds appear as the training points move away from the test point. A

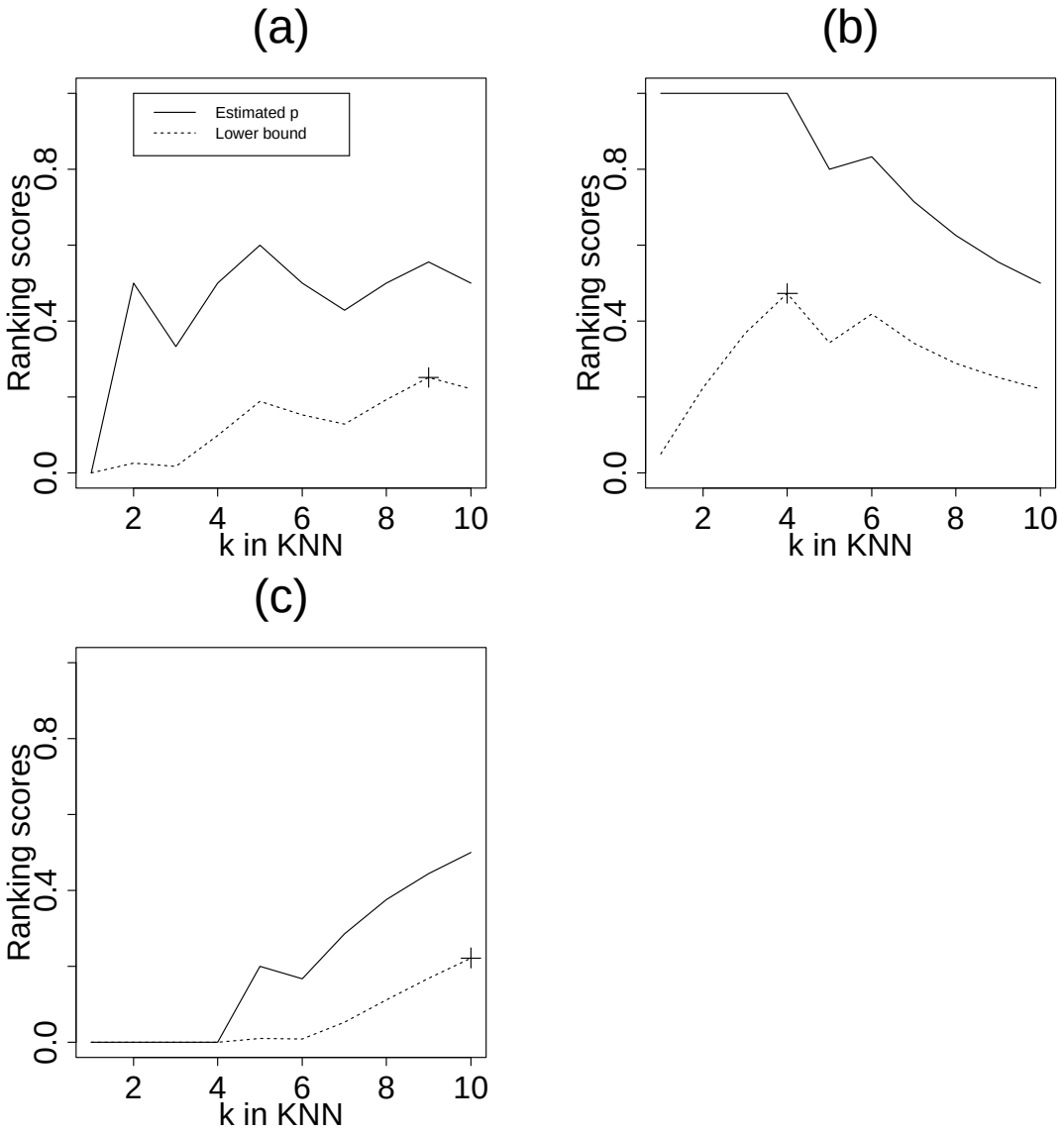

Figure 4.4: $\hat{p}$ and $p^{-}_{0.95}$ as a function of k for three typical trajectories of ten nearest neighbours.

large value of k ($k = 10$ here) is sought as it reduces variance and the algorithm (unfortunately) chases after positive bias. The positive bias should rarely be problematic if actives are rare.

4. It captures the local features adaptively (see Section 4.4).

   Given enough data, a classification algorithm can find complex features of interest in the data, if the algorithm is sufficiently data adaptive. KNN is already a fairly adaptive method; allowing k to vary increases its flexibility.

5. The computation does not increase dramatically relative to regular KNN.

   The most expensive computation is to calculate the distances and sort them. For both methods, they need to calculate and rank exactly the same number of distances for each test point. Distance calculation requires $n_{train}(3p-1)$ operations (the square root operation in the Euclidean distance is unnecessary as it does not change the ranking). Ranking them is an $O(n_{train} \log n_{train})$ calculation. The only difference is that KNN only cares about the hit rate of k nearest points (k operations for each test point) and the k-varying algorithm finds the number of actives for 1 to t nearest neighbours (t operations), obtains the corresponding $\bar{p}_{0.95}$ values from a pre-calculated table and finds the largest value (t operations).

   For cross-validation, both methods need to compute a distance matrix (for recording the pair-wise distances among the compounds in the training set). Therefore, cross-validation for either method has almost the same computation and is not prohibitive if the computer has enough space to save this distance matrix.

As the above points indicate, the k-varying algorithm will not necessarily make the best

choice in every possible scenario. It is interesting to note in the three examples given in Figure 4.4, that the ranking of the corresponding test points seems sensible. Case (b) has the highest $p^{-}_{0.95}(k)$, followed by case (a) and then case (c). That is, the algorithm chooses the point with a large $p(x)$ nearby first, then the point with $p(x) = 0.5$, and lastly the point with $p(x)$ small but with larger values of $p(x)$ nearby. Even though there is bias in cases such as (c), the ranking may still be sensible. In the cases considered here, it seems to offer an advantage over KNN.

Section 4.4 illustrates that the k-varying algorithm performs very well on both a simulated data set and the NCI AIDS data set. Several mechanisms such as the usage of $p^{-}_{0.95}(k)$ and adaption of k work together to achieve such performance. However, the algorithm involves not only a sequential optimization but also a complicated rank problem in pursuit of AHR. These characteristics make relevant theoretical analysis very difficult. Instead, we choose to explain performance via carefully chosen examples in Section 4.4.

4.4 Empirical performance of the k-varying algorithm

4.4.1 A one-dimensional simulated data set

A data set is simulated with a one-dimensional predictor. Figure 4.5 shows the probability of activity function $p(x)$ versus x. The function is designed to have different features. It is flat and near 0 at the beginning, gradually increases or decreases in the intervals (0.15,0.25) and (0.35,0.45), sharply increases or decreases (.65 to .75), has a wide peak between (0.25,0.35) and a sharp peak at 0.7.

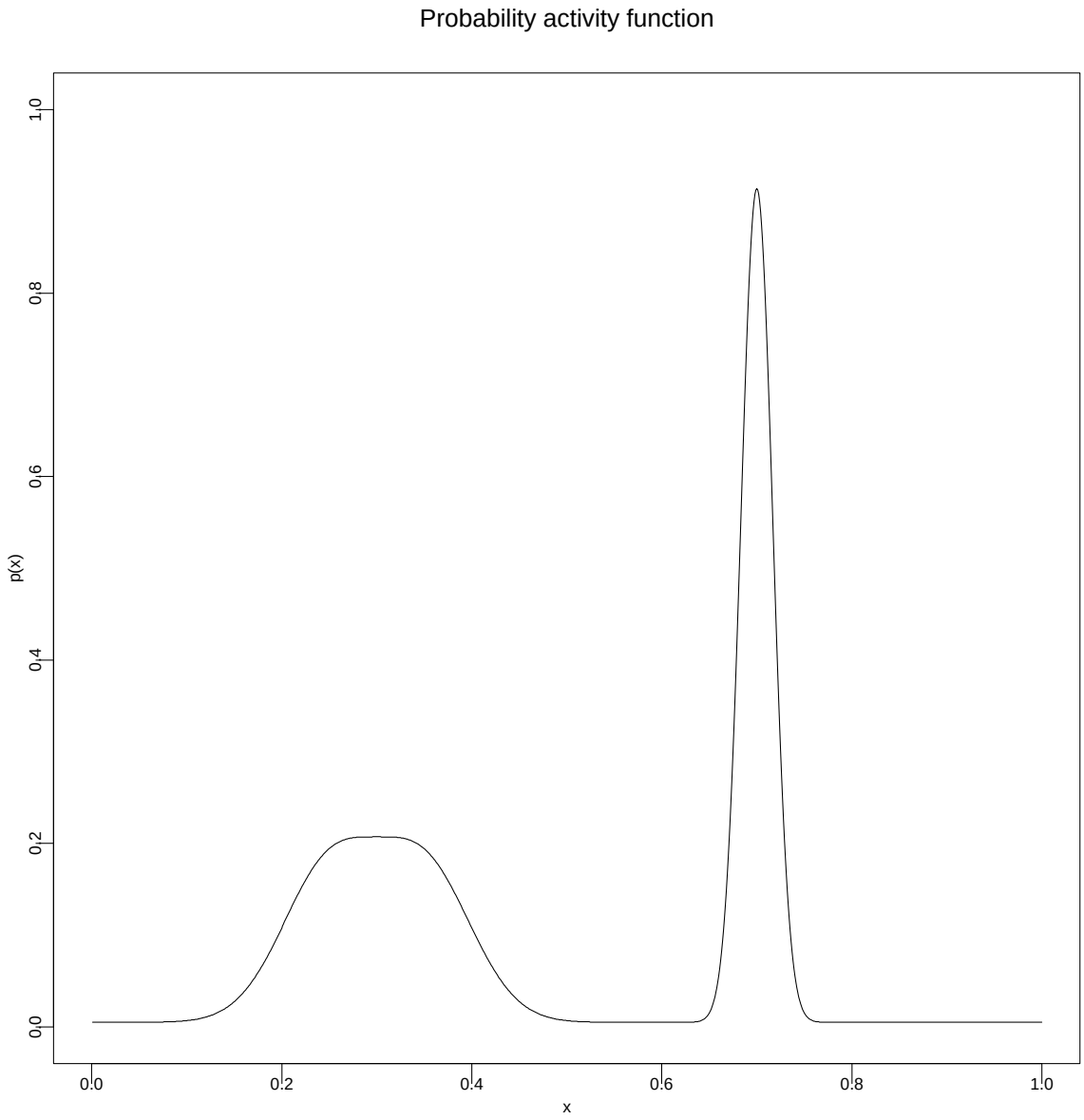

Figure 4.5: The probability of activity function $p(x)$ in the simulation.

The functional form of $p(x)$ is

$$p(x) = \frac{\sqrt{2\pi}}{300} f(x; 0.25, 0.05) + \frac{\sqrt{2\pi}}{120} f(x; 0.35, 0.05) + \frac{\sqrt{2\pi}}{66} f(x; 0.7, 0.017) + 0.005 \qquad (4.2)$$

where $f(x; \mu, \sigma)$ is a normal density function with mean μ and standard deviation σ.

Data are simulated as follows: 300 points x are generated randomly from a uniform distribution on $[0, 1]$ with their Y realizations from independent Bernoulli distributions with probabilities $p(x)$. Over these 300 points, the expected number of actives is 25 with standard deviation 4. The data is randomly divided into the training and test set with equal numbers of active compounds and inactive compounds.

Both KNN and the k-varying algorithm with leave-one-out cross-validation are applied to the training set and make predictions on the test set. In this section, as in the rest of the chapter and the thesis, the value of k in KNN is chosen using AHR. Data simulation and model fitting are repeated 20 times. Figure 4.6 shows hit curves for both KNN (indicated by dotted lines) and the k-varying algorithm (indicated by solid lines) on the test sets for 20 repetitions. The k-varying algorithm is a clear winner several times and often they tie. However, the k-varying algorithm never loses substantially here. In order to have a clear comparison, the average hit curves across 20 iterations for both methods are displayed in Figure 4.7. The k-varying algorithm beats KNN as the average hit curve for the k-varying algorithm is uniformly higher than KNN. We also calculate the AHRs of both methods for 20 iterations and conduct a paired-data one-sided t-test. The p-value (0.0059) suggests the AHR of the k-varying algorithm is statistically significantly bigger than KNN on average.

The scores to rank the test points, $p^-_{0.95}(k)$ from k-varying algorithm and $\hat{p}$ from KNN, can give us some insight why the k-varying algorithm beats KNN. Figure 4.8(a) shows $p^-_{0.95}(k)$ at 3000 test points across 20 simulations and Figure 4.8(b) shows $\hat{p}$ at the same 3000 test points.

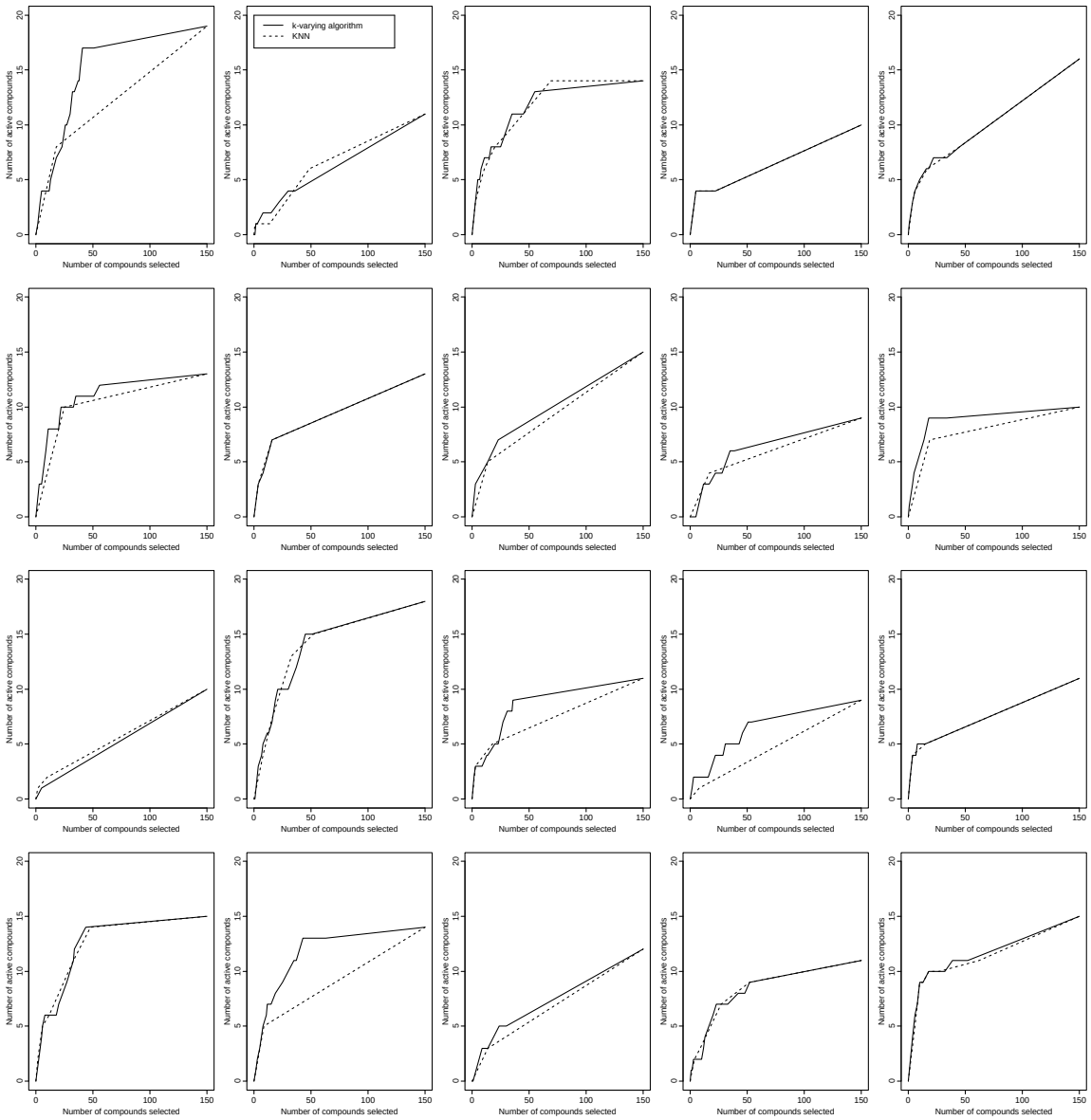

Figure 4.6: The hit curves for both the k-varying algorithm and KNN for 20 simulations. The solid lines indicate hit curves for the k-varying algorithm and the dotted lines indicate hit curves for KNN.

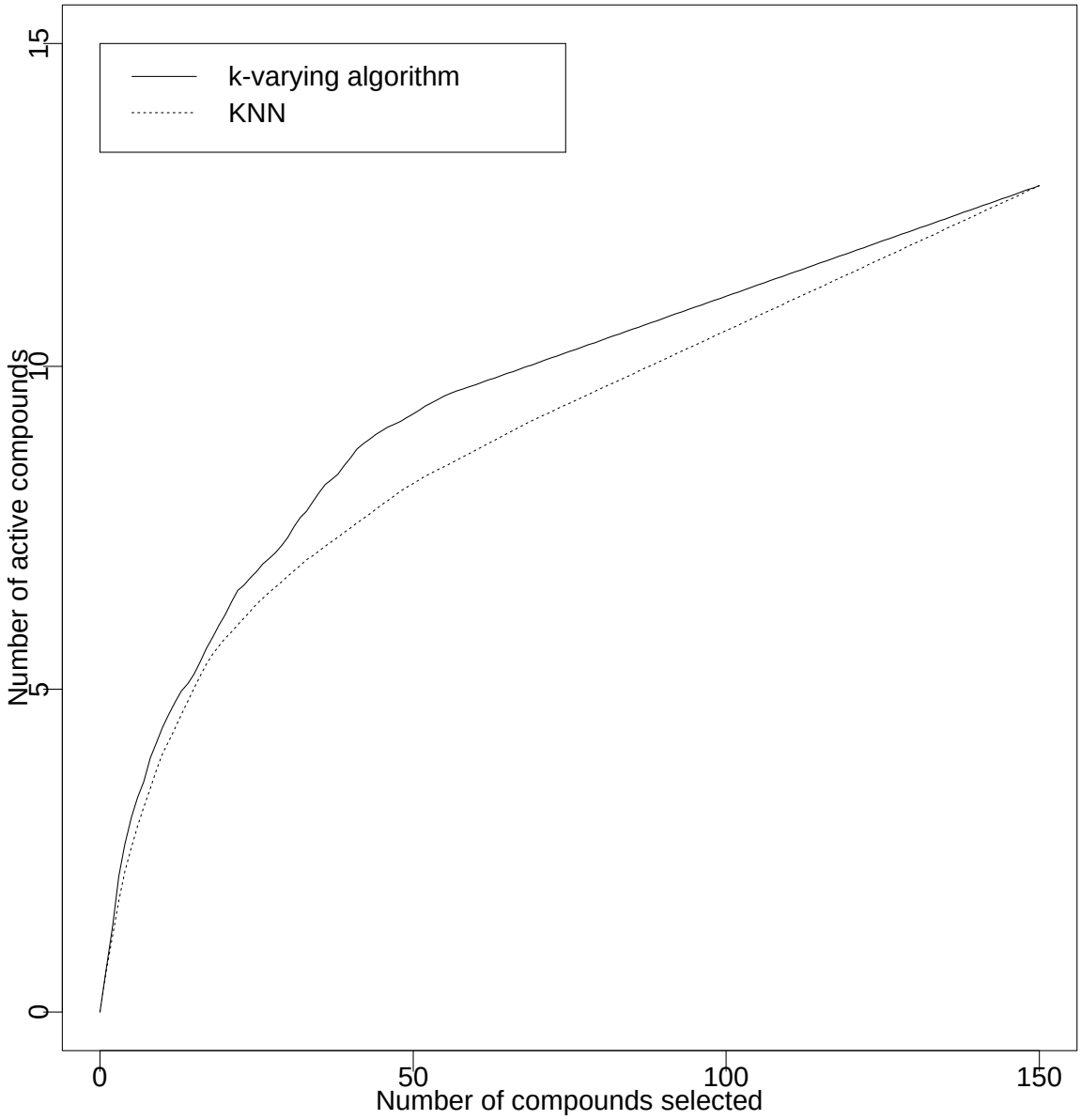

Figure 4.7: The average hit curves for both the k-varying algorithm and KNN across 20 simulations. The solid line indicates the average hit curve for the k-varying algorithm and the dotted line indicates the average hit curve for KNN.

