Efficient Kernel Methods for Statistical Detection

by

Wanhua Su

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, Canada, 2008

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

This research is motivated by a drug discovery problem – the AIDS anti-viral database from the National Cancer Institute. The objective of the study is to develop effective statistical methods to model the relationship between the chemical structure of a compound and its activity against the HIV-1 virus. And as a result, the structure-activity model can be used to predict the activity of new compounds and thus helps identify those active chemical compounds that can be used as drug candidates. Since active compounds are generally rare in a compound library, we recognize the drug discovery problem as an application of the so-called statistical detection problem. In a typical statistical detection problem, we have data $\{y_i, \mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i$ is the predictor vector of the ith observation and $y_i \in \{0, 1\}$ is its class label. The objective of a statistical detection problem is to identify class-1 observations, which are extremely rare. Besides drug discovery problem, other applications of statistical detection include direct marketing and fraud detection.

We propose a computationally efficient detection method called LAGO, which stands for "locally adjusted GO estimator". The original idea is inspired by an ancient game known today as "GO". The construction of LAGO consists of two steps. In the first step, we estimate the density of class 1 with an adaptive bandwidth kernel density estimator. The kernel functions are located at and only at the class-1 observations. The bandwidth of the kernel function centered at a certain class-1 observation is calculated as the average distance between this class-1 observation and its K-nearest class-0 neighbors. In the second step, we adjust the density estimated in the first step locally according to the density of class 0. It can be shown that the amount of adjustment in the second step is approximately inversely proportional to the bandwidth calculated in the first step. Application to the NCI data demonstrates that LAGO is superior to methods such as K nearest neighbors and support vector machines.

One drawback of the existing LAGO is that it only provides a point estimate of a test point's possibility of being class 1, ignoring the uncertainty of the model. In the

second part of this thesis, we present a Bayesian framework for LAGO, referred to as BLAGO. This Bayesian approach enables quantification of uncertainty. Non-informative priors are adopted. The posterior distribution is calculated over a grid of (K, α) pairs by integrating out β_0, β_1 using the Laplace approximation, where K and α are two parameters to construct the LAGO score. The parameters β_0, β_1 are the coefficients of the logistic transformation that converts the LAGO score to the probability scale. BLAGO provides proper probabilistic predictions that have support on (0,1) and captures uncertainty of the predictions as well. By avoiding Markov chain Monte Carlo algorithms and using the Laplace approximation, BLAGO is computationally very efficient. Without the need of cross-validation, BLAGO is even more computationally efficient than LAGO.

Acknowledgements

You Raise Me Up

When I am down and, oh my soul, so weary;
When troubles come and my heart burdened be;
Then, I am still and wait here in the silence, Until you come and sit awhile with me.

You raise me up, so I can stand on mountains;
You raise me up, to walk on stormy seas;
I am strong, when I am on your shoulders;
You raise me up... To more than I can be.

There is no life, no life without its hunger;
Each restless heart beats so imperfectly;
But then you come, and I am filled with wonder;
Sometimes I think, I glimpse eternity.
By Josh Groban

To my supervisors, Dr. Mu Zhu and Dr. Hugh Chipman, without whose insightful guidance, consistent encouragement and endless support, I would not have been able to finish this long journey. They are such wonderful mentors that I really enjoy working with them.

My special gratitude goes to Dr. Jerry F. Lawless, Dr. Ali Ghodsi, Dr. Radu Craiu, Dr. Kumaraswamy Ponnambalam, and Dr. Mary Thompson for being my thesis (or proposal) committee members. I also want to thank Dr. Stanley Young and Dr. William Welch for introducing us to the drug discovery problem.

I would like to thank all the professors, staffs, my fellow graduate students and all my friends in the Department of Statistics and Actuarial Science, University of Waterloo

for their helps in one way or another. I appreciate the inspiring discussion with Fengfei, Peng Zhang, Ker-ai, Yan Yuan, Sunny, Sofia and Longyang. I also thank Hui Shen, who "brought" me to the University of Waterloo; Shun-Fu, Situ, and Jane Shen, who provide help whenever I need and make my life in Canada much easier and more colorful.

All the computation is done on a cluster of computers in the Canada research center in mathematical modeling at the Department of Mathematics and Statistics, Acadia University. I wish to thank Mr. Duane Currie for teaching me how to run parallel jobs on the cluster, and other faculty members, staffs in the department for providing me a supportive and stimulating environment during my visits at Acadia.

For the first time, I really want to formally express my sincere and serious gratitude to my mom and my sister, who have always been my strongest support. Without their love, encouragement and understanding, I would not have been what I am.

Last, I would like to dedicate this thesis to my father, my grand-aunt and Dr. Mirriam Ross. This is not a perfect moment in my life because of the loss of them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are a lot of applications for detecting useful items from a huge database, in which all items can be classified as either object items labeled by 1 or background items labeled by 0. This is a typical binary classification problem. Any classification method, such as K-nearest neighbors (KNN), classification tree, support vector machines (SVM), is able to assign class labels to the data and, hence, identify those class-1 observations. In this thesis, I focus on the cases where the class-1 observations are extremely rare and define those applications as the statistical detection problems. Normally, the detected items will be passed on to the next stages for further investigations, which are relatively expensive. As a result, it might not be possible to test all the potential items. The ideal solution is to rank the items in descending order by their possibility of being class 1 and as a result those more promising items can be picked earlier.

Real examples of the detection problem include drug discovery, direct marketing, credit card fraud detection, etc. The objective of drug discovery is to select those lead chemical compounds (from a huge compound library) that can be used as drug agents to fight a certain disease; the goal of direct marketing is to identify targeted potential customers from a large population; and the objective of credit card fraud detection is to catch those fraudulent transactions from numerous transactions.

A novel method, LAGO, which stands for "locally adjusted GO estimator", has been proposed to handle the detection problem. The method was inspired by the game called "GO". LAGO is constructed in terms of basis functions, with each basis being a kernel function centered at a class-1 observation and whose bandwidth is determined by the average distance from this class-1 observation to its K-nearest class-0 neighbors. We have applied LAGO to the AIDS anti-viral data set, which is a real drug discovery data set from the National Cancer Institute (NCI). The results show that LAGO outperforms some powerful methods such as K-nearest neighbor and support vector machines. This work has already been published in Technometrics (Zhu, Su and Chipman, 2006).

Even though the value of the LAGO estimate is within (0, 1), it is not a probability. Moreover, it is only a point estimate and no uncertainty of the estimate is provided. In some applications, however, a correct ranking of the observations is not enough; what is needed in addition is an accurately calibrated estimate of the observations' true probabilities of being class 1; see Zadrozny and Elkan (2001a, b) for examples. In this thesis, we propose a Bayesian framework for LAGO (BLAGO) that not only provides a prediction of an item's probability of being class 1 but captures the uncertainty of this prediction as well.

The overall structure of the thesis is as follows. Chapter 2 describes the motivating drug discovery problem and how data mining techniques can be applied in drug discovery. A detailed description of the NCI AIDS anti-viral database, the main data set across the thesis, is also given in this chapter. Chapter 3 systematically describes the statistical detection problem – its set-up, properties and applications. Brief introductions of some common methods such as KNN, trees and SVM are given as well. I also discuss different performance metrics for statistical detection. In Chapter 4, our LAGO method is described in great detail, followed by performance comparisons of LAGO to KNN and SVM with application to the NCI AIDS data. We also compare the performance of LAGO with KNN and SVM on two simulated data sets. Chapter 5 discusses the motivation of casting LAGO into a Bayesian framework and shows our approach. Performance of BLAGO is

compared to LAGO and other methods on the NCI data and the two simulated data sets. Conclusions and suggestions for future research are given in Chapter 6.

Chapter 2

Drug Discovery and Data Mining

This chapter briefly describes the cycle of how a new drug is discovered and developed and how data mining techniques help in the early stage of drug discovery. A detailed introduction of the AIDS anti-viral database from NCI is given in this chapter as well.

2.1 Drug Discovery and Development

Drug discovery and development of new pharmaceutical products is a long and challenging process that involves various scientific and medical experts. Drug discovery and development can be broken into four main stages: drug discovery, preclinical test, clinical trial, and post-approval study.

New drugs are discovered in the laboratory by chemists, biologists, scientists and pharmacologists. There are two main tasks in drug discovery. The first one is to identify targets – those cellular and/or genetic factors that are associated with a specific disease. The second task is to search for chemical and biological substances that interact with the targets and hence are helpful in treating a particular disease. The entire drug discovery process includes the following activities:

- Target identification. The disease mechanism defines the possible cause(s) of a par-

ticular disorder, as well as the path or phenotype of the disease. Targets might be of various forms due to different disease mechanisms. For example, they can be some kind of bacteria, fungi, virus or they can be disease genes. Understanding the disease mechanism and identifying the target directs research and helps formulate a possible treatment to slow or reverse the disease process.

- Target validation. Tests are conducted to confirm that a target is critically involved in a disease process and that modulation of the target is likely to have a therapeutic effect.

- Lead identification. Lead/active compounds are those substances that have various interactions with the drug targets and hence are helpful in treating a certain disease. A compound's activity against a particular target can be obtained by either in-vitro (cell-based) or in-vivo (animal model) assays.

- Lead optimization. Studies either in-vivo or in-vitro are conducted to compare various lead compounds in their safety, effectiveness, and how they metabolize and affect the body. The most safe and effective lead compounds are selected for further investigation.

In the preclinical test stage, those lead compounds identified in drug discovery must be tested extensively in living organisms (in-vivo) and in cells in the test tube (in-vitro) to ensure their safety for humans. The results of preclinical tests are used by experts to determine how to best formulate the drug for its intended clinical effect. These results include the amount of drug required to effectively treat the disease (potency), extent to which the drug interacts with the target only rather than other healthy cells (selectivity), presence of any harmful side effects (toxicity), rate at which the drug works, and how long it stays effective (metabolism).

Successful drugs in the preclinical testing are passed on to clinical trials – testing on humans. Clinical trials usually consist of three phases:

- Phase I: Studies are conducted on a small number of healthy volunteers to verify safety and tolerability of the candidate drug. Testing includes observations and careful documentation of how the drug acts in the body – how it is absorbed, distributed, metabolized and excreted.

- Phase II: Further studies are designed for patients suffering from the disease or condition the drug intends to treat to determine effectiveness and safety of the candidate drug in humans. Most phase II studies are randomized double-blinded trials, in which neither the patients nor the investigators know who is receiving the investigational drug or placebo.

- Phase III: Studies are expanded to gather additional information about safety and effectiveness that is needed to evaluate the overall benefit-risk relationship of the drug.

Based on the results of the clinical trial, a document showing substantial evidence that the drug is safe and effective for its proposed use must be submitted to authorities for review, such as Food and Drug Administration (FDA) in the U.S.A. Phase IV trials are conducted after a drug has been approved to market. Studies focus on how well the proven drug works on a broader population, and what the long-term risks and benefits are.

The drug discovery and development process is designed to make sure that only those pharmaceutical products that are both safe and effective are brought to market. It takes the pharmaceutical companies 10 to 12 years on average to discover and present a new drug to the public. This thesis only focuses on the drug discovery. How techniques of data mining are applied in drug discovery will be discussed in the following section.

2.2 The Role of Data Mining in Drug Discovery

Detecting lead compounds is an important procedure in the early stages of drug discovery and development, because only those lead compounds can be used in further stages. A compound library is a collection of thousands or millions of compounds. Obviously, in order to find the most potential drug candidates, biochemists would like to screen as many compounds as possible. Although the high-throughput screening (HTS) technique makes it feasible to screen a large number of compounds against one or several targets in a single day, it is still costly to screen all the available compounds against all the possible targets. This is especially true in cases where the active compounds are extremely rare, which is a common scenario in drug discovery. Therefore, it is necessary to come up with new methods to speed up the assay process.

An alternative approach is to screen only part of the compounds and then build a model relating activity to compound descriptors. These compound descriptors characterize the compounds' chemical structure. With such a structure activity relationship (SAR) model, one can predict the activity of the remaining compounds and test only those most likely to be active (see Figure 2.1). Since the compound descriptors are generated by computer algorithms rather than physically measured, it is often cheap and fast to compute the values of the descriptors for all the compounds. As a result, the database describing the compounds' chemical structure may be enormous depending on the total number of compounds and the number of descriptors. Therefore, knowledge in data-mining such as dimension reduction techniques, modelling strategies and model assessment is useful.

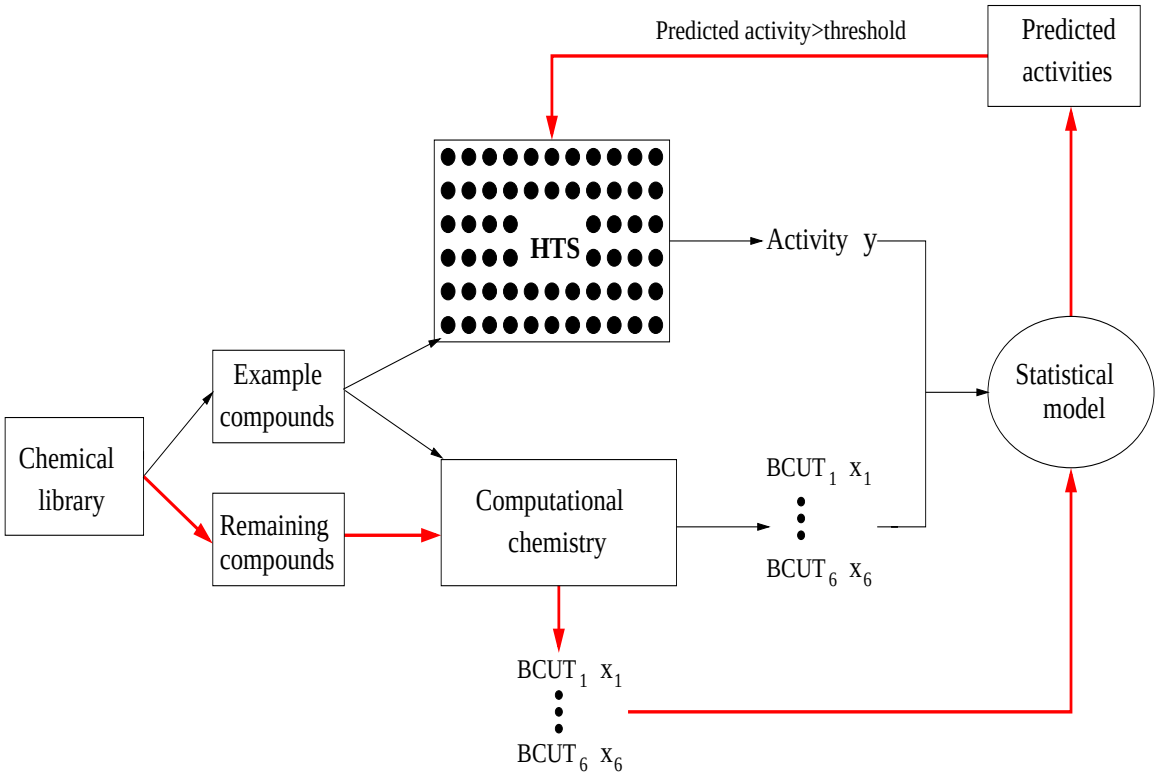

Figure 2.1: Data collection of the NCI database. A compound's activity is measured by HTS, and explanatory variables are produced by computational chemistry. Figure is modified based on an illustration in Welch (2002).

2.3 Data Sets

Two kinds of data sets are used in this thesis: the first one is the NCI AIDS anti-viral database, which is a real drug discovery data set; and the second one is simulated data. We generate two sets of simulated data. One is the "Mysim" data and the other is the the "Mysim-LAGO" data. In this section, however, only the Mysim data is introduced. The Mysim-LAGO data set will be discussed later in Chapter 4 after the method of LAGO has been introduced.

2.3.1 The NCI AIDS Anti-viral Database

The real data set we use is the AIDS anti-viral data set from the National Cancer Institute (NCI). The target of interest is the HIV-1 virus. The screen utilized a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection. Compounds providing at least 50% protection to the CEM cells were retested. Those retested compounds that provided at least 50% protection were listed as moderately active. Compounds that reproducibly provided 100% protection were listed as confirmed active. Six descriptors calculated by GlaxoSmithKline chemists were used to capture the compounds' chemical characteristics, such as their surface areas, bonding patterns, charges, hydrogen bond donor and acceptor ability. The NCI AIDS anti-viral database consists of one integer variable recording the compound ID, six continuous explanatory variables, and one categorical response with three possible values: 0 (inactive), 1 (moderately active) and 2 (confirmed active). There are 29,812 observations in total, among which only 393 are moderately active and 215 are confirmed active.

Those six continuous descriptors of the NCI data are called BCUT descriptors, which are credited to three research groups and individuals. Burden (1989) originally suggested using a matrix to represent the hydrogen-suppressed connection table of the molecule. To construct this connectivity matrix, hydrogen atoms are excluded and atomic numbers of

the non-hydrogen atoms are put on the diagonal. Off diagonal elements are assigned values of 0.1 times the nominal bond-type if two atoms are bonded, and 0.001 if the pair of atoms are not bonded. The two-lowest eigenvalues of the connectivity matrix are used to describe the chemical structure of the molecule. Burden actually mapped the molecules to a two-dimensional chemistry-space and argued that structurally similar compounds would be close to each other in this space. Rusinko and Lipkus (1993) verified Burden's proposal to be successful in finding structurally similar compounds with application to the Chemical Abstracts Service (CAS) Registry File. Based on Burden's (**B**) idea and CAS's (**C**) validation, Pearlman and Smith (1998) at University of Texas (**UT**) extended Burden's suggestion to a multi-dimensional chemistry-space by replacing atomic numbers on the diagonal with other more relevant atomic properties such as atomic charges, polarizabilities, H-bond donor- and acceptor-abilities. Stanton (1999) evaluated the effect of BCUT numbers as measures of molecule structure in quantitative structure-activity relationship (QSAR) study and found that BCUT metrics appear to perform better than some other promising descriptors in capturing structural information. Computational chemists in GlaxoSmithKline provided a set of 67 BCUT descriptors for the NCI data; however, these 67 descriptors are highly correlated (Lam, 2001; Lam *et al.*, 2002). In this thesis, I only focus on the NCI data with six BCUT descriptors which have much smaller correlations while preserving most information contained in the 67 BCUT descriptors.

We notice that some compounds have exactly the same BCUT numbers but different activity responses. These compounds are either the same compound measured more than once or different compounds that have the same chemical structure in this six-dimensional subspace. No matter which case might be, we decided to discard those compounds that share the same values of BCUT descriptors but conflict in activity results. Moreover, only one compound is kept for those compounds that were measured multiple times and had consistent activity results. After this data cleaning procedure, there are still 29,242 compounds remaining, among which only 378 are moderately active and 205 are confirmed

active. To make up a binary detection problem, we treat the inactive compounds as the class-0 observations and combine the moderately active and confirmed active together as the class-1 observations. So, we have 28,659 inactive and 583 active compounds in this non-replicated NCI data set. In the rest of my thesis, whenever I use the NCI data set, I refer to this non-replicated version.

2.3.2 The Mysim Data–A Simulated Data Set

The simulated data set is two dimensional, distributed within a square $[-3,3] \times [-1,5]$ with a distribution as follows (see Figure 2.2):

- The prior distributions for the two classes are $\pi_0 = P(y=0) = \frac{3}{4}$ and $\pi_1 = P(y=1) = \frac{1}{4}$.
- Given $y=0$, the covariates are uniformly distributed over the whole space $[-3,3] \times [-1,5]$. That is $P(\mathbf{x}|y=0) \sim \text{Uniform}([-3,3] \times [-1,5])$.
- Given $y=1$,

$$P(\mathbf{x}|y=1) \sim \begin{cases} \text{Uniform}([-2,-1] \times [3,4]) & \text{with probability } \frac{2}{5}, \\ \text{Uniform}([1,2] \times [0,1]) & \text{with probability } \frac{3}{5}. \end{cases}$$

Given the distribution above, we are able to generate the simulated data as follows: generate 300 class-0 observations from $[-3,3] \times [-1,5]$; generate 40 class-1 observations from region A, and another 60 class-1 observations from region B. As a result, there are 300 class-0 items and 100 class-1 items in this simulated data set. Note that regions A and B contain both class-1 and class-0 observations, while the remaining area contains only class-0 observations.

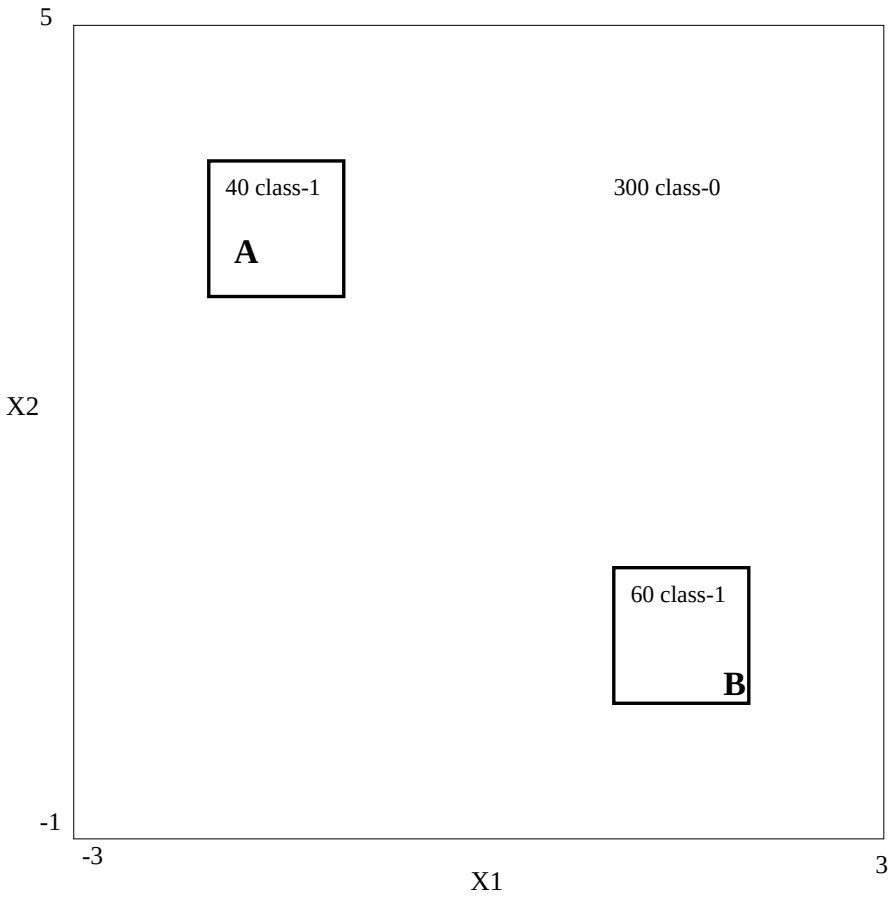

Figure 2.2: Generating mechanism of the Mysim data.

This simulated data set has a very good interpretation in drug discovery in that the two covariates x_1, x_2 can be viewed as two compound descriptors and the response y indicates whether the compound is active (y=1) or inactive (y=0); regions A and B can be viewed as two mechanisms that cause the compounds to be active. Besides identifying active compounds, distinguishing different activity mechanisms is also an important issue in drug discovery. In this thesis, however, I will focus only on the first issue—identifying the active compounds (the class-1 observations). Therefore, we combine items from both regions A

and B, and treat them as class-1 observations. It is straightforward to show

$$\begin{aligned} P(\mathbf{x} \in A|y=0) &= \frac{\text{area}(A)}{\text{Total area}} = \frac{1}{36} \\ P(\mathbf{x} \in B|y=0) &= \frac{\text{area}(B)}{\text{Total area}} = \frac{1}{36} \\ P(\mathbf{x} \in A|y=1) &= \frac{2}{5} \\ P(\mathbf{x} \in B|y=1) &= \frac{3}{5}. \end{aligned} \tag{2.1}$$

By Bayes' rule, we can also calculate an item's probability of being class 1, which is given by

$$\begin{aligned} P(y=1|\mathbf{x} \in A) &= \frac{P(\mathbf{x} \in A|y=1)P(y=1)}{P(\mathbf{x} \in A|y=1)P(y=1) + P(\mathbf{x} \in A|y=0)P(y=0)} = \frac{24}{29} = 0.8276 \\ P(y=1|\mathbf{x} \in B) &= \frac{P(\mathbf{x} \in B|y=1)P(y=1)}{P(\mathbf{x} \in B|y=1)P(y=1) + P(\mathbf{x} \in B|y=0)P(y=0)} = \frac{36}{41} = 0.8780. \end{aligned}$$

2.4 Conclusion

In this chapter I briefly introduce the whole process of how a new drug is discovered and developed; more detail is focused on how data mining techniques and statistical modeling can be applied in the early stage of drug discovery to speed up the screening for active compounds. The essential data set of the thesis, the NCI AIDS anti-viral database, is described in detail in this chapter. There are two basic simulated data sets used in my thesis, which I refer to as the "Mysim" data and the "Mysim-LAGO" data respectively. This chapter only gives the generating mechanism of the Mysim data, whereas the generating mechanism of the Mysim-LAGO data will be shown after the method of LAGO has been introduced in Chapter 4. I will discuss the statistical detection problem in the next chapter.

Chapter 3

Statistical Detection Problems

This chapter introduces the statistical detection problem in great detail, including its set-up, objective, applications, methods and model assessment criteria.

3.1 Introduction

The set up of a statistical detection problem is the same as a binary classification problem. For each observation, we have a vector of predictors, $\mathbf{x}_i$, and the class label y_i, indicating whether the item belongs to the class of interest or not, 1 for yes, 0 for no. There are two main reasons that we would like to treat statistical detection as a special problem. Firstly, the frequencies of the two classes are extremely unbalanced. There are only a few class-1 observations, say 1-2% or even less. In a binary classification problem, however, the proportions of the two classes are roughly half and half. Secondly, the objective of statistical detection is to assign a relative ranking to the data according to their possibility of being class 1, and hence we can identify the class-1 items as early as possible. This is very different from a two-class classification application whose objective is to assign class labels to the observations regardless of whether the more potential ones will appear earlier in the ranking list or not.

Three typical applications of statistical detection are:

1. Drug Discovery: Here $\mathbf{x}_i$ is a vector of descriptors for a chemical compound and y_i indicates whether the compound is considered an active drug agent for a certain disease. Most compounds are inactive and we are interested in detecting the active ones.

2. Credit Card Fraud Detection (Bolton and Hand, 2002): Here $\mathbf{x}_i$ is a vector of descriptors for a credit card transaction and y_i indicates whether the transaction is fraudulent. Most transactions are not fraudulent and we are interested in catching the frauds.

3. Direct Marketing: Here $\mathbf{x}_i$ is a vector of descriptors for a potential customer and y_i indicates whether the potential customer will respond to the advertisement or not. Most clients will not respond and we are interested in identifying the most likely respondents.

3.2 Some Common Methods

K-nearest neighbors (KNN) and tree models were shown to be among the most effective methods for drug discovery (Wang, 2005). Support vector machine (SVM) has a close relationship with our proposed method. This section gives a brief introduction on these methods.

3.2.1 K-Nearest Neighbors (KNN)

The idea of KNN (Fix and Hodges, 1951; Cover and Hart, 1967) is very simple and intuitive. Under the assumption that points close to one another should have a similar response, KNN classifies a new observation according to the class labels of its K-nearest neighbors. In order to identify the neighbors, we must decide how to measure the proximity among

points and how to define the neighborhood. The most popular distance metric is the Euclidean distance. The region of the neighborhood is normally controlled by a tuning parameter, K, which can be chosen by cross-validation.

Figure 3.1 illustrates how KNN works. In this simulated example, we set $K = 5$. Among those five nearest neighbors of the test point tp1 (indicated by symbol '?'), four out of five belong to class 0. Therefore, tp1 is classified as a class-0 observation with an estimated probability of 4/5. Similarly, tp2 (also indicated by a '?') is classified as class-1 with an estimated probability of 4/5.

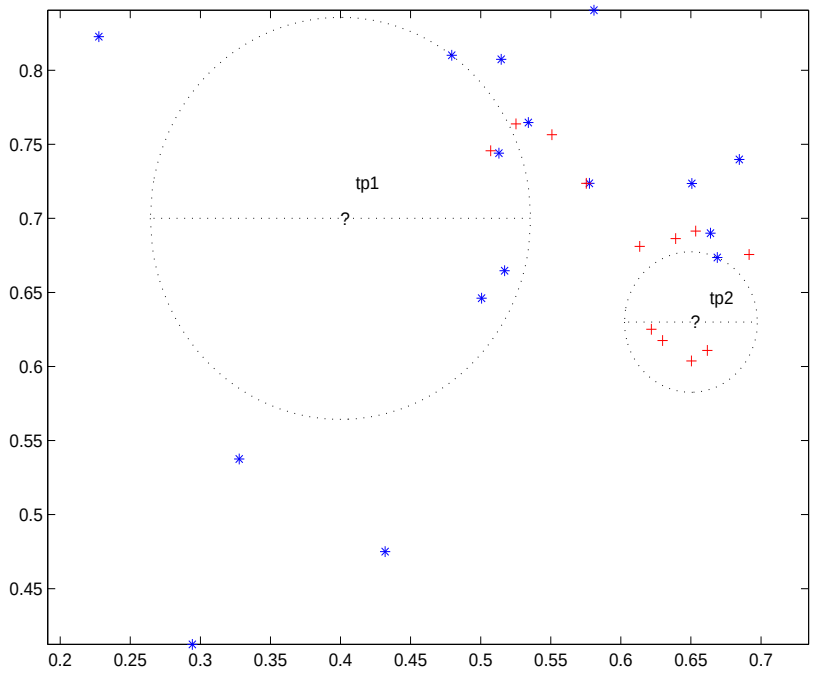

Figure 3.1: Simulated example illustrating KNN with $K = 5$. Here '$*$' denotes the class-0 observations, '+' denotes class-1 observations in the training set and '?' denotes the test data tp_1 and tp_2.

Though simple, KNN is a very powerful method, particularly for those data having some types of special local structure. On the other hand, it has some problems as well:

- The Euclidean distance could be heavily influenced by unimportant explanatory variables. This problem is especially serious in a high dimensional space. Furthermore, Euclidean distance is not scale-invariant. Wang (2005, Chapter 5) proposed a so-called subset KNN method where she suggested building multiple classifiers from subsets of variables and aggregating them into a single overall classifier. She showed that subset KNN might deal well with many weak predictors and its performance is not greatly affected by the curse of dimensionality.

- The method might break down when dealing with categorical data or a mixture of continuous and categorical variables. A meaningful distance metric must be defined for these two cases.

- For a fixed number of nearest neighbors, K, the radius of the neighborhood depends on the density of data. This raises the question of whether K or the radius should be used to define the neighborhood. To address this issue, Wang (2005, Chapter 4) suggested choosing a different value of K adaptively for each test point depending on the density of its neighborhood and the density of the training data as well.

- KNN can be considered to be a weighted average, with equal weights for the K nearest observations and zero weight for all the others. The abrupt change in weight may lead to instability in predictions. Hechenbichler and Schliep (2004) proposed and implemented a weighted KNN method which combines KNN with kernel weights.

3.2.2 Support Vector Machines (SVM)

Support vector machine (SVM) is a novel learning method originally introduced by Cortes and Vapnik (1995). It includes polynomial classifiers, radial basis function (RBF) networks, and single-layer neural networks as special cases. In a binary classification problem, suppose we have data $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, -1\}$. Notice that the two classes here are $\{1, -1\}$ rather than $\{0, 1\}$ in a typical binary classification problem.

The data are said to be linearly separable if there exists a hyperplane $f(\mathbf{x}) = 0$ that perfectly separates the two classes; otherwise, the data are linearly nonseparable. The real-valued function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is called the *decision function* (DF), which can be written as

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \\ &= \mathbf{w}^T \mathbf{x} + b. \end{aligned}$$

A new observation $\mathbf{x}$ is assigned to class 1 if $f(\mathbf{x}) \geq 0$ and otherwise to class -1. The parameters $(\mathbf{w}, b)$ can be estimated from the data.

When the data are linearly separable, we can find some hyperplane that perfectly separates the two classes. Let the margin of a hyperplane $(\mathbf{w}, b)$ be 2γ, where γ is the shortest distance from the hyperplane to a training point. The objective is to find the hyperplane that produces the largest margin. Figure 3.2 shows two hyperplanes and nine simulated data points which are linearly separable. Note that each hyperplane is halfway between the two solid lines. The hyperplane with margin_1 is better than the one with margin_2 in that it has a larger margin. Please note that most of the important results and derivations are based on three books: Vapnik (1995), Cristianini and Shawe-Taylor (2000), and Hastie *et al.* (2001).

It can be shown that finding the maximal margin hyperplane $(\mathbf{w}, b)$ is equivalent to solving the following optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i = 1, \ldots, N. \end{aligned} \tag{3.1}$$

When the data are linearly nonseparable, no solution exists for (3.1). One way to deal with this problem is to still minimize $\frac{1}{2}\|\mathbf{w}\|^2$ while relax all the constraints by introducing some *slack variables* $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)$. These slack variables allow some observations to be

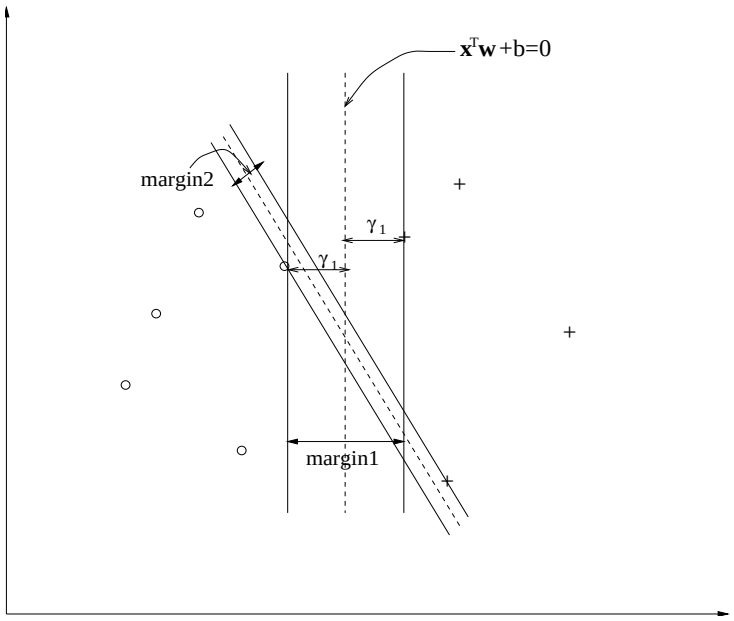

Figure 3.2: Two hyperplances of the nine simulated data points. The dashed lines are the separating hyperplanes (or decision boundaries). The hyperplane with margin_1 is better, since $\text{margin}_1 > \text{margin}_2$. Here, $\text{margin}_i = 2\gamma_i, i = 1, 2$.

on the wrong side of the margin (see Figure 3.3). The optimization problem for linearly nonseparable case is

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, N, \end{aligned} \tag{3.2}$$

where C (short for *Cost*) is a regularization parameter controlling the smoothness of the boundary. A large value of C will discourage any positive ξ_i and result in an overfit wiggly boundary; a small value of C will lead to an over smooth boundary. This trade-off enables our choosing the optimal value of C by cross-validation.

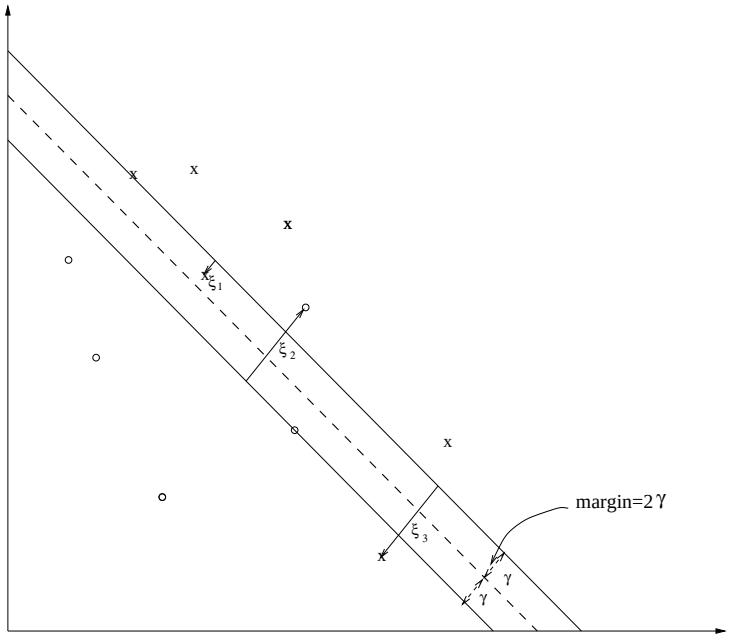

Figure 3.3: The slack variables for eleven simulated data points. Three of them are positive ($\xi_1, \xi_2,$ and ξ_3), and the other eight are zero. The dashed line is the hyperplane with a margin of 2γ.

By using the Lagrangian method, both (3.1) and (3.2) can be rephrased as quadratic programming problems with linear inequality constraints. It can be shown that the solution for $\mathbf{w}$ has the form

$$\hat{\mathbf{w}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i,$$

where $\alpha_i \geq 0$ are Lagrange multipliers. Those observations with strictly positive coefficients α_i are called *support vectors* (SV), since the solution hyperplane depends on these vectors alone. Any margin point (those SV with $\alpha_i > 0$ and $\xi_i = 0$) can be used to solve for b.

Given $\hat{\mathbf{w}}$ and $\hat{b}$, the decision function is given by

$$\begin{aligned} \hat{f}(\mathbf{x}) &= \hat{\mathbf{w}}^T\mathbf{x} + \hat{b} \\ &= \sum_{\mathbf{x}_i \in \mathrm{SV}} \hat{\alpha}_i y_i \mathbf{x}_i^T \mathbf{x} + \hat{b}. \end{aligned} \tag{3.3}$$

So far, we have focused on SVM with a linear boundary. However, a linear boundary is often not flexible enough to separate the two classes. Fortunately, SVM can be generalized easily to construct nonlinear boundaries. The common strategy is to map the original data into a high dimensional space and then construct a linear boundary classifier in the transformed space. The original space of the data is called the *input space* while the transformed high dimensional space is called the *feature space*. Figure 3.4 shows an example of a feature mapping from a two dimensional input space to a two dimensional feature space. The data can not be separated by a linear function in the input space, but can be in the feature space under the mapping Φ. Although the same dimension is used in Figure 3.4 for illustration, the feature space is usually of much higher dimension than the input space.

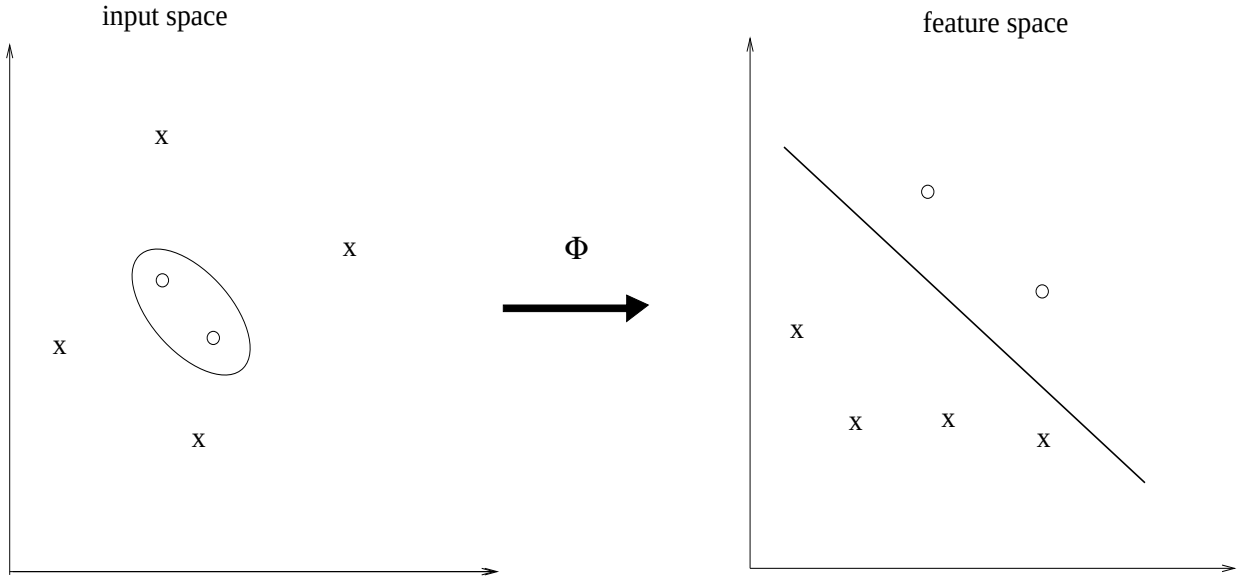

Figure 3.4: An example of mapping where data can be separated by a linear function in the feature space but can not in the input space.

With a nonlinear mapping Φ, SVM is able to produce nonlinear boundaries in the input space by constructing a linear boundary in the feature space. The decision function of SVM now becomes

$$\hat{f}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathrm{SV}} \hat{\alpha}_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + \hat{b}, \tag{3.4}$$

which is in terms of inner products in the feature space. Calculating the inner product of $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})$ might be expensive when the feature space is of high dimension. Fortunately, to calculate (3.4), we do not need to know Φ explicitly but only need to know how to evaluate the inner products $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})$. This can be done by using a suitable kernel function. A kernel function $\mathcal{K}$ is defined as

$$\begin{aligned} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\ &= \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j), \end{aligned}$$

where $\mathbf{x}_i, \mathbf{x}_j$ are two points in the input space. Many different forms of $\mathcal{K}$ are possible, each leading to a different feature space. Mercer's theorem (Mercer, 1909) provides one way to construct kernels. It says that a symmetric function in the input space, $\mathcal{K}$, is a kernel function if and only if the Gram matrix

$$\mathbf{K} = [\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^N$$

is positive semi-definite, i.e., has non-negative eigenvalues. Three common kernels are:

1. polynomial: $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$.
2. radial basis: $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$.
3. sigmoid: $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$.

Here, γ, r and d are the kernel parameters that need to be tuned by the training data.

Given the kernel function $\mathcal{K}$, the decision function (3.4) can be written as

$$\begin{aligned} \hat{f}(\mathbf{x}) &= \sum_{\mathbf{x}_i \in \mathrm{SV}} \hat{\alpha}_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + \hat{b} \\ &= \sum_{\mathbf{x}_i \in \mathrm{SV}} \hat{\alpha}_i y_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + \hat{b}. \end{aligned}$$

SVM has been implemented in R (Meyer, 2007), which makes use of the C++ implementation of SVM by Chang and Lin (2007). The function *svm* implemented in R takes the majority class as class 1 and the minority one as class -1 by default; therefore, a data point which is far away from the separating hyperplane on the negative side casts a lot of confidence that this item belongs to the rare class, and hence should be ranked earlier. For statistical detection, items can be ranked by their *signed distances* to the hyperplane in an increasing order. Those items having a smaller signed distance to the hyperplane would be detected earlier, because they are more likely to be in the rare class.

The signed distance from one point $(\mathbf{x}_i, y_i)$ to a hyperplane $(\mathbf{w}, b)$, $\mathrm{sd}(\mathbf{x}_i; \mathbf{w}, b)$, is calculated as

$$\begin{aligned} \mathrm{sd}(\mathbf{x}_i, \hat{\mathbf{w}}, \hat{b}) &= \frac{1}{\|\hat{\mathbf{w}}\|}(\hat{\mathbf{w}}^T \mathbf{x}_i + \hat{b}) \\ &= \frac{\hat{\mathbf{w}}^T \mathbf{x}_i + \hat{b}}{\sqrt{\hat{\mathbf{w}}^T \hat{\mathbf{w}}}}. \end{aligned} \tag{3.5}$$

3.2.3 Asymmetric Support Vector Machines (ASVM)

In statistical detection, the numbers of observations in two classes are unbalanced. Some researchers (e.g., Veropoulos *et al.*, 1999) have proposed using different penalty parameters (C_+, C_-) to differentiate the cost of false positives from that of false negatives. Suppose that w_+ and w_- are the weights for class 1 and class -1 respectively. Optimization problem

(3.2) becomes

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=-1} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, N, \end{aligned} \tag{3.6}$$

where $C_+ = Cw_+, C_- = Cw_-$. Solving (3.6) is almost the same as solving (3.2). The only difference is that we have to rewrite constraints involving C as constraints with C_+ for $y_i = 1$ and with C_- for $y_i = -1$. We call this the asymmetric support vector machine (ASVM), which can be fitted using the function *svm* in R package {e1071} with four tuning parameters — γ, C, w_+, and w_-. Without loss of generality, we assume $w_+ = 1$ and tune the remaining three parameters, γ, C, and w_-, simultaneously.

3.2.4 Tree-Based Models

As a powerful and conceptually simple method, tree-based models date back at least as early as Morgan and Sonquist (1963). Hawkins and Kass (1982) and Breiman *et al.* (1984) brought statisticians' attention to tree-based methods (i.e., classification and regression trees), and Quinlan (1993) made trees popular in machine learning community.

The basic idea of tree methods is very simple. Through binary recursive partitioning, a tree successively partitions the feature space into a set of rectangles and then fits a simple model (say a constant) in each one. At each partition, the algorithm automatically determines the optimal split (including the splitting variable and the splitting point) such that the resulting two subsets of data are as homogeneous as possible with respect to the response of interest. This partition process is repeated until the stopping criterion is satisfied. It is obvious that a very large tree might overfit the data, while a too small tree might not be able to capture the important structure of the data. The preferred strategy to control the tree size is to grow a large tree and then prune this large tree using some cost-complexity criterion, which is a trade-off between the tree size and the goodness of fit

to the data. See Breiman *et al.* (1984) or Ripley (1996) for details.

The most common tree-based methods are the regression tree and the classification tree which model continuous response and categorical response respectively. When growing a regression tree, the optimal split is the one that minimizes the residual sum of squares; when building a classification tree, the optimal split can be chosen by minimizing misclassification error or Gini index or deviance (cross-entropy).

A major advantage of the recursive binary tree is its interpretability. One major problem with trees is their instability — a small change in the data might lead to a very different set of splits. Breiman (1996) proposed the idea of bagging to reduce the high variance of trees by averaging over many trees.

3.3 Model Assessment

Many criteria can be used to evaluate the performance of different models. Caruana and Niculescu-Mizil (2004) conducted an empirical study to compare nine boolean classification performance metrics using a variety of models. The performance metrics include accuracy, lift, F-score, area under the ROC curve, average precision, precision/recall break-even point, squared error, cross entropy, and probability calibration. They argued that these nine criteria can be grouped into three categories—squared error, cross entropy and calibration are suitable for cases where probabilistic prediction is of interest; area under the ROC curve, average precision, break-even point, and lift are proper for applications where the relative ranking of the predicted value is of major interest; accuracy and F-score are appropriate when the response depends on whether a prediction is larger than a threshold or not. This section introduces several widely used performance metrics and discusses what criteria should be used for statistical detection.

3.3.1 Misclassification Rate

For classification problems, a natural criterion to assess the performance of a model is the misclassification rate (MR). The misclassification table for a binary case can be presented as a 2×2 table:

True	classification result	
label	0	1
0	a (true negative)	b (false positive)
1	c (false negative)	d (true positive)

And the misclassification rate is given by

$$\text{MR} = \frac{b+c}{a+b+c+d}. \tag{3.7}$$

The misclassification rate, however, is not a suitable performance metric for statistical detection due to the rarity of class-1 observations. Take the NCI data for example. Suppose there are 14,622 observations in the test set among which only 292 are active. If a classifier M_0 simply classifies every test compound as inactive. It will misclassify all the 292 active compounds. The resulting misclassification rate of M_0 is $292/14,622$. Suppose another classifier M is applied to the NCI data and the results are summarized in Table 3.1. The misclassification rate of M is $(22+270)/14,622$, which is the same as M_0. However, this does not mean that these two classifiers have the same performance. The classifier M is definitely more useful than M_0 because 22/(22+22) compounds that M classifies as "1" are actually active, while M_0 can not identify a single active compound. The misclassification rate fails to distinguish superior and inferior classifiers when one class is extremely rare, and thus is not suitable for statistical detection.

True label	classification by M	
	0	1
0	14,308	22
1	270	22

Table 3.1: Misclassification table of classifier M for the NCI data.

3.3.2 Hit Curve

A hit curve is constructed as follows: we first rank the items in a non-increasing order by $\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_N)$, their estimated probabilities of being class 1. The larger a $\hat{p}_i$ is, the earlier the corresponding item is selected. A selected case is called a *hit* if it actually belongs to class 1. A function $h(n)$ gives the number of hits among the first n selected items. Plotting $h(n)$ versus n gives the hit curve. For classifiers producing a lot of tied predicted values, interpolation is applied to draw the hit curve. Take KNN for example, if $K = 4$, there will be only five distinct predicted values (i.e., 0/4, 1/4, 2/4, 3/4, 4/4). Ranking items having the same predicted value in a different order will produce very different hit curves. In order to honestly reflect the performance of a classifier, instead of plotting $h(n)$ versus n, we draw a straight line from $(n_{i-1}, h(n_{i-1}))$ to $(n_{i-1} + n_i, h(n_{i-1} + n_i))$ if $n_i \geq \frac{N}{10}$, where n_i is the number of observations having the ith largest $\hat{p}$ and N is the total number of items.

Figure 3.5 shows some typical hit curves. In this illustration, there are 1000 candidates; only 100 of them belong to class 1. The dash-dotted curve is an ideal curve; every item selected is an actual hit until all potential hits are exhausted. The solid curve is that of random selection. The dotted curve is that of some statistical detection method. Hit curves are also known as *gain charts* in some data mining applications. Hit curves appear to be somewhat similar to the receiver operating characteristic (ROC) curves which will be described in Section 3.3.4. The relationship between hit curves and ROC curves will be

discussed later in Section 3.3.5.

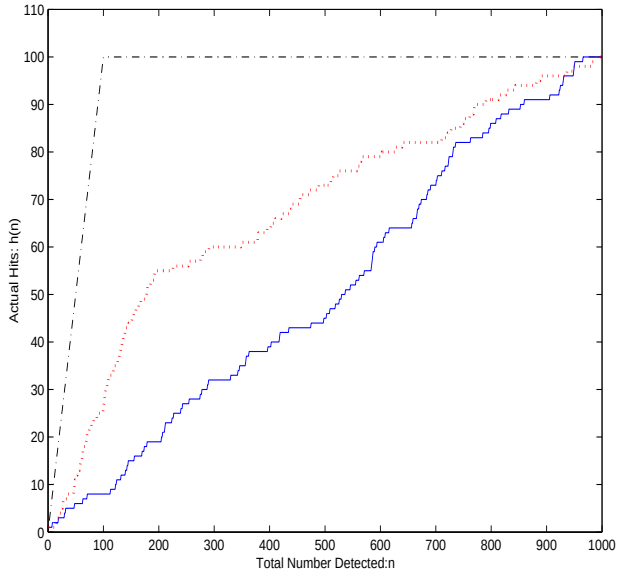

Figure 3.5: Illustration of several typical hit curves. The dash-dotted curve is an ideal curve. The solid curve is that of random selection. The dotted curve is that of a typical statistical detection method.

Two natural numeric summaries of a given hit curve are $h(N)$ (the number of hits among the first N selected items) and $H(N)$ (the area under the hit curve $h(n)$ up to a certain point N). The number of item selected N is often determined by one's budget for further investigation. For both measures, the larger the better.

Considering $h(N)$ alone, however, can sometimes be misleading. Figure 3.6 shows one of such situations. The dotted curve $h_d(n)$ and the solid curve $h_s(n)$ are the hit curves of two hypothetical detection methods. If we set $N = 500$ (the vertical line), then $h_s(N) > h_d(N)$, indicating that the method corresponding to the solid curve is better, but this is not necessarily true. The method corresponding to the dotted curve is more efficient in that it makes much more hits early on. If the budget allows us to assay only 300 cases, the method corresponding to the dotted curve becomes superior to the one corresponding to the solid curve.

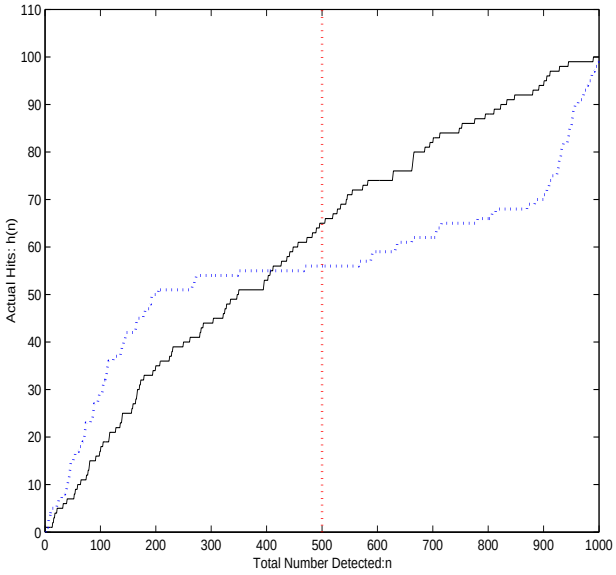

Figure 3.6: Hit curves for two different detection methods. Here, $h_s(500) > h_d(500)$ whereas $H_s(500) < H_d(500)$.

Generally, $H(N)$, the area under the hit curve favors methods that make more hits early on. In Figure 3.6, $H(500)$ is larger for the dotted line since it makes more hits until $n = 400$. The conclusions drawn from the number of hits $h(N)$ or the area under the hit curve $H(N)$ will vary at different values of N. If we consider all the observations in the test set, i.e., $N = 1000$ in Figure 3.6, then both methods have $h(1000) = 100$, the number of class 1 cases. Therefore, $h(N)$ can not be used to compare the overall performance of different methods.

Recall those two different detection methods in Figure 3.6. The area under the solid curve $H_s(1000)$ is larger than $H_d(1000)$, the area under the dotted curve. However this does not necessarily mean that the method referring to the solid curve is better. Practitioners would prefer the dotted curve if they can afford no more than 400 tests. Therefore, a single-valued measure that even more favors early hits than $H(N)$ is desirable. The average precision which will be introduced in the next section meets this need.

3.3.3 Average Precision

The average precision (AP) is defined as

$$\begin{aligned} \text{AP} &= \frac{1}{n_1}\sum_{n=1}^{N} y_{(n)}\frac{\sum_{j=1}^{n} y_{(j)}}{n} \\ &= \frac{1}{n_1}\sum_{n=1}^{N}\frac{y_{(n)}h(n)}{n}, \end{aligned} \tag{3.8}$$

where n_1 is the total number of class-1 observations and $y_{(n)}$ is the class label of the detected item having the nth largest $\hat{p}$. AP is a commonly used measure in information retrieval (see Singhal, 2001). Precision is defined as the percent of retrieved items that are actual hits, or $h(n)/n$; AP can be viewed as the average of the precisions over the points where a retrieved item is a hit. The information retrieval problem can be regarded as a statistical detection problem in some sense. The relevant documents are the class-1 observations which are relatively rare compared to the irrelevant documents. Searching for relevant items in a large database is analogous to detecting the class-1 observations from a huge data set.

The calculation given in (3.8) implies that the range of AP is $(0, 1]$. It is straightforward to show that the AP for the ideal hit curve in Figure 3.5 (the dash-dotted curve) is 1 and that the AP for random selection is π, the fraction of class-1 observations (See Zhu, Su and Chipman, 2006). Table 3.2 illustrates how to calculate the APs of two ranking models for a test set with six observations and three of them belonging to class 1. Model 1 is a more effective ranking model than Model 2 since it gives a larger AP. As we can see, AP is an overall performance measure that favors methods making more hits early on. Although the solid curve in Figure 3.6 has larger area under the curve than the dotted one, we would prefer the method corresponding to the dotted curve since it has a larger AP ($\text{AP}_d = 0.1998, \text{AP}_s = 0.1538$).

To calculate AP with ties, instead of averaging over all possible permutations, we consider 500 random permutations of the $\hat{p}$'s and take the average of those 500 APs.

	Model 1		Model 2	
Ranking	True Class Label	Precision	True Class Label	Precision
1	1	1/1	1	1/1
2	1	2/2	0	1/2
3	0	2/3	0	1/3
4	1	3/4	1	2/4
5	0	3/5	1	3/5
6	0	3/6	0	3/6
	$AP_1 \approx 0.9167$		$AP_2 = 0.7$	

Table 3.2: Illustration of AP calculation: $AP_1 = \frac{1}{3}(1/1 + 2/2 + 3/4) \approx 0.9167$; $AP_2 = \frac{1}{3}(1/1 + 2/4 + 3/5) = 0.7$.

3.3.4 ROC Curve

The Receiver Operating Characteristic (ROC) curve is widely used to evaluate the performance of different ranking systems (Cantor and Kattan, 2000). Suppose that the observations are ranked by their scores of being class 1 $(s_1, s_2, \cdots, s_n)$ with $1 \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$. Let the random variable S_D be the score of a detected item whose probability density function (pdf) and cumulative distribution function (CDF) are f_D and F_D. We further assume that the corresponding pdfs and the CDFs of the class-0 and class-1 observations are f_0, f_1 and F_0, F_1, respectively.

Given a cut-off C, an observation is assigned to class 1 if its score S_D is greater than C; otherwise, it is assumed to be class 0. The ROC curve plots hit rate versus false alarm rate evaluating at each potential cut-off C_i, where

$$\text{Hit Rate} = \Pr(\text{true positive}) = \Pr(S_D > C | S_D \sim f_1) = 1 - F_1(C)$$
$$\text{False Alarm Rate} = \Pr(\text{false positive}) = \Pr(S_D > C | S_D \sim f_0) = 1 - F_0(C).$$

The area under the ROC curve (AUC) is one of the most popular measures to compare the performance of different ranking models. A better ranking model has a larger value of AUC.

Figure 3.7 presents several typical ROC curves. The curve passing the points (0, 0), (0, 1) and (1, 1) represents a perfect model; the dashed curve and the solid curve represent two ranking models; and the dotted line corresponds to the random selection. The AUC of a perfect model is 1 and the AUC of a random ranking is 1/2, which implies that any effective ranking model should have AUC $\in (1/2, 1]$. Figure 3.7 indicates that ranking model 1 is superior to ranking model 2, because it has a larger value of AUC than model 2.

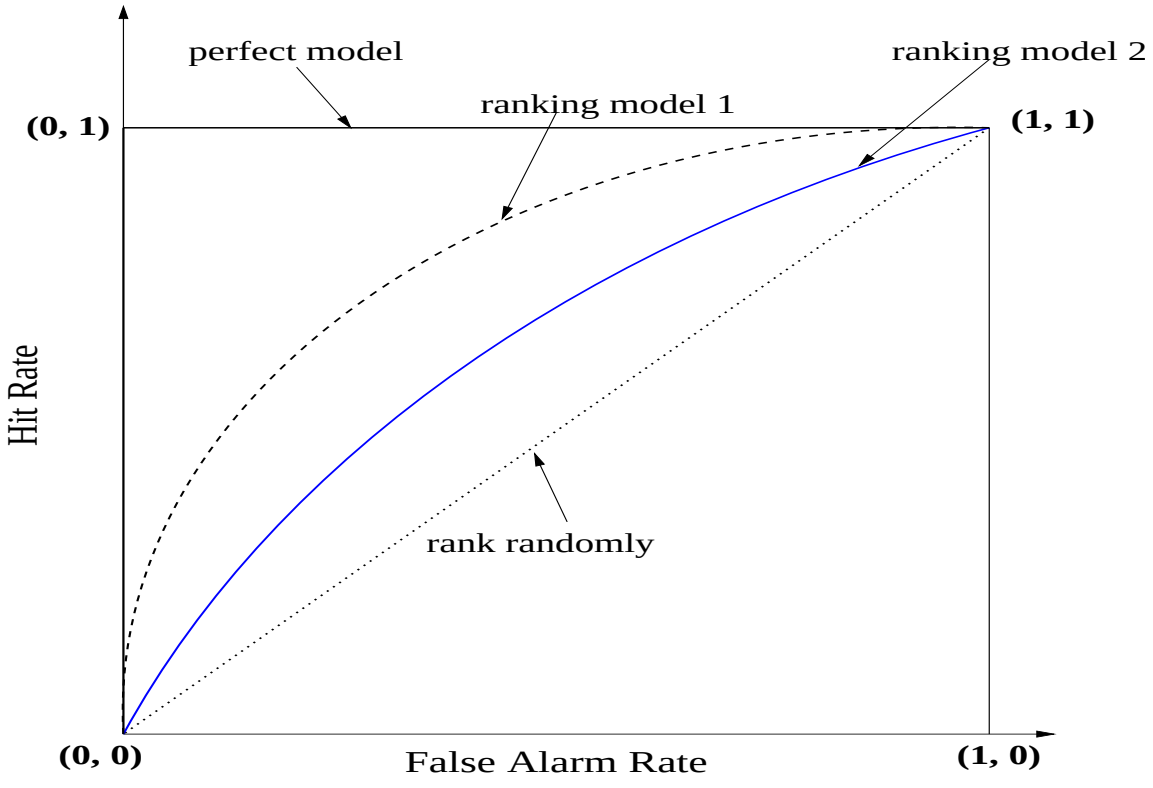

Figure 3.7: Illustration of several typical ROC curves.

3.3.5 Relation Between ROC and Hit Curve

A variation of hit curve is called Cumulative Accuracy Profile (CAP) curve, which is widely used in finance community to evaluate different ranking systems. The hit curve used in my thesis plots number of hits versus the number of detected items, whereas the CAP curve plots the proportion of hits versus the proportion of detected items. Therefore, if all the items rather than only the top n are used, hit curve and CAP curves are equivalent up to normalization. Engelmann *et al.* (2003) explored the connection between ROC curve and CAP curve. In this section, I will give a more clear presentation of the relationship between ROC curve and CAP curve under the context of statistical detection.

Instead of plotting pairs $\{1 - F_0(s_i), 1 - F_1(s_i)\}_{i=1}^n$ in a ROC curve, CAP plots pairs $\{1 - F_D(s_i), 1 - F_1(s_i)\}_{i=1}^n$ (see Figure 3.8). For CAP curves, the one connecting points (0, 0), $(\pi, 1)$ and (1, 1) corresponds to a perfect model, where π is the proportion of class-1 observations; and the curve going through points (0, 0) and (1, 1) corresponds to the random selection. Obviously, the area under the CAP curve of a perfect model is $[(1 - \pi) + 1]/2$, and the area of the random selection is 1/2.

Let AUC_ROC be the area under the ROC curve and AUC_CAP be the area under the CAP curve. Applying the trapezoidal rule, it is straightforward to show that

$$\begin{aligned}
\text{AUC_ROC} &= \sum_{i=1}^{n} \frac{1}{2}[(1 - F_1(s_i)) + (1 - F_1(s_{i+1}))][(1 - F_0(s_{i+1})) - (1 - F_0(s_i))] \\
\text{AUC_CAP} &= \sum_{i=1}^{n} \frac{1}{2}[(1 - F_1(s_i)) + (1 - F_1(s_{i+1}))][(1 - F_D(s_{i+1})) - (1 - F_D(s_i))] \\
&= (1 - \pi)\text{AUC_ROC} + \pi\frac{1}{2}. \qquad (3.9)
\end{aligned}$$

The last step is due to the fact that

$$\begin{aligned}
1 - F_D(s_i) &= \Pr(S_D > s_i) \\
&= \Pr(S_D > s_i | S_D \sim f_1)\Pr(S_D \sim f_1) + \Pr(S_D > s_i | S_D \sim f_0)\Pr(S_D \sim f_0) \\
&= \pi[1 - F_1(s_i)] + (1 - \pi)[1 - F_0(s_i)]. \qquad (3.10)
\end{aligned}$$

Equation (3.9) implies that AUC_CAP(ranking model) can be considered to be a weighted average of AUC_ROC(ranking model) and AUC_ROC(random selection), the weights depend on the proportion of class-1 observations π.

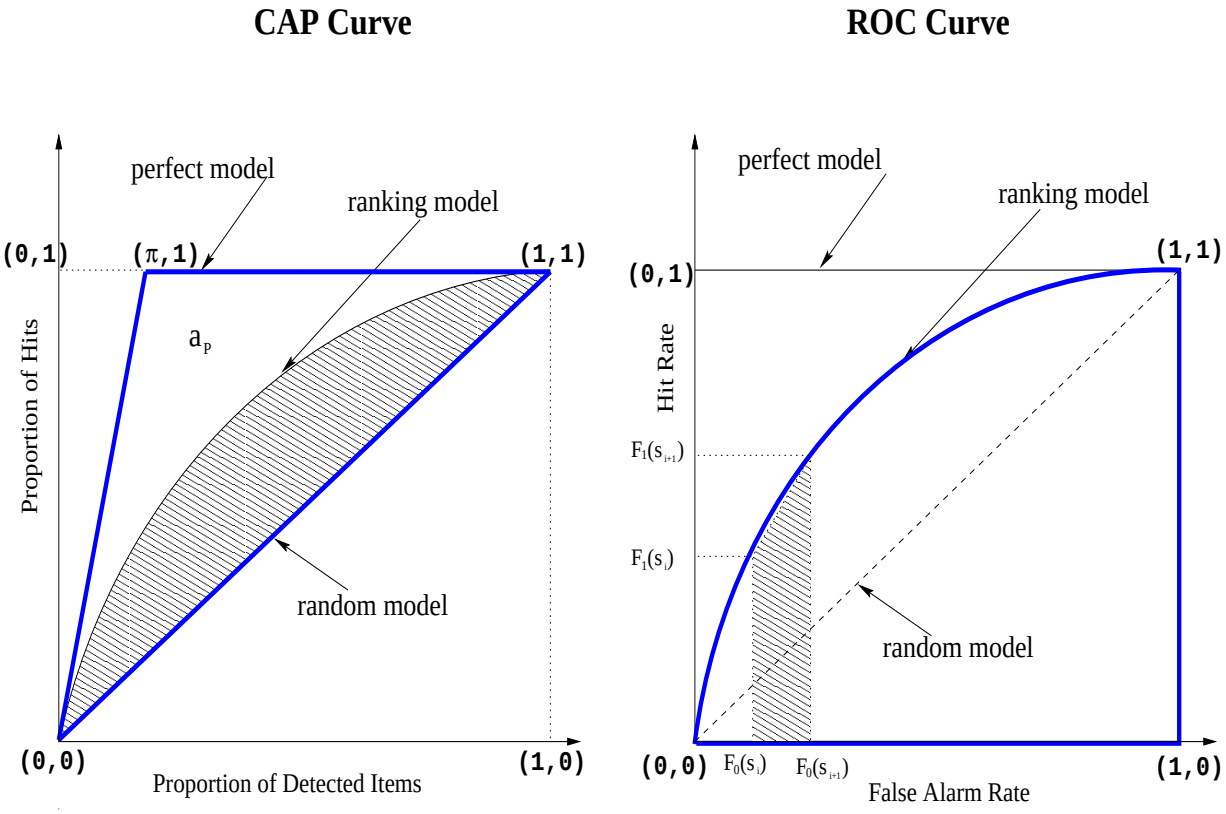

Figure 3.8: Left: A CAP curve where a_R is the shaded area and a_P is marked by the solid line in bold; Right: An ROC curve and AUC_ROC is marked by the solid line in bold.

Besides AUC_CAP, performance of a ranking system can also be summarized by the Accuracy Ratio (AR) which is defined as the ratio of a_R to a_P, where a_R is the discrepancy between AUC_CAP(ranking model) and AUC_CAP(random selection), and a_P is the discrepancy between AUC_CAP(perfect model) and AUC_CAP(random selection). Engelmann *et al.* (2003) showed that there is a linear relationship between AUC_ROC and the

AR of the CAP curve:

$$\begin{aligned} \text{AR} &= \frac{a_R}{a_P} \\ &= \frac{\text{AUC\_CAP(ranking model)} - \frac{1}{2}}{\text{AUC\_CAP(perfect model)} - \frac{1}{2}} \\ &= \frac{(1-\pi)\text{AUC\_ROC} + \pi\frac{1}{2} - \frac{1}{2}}{\frac{(1-\pi)+1}{2} - \frac{1}{2}} \\ &= 2\text{AUC\_ROC} - 1. \end{aligned}$$

3.3.6 Comparison of AP and AUC_ROC

Little research has been done on the properties of average precision (AP). So far, I have never found any theoretical results on AP in literature. In order to have some insight of the relationship between AP and the area under the ROC curve (AUC_ROC), I conducted a permutation study on a small data set with $n_1 = 3$ ones and $n_0 = 50$ zeros, which results in $\binom{53}{3} = 23,426$ distinct permutations of these zeros and ones. Practically, these different permutations can be viewed as different ranking models, and hence AP and AUC_ROC can be calculated for each unique permutation.

Figure 3.9 compares AP and AUC_ROC for each unique permutation and it shows that there is no obvious linear or non-linear relationship between AP and AUC_ROC. For those 23,426 different permutations, one important finding is that there are 18,457 and 151 unique values in AP and AUC_ROC respectively. This somewhat indicates that AP has a higher "resolution" than AUC_ROC in distinguishing different ranking methods. Figure 3.9 also illustrates an important property of AP — it puts much more emphasis on early detection. Only those ranking models whose first detected item is a hit yield large values of AP. When it comes to statistical detection, AP might be a more appropriate measure than AUC_ROC, because we want to identify the class-1 observations as early as possible.

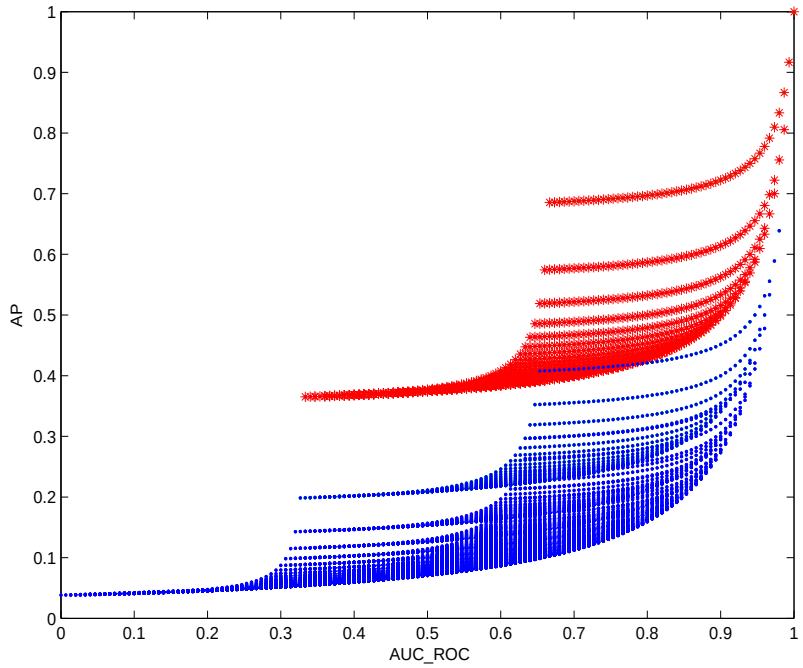

Figure 3.9: Average precision and area under the ROC curve of the permutation study. Stars indicate those permutations starting with 1.

Compared to AUC_ROC, one disadvantage of AP is that it is much more difficult to interpret and make statistical inference on. Hanley and McNeil (1982) gave the meaning of the AUC_ROC in applications of disease diagnostic testing. They pointed out that the AUC_ROC represents the probability that a randomly chosen diseased subject is rated with greater suspicion than a randomly chosen non-diseased subject. Moreover, they showed this probability of correct ranking is the same quantity as the one that is estimated by the well-known nonparametric Wilcoxon statistic. In their following-up paper, Hanley and McNeil (1983) proposed a method of comparing AUC_ROC given by different diagnostic methods.

Interpretation and inference of the evaluation criterion, however, is not of our primary interest; we decide to choose AP as the main performance metric in the whole thesis.

3.4 Conclusion

This chapter introduces the statistical detection problem which includes drug discovery as an special application. Compared to a binary classification problem, a statistical detection problem has two main characteristics. First, the two classes of a statistical detection problem are extremely unbalanced; the majority is the background class and the class of interest is extremely rare. Second, the objective of a statistical detection problem is to rank the items in a descending order by their possibility of being the rare class, and hence the most potential candidates can be picked as early as possible. This is very different from classifying the items into either the rare class or the background class without considering their order.

Methods that are commonly used in classification such as K-nearest neighbors, trees and support vector machines can also be applied to the statistical detection problems. Instead of using them as different classifiers, however, these methods are adopted as various ranking methods in statistical detection. A brief introduction of these three methods are given in this chapter. The reasons that I choose these three methods are two-fold. Firstly, Wang (2005) concluded that K-nearest neighbors and trees are by far the best two methods for the NCI AIDS data among all the methods she tried, i.e., logistic regression, generalized additive model, multivariate adaptive regression spline, neural networks, K-nearest neighbors and trees. Secondly, LAGO, the method which will be introduced in Chapter 4, is closely related to both K-nearest neighbors and support vector machines; therefore, these two methods provide us good benchmarks in evaluating the performance of our proposed method.

Several possible evaluation metrics for statistical detection are discussed in this chapter. Those metrics include misclassification rate; hit curve and its numerical summaries, e.g., number of hits, area under the hit curve and average precision; ROC curve and its numerical summary, i.e., area under the ROC curve. Given the fact that average precision puts more weights on the early hits and thus is more suitable for statistical detection, we choose

average precision as the primary performance measure in this thesis.

Chapter 4

LAGO: A Fast Kernel Method for Statistical Detection

Wang (2005, Chapter 2) compared the performance of some basic models on the NCI data using hit curves. Those methods include logistic regression, generalized additive model (GAM), neural network (NNet), trees, C4.5, and K-nearest neighbors (KNN). She concluded that local methods such as KNN with a relatively small value of K and tree models with many terminal nodes outperform other methods. In this chapter, I introduce a computationally efficient method to handle statistical detection problems, LAGO, which stands for "locally adjusted GO estimator". We apply LAGO to the NCI data and two simulated data sets, the results show that LAGO is competitive with some powerful methods such as KNN and support vector machines (SVM).

For notational convenience, in this chapter we assume that the covariate vectors (or scalars for univariate cases) for the training data are denoted as $\mathbf{x}_i$ (or x_i) with observed responses $y_i \in \{0, 1\}$.

4.1 LAGO: Locally Adjusted GO Estimator

Recall that the objective of statistical detection is to rank the potential candidates by their probability of being class 1. Given a data point $\mathbf{z}$, its posterior probability of being class 1, $p(y=1|\mathbf{z})$, is intuitively a good ranking function. By Bayes' rule, this posterior probability can be expressed as

$$\begin{aligned} p(y=1|\mathbf{z}) &= \frac{p(\mathbf{z}|y=1)p(y=1)}{p(\mathbf{z}|y=1)p(y=1)+p(\mathbf{z}|y=0)p(y=0)} \\ &= \frac{af(\mathbf{z})}{af(\mathbf{z})+1}, \end{aligned} \tag{4.1}$$

where $a=\frac{p(y=1)}{p(y=0)}$ and

$$f(\mathbf{z}) = \frac{p(\mathbf{z}|y=1)}{p(\mathbf{z}|y=0)} \triangleq \frac{p_1(\mathbf{z})}{p_0(\mathbf{z})}. \tag{4.2}$$

Since $a>0$, the posterior probability (4.1) is a monotonically increasing function in f. This means that the ranking by f is equivalent to the ranking by the posterior probability. Knowing f is sufficient for statistical detection. LAGO intends to estimate the density ratio f.

A key feature of the statistical detection problems is that most observations are from class 0 (C_0) and only a few are from class 1 (C_1). This important feature allows us to make the following assumptions:

A1. For practical purposes, the density function p_1 can be assumed to have bounded local support, possibly over a number of disjoint regions, $\mathcal{S}_\gamma \subset \mathbb{R}^d$, $\gamma=1,2,...,\Gamma$, in which case the support of p_1 can be written as

$$\mathcal{S} = \bigcup_{\gamma=1}^{\Gamma} \mathcal{S}_\gamma \subset \mathbb{R}^d.$$

A2. For each class-1 observation $\mathbf{x}_i \in C_1$, there exists at least a certain number of class-0 observations, say m, in a local neighborhood of $\mathbf{x}_i$; moreover, the density function p_0 can be assumed to be relatively flat compared to p_1 in that neighborhood.