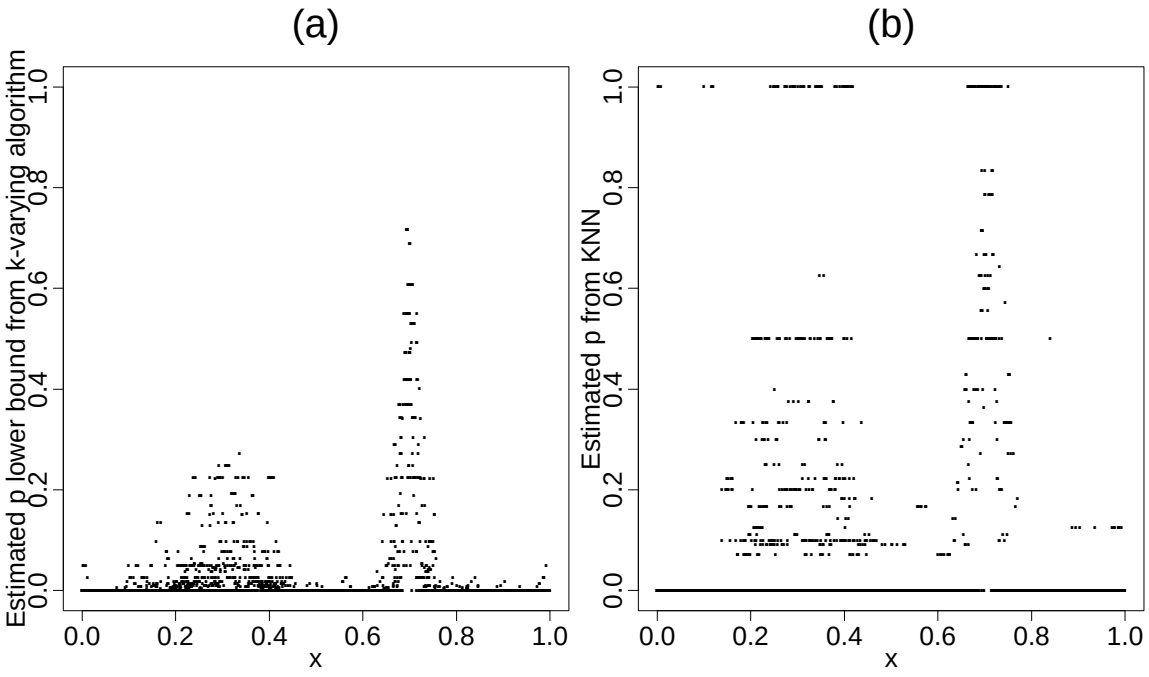

Figure 4.8: The ranking scores for both k-varying algorithm and KNN at 3000 test points across 20 simulations: (a) $p^-_{0.95}(k)$ from k-varying algorithm and (b) $\hat{p}$ from KNN.

The lower bounds $p^-_{0.95}(k)$ are clearly better than the $\hat{p}$ values in capturing the features of the true function $p(x)$ (in Figure 4.5). They are more smooth and the curvature suggested by Figure 4.8(a) is similar to the one in Figure 4.5. While $p^-_{0.95}(k)$ obviously shrink downwards (lower than the true p) at most test points, it may not impact the role of ranking much. In comparison, $\hat{p}$ from KNN is more discrete due to much less choices of k values. Therefore, $p^-_{0.95}(k)$ is able to rank the test compounds better than $\hat{p}$ in these simulated data.

4.4.2 The NCI AIDS data

Now I apply the k-varying algorithm to the NCI AIDS data described in Chapter 1. Figure 4.9 displays the test set hit curves of KNN and the k-varying algorithm for four random

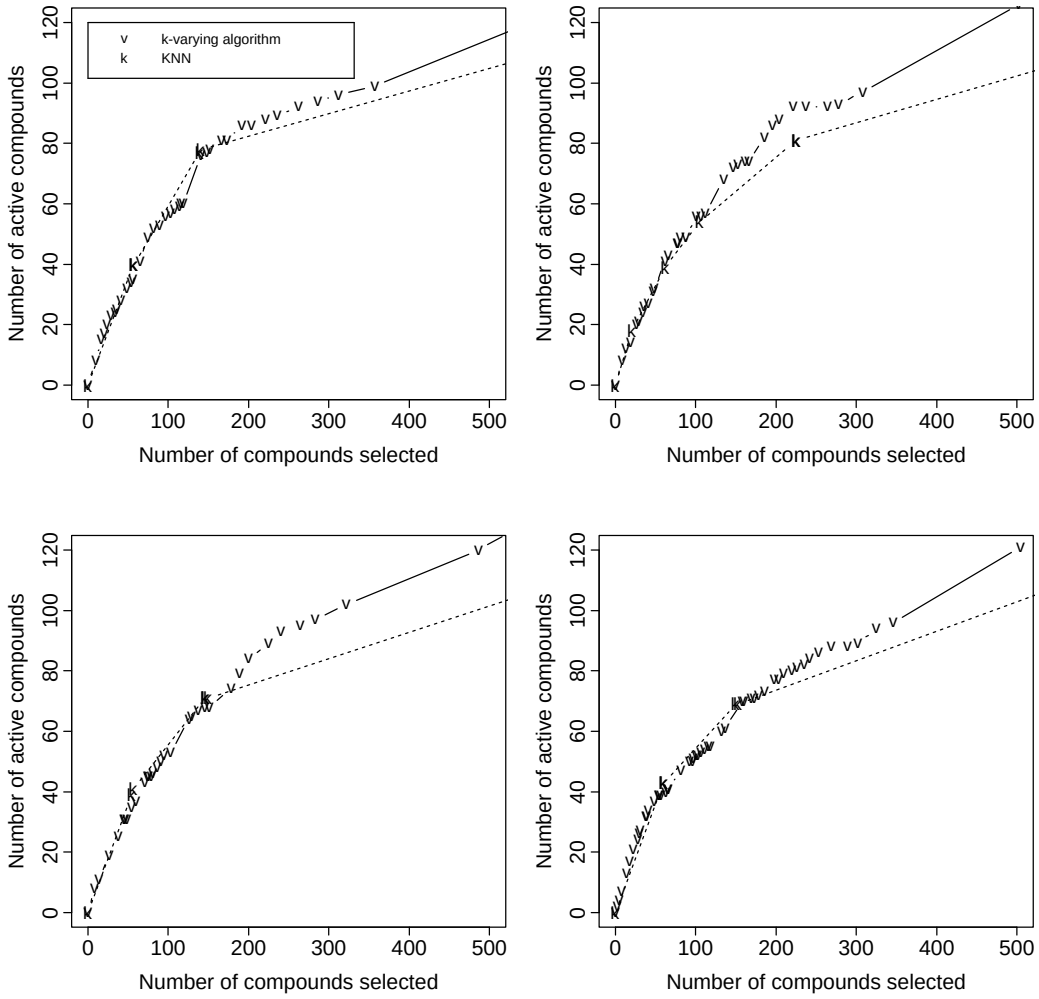

Figure 4.9: Hit curves for KNN and the k-varying algorithm.

splits of the data. The k-varying algorithm has higher hit curves than regular KNN. The hit curves for the two methods are similar for 200 or fewer compounds selected, but the k-varying algorithm gradually surpasses regular KNN afterwards. For instance, when 500 highest-ranked compounds are selected for assay, the k-varying algorithm hits about 15 more actives on average. To determine whether the improvement is significant or not, we conduct a statistical test as in Section 3.4. Table 4.4 lists the AHRs in the test set for four random splits of data provided by both methods. A paired-sample t-test is used, and the alternative hypothesis is that the k-varying algorithm is significantly better than KNN on average:

$$H_0: \qquad \mu_d = 0$$

$$H_a: \qquad \mu_d > 0$$

$$t = \frac{\bar{d}-0}{s_d/\sqrt{4}} = 3.64 \text{ (p-value= 0.018)}$$

As the p-value is very small (0.018), there is strong evidence that the AHR of the k-varying algorithm is statistically significantly larger than that of KNN.

I now examine in greater detail the highest-ranked compounds in the NCI AIDS data selected by the k-varying algorithm and regular KNN. As mentioned in Section 4.1, cross-validation of regular KNN selects a very small value of k to capture the local regions. In the first split of the NCI AIDS data, for instance, $k = 3$ is chosen. Because all equal-distance compounds contribute to the prediction when they happen to be the 3rd neighbours, I have seven instead of four possible values of the estimated probability of activity, which divide the compounds of the test set into seven groups with 57, 1, 81, 2, 614, 7, 14144 compounds,

Table 4.4: AHRs in the test set for four random splits of data provided by regular KNN and the k-varying algorithm.

ith split	AHR(KNN)	AHR(k-varying algorithm)	Difference (d_i)
1	0.2279	0.2490	0.0211
2	0.2121	0.2545	0.0424
3	0.2313	0.2465	0.0152
4	0.2377	0.2530	0.0153

Mean $\overline{d} = 0.0235$

Standard deviation $s_d = 0.0129$

respectively. In each group, the compounds have the same score, and hence random selection within each group is assumed.

The k-varying algorithm selects quite different parameter values. For example, in the first split of the data, the k-varying algorithm selects a ceiling number $t = 9$ by cross-validation of the training data. The scores are composed of 46 distinct values. As a result, 46 groups of compounds instead of 7 (for regular KNN) emerge. The top-ranked smaller groups have higher hit rates. Table 4.5 lists the top five groups of test compounds selected by the k-varying algorithm. There are 29 compounds in those five groups, and the hit rate is 0.79. In comparison, the top group of the regular KNN method has 57 compounds. Of them, 40 are active (hit rate= 0.70). There is considerable overlap between the top 5 k-varying groups and the top KNN group: 26 out of 29 in the top five groups selected by the k-varying algorithm are in the top group selected by the regular KNN method (one of the remaining three compounds is active). Furthermore, all the compounds in the top four groups selected

Table 4.5: The top five groups of compounds selected by the k-varying algorithm.

k-varying algorithm					**Number in**
Group rank	k	Group size	Number active	HR	regular KNN top group
1	9	10	8	0.8	10
2	8	7	7	1	7
3	7	4	2	0.5	4
4	6	3	3	1	3
5	9	5	3	0.6	2
Total		29	23	0.79	26

by the k-varying algorithm are also in the top group selected by KNN. In other words, the regular KNN method ranks highly the same compounds, but their ranks tie with other compounds with a lower hit rate. Except for 4 compounds in the top third group with a hit rate = 0.5, and 5 compounds in the fifth group with a hit rate = 0.6, the other compounds have much better hit rates (as good as 1 in two groups).

It seems that selecting k adaptively refines the descriptor space so that the good active areas stand out. Allowing k to be as large as possible, provided there are active compounds nearby, makes it possible to identify very good active regions with many active compounds. It also provides more reliable ranks by penalizing the small sample of selected neighbours and helps make sure that the scores of the selected top compounds are high and stable.

Table 4.5 illustrates the distribution of k for the 29 compounds in the top five groups selected by the algorithm. The compounds in the first and fifth group have their k's equal to

9 (the largest possible because the ceiling number $t = 9$), and the compounds in the second, third and fourth group have their k's equal to 8,7 and 6, respectively. In comparison, k is fixed to be 3 in the regular KNN method. It cannot pick rather large but remarkable active regions easily with a k of 3. The highly active regions like the second group in Table 4.5 (7 out of 7 are active) cannot stand out when k is fixed to be 3.

4.5 Distance-weighted KNN

As mentioned in Section 4.1, the way that KNN takes an average of the k nearest points is a discrete process. The k nearest compounds are selected to make an equal contribution to the prediction of a test compound, no matter where they are located relative to the test point. In a sparse region, the kth compound may be far away from the test compound and be unrelated to the test compound and have little or no prediction power. However, this compound must contribute to the prediction the same as the other $k-1$ compounds according to the KNN rule. This does not sound reasonable. Here, I endeavour to avoid the problem by combining KNN with kernel weights. Independently of this research, a kernel-weighted KNN was proposed and implemented by Hechenbichler and Schliep (2004). This section is included because in a drug discovery context, the method has some desirable properties.

If weights are considered, the estimated probability of activity (4.1) can be easily changed into

$$\hat{p}(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in N_k(\mathbf{x})} w_i y_i}{\sum w_i}, \tag{4.3}$$

where w_i is a function of the distance between the ith nearest neighbour and the test compound and is usually chosen to be a monotone decreasing function with the distance.

Similar approach can be found in . The only difference is the output: I estimate the probability of activity and they estimate class label.

In (4.3), the probability estimate $\hat{p}(\mathbf{x})$ at a test compound with descriptor values $\mathbf{x}$ takes a weighted average among the selected k nearest points. In this respect, I can automatically give the points close to the test compound more weight and the points far away less weight. This, in some sense, alleviates the bias drawn by including the points far away.

There are several kernel weight functions, and I choose the widely-used Gaussian kernel function, which provides smooth weights:

$$K_G(d, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d^2}{2\sigma^2}\right), \tag{4.4}$$

where d is the Euclidean distance between the compounds and σ is the standard deviation of the corresponding probability density function. The estimated probability of activity is

$$\hat{p}(\mathbf{x}; k, \sigma) = \frac{\sum_{\mathbf{x}_i \in N_k(\mathbf{x})} K_G(d_i, \sigma) y_i}{\sum_{i=1}^{k} K_G(d_i, \sigma)}, \tag{4.5}$$

where d_i is the distance between the compound (x) and its ith neighbour. With this formulation, there are two parameters that need to be chosen: the number of near neighbours, k, and the standard deviation σ in the kernel function. They are chosen by cross-validation.

The algorithm is as follows:

- *Apply leave-one-out cross-validation for each compound on the training set*
  - *Loop through the training compounds. For* $i = 1, 2, \ldots, n_{train}$*,*
    - *Find the* K *nearest points in the training set (*K *is the largest possible number of nearest neighbours examined)*
    - *Loop through some grid values of* σ *(indicated by* Θ*) and values of* k*. For* $\sigma \in \Theta$ *and* $k = 1, 2, \ldots, K$*,*

  - · *Calculate the Gaussian weight between the i-th training compound and its k-th nearest point*
  - · *Calculate the prediction $\hat{p}(x_i; k, \sigma)$ using (4.5) (the weighted average among the k nearest neighbours)*
- *Loop through all values of k and grid values of σ. For $k = 1, 2, \ldots, K$ and $\sigma \in \Theta$,*
  - ∗ *Calculate the AHR for the weighted KNN with the parameter pair (k, σ), and find one pair which maximizes AHR*

- *For the chosen pair (k, σ), perform the weighted KNN on the test compounds.*

Basically, in the cross-validation step, the $\hat{p}_i(k, \sigma)$ is calculated for each i, k, σ combination on the training set, and then an AHR is generated over all i for a fixed pair of k and σ. The pair of k and σ with largest AHR is selected and the weighted KNN method with the chosen k and σ is then applied to the test set.

I applied the distance-weighted KNN to the NCI AIDS data ($K = 15$; $\Theta = \{0.1, 0.2, \ldots, 1.5\}$), and Figure 4.10 compares it to the k-varying algorithm and regular KNN. Note that both the distance-weighted KNN and regular KNN are trained on AHR, the only difference between them is the change associated with weighting.

It seems that both modifications of KNN improve the hit curves and AHRs. We now confirm this with statistical tests similar to those in Section 4.4.2. The same four training/test splits are used here and in Section 4.4.2.

Table 4.6 shows the AHRs in the test set for four random splits of data provided by the weighted KNN and KNN methods. I can test whether the AHR of the weighted KNN is significantly larger than that of KNN on average using a paired-sample t test:

$$H_0: \qquad \mu_d = 0$$

$$H_a: \qquad \mu_d > 0$$

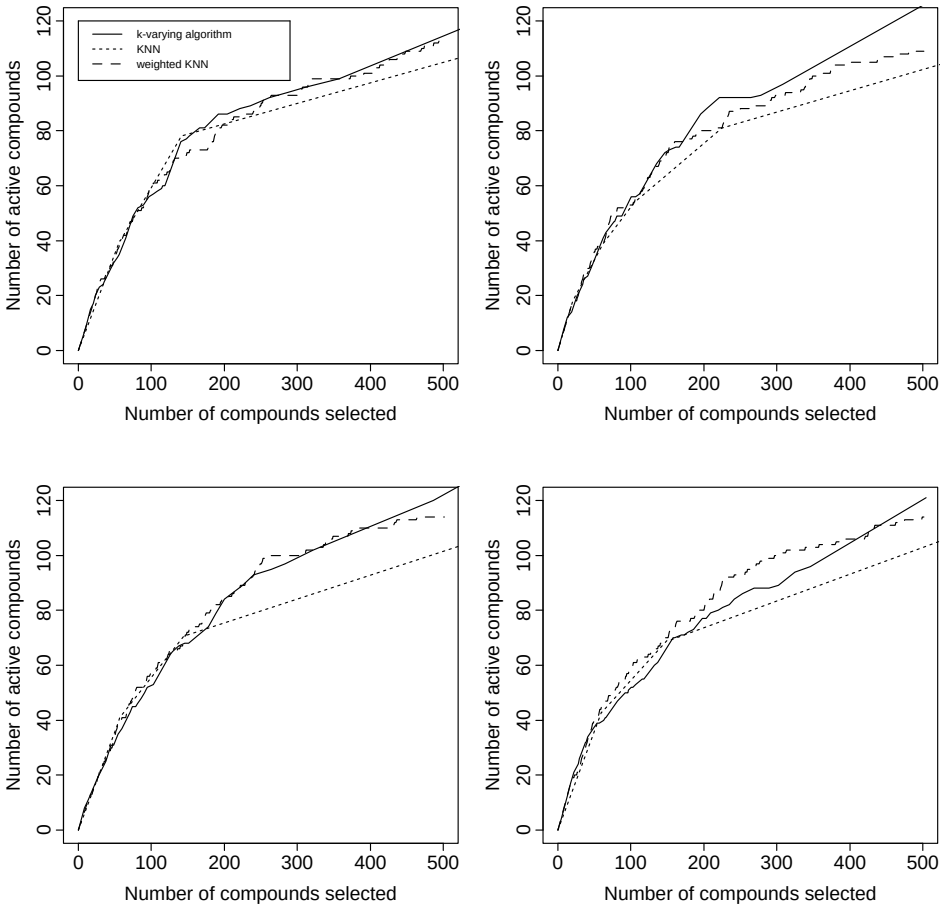

Figure 4.10: Hit curves for the k-varying algorithm, regular KNN, and weighted KNN.

$$t = \frac{\overline{d}-0}{s_d/\sqrt{4}} = 4.44 \text{ (p-value= 0.011)}.$$

Considering the p-value is very small (0.011), there is strong evidence that AHR for the weighted KNN is significantly bigger than for KNN.

Table 4.7 displays the AHRs in the test set for four random splits of data provided by the weighted KNN and the k-varying algorithm. It is also a paired experiment, and I use the following to test whether these two methods are the same:

$$H_0: \qquad \mu_d = 0$$

$$H_a: \qquad \mu_d \neq 0$$

$$t = \frac{\overline{d}-0}{s_d/\sqrt{4}} = -0.13 \text{ (p-value= 0.90)}$$

Since the p value is very large (0.90), there is no evidence to claim that these two methods are statistically significantly different.

I believe the weights in the weighted KNN help to break the ties of the score ($\hat{p}$) and thus improve the performance. However, one drawback is that it is difficult to find the two optimal parameters, k and σ, unless their ranges are known. An alternative is to use all training compounds instead of choosing only the k nearest points and then weight them so that the far-away points contribute little to the prediction:

$$\hat{p}(\mathbf{x}) = \frac{\sum_{all\mathbf{x}} w_i y_i}{\sum w_i}. \tag{4.6}$$

However, enormous computer space and intensive computation are needed to handle the increasing number of distances and weights. It is very inconvenient for a large data set.

Table 4.6: AHRs in the test set for four random splits of data provided by KNN and the weighted KNN methods.

Split	AHR(KNN)	AHR(Weighted KNN)	Difference (d_i)
1	0.2279	0.2447	0.0168
2	0.2121	0.2466	0.0345
3	0.2313	0.2442	0.0129
4	0.2377	0.2704	0.0327

Mean $\overline{d} = 0.0242$

Standard deviation $s_d = 0.0109$

Table 4.7: AHRs in the test set for four random splits of data provided by the weighted KNN and the k-varying algorithm.

Split	AHR(Weighted KNN)	AHR(k-varying algorithm)	Difference (d_i)
1	0.2447	0.2490	0.0043
2	0.2466	0.2545	0.0079
3	0.2442	0.2465	0.0023
4	0.2704	0.253	-0.0174

Mean $\overline{d} = -0.0007$

Standard deviation $s_d = 0.0114$

An alternative to weighted KNN is to estimate the class density using KNN weighted by the distance

$$\hat{f}_j(\mathbf{x_0}) = \frac{1}{K}\sum_{i=1}^{K}\frac{i-1}{N}\frac{1}{v_{ij}}$$

where v_{ij} is the smallest hyper-volume enclosing all the points of the jth class at least as near to x_0 as the ith nearest neighbour (e.g. Parthasarthy & Chatterji, 1990) and then to apply Bayes rule to make the classification (same as kernel density classification explained by Section 6.6.2 of Hastie, Tibshirani and Freidman (2001)).