Assumptions A1 and A2 imply that, in order to estimate the density ratio f, we can simply estimate p_1 and adjust it locally according to the density p_0 nearby. This two-step strategy would give us a significant computational advantage, considering that the class-1 observations are extremely rare. Sections 4.1.1 and 4.1.2 illustrate how to construct LAGO in two steps. We first consider the univariate case where the predictor $z \in \mathbb{R}$ is a scalar and then generalize this to multivariate case where $\mathbf{z} \in \mathbb{R}^d$ for $d > 1$ in Section 4.1.4.

4.1.1 Step 1: Estimating p_1

The first step of constructing LAGO is to estimate p_1 by an adaptive bandwidth kernel density estimate:

$$\hat{p}_1(z) = \frac{1}{n_1} \sum_{y_i=1} \mathcal{K}(z; x_i, r_i), \tag{4.3}$$

where n_1 is the number of class-1 observations and $\mathcal{K}(z; x_i, r_i)$ denotes a kernel function centered at x_i with bandwidth r_i. The kernel function $\mathcal{K}(z; x_i, r_i)$ is defined to satisfy the following conditions:

- $\mathcal{K}(z; x_i, r_i)$ is symmetric around x_i.
- The bandwidth of $\mathcal{K}(z; x_i, r_i)$ is r_i.
- $\mathcal{K}(z; x_i, r_i)$ integrates to 1 within its support.

For each $x_i \in C_1$, we define bandwidth r_i to be the average distance between x_i and its K-nearest neighbors from C_0, i.e.,

$$r_i = \frac{1}{K} \sum_{w \in N(K, x_i)} |w - x_i|,$$

where $N(K, x_i)$ refers to the set that contains the K-nearest class-0 neighbors of x_i and K is a tuning parameter that can be selected by cross-validation. This particular bandwidth selection method is originally inspired by an ancient game now known as GO (see Figure

4.1). In this game, two players take turns placing white or black small stones on a 19×19 square board and try to possess as many territories as possible. At any stage during the game, a player should evaluate the strategic values of several possible moves according to the relative strength of his or her position on the board. Ignoring all the rules of the game, which are irrelevant for the key idea, the basic principles of the evaluation can be summarized as follows: Each stone on the board exerts a certain amount of influence on its immediate neighborhood; the amount of influence is inversely related to how close the opponent's stones surrounding it.

As for kernel density estimation, the scenario is similar: Each class-1 observation, x_i, exerts a certain amount of influence over its immediate neighborhood; the extent of this influence is controlled by the bandwidth parameter. Following the basic evaluation rules from the game of GO, we allow each $x_i \in C_1$ to have a different bandwidth r_i depending on the distance between x_i and its neighboring observations from C_0.

4.1.2 Step 2: Locally Adjusting p_1

The next step is to adjust the density estimate $\hat{p}_1$ locally according to the density of class 0 nearby. We can treat (4.3) as a mixture and adjust each component (centered at x_i) accordingly. Based on (4.2), density p_0 around every $x_i \in C_1$, denoted by $p_0(z; x_i)$, should be estimated and divided by the term $\mathcal{K}(z; x_i, r_i)$. For a detection problem, there are a lot of class-0 observations within a neighborhood of each class-1 observation; therefore, it is reasonable to assume that the density $p_0(z; x_i)$ is relatively flat compared to the density $p_1(z; x_i)$ within a neighborhood of x_i. This implies that we can simply estimate $p_0(z; x_i)$ within that neighborhood as a constant, say c_i. As a result, the estimate of the density ratio f can be written as

$$\hat{f}(z) = \frac{1}{n_1} \sum_{y_i=1} \frac{\mathcal{K}(z; x_i, r_i)}{c_i}, \tag{4.4}$$

for appropriately estimated constants c_i, $i = 1, 2, \ldots, n_1$. Theorem 1 shows the relationship between c_i and r_i, which makes the estimation of c_i unnecessary in practice. The detailed

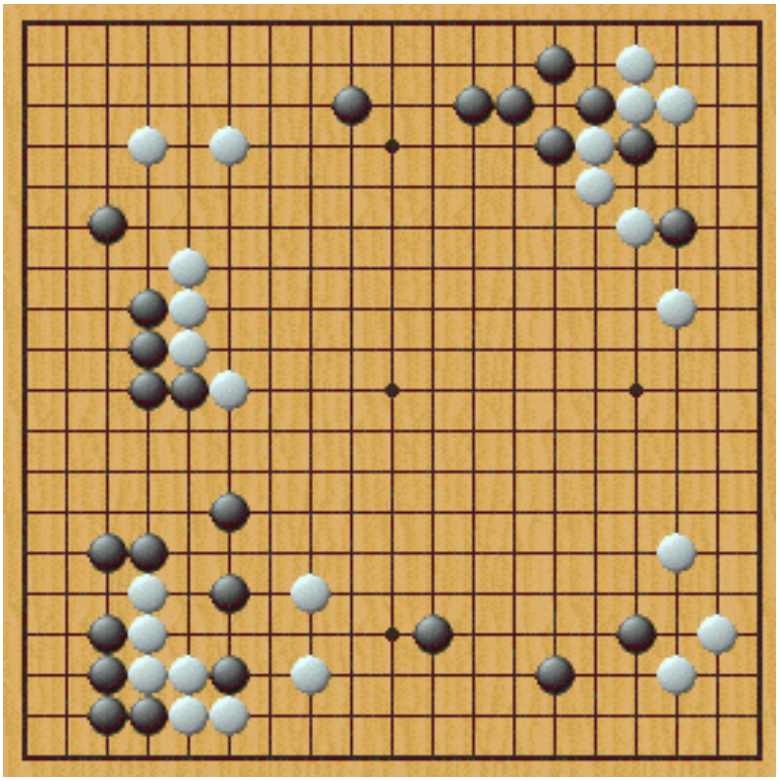

Figure 4.1: The ancient game of Go is a game in which each player tries to claim as many territories as possible on the board. Image taken from http://go.arad.ro/Introducere.html.

proof of Theorem 1 is given in Appendix. The idea of the proof consists of calculating the density function of an order statistic.

Theorem 1. *Let x_0 be a fixed observation from class 1. Suppose that $w_1, w_2, \ldots, w_m$ are iid observations from class 0 that are uniformly distributed around x_0, say on the interval $[x_0 - \frac{1}{2c_0}, x_0 + \frac{1}{2c_0}]$. If r_0 is the average distance between x_0 and its K nearest neighbors from class 0 ($K < m$), then we have*

$$E(r_0) = \frac{K+1}{4(m+1)c_0}. \tag{4.5}$$

Theorem 1 implies that the bandwidth r_i is on average proportional to $1/c_i$, since the

constants K and m do not depend on x_i. Given that r_i has already been calculated in step 1, it is not necessary to estimate c_i separately for each x_i. This provides us an extra computational advantage. Replacing $1/c_i$ in (4.4) with r_i gives the LAGO estimate

$$\hat{f}(z) = \frac{1}{n_1} \sum_{y_i=1} r_i \mathcal{K}(z; x_i, r_i). \tag{4.6}$$

We use the acronym LAGO to refer to this resulting estimator $\hat{f}$ as the "locally adjusted GO estimator".

4.1.3 Asymptotic Properties of LAGO

This section investigates the asymptotic properties of LAGO given in (4.6). For simplicity, we only consider the case where $K = 1$, i.e., the bandwidth r_i for the kernel function centered at $x_i \in C_1$ is defined to be the distance between x_i and its nearest class-0 neighbor. Assume that the class-1 observations $x_1, x_2, \ldots, x_{n_1} \overset{iid}{\sim} p_1(\cdot)$, and the class-0 observations $w_1, w_2, \ldots, w_{n_0} \overset{iid}{\sim} p_0(\cdot)$ with $F_0(\cdot)$ and $F_1(\cdot)$ being the cdfs of class-0 and class-1 observations respectively. Let $u_{ij} = |w_j - x_i|$ and hence $r_i = \min\{u_{ij}\}, j = 1, \ldots, n_0$. Then the expectation of the LAGO estimate is given by

$$\begin{aligned} \mathrm{E}[\hat{f}(z)] &= \mathrm{E}[\frac{1}{n_1} \sum_{y_i=1} r_i \mathcal{K}(z; x_i, r_i)] \\ &= \frac{1}{n_1} \sum_{y_i=1} \mathrm{E}[r_i \mathcal{K}(z; x_i, r_i)]. \end{aligned} \tag{4.7}$$

The joint distribution of $x_i \in C_1$ and r_i, $\psi(x_i, r_i) = \psi(r_i|x_i)p_1(x_i)$, is needed to evaluate the expectations in (4.7). The conditional cdf of r_i given $x_i \in C_1$ is

$$\begin{aligned} \Psi(r|x_i) &= P(r_i \le r|x_i) \\ &= 1 - \prod_{w_j \in C_0} P(|w_j - x_i| \ge r) \\ &= 1 - \prod_{w_j \in C_0} [1 - P(x_i - r < w_j < x_i + r)] \\ &= 1 - [1 - F_0(x_i + r) + F_0(x_i - r)]^{n_0}. \end{aligned} \tag{4.8}$$

Taking the derivative of (4.8) with respect to r gives the conditional density

$$\psi(r|x_i) = n_0(1 - F_0(x_i + r) + F_0(x_i - r))^{n_0-1}[p_0(x_i + r) + p_0(x_i - r)]. \tag{4.9}$$

Assume that a uniform kernel is used, that is,

$$\mathcal{K}(z; x_i, r_i) = \begin{cases} \frac{1}{2r_i} & \text{if } |z - x_i| \leq r_i; \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

For simplicity in the notations and without ambiguity, I will omit the index i for the proof hereafter. Combining equations (4.7), (4.9), and (4.10) gives

$$\begin{aligned} & \mathrm{E}[r\mathcal{K}(z; x, r)] \\ = & \int\int r\mathcal{K}(z; x, r)\psi(r|x)p_1(x)drdx \\ = & \int\int_{|z-x|\leq r} \frac{1}{2}\psi(r|x)p_1(x)drdx \\ = & \frac{1}{2}\int_{-\infty}^{z}\int_{z-x}^{\infty}\psi(r|x)p_1(x)drdx + \frac{1}{2}\int_{z}^{\infty}\int_{x-z}^{\infty}\psi(r|x)p_1(x)drdx. \end{aligned} \tag{4.11}$$

The integration region is the upper area bounded by the bold line in Figure 4.2.

Given the fact that

$$\begin{aligned} \Psi(\infty|x) &= 1; \\ \Psi(z - x|x) &= 1 - [1 - F_0(z) + F_0(2x - z)]^{n_0}. \end{aligned}$$

The first integral in (4.11) becomes

$$\begin{aligned} & \int_{-\infty}^{z}\int_{z-x}^{\infty}\psi(r|x)p_1(x)drdx \\ = & \int_{-\infty}^{z}[\Psi(r|x)|_{r=z-x}^{\infty}]p_1(x)dx \\ = & \int_{-\infty}^{z}[1 - F_0(z) + F_0(2x - z)]^{n_0}p_1(x)dx \\ = & \int_{-\infty}^{z}[1 - F_0(z) + F_0(z) + p_0(z)(2x - 2z) + \frac{p_0'(\xi)(2x - 2z)^2}{2!}]^{n_0}p_1(x)dx, \end{aligned} \tag{4.12}$$

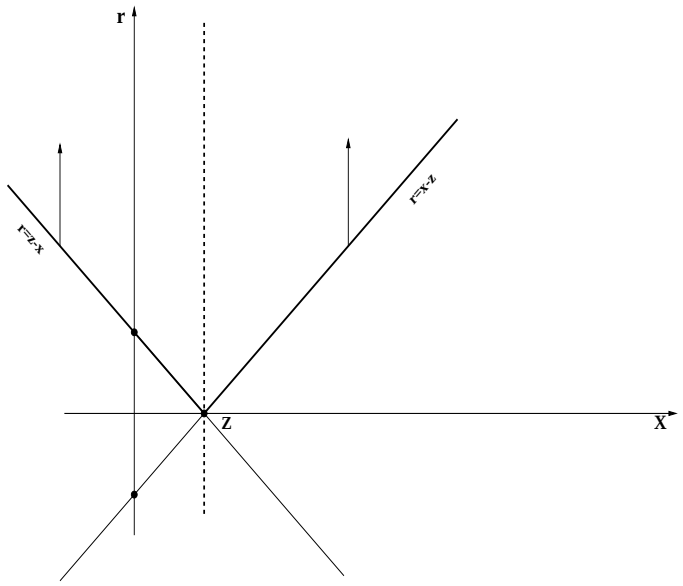

Figure 4.2: Region yields non-zero integrand for the integration in equation (4.11).

where ξ falls somewhere between $2x - z$ and z. The last step is due to the application of mean value theorem to the term $F_0(2x - z)$. Under the assumption that $p_0(z)$ is roughly a constant within the local neighborhood $|z - x| \leq r$, $p_0(z)$ should be bounded and $p_0^{'}(\xi)$ should be relatively small. Let $|z-x| \leq r \to 0$, the term $p_0^{'}(\xi)(2x-2z)^2$ should be a higher order term of $p_0(z)(2x - 2z)$ in r, and hence can be ignored. Because of the fact that

$$\lim_{n\to\infty}(1 + \frac{a}{n})^n = e^a,$$

the right hand side of equation (4.12) can be expressed as

$$\begin{aligned}
& \int_{-\infty}^{z}[1 - F_0(z) + F_0(z) + p_0(z)(2x - 2z) + \frac{p_0^{'}(\xi)(2x - 2z)^2}{2!}]^{n_0}p_1(x)dx \\
\approx & \int_{-\infty}^{z}[1 + 2p_0(z)(x - z)]^{\frac{1}{2p_0(z)(x-z)}2p_0(z)(x-z)n_0}p_1(x)dx \\
\to & \frac{1}{e^{2n_0p_0(z)z}}\int_{-\infty}^{z}e^{2n_0p_0(z)x}p_1(x)dx. \qquad (4.13)
\end{aligned}$$

Using integration by parts, equation (4.13) becomes

$$\begin{aligned}
& \frac{1}{e^{2n_0p_0(z)z}}\int_{-\infty}^{z} e^{2n_0p_0(z)x}p_1(x)dx \\
= & \frac{1}{e^{2n_0p_0(z)z}}\left[\frac{e^{2n_0p_0(z)x}}{2n_0p_0(z)}p_1(x)|_{-\infty}^{z} - \int_{-\infty}^{z}\frac{e^{2n_0p_0(z)x}}{2n_0p_0(z)}p_1^{'}(x)dx\right] \\
= & \frac{1}{e^{2n_0p_0(z)z}}\left[\frac{e^{2n_0p_0(z)z}}{2n_0p_0(z)}p_1(z) - \frac{e^{2n_0p_0(z)x}}{(2n_0p_0(z))^2}p_1^{'}(x)|_{-\infty}^{z} + \int_{-\infty}^{z}\frac{e^{2n_0p_0(z)x}}{(2n_0p_0(z))^2}p_1^{''}(x)dx\right] \\
= & \frac{p_1(z)}{2n_0p_0(z)} - \frac{p_1^{'}(z)}{(2n_0p_0(z))^2} + \frac{1}{e^{2n_0p_0(z)z}}\int_{-\infty}^{z}\frac{e^{2n_0p_0(z)x}}{(2n_0p_0(z))^2}p_1^{''}(x)dx. \qquad (4.14)
\end{aligned}$$

Similarly,

$$\begin{aligned}
& \int_{z}^{\infty}\int_{x-z}^{\infty}\psi(r|x)p_1(x)drdx \\
\rightarrow & \frac{p_1(z)}{2n_0p_0(z)} + \frac{p_1^{'}(z)}{(2n_0p_0(z))^2} - \frac{1}{e^{2n_0p_0(z)z}}\int_{-\infty}^{z}\frac{e^{2n_0p_0(z)x}}{(2n_0p_0(z))^2}p_1^{''}(x)dx. \qquad (4.15)
\end{aligned}$$

Taking the sum of (4.14) and (4.15) gives

$$\mathrm{E}[r_i\mathcal{K}(z;x_i,r_i)] \rightarrow \frac{p_1(z)}{2n_0p_0(z)} = \frac{1}{2n_0}f(z).$$

□

Therefore, $\mathrm{E}(\hat{f}(z)) \rightarrow \frac{1}{2n_0}f(z)$, where n_0 is the number of class-0 observations in the training data. This implies that if $K = 1$, uniform kernel is used, and p_0 is roughly a constant within a small neighborhood of $|x - z| \leq r$, the LAGO estimate is asymptotically unbiased up to a constant.

I conduct a univariate simulation to verify the result. Consider the following distribution:

$$x|C_1 \sim \begin{cases}
\mathrm{UNIF}[-8.5,-7.5] & \text{with probability } \frac{1}{4}, \\
\mathrm{UNIF}[1.5,2.5] & \text{with probability } \frac{1}{4}, \\
\mathrm{UNIF}[-10,-8.5) & \text{with probability } \frac{1}{8}, \\
\mathrm{UNIF}(-7.5,-6] & \text{with probability } \frac{1}{8}, \\
\mathrm{UNIF}(-6,1.5) & \text{with probability } \frac{1}{8}, \\
\mathrm{UNIF}(2.5,10] & \text{with probability } \frac{1}{8}.
\end{cases}$$

$$x|C_0 \sim \begin{cases} \text{UNIF}[-10,-6] & \text{with probability } \frac{1}{2}, \\ \text{UNIF}(-6,10] & \text{with probability } \frac{1}{2}. \end{cases}$$

That means the density of class 1 is

$$p_1(x) = \begin{cases} \frac{1}{4} & \text{when } x \in [-8.5,-7.5] \cup [1.5,2.5], \\ \frac{1}{12} & \text{when } x \in [-10,-8.5) \cup (-7.5,-6], \\ \frac{1}{60} & \text{when } x \in (-6,1.5) \cup (2.5,10]; \end{cases}$$

and the density of class 0 is

$$p_0(x) = \begin{cases} \frac{1}{8} & \text{when } x \in [-10,-6], \\ \frac{1}{32} & \text{when } x \in (-6,10]. \end{cases}$$

And hence the density ratio $f(x) = \frac{p_1(x)}{p_0(x)}$ is given by

$$f(x) = \begin{cases} \frac{1/4}{1/8} = 2 & \text{when } x \in [-8.5,-7.5], \\ \frac{1/12}{1/8} = \frac{2}{3} & \text{when } x \in [-10,-8.5) \cup (-7.5,-6], \\ \frac{1/4}{1/32} = 8 & \text{when } x \in [1.5,2.5], \\ \frac{1/60}{1/32} = \frac{8}{15} & \text{when } x \in (-6,1.5) \cup (2.5,10]. \end{cases}$$

Let n_0 and n_1 be the number of class-0 and class-1 observations. The simulation procedure is as follows:

1. Set the input of the test data to be 81 points over the interval [-10, 10], spaced equally apart. That is $-10, -9.75, -9.5, \ldots, 9.5, 9.75, 10$.

2. Simulate the training data in the following way: Generate $\frac{1}{2}n_0$ data points uniformly from interval [-10, -6] and another $\frac{1}{2}n_0$ from (-6, 10]; generate $\frac{1}{4}n_1 = 60$ data points uniformly from [-8.5, -7.5] and another 60 from [1.5, 2.5]; generate $\frac{1}{8}n_1 = 30$ observations uniformly from each of the four intervals: [-10, -8.5), (-7.5, -6], (-6, 1.5), and (2.5, 10].

3. Fit LAGO on the training data with the uniform kernel and parameter $K = 1$. Predict the LAGO scores for the test data generated in step 1.

4. Repeat steps 2 and 3 10,000 times. For each test point z, compare $2n_0\overline{\hat{f}(z)}$ with its true density ratio $f(z)$, where $\overline{\hat{f}(z)}$ is the average of the LAGO score $\hat{f}(z)$ over the 10,000 experiments.

Figure 4.3 compares $2n_0\overline{\hat{f}(z)}$ with its true density ratio $f(z)$ for $z = -10, -9.75, \ldots, 9.75, 10$. Except for the boundary points $z = -10, -8.5, -7.5, 1.5, 2.5, 10$, the rescaled LAGO score $2n_0\overline{\hat{f}(z)}$ is reasonably close to the true density ratio $f(z)$. The downward boundary effect is a common issue for the kernel density estimator.

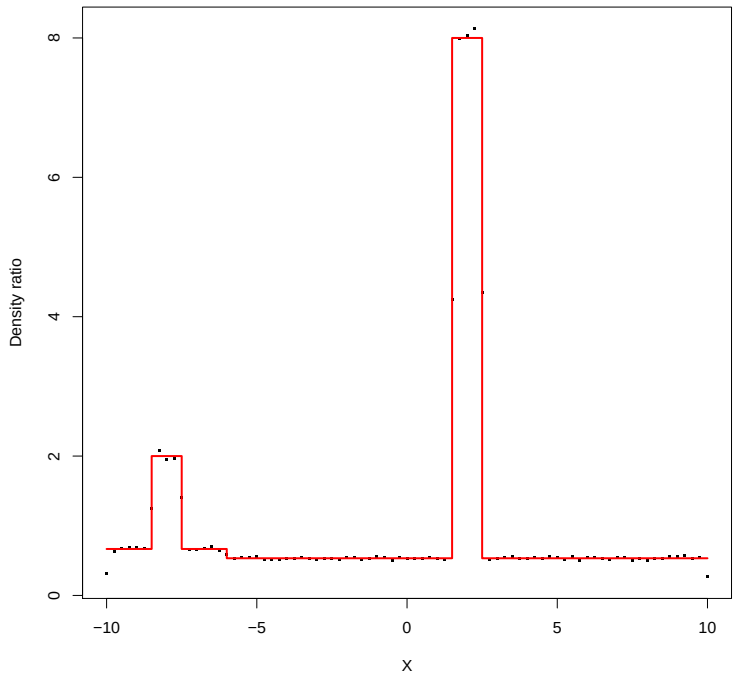

Figure 4.3: A comparison of the average LAGO scores over 10,000 experiments with the the true density ratios (the line) for each test point. The LAGO scores are multiplied by $2n_0$, the scaling factor derived in the previous section. Here, $n_0 = 960$, and $n1 = 240$.

4.1.4 Generalization to Multivariate Case

The LAGO estimate (4.6) can be easily generalized to multivariate cases where predictors $\mathbf{z} \in \mathbb{R}^d$ for $d > 1$ by applying the Naive Bayes principle. That is we model each dimension independently and multiply the marginal models in each dimension together as the final estimate. The Naive Bayes principle might be inappropriate if there is evidence that the predictors are correlated. In those cases, one can transform the original data using a method such as principle component analysis and then apply LAGO in the transformed space. The algorithm is summarized as follows:

1. For each class-1 observation in the training set, $\mathbf{x}_i \in C_1$, find its K-nearest class-0 neighbors $N(K, \mathbf{x}_i)$ in $\mathbb{R}^d$ and compute the bandwidth vector $\mathbf{r}_i = (r_{i1}, r_{i2}, \ldots, r_{id})^T$, where r_{ij} is the average distance between $\mathbf{x}_i$ and its K nearest class-0 neighbors in the jth dimension, that is $r_{ij} = \frac{1}{K} \sum_{\mathbf{w} \in N(K, \mathbf{x}_i)} |w_j - x_{ij}|$.

2. For every new observation $\mathbf{z} = (z_1, z_2, \ldots, z_d)^T$, its LAGO score is given by

$$\hat{f}(\mathbf{z}) = \frac{1}{n_1} \sum_{y_i = 1} \{ \prod_{j=1}^{d} r_{ij} \mathcal{K}(z_j; x_{ij}, \alpha r_{ij}) \}. \tag{4.16}$$

Notice that we have introduced an extra global tuning parameter $\alpha > 0$ in (4.16). The initial reason is that the performance measures are very insensitive to K. Introducing α not only makes the overall tuning more effective but also improves the performance of LAGO. Clearly, setting $\alpha = 1$ is the same as not introducing this extra tuning parameter. By (4.6), with the extra parameter α the LAGO score $\hat{f}(\mathbf{z})$ given in (4.16) should be multiplied by a factor of α^d; however, this factor is not included because all monotonic transformations of $\hat{f}(\mathbf{z})$ are equivalent in terms of ranking. In general, increasing K and increasing α would both increase the bandwidth r_{ij}. However, there is a fundamental difference between the effects of α and of K on r_{ij}. The effect of K depends on the density of the data and hence is not identical in every dimension. Whereas, the parameter α stretches ($\alpha > 1$) or shrinks ($\alpha < 1$) the bandwidths identically in every dimension. Moreover, our experience indicates

that the effect of α on the bandwidths are much stronger than that of K; stretching the bandwidths by a factor of 2 might be equivalent to increasing K by a hundred. In practice, it won't hurt much if we fix K at a reasonable value and only tune α carefully.

The three kernel functions considered in this thesis are the uniform kernel, the triangular kernel and the Gaussian kernel (see Figure 4.4). Our experience has indicated that if a uniform kernel is used, a lot of observations receive the same LAGO score. These ties make the ranking ineffective. Significant improvements can be obtained by choosing $\mathcal{K}$ to be triangular or Gaussian.

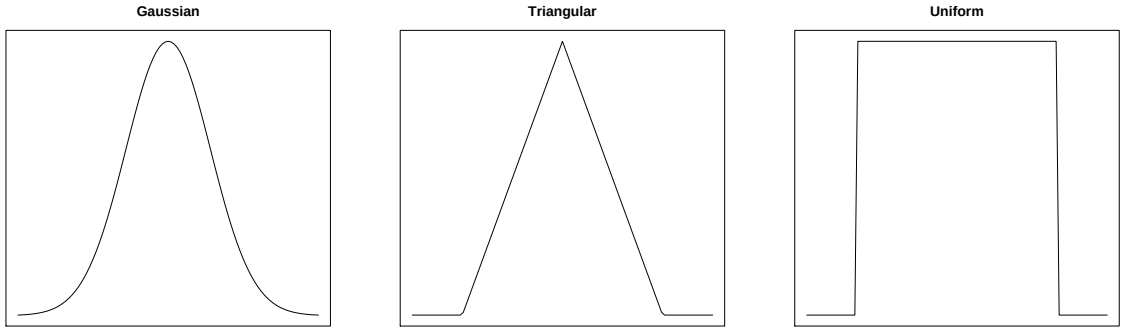

Figure 4.4: The three kernel functions used for LAGO. Left: Gaussian, $\mathcal{K}(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$. Middle: Triangular, $\mathcal{K}(u) = \frac{1-|u|}{2}, |u| \leq 1$. Right: Uniform, $\mathcal{K}(u) = \frac{1}{2}, |u| \leq 1$.

4.1.5 An Illustrative Example

In this section, I illustrate how to construct the LAGO score for a univariate case by a toy example (see Figure 4.5). Suppose x_1 and x_2 are two class-1 observations, whereas x_3, x_4, and x_5 are their class-0 neighbors. Assume that $K = 2$, for $x_1 \in C_1$, its 2 nearest neighbors are x_3 and x_4, therefore r_1 is given by the average distance from x_1 to x_3 and x_1 to x_4. That is,

$$r_1 = \frac{|x_3 - x_1| + |x_4 - x_1|}{2}.$$

Similarly,

$$r_2 = \frac{|x_4 - x_2| + |x_5 - x_2|}{2}.$$

The LAGO score contributed by x_1 is shown by the Gaussian kernel which centers at x_1 and has bandwidth αr_1. That is

$$r_1 \phi_{\alpha r_1}(z - x_1) = r_1 \frac{1}{\sqrt{2\pi}\alpha r_1} \exp\{-\frac{(z - x_1)^2}{2\alpha^2 r_1^2}\} \propto \exp\{-\frac{(z - x_1)^2}{2\alpha^2 r_1^2}\}.$$

Summing up the scores over all the class-1 observations, i.e., x_1 and x_2, yields the LAGO score

$$\hat{f}(z) = \exp\{-\frac{(z - x_1)^2}{2\alpha^2 r_1^2}\} + \exp\{-\frac{(z - x_2)^2}{2\alpha^2 r_2^2}\},$$

which is given by the dashed line in Figure 4.5.

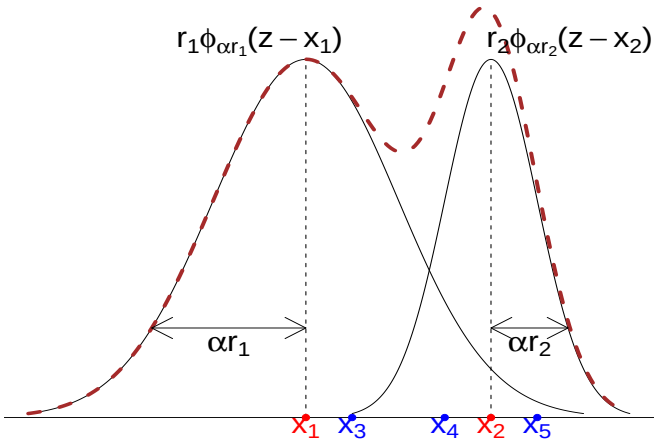

Figure 4.5: The LAGO score (dashed line) contributed by two class-1 observations given $K = 2$.

4.2 Connection to Existing Models

In this section, we show that LAGO can be viewed as a radial basis function (RBF) network and discuss the connection between LAGO and SVM.

4.2.1 RBF Networks

In the field of functional approximation, it is popular to approximate a function $f(\mathbf{x})$ using a number of basis functions, e.g.,

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta_m h_m(\mathbf{x}),$$

where $h_m(\mathbf{x})$ is the mth basis function with β_m being its corresponding coefficient, and M is the total number of basis functions.

Kernel functions are one type of basis function. Treating the kernel function $\mathcal{K}_m$ as a basis function h_m leads to the radial basis function approximation

$$\begin{aligned} f(\mathbf{x}) &= \sum_{m=1}^{M} \beta_m \mathcal{K}_m(\mathbf{x}) \\ &= \sum_{m=1}^{M} \beta_m \mathcal{K}(\mathbf{x}; \boldsymbol{\xi}_m, \boldsymbol{\lambda}_m). \end{aligned} \tag{4.17}$$

Here $\boldsymbol{\xi}_m$ is the center of the mth kernel function and $\boldsymbol{\lambda}_m = (\lambda_{m1}, \ldots, \lambda_{md})^T$ is a vector of the corresponding radii in each dimension. A typical choice of $\mathcal{K}(\cdot)$ may be the product of d standard Gaussian density functions, where d is the dimension of $\mathbf{x}$ and $\boldsymbol{\xi}_m$. The parameters $\{\boldsymbol{\lambda}_m, \boldsymbol{\xi}_m, \beta_m\}$, $m = 1, \ldots, M$ can be chosen by a combination of least squares, cross-validation and some unsupervised methods.

4.2.2 LAGO as an RBF Network

LAGO described in this chapter can be viewed as a highly specialized RBF network. In particular, we choose all and only those class-1 observations in the training data $\mathbf{x}_i \in C_1$ to be the centers $\boldsymbol{\xi}_m$ and use their corresponding bandwidth vector $\mathbf{r}_i = (r_{i1}, r_{i2}, \ldots, r_{id})^T$ as the radius $\boldsymbol{\lambda}_m$ in (4.17). Moreover, the coefficients are specified implicitly as

$$\beta_i = \frac{1}{n_1} \prod_{j=1}^{d} r_{ij}.$$

Since LAGO is a special RBF network, it is possible to re-parameterize LAGO in a more general way. Recall the first step we construct LAGO score – estimating p_1 by an adaptive kernel density estimator. Figure 4.6 illustrates the effect of the density p_0 on the ranking function f. Suppose that the density of class 1, p_1, has two components. If density of class 0, p_0, is flat (constant), the ranking by f is equivalent to the ranking by p_1, since all monotonic transformations of f are equivalent. However, if class 0 has density function $p_0^{'}$ rather than a constant, it will have two main effects on the ranking function f. Let $f = \frac{p_1}{p_0}$ and $f^{'} = \frac{p_1}{p_0^{'}}$. By comparing f and $f^{'}$ in Figure 4.6, it is clear that for each component of f we should stretch its bandwidth and lift its height if $p_0^{'}$ is relatively low nearby; on the other hand, we should shrink its bandwidth and lower its height if $p_0^{'}$ is relatively high nearby. We call the effect on the bandwidth the α-effect and the one on the height the β-effect.

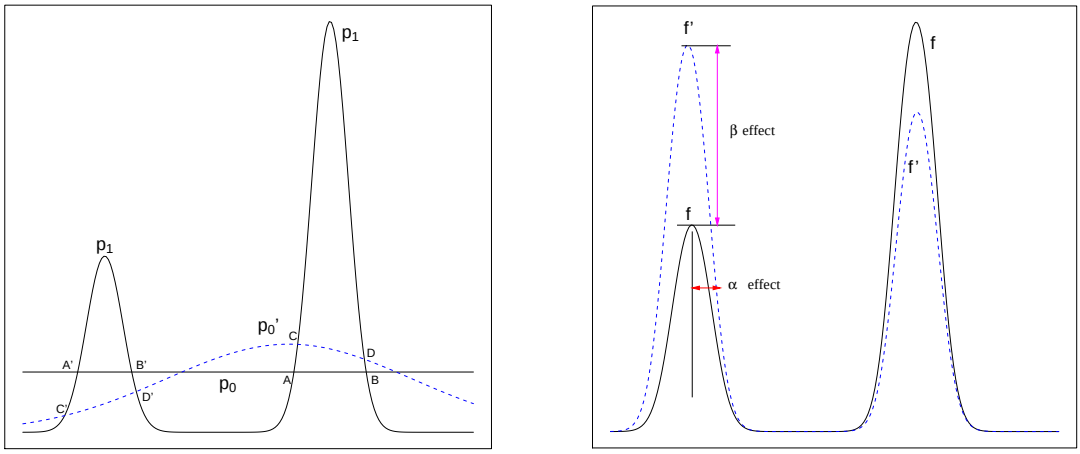

Figure 4.6: Illustration of the ripple effects. Left: The density functions of p_0 and p_1. Right: The density ratio $f = \frac{p_1}{p_0}$.

We can construct LAGO in the form of an RBF network, assuming that each component is a kernel function belonging to a location-scale family, that is,

$$\mathcal{K}(x; x_i, r_i) = \frac{1}{r_i}\mathcal{K}(\frac{x - x_i}{r_i}).$$

Now, we can explicitly parameterize the α-effect and the β-effect as follows

$$r_i^{\beta'} \frac{1}{\alpha r_i}\mathcal{K}(\frac{x - x_i}{\alpha r_i}) \propto r_i^{\beta'-1}\mathcal{K}(\frac{x - x_i}{\alpha r_i}) = r_i^{\beta}\mathcal{K}(\frac{x - x_i}{\alpha r_i}),$$

where α and β are two tuning parameters to be chosen by empirical methods such as cross-validation. Theorem 1 in Section 4.1.2 has argued that the parameter β should be theoretically set to 0 (or $\beta' = 1$). We have run an empirical study on the NCI data to verify the claim that β should be 0 by cross-validation. To save computation, we simply fix $K = 5$ and select α and β simultaneously using 5-fold cross-validation. Figure 4.7 presents the contour plot of average precision based on a 9×9 grid with $\alpha = 0.1, 0.25, 0.5, 1, 1.5, 2, 3, 4, 5$ and $\beta = -1, -0.5, -0.1, -0.01, 0, 0.01, 0.1, 0.5, 1$. It is obvious that the optimal value of β chosen by cross-validation is around 0, which supports the theoretical result.

4.2.3 SVM as an RBF Network

Recall that we use the signed distance (3.5) as the ranking function for SVM in a statistical detection problem. Since all the monotonic transformations of the signed distance are equivalent in ranking, it suffices to use only the inner product term $\hat{\mathbf{w}}^T\mathbf{x}_i$. As a result, the ranking function becomes

$$\hat{f}_{\text{svm}}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \text{SV}} \hat{\alpha}_i y_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i), \tag{4.18}$$

which forms an RBF network with all the support vectors as centers of the kernel functions and $\hat{\alpha} y_i$ as the coefficients. Schölkopf *et al.* (1997) argued that SVMs can be regarded as automatically constructed RBF networks in that the support vectors and $\hat{\alpha}_i$ are automatically determined by the algorithm. Because both SVM and LAGO can be viewed as RBF networks, SVM provides us a good benchmark of the performance of LAGO. In addition,

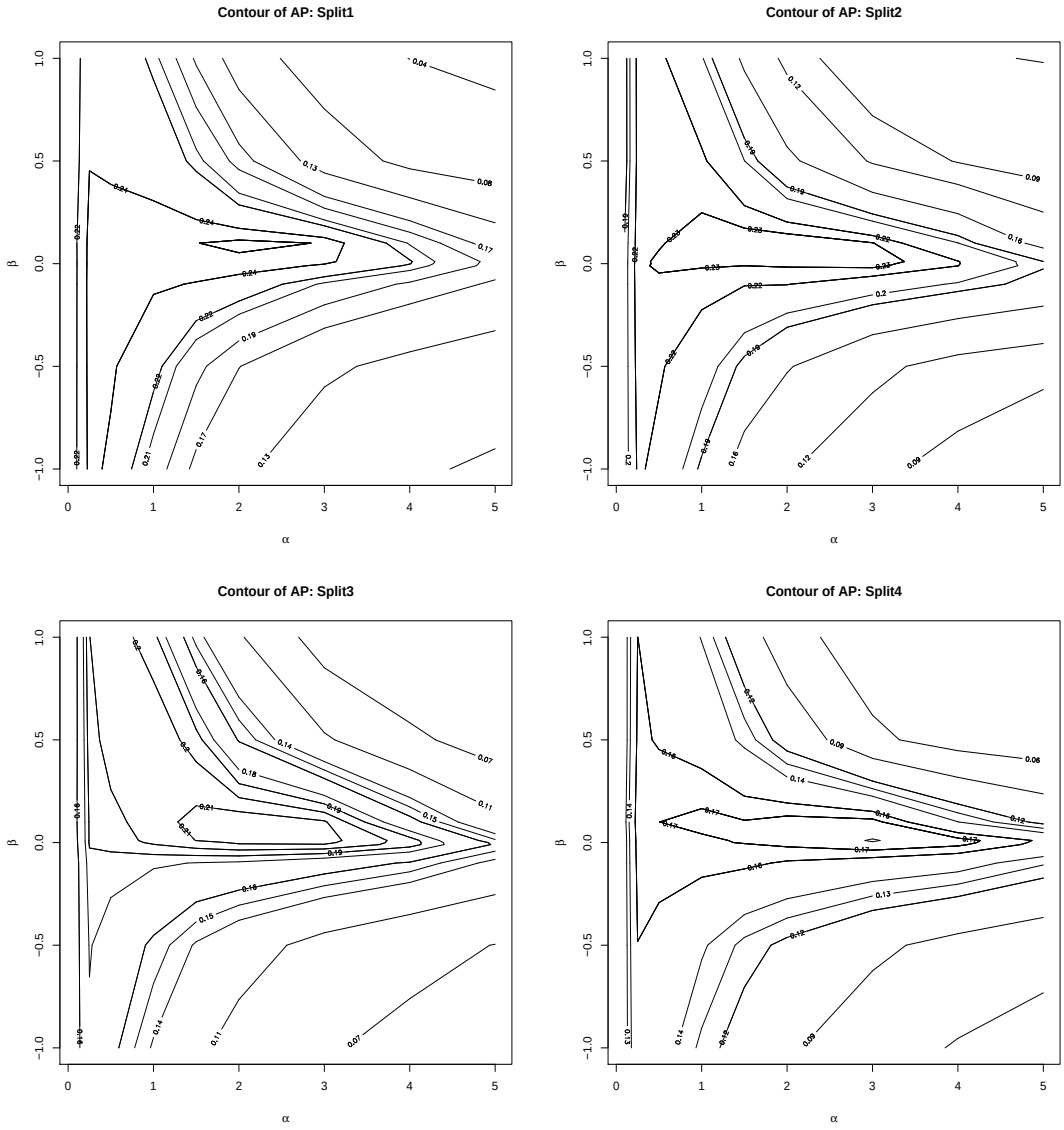

Figure 4.7: Contours of average precision on the four training sets of the NCI data for different values of α, β, and fixed $K = 5$.

it is also instructive to compare the centers SVM picked (i.e., the support vectors) with those we chose in LAGO (all class-1 training observations).

4.3 Application to the NCI AIDS Data

In this section, we apply LAGO to the NCI AIDS data described in Section 2.3.1 and compare its performance with KNN, SVM and ASVM. The reason why we chose SVM as one of the benchmarks has been given in Section 4.2.3. KNN is chosen for two reasons. First, nearest-neighbor method is applied to calculate the bandwidth in LAGO. Second, a previous study (Wang 2005, Chapter 4) based on the original NCI data (with some replicates) concluded that KNN significantly outperforms all other methods at a 5% significance level in terms of average precision. Those methods included trees, MARS, Neural Network, GAM and logistic regression. The comparison was based on four experiments using pairwised t-test. In each experiment, the dataset was randomly divided into the training and test sets with equal sizes in both the active and the inactive compounds. Models were fit on the training sets and performance was assessed on the test sets.

We conduct a similar study on the non-replicated NCI data by evenly splitting the data into training and test sets (see Table 4.1) four times.

Activity class	Data set	Training	Test
0(Inactive)	28,659	14,329	14,330
1(Moderately active)	378	189	189
2(Conformed active)	205	102	103
Total	29,242	14,620	14,622

Table 4.1: Training/test split of the NCI data.

In each experiment, five-fold cross-validation over some pre-determined grid values is

used on the training set to choose the optimal tuning parameters which yield the largest average precision. We randomly divided the training data into five folds in such a way that each fold has roughly the same activity frequencies (see Table 4.2).

	Inactive	Moderately active	Confirmed active
Fold 1	2866	38	20
Fold 2	2866	38	20
Fold 3	2866	38	20
Fold 4	2866	38	20
Fold 5	2865	37	22
Total	14329	189	102

Table 4.2: Randomly divide the training data into five folds.

The five-fold cross-validation procedure to choose the optimal parameters is as follows:

Loop through all possible combinations of the grid values of the parameters

1. Keep fold 1 aside for validation and train the model with the other four folds. Suppose we get model M1. Apply M1 to fold 1 data to obtain a vector of estimated probabilities $\mathbf{p_1}$.

2. Repeat step 1 for all the other folds. Now, we have $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5)$, the estimated probability vector of being active for all the training observations.

3. Sort $\mathbf{p}$ from the largest to the smallest to get $\mathbf{p}' = (p_{(1)}, p_{(2)}, \ldots, p_{(14620)})$ and sort the class label of the training data accordingly to obtain $\mathbf{y}' = (y_{(1)}, y_{(2)}, \ldots, y_{(14620)})$.

4. Calculate the average precision by equation (3.8).

End loop

The optimal combination of parameters is the one that gives the largest cross-validated average precision.

4.3.1 Results

We compare the performances of six models in average precision on four random training/test splits. These six methods are LAGO with a Gaussian kernel (LAGO-G), LAGO with a triangular kernel (LAGO-T), LAGO with a uniform kernel (LAGO-U), KNN, SVM and ASVM. The four training/test splits are referred to as "$\text{split}_1, \ldots, \text{split}_4$". For each split, all tuning parameters are chosen using five-fold cross-validation on the training set and performance measures are compared on the test set. For LAGO, there are two tuning parameters – K (number of class-0 neighbors determining the bandwidth) and the stretching parameter α. The grid values we considered are $K \in \{1, 2, \mathbf{k}\}$ and $\alpha \in \{0.1, 0.25, 0.5, 1, 1.5, 2, 3, 4, 5\}$, where vector $\mathbf{k}$ is given by

$$\mathbf{k} = \{3 \times 5^{\frac{i}{6}}\}_{i=0}^{19}.$$

The reason for this choice of K values is that I want K to increase linearly in log-scale. Note that the effect of K's being increased from 2 to 3 is more influential than K's being increased from 22 to 23 for multivariate applications where the dimension of predictors $d > 1$. Assuming that all the data are contained in a unit hypersphere, to capture a fraction R of the observations, we need a hypersphere whose radius is proportional to $R^{1/d}$. For the NCI data the dimension of the predictors is $d = 6$, that is why I set the grid of K values in the above form.

For KNN, there is only one tuning parameter, K, the number of nearest neighbors. The grid values of K we considered for KNN are the same as those of LAGO. For SVM, there are two tuning parameters: γ (the bandwidth parameter) and C (the penalty parameter); as for ASVM, there are three tuning parameters: C, γ and w_1 (weight parameter for class

1). The performance of SVM and ASVM is very sensitive to the choice of parameters; therefore, it is important to choose the parameters carefully. Hsu, Chang and Liu (2007) provided two main guidelines in tuning the parameters for SVM:

1. It is better to set the grids in exponential scale;
2. Have a coarse search in a wider range first and then shrink the range for a finer search.

Following these two guidelines, I selected the grid values for γ, C, and w_1 as follows. For both SVM and ASVM, the grid values of γ are $\exp\{2, 3, \ldots, 9\} = \{7.39, 20.09, 54.60, 148.41, 403.43, 1096.63, 2980.96, 8103.08\}$. For SVM, the considered values of C are $\exp\{-4.7, -3.6, -2.5, -1.4, -0.3, 0.8, 1.9, 3.0\} = \{0.01, 0.03, 0.08, 0.25, 0.74, 2.23, 6.69, 20.09\}$. As for ASVM, a narrow range of C is searched, i.e., $C \in \exp\{-4.7, -3.62, -2.54, -1.46, -0.38, 0.7\} = \{0.01, 0.03, 0.08, 0.23, 0.68, 2.01\}$; and $w_1 \in \exp\{-3, -2, -1, 0, 1, 2, 3\} = \{0.05, 0.14, 0.37, 1, 2.72, 7.39, 20.09\}$.

The parameters selected by cross-validation for all methods are given in Table 4.3. The resulting average precisions are reported in Table 4.4.

	Uniform		Triangular		Gaussian		KNN	SVM		ASVM		
	K	α	K	α	K	α	K	γ	C	γ	C	w_1
split_1	5	1	5	3	4	0.5	5	1096.63	2.23	148.41	0.68	2.72
split_2	7	1.5	9	3	9	1	5	54.60	2.23	54.60	0.68	2.72
split_3	5	1.5	7	3	7	1.5	5	54.60	0.74	54.60	0.23	2.72
split_4	4	1.5	4	4	2	2	5	20.09	2.23	20.09	0.68	7.39

Table 4.3: Tuning parameters selected for different models using cross-validation

	Uniform	Triangular	Gaussian	KNN	SVM	ASVM
split_1	0.1776	0.2461	0.2292	0.2008	0.1926	0.2091
split_2	0.1942	0.2406	0.2503	0.1864	0.2105	0.2055
split_3	0.1881	0.2562	0.2554	0.1876	0.1688	0.2078
split_4	0.1983	0.2756	0.2713	0.2373	0.2063	0.2600

Table 4.4: Test-set average precisions of different methods for the NCI data.

Figure 4.8 plots the average precisions, and Figure 4.9 shows the hit curves when different methods are applied to the four different test sets. Since after the top 500 detected items, the precisions of all the methods are no better than random selection and the hit curves begin to level out, all the hit curves for the NCI data are plotted up to the first 500 selected compounds.

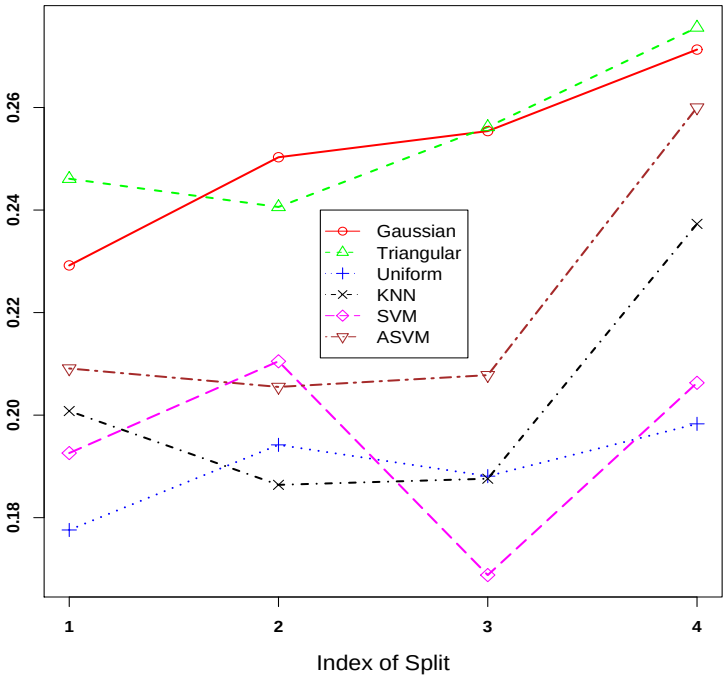

Figure 4.8: The average precisions of different methods evaluated on the test data.

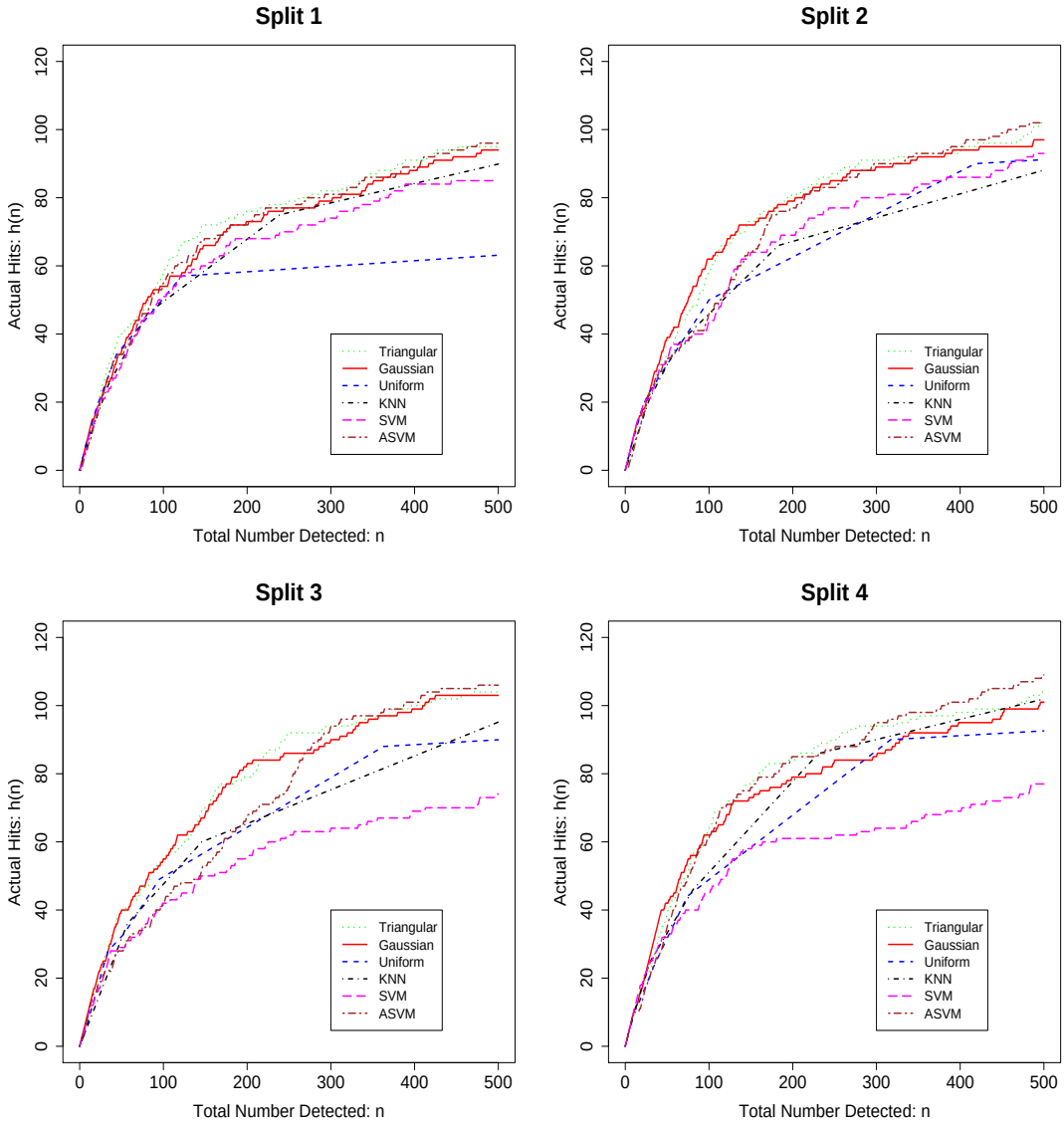

Figure 4.9: Hit curves of different methods for the four random splits of the NCI data. Upper left: The 1st split; Upper right: The 2nd split; Lower left: The 3rd split; Lower right: The 4th split.

Since we measure performance four times with different training and test splits, it is possible to compare those different methods using analysis of variance (ANOVA). Let

$\mu_K, \mu_S, \mu_A, \mu_U, \mu_T$ and μ_G be the average performance (in terms of average precision) of KNN, SVM, ASVM, LAGO-U, LAGO-T, and LAGO-G, respectively. We can further compare the difference among methods by constructing a set of contrasts. The expressions and estimates of the five non-orthogonal contrasts are shown in Table 4.5 and the ANOVA results are given in Table 4.6. The main conclusion is (LAGO-T $\sim$ LAGO-G) $\succ$ ASVM $\succ$ (SVM $\sim$ LAGO-U $\sim$ KNN), where "$\sim$" means "not significantly different from" and "$\succ$" means "significantly better than". This conclusion holds at the 5% significance level.

Contrast	Expression	Estimate
Cntr1	$\mu_T - \mu_G$	0.0031
Cntr2	$\mu_G - \mu_A$	0.0340
Cntr3	$\mu_A - \mu_S$	0.0261
Cntr4	$(\mu_K + \mu_U)/2 - \mu_S$	0.0017
Cntr5	$\mu_K - \mu_U$	0.0134

Table 4.5: Estimated contrasts for the NCI data.

Source	SS($\times 10^{-4}$)	df	MS($\times 10^{-4}$)	F_0	P-Value
Method	162.061	5	32.412	19.021	< 0.0001
Cntr1	*0.189*	*1*	*0.189*	*0.111*	*0.7439*
Cntr2	*23.137*	*1*	*23.137*	*13.578*	*0.0022*
Cntr3	*13.590*	*1*	*13.590*	*7.975*	*0.0128*
Cntr4	*0.081*	*1*	*0.081*	*0.048*	*0.8303*
Cntr5	*3.615*	*1*	*3.615*	*2.122*	*0.1659*
Splits	41.353	3	13.784	8.089	0.0019
Error	25.560	15	1.704		
Total	228.975	23			

Table 4.6: ANOVA analysis of differences among methods for the NCI data.

4.3.2 Comparison to SVM

Although constructed in different ways, both LAGO and SVM can be viewed as RBF networks; therefore, it is instructive to compare the structures of these two RBF networks. Note that LAGO puts basis functions at and only at all of the class-1 observations in the training set, whereas SVM assigns basis functions on all of the support points.

Table 4.7 gives the numbers of training observations from both classes that are chosen as support vectors by SVM and ASVM for the four random splits of the NCI data. Recall that we have 291 active compounds and 14,329 inactive in each training set. Both SVM and ASVM includes almost all the class-1 observations (active compounds) as their support vectors. In addition, SVM and ASVM also select a fairly large number of class-0 observations (inactive compounds) as support vectors. In this sense, LAGO can be viewed as a sparse SVM without the need to search for support vectors. Instead of searching for the support vectors by solving a quadratic programming, LAGO directly selects only the class-1 observations as support vectors. This sparsity not only saves computational cost but also improve the performance.

	SVM		ASVM	
	C_0	C_1	C_0	C_1
Split 1	13724	279	5818	280
Split 2	1280	278	2250	276
Split 3	2202	287	2663	288
Split 4	1037	287	2247	270
Total Possible	14329	291	14329	291

Table 4.7: Numbers of support vectors from classes C_0 and C_1.

4.3.3 Computational Complexity

This section discusses the theoretical computational complexity and compares the empirical computational times of KNN, LAGO-G, SVM using their own compiled R packages on the same machine. The Gaussian kernel is used for SVM for a more fair comparison with LAGO-G.

Given a certain value of the parameter K, there is no computational cost in fitting a KNN. However, once cross-validation is used to choose K, the computational cost is dominated by the calculation of the N by N pairwise distance matrix, where N is the number of observations in the training set. The computational complexity is $O(dN^2)$, where d is the dimension of the input, e.g., $d = 6$ for the NCI data.

As for LAGO-G, given a pair of K and α values, the major cost is calculating the distance matrix whose size is n_1 by n_0, where n_1 and n_0 are respectively the numbers of class-1 and class-0 observations in the training set. Therefore, the computational complexity of fitting a LAGO-G is of order $O(dn_1n_0)$. LAGO-G is faster and cheaper than KNN in calculating and storing the distance matrix.

Given a configuration of the parameters γ and C, the major cost of SVM is in solving a quadratic programming (QP) with linear inequality constraints. General QP solvers require $O(N^3)$ computations in each iteration (Chin 1999, Chapter 3). In practice, the number of iterations required for the solvers to converge varies considerably for different data sets. Direct solution of the QP problem using general QP solvers might be infeasible due to the demanding computation and memory requirements; more practical approaches need to be found. The most popular approach is the decomposition method, which breaks the original QP problem into a series of smaller sub-QP problems; within each sub-problem, only a subset of the unknowns — the so-called *working set*, are updated. Two well-known implementations of SVM are SVM$^{\text{light}}$ (Joachims, 1998) and LIBSVM (Chang and Lin, 2007). Both of them apply the decomposition idea. In LIBSVM, the C++ source codes of the built-in function *svm* in the R package {e1071} (Meyer, 2007), the authors adopt

the sequential minimal optimization (SMO) (Platt, 1999b) approach, which restricts the size of the working set to be 2, the smallest possible value. With some clever strategies, both Joachims (1998) and Chang and Lin (2007) claimed that SVM can be solved with complexity # iterations $\times O(dNq)$, where q is the size of the working set. The number of iterations really depends on the number of support vectors of the resulting SVM; the more support vectors, the more iterations are required for the algorithms to converge.

Given a new data point, in order to calculate its score of being class 1, the computational cost of KNN is in finding its K nearest neighbor, whose complexity is $O(dN)$; the cost of LAGO-G is in evaluating the kernel functions with complexity $O(dn_1)$; the cost of SVM is also in evaluating the kernel functions, whose complexity is $O(dn_{SV})$ with n_{SV} being the number of support vectors of the resulting SVM. In general, $n_1 < n_{SV} < N$ for statistical detection. This means LAGO is faster than KNN and SVM in prediction. Table 4.8 summarizes the computational complexities of KNN, LAGO-G, and SVM. Note that the values in Table 4.8 correspond to the training time with fixed parameters, e.g., a single K value in KNN. If these parameters are chosen via cross-validation or other methods, the models need to be trained many times.

	KNN	LAGO-G	SVM
Train	None	$O(dn_1n_0)$	# iterations $\times O(dNq)$
Predict	$O(dN)$	$O(dn_1)$	$O(dn_{SV})$

Table 4.8: The theoretical computational complexities of KNN, LAGO-G, SVM in training and predicting. N: the total number of observations in the training set; n_0: number of class-0 training observations; n_1: number of class-1 training observations; n_{SV}: number of support vectors; q: the size of the working set in the decomposition method.

Since KNN, LAGO and SVM have been implemented in compiled R packages, it is interesting to compare the practical computation times of these three methods. An empir-

ical study is conducted on the first split of the NCI data. The whole training set consists of 14,620 compounds among which 291 are active. The empirical computational times of KNN, LAGO-G and SVM are compared on a series of nested subsets of the full training set. The subsets are chosen by taking stratified random samples from the training set, which means that all the subsets have the same proportion of active compounds. The next subset is half of the size of the previous one. The parameters of the models are fixed at their corresponding optimal choices on the full training set by cross-validation (see Table 4.3).

Table 4.9 reports the sample size of the training set with number of active compounds in bracket, the execution times of the three methods, and the number of support vectors of the resulting support vector machine from the two classes. The computational times reported in Table 4.9 include the time for model fitting and making prediction for a test set of the same size as the training set. Results of KNN are obtained by using the function *knn* in R library {class}.

Training size	KNN	LAGO-G	SVM	# of SVs	
N (n_1)	$K = 5$	$(K = 4, \alpha = 0.5)$	$(\gamma = 1096, C = 2.23)$	C_0	C_1
14620 (291)	4.874	3.254	284.270	13724	279
7309 (146)	1.220	0.786	66.638	7023	142
3656 (72)	0.326	0.158	10.277	3562	72
1828 (36)	0.080	0.042	1.638	1791	36
914 (18)	0.022	0.016	0.408	895	18
Order	$O(N^2)$	$O(N^2)$	$O(N^{2.6})$		

Table 4.9: The computational times (in second) of KNN, LAGO-G, SVM in model fitting and testing together based on a series of nested subsets of the first split of the NCI data.

For the NCI data, both KNN and LAGO-G are significantly faster than SVM with the

Gaussian kernel, whereas LAGO-G is about 1.5 times as fast as KNN. By fitting a linear regression on log(Time) versus $\log N$, we get the empirical computational complexities of order $O(N^2)$ for both KNN and LAGO-G. The empirical computational complexity for SVM is roughly of order $O(N^{2.6})$. Compared to SVM, LAGO is more efficient in the sense that it does not require any iterative optimization whereas SVM does.

4.4 Application to Simulated Data

In this section, we apply LAGO to two simulated data sets. The first one is the Mysim data that are generated from a mechanism described in Section 2.3.2; and the second one is the Mysim-LAGO data that are generated from a mechanism similar to LAGO. Performance of LAGO on these two simulated data sets is compared with KNN, SVM, and ASVM.

4.4.1 The Mysim Data

LAGO-G, LAGO-T, LAGO-U, KNN, SVM and ASVM are compared in terms of average precision on 100 experiments. In each experiment, we generate a set of training data, a set of validation data, and a set of test data from the same mechanism: generate 300 class-0 observations from $[-3, 3] \times [-1, 5]$; generate 40 class-1 observations from region A, and another 60 class-1 observations from region B (see Figure 2.2). For all the methods, tuning parameters are chosen on the validation data and performance is compared on the test sets. For LAGO, the two tuning parameters are K and α which take values from $K \in \{2, 3, 4, 5, 7, 9, 11, 15, 20, 26, 34, 44, 57, 75, 98, 128\}$ and $\alpha \in \{0.1, 0.25, 0.5, 1, 1.5, 2, 3, 4, 5\}$. KNN has only one tuning parameter K which takes odd numbers from 3 to 65. For SVM and ASVM, the grid values for the tuning parameters are the same as those given in Section 4.3.1.

Figure 4.10 presents boxplots of the 100 experimental average precisions for all the methods compared. Table 4.10 shows the expressions of six non-orthogonal contrasts and

their corresponding estimated values.

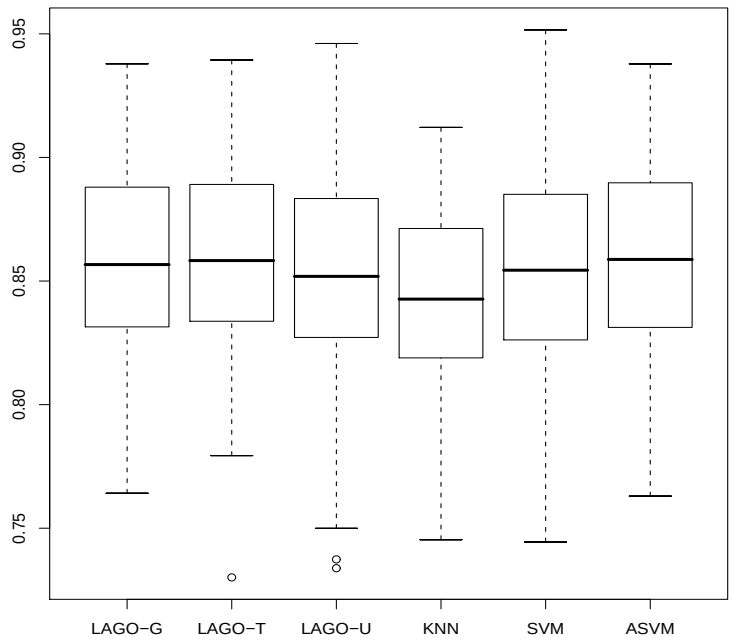

Figure 4.10: The test-set average precisions of different methods evaluated on 100 experiments of the Mysim data.

Contrast	Expression	Estimate
Cntr1	$(\mu_G + \mu_T)/2 - \mu_U$	0.0062
Cntr2	$\mu_G - \mu_T$	0.0005
Cntr3	$\mu_S - \mu_A$	0.0024
Cntr4	$\mu_U - \mu_K$	0.0096
Cntr5	$(\mu_G + \mu_T)/2 - (\mu_S + \mu_A)/2$	0.0030
Cntr6	$(\mu_G + \mu_T + \mu_S + \mu_A)/4 - \mu_K$	0.0143

Table 4.10: Estimated contrasts for the Mysim data.

The ANOVA results of comparing differences among methods are given in Table 4.11. The conclusion of the comparison is (ASVM $\sim$ SVM $\sim$ LAGO-T $\sim$ LAGO-G) $\succ$ LAGO-U $\succ$ KNN, where "$\sim$" means "not significantly different from" and "$\succ$" means "significantly better than". This conclusion holds at the 5% significance level.

Source	SS ($\times 10^{-3}$)	df	MS ($\times 10^{-3}$)	F_0	P-Value
Methods	17.785	5	3.557	6.404	<0.0001
Cntr1	*2.570*	*1*	*2.570*	*4.626*	*0.0339*
Cntr2	*0.012*	*1*	*0.012*	*0.021*	*0.8856*
Cntr3	*0.291*	*1*	*0.291*	*0.523*	*0.4712*
Cntr4	*4.566*	*1*	*4.566*	*8.220*	*0.0051*
Cntr5	*0.884*	*1*	*0.884*	*1.591*	*0.2101*
Cntr6	*16.309*	*1*	*16.309*	*29.361*	*<0.0001*
Replicates	622.811	99	6.291	11.326	<0.0001
Error	274.953	495	0.555		
Total	915.549	599			

Table 4.11: ANOVA analysis of differences among methods for the Mysim data.

4.4.2 The Mysim-LAGO Data

In this simulation study, we generate data from the mechanism similar to LAGO by pre-specifying the locations and the bandwidths of the kernel functions given in (4.16); we refer to this simulated data as the Mysim-LAGO data. Predictors are generated using the same mechanism described in Section 4.4.1 for the Mysim data: $(x_1, x_2) \in [-3, 3] \times [-1, 5]$. Class labels, however, need to be generated from LAGO. Note that LAGO is an estimate of the density ratio of class 1 and class 0, hence the LAGO score itself is not a probabilistic estimate. However, we can easily change LAGO score $\hat{f}(\cdot)$ to the probability scale using a

logistic transformation $\exp(\beta_0+\beta_1\hat{f}(\cdot))/(1+\exp(\beta_0+\beta_1\hat{f}(\cdot)))$, where $\beta_1 > 0$ which ensures this logistic transformation to be a monotonically increasing function in the LAGO score $\hat{f}(\cdot)$ and, hence, preserves the same ranking as $\hat{f}$. Given a data point $\mathbf{x}$ and its probability of being class 1, $p_{\mathbf{x}}$, it is very easy to determine $\mathbf{x}$'s class label (either 1 or 0) by a Bernoulli distribution with probability of success $p_{\mathbf{x}}$.

Recall that LAGO is a special RBF network where we put kernel functions at and only at the class-1 training points. In order to generate data from LAGO, we need to know the locations and the bandwidths of the kernel functions. For simplicity, we put five kernels within the square $[-2,-1]\times[3,4]$ and another nine within the square $[1,2]\times[0,1]$ (see Figure 4.11). All the kernel functions are assigned the same bandwidth $r = 0.25$. A logistic link is used to transfer the LAGO score to probability. I use $(\beta_0,\beta_1) = (-2.8, 45)$ and the resulting probability surface is shown in Figure 4.11.

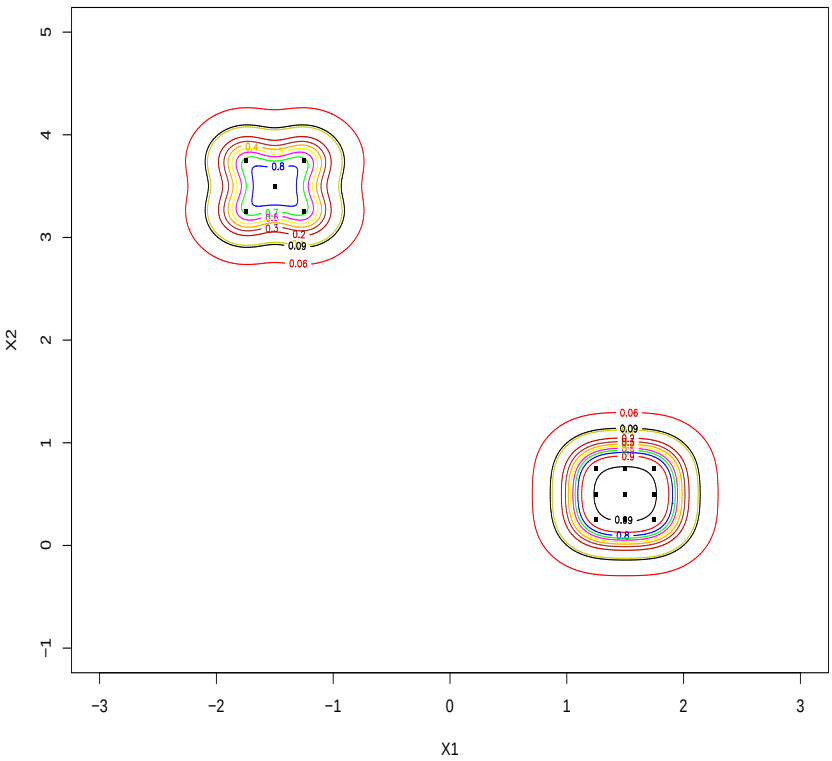

Figure 4.11: Probability surface of the Mysim-LAGO data. Dots represent the centers of the fourteen kernel functions. The bandwidth of all the kernels is $r = 0.25$, and $(\beta_0,\beta_1) = (-2.8, 45)$.

The performance of LAGO, KNN, SVM and ASVM is compared on the Mysim-LAGO data. Like the simulation study in Section 4.4.1, comparison is conducted on average precision on 100 experiments. In each experiment, we generate a set of training data, a set of validation data, and a set of test data from the probability surface calculated by LAGO with kernel functions at those $5 + 9$ locations, bandwidth $r = 0.25$, and logistic transformation coefficient $(\beta_0, \beta_1) = (-2.8, 45)$. The sample sizes of the training, validation and test data sets are all 400. For all the methods, tuning parameters are chosen on the validation data and performance is compared on the test sets. The considered grid values for the tuning parameters of different methods are the same as those in Section 4.4.1.

Figure 4.12 presents boxplots of the 100 experimental average precisions for all the methods compared. Table 4.12 lists the expressions of five non-orthogonal contrasts and their corresponding estimated values. Table 4.13 shows the ANOVA table of comparing differences among methods. The conclusion of the comparison is (LAGO-T $\sim$ LAGO-G $\sim$ KNN) $\succ$ (LAGO-U $\sim$ ASVM $\sim$ SVM), where "$\sim$" means "not significantly different from" and "$\succ$" means "significantly better than". This conclusion holds at the 5% significance level. LAGO and KNN are relatively more stable than SVM and ASVM; notice those outliers of SVM and ASVM in Figure 4.12.

Contrast	Expression	Estimate
Cntr1	$(\mu_G + \mu_T)/2 - \mu_U$	0.0342
Cntr2	$\mu_T - \mu_G$	0.0012
Cntr3	$\mu_A - \mu_S$	0.0116
Cntr4	$\mu_K - \mu_U$	0.0253
Cntr5	$(\mu_G + \mu_T)/2 - (\mu_A + \mu_S)/2$	0.0290
Cntr6	$(\mu_G + \mu_T)/2 - \mu_K$	0.0089
Cntr7	$(\mu_A + \mu_S)/2 - \mu_U$	0.0052

Table 4.12: Estimated contrasts for the Mysim-LAGO data.

Source	SS ($\times 10^{-3}$)	df	MS ($\times 10^{-3}$)	F_0	P-Value
Method	129.795	5	25.959	7.775	<0.0001
Cntr1	*77.955*	*1*	*77.955*	*23.348*	*<0.0001*
Cntr2	*0.070*	*1*	*0.070*	*0.021*	*0.8851*
Cntr3	*6.734*	*1*	*6.734*	*2.017*	*0.1562*
Cntr4	*32.094*	*1*	*32.094*	*9.612*	*0.0020*
Cntr5	*84.353*	*1*	*84.353*	*25.264*	*<0.0001*
Cntr6	*5.233*	*1*	*5.233*	*1.567*	*0.2112*
Cntr7	*1.769*	*1*	*1.769*	*0.530*	*0.4670*
Replicate	4015.815	99	40.564	12.149	<0.0001
Error	1652.726	495	3.339		
Total	5798.336	599			

Table 4.13: ANOVA analysis of differences among methods for the Mysim-LAGO data.

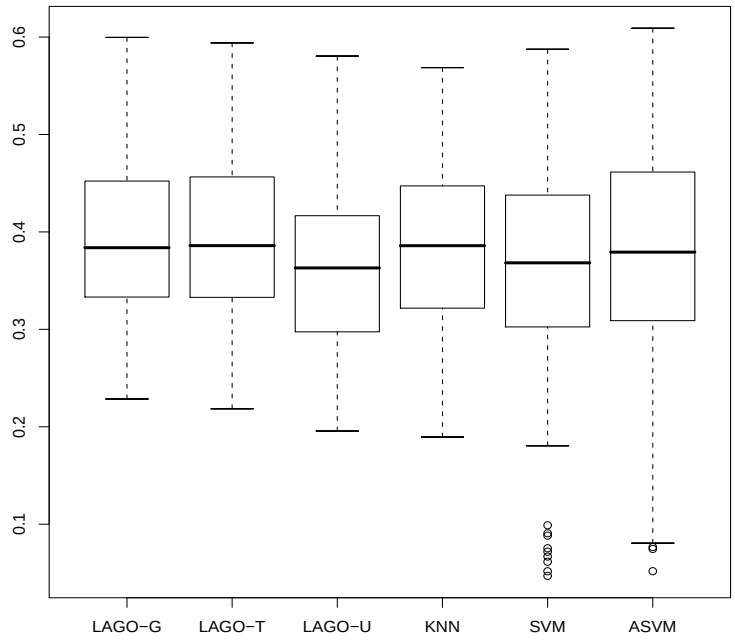

Figure 4.12: The test-set average precisions of different methods evaluated on 100 Mysim-LAGO data sets.

4.5 Conclusion

A computationally efficient statistical detection method called LAGO, which stands for locally adjusted GO estimator, is proposed in this chapter. LAGO is constructed to approximate the density ratio f in two steps. In the first step, the density p_1 is estimated by a kernel density estimator where the bandwidth is calculated as the average distance between each class-1 observation and its K nearest neighbors from class 0. This special bandwidth selection is originally inspired by an ancient game which is now known as GO. In the second step, instead of estimating p_0 functionally, we adjust p_1 piecewisely by a local constant. It can be shown that the amount of adjustment is roughly inversely proportional to the bandwidth calculated in the first step.

The relationship between LAGO and support vector machine (SVM) is also discussed in this chapter. Principally, LAGO can be viewed as a special self-defined SVM where all and only the class-1 observations are support vectors and the kernel parameter γ is set to be proportional to the bandwidth parameter calculated in the first step of constructing LAGO. This gives LAGO a significant computational advantage over SVM.

LAGO methods with three different kernel functions (Gaussian or LAGO-G, triangular or LAGO-T, uniform or LAGO-U), together with KNN, SVM and ASVM, are applied to the NCI data and the two simulated data sets. Results show that LAGO-G and LAGO-T are the two and the only two methods that are consistently among the best models across all the data sets.

It can be shown that the LAGO estimator is asymptotically unbiased up to a constant under some ideal conditions. These conditions are:

1. The density of the class 0, p_0, should be roughly a constant within a tiny neighborhood of each class-1 observations.

2. For each class-1 observation, only the nearest neighboring observation from class 0 is used to calculate the bandwidth, i.e., $K = 1$.