4.6 Conclusion

Allowing k to vary improves the flexibility of the method and is shown to be superior to the regular KNN algorithm for both a one-dimensional simulated data set and the NCI AIDS data. The ability of weighted KNN to smooth the estimate of $\hat{p}(\mathbf{x})$ also seems promising. The k-varying algorithm is easy to implement and has almost the same computation cost as KNN. However, it is hard to get good insight as the algorithm itself involves different factors such as $p^{-}_{0.95}(k)$ and AHR.

Chapter 5

Identifying and Aggregating Relevant Subsets of Variables

5.1 Introduction

For drug-discovery data, it is possible that only a subset of descriptor variables is relevant to a chemical mechanism causing biological activity (Pearlman and Smith 1999). Moreover, activity might be caused by several different mechanisms, characterized by different subsets of variables. Thus, studying subsets of variables is of practical interest, especially when the descriptor set has huge dimension. This chapter proposes a novel method of averaging across multiple classifiers based on building classifiers on subspaces (subsets of variables). It also suggests diagnostics for identifying important subsets of variables and hence further reducing the impact of the curse of dimensionality.

This chapter is organized as follows. Section 5.2 outlines the proposed method based on aggregating classifiers from subsets of explanatory variables and compares it with other

aggregation methods, namely bagging and boosting. In Section 5.3 it is shown that the proposed method copes well with weak classifiers that might be among those aggregated, and in Section 5.4 tools for distinguishing weak and strong classifiers are proposed. Finally, conclusions are drawn in Section 5.5.

5.2 Aggregating classifiers built from subsets of variables

Now I propose building multiple classifiers from subsets of variables and aggregating them into an overall classifier. This approach takes advantage of the idea of multiple mechanisms causing activity in drug discovery (subsets of variables) and can also be applied to almost any classification method.

Chapter 2 demonstrates that KNN is a competitive classifier for the NCI AIDS data. This is the motivation to apply the subsets averaging idea to KNN first. For example, the NCI AIDS data have $\binom{6}{1} = 6$ one-dimensional subsets, $\binom{6}{2} = 15$ two-dimensional subsets, and so on up to $\binom{6}{6} = 1$ six-dimensional subset. In total, there are 63 subsets and 63 KNN classifiers are built. For each of the 63 possible subsets of variables, leave-one-out cross-validation on the training set is used to choose a value for k, the number of nearest neighbours in KNN. The value of k chosen for each subset optimizes the AHR criterion. (This is consistent with the implementation of the regular (fixed k) KNN method in Chapter 4.) Every compound in the test data then has 63 estimates of the probability of activity. For now, the 63 estimates are averaged (with equal weights) to provide a single estimate for each test compound. The estimates or scores from the aggregated classifier generate a hit curve (symbol "S" for subset KNN in Figure 5.1). Subset KNN substantially outperforms the regular KNN method using

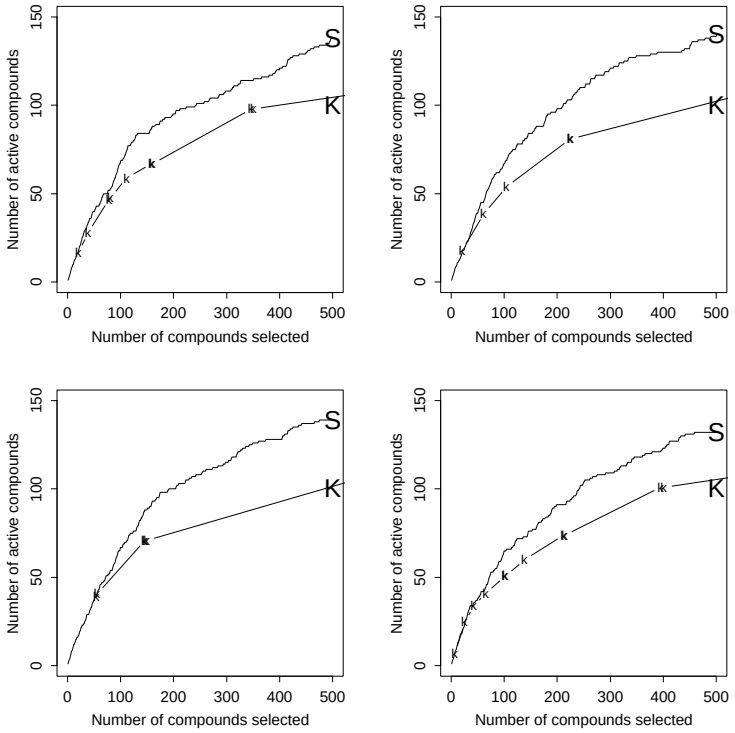

Figure 5.1: Hit curves from four random training/test data splits of the NCI AIDS data for subset KNN (averaged over all 63 subsets of variables) with symbol "S" and KNN (six variables) with symbol "K".

all six descriptor variables (symbol "K" in Figure 5.1). The two methods are tried for the same four random training/test data splits used in earlier chapters. Averaged over the four splits, subset KNN finds 31% more actives than regular KNN when 500 compounds are selected.

The same idea can be applied to classification trees (default trees in Splus). Figure 5.2 shows that averaging classification trees across the 63 subsets of variables (the subset tree) outperforms the single tree. Again, the improvement is consistent over the four random splits of the data. On average, the subset tree finds 37% more actives when 500 are selected than a single tree based on all the variables. In Figure 5.2, S-plus default trees are used, as in Section 2.4.1. A comparison of Figures 5.1 and 5.2 reveals that subset KNN performs slightly better than subset tree.

Existing methods of aggregating classifiers include bagging (Breiman 1996) and boosting (e.g. Freund and Schapire 1997 and Friedman 2001, 1999). Both methods have drawn a great deal of attention by providing more stable and accurate predictions than a single model.

Bagging simply involves drawing random samples with replacement from the original data, say, 100 times. Each bootstrap sample has the same size as the original data. For each sample, a statistical model is built, and then the final prediction is a function of the prediction from each model. The mean function is commonly used to combine predictions.

Boosting is described as "one of the most powerful learning ideas introduced in the last ten years" in the book of Hastie, Tibshirani, and Friedman (2001, p299). It can be applied to any learning algorithm for improving predictive accuracy. In general, boosting builds an additive model on a set of elementary "basis" functions or, more specifically, some individual

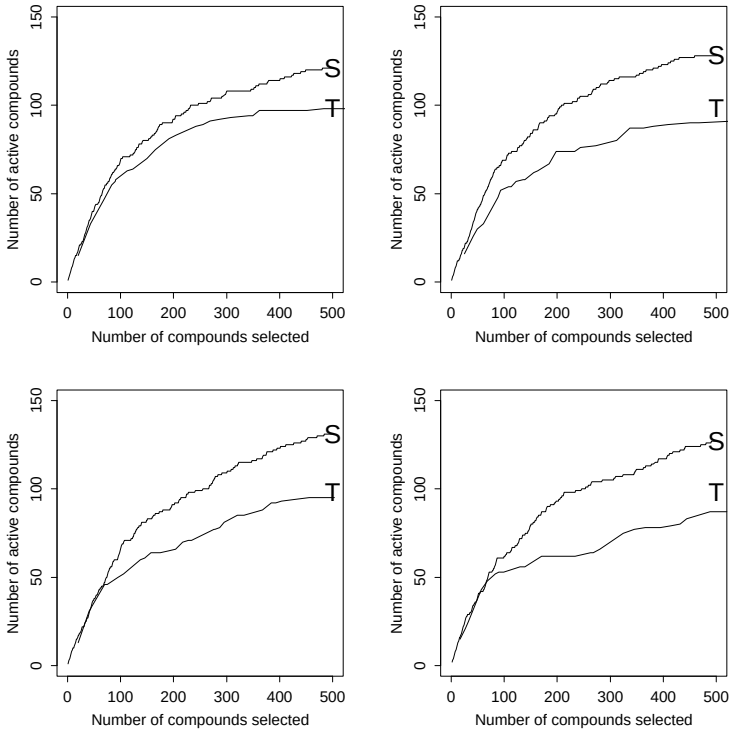

Figure 5.2: Hit curves from four random training/test splits of the NCI AIDS data for the subset tree (averaged over all 63 subsets of variables) with symbol "S" and tree (six variables) with symbol "T".

classifiers. When applied to trees, the model (boosted tree) is a sum of many simple trees:

$$f_T(\mathbf{x}) = \sum_{m=1}^{M} T_m(\mathbf{x}; \gamma_m, R_m) \tag{5.1}$$

where

$$T_m(\mathbf{x}; \gamma_m, R_m) = \sum_{i}^{I_m} \gamma_{mi} I(\mathbf{x} \in R_{mi}) \tag{5.2}$$

and $R_{mi}, i = 1, 2, \ldots, I_m$, are disjoint regions in X space. A constant γ_{mi} is assigned to each such region. In terms of tree models, R_{mi} is the ith terminal node in tree m and it has fitted value of γ_{mi}.

Ideally, γ_{mi} and R_{mi} are fitted by minimizing a loss function:

$$\min_{\{\gamma_m\, R_m\}_1^M} \sum_{j=1}^{N} L(y_j, \sum_{m=1}^{M} T_m(\mathbf{x}_j; \gamma_m, R_m)). \tag{5.3}$$

The multiple additive regression trees (MART) implementation of the boosting algorithm (Friedman 2001) is used here. The loss function for the classification version of MART is negative binomial log-likelihood (Friedman 2000)

$$L(y, F) = \log(1 + \exp(-2yF)), \qquad y \in \{-1, 1\}, \tag{5.4}$$

where

$$F(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2}} \log \Big[\frac{\Pr(\mathbf{Y} = \mathbf{1}|\mathbf{x})}{\Pr(\mathbf{Y} = -\mathbf{1}|\mathbf{x})} \Big]. \tag{5.5}$$

Due to high computation, the solution to (5.3) in MART is approximated by iteratively adding a single tree at a time to the expansion without adjusting the parameters of those that have already been included. That is, when adding tree $k + 1$, we minimize

$$\sum_{j=1}^{N} L(y_j, \sum_{m=1}^{k} T_m(\mathbf{x}_j; \gamma_m, R_m) + T_{k+1}(\mathbf{x}_j; \gamma_{k+1}, R_{k+1})) \tag{5.6}$$

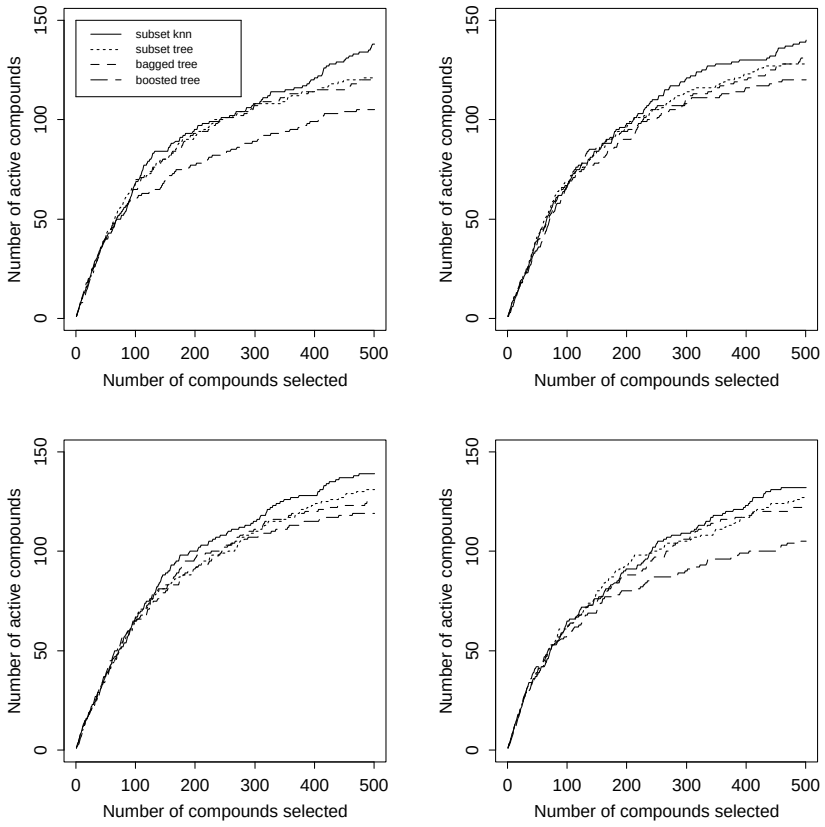

Figure 5.3: Hit curves from four random training/test splits of the NCI AIDS data for subset KNN, subset tree, boosted tree, and bagged tree classifiers.

as a function of γ_{k+1} and R_{k+1}, holding $\gamma_1, \ldots, \gamma_k$ and $R_1, \ldots, R_k$ fixed. After M iterations, (5.6) will have final form (5.3).

At the time this research was conducted MART software was available from `http://www-stat.stanford.edu/~jhf/`, including an R interface. In my thesis, the default options of MART are applied.

Figure 5.3 compares the hit curves for subset KNN, subset tree, bagged tree, and boosted

Table 5.1: AHRs for subset KNN, subset tree, bagged tree, and boosted tree.

	1st split	2nd split	3rd split	4th split	average(sd)
Subset KNN	0.3173	0.3272	0.3221	0.3117	0.3196(0.0066)
Subset tree	0.3107	0.3200	0.3003	0.3067	0.3094(0.0082)
Bagged tree	0.3151	0.3058	0.3023	0.3008	0.3061(0.0064)
Boosted tree	0.2583	0.2408	0.2488	0.2263	0.2435(0.0136)

Table 5.2: Pairwise significance tests among the selected methods: subset KNN, subset tree, bagged tree and boosted tree.

Classifier 1	Classifier 2	$\overline{d}$	s_d	t statistics	p value
Subset KNN	Bagged tree	0.0136	0.0088	3.06	0.027
Subset KNN	Boosted tree	0.0760	0.0128	11.86	0.00064
Subset tree	Bagged tree	0.0034	0.0084	0.81	0.24
Subset tree	Boosted tree	0.0659	0.0161	8.19	0.0019
Subset KNN	Subset tree	0.01015	0.0078	2.59	0.040

tree classifiers for four random training/test splits of the data. It shows that aggregation over subsets of variables outperforms bagging and boosting. Table 5.1 lists AHRs for the four methods applied to the four random splits.

The pairwise hypothesis tests (paired t-tests) comparing the four classifiers are given in Table 5.2. Section 3.4 contains details of similar tests. The alternative hypothesis is whether the AHR for "Classifier 1" is larger than that for "Classifier 2" on average.

In summary, based on AHR, there is a statistically significant positive difference when

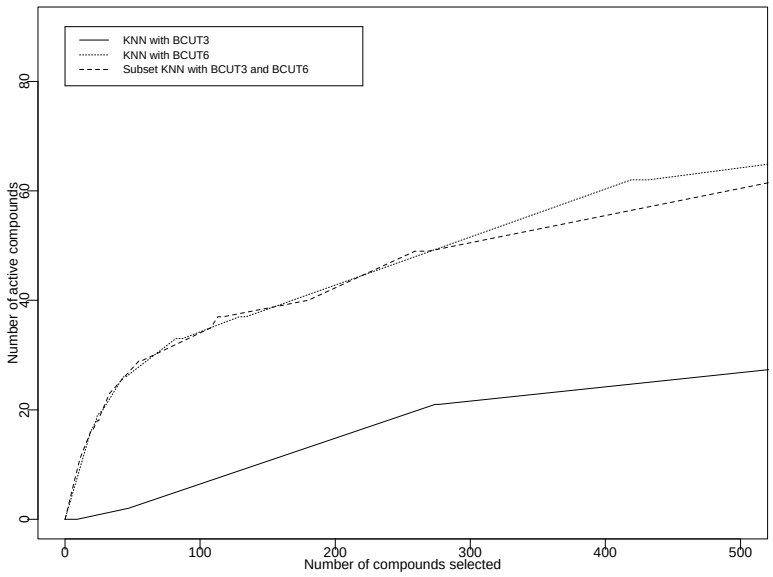

Figure 5.4: Hit curves from the first split of the NCI AIDS data for classifiers based on: KNN using only BCUT3 (solid line); KNN using only BCUT6 (dotted line) and averaging both classifiers (dashed line).

subset KNN is compared with subset tree, with bagged tree, or with boosted tree. There is no strong evidence that the subset tree outperforms the bagged tree. Subset trees do appear to be superior to the boosted tree.

It is perhaps surprising that aggregating KNN over all subsets is so effective. Some, possibly most, subsets lead to weak classifiers. Figure 5.4 illustrates, however, that such weak classifiers do not necessarily harm the stronger classifiers based on important variables. If only one subset (BCUT6) is used, KNN provides a fairly good hit curve (AHR=0.131), whereas BCUT3 has a poor hit curve (AHR=0.030). When these two classifiers are averaged, the overall performance (AHR=0.138) is about the same as the good one. Averaging such weak classifiers does not necessarily harm the stronger classifiers based on important variables. In the next section, it is explained in detail and some theoretic justification is also provided. Section 5.4 suggests a way to choose the good subsets to further improve the

model performance.

5.3 Weak and strong classifiers and the curse of dimensionality

Subset averaging may deal well with many weak variables, that is, it is not greatly affected by the curse of dimensionality. To illustrate this concept, I construct a new data set with some predictors that are known to be irrelevant. Six irrelevant explanatory variables are added to the original six in the NCI AIDS data. The values of the new variables, BCUT7, ..., BCUT12, are generated from independent random permutations of the values of BCUT1, ..., BCUT6, respectively. Thus, the values of BCUT7, ..., BCUT12 are also randomly permuted with respect to the values of the response variable and are irrelevant by construction. These data will be referred to as the "augmented NCI data".

KNN based on a 12-dimensional distance metric using all 12 explanatory variables has a very poor hit curve, as shown by the short dashed line in Figure 5.5. Its hit curve is much lower than that for KNN with the original six BCUT descriptors (dotted line). It is unable to deal with a high-dimensional space of explanatory variables; that is, it suffers from the curse of dimensionality. The figure relates to the first training/test split of the data, the other splits give similar results. In contrast, averaging over the 78 KNN classifiers from all subsets of one or two variables from the the augmented NCI data (12 variables in total) yields a substantially better hit curve: the solid line in Figure 5.5. It is obviously higher than KNN with the original six BCUT numbers. Here, most subsets have one or two irrelevant variables, yet subset averaging produces a hit curve nearly as good as the subset averaging method using only the six original, presumably informative, variables (subset KNN indicated

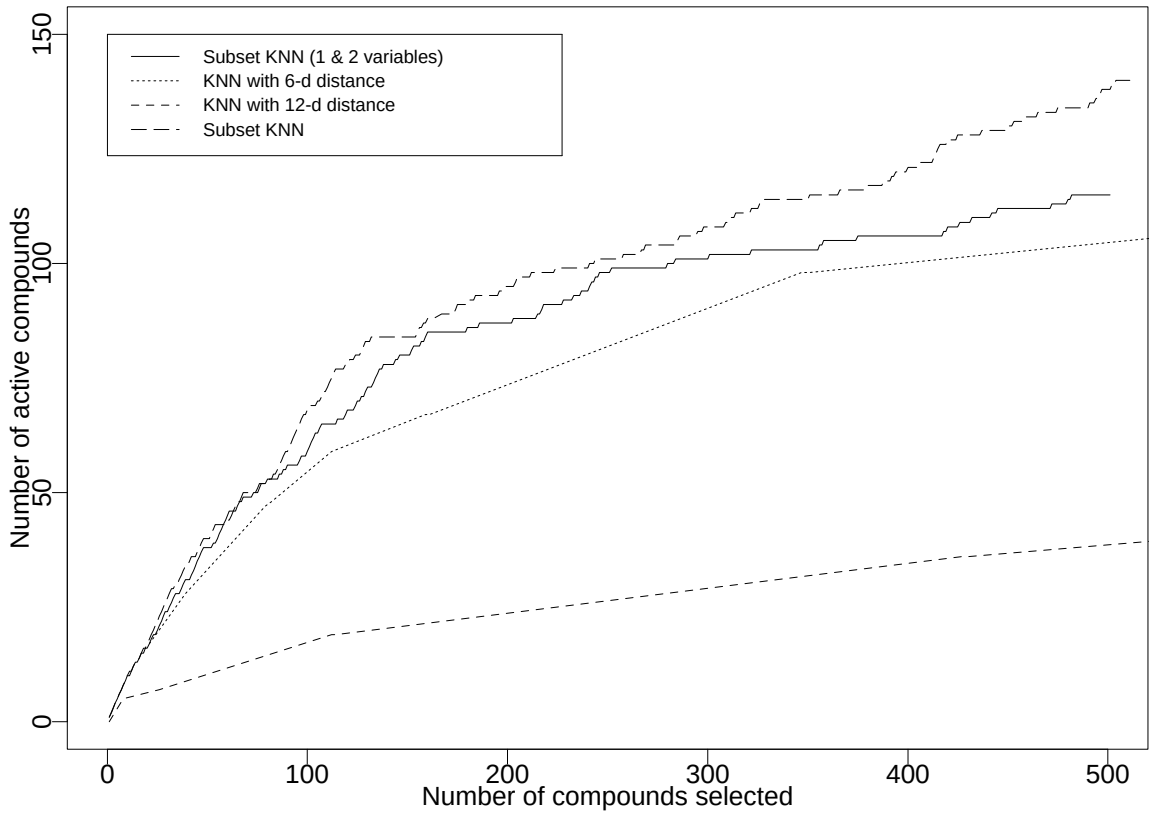

Figure 5.5: Hit curves for KNN classifiers based on 12 variables (augmented NCI data): aggregate of classifiers from all subsets of one or two variables (solid line), KNN with 12-dimensional distance metric (short dashed line) and subset KNN (long dashed line). In comparison, KNN with the original six BCUT descriptors is also shown (dotted line).

by the long dashed line). Indeed, it substantially outperforms KNN with only the original variables and a six-dimensional metric (dotted line). Note that only one dimensional and two dimensional subsets are considered due to the intensive computation. However, this limitation does not affect the performance much on the NCI AIDS data since the hit curve of subset KNN on 1- and 2-dimensional subspaces of the 12-dimensional space is close to the hit curve of subset KNN on the 6-dimensional space.