3. The uniform kernel is used.

Due to the extremely unbalanced feature of statistical detection problems, it should be safe to assume that within a local neighborhood the density of class 0, p_0, is relatively flat in comparison with p_1, the density of class 1. In practice, condition (1) should be reasonably easy to satisfy. Conditions (2) and (3) make the proof much easier; however, they are not necessary conditions. Relaxing these conditions will not create any fundamental difficulty, but will undoubtedly make explicit calculations such as (4.8) and (4.11) much more tedious. If condition (1) is not satisfied, LAGO is a biased estimator of the density ratio f. This bias, however, will not affect LAGO's performance in statistical detection problems where interest focuses on a correct relative ranking rather than an accurate prediction.

In the next chapter, I will discuss how to place LAGO into a Bayesian framework.

Chapter 5

Bayesian LAGO

5.1 Motivation

LAGO only provides a point estimate of a test point's possibility of being class 1; however, the uncertainty of this estimate is not quantified. Moreover, LAGO is not a well-calibrated probabilistic estimate. In some applications, a correct ranking is not enough; what is needed is an accurately calibrated prediction. Zadrozny and Elkan (2001a, b) gave examples of scenarios where estimation of $p(y = 1|\mathbf{z})$ is important. In drug discovery, it is also beneficial to know a compound's probability of being active $p(y = 1|\mathbf{z})$ given its BCUT descriptors $\mathbf{z}$. For example, suppose the cost of further investigating a promising compound $\mathbf{z}_i$ is $t(\mathbf{z}_i)$, researchers can pick the top m compounds that are most likely to be active for further investigation such that the total expected cost is still within the budget, i.e., $\sum_{i=1}^{m} p(y_i = 1|\mathbf{z}_i)t(\mathbf{z}_i) \leq$ budget. Moreover, it is even more desirable if the variability of $p(y = 1|\mathbf{z})$ is also known. For instance, if two compounds, $\mathbf{z}_j$ and $\mathbf{z}_k$, are both predicted to be active with a high probability of 0.9 but the variability of $\mathbf{z}_j$ is twice as much as $\mathbf{z}_k$, then it is natural for a pharmaceutical company to spend its limited resources investigating compound $\mathbf{z}_k$ rather than $\mathbf{z}_j$.

In Section 4.1.3 we show that under certain circumstances LAGO is unbiased for the

density ratio f up to a constant and hence can be transformed to an unbiased estimate for the posterior probability $p(y = 1|\mathbf{z})$ by equation (4.1). However, those assumptions are restrictive and thus in many situations the transformation suggested by (4.1) will not provide a good estimation of $p(y = 1|\mathbf{z})$. We would like to be able to estimate the probability $p(y = 1|\mathbf{z})$ in a broader range of circumstances and obtain the variability of the probabilistic estimate. Bayesian methods provide a nice framework to fulfill this goal.

One intuitive way to obtain a probabilistic LAGO, $p_{\hat{f}}$, is to pass the original LAGO score through either a logistic transformation or a probit transformation, that is

$$\begin{aligned} \text{Logit link:} \quad & p_{\hat{f}} = \frac{e^{\beta_0+\beta_1\hat{f}}}{1+e^{\beta_0+\beta_1\hat{f}}} \\ \text{Probit link:} \quad & p_{\hat{f}} = \Phi(\beta_0 + \beta_1\hat{f}), \end{aligned} \tag{5.1}$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution and $\beta_1 > 0$ which ensures (5.1) is a monotonically increasing function in the LAGO score $\hat{f}$. This can be viewed as a two-stage approach: train LAGO and choose the optimal pair of parameters (K, α) in the first stage; fit a logistic regression and obtain the regression coefficients (β_0, β_1) in the second stage. This two-stage approach is popular in practice and widely discussed in literature (see Section 5.2.1). The Bayesian approach, on the other hand, allows us to execute these two steps simultaneously.

In this chapter, we propose a Bayesian framework for LAGO (BLAGO) that calculates the posterior distribution over a grid of (K, α), integrating out β_0 and β_1 using the Laplace approximation. Inference on any quantity of interest can be made under this Bayesian framework and a more accurate probabilistic estimate can be achieved using the idea of Bayesian Model Averaging (see Hoeting *et al.*, 1999 for an overview). BLAGO not only provides an estimation of a test point's probability of being class 1 but also captures the uncertainty of this estimation.

The rest of this chapter is organized as follows. Section 5.2 reviews the materials related to BLAGO. The methodology and implementation of BLAGO are discussed in Section 5.3.

A frequentist approach that enables LAGO to produce probabilistic estimates is presented in Section 5.4. BLAGO is applied to the NCI data in Section 5.5. Two aspects of the performance of BLAGO are discussed:

1. Performance comparison with LAGO-G, SVM and ASVM in terms of average precision. The reason LAGO-G, SVM and ASVM are picked is that LAGO-G, LAGO-T, ASVM outperform SVM, LAGO-U and KNN; the performances of LAGO-G and LAGO-T are almost identical; and the performances of SVM, LAGO-U and KNN are similar.

2. Comparison of BLAGO with LAGO in terms of deviance.

The application of BLAGO on two simulated data sets is shown in Sections 5.6 and 5.7. In each case, two aspects of the performance of BLAGO are considered:

1. Performance comparison with LAGO-G, KNN, SVM and ASVM in terms of average precision based on the results of 100 experiments. In each experiment, models are fit on the training set and performance is compared on the test set. The sample sizes of both the training and test set are 400.

2. Performance of BLAGO in estimating true probabilities and quantifying uncertainty for 13 representative points.

5.2 Literature Review

Several ingredients are required to develop a Bayesian approach to a certain model, e.g., LAGO. First, the predictions of the model must be well-calibrated so that they correspond to class probabilities. Second, the model must be specified in the Bayesian framework. Finally, efficient approximations and computational methods are required to evaluate the posterior probability distribution. In Sections 5.2.1 - 5.2.4, we discuss these issues and review related work.

5.2.1 Transforming Ranking Scores to Probabilities

For classification problems, some classifiers produce scores that rank items well but are not well-calibrated. Take SVM for example. The signed distance indicates the confidence that SVM has in its prediction. The larger the score, the higher probability that the given observation belongs to class 1. However, the signed distance itself is not a well-calibrated probabilistic estimate. The situation is similar for the LAGO score. Even though the LAGO score, whose range is $(0, 1)$, is able to rank the observations well by their possibilities of being class 1, it is not a well-calibrated estimate of the true probability.

Several papers have considered calibrating a score function to a probability. Let $p(q|\mathbf{x})$ be the probability that $\mathbf{x}$ belongs to class q. When $q \in \{1, -1\}$, Platt (1999a) proposed a parametric approach to map the signed distance of SVM $sd(\mathbf{x})$ into probability estimates $\hat{p}(q|\mathbf{x})$ through a sigmoid (or logistic) function

$$\hat{p}(q = 1|\mathbf{x}) = \frac{1}{1 + e^{Asd(\mathbf{x})+B}},$$

where parameters A and B are estimated via maximum likelihood. This method works well if the relationship between SVM scores and the probabilities $p(q|\mathbf{x})$ appears to be sigmoidal.

Zadrozny and Elkan (2002) suggested using binning, which is a non-parametric method, if the mapping function is unknown. In binning, the training observations are sorted by their scores and the sorted set is divided into b subsets of equal size, called bins. The lower and upper boundary scores of each bin are computed. Given a data point $\mathbf{x}$, its probability of being class q is estimated by the proportion of class-q training points in the bin where $sd(\mathbf{x})$ lies. One disadvantage of binning is the difficulty in selecting the number of bins b. Results can be very sensitive to this choice.

Besides binning, Zadrozny and Elkan (2002) proposed another non-parametric method using isotonic regression. This method is motivated by the fact that if the classifier ranks items correctly, the mapping from scores into probabilities should be isotonic (non-

decreasing). Isotonic regression is a non-parametric form of regression in which the mapping function is assumed to be chosen from the class of all isotonic functions. Zadrozny and Elkan applied the pair-adjacent violators (PAV) algorithm to find the step-wise constant isotonic regression such that the mean-squared error is minimized.

All the methods discussed above are two-stage approaches, in which $sd(\mathbf{x})$ is treated as a known covariate during estimation of the transformation.

5.2.2 Casting Statistical Methods in Bayesian Framework

Bayesian methods have been applied to statistical methods such as Generalized Linear Models (e.g., Zeger and Karim, 1991; Albert and Chib, 1993; Mallick and Gelfand, 1994), Neural Networks (Neal, 1996), tree models (Chipman *et al.*, 1998, 2000, 2006), SVM (Tipping, 2001), and KNN (Holmes and Adams, 2002). Since BLAGO is closely related to the work of Holmes and Adams (2002), we will discuss their work in detail.

Holmes and Adams (2002) introduced a Bayesian framework for KNN (BKNN) to capture model uncertainty. BKNN outperforms KNN significantly for applications where un-equal misclassification weights are used for different classes. They adopted the pseudo-likelihood (see Section 5.2.3 for more details) of the data given the parameters β and K as follows,

$$p(\mathbf{Y}|\mathbf{X},\beta,K)=\prod_{i=1}^{n}\frac{\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i)\}}{\sum_{q=1}^{Q}\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=q)\}}, \tag{5.2}$$

where $(\mathbf{Y},\mathbf{X})$ are the training data, $N(\mathbf{x}_i,K)$ denotes the set of $\mathbf{x}_i$'s K nearest neighboring observations, $\beta>0$ is a parameter that controls the strength of association between the neighboring y_i, and Q is the total number of classes. The pseudo-likelihood in (5.2) is not a full likelihood, because the likelihood component for response y_i depends on the class labels of the other responses y_j with $j\in N(\mathbf{x}_i,K)$. Take $Q=2$ for example. Each piece

in (5.2) can be rewritten as

$$\begin{aligned} p(y_i|\mathbf{x}_i,\beta,K) &= \frac{\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i)\}}{\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i)\}+\exp\{(\beta/K)\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j\neq y_i)\}} \\ &= \frac{\exp\{(\beta/K)[2\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i)-K]\}}{1+\exp\{(\beta/K)[2\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i)-K]\}} \\ &= \frac{\exp\{-\beta+2\beta\hat{p}_{\mathrm{k}}(y_i)\}}{1+\exp\{-\beta+2\beta\hat{p}_{\mathrm{k}}(y_i)\}}, \end{aligned} \tag{5.3}$$

which is equivalent to fitting a logistic regression on

$$\hat{p}_{\mathrm{k}}(y_i)=\frac{1}{K}\sum_{j\in N(\mathbf{x}_i,K)}\mathrm{I}(y_j=y_i) \tag{5.4}$$

with slope parameter (2β) and intercept $(-\beta)$.

Treating parameters β and K as random variables with posterior distribution $p(\beta,K|\mathbf{Y},\mathbf{X})$, the marginal predictive distribution for a new data point $(\mathbf{x}_{n+1},y_{n+1})$ based on the training data $(\mathbf{X},\mathbf{Y})$ is given by

$$p(y_{n+1}|\mathbf{x}_{n+1},\mathbf{X},\mathbf{Y})=\sum_K\int p(y_{n+1}|\mathbf{x}_{n+1},\mathbf{X},\mathbf{Y},\beta,K)p(\beta,K|\mathbf{X},\mathbf{Y})d\beta. \tag{5.5}$$

Here the posterior distribution

$$p(\beta,K|\mathbf{X},\mathbf{Y})\propto p(\mathbf{Y}|\mathbf{X},\beta,K)p(\beta,K).$$

Except for the fact that β should be positive, little prior information is available for K and β. Therefore, Holmes and Adams adopted the independent default prior densities,

$$p(\beta,K)=p(\beta)p(K);\quad p(K)=\mathrm{UNIF}[1,\ldots,k_{\max}],k_{\max}=n;\quad p(\beta)=c\mathrm{I}(\beta>0),$$

where c is a constant and the improper prior on β is uniform on $\mathbb{R}^+$.

Holmes and Adams applied Markov chain Monte Carlo based on a random walk Metropolis-Hastings algorithm to draw M samples from the posterior $p(\beta,K|\mathbf{X},\mathbf{Y})$ and then (5.5) can be approximated by

$$p(y_{n+1}|\mathbf{x}_{n+1},\mathbf{X},\mathbf{Y}) \approx \frac{1}{M}\sum_{j=1}^{M}p(y_{n+1}|\mathbf{x}_{n+1},\mathbf{X},\mathbf{Y},\beta^{(j)},K^{(j)}), \tag{5.6}$$

where $K^{(j)}$ and $\beta^{(j)}$ are the jth sample in the converged chain.

In Section 5.7.2, BKNN is applied to the Mysim-LAGO data in estimating the posterior probabilities $p(y_i = 1|\mathbf{x}_i)$ for several pre-chosen points $\mathbf{x}_i$. I adopt Homles and Adams (2002)'s Matlab codes with the following settings:

- Priors: $p(K) = \text{UNIF}[1, \ldots, 250]$, and $\beta \sim N(10, 5^2)$. The prior of K is the default given by the original algorithm. The prior of β is chosen to ensure that the range of β is wide enough under the constraint that $\beta > 0$.

- A joint proposal from the current state (K, β) to a new one $(\hat{K}, \hat{\beta})$:

$$\begin{aligned} \hat{K} &= K \pm \text{UNIF}[0, 1, 2], \\ \hat{\beta} &= \beta + N(0, 1). \end{aligned}$$

  The resulting acceptance rate is around 30%.

- The first 100,000 samples are discarded for burn-in and the later 100,000 samples are used for inference.

5.2.3 Pseudo-likelihood

The idea of using pseudo-likelihood was first introduced by Besag (1974, 1975) to model the spatial interaction of lattice systems. Suppose that a system consists of n sites, each associated with a random variable $l_i, i = 1, \ldots, n$. Then the joint probability distribution of the variables can be expressed as a product of conditional distributions

$$\prod_{i=1}^{n} P(l_i|l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_n). \tag{5.7}$$

If l_i are independent, (5.7) becomes the common full likelihood. However, this independence assumption is not very realistic for modeling spatial relationships. Some dependence structures can be proposed, and different dependence assumptions result in different kinds

of models. For example, the conditional distribution of l_i might depend only on those l_j where site j is in the neighborhood of site i. This assumption leads to Markov random field models, which have been widely used in image processing (Besag, 1986) and modeling network tomography (Liang and Yu, 2003; Robins *et al.*, 2007). In image processing, it might not be possible to write down the full likelihood for all the pixels in a picture. However, it is much easier to break down the whole picture to several local regions and then construct conditional likelihood for the pixels involved in each region. Pixels in the same region are expected to share some common characteristics such as color and intensity. Under the assumption that dependence structures only exist within the local regions and two pixels belongs to different regions are independent, taking the product of the conditional likelihoods over the local regions gives the psuedo-likelihood. Similarly, in modeling social networks, it is relatively hard to consider the complex global dependence structure for all nodes. Breaking the whole network into sub-networks and focusing the dependence structures only within the sub-networks makes the problem much easier to solve.

Besag (1975) illustrated how to construct a class of valid probability distributions associated with the site variables $l_1, \ldots, l_n$ according to the Hammersley-Clifford theorem. The theorem says that the joint probability distribution of $l_1, \ldots, l_n$ must be a product of functions, one function corresponds to one *clique*. A clique is defined as any set of sites which either consists of a single site or else in which every site is a neighbor of every other site in the set. By only considering cliques containing no more than two sites and restricting the conditional distribution to the exponential family, Besag (1975) proposed the so-called auto-logistic model when l_i are binary variables. The conditional probabilities are given by

$$P(l_i|l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_n) = \frac{\exp\{l_i(\alpha_i + \sum_{j\neq i} \beta_{i,j} l_j)\}}{1 + \exp\{\alpha_i + \sum_{j\neq i} \beta_{i,j} l_j\}}, \tag{5.8}$$

where $\beta_{i,j} \equiv \beta_{j,i}$ and $\beta_{i,j} = 0$ unless sites i and j are neighbors of each other. There might be too many parameters in model (5.8), some parameters can or need to be set to zero, equated or constrained.

For KNN, the membership probabilities for a given data point are determined by the class labels of its K nearest neighboring observations. Note that, the pseudo-likelihood of BKNN given in (5.3) is actually a reduced auto-logistic model by restricting $\alpha_i = -\beta$ and $\beta_{i,j} = \frac{2\beta}{K}$, i.e., an auto-logistic model with a single free parameter β.

For BLAGO, a data point's probability of being class 1 is conditional on the nearby class-1 observations and their corresponding K nearest class-0 neighbors in the training set. To model this dependence structure, we also adopt a similar form of pseudo-likelihood used in BKNN by replacing $\hat{p}_{\mathrm{k}}(y_i)$ in (5.3) with the LAGO estimate given in (4.16) and introducing an extra free intercept parameter. This will be described later in Section 5.3.

5.2.4 Techniques for Approximate Bayesian Inference

The objective of Bayesian methods is to produce predictive probabilities for the test data and conduct statistical inference on some quantities of interest. To achieve this objective, evaluating the moments of a function with respect to the posterior distribution for model parameter $\boldsymbol{\beta}$ is unavoidable. Suppose that the posterior distribution $Q(\boldsymbol{\beta}) = \frac{1}{c}\mathrm{L}(\boldsymbol{\beta})\pi(\boldsymbol{\beta})$, where $\pi(\boldsymbol{\beta})$ is the prior distribution for $\boldsymbol{\beta}$, $\mathrm{L}(\boldsymbol{\beta})$ is the likelihood function of the data given $\boldsymbol{\beta}$, and $c = \int \mathrm{L}(\boldsymbol{\beta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta}$ is the normalizing constant. We are interested in the posterior expectation of a quantity $g(\boldsymbol{\beta})$, that is,

$$\mathrm{E}[g(\boldsymbol{\beta})] = \int g(\boldsymbol{\beta})Q(\boldsymbol{\beta})d\boldsymbol{\beta}. \tag{5.9}$$

In some problems, (5.9) can be evaluated analytically; however, numerical methods of approximation are required in most applications. The most widely used approximation methods include quadrature approximation, the Laplace approximation based on asymptotics, and (Markov chain) Monte Carlo integration.

Quadrature rules approximate the posterior expectation in (5.9) by

$$\mathrm{E}[g(\boldsymbol{\beta})] \approx \sum_{j=1}^{M} \omega_j g(\boldsymbol{\beta}_j)Q(\boldsymbol{\beta}_j),$$

for some weights ω_j and grid points $\boldsymbol{\beta}_j$. Depending on how the weights and the grid points are specified, quadrature approximation has several variations, which include simple methods such as trapezium's rule and Simpson's rule and advanced methods such as Gauss-quadrature rules (Jerri, 1999). The quadrature methods were originally designed to compute one-dimensional integrals. For multivariate cases, one approach is to phrase the multiple integral as repeated one-dimensional integrals by using Fubini's theorem (e.g., Thomas and Finney, 1996). An outstanding problem of this approach is that the number of function evaluations increases exponentially with the dimension of $\boldsymbol{\beta}$. Two alternative methods can be used to overcome this so-called curse of dimensionality problem: the Laplace approximation and Monte Carlo methods.

The Laplace approximation (e.g., De Bruijn, 1970) has been used by various authors to derive approximate posterior moments, marginal densities (Tierney and Kadane, 1986; Tierney *et al.*, 1989), and Bayes factors (Kass and Raftery, 1995). In general, if r is a smooth function of a $d-$dimensional vector $\boldsymbol{\beta}$ having a minimum at $\hat{\boldsymbol{\beta}}$ and b is another smooth function of $\boldsymbol{\beta}$, then, under suitable regularity conditions (Kass *et al.*, 1990), the well-known Laplace approximation to an integral is given by

$$\begin{aligned} I_n &= \int b(\boldsymbol{\beta}) \exp\{-nr(\boldsymbol{\beta})\} d\boldsymbol{\beta} \\ &= (\frac{2\pi}{n})^{\frac{d}{2}} (\det \Sigma_{\hat{\boldsymbol{\beta}}})^{\frac{1}{2}} \exp\{-nr(\hat{\boldsymbol{\beta}})\}\{b(\hat{\boldsymbol{\beta}}) + O(n^{-1})\}, \end{aligned} \tag{5.10}$$

where $\Sigma_{\hat{\boldsymbol{\beta}}}$ is the inverse of the Hessian matrix of $r(\cdot)$ evaluated at the minimum $\hat{\boldsymbol{\beta}}$.

Writing the normalizing constant c explicitly, the posterior expectation in (5.9) can be expressed as the ratio of two integrals

$$\begin{aligned} \mathrm{E}[g(\boldsymbol{\beta})] &= \int g(\boldsymbol{\beta}) Q(\boldsymbol{\beta}) d\boldsymbol{\beta} \\ &= \frac{\int g(\boldsymbol{\beta}) \mathrm{L}(\boldsymbol{\beta}) \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}}{\int \mathrm{L}(\boldsymbol{\beta}) \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}}. \end{aligned} \tag{5.11}$$

Tierney and Kadane (1986) claimed that when function g is positive, by applying the Laplace approximation to both the numerator and the denominator of (5.11), the error of

the approximate posterior mean will be reduced to $O(n^{-2})$ instead of $O(n^{-1})$. Later, they extended their method to non-positive functions using moment generating functions (see Tierney *et al.*, 1989).

For Gauss-hermite quadrature, Liu and Pierce (1994) considered a systematic method for transforming the variable of integration to ensure that the integrand is sampled in an appropriate region. They claimed that the Laplace approximation is equivalent to the order one Gauss-Hermite quadrature in their approach, the effectiveness of which depends on whether the ratio of the integrand to some Gaussian density is a smooth function that can be well approximated by a low-order polynomial. This principle of effectiveness sheds some light on under what circumstance the Laplace approximation is accurate enough.

The advantage of the Laplace approximation is its simplicity in computation. The main cost is to calculate the minimum $\hat{\boldsymbol{\beta}}$, which generally can be obtained by Newton-type algorithms. Tierney and Kadane (1986) mentioned that the Laplace approximation is twenty times faster in computational time than a 20-point Gauss-Hermite quadrature in one application. The Laplace approximation might not always be accurate (see Kass *et al.*, 1990; Liu and Pierce, 1994). One assumption of applying the Laplace approximation is that the function $r(\boldsymbol{\beta})$ should be unimodal; it might not be straightforward to extend to multimodal cases.

Monte Carlo methods can be applied in situations where the Laplace approximation is either not valid or works poorly. Suppose $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(M)}$ are independent samples from Q, then expectation in (5.9) can be estimated by

$$\mathrm{E}[g(\boldsymbol{\beta})] \approx \frac{1}{M} \sum_{t=1}^{M} g(\boldsymbol{\beta}^{(t)}). \tag{5.12}$$

Different methods can be used to draw samples from Q such as importance sampling, rejection-acceptance techniques, Markov chain Monte Carlo (MCMC) methods such as Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith, 1990) and the Metroplis-Hastings sampling (Metropolis *et al.*, 1953; Hastings, 1970) and so on.

5.3 Bayesian LAGO

In this section, we place LAGO into a Bayesian framework, under which uncertainty can be quantified. Two main components of a Bayesian method are the likelihood function and the prior distribution. We discuss these two issues in the following two sections.

5.3.1 Likelihood Function

As mentioned before, given the training data $\{y_i, \mathbf{x}_i\}_{i=1}^n$ and a certain pair of (K, α), the LAGO scores $\hat{f}(\mathbf{x}_i)$ are not probabilities. We can map them to the probability scale by fitting a generalized linear model with a logit link function as follows,

$$\theta_i = p(y_i = 1|\mathbf{x}_i) = \frac{\exp\{\beta_0 + \beta_1 \hat{f}(\mathbf{x}_i)\}}{1 + \exp\{\beta_0 + \beta_1 \hat{f}(\mathbf{x}_i)\}}, \tag{5.13}$$

where $\beta_1 > 0$ ensures (5.13) to be a monotonically increasing function in the LAGO scores $\hat{f}(\mathbf{x}_i)$. Then, the pseudo-likelihood function can be written as the product of likelihoods of n Bernoulli variables, which is given by

$$\mathrm{L}(\boldsymbol{\theta}; \mathbf{Y}) = \prod_{i=1}^{n} \theta_i^{y_i} [1 - \theta_i]^{1-y_i}. \tag{5.14}$$

The corresponding log likelihood is

$$l(\boldsymbol{\theta}; \mathbf{Y}) = \sum_{i=1}^{n} [y_i \log(\frac{\theta_i}{1 - \theta_i}) + \log(1 - \theta_i)]. \tag{5.15}$$

Like (5.2), the expression given in (5.14) is not a proper likelihood function, because the component θ_i depends on other data points through the LAGO score $\hat{f}(\mathbf{x}_i)$. Note that θ_i and θ_j are not independent in that the basis functions in estimating $\hat{f}(\mathbf{x}_i)$ and $\hat{f}(\mathbf{x}_j)$ are correlated because they share the same kernel functions. Ignoring this dependence structure and using (5.14) as the proper likelihood function will lead to underestimate of the model uncertainty. I will come back to this issue later in Section 5.7.2.

Given the fact that

$$\theta_i = \frac{e^{\mathbf{v}_i^T\boldsymbol{\beta}}}{1+e^{\mathbf{v}_i^T\boldsymbol{\beta}}},\ 1-\theta_i = \frac{1}{1+e^{\mathbf{v}_i^T\boldsymbol{\beta}}},\ \log\frac{\theta_i}{1+\theta_i} = \mathbf{v}_i^T\boldsymbol{\beta},$$

where $\mathbf{v}_i = (1, \hat{f}(\mathbf{x}_i))^T$, the likelihood given in (5.15) can also be written in terms of $\boldsymbol{\beta}$ as follows

$$l(\boldsymbol{\beta}; \mathbf{Y}) = \sum_{i=1}^{n}[y_i(\mathbf{v}_i^T\boldsymbol{\beta}) - \log(1+e^{\mathbf{v}_i^T\boldsymbol{\beta}})]. \tag{5.16}$$

5.3.2 Prior Distributions

The BLAGO model has four parameters: K and α for constructing the LAGO score; (β_0, β_1) for the mapping. Following Holmes and Adams (2002), we assume the prior distribution of K is uniform on a discrete grid of values $\{2, \cdots, K_{max}\}$, the prior of α is also uniform on a discrete grid (given below), and the prior of $\boldsymbol{\beta} = (\beta_0, \beta_1)$ follows a bivariate distribution BVN$(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. We also assume prior independence of K, α and $\boldsymbol{\beta}$.

In order to explore the parameter space extensively, K_{max} should be set as large as possible. A larger value of K_{max}, however, means a greater expense in computation. To balance this trade-off, I choose K_{max} according to my experience from fitting LAGO in Chapter 4. Let $K_{max} = 20$ for the NCI data and $K_{max} = 60$ for the two simulated data sets. As for α, I don't want it to be too large to lose its capability of capturing the local structure of the data. Although α could be any positive value, I specify a uniform prior on a grid, i.e., $\alpha \sim$ UNIF$\{0.1, 0.25, 0.5, 1, 1.08, 1.16, 1.25, 1.34, 1.45, 1.56, 1.67, 1.8, 1.94, 2.09, 2.25, 2.42, 2.61, 2.81, 3.02, 3.25, 3.5, 3.77, 4.06, 5\}$. This grid has the same range as the one used for tuning LAGO but has more values.

The scale of the LAGO score given in (4.16) changes dramatically when (K, α) vary, which causes a huge fluctuation in the parameters (β_0, β_1) in (5.13). To reduce the dependence between (K, α) and βs, we re-scale the LAGO score with mean 0 and standard deviation 1 before fitting the logistic regression. Technical details are provided later in Section 5.4. Moreover, the scale of the LAGO score is very different for various data

sets. Scaling makes the prior distribution of (β_0, β_1) universal without worrying about the scales of the LAGO scores for different applications. Except for $\beta_1 > 0$, we do not have much information in $\boldsymbol{\beta} = (\beta_0, \beta_1)$. Therefore, a fairly diffuse prior with mean vector $\boldsymbol{\mu}_0 = (-10, 20)$, and diagonal covariance matrix $\boldsymbol{\Sigma}_0 = \text{diag}(100, 100)$ is chosen for $\boldsymbol{\beta}$. For β_1, a variance of 100 is large enough to ensure its 95% lower confidence bound to be non-negative. According to my experience, this prior is by far diffuse enough to cover almost all possible (β_0, β_1) values given different grid values of (K, α).

5.3.3 Computational Details

With the likelihood function and the specification of the prior distributions for the parameters, now we are able to calculate the posterior distribution and make inference on model parameters and predictions.

Instead of drawing posterior samples using the MCMC techniques, our approach evaluates the posterior distribution over a grid of (K, α) pairs using the Laplace approximation. That is, we do not explicitly draw $\boldsymbol{\beta} = (\beta_0, \beta_1)$, but integrate it out using the Laplace approximation. Given a new data point $(y, \mathbf{z})$, the quantity of interest is its probability of being class 1, $\theta = Pr(y = 1|\mathbf{z})$, whose posterior distribution is as follows,

$$Pr(\theta|\text{train}) = \sum_{j=1}^{M} Pr(\theta|\text{train}, K^{(j)}, \alpha^{(j)}) Pr(K^{(j)}, \alpha^{(j)}|\text{train}), \tag{5.17}$$

where $\{K^{(j)}, \alpha^{(j)}\}_{j=1}^{M}$ are all the (K, α) pairs considered, i.e., $K \times \alpha = \{2, 3, \ldots, 20\} \times \{0.1, 0.25, 0.5, 1, 1.08, 1.16, 1.25, 1.34, 1.45, 1.56, 1.67, 1.8, 1.94, 2.09, 2.25, 2.42, 2.61, 2.81, 3.02, 3.25, 3.5, 3.77, 4.06, 5\}$ for the NCI data. Therefore, there are $19 \times 24 = 456$ distinct (K, α) pairs all together. For the simulated data, since only odd values of K are used, $29 \times 24 = 696$ distinct (K, α) pairs are considered. Equation (5.17) is an average of the posterior distributions under each (K, α) pair, weighted by its posterior probability. By Bayes rule, the

posterior probability of each (K, α) pair is expressed as

$$Pr(K^{(j)}, \alpha^{(j)}|\text{train}) = \frac{Pr(\text{train}|K^{(j)}, \alpha^{(j)})Pr(K^{(j)}, \alpha^{(j)})}{\sum_{l=1}^{M} Pr(\text{train}|K^{(l)}, \alpha^{(l)})Pr(K^{(l)}, \alpha^{(l)})}. \tag{5.18}$$

Since a uniform prior is assigned to each (K, α) pair , the terms $Pr(K^{(j)}, \alpha^{(j)}) = Pr(K^{(l)}, \alpha^{(l)}) = 1/M$, and thus cancelled. Therefore, calculating (5.18) only requires the calculation of the conditional probability $Pr(\text{train}|K^{(j)}, \alpha^{(j)})$, which is given by

$$\begin{aligned} Pr(\text{train}|K^{(j)}, \alpha^{(j)}) &= \int Pr(\text{train}|\boldsymbol{\beta}, K^{(j)}, \alpha^{(j)})Pr(\boldsymbol{\beta}|K^{(j)}, \alpha^{(j)})d\boldsymbol{\beta} \\ &= \int L(\boldsymbol{\beta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta} \\ &= \int \exp(\mathcal{L}(\boldsymbol{\beta}))d\boldsymbol{\beta} \\ &= \int \exp\{-n(-\frac{1}{n}\mathcal{L}(\boldsymbol{\beta}))\}d\boldsymbol{\beta}, \end{aligned} \tag{5.19}$$

where $\pi(\boldsymbol{\beta})$ is the prior distribution of $\boldsymbol{\beta}$ with $\log \pi(\boldsymbol{\beta}) = \omega(\boldsymbol{\beta})$, and $\mathcal{L}(\boldsymbol{\beta}) = \log(\mathrm{L}(\boldsymbol{\beta})\pi(\boldsymbol{\beta})) = l(\boldsymbol{\beta}) + \omega(\boldsymbol{\beta})$ is the log posterior of $\boldsymbol{\beta}$. By letting

$$b(\boldsymbol{\beta}) = 1, r(\boldsymbol{\beta}) = -\frac{1}{n}\mathcal{L}(\boldsymbol{\beta}),$$

and applying the Laplace approximation given in (5.10), integral (5.19) can approximated by

$$\begin{aligned} & \int \exp\{-n(-\frac{1}{n}\mathcal{L}(\boldsymbol{\beta}))\}d\boldsymbol{\beta} \\ \approx\ & (2\pi)^{d/2} \exp\{\mathcal{L}(\boldsymbol{\beta}^*|K^{(j)}, \alpha^{(j)})\}|H(\boldsymbol{\beta}^*|K^{(j)}, \alpha^{(j)})|^{-1/2} \\ =\ & (2\pi) \exp\{\mathcal{L}(\boldsymbol{\beta}_j^*)\}|H(\boldsymbol{\beta}_j^*)|^{-1/2}, \end{aligned} \tag{5.20}$$

where $\mathcal{L}(\boldsymbol{\beta}_j^*)$ is the log posterior of $\boldsymbol{\beta}$ evaluated at the posterior mode $\boldsymbol{\beta}_j^*$ and $H(\boldsymbol{\beta}_j^*) = \mathcal{L}''(\boldsymbol{\beta}_j^*)$ is the Hessian matrix of the log posterior of $\boldsymbol{\beta}$ evaluated at $\boldsymbol{\beta}_j^*$. The Laplace approximation in (5.20) is the result of a second order Taylor expansion of $\mathcal{L}(\boldsymbol{\beta})$ around the posterior mode $\boldsymbol{\beta}_j^*$.

The posterior mode of $\boldsymbol{\beta}$ is the value $\boldsymbol{\beta}^*$ that maximizes $\mathcal{L}(\boldsymbol{\beta})$, i.e., the solution to the equation $\mathcal{L}^{'}(\boldsymbol{\beta}) = 0$. By Newton's method, new $\boldsymbol{\beta}$ can be updated by

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i - [\mathcal{L}^{''}(\boldsymbol{\beta}_i)]^{-1}\mathcal{L}^{'}(\boldsymbol{\beta}_i).$$

Given that $\pi(\boldsymbol{\beta}) \sim \text{BVN}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, taking the derivative of $\mathcal{L}(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ gives

$$\begin{aligned}
\mathcal{L}^{'}(\boldsymbol{\beta}) &= l^{'}(\boldsymbol{\beta}) + \omega^{'}(\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n}[y_i\mathbf{v}_i - \frac{e^{\mathbf{v}_i^T\boldsymbol{\beta}}}{1+e^{\mathbf{v}_i^T\boldsymbol{\beta}}}\mathbf{v}_i] - \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_0) \\
&= \sum_{i=1}^{n}(y_i - \theta_i)\mathbf{v}_i - \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_0) \\
&= \text{V}^T(\mathbf{Y} - \boldsymbol{\theta}) - \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_0), \qquad (5.21)
\end{aligned}$$

where V is a $n \times 2$ matrix

$$\text{V} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}, \ \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \ \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}.$$

The second derivative of $\mathcal{L}$ is given by

$$\begin{aligned}
\mathcal{L}^{''}(\boldsymbol{\beta}) &= l^{''}(\boldsymbol{\beta}) + \omega^{''}(\boldsymbol{\beta}) \\
&= -\sum_{i=1}^{n}\frac{e^{\mathbf{v}_i^T\boldsymbol{\beta}}}{(1+e^{\mathbf{v}_i^T\boldsymbol{\beta}})^2}\mathbf{v}_i^T\mathbf{x}_i - \boldsymbol{\Sigma}_0^{-1} \\
&= -\sum_{i=1}^{n}\theta_i(1-\theta_i)\mathbf{v}_i^T\mathbf{x}_i - \boldsymbol{\Sigma}_0^{-1} \\
&= -V^TDV - \boldsymbol{\Sigma}_0^{-1}, \qquad (5.22)
\end{aligned}$$

where D is a $n \times n$ diagonal matrix with diagonal entries $d_{ii} = \theta_i(1-\theta_i)$ for $i = 1, 2, \ldots, n$.

The iterative algorithm terminates when the sum of absolute difference between $\boldsymbol{\beta}_{i+1}$ and $\boldsymbol{\beta}_i$ is less than 10^{-6}. The algorithm normally converges after several iterations; therefore, the computation for solving the posterior mode is fairly fast.

Given the posterior probabilities of $Pr(K^{(j)}, \alpha^{(j)}|\text{train}), j = 1, \ldots, M$, the posterior mean of θ can be calculated by

$$\begin{aligned}
\text{E}(\theta|\text{train}) &= \int\int\int \theta Pr(K, \alpha, \boldsymbol{\beta}|\text{train}) d\boldsymbol{\beta} dK d\alpha \\
&= \int\int\int \theta Pr(\boldsymbol{\beta}|K, \alpha, \text{train}) Pr(K, \alpha|\text{train}) d\boldsymbol{\beta} dK d\alpha \\
&= \sum_{j=1}^{M} \underbrace{\left[\int \theta Pr(\boldsymbol{\beta}|\text{train}, K^{(j)}, \alpha^{(j)}) d\boldsymbol{\beta}\right]}_{\boldsymbol{\beta} \sim \text{BVN}(\boldsymbol{\beta}_j^*, H^{-1}(\boldsymbol{\beta}_j^*))} \underbrace{Pr(K^{(j)}, \alpha^{(j)}|\text{train})}_{\text{Laplace}}. \qquad (5.23)
\end{aligned}$$

The posterior distribution $Pr(K, \alpha, \boldsymbol{\beta}|\text{train})$ can be broken down into two pieces, i.e., $Pr(\boldsymbol{\beta}|K, \alpha, \text{train})$ and $Pr(K, \alpha|\text{train})$. The inner integration can be approximated using Monte Carlo by drawing samples from the posterior distribution $Pr(\boldsymbol{\beta}|\text{train}, K^{(j)}, \alpha^{(j)})$ which is $\text{BVN}(\boldsymbol{\beta}_j^*, H^{-1}(\boldsymbol{\beta}_j^*))$. The posterior mean is a weighted average weighted by the posterior probabilities $Pr(K^{(j)}, \alpha^{(j)}|\text{train})$, which can be calculated by (5.18) using the Laplace approximation. In order to estimate the inner integral in (5.23), 100 samples of (β_0, β_1) are simulated from a bivariate normal distribution $\text{BVN}(\boldsymbol{\beta}_j^*, H^{-1}(\boldsymbol{\beta}_j^*))$. As a result, there are 100 samples of θ values for each $(K^{(j)}, \alpha^{(j)})$ pair, denoted as $\theta_{jl}, l = 1, \ldots, 100$. Finally, the inner integral can be estimated by the sample mean as follows:

$$\hat{\theta}_j = \int \theta Pr(\boldsymbol{\beta}|\text{train}, K^{(j)}, \alpha^{(j)}) d\boldsymbol{\beta} \approx \frac{1}{100} \sum_{l=1}^{100} \theta_{jl}.$$

Then, the posterior mean of θ is given by

$$\text{E}(\theta|\text{train}) = \sum_{j=1}^{M} \hat{\theta}_j Pr(K^{(j)}, \alpha^{(j)}|\text{train}). \qquad (5.24)$$

To sum up, the detailed procedures of BLAGO are as follows:

1. Assign uniform prior over a grid of (K, α) pairs and $\text{BVN}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ to $\boldsymbol{\beta}$ with $\boldsymbol{\mu}_0 = (-10, 20), \boldsymbol{\Sigma}_0 = \text{diag}(100, 100)$.

2. For each $(K^{(j)}, \alpha^{(j)})$ pair, calculate its posterior probability using (5.18) and (5.20). Meanwhile, store its corresponding posterior mode $\boldsymbol{\beta}_j^*$ as well as $H(\boldsymbol{\beta}_j^*)$, the Hessian

matrix evaluated at the posterior mode $\boldsymbol{\beta}_j^*$. Compute and save the mean and standard deviation of the LAGO score for the training data, and denote as $m^{(j)}$ and $s^{(j)}$ respectively.

3. Given a test point $(y, \mathbf{z})$, we are able to make inference on it's probability of being class 1 as follows:

   - For each $(K^{(j)}, \alpha^{(j)})$ pair, generate 100 samples of (β_0, β_1) from $\text{BVN}(\boldsymbol{\beta}_j^*, H^{-1}(\boldsymbol{\beta}_j^*))$, denoted as $\{\beta_0^{(jl)}, \beta_1^{(jl)}\}_{l=1}^{100}$.
   - Given $(K^{(j)}, \alpha^{(j)}, \beta_0^{(jl)}, \beta_1^{(jl)})$, calculate $\mathbf{z}$'s standardized LAGO score, $g_j = \frac{f_j - m^{(j)}}{s^{(j)}}$, where f_j is the LAGO score of $\mathbf{z}$ based on $(K^{(j)}, \alpha^{(j)})$. Then, $\mathbf{z}$'s probability of being class 1, θ_{jl}, is given by
   $$\theta_{jl} = \frac{\exp\{\beta_0^{(jl)} + \beta_1^{(jl)} g_j\}}{1 + \exp\{\beta_0^{(jl)} + \beta_1^{(jl)} g_j\}},$$
   for $j = 1, \ldots, M, l = 1, \ldots, 100$.
   - Calculate the posterior mean of θ by (5.24).

4. Repeat steps 2 and 3 for all $\{(K^{(j)}, \alpha^{(j)})\}_{j=1}^{M}$. We have $N = M \times 100$ posterior samples of θ, i.e., θ_{jl}, for $j = 1, \ldots, M, l = 1, \ldots, 100$. Note that the associated weight for each θ_{jl} is $\frac{1}{100} Pr(K^{(j)}, \alpha^{(j)} | \text{train})$. A 95% posterior credibility interval of θ can be constructed by finding the 2.5th and 97.5th percentiles of those N posterior samples:

   - By sorting θ_{jl} from the smallest to the largest, we get a vector $(\theta_{(1)}, \ldots, \theta_{(N)})$ and the corresponding weight vector $(\omega_{(1)}, \ldots, \omega_{(N)})$.
   - Find the endpoints of the 95% credibility interval q_l and q_u, such that $q_l = \theta_{(m)}, \sum_{i=1}^{m} \omega_{(i)} = 0.025; q_u = \theta_{(n)}, \sum_{i=1}^{n} \omega_{(i)} = 0.975$.

BLAGO has a huge computational advantage over the popular MCMC algorithms. BLAGO evaluates the posterior probabilities over a grid of (K, α) values. The number of distinct

pairs is normally several hundred. The MCMC algorithm, however, normally requests at least thousands or hundreds of thousands of iterations to reach the equilibrium distribution. Within each iteration, both methods need to calculate the LAGO scores. Given the fact that MCMC requires much more iterations, BLAGO is much faster. Moreover, BLAGO is based on a grid of independent (K, α) pairs and thus is very easy to cast in parallel computing. It is hard to apply parallel computing to MCMC algorithms because of the built-in dependence structure between the current state and the previous state.

5.4 A Frequentist Approach

Earlier in this chapter, we saw the need for a version of LAGO capable of producing probabilistic predictions, instead of just a score. This enables comparison with other probabilistic models. If the objective is only to obtain point estimates of predicted probabilities, rather than interval estimates, a fully Bayesian approach may not be necessary. In this section, we outline a maximum likelihood approach to LAGO that yields probabilistic predictions. Suppose $y_i, i = 1, \ldots, n$ is the class label of the ith test data point, and $\hat{p}_i$ is its estimated probability of being class 1, the deviance of $\hat{p}_i, i = 1, \ldots, n$ is defined as

$$-2 \sum_{i=1}^{n} [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]. \tag{5.25}$$

Like BLAGO, the frequentist approach also has four parameters: K and α for fitting LAGO, β_0 and β_1 for the logistic transformation. The optimal model can be chosen by cross-validation. The procedure is almost the same as the 5-fold cross-validation for fitting LAGO, except that in each iteration the resulting LAGO scores are standardized and then transformed to probabilities by fitting a logistic regression. Finally, deviance is calculated based on the fitted probabilites, the best model is the combination of (K, α) that gives the smallest deviance. The detailed procedure is described as follows:

Loop through all possible grid values of $\{(K^{(j)}, \alpha^{(j)})\}_{j=1}^{M}$

1. Keep fold 1 aside for validation and train LAGO with the other four folds. Suppose we get model M1. Apply M1 to fold 1 data to obtain a vector of LAGO scores $\hat{\mathbf{f}}_1$.

2. Repeat step 1 for all the other folds. Now, we have $\hat{\mathbf{f}} = (\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \hat{\mathbf{f}}_3, \hat{\mathbf{f}}_4, \hat{\mathbf{f}}_5)$, a vector of the LAGO scores for all the training observations.

3. Denote the mean and standard deviation of vector $\hat{\mathbf{f}}$ as $m^{(j)}$ and $s^{(j)}$, and save them. Re-scale $\hat{\mathbf{f}}$ to $\hat{\mathbf{g}}$ which has mean 0 and standard deviation 1.

4. Fit a logistic regression with $\hat{\mathbf{g}}$ as the covariate. Suppose the fitted coefficients are $\hat{\beta}_0, \hat{\beta}_1$ respectively, then the estimated probabilities of being class-1 for all the training observations are
$$\hat{\mathbf{p}} = \frac{\exp\{\hat{\beta}_0 + \hat{\beta}_1 \hat{\mathbf{g}}\}}{1 + \exp\{\hat{\beta}_0 + \hat{\beta}_1 \hat{\mathbf{g}}\}}.$$

5. Calculate the deviance of $\hat{\mathbf{p}}$ by equation (5.25).

End loop

The optimal combination $(K^{'}, \alpha^{'})$ is the one that yields the smallest cross-validated deviance. Let $m^{'}$ and $s^{'}$ be the mean and standard deviation of the LAGO scores of the training data given the optimal $(K^{'}, \alpha^{'})$ pair, and $(\hat{\beta}_0^{\prime},\ \hat{\beta}_1^{\prime})$ be the corresponding fitted coefficients of the logistic regression.

Given a test point $\mathbf{z}$, we can calculate its LAGO score $\hat{f}(\mathbf{z})$ by (4.16) based on $(K^{'}, \alpha^{'})$, and its probability of being class 1 can be estimated by

$$\hat{p}_{\mathbf{z}} = \frac{\exp\{\hat{\beta}_0^{\prime} + \hat{\beta}_1^{\prime} \hat{g}(\mathbf{z})\}}{1 + \exp\{\hat{\beta}_0^{\prime} + \hat{\beta}_1^{\prime} \hat{g}(\mathbf{z})\}}, \tag{5.26}$$

where

$$\hat{g}(\mathbf{z}) = \frac{\hat{f}(\mathbf{z}) - m^{'}}{s^{'}}$$

is the standardized LAGO score of $\mathbf{z}$ re-scaled by $m^{'}$ and $s^{'}$.

A 95% confidence interval can be obtained by calculating (p_l, p_u), the 95% confidence interval for $\hat{\beta}_0' + \hat{\beta}_1' \hat{g}(\mathbf{z})$, and then applying a logistic transformation. That is,

$$\begin{aligned} p_l &= \hat{\beta}_0' + \hat{\beta}_1' \hat{g}(\mathbf{z}) - 1.96\sqrt{\mathrm{Var}(\hat{\beta}_0' + \hat{\beta}_1' \hat{g}(\mathbf{z}))}, \\ p_u &= \hat{\beta}_0' + \hat{\beta}_1' \hat{g}(\mathbf{z}) + 1.96\sqrt{\mathrm{Var}(\hat{\beta}_0' + \hat{\beta}_1' \hat{g}(\mathbf{z}))}, \end{aligned}$$

and a 95% confidence interval for $p_{\mathbf{z}}$ is

$$\left(\frac{e^{p_l}}{1+e^{p_l}}, \frac{e^{p_u}}{1+e^{p_u}} \right). \tag{5.27}$$

This approach to calculating a 95% confidence interval for $p_{\mathbf{z}}$ is somewhat naive. By fixing (K, α) at the optimal value given by cross-validation, we are ignoring the uncertainty of the choice of (K, α) pair in the interval (5.27). Evidence will be shown in Section 5.8.1. We refer to this frequentist approach as "frequentist-LAGO".

5.5 Application to the NCI Data

The first part of this section compares the performance of BLAGO with LAGO and other methods in terms of average precision. The output of BLAGO is a probabilistic estimate and LAGO can also produce a probabilistic estimate via frequentist-LAGO. As mentioned in Section 3.3, average precision is a good performance metric in comparing different ranking models; when it comes to comparing probabilistic estimates, cross entropy (or deviance) might be a better choice. The second part of this section compares the performance of BLAGO and LAGO in terms of deviance.

5.5.1 Performance Comparison in Average Precision

As shown in Chapter 4, LAGO using either the Gaussian (LAGO-G) or the triangular (LAGO-T) kernel, and ASVM significantly outperform LAGO using the uniform (LAGO-U) kernel, SVM and KNN. SVM has similar performance to KNN and LAGO-U. Therefore,

we only focus on the comparison of BLAGO with LAGO-G, SVM and ASVM. For all the methods, models are fit on the training set and performance is compared on the test set in terms of average precision. Table 5.1 and Figure 5.1 compare the test-set average precisions of these four competing methods for the NCI data.

Methods	LAGO-G	SVM	ASVM	BLAGO
Split 1	0.2292	0.1926	0.2091	0.2511
Split 2	0.2503	0.2105	0.2055	0.2696
Split 3	0.2554	0.1688	0.2078	0.2511
Split 4	0.2713	0.2063	0.2600	0.2767

Table 5.1: The test-set average precisions on the four random splits of the NCI data using different methods.

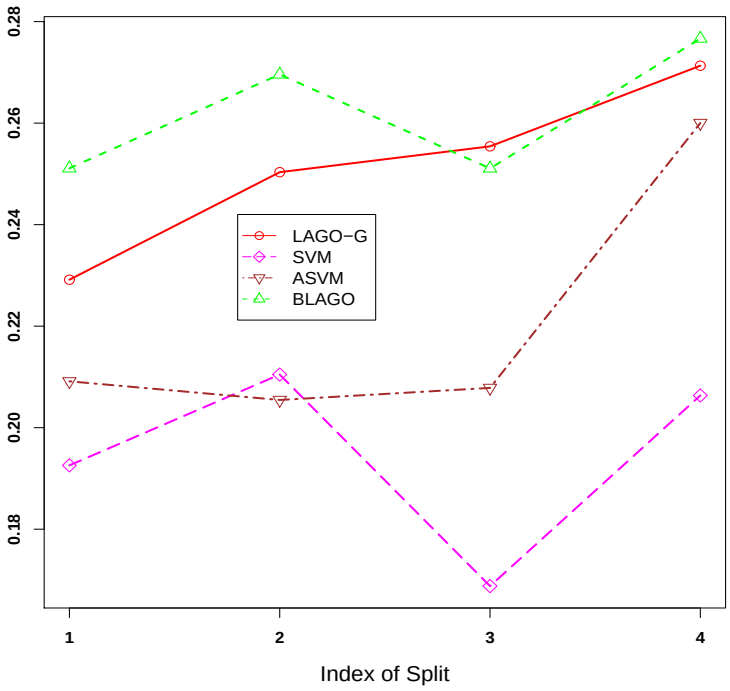

Figure 5.1: The test-set average precisions of different methods evaluated on the four random splits of the NCI data.

Note that results for LAGO-G, SVM and ASVM are copied from Table 4.4 in Section 4.3.1. Figure 5.2 presents their corresponding hit curves on four different test sets (splits).

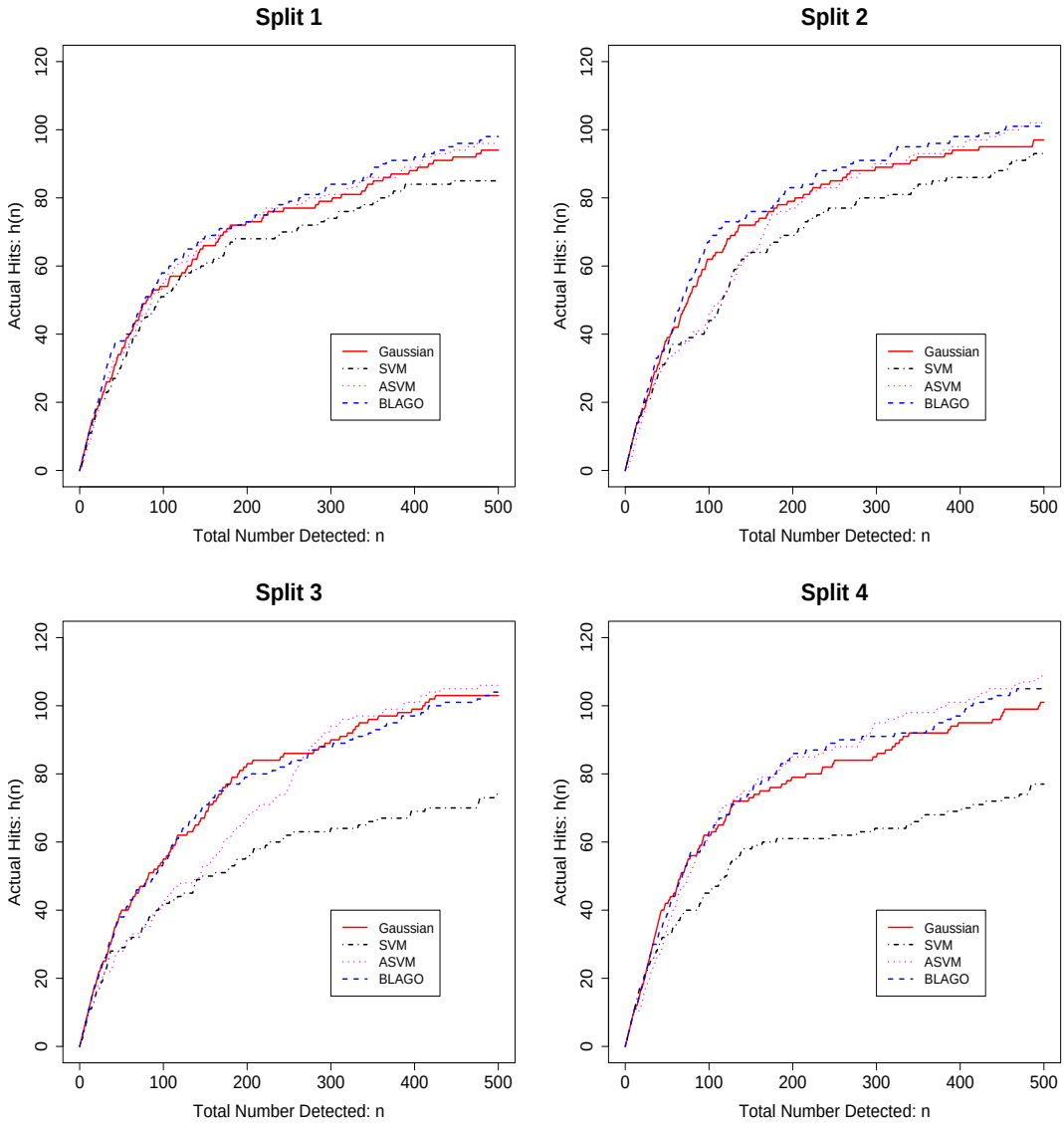

Figure 5.2: Hit curves of different methods for four random splits of the NCI data under criterion of average precision. Upper left: The 1st split; Upper right: The 2nd split; Lower left: The 3rd split; Lower right: The 4th split.

Based on the average precisions on four experiments, ANOVA can be conducted to formally compare the difference among BLAGO, LAGO-G, SVM and ASVM. Let $\mu_G, \mu_S, \mu_A, \mu_B$ be the average performance (in terms of average precision) of LAGO-G, SVM, ASVM, BLAGO respectively. The ANOVA results are given in Table 5.3, the expressions and estimates of the three non-orthogonal contrasts are shown in Table 5.2.

Contrast	Expression	Estimate
Cntr1	$\mu_B - \mu_G$	0.0106
Cntr2	$\mu_A - \mu_S$	0.0261
Cntr3	$(\mu_G + \mu_B)/2 - \mu_A$	0.0362

Table 5.2: Estimated contrasts for the NCI data.

Source	SS($\times 10^{-4}$)	df	MS($\times 10^{-4}$)	F_0	P-Value
Method	112.846	3	37.615	20.574	0.0002
Cntr1	*2.228*	*1*	*2.228*	*1.218*	*0.2983*
Cntr2	*13.590*	*1*	*13.590*	*7.433*	*0.0234*
Cntr3	*34.980*	*1*	*34.980*	*19.132*	*0.0018*
Splits	29.122	3	9.707	5.309	0.0222
Error	16.455	9	1.828		
Total	158.423	15			

Table 5.3: ANOVA analysis of differences among methods for the NCI data.

The main conclusion is (LAGO-G $\sim$ BLAGO) $\succ$ ASVM $\succ$ SVM. Even though the difference between BLAGO and LAGO-G is not significant, BLAGO does beat LAGO-G in three out of those four random splits. Facilitated by Bayesian techniques, BLAGO provides not only a probabilistic estimate but also a more effective ranking than LAGO. Without

the need of the time-consuming cross-validation, BLAGO is computationally twice as fast as LAGO-G.

5.5.2 Performance Comparison in Deviance

In the previous section, we have shown that BLAGO and LAGO-G are not significantly different in terms of average precision. In this section we compare the performances of these two methods using deviance as the criterion.

Recall that the original LAGO score given in (4.16) is not a probability and thus can not be applied to the deviance calculation (5.25) directly. To obtain the deviance for LAGO, we adopt frequentist-LAGO given in Section 5.4. For both methods, models are fit using the training data, and performances are compared on the test set. Figure 5.3 shows the log-posterior of (K, α) given by BLAGO, and Figure 5.4 plots the contour of the cross-validated deviance given by frequentist-LAGO. The contour plots of cross-validated deviance and log-posterior look similar, except that cross-validation selects slightly larger K and α.

Table 5.4 presents the parameter estimates and the test-set deviance for frequentist-LAGO and BLAGO. For frequentist-LAGO, 5-fold cross-validated deviance is used to choose the parameters. For BLAGO, 95% posterior credibility intervals of the parameters are given. The optimal K and α values of frequentist-LAGO generally lie within or close to the boundary of the posterior intervals given by BLAGO. Notice that the posterior probability of BLAGO has almost all mass in small K values, i.e., $K = 2, 3, 4, 5, 6$. A pair-wise t-test shows that there is no significant difference in test-set deviance between frequentist-LAGO and BLAGO (p-value=0.19); however, BLAGO consistently gives smaller deviance than LAGO across all the four random splits.

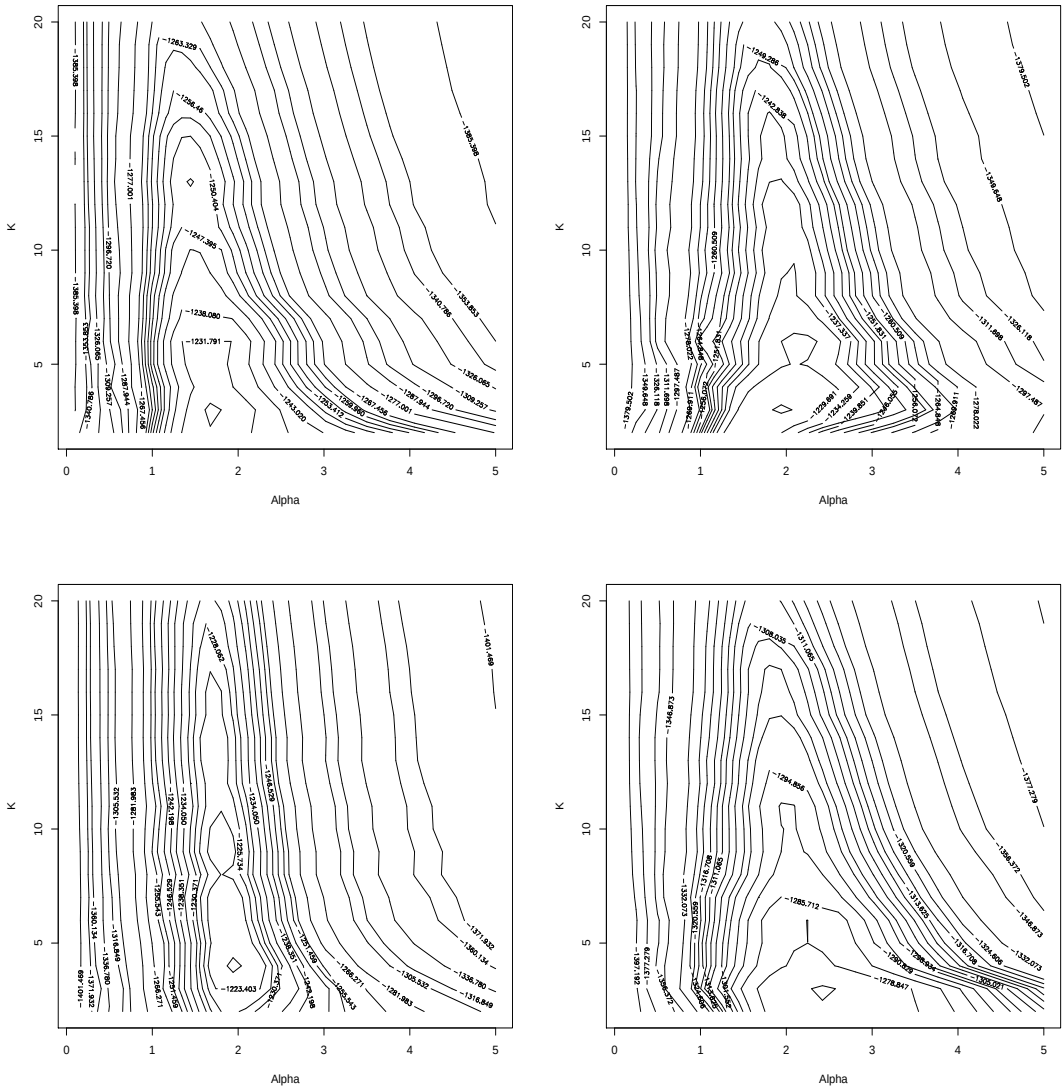

Figure 5.3: Contour plots of log-posterior of the (K, α) pairs for the NCI data. Upper left: The 1st split; Upper right: The 2nd split; Lower left: The 3rd split; Lower right: The 4th split.

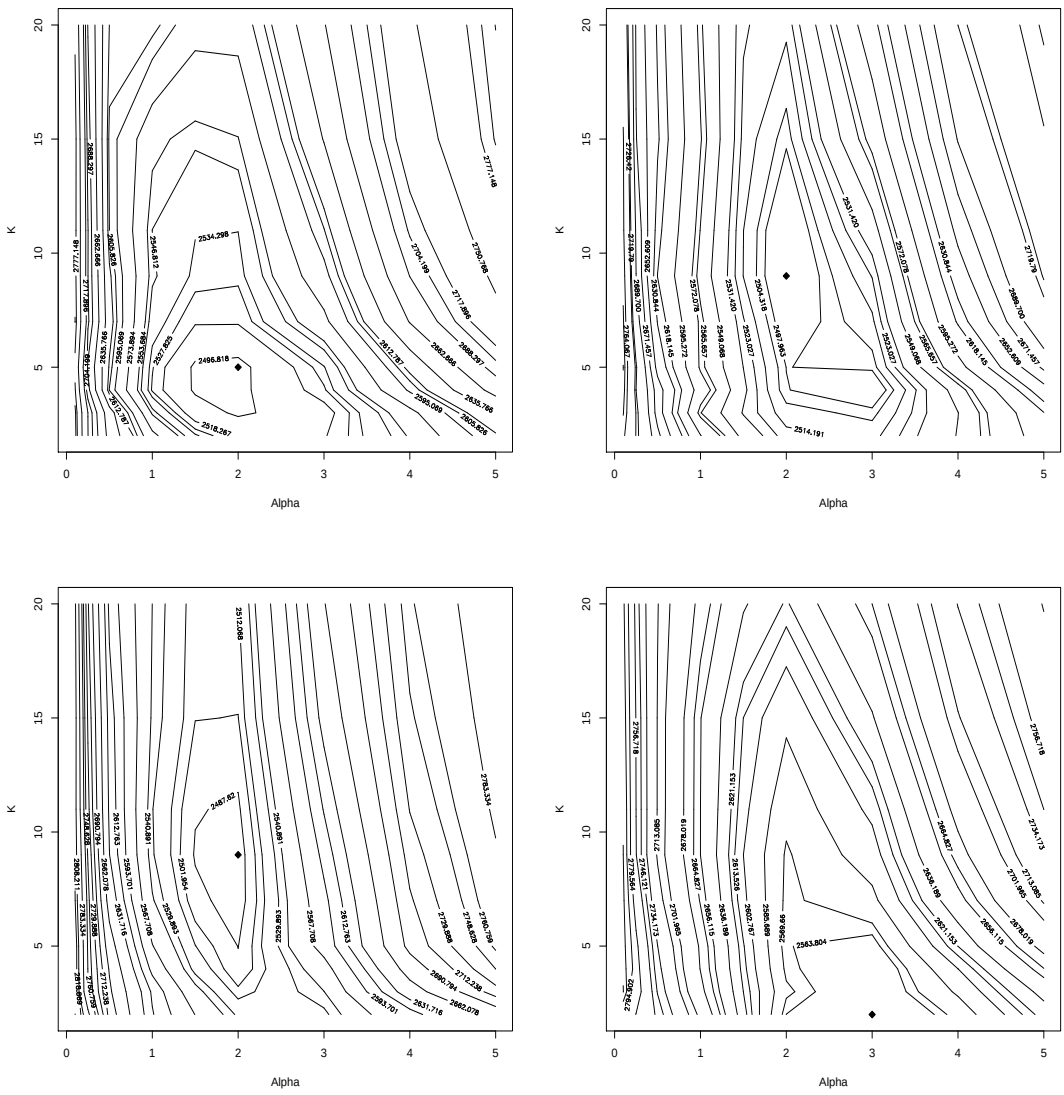

Figure 5.4: Contour plots of deviance of (K, α) by cross-validation on the training sets of the NCI data. Diamonds indicate the optimal choice of the parameters by cross-validation. Upper left: The 1st split; Upper right: The 2nd split; Lower left: The 3rd split; Lower right: The 4th split.

	Frequentist-LAGO			BLAGO		
	K	α	Deviance	K	α	Deviance
Split1	4	2	2466.337	[2, 6]	(1.45, 1.94)	2453.669
Split2	9	2	2497.138	[2, 3]	(1.56, 2.25)	2414.047
Split3	9	2	2459.255	[3, 7]	(1.80, 2.09)	2456.062
Split4	2	3	2419.649	[2, 3]	(2.09, 2.81)	2379.117

Table 5.4: Tuning parameters selected for frequentist-LAGO using cross-validated deviance. The 95% posterior credibility intervals of K and α given by BLAGO.

5.6 Application to the Mysim Data

We apply BLAGO to the Mysim data, the simulated data described in Section 2.3.2. Recall the generating mechanism for the Mysim data (see Figure 5.5): generate 300 class-0 observations uniformly over the whole square, generate 40 class-1 uniformly from mini-square A and another 60 class-1 uniformly over mini-square B. Those 13 points in Figure 5.5 are chosen for evaluating BLAGO's performance in making inference. Squares A and B are two class-1 regions. Points #1 to 7 lie within the class-0 region; points #8 to 10 are in region A, and points #11 to 13 are in region B. Darker dots (points #4 to 9, 11 and 12) refer to those points that are on the boundary of class-1 regions and relatively difficult to predict precisely.

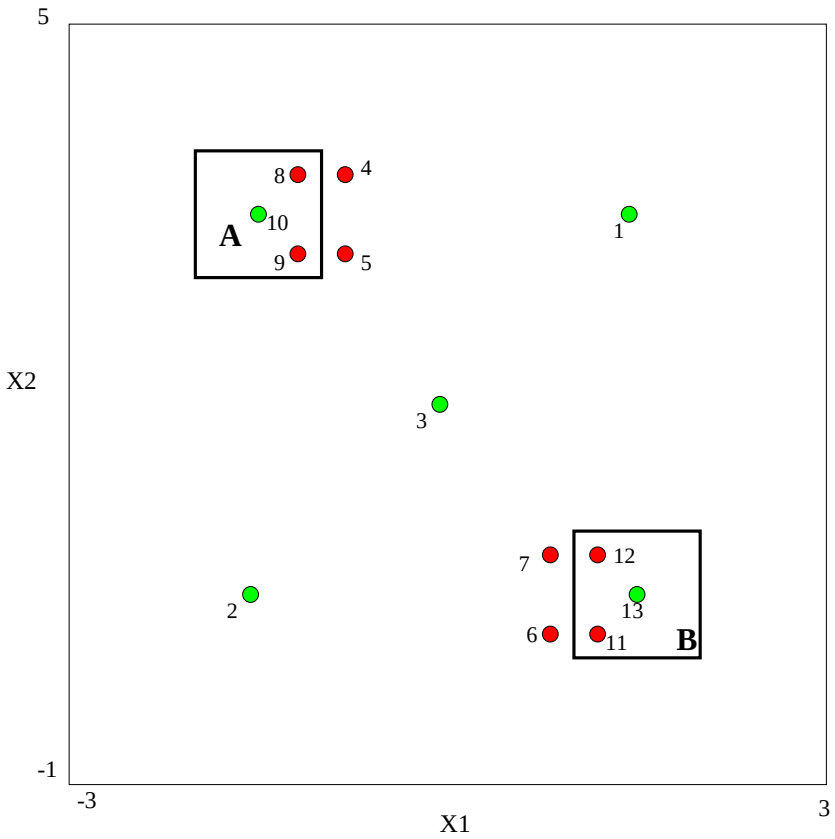

Figure 5.5: Data generating mechanism of the Mysim data and thirteen selected points for coverage analysis.

5.6.1 Performance Comparison in Average Precision

In this simulation study, we apply BLAGO to the same 100 experimental data sets as in Section 4.4.1. Since LAGO-T, LAGO-G, SVM and ASVM are the best methods for the Mysim data, we focus on comparing BLAGO to LAGO-G, SVM and ASVM. In each experiment, models are fit on the training set and performance is compared on the test set in terms of average precision. Both the training and the test sets have 400 observations.

Figure 5.6 presents the boxplots of the 100 test-set average precisions for all the four competing methods. Based on the results of the 100 experiments, ANOVA is conducted

to compare the difference among methods of LAGO-G, SVM, ASVM and BLAGO. The ANOVA results shown in Table 5.5 indicate that there is no significant difference among the methods.

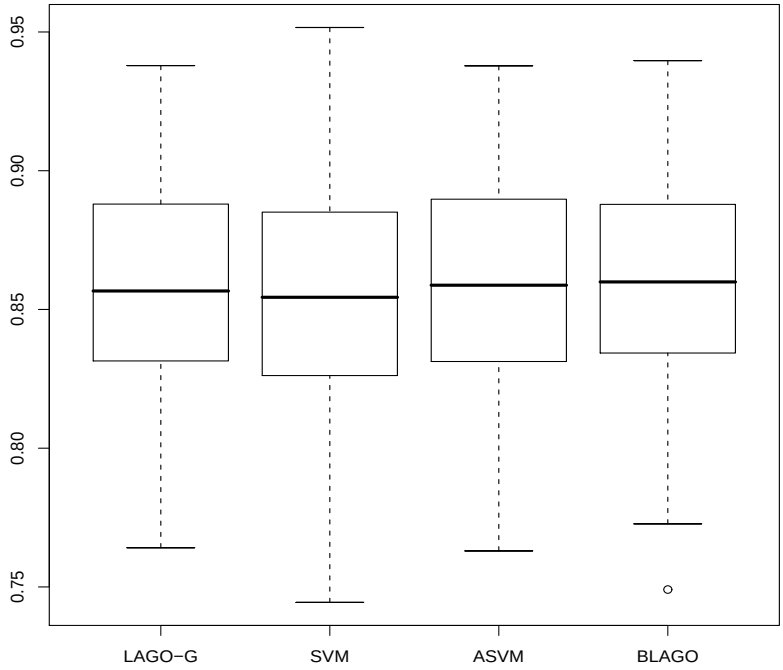

Figure 5.6: The test-set average precisions of different methods evaluated on 100 simulated Mysim data sets.

Source	SS($\times 10^{-4}$)	df	MS($\times 10^{-4}$)	F_0	P-Value
Method	26.075	3	8.692	1.549	0.2019
Replicates	4433.773	99	44.786	7.982	< 0.0001
Error	1666.494	297	5.611		
Total	6126.343	399			

Table 5.5: ANOVA analysis of differences among methods for the Mysim data.

5.6.2 BLAGO Inference

In this section, we evaluate the performance of BLAGO in terms of coverage probability based on those 13 representative points shown in Figure 5.5.

According to the data generating mechanism, it is straightforward to calculate a test point's true probability of being class 1, i.e., 24/29 if the test point is inside region A, 36/41 within region B, and 0 otherwise. The coverage probabilities can be estimated by counting the number of times out of 100 that the 95% posterior credibility intervals contain the true probabilities. Table 5.6 shows that the coverage rates are much lower than their nominal value (95%). Representative points #1 to 7, whose true probabilities are zero, have zero coverage. However, this is not surprising. Since a Gaussian kernel is used, all the predictive probabilities are positive and hence the posterior credibility intervals do not cover zero.

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}
0	0	0	0	0	0	0	28	29	80	65	56	4

Table 5.6: Coverage rate (in %) of the Mysim data over 100 experiments. Points #8 to 10 are in region A and points #11 to 13 are in region B.

Boxplots of posterior means for predictions over 100 experiments are displayed in Figure 5.7, which indicates that BLAGO is doing a reasonable job. First, BLAGO preserves the correct order of the true probabilities of those thirteen representative points. Second, although the true probabilities of being class 1 are all zero, points #4, 5, 6, 7 have a much larger predictive probability and variation than points #1, 2, and 3; this makes sense given the fact that points #4, 5, 6, 7 are on the boundary of the class-1 regions. Third, being the center of region B, point #13 has a much higher predictive probability and smaller variability than boundary points #11 and 12, even though these three have the same true values. The situation is similar for interior point 10 and boundary points #8 and 9 of

region A. It is claimed in Chapter 4 that in general LAGO is a biased estimate of the density ratio f; it seems that BLAGO carries on this bias in estimating the probability. Note that the bias is relatively large for boundary points #4 to 9.

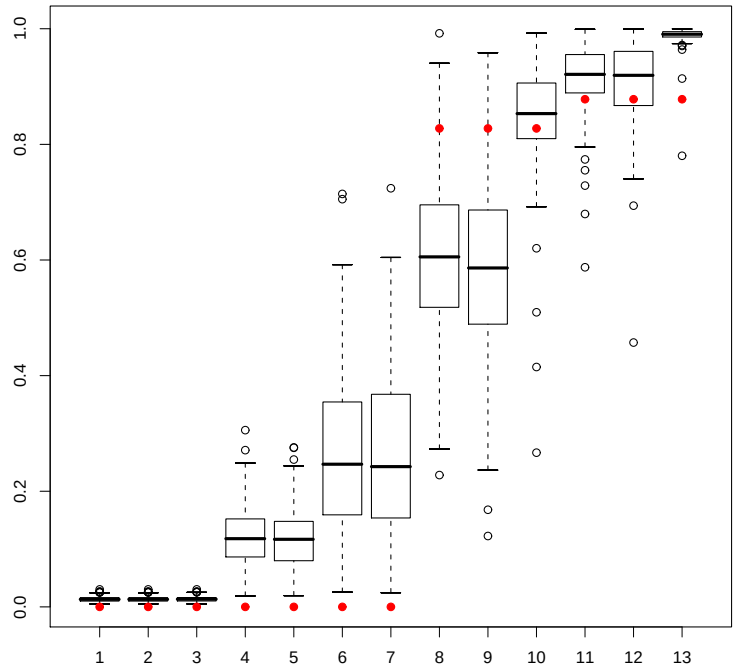

Figure 5.7: Boxplots of the 100 experimental posterior means given by BLAGO for each representative point. Filled dots represent the true probabilities.

It is also interesting to further investigate the performance of BLAGO in a single experiment. Take experiment #70 for example. Figure 5.8 plots the predictions and their 95% posterior credibility intervals for those 13 selected points. Note that experiment #70 is by far NOT the best experiment but a very typical one. Like Figure 5.7, Figure 5.8 shows that BLAGO works well in that it yields a much more precise prediction and a much narrower credibility interval for those points in interior regions where we are more confident in our prediction.