Strong and weak subsets and classifiers are mentioned frequently in the previous sections. Now let us look at these terms in detail. Note that the following derivation can be applied to all the averaged methods not only to subset averaging I describe previously.

A weak subset of variables $\mathbf{x}$ has a fairly flat $p(\mathbf{x})$ function. That is, the variables do not provide much information about $Y = 0$ versus $Y = 1$. The classifiers built using only these variables are likely to be very weak.

Conversely, a strong subset has a probability function $p(\mathbf{x})$ that is not flat across the $\mathbf{x}$ space. Ideally, $p(\mathbf{x})$ varies a lot. For the best cases, $p(\mathbf{x}) = 0$ *or* 1. Zero Bayes risk and a perfect AHR are possible based on such a $p(\mathbf{x})$. A classifier based on these variables could be strong or weak. The classifier needs to **estimate** $p(\mathbf{x})$ well to be strong in general or rank the compounds well to be strong in our case. Therefore, a set of relevant variables is a necessary but not a sufficient condition for a strong classifier.

Suppose $\mathbf{t}_1$ and $\mathbf{t}_2$ are the explanatory variables used for ranking two test compounds and

$$p(\mathbf{t}_1) - p(\mathbf{t}_2) > 0.$$

Thus, $\mathbf{t}_1$ should be ranked ahead of $\mathbf{t}_2$. Similarly, if

$$p(\mathbf{t}_1) - p(\mathbf{t}_2) < 0,$$

$\mathbf{t}_1$ should be ranked after $\mathbf{t}_2$. If

$$p(\mathbf{t}_1) - p(\mathbf{t}_2) = 0,$$

$\mathbf{t}_1$ and $\mathbf{t}_2$ have the same ranks. Now let us look at weak and strong classifiers in terms of ranking.

Consider a classifier, $\widehat{p}$ (strong or weak or an aggregate), and its performance in ranking the points, $\mathbf{t}_1$ and $\mathbf{t}_2$, when $p(\mathbf{t}_1) - p(\mathbf{t}_2) > 0$, say. The probability of a correct ranking depends on the distribution of $\widehat{p}(\mathbf{t}_1) - \widehat{p}(\mathbf{t}_2)$. We will use the signal-to-noise ratio

$$SN[\widehat{p}(\mathbf{t}_1) - \widehat{p}(\mathbf{t}_2)] = \frac{E[\widehat{p}(\mathbf{t}_1) - \widehat{p}(\mathbf{t}_2)]}{sd[\widehat{p}(\mathbf{t}_1) - \widehat{p}(\mathbf{t}_2)]}$$

as a measure of performance. If $SN[\widehat{p}(\mathbf{t}_1) - \widehat{p}(\mathbf{t}_2)]$ is large and positive, there is a large probability of ranking the point $\mathbf{t}_1$ ahead of $\mathbf{t}_2$.

For a weak classifier, $\widehat{p}_w$, $E[\widehat{p}_w(\mathbf{t}_1) - \widehat{p}_w(\mathbf{t}_2)]$ is small relative to $sd[\widehat{p}_w(\mathbf{t}_1) - \widehat{p}_w(\mathbf{t}_2)]$. For simplicity below we take

$$E[\widehat{p}_w(\mathbf{t}_1) - \widehat{p}_w(\mathbf{t}_2)] = 0 \qquad \text{and} \qquad sd[\widehat{p}_w(\mathbf{t}_1) - \widehat{p}_w(\mathbf{t}_2)] = \sigma_w. \tag{5.7}$$

Similarly, for a strong classifier, $\widehat{p}_s$, $E[\widehat{p}_s(\mathbf{t}_1) - \widehat{p}_s(\mathbf{t}_2)]$ is assumed to be $d_s > 0$ below and $sd[\widehat{p}_s(\mathbf{t}_1) - \widehat{p}_s(\mathbf{t}_2)] = \sigma_s$.

Now consider averaging S strong classifiers $\widehat{p}_{sj}$ for $j = 1, 2, \ldots, S$ and W weak classifiers $\widehat{p}_{wj}$ for $j = 1, 2, \ldots, W$ to give an aggregate classifier, $\widehat{p}$:

$$\widehat{p}(\mathbf{t}) = \frac{\sum_{j=1}^{S} \widehat{p}_{sj}(\mathbf{t})}{(S+W)} + \frac{\sum_{j=1}^{W} \widehat{p}_{wj}(\mathbf{t})}{(S+W)}. \tag{5.8}$$

To compute its signal-to-noise ratio, we need

$$E(\widehat{p}(\mathbf{t_1}) - \widehat{p}(\mathbf{t_2})) = \frac{S}{S+W}\overline{d_s} + \frac{W}{S+W}0 \text{ (see 5.7)} \tag{5.9}$$

where $\overline{d_s}$ is the average of $E(\widehat{p}_{sj}(\mathbf{t_1}) - \widehat{p}_{sj}(\mathbf{t_2}))$ with $j = 1, 2, \ldots, S$, and

$$V(\widehat{p}(\mathbf{t_1}) - \widehat{p}(\mathbf{t_2})) = \frac{1}{(S+W)^2}(\sum_{j=1}^{S}\sigma_{sj}^2 + \sum_{j=1}^{W}\sigma_{wj}^2 + 2\sum_{i=1}^{S-1}\sum_{j=i+1}^{S}\sigma_{s,ij} + 2\sum_{i=1}^{W-1}\sum_{j=i+1}^{W}\sigma_{w,ij} + 2\sum_{i=1}^{S}\sum_{j=1}^{W}\sigma_{sw,ij}) \tag{5.10}$$

where

$$\begin{aligned}
\sigma_{sj}^2 &= V(\widehat{p}_{sj}(\mathbf{t}_1) - \widehat{p}_{sj}(\mathbf{t}_2)),\\
\sigma_{wj}^2 &= V(\widehat{p}_{wj}(\mathbf{t}_1) - \widehat{p}_{wj}(\mathbf{t}_2)),\\
\sigma_{s,ij} &= cov(\widehat{p}_{si}(\mathbf{t}_1) - \widehat{p}_{si}(\mathbf{t}_2), \widehat{p}_{sj}(\mathbf{t}_1) - \widehat{p}_{sj}(\mathbf{t}_2)),\\
\sigma_{w,ij} &= cov(\widehat{p}_{wi}(\mathbf{t}_1) - \widehat{p}_{wi}(\mathbf{t}_2), \widehat{p}_{wj}(\mathbf{t}_1) - \widehat{p}_{wj}(\mathbf{t}_2)),\\
\sigma_{sw,ij} &= cov(\widehat{p}_{si}(\mathbf{t}_1) - \widehat{p}_{si}(\mathbf{t}_2), \widehat{p}_{wj}(\mathbf{t}_1) - \widehat{p}_{wj}(\mathbf{t}_2)).
\end{aligned}$$

Here, expectation and variance are with respect to different training samples (different Y realizations). The equation (5.10) can be further simplified as

$$V(\widehat{p}(\mathbf{t_1}) - \widehat{p}(\mathbf{t_2})) = \frac{1}{(S+W)^2}(S\overline{\sigma_s^2} + W\overline{\sigma_w^2} + S(S-1)\overline{\sigma}_{ss} + W(W-1)\overline{\sigma}_{ww} + 2SW\overline{\sigma}_{sw}) \tag{5.11}$$

where

$$\begin{aligned}
\overline{\sigma_s^2} &= \frac{\sum_{j=1}^{S}\sigma_{sj}^2}{S},\\
\overline{\sigma_w^2} &= \frac{\sum_{j=1}^{W}\sigma_{wj}^2}{W},\\
\overline{\sigma}_{ss} &= \frac{\sum_{i=1}^{S-1}\sum_{j=i+1}^{S}\sigma_{s,ij}}{\frac{S(S-1)}{2}},\\
\overline{\sigma}_{ww} &= \frac{\sum_{i=1}^{W-1}\sum_{j=i+1}^{W}\sigma_{w,ij}}{\frac{W(W-1)}{2}},\\
\overline{\sigma}_{sw} &= \frac{\sum_{i=1}^{S}\sum_{j=1}^{W}\sigma_{sw,ij}}{SW}.
\end{aligned}$$

Now letting

$$\overline{\rho}_s = \frac{\overline{\sigma}_{ss}}{\overline{\sigma_s^2}},$$
$$\overline{\rho}_w = \frac{\overline{\sigma}_{ww}}{\overline{\sigma_w^2}},$$

the signal-to-noise ratio for the averaged classifier is

$$SN = \frac{E(\widehat{p}(\mathbf{t_1}) - \widehat{p}(\mathbf{t_2}))}{sd(\widehat{p}(\mathbf{t_1}) - \widehat{p}(\mathbf{t_2}))} = \frac{\overline{d_s}}{\sqrt{\frac{\overline{\sigma}_s^2}{S}(1 + \overline{\rho}_s(S-1)) + \frac{W\overline{\sigma}_w^2}{S^2}(1 + \overline{\rho}_w(W-1)) + \frac{2W}{S}\overline{\sigma}_{sw}}}. \quad (5.12)$$

The magnitude of the signal-to-noise ratio depends on how good the individual strong classifier is ($\overline{d_s}$, $\overline{\sigma_s^2}$), on variation in weak classifiers rankings of the points ($\overline{\sigma_w^2}$), the correlation among the strong and weak classifiers ($\overline{\rho}_s$, $\overline{\rho}_w$ and $\overline{\sigma}_{sw}$) and the number of strong and weak classifiers (S and W). In general,

1. The strength of the individual strong classifier affects the accuracy of the ranking. E.g. if $\overline{d_s}$ increases or $\overline{\sigma_s^2}$ decreases, SN increases;

2. The dependence among weak or strong or weak and strong classifiers affects the accuracy of the ranking. E.g. if any of the positive covariance terms decrease, SN increases.

We now consider different assumptions about the correlations and the impact on the signal-to-noise ratio. In both cases we assume $\overline{\sigma}_{sw} = 0$. Note that the assumption is satisfied when the strong classifiers are uncorrelated with weak classifiers.

Firstly, I want to prove $\overline{\rho}_s \leq 1$ and $\overline{\rho}_w \leq 1$. It is shown that

$$Cov(X, Y) \leq (V(X) + V(Y))/2, \quad (5.13)$$

which is derived from $V(X-Y) \geq 0$. $Cov(X,Y) = (V(X)+V(Y))/2$ only happens when $X = Y + \text{constant}$.

Then, relating to our setting about the covariances

$$\overline{\sigma}_{ss} = \frac{\sum_{i=1}^{S-1}\sum_{j=i+1}^{S}\sigma_{s,ij}}{\frac{S(S-1)}{2}} \leq \frac{\sum_{j=1}^{S}\sigma_{sj}^2}{S} = \overline{\sigma_s^2}. \tag{5.14}$$

That is, $\overline{\sigma}_{ss} \leq \overline{\sigma_s^2}$ and $\overline{\rho}_s \leq 1$. When all the strong classifiers are the same or differ by a constant, $\overline{\rho}_s = 1$.

Similarly, $\overline{\rho}_w \leq 1$. When all the weak classifiers are the same or differ by a constant, $\overline{\rho}_w = 1$.

1. In the best scenario, $\rho_s = 0$ and $\rho_w = 0$ (here we assume that negative average correlation is not likely in practice). The typical situation could be all the weak classifiers independent from each other and all strong classifiers independent from each other. Then,

   $$SN = \frac{\overline{d_s}}{\sqrt{\frac{1}{S}\overline{\sigma_s^2} + \frac{W}{S^2}\overline{\sigma_w^2}}}. \tag{5.15}$$

   As more and more subsets are considered, SN does not decrease providing W increases no faster than the square of S.

2. In the worst scenario, $\rho_s = 1$ and $\rho_w = 1$. That is, all the weak classifiers are the same or differ by a constant and all the strong classifiers are the same or differ by a constant as well.

   Thus, the signal-to-noise ratio is

   $$SN = \frac{\overline{d_s}}{\sqrt{\overline{\sigma_s^2} + \frac{W^2}{S^2}\overline{\sigma_w^2}}}. \tag{5.16}$$

As more and more subsets are considered, SN does not decrease providing S increases linearly with W.

The argument that the strength and dependency of the classifiers impact on the final performance of the aggregated model is also addressed in the context of random forests by Breiman (2001). Here, as in that paper, the best performance arises when classifiers are not strongly correlated and we have sufficient strong classifiers. However, our approaches are totally different. I look at the ranking problem and care about the accuracy of ranking while Breiman (2001) studies the classification problem and minimizes the classification errors.

In addition, I show that the number of strong and weak classifiers affects the accuracy of the ranking relating to the dependency among those classifiers. As more and more subsets are considered (either by considering more subset sizes or by introducing more variables), more and more weak classifiers are probably introduced. But even in the worst case, performance is not necessarily impaired. In practice, we are probably somewhere between the worst and best cases, and the number of weak classifiers can increase faster than linear in the number of strong classifiers without substantially degrading the performance of the strong classifiers alone. Note that the above derivation is general in the sense that the conclusions can be applied to all the averaged methods (formulated by 5.8) which are interested in the ranking problem and care about the signal-to-noise ratio (formulated by 5.9). The subset averaging method I described previously is just one application.

5.4 Identifying useful variables

Identifying important subsets of variables is of scientific interest since these subsets may relate to the characteristics of the chemical structure leading to activity. For this purpose, I

now illustrate how to choose k for each subset and identify the important subsets. For these objectives, I will show for the NCI AIDS data that the AHR is superior to the empirical MSE of the estimated probability of activity for identifying important subsets.

In the thesis, the argument has been frequently made that conventional criteria do not appear to be good in terms of ranking. For example, Section 2.5 shows that deviance does not pick good trees for ranking. Chapter 3 also provides further exploration of the discrepancy between the misclassification rate and hit curves. This section reinforces this argument by showing that the empirical MSE for a regular KNN method is not as effective as AHR for the identification of useful variables. The NCI AIDS data are used to make this argument.

First, I need to explain how the number of neighbours, k, in KNN is chosen for any given subset of variables. Using leave-one-out cross-validation on the training data, I compute the AHR (a summary of the hit curve) as a function of k and choose the value of k giving the largest AHR. Figure 5.6 draws AHR versus k for various subsets of explanatory variables in the augmented NCI data.

In Figure 5.6 (a), cross validation suggests that $k = 22$ is optimal for the subset BCUT4, yielding an AHR of around 0.1. The other three pictures in Figure 5.6 have the same scale but relate to different subsets. Figure 5.6 (b) is the analogous plot for the subset with two important variables (BCUT4 and BCUT6). The optimal AHR is much larger than the one based on BCUT4 only in Figure 5.6 (a) since there is more information from two important variables. In contrast, BCUT10 is irrelevant by construction and performs poorly by itself or in combination with a relevant variable (Figure 5.6 (c) and (d)).

For comparison, I have also plotted the empirical MSE ($\frac{\sum_{i=1}^{n}(y_i-\hat{p}(\mathbf{x_i}))^{\mathbf{2}}}{n}$) versus k for the same subsets of variables in Figure 5.7. Although a minimum (i.e. optimal) MSE can be seen in panels (a) and (b), it is a small dip. The AHR peaks in Figure 5.6 (a-b) are

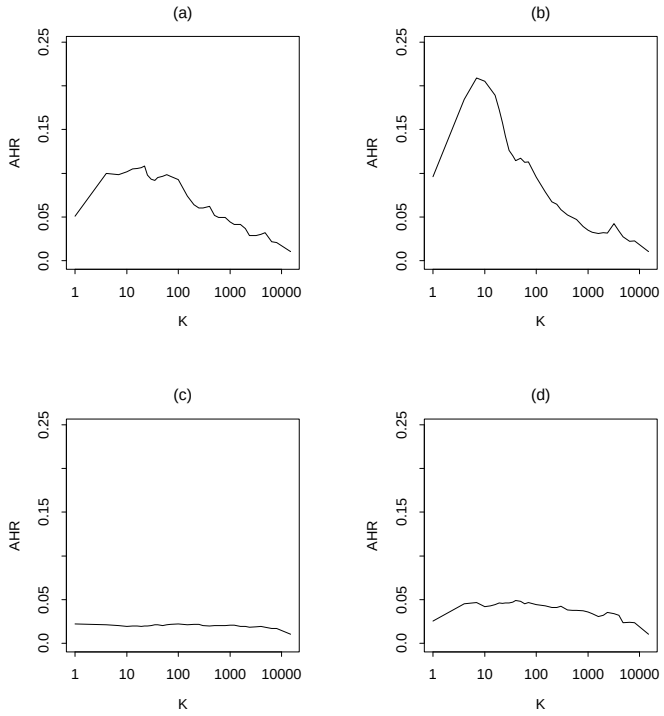

Figure 5.6: AHR versus k for KNN classifiers based on (a) BCUT4, (b) BCUT4 and BCUT6, (c) BCUT10, and (d) BCUT4 and BCUT10. For the augmented NCI data, BCUT10 is an irrelevant variable.

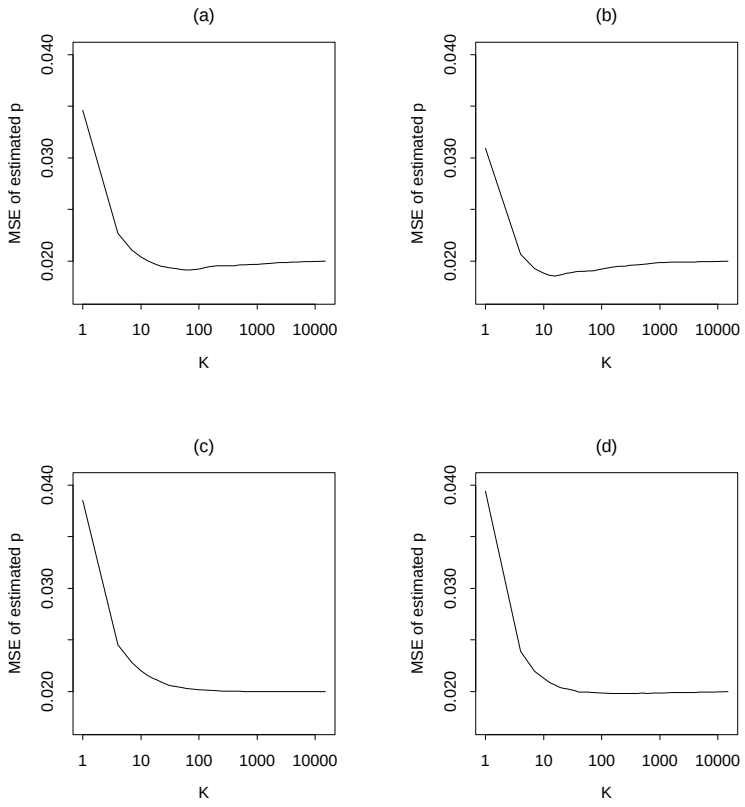

Figure 5.7: Estimated mean square error in estimation of p versus k for KNN classifiers based on (a) BCUT4, (b) BCUT4 and BCUT6, (c) BCUT10, and (d) BCUT4 and BCUT10.

more pronounced. Moreover, the MSE values remain close to the optimum for very large k, while AHR shows significant degradation as k increases. In panels (c-d) of Figure 5.7, the sharp initial decrease in MSE suggests an improvement in performance, even though these variables are irrelevant. Plots for the misclassification rate (not shown) indicate similar results. Therefore, I make use of AHR rather than the conventional criteria such as MSE and misclassification error.