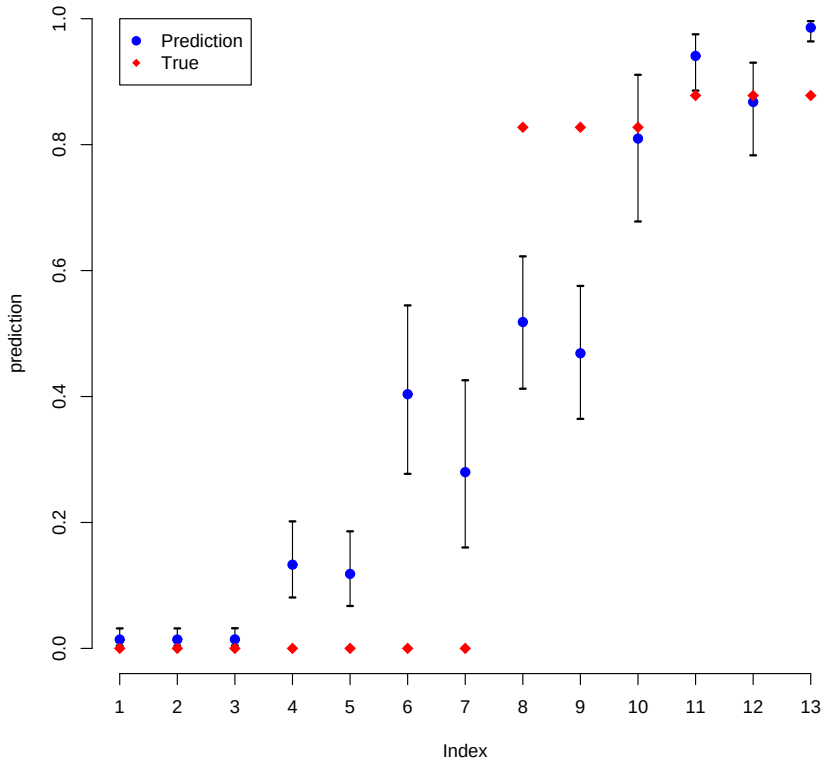

Figure 5.8: Predictions (filled dots) of the 13 representative points and their 95% posterior credibility intervals. Diamonds are the true probabilities.

5.7 Application to the Mysim-LAGO Data

In this section, we apply BLAGO to the Mysim-LAGO data where we generate data from the LAGO model by putting kernel functions with a certain bandwidth at 14 locations (see Figure 4.11 in Section 4.4.2). Similar to the simulation study on the Mysim data in Section 5.6, thirteen representative points are selected for evaluating BLAGO's performance in making statistical inference. Figure 5.9 shows the probability surface of the Mysim-LAGO data given those 14 kernel functions and $(\beta_0, \beta_1) = (-2.8, 45)$; those 13 selected locations are also shown in the same figure.

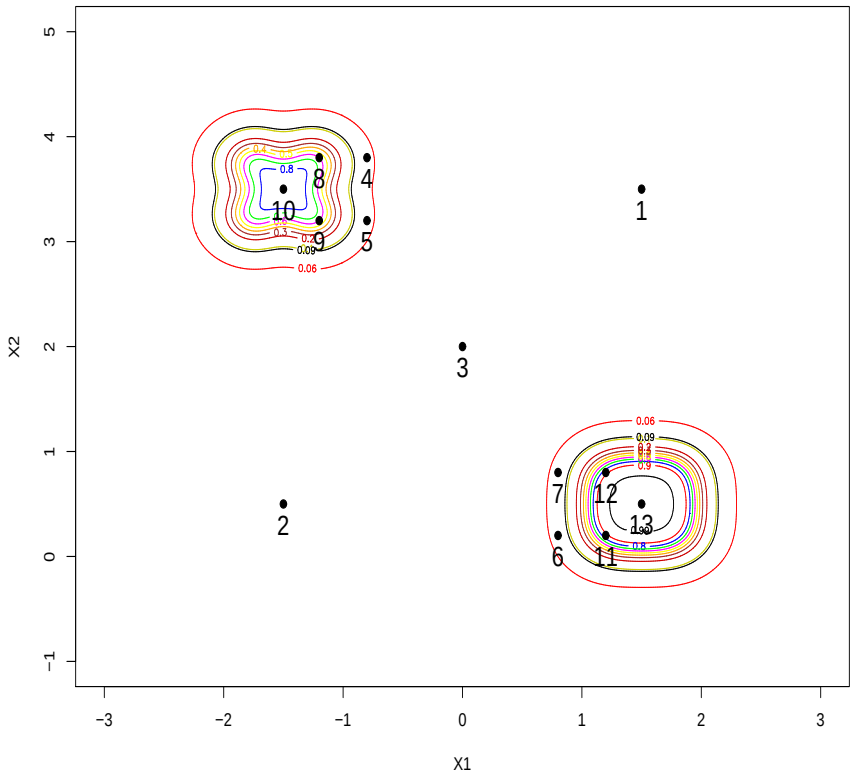

Figure 5.9: Probability surface of the Mysim-LAGO data given $(\beta_0, \beta_1) = (-2.8, 45)$ and the locations of 13 representative points.

5.7.1 Performance comparison in Average Precision

Section 4.4.2 concluded that LAGO-G, LAGO-T and KNN are among the best models for the Mysim-LAGO data in terms of average precision. BLAGO is applied to the same 100 experimental data sets as in Section 4.4.2. We focus on comparing BLAGO to LAGO-G, KNN, SVM, ASVM in terms of average precision using ANOVA. In each experiment, models are fit on the training set and performance is compared on the test set. Both the training and test set have 400 observations.

The boxplots of the 100 test-set average precisions for all the four compared methods are presented in Figure 5.10. Five nonorthogonal contrasts are constructed to further

compare the discrepancy between methods. The ANOVA results are shown in Table 5.8; the expressions and estimates of the contrasts are shown in Table 5.7. The main conclusion is (LAGO-G $\sim$ BLAGO $\sim$ KNN) $\succ$ (ASVM $\sim$ SVM).

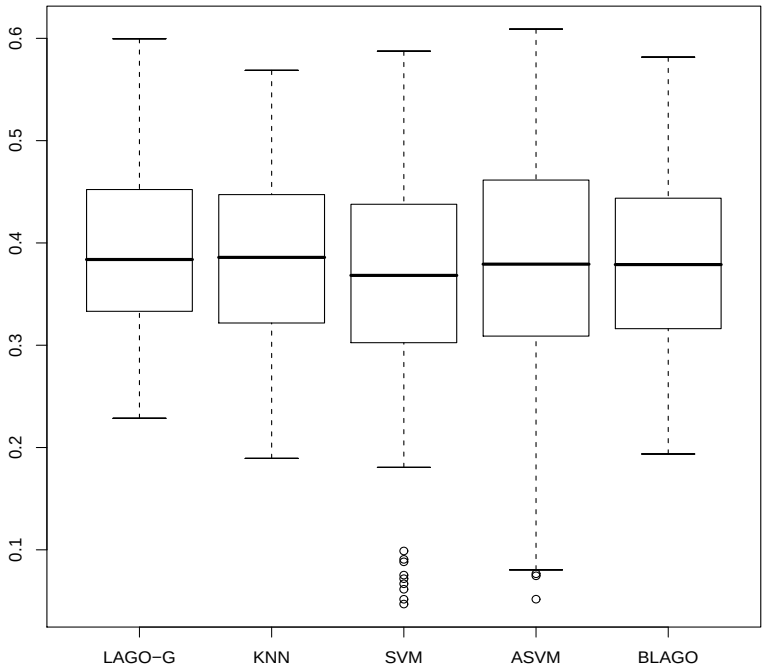

Figure 5.10: The test-set average precisions of different methods evaluated on 100 simulated Mysim-LAGO data that are generated from LAGO.

Contrast	Expression	Estimate
Cntr1	$(\mu_G + \mu_B + \mu_K)/3 - (\mu_S + \mu_A)/2$	0.0226
Cntr2	$\mu_A - \mu_S$	0.0116
Cntr3	$\mu_G - \mu_B$	0.0092
Cntr4	$\mu_K - \mu_B$	0.0010
Cntr5	$\mu_G - \mu_K$	0.0083

Table 5.7: Estimated contrasts for the Mysim-LAGO data.

Source	SS($\times 10^{-2}$)	df	MS($\times 10^{-2}$)	F_0	P-Value
Method	73.275	5	18.319	4.964	0.0006
Cntr1	*61.387*	*1*	*61.387*	*16.633*	*0.0001*
Cntr2	*6.734*	*1*	*6.734*	*1.825*	*0.1775*
Cntr3	*4.265*	*1*	*4.265*	*1.156*	*0.2830*
Cntr4	*0.047*	*1*	*0.047*	*0.013*	*0.9104*
Cntr5	*3.419*	*1*	*3.419*	*0.926*	*0.3364*
Replicate	3604.306	99	36.407	9.865	< 0.0001
Error	1461.469	495	3.691		
Total	5139.050	599			

Table 5.8: ANOVA analysis of differences among methods for the Mysim-LAGO data.

5.7.2 BLAGO Inference

To explore BLAGO's capability of making statistical inference, a more complete study is conducted for the Mysim-LAGO data by comparing BLAGO with frequentist-LAGO and Bayesian KNN (BKNN, see Section 5.2.2 for details) in coverage rate, bias, mean square error, and average length of the confidence (or credibility) intervals over 100 experiments. In each experiment, models are fit on the training data and predictions are made for the 13 pre-chosen points. We have tried two different sample sizes for the training set, i.e., $n = 400$ and 4000.

Boxplots of 100 predictions of the 13 interesting locations based on BLAGO, frequentist-LAGO, BKNN are given in Figure 5.11. The boxplots indicates that increasing the sample size dramatically reduces the biases and the variations of the predictions across those 100 experiments. More discussion on the bias and the variation will be given later this section.

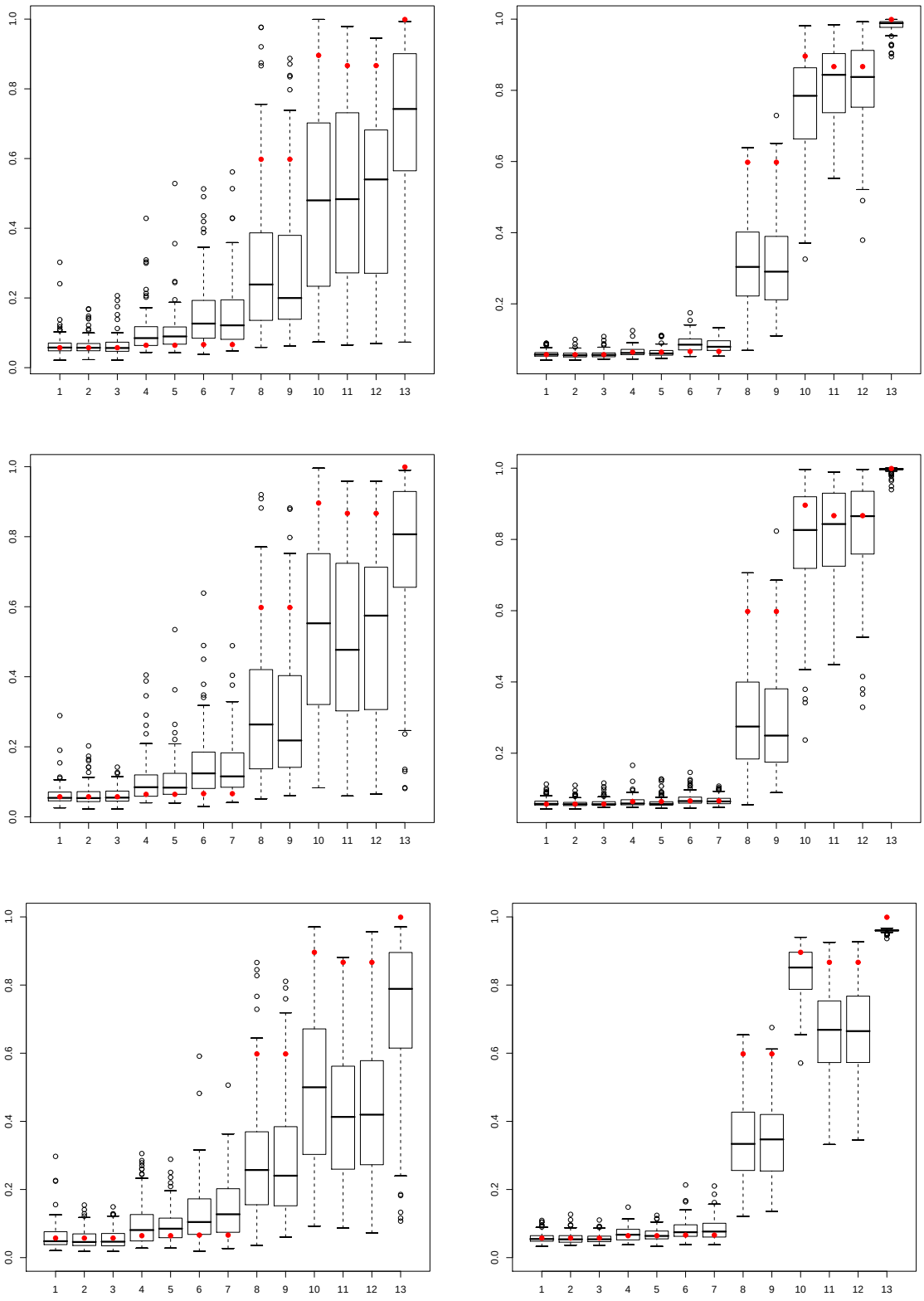

Figure 5.11: Boxplots of predictions of 100 experiments for the 13 representative points of the Mysim-LAGO data. Filled dots represent the true probabilities. Upper: BLAGO; Middle: frequentist-LAGO; Lower: Bayesian KNN. Left: $n = 400$, right: $n = 4000$.

The number of times out of those 100 experiments that the posterior intervals cover the true probabilities (Coverage) of BLAGO, frequentist-LAGO and BKNN for each of those 13 selected points are shown in Table 5.9. The bias, standard deviation (Std), mean square error (MSE), and the average lengths of the intervals of the three methods are presented in the same table as well.

Table 5.9 indicates that the coverage probability is far less than 0.95 no matter which method and which sample size is used, especially for points 6 to 10 and 13. The reasons for this low coverage rate are mainly twofold:

- The likelihood function given in equations (5.14) is not a full likelihood function but a so-called pseudo-likelihood, which is in a similar form to the one widely used in modeling social networks. Strauss and Ikeda (1990) proposed using the maximum pseudo-likelihood estimate (MPLE) as an attractive alternative to the maximum likelihood estimate for networks where the full likelihood function is intractable due to the complex dependence structure of the networks. Not much is known about the behavior of MPLE. Some researchers argue that inference based on pseudo-likelihood might be problematic since it ignores at least part of the dependence structure of the data. Both Wasserman and Robins (2005) and Snijders (2002) pointed out that MPLE might be substantially biased and the standard deviations of the parameters are generally underestimated. These arguments are partly supported by the findings of BLAGO. Recall that LAGO estimator (4.16) is constructed by putting kernel functions at and only at class-1 observations. This special way of building basis functions totally ignores the randomness of the response variables y_i. Without taking into account this randomness, BLAGO, whose inference is made based on the pseudo-likelihood function (5.14), inevitably underestimates the variability of the data. The Bayesian KNN proposed by Holmes and Adams (2002) also uses the pseudo-likelihood and hence suffers from this problem as well. Some evidence will be shown in Section 5.8.1.

	n=400					n=4000				
	Coverage	Bias	Std	MSE	Length	Coverage	Bias	Std	MSE	Length
	84	0.00791	0.0365	0.00140	0.0642	75	0.00241	0.0101	0.00011	0.0166
	85	0.00655	0.0277	0.00081	0.0624	64	0.00038	0.0095	0.00009	0.0159
	85	0.00678	0.0315	0.00104	0.0614	65	0.00163	0.0109	0.00012	0.0160
	71	0.03715	0.0620	0.00522	0.1125	56	0.00139	0.0130	0.00017	0.0184
	75	0.04045	0.0640	0.00574	0.1140	52	0.00007	0.0133	0.00018	0.0180
	57	0.09026	0.0986	0.01787	0.1946	42	0.02197	0.0237	0.00105	0.0396
BLAGO	56	0.08810	0.0965	0.01708	0.1862	51	0.01813	0.0189	0.00069	0.0389
	23	-0.29755	0.2177	0.13591	0.2857	6	-0.28196	0.1259	0.09535	0.1207
	24	-0.31050	0.2071	0.13931	0.2991	10	-0.27958	0.1332	0.09592	0.1257
	36	-0.40884	0.2640	0.23686	0.4333	30	-0.14510	0.1486	0.04314	0.1637
	41	-0.36391	0.2600	0.20003	0.4087	46	-0.05696	0.1162	0.01674	0.1372
	39	-0.36475	0.2462	0.19366	0.4373	45	-0.04529	0.1141	0.01506	0.1361
	10	-0.31563	0.2585	0.16645	0.4288	32	-0.01792	0.0206	0.00075	0.0351
	81	0.00833	0.0444	0.00204	0.0568	69	0.00273	0.0109	0.00013	0.0155
	75	0.00861	0.0382	0.00153	0.0552	63	-0.00004	0.0096	0.00009	0.0150
	78	0.00551	0.0293	0.00089	0.0530	64	0.00145	0.0104	0.00011	0.0152
	55	0.04096	0.0655	0.00597	0.0805	51	0.00083	0.0127	0.00016	0.0163
	54	0.04435	0.0721	0.00717	0.0830	49	0.00008	0.0144	0.00021	0.0162
	28	0.08688	0.0957	0.01671	0.1144	29	0.02087	0.0238	0.00100	0.0205
Freq	31	0.07795	0.0787	0.01227	0.1087	35	0.01804	0.0209	0.00076	0.0200
	22	-0.29127	0.1984	0.12419	0.2301	6	-0.28016	0.1261	0.09440	0.0907
	19	-0.30469	0.1958	0.13115	0.2212	8	-0.28193	0.1303	0.09645	0.0894
	24	-0.39033	0.2663	0.22330	0.3206	27	-0.14013	0.1436	0.04027	0.1333
	26	-0.35479	0.2496	0.18818	0.3371	37	-0.05853	0.1226	0.01846	0.1218
	22	-0.35069	0.2385	0.17986	0.3478	41	-0.04629	0.1215	0.01690	0.1182
	4	-0.25946	0.2320	0.12114	0.3505	10	-0.01583	0.0167	0.00053	0.0240
	76	0.00435	0.0418	0.00177	0.0655	56	0.00131	0.0158	0.00025	0.0223
	69	-0.00252	0.0274	0.00076	0.0616	49	-0.00017	0.0169	0.00029	0.0222
	57	-0.00127	0.0291	0.00085	0.0606	53	-0.00105	0.0141	0.00020	0.0205
	62	0.03699	0.0680	0.00599	0.1120	47	0.00454	0.0199	0.00042	0.0277
	70	0.03217	0.0528	0.00383	0.1016	58	0.00295	0.0182	0.00034	0.0285
	49	0.06471	0.0939	0.01300	0.1374	46	0.01640	0.0288	0.00110	0.0379
BKNN	41	0.07954	0.0929	0.01495	0.1509	51	0.01720	0.0321	0.00132	0.0397
	20	-0.30465	0.1826	0.12616	0.2547	9	-0.24994	0.1208	0.07706	0.1231
	17	-0.30926	0.1798	0.12799	0.2402	11	-0.25241	0.1139	0.07669	0.1270
	22	-0.40152	0.2291	0.21369	0.4017	36	-0.06044	0.0712	0.00872	0.0912
	9	-0.44599	0.1997	0.23879	0.2815	13	-0.20114	0.1251	0.05610	0.1356
	14	-0.42418	0.2145	0.22593	0.2917	10	-0.19561	0.1234	0.05350	0.1412
	0	-0.27607	0.2258	0.12720	0.4171	0	-0.03972	0.0046	0.00160	0.0151

Table 5.9: Coverage, bias, standard deviation, mean square error, average length of the intervals for the 13 representative points calculated by BLAGO, frequentist-LAGO, and BKNN over 100 experiments. The sample sizes of the training data are $n = 400$ and $n = 4000$.

In practice, a 95% confidence interval of an estimator $\hat{\theta}$ is normally calculated as $[\hat{\theta} - 2 \times \text{std}(\hat{\theta}), \hat{\theta} + 2 \times \text{std}(\hat{\theta})]$, where $\text{std}(\hat{\theta})$ is the standard deviation of $\hat{\theta}$. This practical principle suggests that the length of a 95% confidence interval should be roughly $4 \times \text{std}(\hat{\theta})$ if it honestly captures the variability. Therefore, $4 \times \text{std}(\hat{\theta})$ provides us a good benchmark to investigate the intervals given by BLAGO, frequentist-LAGO and BKNN are of proper length in capturing model uncertainty. Given that 100 independent experiments have been run, each test point has 100 predicted probabilities and 100 95% posterior credibility intervals. These 100 predicted probabilities can provide an estimate of $\text{std}(\hat{\theta})$, which can be used in a comparison with the width of a 95% posterior credibility interval. Any case in which the average length of those 100 intervals is smaller than $4 \times \text{std}(\hat{\theta})$ gives evidence that the method underestimates variability.

Figure 5.12 compares the average length of the 100 credibility/confidence intervals with $4 \times \text{std}(\hat{\theta})$, and it shows that the average length of intervals is generally smaller than $4 \times \text{std}(\hat{\theta})$ for all three approaches. This indicates that all these three methods underestimate the variability of the models, and hence result in shorter intervals and lower coverage rates than they should be.

Although both BLAGO and frequentist-LAGO underestimate the variability of their own predictions, Table 5.9 indicates that the average lengths of the credibility intervals of BLAGO are larger than the average lengths of the confidence intervals of frequentist-LAGO. This is due to the fact that frequentist-LAGO does not take into account the variability coming from the parameters K and α, while BLAGO does by Bayesian model averaging.

- Pseudo-likelihood is not the only factor that affects the coverage rate. The bias of the predictive probabilities obtained by BLAGO, frequentist-LAGO and BKNN are shown in Figure 5.13, which implies that all of the three approaches generally overestimate those selected points belonging to the class-0 region, while underestimate

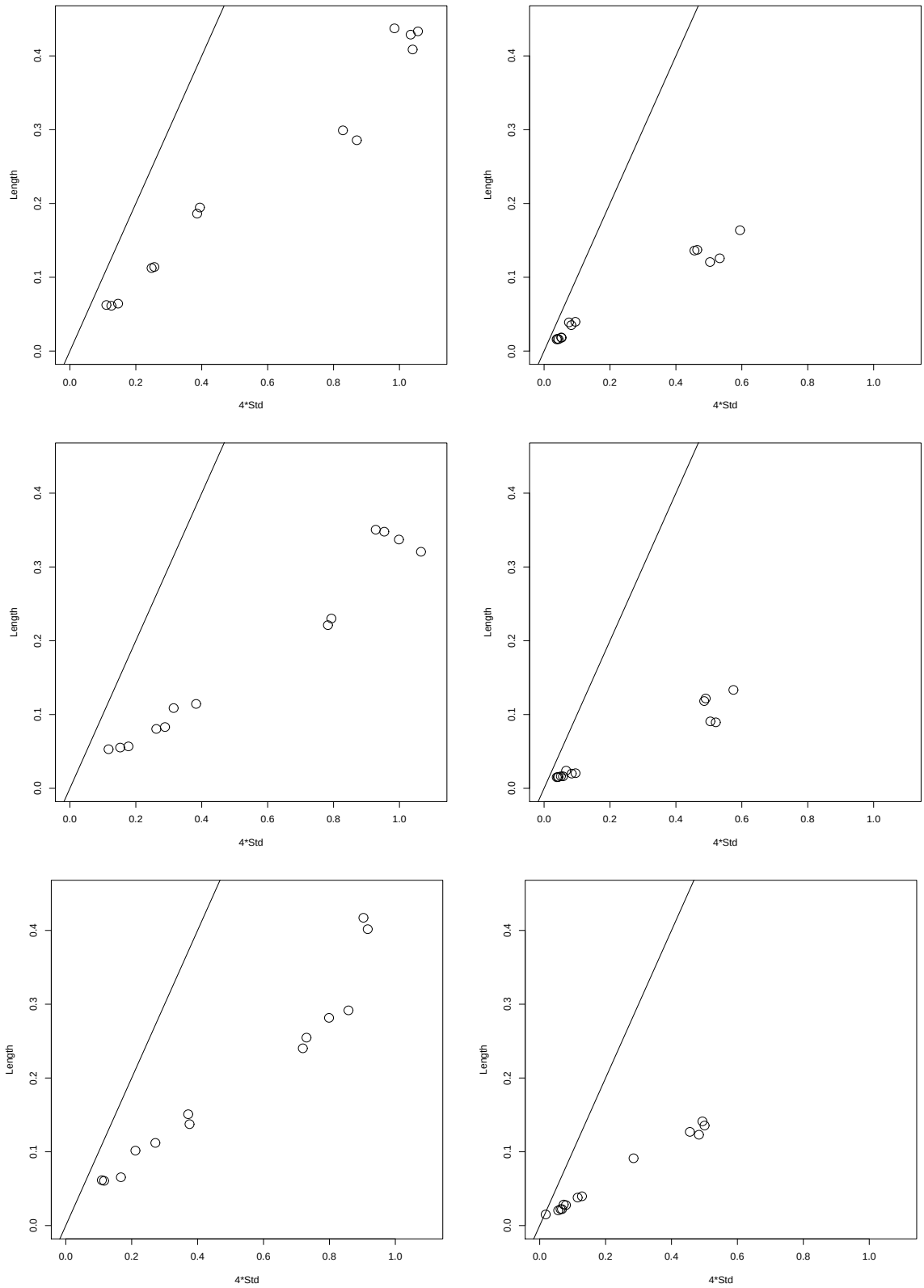

Figure 5.12: Comparison of the average length of the 100 credibility intervals and four times the standard deviation of the 100 predicted probabilities given by BLAGO (upper), frequentist-LAGO (middle) and BKNN (lower) under different sample sizes $n = 400$ (left) and $n = 4000$ (right). All comparisons use the Mysim-LAGO data.

those belonging to the class-1 region. Increasing sample size does reduce the bias; however, the biases of points #8 and #9 are insensitive to the change of sample size. When the sample size is large, the predicted probabilities given by BLAGO and frequentist-LAGO are almost identical. BKNN works significantly worse than BLAGO and frequentist-LAGO at locations #11 and #12.

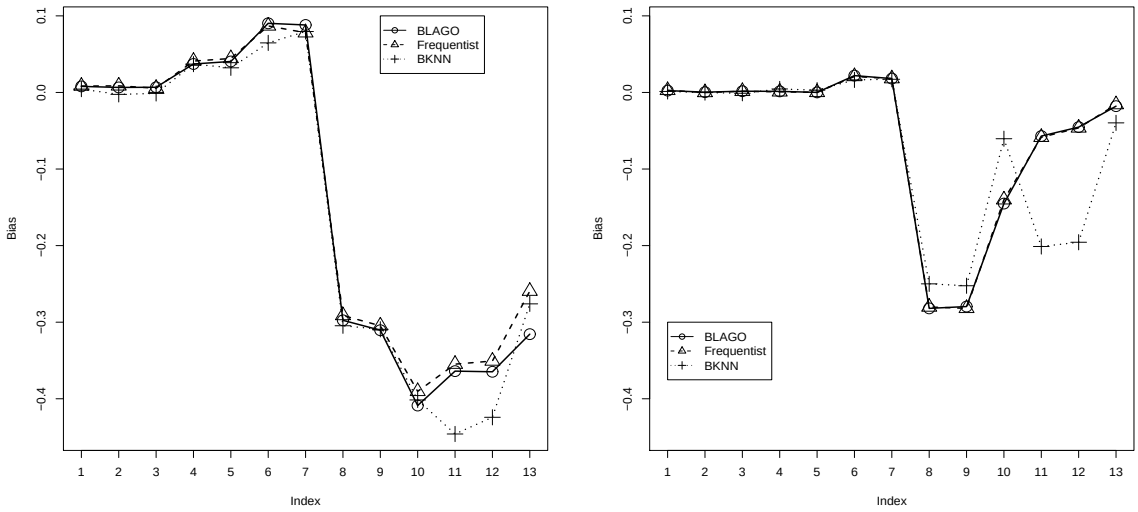

Figure 5.13: Bias of the predicted probabilities obtained by BLAGO (solid line), frequentist-LAGO (slashed line) and BKNN (dotted line) under sample size $n = 400$ (left) and $n = 4000$ (right). Based on the Mysim-LAGO data.

It is well known that kernel density estimator (KDE) is biased. LAGO is related to KDE in that it estimates p_1, the density of class 1, by an adaptive bandwidth KDE. Bias might be introduced when estimating p_1 and carried over to BLAGO. Some authors point out that the ordinary bootstrap confidence bands for the nonparametric regression and density estimation generally have under-coverage rate. In order to obtain an asymptotically correct coverage rate, Härdle & Marron (1991) and Eubank & Speckma (1993) propose two different bias-corrected confidence bands; Hall & Owen (1993) and Chen (1996) suggest using empirical likelihood to construct confidence

bands in problems of nonparametric density estimation; Galindo *et al.* (2001) applied the idea of estimating equations to construct bootstrap confidence intervals for four generalized local polynomial nonparametric regression models. Bias is another main reason causing the low coverage rate of BLAGO. We might be able to improve the coverage rate considerably if we could find a way to correct the bias.

As a result, the biased prediction caused by LAGO and the underestimated standard deviations caused by the pseudo-likelihood function altogether make the coverage rates of the 95% credibility intervals lower than the nominal level. To have a better insight of to what extent BLAGO underestimates the variability, I will compare the intervals given by BLAGO, frequentist-LAGO and BKNN with the bootstrap confidence intervals in terms of coverage rate and average length of intervals in Section 5.8.1.

We can also evaluate the performance of BLAGO using the results based on one experiment, say experiment #70. Again, experiment #70 is not the one that yields the best results; it is used only for illustration. Figure 5.14 plots the predictions and their 95% posterior credibility intervals for those 13 selected points. As shown in Figure 5.14, the bias of BLAGO is dramatically corrected by increasing the sample size from 400 to 4000, except for points #8 and #9, which are two "difficult" points on the boundary of the class-1 region and surrounded by quite a few class-0 observations.

As mentioned before, the credibility intervals given by BLAGO (when $n = 4000$) seem to be too narrow. However, they do provide some information about the data:

- The predictions are quite close to the true values except for points #8 and #9.
- Even though the credibility intervals are too narrow, they do tell us some uncertainty of the prediction. The intervals are relatively narrower for points in a more pure region, e.g., points #1 to 5, and points #11 and 13.

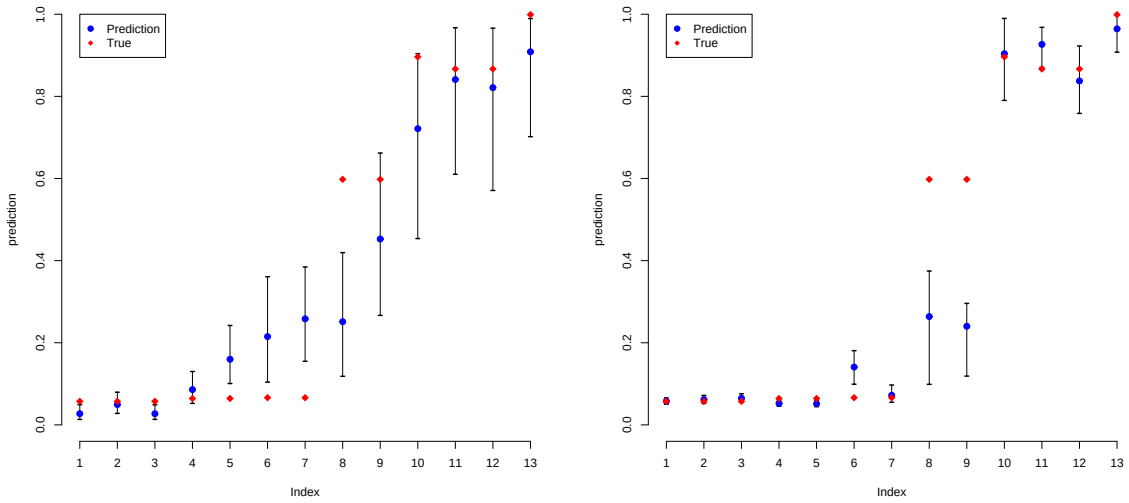

Figure 5.14: Predictions (filled dots) of the 13 representative points and their 95% posterior credibility intervals. Diamonds are the true probabilities. Left: $n = 400$; right: $n = 4000$.

5.8 Discussion

This section discusses three special topics of BLAGO. The first topic is comparing the 95% posterior (or confidence) intervals given by BLAGO and BKNN (or frequentist-LAGO) with the bootstrap confidence intervals. As mentioned in Section 5.7.2, BLAGO tends to underestimate the variation of the prediction, and hence yields a too narrow credibility interval and a too low coverage rate. Bootstrap confidence intervals might be able to account for all sources of variation of the prediction and provide us a benchmark of the amount of variation BLAGO under-counts.

Since BLAGO results in a probabilistic estimate of a point's probability of being class 1 and the uncertainty of this estimate as well, one intuitive question is whether accounting for the uncertainty improves the ranking performance. This question will be addressed in the second part of this section via comparing the average precisions calculated by using

the posterior mean and the 2.5th posterior percentile as the ranking score.

In order to cast LAGO into a Bayesian framework, MCMC methods were also attempted. The last part of this section will discuss briefly the MCMC approaches we have tried.

5.8.1 Bootstrap Confidence Interval

One important application of bootstrap techniques (Efron and Tibshirani, 1993) is to construct confidence intervals.

Recall that the quantity of interest is θ, a data point's probability of being class 1. Considering that the end points of both the BLAGO credibility interval and the confidence interval of frequentist-LAGO are obtained by finding the 2.5th and 97.5th percentiles, we decide to adopt the percentile method to construct the bootstrap confidence interval for θ. The detailed procedure to construct bootstrap confidence intervals for those 13 representative points of the Mysim-LAGO data set is as follows:

1. Generate one set of training data $\mathcal{D}$ and a set of validation data $\mathcal{V}$ using the same mechanism given in Section 4.4.2.

2. Given the training data $\mathcal{D}$, generate 500 bootstrap sampling with replacement from $\mathcal{D}$, denoted as $\mathcal{D}_1, \ldots, \mathcal{D}_{500}$.

3. For each bootstrap sample $\mathcal{D}_b$, fit LAGO over a grid of (K, α) pairs and pick the best (K, α) combination, $(K^{'}, \alpha^{'})$, which gives the smallest deviance evaluated on the validation set. Obtain the coefficients of the logistic transformation $(\beta_0^{'}, \beta_1^{'})$ by fitting a logistic regression on the standardized LAGO scores. Given $(K^{'}, \alpha^{'}, \beta_0^{'}, \beta_1^{'})$, calculate the probabilities of being class 1 for the 13 points, denote as $\boldsymbol{\theta}_b = (\theta_{b1}, \ldots, \theta_{b13}), b = 1, \ldots, 500$.

4. The end points of the 95% confidence interval for the jth point can be obtained by

finding the 2.5th and 97.5th percentiles of the prediction vector $(\theta_{1j}, \ldots, \theta_{500j}), j = 1, \ldots, 13$.

5. Repeat steps 1 to 4 one hundred times to calculate the coverage rate for every representative point by computing the proportion of times that those 100 confidence intervals cover the true probability. The average length of the 100 confidence intervals can be calculated as well.

Table 5.10 reports the coverage rates and the average lengths of the intervals given by the bootstrap method (Boot), frequentist-LAGO (Freq), BLAGO and BKNN. The standard deviations for the selected points calculated by frequentist-LAGO, BLAGO and BKNN across 100 trials are also given in Table 5.10.

Coverage Rate (%)				Average Length				Standard Deviation		
Boot	Freq	BLAGO	BKNN	Boot	Freq	BLAGO	BKNN	Freq	BLAGO	BKNN
67	81	84	76	0.1154	0.0568	0.0642	0.0655	0.0444	0.0365	0.0418
65	75	85	69	0.1225	0.0552	0.0624	0.0616	0.0382	0.0277	0.0274
63	78	85	57	0.1100	0.0530	0.0614	0.0606	0.0293	0.0315	0.0291
100	55	71	62	0.2911	0.0805	0.1125	0.1120	0.0655	0.0620	0.0680
99	54	75	70	0.3003	0.0830	0.1140	0.1016	0.0721	0.0640	0.0528
94	28	57	49	0.4241	0.1144	0.1946	0.1374	0.0957	0.0986	0.0939
95	31	56	41	0.4133	0.1087	0.1862	0.1509	0.0787	0.0965	0.0929
74	22	23	20	0.7064	0.2301	0.2857	0.2547	0.1984	0.2177	0.1826
74	19	24	17	0.6937	0.2212	0.2991	0.2402	0.1958	0.2071	0.1798
74	24	36	22	0.8402	0.3206	0.4333	0.4017	0.2663	0.2640	0.2291
74	26	41	9	0.7864	0.3371	0.4087	0.2815	0.2496	0.2600	0.1997
78	22	39	14	0.8071	0.3478	0.4373	0.2917	0.2385	0.2462	0.2145
44	4	10	0	0.7829	0.3505	0.4288	0.4171	0.2320	0.2585	0.2258

Table 5.10: Coverage rate, average length of the intervals for each location calculated by the bootstrap method, frequentist-LAGO, BLAGO and BKNN over 100 experiments.

Although the coverage rates given by the bootstrap method are not equal to 95%, they are in general closer to the nominal value than the rates given by BLAGO, frequentist-LAGO and BKNN. The inaccuracy of the coverage rate of the bootstrap intervals is due to the bias of the prediction.

Figure 5.15 plots the average lengths of 100 intervals against 4 times the standard deviations of the corresponding predictions for all the 13 selected locations given by BLAGO and bootstrap method. Two reference lines both of which go through the origin are also displayed, one with slope 1 and the other with slope 1/2. It is easy to see that the average length of the bootstrap intervals is roughly four times the standard deviation of the prediction, which indicates that bootstrap interval is relatively honest in representing the uncertainty of the prediction. However, the bootstrap method is computationally expensive. The computational cost of bootstrap is more than 500 times as much as that of BLAGO. Applying the bootstrap method to a large data set might be infeasible.