Figure 5.6 clearly shows that BCUT10 is not useful, and I now quantify this argument to screen variables. For a given subset of variables, I compare the classifier's optimal AHR with a null distribution of AHR values that arise under random ranking of the training compounds. The null distribution is obtained by choosing the training compounds in random order, computing the AHR, and repeating many times. To illustrate, I generate 500 random orderings, each with 14906 compounds, of which 304 are active (the same as in the training set of the NCI AIDS data). The AHRs computed from the lists can be viewed as 500 observations from the null distribution of the AHR.

Figure 5.8 displays the density of the null distribution of AHR and the AHR values from KNN for all subsets of one and two variables. We see significant differences from the null distribution of AHR values for certain types of subset. One example would be classifiers using one of the original variables BCUT1, ..., BCUT6 (denoted by open diamonds) all of which have AHR values above the null distribution. Subsets involving two of these variables perform even better. With one or two irrelevant variables from BCUT7, ..., BCUT12, however, the AHR value is typically consistent with the null distribution. Mixing one relevant and one irrelevant variable produces intermediate AHR values, which may or may not be significant (but are clearly inferior to the values from two relevant variables). Here, the AHR values distinguish the relevant and irrelevant variables fairly well.

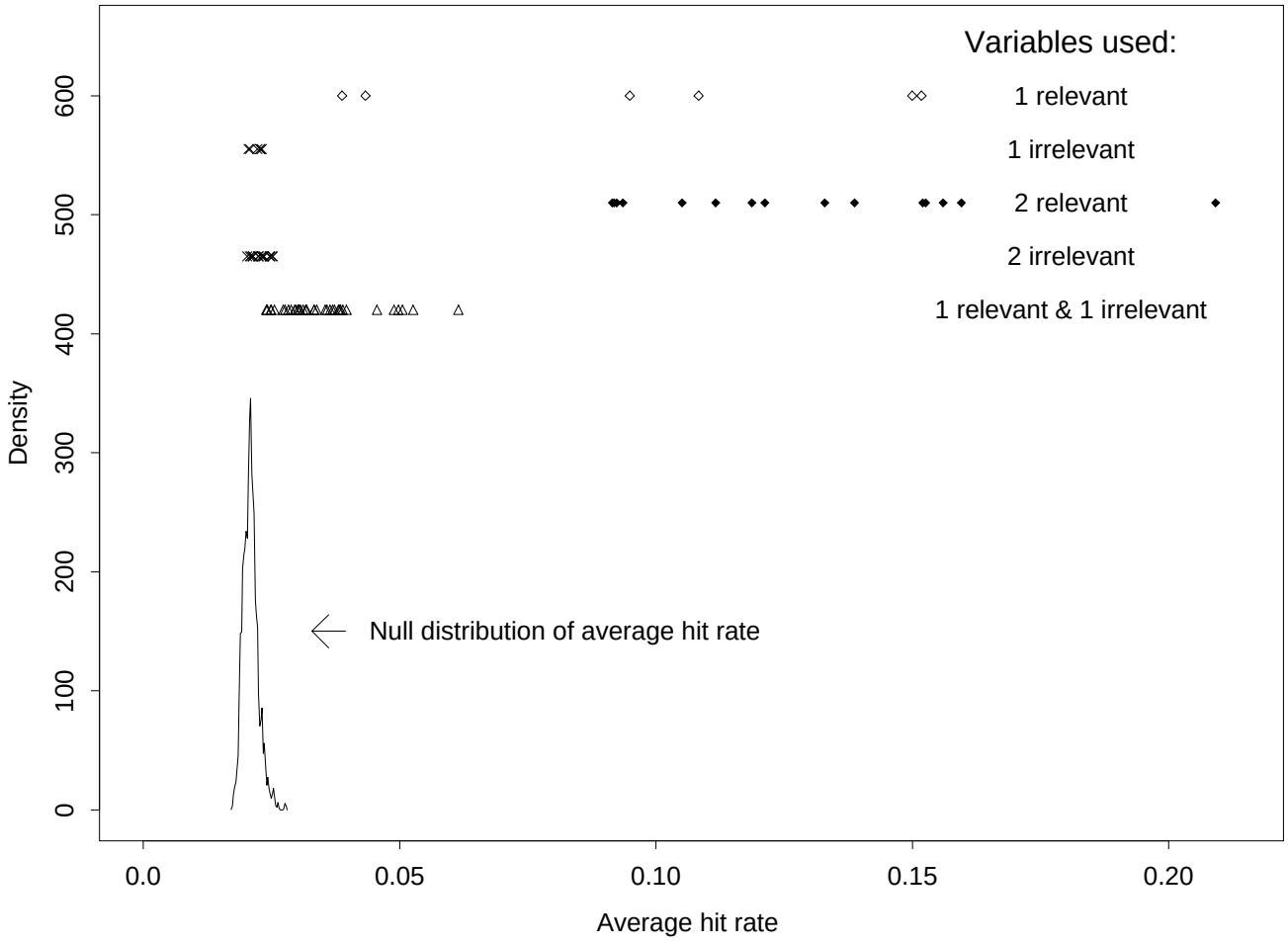

Figure 5.8: AHRs for KNN classifiers based on one or two explanatory variables, compared with the null distribution.

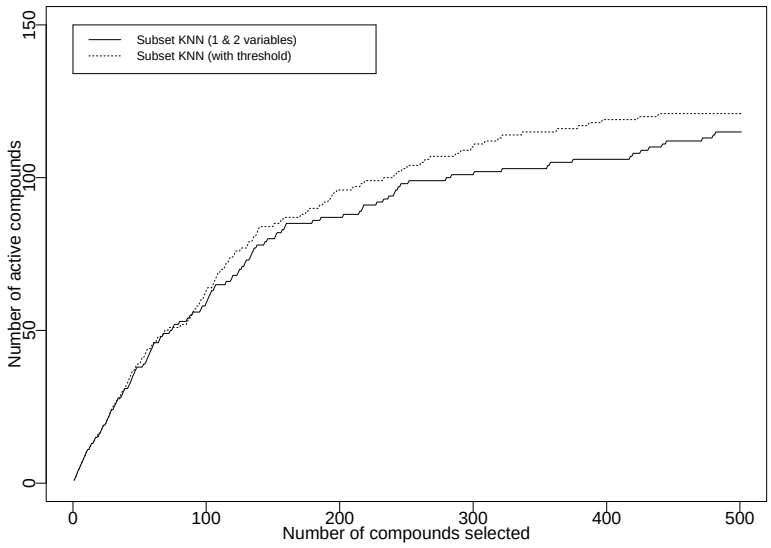

Figure 5.9: Hit curves for KNN classifiers based on 12 variables (augmented NCI data): aggregate of classifiers from all subsets of one or two variables (solid line), and aggregate of classifiers from such subsets with their corresponding AHRs exceeding the 95% quantile of the null distribution (dotted line).

The null distribution may be used to set a threshold. For example, I use only those one- and two-dimensional subsets with an AHR exceeding the 95% quantile of the null distribution. Averaging only the subsets meeting this criterion leads to the test-data hit curve shown as a dotted line in Figure 5.9. It slightly outperforms averaging over all subsets of one and two variables (solid line in Figure 5.9) which is also displayed as a solid line in Figure 5.5. The weak classifiers do not impact much.

5.5 Conclusion

I have empirically demonstrated that aggregating classifiers across subsets of variables can improve the performance of KNN and tree classifiers in ranking objects in terms of their

probability of a desired class. The method is general and could be easily applied to other classifiers. Some theoretical explanation has been also provided to show why strong classifiers from relevant variables tend to dominate this aggregation method. In situations where many variables combine together as complex interactions, higher dimensional subspaces may be necessary. Other applications where ranking is of interest and subset averaging might be useful include identifying rare purchasers in direct mailing, retrieving information in web search studies and so on.

Chapter 6

Method Performance on Other Data Sets

Chapter 4 and Chapter 5 introduce three methods developed in this thesis: the k-varying algorithm, weighted KNN and subset KNN, and apply them to the NCI AIDS data with BCUT descriptors. In order to have a broader evaluation of these methods, this chapter will add three other data sets: NCI AIDS data with Constitutional descriptors, Mutagenicity data with BCUT descriptors and Mutagenicity data with Constitutional descriptors. As a baseline, KNN is also applied to these data sets.

6.1 Mutagenicity data and Constitutional descriptors

The response variable of the Mutagenicity data set is based on the Ames test. Usually four strains of bacteria are tested, with or without metabolic activation. If any of the eight tests is positive, the compound is considered positive. This data set comes from various public

sources like EPA, NIH, etc. It was compiled by GlaxoSmithKline chemists. It includes 1866 compounds. Of them, 898 are mutagenic ($y = 1$) and 968 are non-mutagenic ($y = 0$).

The 47 Constitutional descriptors include information such as molecular weight, atomic weight, atomic counts, etc. The basic information of descriptors is listed in Appendix A. Descriptors in this class are not determined by the connectivity or conformation of the molecule. In comparison, BCUT descriptors are determined by the connectivity, i.e., topological relations between different atoms within the molecule. The response of the NCI data sets and the BCUT descriptors have been described earlier in Section 2.3. For the mutagenicity data, the same six BCUT variables are used as were used in NCI AIDS data. All descriptors in the four data sets were computed by GlaxoSmithKline chemists, although similar descriptors could be computed with the Dragon software (Mannhold 2000). Note that all the BCUT and the Constitutional variables were scaled to have mean 0 and variance 1 before any analysis.

6.2 Performance of the methods

Each method is applied across all the data sets. Since for the mutagenicity data set the number of mutagenic compounds is almost the same as the number of non-mutagenic compounds, misclassification rate is reported while AHR is reported for the NCI AIDS data sets.

Table 6.1 lists the AHRs measured on the test sets for four random splits of the two NCI AIDS data sets (two descriptor sets). Due to computational reasons (the Constitutional descriptors have 47 variables and there are about 30,000 observations), subset KNN with the Constitutional descriptors only considers low dimensional subsets. Only two dimensional subsets are actually used because the one dimensional subsets produce many ties for KNN.

Table 6.1: AHRs measured on the test sets for four random splits of NCI AIDS data on two descriptor sets respectively.

NCI AIDS Data								
method	BCUTs				Constitutional			
	split #				split #			
	1	2	3	4	1	2	3	4
KNN	0.2279	0.2121	0.2313	0.2377	0.1905	0.2027	0.1643	0.1625
k-varying algorithm	0.2490	0.2545	0.2465	0.2530	0.2643	0.2485	0.2054	0.2149
Weighted KNN	0.2447	0.2466	0.2442	0.2704	0.1704	0.1943	0.1426	0.1752
subset KNN	0.3107	0.3200	0.3003	0.3067	0.1372	0.1444	0.1131	0.1286

Also due to high computational cost, no cross-validation is applied for KNN, k-varying algorithm and subset KNN on the Constitutional descriptors. k is fixed to 10 for both KNN and subset KNN; t is fixed to be 30 in the k-varying algorithm. Weighted KNN does use cross-validation as the algorithm written in C is efficient.

The pairwise hypothesis tests (paired t-tests) comparing the four classifiers for the NCI AIDS data with BCUT descriptors are given in Table 6.2. Section 3.4 contains details of similar tests. The alternative hypothesis is whether the AHR for "Classifier 1" is larger than that for "Classifier 2" on average.

Based on AHR, there is a statistically significant positive difference when subset KNN is compared with weighted KNN, with k-varying algorithm, or with KNN. There is no strong evidence that weighted KNN outperforms k-varying algorithm. k-varying algorithm does

Table 6.2: Pairwise significance tests among the selected methods for the NCI AIDS data with BCUT descriptors: subset KNN, weighted KNN, k-varying algorithm and KNN.

Classifier 1	Classifier 2	$\overline{d}$	s_d	t statistics	p value
k-varying algorithm	KNN	0.0235	0.0129	3.644	0.018
weighted KNN	k-varying algorithm	0.000725	0.0114	0.128	0.45
Subset KNN	weighted KNN	0.0580	0.0161	7.21	0.0028

Table 6.3: Pairwise significance tests among the selected methods for the NCI AIDS data with Constitutional descriptors: subset KNN, weighted KNN, k-varying algorithm and KNN.

Classifier 1	Classifier 2	$\overline{d}$	s_d	t statistics	p value
weighted KNN	subset KNN	0.0398	0.00996	7.988	0.0020
KNN	weighted KNN	0.009375	0.01587	1.182	0.16
k-varying algorithm	KNN	0.05328	0.01445	7.375	0.0026

appear to be superior to KNN.

Similarly, Table 6.3 displays the pairwise hypothesis tests (paired t-tests) comparing the four classifiers for the NCI AIDS data with Constitutional descriptors. Based on AHR, there is a statistically significant positive difference when k-varying algorithm is compared with KNN, with weighted KNN, or with subset KNN. There is no strong evidence that KNN outperforms weighted KNN. Weighted KNN does appear to be superior to subset KNN.

In general, subset KNN is best based on the BCUTs while the k-varying algorithm is best based on Constitutional descriptor set. Since the NCI data with the BCUT descriptor

Table 6.4: Misclassification rates measured on the test sets for four random splits of mutagenicity data on two descriptor sets respectively.

Mutagenicity Data								
method	BCUTs				Constitutional			
	split #				split #			
	1	2	3	4	1	2	3	4
KNN	0.2650	0.2468	0.3090	0.2918	0.2607	0.2296	0.2468	0.2146
k-varying algorithm	0.2735	0.2639	0.3111	0.2939	0.2393	0.2275	0.2521	0.2244
Weighted KNN	0.2564	0.2639	0.2832	0.2618	0.2329	0.2167	0.2532	0.2167
Subset KNN	0.2564	0.2158	0.2542	0.2371	0.2948	0.3333	0.3226	0.2863

set and with the Constitutional descriptor set are from different sources, they are slightly different. The latter is a subset of the former and has 29374 compounds compared to 29812 compounds in the BCUT data set. As its active ratio is lower in the data set with Constitutional descriptors (1.7% active), AHR values are not directly comparable across the two descriptor sets.

Table 6.4 lists the misclassification rates measured on the test sets for four random splits of the mutagenicity data set on two descriptor sets respectively. The smaller the misclassification, the better the performance. Here, cross-validation is applied to each method and the misclassification rate is used to select tuning parameters.

Note that k-varying algorithm is excluded from further analysis (paired t-test) since it is designed for unbalanced data (i.e., for ranking with rare targets).

Table 6.5: Pairwise significance tests among the selected methods for the Mutagenicity data with BCUT descriptors: subset KNN, weighted KNN, KNN.

Classifier 1	Classifier 2	$\overline{d}$	s_d	t statistics	p value
weighted KNN	KNN	0.01183	0.02139	1.106	0.18
subset KNN	weighted KNN	0.02545	0.01978	2.573	0.041

Table 6.6: Pairwise significance tests among the selected methods for the Mutagenicity data with Constitutional descriptors: subset KNN, weighted KNN, KNN.

Classifier 1	Classifier 2	$\overline{d}$	s_d	t statistics	p value
KNN	subset KNN	0.0713	0.0286	4.988	0.0077
weighted KNN	KNN	0.00805	0.0156	1.035	0.19

The pairwise hypothesis tests (paired t-tests) comparing the three classifiers for the Mutagenicity data with BCUT descriptors are given in Table 6.5. The alternative hypothesis is whether the misclassification rate for "Classifier 1" is lower than that for "Classifier 2" on average.

Based on misclassification rate, there is a statistically significant lower misclassification rate when subset KNN is compared with weighted KNN, or KNN. There is no strong evidence that weighted KNN outperforms KNN.

Similarly, Table 6.6 displays the pairwise hypothesis tests (paired t-tests) for the Mutagenicity data with Constitutional descriptors to compare the three classifiers. Based on misclassification rate, there is no strong evidence that weighted KNN outperforms KNN. There is a statistically significant lower misclassification rate when weighted KNN and KNN

are compared with subset KNN.

In general, subset KNN is best based on BCUTs while it is worst based on the Constitutional descriptor set in term of misclassification rate. For Constitutional descriptor set, KNN and weighed KNN are comparable to each other and better than subset KNN.

6.3 Conclusion

Subset KNN is not powerful using the Constitutional descriptors on both the NCI AIDS data set and the Mutagenicity data set possibly because low dimensional (one or two) subsets do not include enough information about the compounds compared to the total 47 Constitutional descriptors. However, as I mentioned in Section 2.3.1 that Pearlman and Smith (1999) showed that it may be possible to find fairly low-dimensional (2-D or 3-D) subsets of BCUT variables such that active compounds are clustered in the relevant subspace.Therefore it is probably not surprising that subset KNN performs well on BCUT descriptors.

The k-varying algorithm does well on the NCI AIDS data. As is often the case in data mining, no single method dominates in all problems. For example, there is no method which is best at both descriptor sets. However, the newly developed methods in this thesis, namely, the k-varying algorithm, weighted KNN and subset KNN appear quite promising. Different methods are best for the different data sets, but some of the proposed methods are always amongst the best.

Chapter 7

Estimating Activity and Error Rates for Assessing Model Performance

The previous chapters try to build classifiers, modify them, and propose novel adaptations in the pursuit of a higher hit curve (or AHR) to reach the goal of drug discovery. A perfect hit curve is the objective. However, there are a large number of systematic and random errors during the assay process. Assay results are not infallible: the assayed active compounds might not be truly active, and the assayed inactive compounds might be truly active. The y-axis of the hit curve is indeed the number of compounds assayed active instead of the number of truly active compounds. Therefore, even a perfect classifier (which divides the truly active and inactive compounds perfectly) cannot have an ideal hit curve.

This chapter introduces a way to estimate the ranges of the error rates and hence illustrates quantitatively how much more improvement is possible in a model's performance. Section 7.1 briefly introduces the likelihood function for the data from the assay process. Based on the likelihood function for the NCI AIDS data, activity and error rates are es-

Table 7.1: False positive and false negative probability matrix for the assay process.

Probability matrix		**Assay**	
		inactive	active
True status	inactive	$1-\theta_+$	θ_+
	active	θ_-	$1-\theta_-$

timated in Section 7.2. Section 7.3 visualizes possible hit curves subject to the different estimations and argues that the performance of models is greatly affected by the activity and error rates. Section 7.4 applies similar approaches to a data set with more repeat measurements and shows that the parameter estimation is more reliable. A short summary is given in Section 7.5.

7.1 Likelihood function

In this section, I develop a probability model for the measurement process. The parameters of the model are estimated from the likelihood function and the estimation provides insight into the accuracy of the assay procedure and possible performance bounds for classifiers.

To estimate the errors produced by the assay, I assume the following probability setting (Table 7.1):

The probability of inactive status in an assay given an active compound is θ_-, i.e.,

$$\text{P(inactive status in assay} \mid \text{real active)} = \theta_-$$

Similarly,

$$\text{P(active status in assay} \mid \text{real active)} = 1 - \theta_-$$
$$\text{P(active status in assay} \mid \text{real inactive)} = \theta_+$$
$$\text{P(inactive status in assay} \mid \text{real inactive)} = 1 - \theta_+$$

θ_+ and θ_- are commonly called the *false positive rate* and *false negative rate*, respectively. If a compound randomly selected from a population of compounds is truly active with probability π and truly inactive with probability $1 - \pi$, the probability distribution of the observed activity, Y, of the compound can be easily calculated:

$$P(Y = 0) = \theta_- \pi + (1 - \theta_+)(1 - \pi)$$

$$P(Y = 1) = (1 - \theta_-)\pi + \theta_+(1 - \pi).$$

As before $Y = 0, 1$ indicates that the compound is observed to be inactive and active, respectively.

Given the data, n_0 and n_1 are the observed number of compounds whose measured activities are 0 and 1, respectively. Assuming those compounds are randomly selected from a population of compounds with the above probability settings, the observed log-likelihood is a function of n_0 and n_1:

$$l(n_0, n_1) = n_0 \log P(Y = 0) + n_1 \log P(Y = 1) + \text{constant} \tag{7.1}$$

In (7.1), each compound is assayed once. However, drug discovery data may have some duplicated compounds. To estimate the assay error, those repeated measurements provide much more valuable information than do the unique ones. This argument is advanced in Section 7.2. Below, I derive the correct likelihood to include such auxiliary information.

Suppose a compound has r replicate assays. Let $A_r \in \{0, 1, \ldots, r\}$ denote denote the number of active assays observed for this compound. For instance, there are three possible values of A_r for $r = 2$ replicates: $A_2 = 0$, $A_2 = 1$, $A_2 = 2$. For example, $A_2 = 1$ indicates that two identical compounds are tested: one is assayed active, and the other is assayed inactive. Assuming statistical independence of the r observed assay results, it is easy to find the probability of each case:

$$\begin{aligned} P(A_2 = 0) &= P(A_2 = 0 \mid actually \quad active)\pi + \\ & P(A_2 = 0 \mid actually \quad inactive)(1 - \pi) \\ &= \theta_-^2 \pi + (1 - \theta_+)^2 (1 - \pi). \end{aligned}$$

Similarly,

$$P(A_2 = 1) = 2[(1 - \theta_-)\theta_- \pi + (1 - \theta_+)\theta_+ (1 - \pi)] \tag{7.2}$$

$$P(A_2 = 2) = (1 - \theta_-)^2 \pi + \theta_+^2 (1 - \pi). \tag{7.3}$$

In general, the probability of observing $A_r = a$ can be written as

$$P(A_r = a) = \binom{r}{a} [(1 - \theta_-)^a \theta_-^{r-a} \pi + (1 - \theta_+)^{r-a} \theta_+^a (1 - \pi)].$$

Let n_{ra} be the number of compounds in the data with $A_r = a$. The log-likelihood function can be written as

$$l(\pi, \theta_+, \theta_-) = \sum_{r,a} n_{ra} \log P(A_r = a). \tag{7.4}$$

Since the binomial coefficients $\binom{r}{a}$ are only additive constants in the log-likelihood, I exclude them when computing the log-likelihood values in later sections.

Table 7.2: Observed n_{ra} counts for the NCI AIDS data.

r	a							
	0	1	2	3	4	5	6	7
1	28167	573						
2	463	11	8					
3	23	0	0	2				
4	4	0	0	0	0			
5	1	0	1	0	0	0		
6	0	0	0	0	0	0	0	
7	1	0	0	0	0	0	0	0

7.2 Parameter estimation for the NCI AIDS data

Table 7.2 lists the observed counts in the NCI AIDS data. For instance, when $r = 2$, there are 482(=463+11+8) compounds tested twice. Of them, 463 compounds are assayed to be inactive for both tests; 11 are inactive for one test and active for the other; and 8 are active for both tests. Putting the frequencies in Table 7.2 into (7.4) gives the log likelihood for the NCI AIDS data.

The log likelihood is a function of three parameters. It can be examined visually one or two parameters at a time via the profile likelihood (e.g. Kalbfleisch 1985, Section 10.3). For example, the profile log-likelihood function of (7.4) on π can be written as

$$l_p(\pi) = \sup_{\theta_+,\theta_-} l(\pi, \theta_+, \theta_-). \tag{7.5}$$

According to the likelihood ratio test, under some general conditions, the distribution of

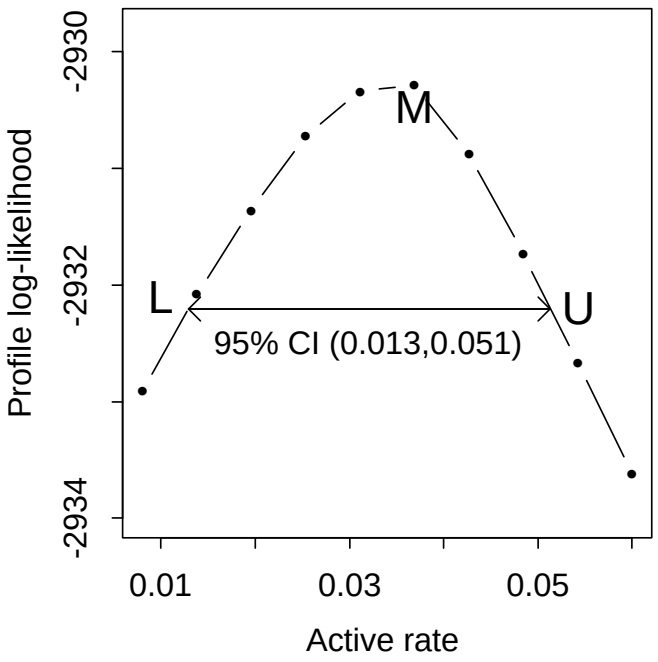

Figure 7.1: The 95% profile likelihood confidence interval for π: lower bound $\pi = 0.013$ (indicated by "L"), upper bound $\pi = 0.051$ (indicated by "U"), and the maximum likelihood estimate $\pi = 0.037$ (indicated by "M").

$-2(l_p(\pi) - l(\hat{\pi}, \hat{\theta}_+, \hat{\theta}_-))$ is approximately χ^2 with 1 degree of freedom, where $\hat{\pi}$, $\hat{\theta}_+$, and $\hat{\theta}_-$ are the maximum likelihood estimates.

Figure 7.1 displays the profile log-likelihood, $l_p(\pi)$, as a function of the active rate, π. The segment from letter "L" to "U" illustrates the 95% confidence interval of active rate π based on the profile likelihood. The confidence interval is from 0.013 to 0.051, and the maximum likelihood estimate indicated by "M" is approximately 0.037. Although the observed proportion of actives (around 0.020) is inside the confidence interval of π, it differs slightly

from the maximum likelihood estimate "M". This is related to the fact illustrated later in this section that the maximum likelihood estimate of the false negative rate (θ_-) is rather high.

Some interesting patterns emerge when the likelihood function is split into two parts: one from unrepeated measurements and the other from repeated measurements. Let $l_u(\pi, \theta_+, \theta_-)$ denote the log-likelihood from the un-replicated assays ($r = 1$) and $l_r(\pi, \theta_+, \theta_-)$ from the repeated assays ($r > 1$):

$$l(\pi, \theta_+, \theta_-) = l_u(\pi, \theta_+, \theta_-) + l_r(\pi, \theta_+, \theta_-), \tag{7.6}$$

where

$$l_u(\pi, \theta_+, \theta_-) = \sum_a n_{1a} \log P(A_1 = a) \tag{7.7}$$

$$l_r(\pi, \theta_+, \theta_-) = \sum_{r>1;a} n_{ra} \log P(A_r = a). \tag{7.8}$$

Let $\widetilde{\theta}_+(\pi)$ and $\widetilde{\theta}_-(\pi)$ be false positive and false negative rates which maximize the log-likelihood at particular values of π,

$$(\widetilde{\theta}_+(\pi), \widetilde{\theta}_-(\pi)) = \text{argsup}_{\theta_+,\theta_-} l(\pi, \theta_+, \theta_-). \tag{7.9}$$

Then the profile log-likelihood $\sup_{\theta_+,\theta_-} l(\pi, \theta_+, \theta_-)$ for π is composed of two parts, the log-likelihood from unique observations and from repeated observations:

$$\begin{aligned} l_p(\pi) &= \sup_{\theta_+,\theta_-} l(\pi, \theta_+, \theta_-) = l(\pi, \widetilde{\theta}_+(\pi), \widetilde{\theta}_-(\pi)) \\ &= l_u(\pi, \widetilde{\theta}_+(\pi), \widetilde{\theta}_-(\pi)) + l_r(\pi, \widetilde{\theta}_+(\pi), \widetilde{\theta}_-(\pi)). \end{aligned} \tag{7.10}$$

Similar to Figure 7.1, Figure 7.2 draws each likelihood contribution $l_u(\pi, \widetilde{\theta_+}(\pi), \widetilde{\theta_-}(\pi))$ or $l_r(\pi, \widetilde{\theta_+}(\pi), \widetilde{\theta_-}(\pi))$ versus π. The left panel is for the unique observations and the right panel

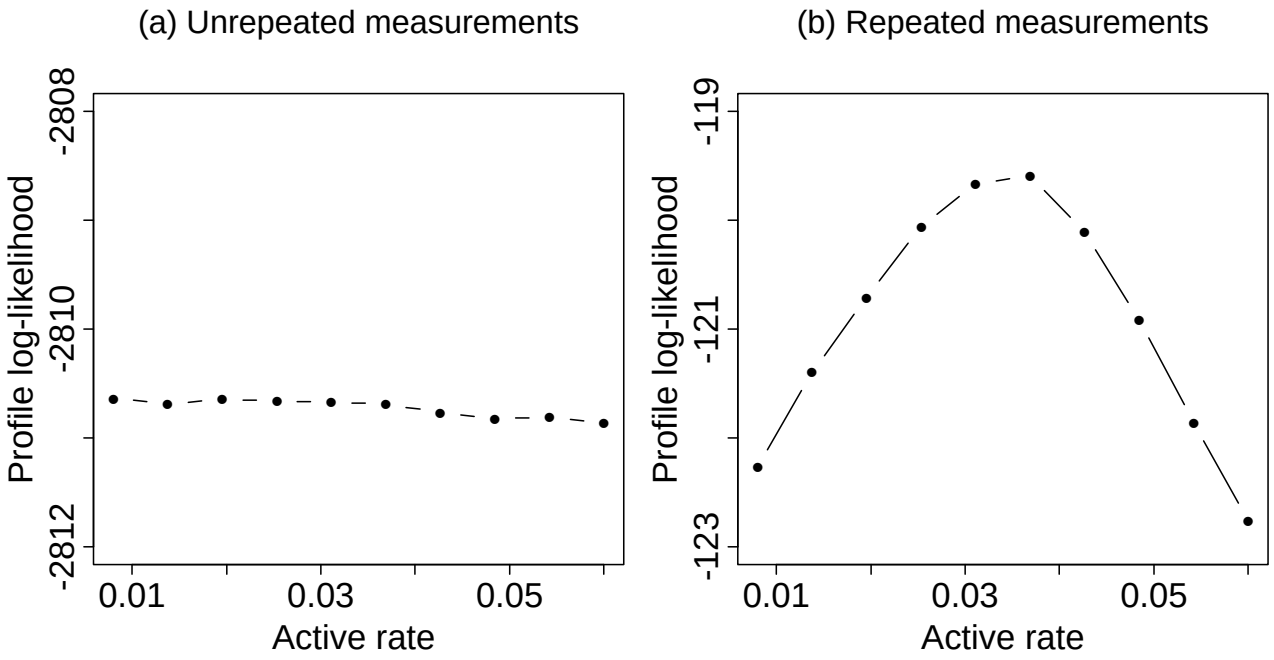

Figure 7.2: Log-likelihood contribution from (a) unique observations and (b) repeated observations.

is for the duplicated observations. To facilitate comparison, the y-axis units in Figure 7.1 and 7.2 are the same.

The large negative values in the first part of the profile log-likelihood (Figure 7.2(a)) reflect the large number of unrepeated measurements. The likelihood contribution from these unrepeated measurements hardly varies with π, indicating that these observations are uninformative with respect to π. The peak in Figure 7.2(b) is almost the same as that in Figure 7.1 but with a big shift along the y-axis. Thus, the intuition that repeated measurements provide more information on parameters is confirmed. Note that the increased proportion of active compounds in the repeated-measurement group might also provide a big contribution to the change of the likelihood.

In the NCI AIDS data, very few compounds are measured more than two times. Therefore, I do not split the likelihood further.

I now visualize the behavior of the log likelihood with respect to θ_+ and θ_- by fixing π at several values. Ideally, the selected π values represent a good coverage of the confidence interval for π in Figure 7.1. Then, I examine the confidence region of θ_+ and θ_- for each chosen value of π. Three selected points are displayed in Figure 7.1: the lower bound ($\pi = 0.013$, indicated by "L"), the upper bound ($\pi = 0.051$, indicated by "U"), and the maximum likelihood estimate ($\pi = 0.037$, indicated by "M").

The contour plots of $l(\pi, \theta_+, \theta_-)$ for θ_+ and θ_- under the three chosen values of π are displayed in Figure 7.3 with all the figures on the same scale. Some values of (θ_+, θ_-) in Figure 7.3 are particularly meaningful. (0,0) located on the left bottom, indicates there are no errors at all. The contours shown include fairly large values of θ_-, so the θ_- range goes up to 1. The corner (0,1) corresponds to observing only inactive compounds. In contrast, θ_+ is very small for all contours shown, and its range is truncated at about 0.016.

In Figure 7.3, the local maximum log-likelihood estimate is indicated by the number "1". The legends number four contour lines from "2" (the contour closest to the center) to "5" (the farthest one) and give the corresponding log-likelihood values which differ from the maximum log-likelihood by $2, 5, 10$, and 15, respectively.

Given $\pi = c$, according to the likelihood ratio test, the distribution of $-2(l(c, \theta_+, \theta_-) - l(c, \hat{\theta}_+, \hat{\theta}_-))$ is approximately χ^2 with 2 degrees of freedom. Figure 7.4 displays the simultaneous 95% confidence region for θ_+ and θ_- under the same chosen values of π (Note the scales now differ.).

The contour plots for θ_+ and θ_- change dramatically according to the different values of π. Specifically, the 95% confidence intervals shown in Figure 7.4 contain fairly small values

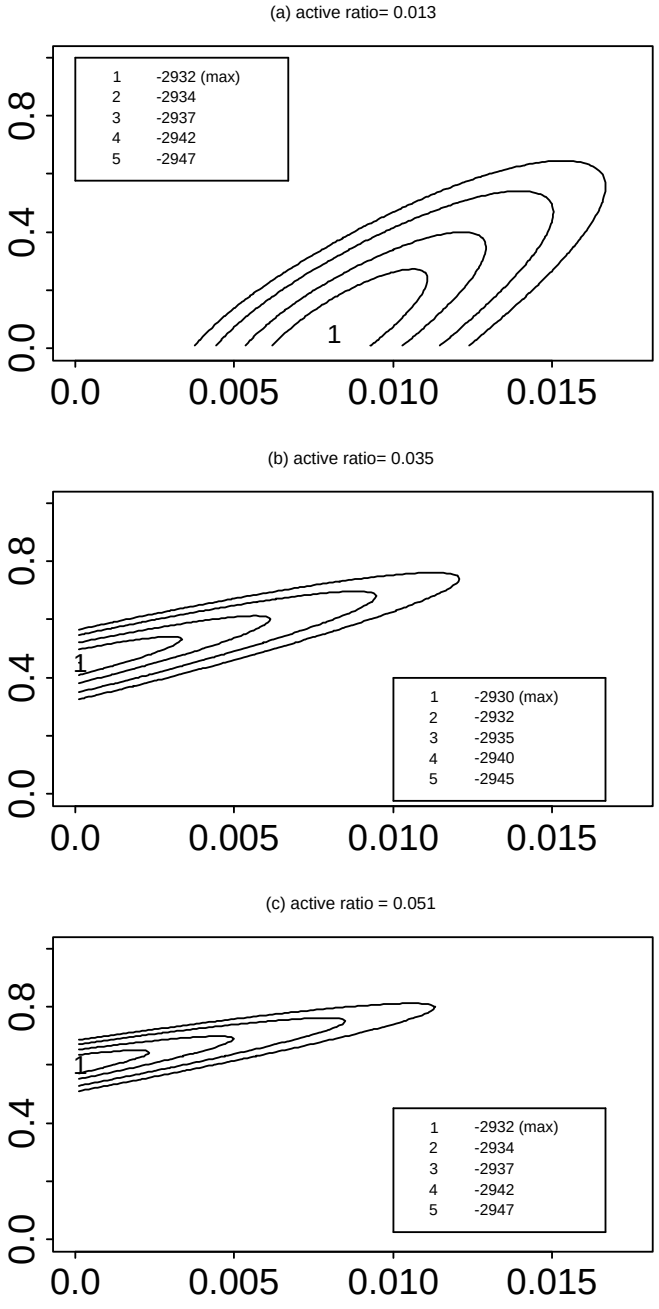

Figure 7.3: Contours for $L(\pi, \theta_+, \theta_-)$ as a function of (θ_+, θ_-) under three values of π: (a) $\pi = 0.013$, (b) $\pi = 0.037$, and (c) $\pi = 0.051$. 1: the maximum log-likelihood. The horizontal axis corresponds to θ_+ and the vertical to θ_-. Log-likelihood values defining the contours are listed in the box for each figure. The log-likelihood decreases as points move away from the mode.

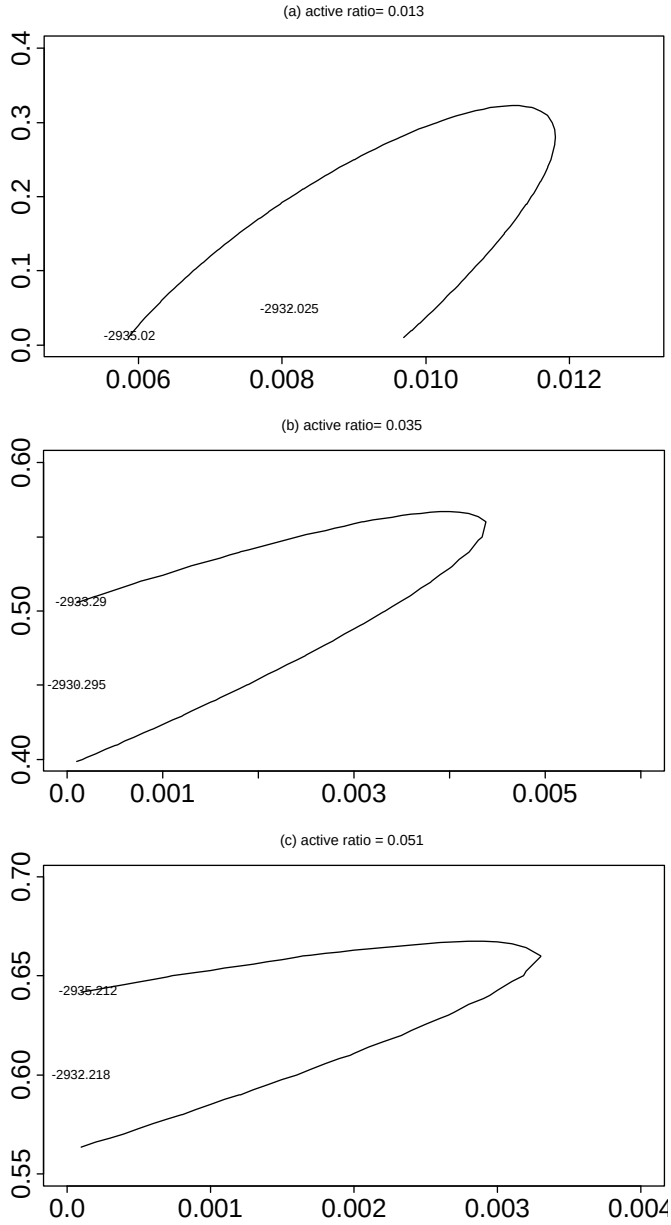

Figure 7.4: The 95% confidence region for θ_+ and θ_- under three values of π: (a) $\pi = 0.013$, (b) $\pi = 0.037$, and (c) $\pi = 0.051$. The horizontal axis corresponds to θ_+ and the vertical to θ_-.

of the false positive rate (θ_+), from 0 to about 0.016. However, the false negative rate (θ_-) can be quite large. For example, θ_- in the left bottom panel takes values between 0.56 and 0.67. Even for the maximum likelihood estimate of π, the values of θ_- are high, and the center is around 0.45. Some small values are taken only when $\pi = 0.013$. However, the confidence interval still goes up to 0.3. As the value of π increases, the value of θ_- tends to increase quickly; θ_+ becomes smaller but is less sensitive. On the other hand, a small change in the false positive rate (θ_+) leads to a big change in the false negative rate (θ_-).

Clearly, the false negative rate could be quite large. Such large error rates deteriorate the performance of a classifier. The relationship between the estimated error rates and the hit curve is investigated in Section 7.3.

7.3 Relationship between error rates and hit curve

As mentioned at the beginning of the chapter, the perfect classifier which classifies the active and inactive compounds perfectly cannot have a perfect hit curve because of the presence of assay errors. This section investigates the hit curve of the perfect classifier and compares it to the hit curve produced by KNN, which gives us some idea of how much the model can be improved.

Given the values of π, θ_+ ,and θ_-, it is very easy to compute an expected hit curve for the perfect classifier. The y label of the hit curve is the expected number of observed hits instead of the unknown number of true active compounds. For example, in the NCI AIDS data, when $\pi = 0.037$, $\theta_+ = .0001$, and $\theta_- = 0.45$ (maximum likelihood estimates) with 14906 test compounds, there are 552 $(= 14906 \times 0.037)$ true actives. Of them, $552(1 - 0.45) = 303$ compounds are expected to be assayed active. On the other hand, there are 14354 (=

$14906(1-0.037))$ true inactives. Of them, $1 \quad (= 14384 \times 1e-04)$ is expected to be assayed active. In total, I expect $303+1=304$ active assays out of 14906 test compounds. For the perfect classifier, the true active compounds are always selected ahead of the inactive ones. Therefore, its hit curve must pass three points:

$$(0\ ,\ 0),\ (552\ ,\ 303),\ \text{and}\ (14906\ ,\ 304).$$

A line connecting these three points will be called the "expected hit curve". Figure 7.5 displays the first part of this hit curve (up to 1000 highest-ranked compounds).

I now show how spread out the possible hit curves could be subject to different but plausible values of π, θ_+, and θ_-. First, I intentionally choose 30 combinations of (π, θ_+, θ_-) which cover their simultaneous confidence region very well. Basically, in each of the three confidence regions of θ_+ and θ_- in Figure 7.4 when π equals $0.013, 0.037$, and 0.051, respectively, 10 points are picked to provide a good coverage of the boundary of the 95% confidence region. The selected points labeled "0" to "9" are displayed in (a), (c), and (e) of Figure 7.6. Each of the 30 points corresponds to a particular (π, θ_+, θ_-) value, and the corresponding expected hit curve can be easily constructed and drawn in the right panels of Figure 7.6. The corresponding expected hit curves are also labeled from "0" to "9". These 30 lines provide a range of possible expected hit curves for a perfect classifier.

For comparison, I also draw the hit curve for subset KNN (solid line) based on the first split of data. Subset KNN has the best hit curve among the methods explored in this thesis (Section 5.2).

When $\pi = 0.013$ in the top panel of Figure 7.6, all 10 expected hit curves are just on top of the hit curve of subset KNN: there is not much room for improving the method. In the other two pictures, when $\pi = 0.037$ and 0.051, all the expected hit curves are dominated

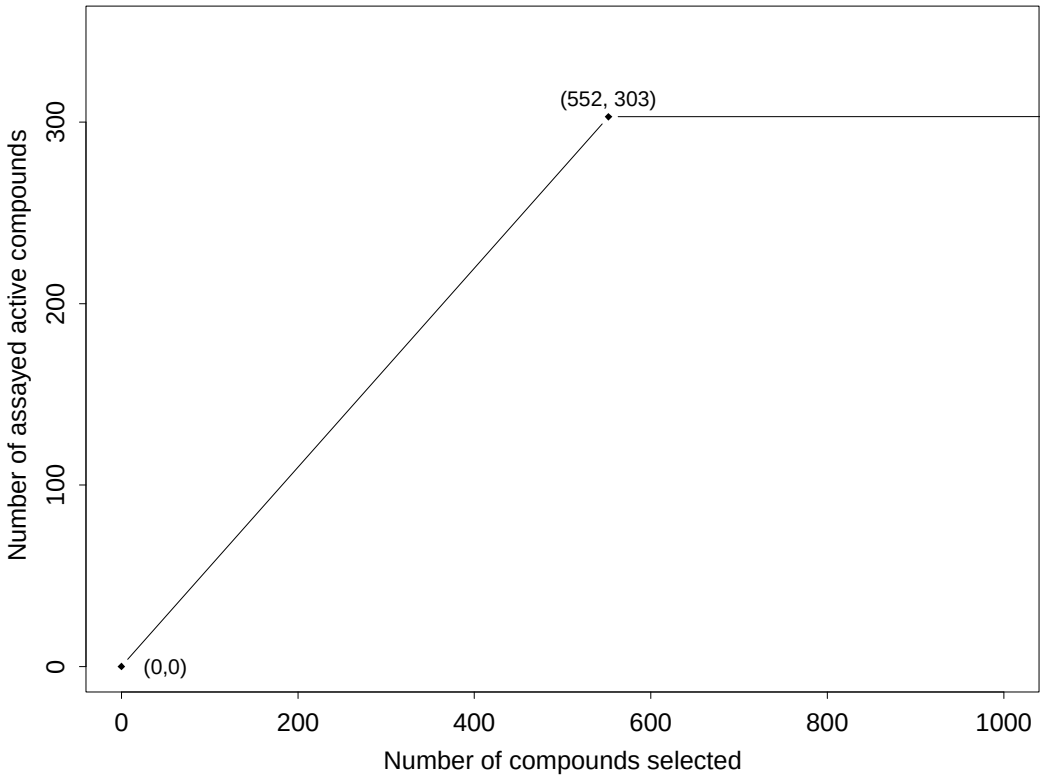

Figure 7.5: Expected hit curve for a perfect classifier when $\pi = 0.037$, $\theta_+ = .0001$, and $\theta_- = 0.45$ (maximum likelihood estimates).

by the curve from subset KNN up to about 100–200 highest-ranked compounds. This early domination suggests possible violations of the model for multiple assays developed in Section 7.1. However, in both plots, as the expected number of observed active compounds is larger than 304 active compounds in the test sets of the NCI AIDs data even with a high false negative rate, subset KNN based on the NCI AIDs data is not able to have a hit curve as high as expected hit curves in the end.

One reason for the large variation in the expected hit curves is the high level of parameter

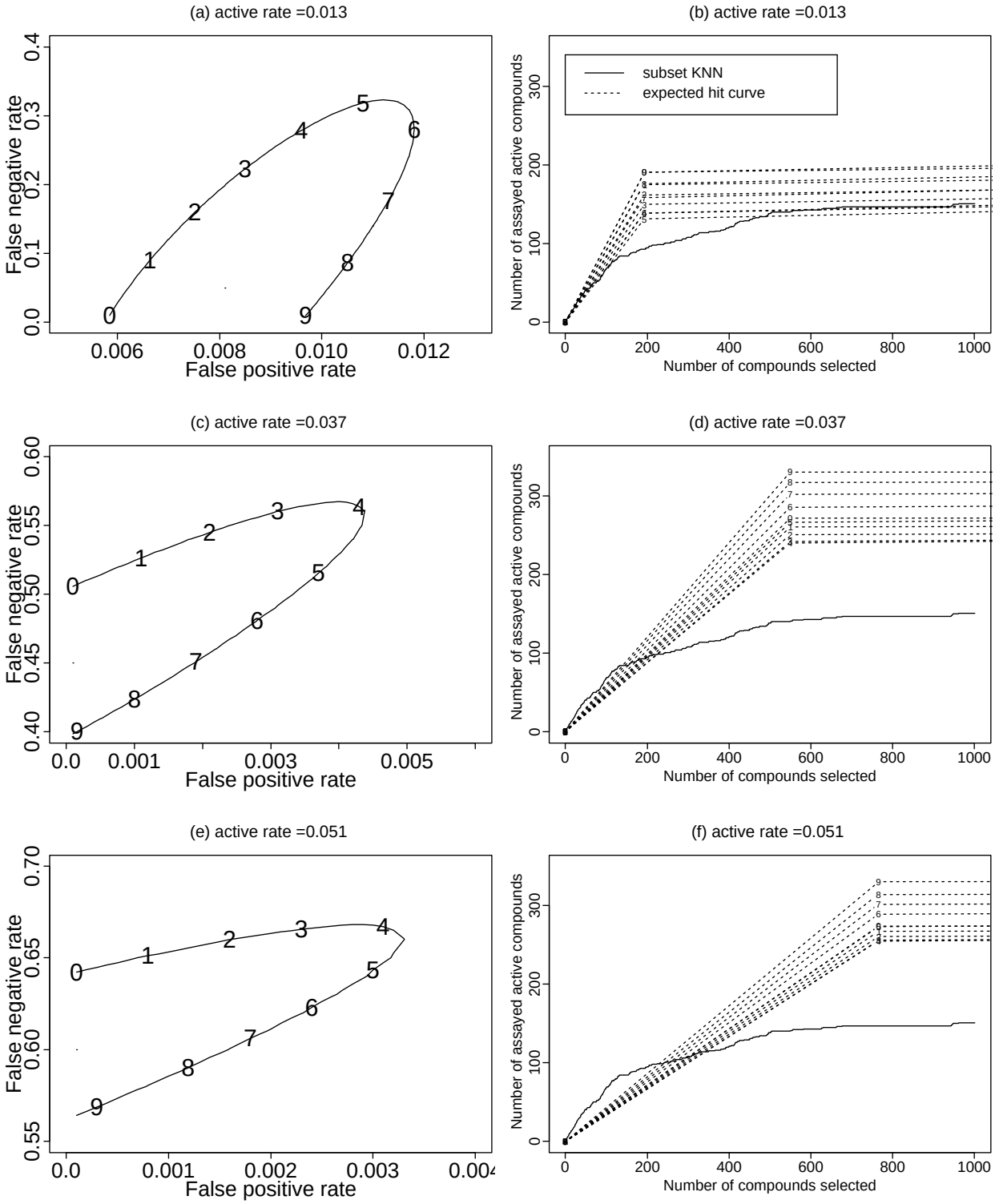

Figure 7.6: Expected hit curves of 30 combinations of (π, θ_+, and θ_-).

uncertainty. This may stem from the limited number of replicate observations. Therefore, in Section 7.4, a similar approach is applied to a data set with more repeated measurements.

7.4 Estimating activity and error rates on a data set with 6 repeated measurements

The data set presented in this section was originally provided by Brent Stockwell, a Fellow of the Whitehead Institute at MIT. In his study, 1040 compounds were each assayed six times. The first three replicates were generated as one group and the last three replicates as the other group. All the measurements are continuous, and the locations of the compounds on assay plates are recorded in the data. There are 13 plates: each plate has 8 rows and 10 columns (i.e. 80 cells per plate). The data set is referred to as the "REPS data". Because it has many repeated measurements, it provides a good comparison with the NCI AIDS data investigated in Section 7.2.

Figure 7.7 gives boxplots of the assay values for the 1040 compounds, by replicate number (R1, ..., R6). The first three measurements (R1, R2, and R3) are in one group, and the next three (R4, R5, and R6) are in the other group. Some differences in the distributions are apparent. Although the interquartile range of the measurements in R4-R6 is narrower than in R1-R3, R4-R6 seem to have more large values (outliers). Because of the measurement inconsistency between the groups, I treat R1-R3 and R4-R6 as different experiments. Therefore, the parameter estimation is performed separately for R1-R3 and R4-R6.

The compounds were not randomly assigned to wells on the plates. In fact, a given compound always appears in the same row and column of the plate. While statisticians worry about such designs, they are often run in practice due to technical difficulties, operational

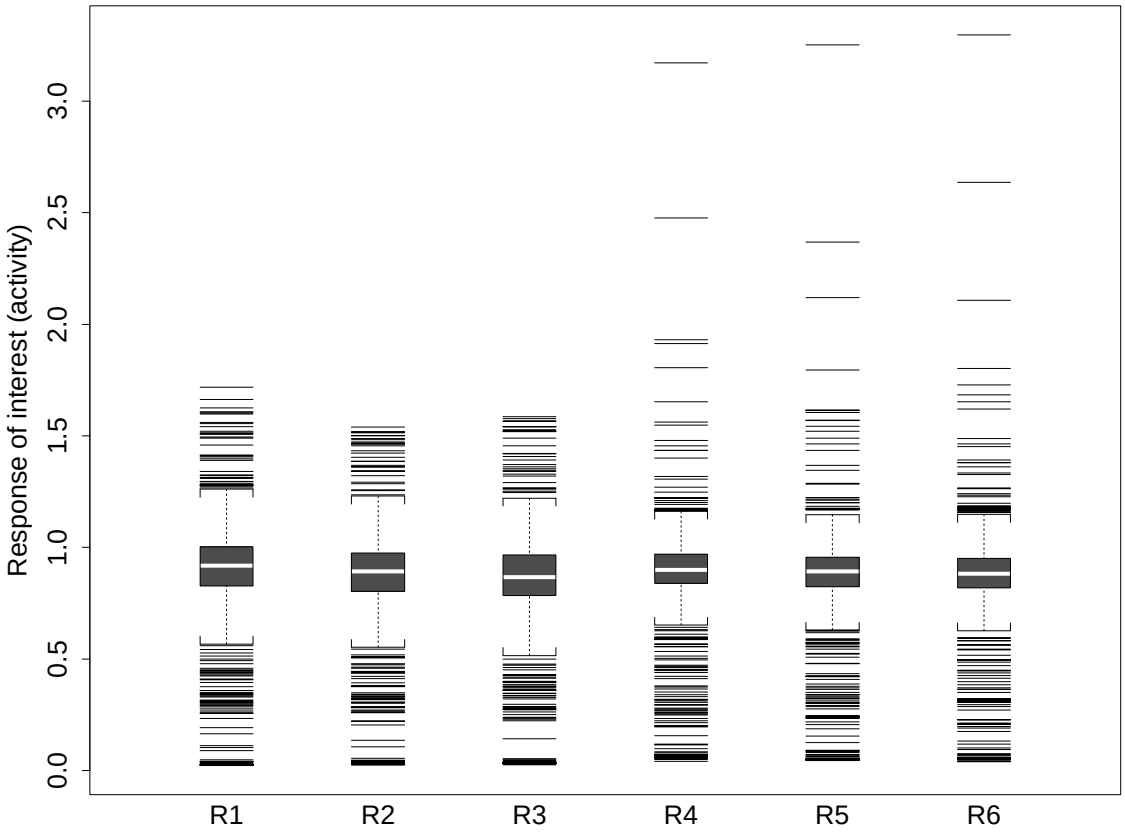

Figure 7.7: Boxplots for six repeated measurements.

convenience, and so on.

It might be desirable to correct some effects caused by the location of the compounds (such as groups, plates, rows, columns) if the design had been randomized. This is difficult to do for the data at hand. The compound effects are aliased with plates and with row/column effects. This makes it impossible to correct for rows/columns or plates without removing the compound effects partially. Therefore, we decide to use raw data without any correction.

In order to facilitate comparisons with the earlier approaches in Section 7.2 for the NCI AIDS data with a binary response, I partition the continuous response (property of interest) of R1-R3 and R4-R6 into categories by setting a threshold: the measurements larger than the threshold are observed active, and those less than it are observed inactive. Thresholds are separately chosen for R1-R3 and R4-R6, respectively, such that both observed proportions of actives are set to be the same as the NCI AIDS data (2% active). In this respect, the likelihood approach based on newly partitioned data can be readily compared to the NCI AIDS data.

Let A_{3m} indicate the observed number of actives among the three assayed replicates in the mth group of replicates and n_{ma} indicate the number of compounds in the data with $A_{3m} = a$ where $m = 1$ represents R1-R3 and $m = 2$ represents R4-R6. A_{3m} takes a value from 0 to 3. Similarly, π_m is the active rate in the mth groups of replicates; θ_{m+} and θ_{m-} are the false positive rate and false negative rate in the mth groups of replicates.

With the above set-up, similar to that in Section 7.1 the probability of observing $A_{3m} = a$ can be easily calculated:

$$P(A_{3m} = a) = \binom{3}{a}[(1-\theta_{m-})^a \theta_{m-}^{3-a}\pi_m + (1-\theta_{m+})^{3-a}\theta_{m+}^a(1-\pi_m)].$$

The observed log-likelihood function of R1-R3 or R4-R6 is a function of three parameters:

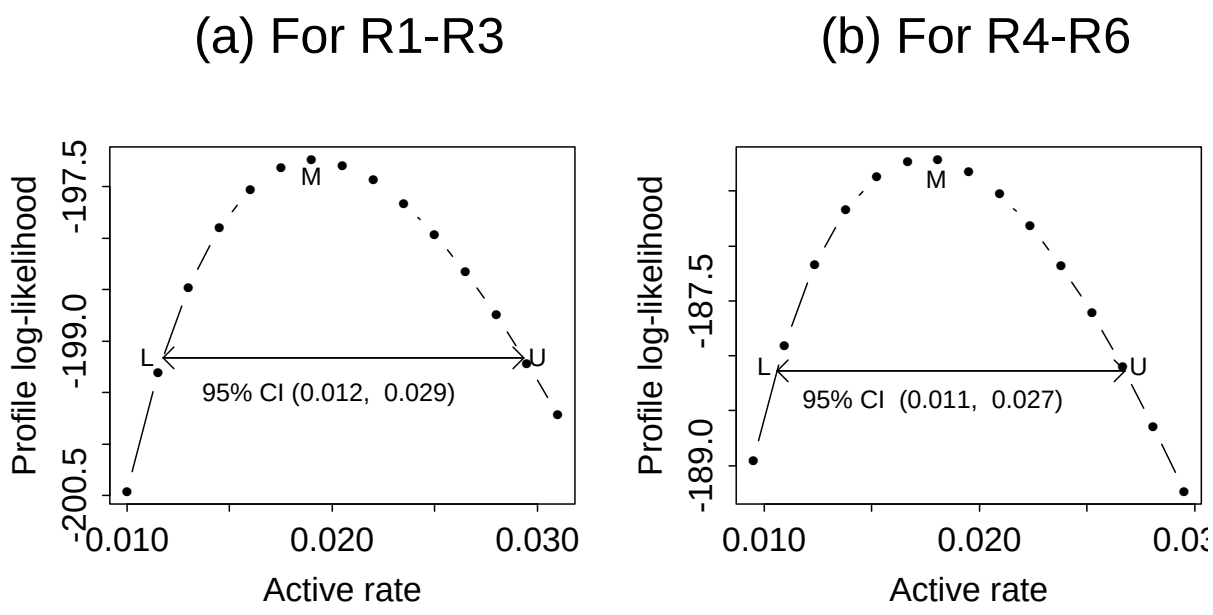

Figure 7.8: The 95% profile likelihood confidence interval for π_m. (a) for R1-R3: lower bound $\pi = 0.012$ (indicated by "L"), upper bound $\pi = 0.029$ (indicated by "U") and the maximum likelihood estimate $\pi = 0.020$ (indicated by "M"). (b) for R4-R6: lower bound $\pi = 0.011$ (indicated by "L"), upper bound $\pi = 0.027$ (indicated by "U") and the maximum likelihood estimate $\pi = 0.018$ (indicated by "M").

$$l(\pi_m, \theta_{m+}, \theta_{m-}) = \sum_{a=0}^{3} n_{ma} \log P(A_{3m} = a). \tag{7.11}$$

As in Section 7.2, the profile likelihood function on π_m can be constructed, and the 95% profile likelihood confidence interval is displayed in Figure 7.8: (a) is for R1-R3, and (b) is for R4-R6. Both confidence intervals for π_m are quite similar with a center at about 2% (the observed proportion of actives). Each of them is much narrower than that for the NCI AIDS data. Both R1-R3 and R4-R6 seem to have more information for estimating π, even though

each of them has a small number of assays (1046×3 =3138) compared to the 14906 assays in the NCI AIDS data.

I now similarly construct the confidence regions for θ_{m+} and θ_{m-}. Three representative values of π_m (the lower bound indicated by "L", the maximum likelihood estimation indicated by "M", and the upper bound indicated by "U") are chosen, as shown in Figure 7.8. For each chosen value of π_m, the contour plots are drawn in Figure 7.9. The first row corresponds to the contours for R1-R3 and the second row for R4-R6. All the contours look similar compared to the dramatic change of contours in the NCI AIDS data. However, a simple pattern follows. The false positive rate is small and in a reasonable range (0 - 0.015), but the false negative rate can be very large (up to 0.6). Different from those for the NCI AIDS data, the parameter estimates for both R1-R3 and R4-R6 do not change dramatically depending on the different values of π_m; furthermore, the estimates for R1-R3 and R4-R6 are comparable.

Table 7.3 displays the maximum likelihood estimates of θ_+ and θ_- for the NCI AIDS data, R1-R3, and R4-R6 under the three values of π. All the maximum likelihood estimates of θ_{m+} and θ_{m-} in both R1-R3 and R4-R6 are almost the same. The latter tends to have a slightly larger false positive rate but a slightly smaller false negative rate. However, the difference is very small. In contrast, the maximum likelihood estimates of θ_+ and θ_- for the NCI AIDS data change a lot. The repeated measurements lead to more reliable parameter estimation.

I also tried to combine R1-R3 and R4-R6 and treat it as one data set with six repeated measurements, and the parameter estimation is not much different. In particular, the range of estimates of the false negative rate is still quite large; the maximum likelihood estimate at some reasonable values of π can easily be above 0.2.

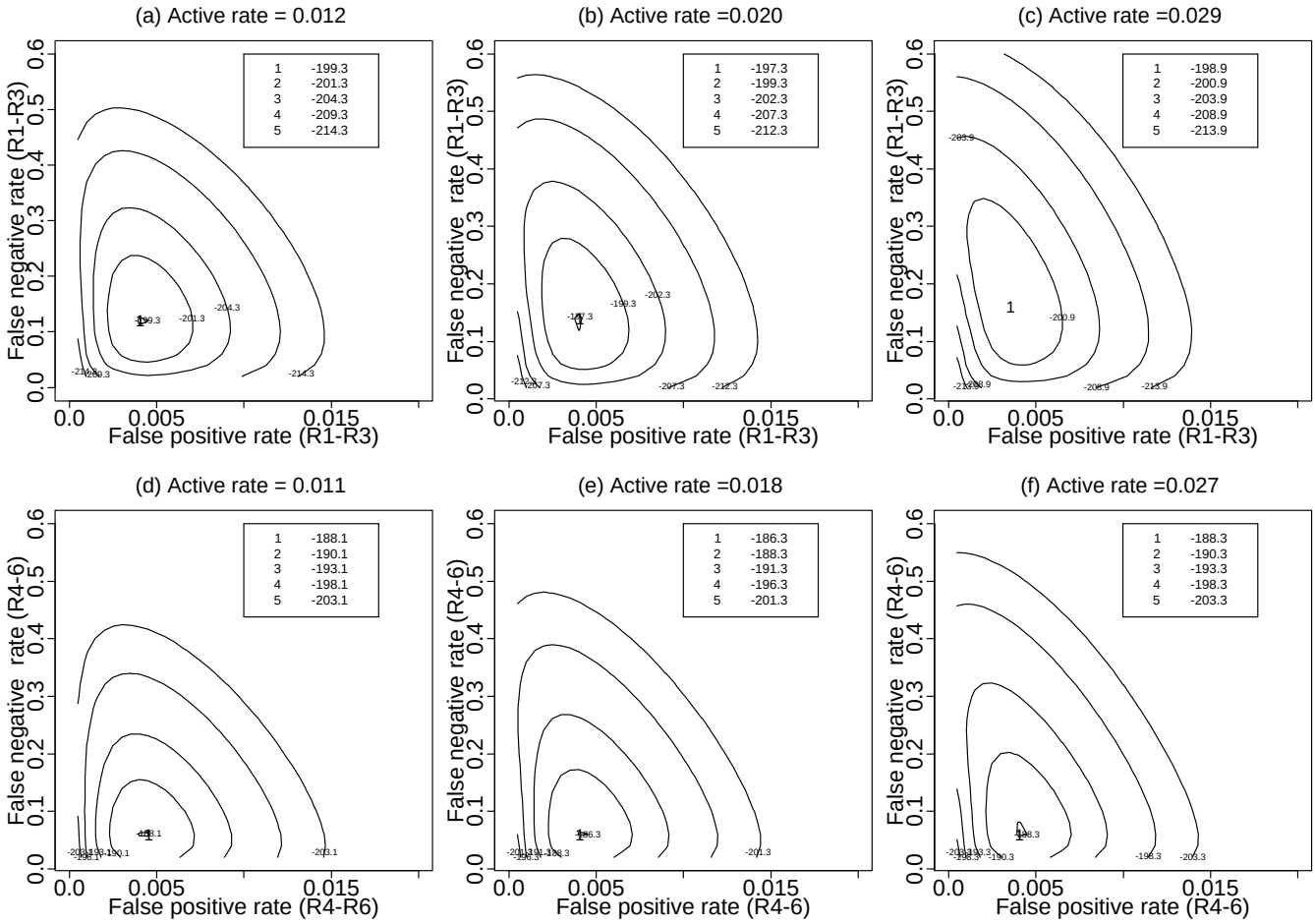

Figure 7.9: Contours for θ_{m+} and θ_{m-}. The first row is for R1-R3, and the second row is for R4-R6. 1: the local maximum of log-likelihood. The log-likelihood values defining the contours are listed in the box for each figure. The log-likelihood decreases as points move away from the mode.

Table 7.3: Maximum likelihood estimates of θ_+ and θ_- for the NCI AIDS data, R1-R3 and R4-R6 at three values of π. The second row of estimates corresponds to the mle values for all three parameters.

NCI AIDS data			R1-R3			R4-R6		
π	θ_+	θ_-	π_1	θ_{1+}	θ_{1-}	π_2	θ_{2+}	θ_{2-}
0.013	0.0081	0.05	0.012	0.004	0.12	0.011	0.0045	0.06
0.037	0.0001	0.45	0.020	0.004	0.14	0.018	0.004	0.06
0.051	0.0001	0.6	0.029	0.0035	0.16	0.027	0.004	0.06

7.5 Conclusion

This chapter describes a method to estimate activity rates and error rates using the likelihood approach. This method is applied to the NCI AIDS data and to the REPS data. By combining auxiliary information about repeat tests of the same compound, likelihood methods can extract interesting information about the magnitudes of the measurement errors made in the assay process. These estimates can be used to assess model performance, which sheds new light on how various models handle the large random or systematic errors often present in HTS data. A similar approach is also applied to two data sets with more repeat measurements. Clearly, repeated measurements provide much more information for estimating the parameters. On the other hand, although these considerations do not directly impact the choice of a predictive model, it is clear that improved assay quality will have a positive effect on predictive performance.

Chapter 8

Discussion and Future Research

8.1 Summary of the thesis

The basic goal of my thesis is to identify good statistical methods, to modify them, and to propose novel methods tailored to a particular drug discovery scenario. Drug discovery has many characteristics, but the thesis deals mainly with two problems:

- The data are extremely unbalanced, with the interesting category representing a small proportion of the data.
- One category (active) is much more interesting than the other (inactive). The most promising compounds must be ranked accurately. It is very different from the common classification problem, which pursues the least classification error.

Due to this imbalance and the need to rank compounds, the conventional classification criteria such as misclassification rate and deviance are no longer good standards for comparing statistical methods. Thus, AHR is introduced in Chapter 3 to take the place of those

common criteria and is shown to be very relevant to the goal of drug discovery. Most of the approaches in this thesis are based on AHR.

Chapter 2 claims that for the NCI AIDS data, local methods such as trees and KNN outperform GLM, NN, and MARS, which assume more smoothness. The superiority of both methods can be explained by the fact that the data present strong local behaviour and that the probability of activity can change very quickly as one moves through the space of explanatory variables. Furthermore, the most promising methods are good at ranking the active compounds ahead of others. The highest-ranked compounds appear at the beginning of the hit curve and contribute the most to the AHR. The ranking of most inactive compounds is relatively unimportant: as long as the ranking is low, the compound is neglected by hit curves and its contribution to AHR is negligible. Based on the above considerations, I propose to choose k adaptively in Chapter 4, which increases the model's flexibility and improves AHR for the BCUT data.

Obtaining accurate predictions of the probability of activity enables a better ranking of the compounds. However, it is not the only way. Chapter 5 argues that once the predicted activity is a monotone increasing function of the true activity, a perfect ranking is present. Therefore, the ranking problem has more tolerance of some types of bias. Subset averaging is a good example. When I average the predicted activity from the subsets of the augmented NCI AID data, more than half of them are junk predictions, and the final prediction is a biased estimate of the true probability of activity. However, this bias does not appear to degrade performance. A theoretical explanation has been provided to show that strong classifiers from relevant variables tend to dominate this aggregation method.

Chapter 6 presents further comparisons of k-varying, weighted KNN, and subset KNN methods on NCI AIDS data with a second descriptor set, and on a toxicity data set with

two different descriptor sets. Although different methods are best for the different data sets, some of the proposed methods are always amongst the best.

Estimating activity rates and error rates for assessing model performance in Chapter 7 sheds light on how various models handle large random or systematic errors in drug screening data.

8.2 Future research

As an algorithm, the k-varying algorithm is far from mature. It is designed for unbalanced data sets to pursue the largest AHR. Both the process of sequentially selecting k and AHR as a performance measurement make the algorithm hard to understand from a theoretical perspective while implementation is relatively straightforward. Although empirical evidence that the k-varying algorithm performs very well has been presented, it is still desirable to have more insight into the algorithm. Another important issue here is the possibility of modifying the method to make it easy to be used for balanced data.

KNN is a simple method, and some simple modifications can improve it. Weighting KNN, for instance, improves it. An interesting future direction would be a method that combines weighting and the k-varying algorithm.

After studying the classification in pursuit of low classification error, Friedman (1997) points out that the bias and variance components of the estimation error combine to influence the classification in a very different way than with the squared error on the probabilities themselves. If $\hat{y}$ is a predicted class label of y, the probability of one classifier making a wrong classification is

$$Pr(\hat{y} \neq y) = I(p < \frac{1}{2}) \int_{1/2}^{\infty} f(\hat{p}) d\hat{p} + I(p \geq \frac{1}{2}) \int_{-\infty}^{1/2} f(\hat{p}) d\hat{p},$$

where $f(\hat{p})$ is the density function of $\hat{p}$, an estimate of the probability of the activity function $p(\mathbf{x})$. Therefore, the boundary error depends on $f(\hat{p})$, which is subject to the variation of the training set and the sampled data under study.

Friedman makes a simple normality assumption on the distribution of $\hat{p}$,

$$f(\hat{p}) = \frac{1}{\sqrt{2\pi \mathrm{var}\hat{p}}} \exp(-\frac{1}{2} \frac{(\hat{p} - \mathrm{E}\hat{p})}{\mathrm{var}\hat{p}}),$$

and shows

$$Pr(\hat{y} \neq y) = \Phi(\mathrm{sign}(p - 1/2) \frac{\mathrm{E}\hat{p} - 1/2}{\sqrt{\mathrm{var}\hat{p}}}), \tag{8.1}$$

where

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{z}^{\infty} e^{-\frac{1}{2}u^2} du$$

is the upper tail area of the standard normal distribution.

(8.1) shows the mean classification error is a function of the first two moments of the estimate $\hat{p}$. However, it is no longer a simple additive effect with variance and a squared bias. Multiplicative effects are present. The formula is very complex, but the main result is that bias is not as important as variance and that variance tends to dominate bias in classification.

My thesis obtains similar results, although my goal is different. I am trying to rank the objects instead of making classifications. Further exploration of bias/variance trade-off on ranking would be a very interesting research topic. This is a challenging problem since the theoretical framework for ranking is very complicated. However, further theoretical

exploration might shed light on the ranking and classification problem and thus help find more efficient classifiers.

In the subset averaging method, some subsets are better than others in terms of model performance. Weighting the subsets according to their relevance to the response could improve the method. Dimension reduction is another interesting topic. Scientists are concerned about important subsets since such subsets are closely related to the substructures of the compounds. Promising drugs are usually made by regrouping such promising substructures. Dimension reduction is not an issue in my thesis because even the augmented NCI AIDS data have only 12 variables. However, descriptor sets can have hundreds of variables. A high dimension is a disaster to many classification methods. For example, due to intensive computation, it is almost impossible to perform subset averaging among all subsets. Efficient identification of the important subsets can be very helpful.

As mentioned above, my thesis concentrates on the ranking problem. The corresponding drug data are often extremely unbalanced and have strong local behaviour. In fact, there are many kinds of drug data. Moreover, there are different sets of descriptors such as Constitutional and Topological indexes, though my thesis focuses primarily on BCUT numbers. Experience tells us there is no dominant set of descriptors which is consistently better than the others on all or most drug data. Similarly, there is no dominant statistical method.

It would be very useful if I could generalize the interaction among data sets, descriptor sets and statistical methods. Feng et al. (2003) introduces some preliminary work, and the results suggest that certain descriptors work better with certain methods. This is also confirmed by the results in Chapter 7.

Appendix A

List of the Constitutional descriptors

No.	Symbol	Definition
1	MW	molecular weight
2	AMW	average molecular weight
3	Sv	sum of atomic van der Waals volumes (scaled on Carbon atom)
4	Se	sum of atomic Sanderson electronegativities (scaled on Carbon atom)
5	Sp	sum of atomic polarizabilities (scaled on Carbon atom)
6	Ss	sum of Kier-Hall electrotopological states
7	Mv	mean atomic van der Waals volume (scaled on Carbon atom)
8	Me	mean atomic Sanderson electronegativity (scaled on Carbon atom)
9	Mp	mean atomic polarizability (scaled on Carbon atom)
10	Ms	mean electrotopological state
11	nAT	number of atoms
12	nSK	number of non-H atoms

13	nBT	number of bonds
14	nBO	number of non-H bonds
15	nBM	number of multiple bonds
16	SCBO	sum of conventional bond orders (H-depleted)
17	nCIC	number of rings
18	nCIR	number of circuits
19	RBN	number of rotatable bonds
20	RBF	rotatable bond fraction
21	nDB	number of double bonds
22	nTB	number of triple bonds
23	nAB	number of aromatic bonds
24	nH	number of Hydrogen atoms
25	nC	number of Carbon atoms
26	nN	number of Nitrogen atoms
27	nO	number of Oxygen atoms
28	nP	number of Phosphorous atoms
29	nS	number of Sulfur atoms
30	nF	number of Fluorine atoms
31	nCL	number of Chlorine atoms
32	nBR	number of Bromine atoms
33	nI	number of Iodine atoms
34	nB	number of Boron atoms

35	nHM	number of heavy atoms
36	nX	number of halogen atoms
37	nR03	number of 3-membered rings
38	nR04	number of 4-membered rings
39	nR05	number of 5-membered rings
40	nR06	number of 6-membered rings
41	nR07	number of 7-membered rings
42	nR08	number of 8-membered rings
43	nR09	number of 9-membered rings
44	nR10	number of 10-membered rings
45	nR11	number of 11-membered rings
46	nR12	number of 12-membered rings
47	nBnz	number of benzene-like rings

Bibliography

[1] Breiman, L. 2001. "Random forests", *Machine Learning*, 45(1), 5–32.

[2] Breiman, L. 1996. "Bagging predictors", *Machine Learning*, 24(2), 123–140.

[3] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. 1984. *Classification and Regression Trees*, Wadsworth International Group, Belmont CA.

[4] Burden, F.R. 1989. "Molecular Identification Number For Substructure Searches", *Journal of Chemical Information and Computer Sciences*, 29, 225–227.

[5] Clark, L.A. and Pregibon, D. 1992. "Tree-Based Models" in *Statistical Models in S*, edited by J. M. Chambers and T. J. Hastie. Wadsworth & Brooks/Cole Advanced Books & Software Pacific Grove, California. 377–419.

[6] Craven, P. and Wahba G. 1979. "Smoothing noisy data with spline functions", *Numerical Mathematics*, 31, 377–403.

[7] Dasarathy, B.V. 1990. *Nearest neighbour (NN) Norms: NN pattern classification techniques,* Los Alamitos, Calif. : IEEE Computer Society Press.

[8] Fayyad, Piatesky-Shapiro, and Smyth. 1996. *Data Mining and Knowledge discovery in database: An overview,* Communications of the ACM 39(11), November.

[9] Feng, J., Lurati, L., Robinson, T., Wang, Y., Yuan, S., Ouyang, H., Young, S., Banks, T. 2003. "Predictive Toxicology: Benchmarking Molecular Descriptors and Statistical Methods", *J. Chem. Inf. Comput. Sci,* 43, 1463–1470.

[10] Freund, Y. and Schapire, R. 1997. "A decision-theoretic generalization of online learning and an application to boosting", *Journal of Computer and System Sciences*, 55, 119–139.

[11] Friedman, J. 2001. "Greedy function approximation: the gradient boosting machine", *Annals of Statistics*, 29, 1189–1232.

[12] Friedman, J., Hastie, T. and Tibshirani, R. 2000 "Additive Logistic Regression: a Statistical View of Boosting", *Annals of Statistics*, 28, 337–407.

[13] Friedman, J. 1999. "Stochastic gradient boosting", Technical report, Dept. Statistics, Stanford University.

[14] Friedman, J. 1997. "On bias, variance, 0-1 loss and the curse of dimensionality", *J. Data Mining and Knowledge Discovery*, 1, 55–77.

[15] Friedman, J. 1991. "Multivariate adaptive regression splines (with discussion)", *The Annals of Statistics*, 19, 1–141.

[16] Harman, D. K. (ed.) 2000. *Overview of the Eighth Text REtrieval Conference (TREC-8)*, Appendix, NIST Special Publication, A1.
http://trec.nist.gov/pubs/trec8/appendices/A/appendixa.cover.pdf

[17] Harman, D. K. (ed.) 1995. *Overview of the Third Text REtrieval Conference (TREC-3)*, NIST Special Publication, A5–A15.

[18] Hastie, T., Tibshirani, R., and Friedman, J.H. 2001. *The elements of statistical learning: data mining, inference, and prediction*, Springer, New York.

[19] Hastie, T. and Tibshirani, R. 1990. *Generalized additive models*, Chapman and Hall, London.

[20] Hawkins, D.M. and Kass, G.V. 1982. "Automatic Interaction Detection" in *Topics in Applied Multivariate Analysis*, ed. Hawkins, D. M.. Cambridge University Press, 269–302.

[21] Hawkins, D.M., Young, S.S., and Rusinko, A. 1997. "Analysis of a Large Structure-Activity Data Set Using Recursive Partitioning", *Quantitative Structure Activity Relationships* 16, 296–302.

[22] Hechenbichler K. and Schliep K.P. 2004. "Weighted k-Nearest-Neighbor Techniques and Ordinal Classification", Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich
http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps

[23] Jones-Herzog, D.K., Mukhopadyay, P., Keefer, C.E., and Young, S.S. 1999. "Use of recursive partitioning in the sequencial screening of G-protein-coupled receptors", *Journal of Pharmacological and Toxicological Methods*, 42, 207–215.

[24] Kalbfleisch J. G. 1985. *Probability and statistical inference*, Springer, New York.

[25] King, R. D., Muggleton, S., Lewis, R. A., and Sternberg, M.J.E. 1992. "Drug design by maching learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase", *Proc. Natl. Acad. Sci,* 89, 11322–11326.

[26] Klopman, G. 1984. "Artificial Intelligence Approach to Structure-Activity Studies. Computer Automated Structure Evaluation of Biological Activity of Organic Molecules", *American Chemical Society*, Vol. 106, No. 24, 7315–7321.

[27] Kooperberg, C., Bose, S. , and Stone, C. J. 1997. "Polychotomous regression", *Journal of the American Statistical Association*, 92, 117–127.

[28] Lam, R. 2001. "Design and Analysis of Large chemical Databases for Drug Discovery", Ph.D thesis presented to Department of Statistics and Actuarial Science, University of Waterloo, Canada.

[29] Levy, M.D. 2000. "The drug discovery and development process in the new millennium", *Journal of the Canadian Association of Gastroenterology*, Vol. 14, Issue 7.
http://www.pulsus.com/GASTRO/14_07/levy_ed.htm

[30] Livingstone, D. 1995. *Data analysis for chemists*, Oxford University Press.

[31] Mannhold, R. , Kubinyi, H. and Timmerman, H. (ed.) 2000. *Handbook of Molecular Descriptors*, WILEY - VCH.
http://www.disat.unimib.it/chm/dragon.htm

[32] Morgan, J. N. and Sonquist, J. A. 1963. "Problems in the analysis of survey data, and a proposal", *Journal of the American Statistical Association*, 58, 415–434.

[33] Parker, D. 1985. "Learning logic", *Technical Report TR-87*, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA.

[34] Parthasarthy, G and Chatterji, B. N. 1990. "A class of new KNN methods for low sample problems", *IEEE Transactions on System, Man and Cybernetics*, 20, 715-718.

[35] Pearlman, R. S. and Smith, K. M. 1999. "Metric validation and the receptor-relevant subspace concept", *Journal of Chemical Information and computer Sciences*, 39, 28–35.

[36] Pearlman, R. S. and Smith, K. M. 1998. "Novel software tools for chemical diversity", *Perspectives in Drug Discovery and Design,* 9/10/11, 339–353.

[37] Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, California.

[38] Ripley, B.D. 1996. *Pattern recognition and neural networks*, Cambridge University Press.

[39] Ripley, B.D. 1993. "Statistical aspects of neural networks." In *Networks and Choas -Statistical and Probabilistic Aspects*, eds O.E. Barndorff-Nielsen, J. L. Jensen and W. S. Kendall, Chapman & Hall, London, 40–123.

[40] Rusinko, A., Farmen, M.W., Lambert, C.G., Brown, P.L., and Young, S.S. 1999. "Analysis of a large structure/biological activity data set using recursive partitioning", *Journal of Chemical Information and Computer Sciences*, 39, 1017–1026.

[41] Service, R.F. 1996. "Combinatorial chemistry hits the drug market", *Science*, 272.

[42] Tatsuoka, K., Gu, C., Sacks, J., and Young, S. S. 1998. "Predicting extreme values in large datasets", Technical Report, National Institute of Statistical Sciences.

[43] Venables, W.N., and Ripley, B.D. 1999. *Modern Applied Statistics with S-plus*, 3rd edition. Springer, New York.

[44] Werbos, P. 1974. *Beyond Regression*, PHD thesis, Harvard University.

[45] Young, S. S., and Hawkins, D. M. 1998. "Using recursive partitioning to analyze a large SAR data set", *SAR and QSAR in Environmental Research*, 8, 183–193.