One interesting finding is that the average length of the BLAGO credibility interval is roughly twice as large as the corresponding standard deviation. If we continue to regard four times the standard deviation as the "gold standard", then the Bayesian posterior interval given by BLAGO is about half as long as it should be. This consistent shift suggests that BLAGO is able to capture the uncertainty of the prediction to some extent and we can still rely on it to assess the relative uncertainty of the prediction, although it underestimates overall uncertainty.

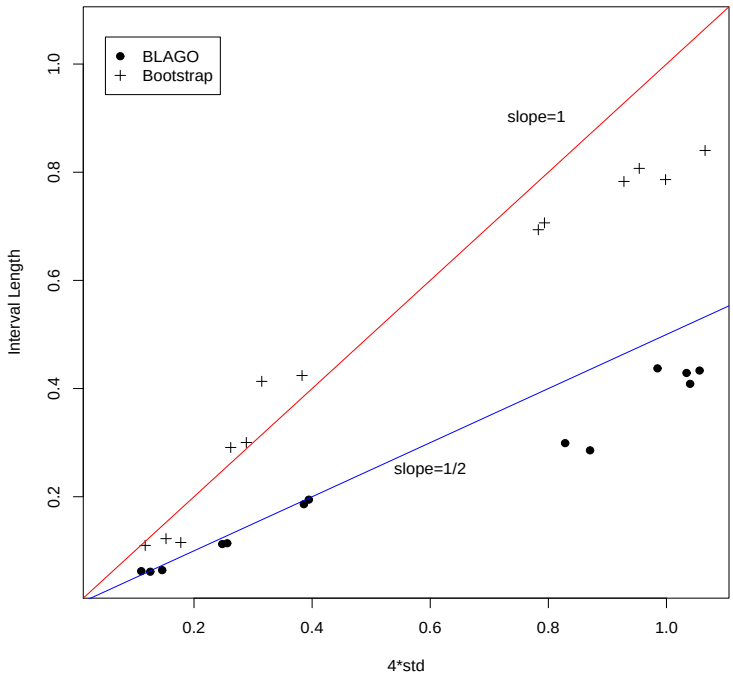

Figure 5.15: Average lengths of 100 intervals versus 4 times the standard deviations of the corresponding predictions for the 13 representative locations given by BLAGO and bootstrap method. Both the two reference lines go through the origin, one with slope 1 and the other with slope 1/2.

5.8.2 Using Lower Confidence Bound for Ranking

To answer the intuitive question whether accounting for the uncertainty improves the ranking performance of BLAGO, we calculate the average precision using the lower posterior confidence bound (i.e., the 2.5th posterior percentile) as the ranking score, and compare this average precision with the one calculated by using the posterior mean. The 2.5th posterior percentile is more suitable for ranking than the 97.5th posterior percentile; because the item has the largest 2.5th posterior percentile is the one that has the maximum lowest probability of being class 1, and hence should be ranked the first. In other words, items

having larger lower confidence bounds should appear earlier in the ranking.

Table 5.11 shows the average precisions for the NCI data when the posterior mean and the lower bound of the 95% posterior credibility interval are used to rank the compounds. Figure 5.16 compares the values of the lower bounds and the predicted probabilities, i.e., the posterior means.

	Posterior Mean	Lower Bound
Split1	0.2512	0.2388
Split2	0.2696	0.2539
Split3	0.2509	0.2515
Split4	0.2767	0.2687

Table 5.11: Comparing the test-set average precisions of the NCI data when the posterior mean and the 2.5th posterior percentile (lowerbound) are used as the ranking scores.

Both Table 5.11 and Figure 5.16 indicate that taking into account the variation of the predicted value will not affect the ranking dramatically. There is no significant difference in average precision between the posterior mean and the lower bound according to the result of a pairwise t-test (p-value=0.36).

The "banana" shapes in Figure 5.16 are not surprising; because as we know, the variance of a Bernoulli variable is larger when the probability of success p is closer to 0.5.

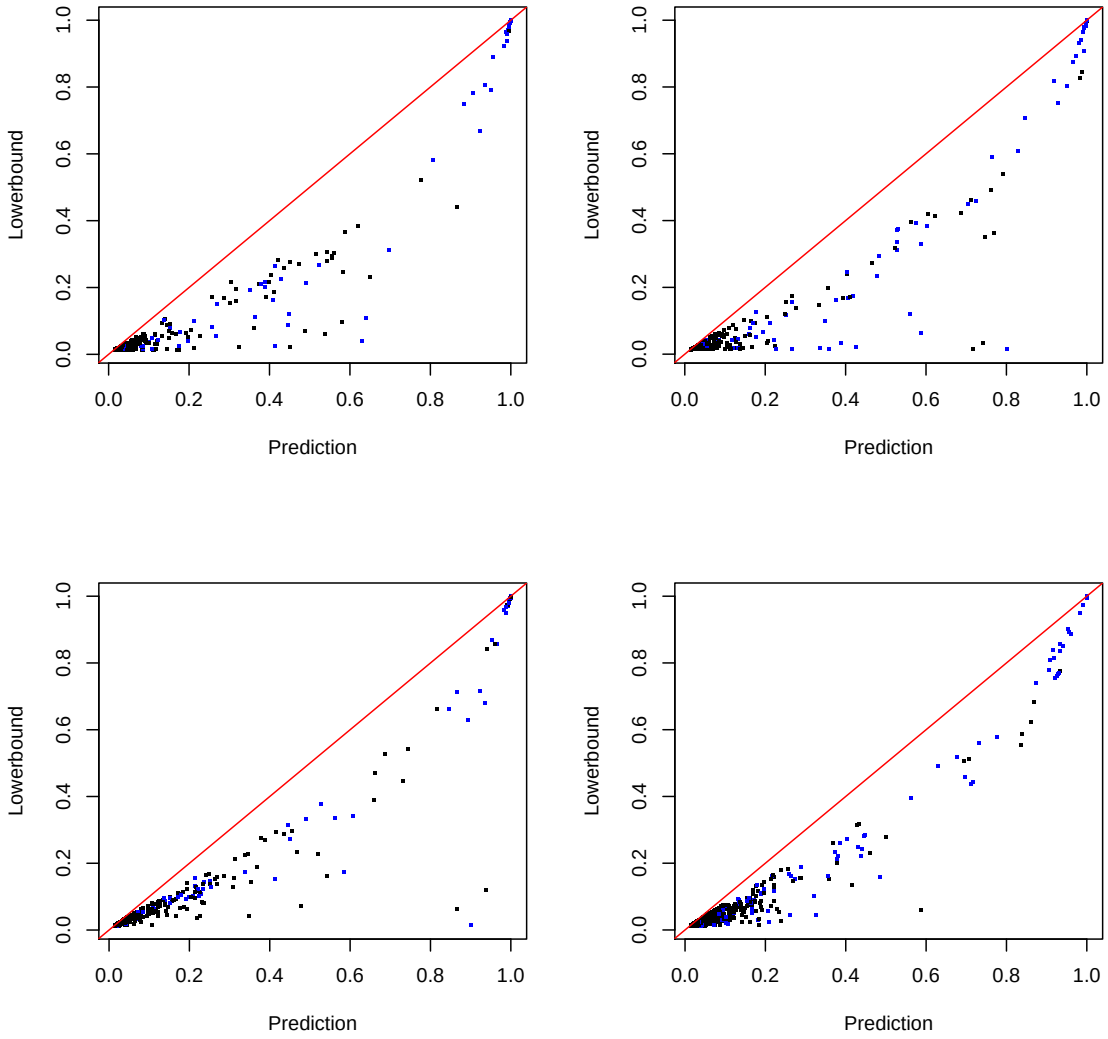

Figure 5.16: Comparing the lower bounds of the credibility intervals (lowerbound, the y-axis) and the posterior mean (prediction, the x-axis) for the four random splits of the NCI data. Upper left: the 1st split; Upper right: the 2nd split; Lower left: the 3rd split; Lower right: the 4th split.

As for the Mysim data, Figure 5.17 compares the 100 average precisions when the lower bound is used as the ranking score with those when posterior mean is adopted as the ranking score. A pairwise t-test shows that there is no significant difference in average precision between these two ranking scores (p-value=0.75).

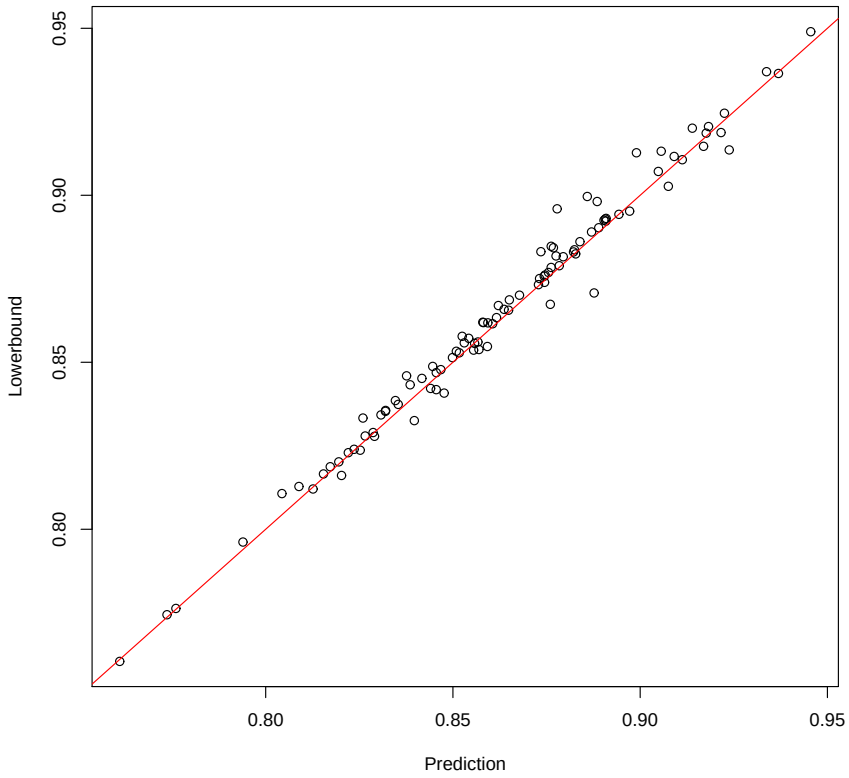

Figure 5.17: The test-set average precisions of lower bound (lowerbound) and posterior mean (prediction) based on 100 experiments of the Mysim data.

The comparison for the Mysim-LAGO data is presented in Figure 5.18. Pairwise t-tests show that there is no significant difference between these two ranking scores in average precision, with p-value being 0.33 for $n = 400$ and 0.93 for $n = 4000$. When the sample size is large enough, the rankings are identical no matter the posterior mean or the lower bound is used for ranking.

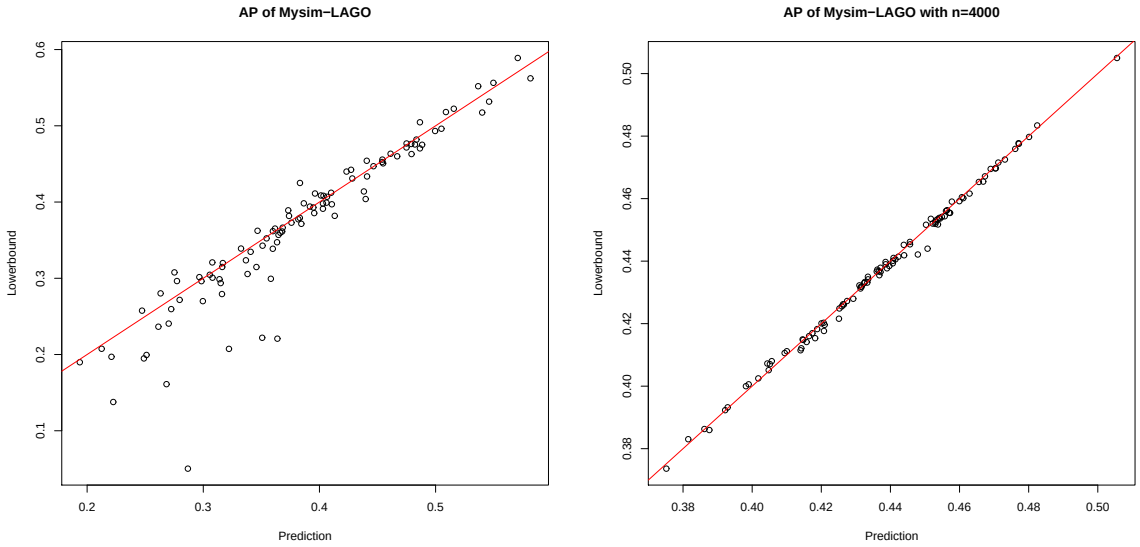

Figure 5.18: The test-set average precisions of the lower bound (lowerbound) and the posterior mean (prediction) based on 100 experiments of the Mysim-LAGO data. Left: when sample size $n = 400$; Right: when $n = 4000$.

5.8.3 MCMC Methods

Besides the Laplace approximation for computing the posterior mentioned in Section 5.3, we also tried two MCMC algorithms with random walk proposals. These two MCMC approaches were tried before adopting the more efficient Laplace approximation approach. We document the two MCMC approaches here briefly.

In both MCMC approaches, we assume the prior distributions of K and α are independent and non-informative. The first approach adopts the logit link function to compute the likelihood function and Metropolis-Hastings algorithm to obtain the posterior distributions. The second one applies the probit link to calculate the likelihood and hybrid of Gibbs sampling and Metropolis to get the posterior distributions. In the first approach, we do not draw $\boldsymbol{\beta}$ explicitly, but integrate it out using the Laplace approximation. In

the second approach, we borrow Albert and Chib's (1993) approach for computing the exact posterior distribution of $\boldsymbol{\beta}$. The main idea is to introduce N independent continuous latent variables $Z_1, \ldots, Z_N$ such that $Z_i \sim N(\mathbf{t}_i^T\boldsymbol{\beta}, I)$, and define $y_i = 1$ if $Z_i > 0$ and $y_i = 0$ if $Z_i \leq 0$, where $\mathbf{t}_i = (1, g_i)^T$ with g_i being the standardized LAGO score for the ith training point $(\mathbf{x}_i, y_i)$. Given the data $\mathbf{Y} = (y_1, \ldots, y_N)$, the joint posterior distribution $f(K, \alpha, \boldsymbol{\beta}, \mathbf{Z}|\mathbf{Y})$ is given by

$$f(K, \alpha, \boldsymbol{\beta}, \mathbf{Z}|\mathbf{Y}) \propto f(\mathbf{Z}|K, \alpha, \boldsymbol{\beta}) f(\boldsymbol{\beta}) f(K) f(\alpha) \mathcal{I}(\mathbf{Y}; \mathbf{Z}), \tag{5.28}$$

where $\mathcal{I}(\mathbf{Y}; \mathbf{Z})$ is indictor function checking whether y_i and Z_i are consistent or not. Equation (5.28) is under the assumption that the priors of K, α and $\boldsymbol{\beta}$ are independent. Suppose the prior of $\boldsymbol{\beta}$ is bivariate normal with mean $\boldsymbol{\mu}_0$ and variance-covariance matrix Σ_0, the full conditonals are as follows:

$$f(\mathbf{Z}|K, \alpha, \boldsymbol{\beta}, \mathbf{Y}) \propto f(\mathbf{Z}|K, \alpha, \boldsymbol{\beta}) \mathcal{I}(\mathbf{Y}; \mathbf{Z}) = \prod_{i=1}^{N} f(Z_i|K, \alpha, \boldsymbol{\beta}, y_i), \tag{5.29}$$

where $f(Z_i|K, \alpha, \boldsymbol{\beta}, y_i) \sim N(\mathbf{t}_i^T\boldsymbol{\beta}, 1)$ truncated on the left by 0 if $y_i = 1$ or truncated on the right by 0 if $y_i = 0$ for $i = 1, \cdots, N$;

$$f(\boldsymbol{\beta}|K, \alpha, \mathbf{Z}, \mathbf{Y}) \propto f(\mathbf{Z}|K, \alpha, \boldsymbol{\beta}) f(\boldsymbol{\beta}) \sim \text{MVN}(\tilde{\boldsymbol{\beta}}, \tilde{\Sigma}) \tag{5.30}$$

where $T = [\mathbf{t}_1^T, \cdots, \mathbf{t}_N^T]^T$, $\tilde{\boldsymbol{\beta}} = (\Sigma_0^{-1} + T^T T)^{-1}(\Sigma_0^{-1}\boldsymbol{\mu}_0 + T^T\mathbf{Z}^{(j)})$ and $\tilde{\Sigma} = (\Sigma_0^{-1} + T^T T)^{-1}$;

$$f(K, \alpha|\boldsymbol{\beta}, \mathbf{Z}, \mathbf{Y}) \propto f(\mathbf{Z}|K, \alpha, \boldsymbol{\beta}) f(K) f(\alpha) \propto \prod_{i=1}^{N} N(\mathbf{t}_i^T\boldsymbol{\beta}, 1). \tag{5.31}$$

We draw K and α by the Metropolis-Hastings algorithm, and draw $\mathbf{Z}$ and $\boldsymbol{\beta}$ by Gibbs sampling.

The MCMC algorithms were abandoned eventually because of the following reasons:

- The parameters K, α and (β_0, β_1) are highly correlated, which makes the Markov chain take many iterations to explore the parametric space and results in slow convergence. Even though reparametrization might be able to reduce the correlation

among the parameters, there is no general guideline to determine a suitable transformation, especially for mixture of integer and continuous variables. See Chapter 3 and 5 of Ganerman and Lopes' book (Gamerman and Lopes, 2006) for more detail.

- To decide the number of iterations that ensures convergence is still a challenging problem. Cowles and Carlin (1996) reviewed and compared thirteen state-of-the-art convergence diagnostic methods. Some methods are designed for multiple chains while others are designed for single chains; some methods are quantitative while others are informal and graphical; some methods are only applicable to Gibbs algorithm and some are suitable to any MCMC algorithm. Among those thirteen methods discussed, the most popular ones are Gelman's diagnostics (Gelman and Rubin, 1992), Geweke's diagnostics (Geweke, 1992), and Raftery's diagnostics (Raftery and Lewis, 1992). All of these three methods have already been implemented in the R library called *coda*. I have applied these three diagnostics methods on the probit MCMC approach with 50,000 iterations for the NCI data. Gelman's test suggests that 50,000 iterations is enough; Geweke's test, however, fails in some parallel chain; Raftery's test in general requires much more iterations than 50,000. Different diagnostics give different conclusions, it is hard to claim that 50,000 iterations is adequate to ensure the Markov chain to reach its equilibrium distribution so that inference based on the posterior samples is valid.

- Last but not least, even though the computational cost of BLAGO using the Laplace approximation is much cheaper than MCMC, its prediction performance is better than or at least as good as MCMC algorithms.

5.9 Conclusion

We propose and implement a Bayesian approach for LAGO, referred to as BLAGO. In our Bayesian approach, we evaluate the posterior distribution on a fine grid of (K, α) via the Laplace approximation. A test point's probability of being class 1 is estimated by the posterior mean, and the uncertainty of this estimate is captured by a 95% credibility interval.

We apply BLAGO to the NCI data and two simulated data sets. Results indicate that BLAGO provides a ranking which is at least as effective as LAGO. One advantage gained from BLAGO is a probabilistic estimate, which is essentially needed for decision-making applications associated with costs. Despite the fact that BLAGO seems to underestimate overall uncertainty, that the length of its credibility interval is still approximately proportional to the corresponding standard deviation suggests that we can still rely on it to assess the relative uncertainty of its prediction.

Computationally, BLAGO is much faster than if it were implemented via MCMC. Furthermore, the Bayesian framework removes the need to select parameters K and α via cross-validation, making it more computationally efficient than LAGO.

Chapter 6

Conclusions and Future Research

6.1 Summary of the Thesis

The basic objective of my thesis is to propose novel statistical methods to address statistical detection problems. The set up of the detection problem is the same as a binary classification problem. The reasons that we would like to treat statistical detection as a special problem are twofold. First, the frequencies of the two classes are extremely unbalanced. There are only a few observations belonging to the class of interest (class 1), say 1-2% or even less. Second, the objective of statistical detection is to assign a relative ranking to the data according to their probability of being class 1 and, hence, find the class-1 items as early as possible. This is very different from the common classification problem, which pursues the least classification error. We use the drug discovery problem as the main application to illustrate our methods. Other applications of the detection problem include direct marketing and fraud detection.

Chapter 2 describes the cycle of drug discovery and development, including its tasks in each stage. How data mining techniques help speed up the screening to identify the active compounds in the early stage of drug discovery is also discussed. Chapter 2 presents the AIDS antiviral database from the National Cancer Institute, the main drug discovery

application used throughout this thesis.

The statistical detection problem is defined in Chapter 3. We briefly introduce several popular and powerful statistical methods that can be applied to statistical detection. These methods are KNN, trees, SVM and ASVM. Model assessment of statistical detection is discussed in great detail in this chapter. Due to the two main characteristics of statistical detection, we argue that the conventional misclassification rate is no longer a good standard; average precision which favors early detection is better.

LAGO, a computationally efficient adaptive statistical detection method is presented in detail in Chapter 4. LAGO, standing for the locally adjusted "GO" estimator, is used to approximate the density ratio $f = \frac{p_1(\cdot)}{p_0(\cdot)}$ in the form of basis functions. Each component is a kernel function centered at a class-1 observation x_i and with αr_i as the bandwidth, where r_i is the average distance from x_i to its K nearest class-0 neighbors, and α is a global tuning stretching parameter. For multivariate cases, each component is a product of univariate kernel functions across all dimensions. Wang (2005) argued that KNN is one of the best methods for the NCI data and because both LAGO and SVM can be viewed as radial basis function networks, KNN and SVM provide very good benchmarks for performance comparison. Applications to the NCI AIDS anti-viral database and two simulated data sets illustrate that LAGO is competitive with KNN and SVM.

One main drawback of LAGO is that it is not a probabilistic estimate and it is difficult to make inference on the prediction. Chapter 5 tailors LAGO to a Bayesian framework. We take into account the model uncertainty by assigning prior distributions to the parameters. An analytical method with the Laplace approximation is used to obtain the posterior distributions. Bayesian LAGO results in proper probabilistic predictions that have support on (0,1). Computationally, BLAGO is more efficient than MCMC algorithms and LAGO.

The following section proposes some promising extensions of Bayesian LAGO and LAGO as some possible directions of future research.

6.2 Future Research

6.2.1 Asymmetric LAGO

To rank the items, we only need their relative scores of being class 1. These scores do not need to be well calibrated to their true probabilities; instead it may suffice if the scores preserve the correct ranking. As for BLAGO, however, we do need a well-calibrated probabilistic prediction to evaluate the likelihood function (5.14). Equation (5.13) implies that a well-calibrated probabilistic prediction θ_i relies on a accurate LAGO score $\hat{f}(\mathbf{x}_i)$. For example, we should correct the bias of LAGO score as much as we can before feeding it into a logistic regression if this bias exists. This motivates us to study the bias of LAGO estimate. And as a result, we come up with a more advanced LAGO method that can correct the bias to some extent. We call this new LAGO method the asymmetric LAGO (ALAGO).

Bias of LAGO and Correction by ALAGO

Recall that LAGO given in (4.6) is constructed to estimate the density ratio (4.2) in two steps. The first step is to estimate $p_1(\cdot)$ by an adaptive kernel density estimate; and the second step is to locally adjust $\hat{p}_1$ by a local constant. Now, let's examine the bias introduced by this adjustment. For simplicity, we only consider the univariate case with $\alpha = 1$. Estimating p_1 by a adaptive kernel density estimate, the density ratio becomes

$$\begin{aligned}
& \frac{p_1(z)}{p_0(z)} \\
\approx\ & \frac{\frac{1}{n_1}\sum_{y_i=1}\mathcal{K}(z;x_i,r_i)}{p_0(z)} \\
=\ & \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(z)}\mathcal{K}(z;x_i,r_i) \\
=\ & \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(x_i)-p_0(x_i)+p_0(z)}\mathcal{K}(z;x_i,r_i) \\
=\ & \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(x_i)[1-\frac{p_0(x_i)-p_0(z)}{p_0(x_i)}]}\mathcal{K}(z;x_i,r_i). \qquad (6.1)
\end{aligned}$$

Applying the well-known series expansion

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots$$

and the mean value theorem, equation (6.1) becomes

$$\begin{aligned}
&= \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(x_i)}\mathcal{K}(z;x_i,r_i)\left[1+\frac{p_0(x_i)-p_0(z)}{p_0(x_i)}+\frac{[p_0(x_i)-p_0(z)]^2}{p_0^2(x_i)}+\cdots\right] \\
&= \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(x_i)}\mathcal{K}(z;x_i,r_i)\left[1+\frac{(x_i-z)p_0^{'}(\xi)}{p_0(x_i)}+\frac{[(x_i-z)p_0^{'}(\xi)]^2}{p_0^2(x_i)}+\cdots\right] \\
&\approx \frac{1}{n_1}\sum_{y_i=1}\frac{1}{p_0(x_i)}\mathcal{K}(z;x_i,r_i)+\frac{1}{n_1}\sum_{x_i\in C_1}\frac{(x_i-z)p_0^{'}(\xi)}{p_0^2(x_i)}\mathcal{K}(z;x_i,r_i), \qquad (6.2)
\end{aligned}$$

where ξ is somewhere between x_i and z.

Note that Theorem 1 given in Section 4.1.2 implies that the first term of (6.2) is the LAGO score. If within a neighborhood of x_i, the density p_0 is roughly a constant, (6.2) becomes LAGO score given in (4.6) and no bias exists up to a normalizing constant. However, if the density p_0 within a local region of x_i is not constant, the LAGO score might underestimate or overestimate the density ratio. For example, assume that p_0 is a monotonically increasing function within a neighborhood of x_i, which means that the distribution of class-1 observations is more dense on the right-hand side of x_i than on its left-hand side, i.e., $p_0^{'}(x_i) > 0$. If z is on the left-hand side of all $x_i \in C_1$, we have $x_i - z > 0$ and hence the second term of (6.2) is positive; on the other hand, if z is on the right-hand side of all x_i's, we have $x_i - z < 0$ and hence the second term of (6.2) is negative. This means LAGO score tends to underestimate the density ratio for those points on the left-hand side of the training class-1 observations; whereas LAGO score tends to overestimate the density ratio for those data points that are on the right-hand side of the training class-1 observations. However, if the density of p_0 is instead monotonically decreasing within a neighborhood of x_i, the bias in the estimated density ratio will be the other way around. We might be able to somehow correct this bias by using different bandwidths on different sides of x_i.

For ALAGO, we still use the idea of nearest neighbors to define the bandwidths of the kernel functions. Given a class-1 observation $x_i \in C_1$, we calculate its left-hand-side bandwidth $r_{i,left}$ by using only the left-hand-side class-0 neighbors $N_l(K_l, x_i)$ and its right-hand-side bandwidth $r_{i,right}$ by using only the right-hand-side class-0 neighbors $N_r(K_r, x_i)$. That is

$$\begin{aligned} r_{i,left} &= \frac{1}{K_l} \sum_{w \in N_l(K_l, x_i)} |w - x_i| \\ r_{i,right} &= \frac{1}{K_r} \sum_{w \in N_r(K_r, x_i)} |w - x_i|, \end{aligned}$$

where K_l and K_r indicate the number of elements in set $N_l(K_l, x_i)$ and $N_r(K_r, x_i)$ respectively. Set the bandwidth to be a large number if there is no class-0 observation on that side, e.g., let $r_{i,left} = a$ if $N_l(K_l, x_i) = \emptyset$, where $a = \max\{x\} - \min\{x\}$. Given a test point z, its LAGO score contributed by x_i will depend on whether z is smaller or larger than x_i. If $z < x_i$, replace r_i with $r_{i,left}$ in equation (4.6); otherwise, replace r_i with $r_{i,right}$. Figure 6.1 compares the kernel functions of $x_i \in C_1$ for LAGO and ALAGO given $K = 2$ (for LAGO) or $K_l = K_r = 1$ (for ALAGO). Those two points $x_i - 2$ and $x_i + 1$ are two class-0 neighbors of x_i that are two units and one unit away from x_i respectively. Obvious, the bandwidth r_i for LAGO is $r_i = \frac{|-2|+|1|}{2} = \frac{3}{2}$; while for ALAGO, $r_{i,left} = 2, r_{i,right} = 1$.

The approach of ALAGO is equivalent to modeling the density of p_0 within a neighborhood of x_i with a step function rather than a local constant. Recall that if $p_0'(x_i) > 0$, the density ratio of those points on the left-hand side of x_i will be underestimated. This downward bias can be corrected to some extent by using a larger bandwidth and hence the LAGO score will be increased. Similarly, by using a smaller bandwidth for the data points on the right-hand side of x_i, we are able to reduce their LAGO scores and hence correct the upward bias.

LAGO or ALAGO?

ALAGO is a more complex model than LAGO. In cases where the distribution of class-0

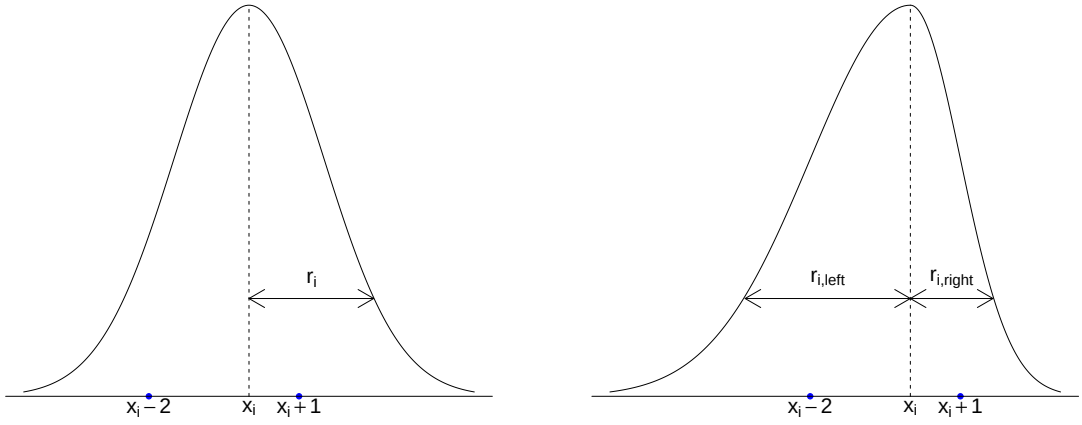

Figure 6.1: Left: Kernel function used in LAGO with $K = 2$. Right: Kernel function used in ALAGO with $K_l = K_r = 1$. Those two points, $x_i - 2$ and $x_i + 1$ indicate the two nearest class-0 neighbors of x_i.

observations is balanced around $x_i \in C_1$, ALAGO and LAGO should give similar results. In order to determine whether the more complex ALAGO is necessary, a pair-wise t-test can be conducted to compare the left-hand-side bandwidths and right-hand-side bandwidths. If the left-hand-side and right-hand-side bandwidths are significantly different, we should adopt ALAGO; otherwise, we should go directly for LAGO. This approach will be illustrated in the next section. For multivariate cases, a pair-wised t-test can be conducted in each dimension and the significance level can be adjusted by Bonferroni correction (Montgomery, 2000).

A Toy Example

A simulation was conducted to compare the performance of LAGO and ALAGO. Suppose $x|C_1 \sim N(-1, 0.22^2)$ and $x|C_0 \sim N(1, 0.5^2)$. Clearly, p_0 is not flat and thus bias might exist as suggested above. A set of training data are generated as follows: generate 100 class-1 and 1000 class-0 observations. Validation data and test data are also generated in

the same way. Fixing $K = 6$ for LAGO and $K_l = K_r = 3$ for ALAGO, choose the optimal α that minimizes absolute error using the validation data. The absolute error is defined as

$$\text{Error}_a = \frac{1}{n}\sum_{i=1}^{n} |\hat{p}_i - p_i|, \tag{6.3}$$

where n is the total number of observations in the validation set; p_i is the ith validated point's true probability of being class 1, which can be calculated by Bayes' rule; and $\hat{p}_i$ is the corresponding predicted probability, which is obtained by passing the (A)LAGO score through a logistic model given in (5.13), and (β_0, β_1) are the MLE of the fitted logistic regression. Cross-validation on a grid of $\alpha \in \{0.01, 0.1, 0.25, 0.5, 1, 1.5, 2, 3, 4, 5, 10\}$ indicates $\alpha = 1$ for LAGO and $\alpha = 1.5$ for ALAGO. The resulting MLE of the logistic regression is $(\hat{\beta}_0, \hat{\beta}_1) = (-6.95, 15.80)$ for LAGO and is $(\hat{\beta}_0, \hat{\beta}_1) = (-8.10, 13.47)$ for ALAGO. We are "cheating" in such a way that we used the true probability p_i to choose the optimal α. However, since the focus here is on comparing LAGO and ALAGO, we let both algorithms "cheat" in the same way and pick their own optimal α values.

The original LAGO and ALAGO scores are compared to the true density ratio $f = \frac{p_1}{p_0}$ in Figure 6.2, which shows that even though the scales of (A)LAGO score and the true density ratio f are very different, the ALAGO score is monotonically increasing in f and hence preserves the right ranking of f; however, LAGO wrongly ranks some observations in the tail. Figure 6.3 compares the LAGO and ALAGO predictions of the test data after the logistic transformation (5.13) to their true probabilities of being class 1. Notice that the original LAGO method underestimates some class-1 observations' probabilities of being class 1 and ALAGO is able to significantly correct the bias.

The p-value of a pairwise t-test on comparing the left-hand-side bandwidths and right-hand-side bandwidths is less than 10^{-15}, which means the densities of class 0 on different sides of $x_i \in C_1$ are significantly different.

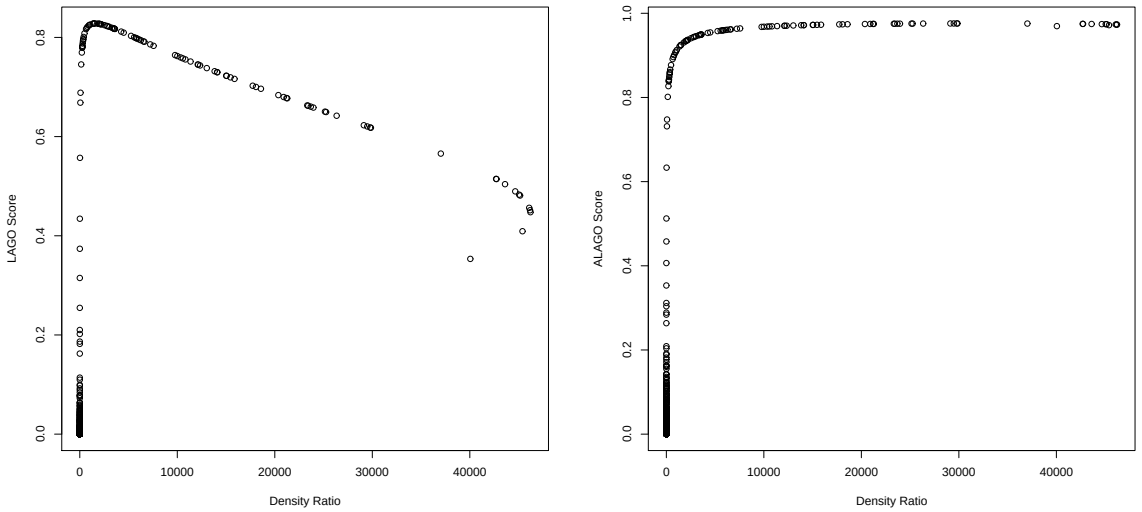

Figure 6.2: Left: LAGO score versus the true density ratio. Right: ALAGO score versus the true density ratio

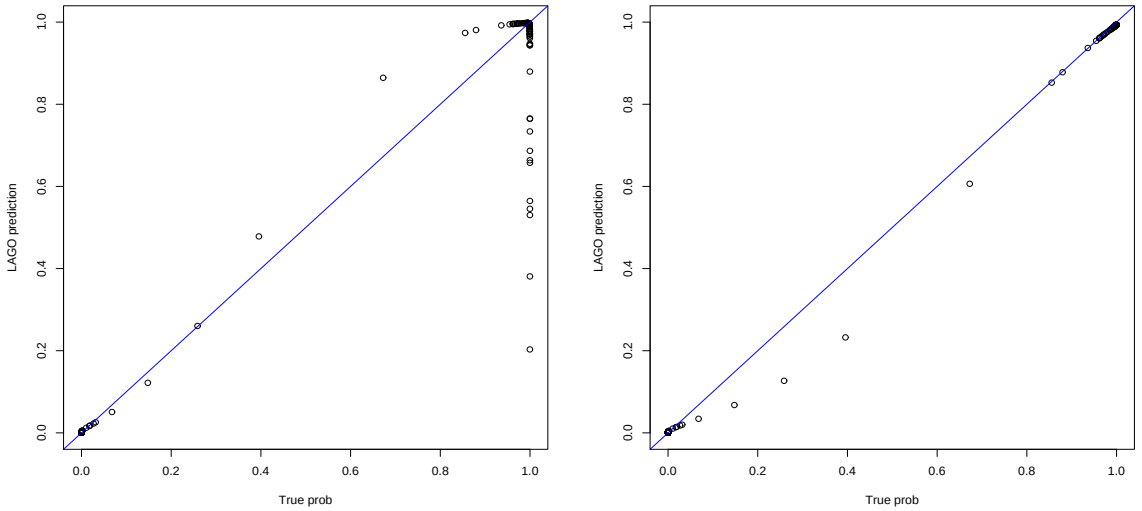

Figure 6.3: Left: LAGO prediction versus the true probability. Right: ALAGO prediction versus the true probability

The performance of LAGO and ALAGO on the test set in terms of absolute error, average precision and deviance are summarized in Table 6.1, which shows that ALAGO is superior to LAGO in all three criteria.

Performance	LAGO	ALAGO	Optimal
Average Precision	0.9896	0.9991	0.9991
Absolute Error	7.4174	1.4735	0
Deviance	34.6974	13.6768	13.3432

Table 6.1: Comparison of LAGO and ALAGO on a toy example. The optimal results are obtained by Bayes' rule.

6.2.2 BLAGO-II

In Chapter 5, we introduced the Bayesian LAGO which estimates a point's probability of being class 1 by

$$\begin{aligned} \text{logit}\{p_{\hat{f}(z)}\} &= \beta_0 + \beta_1 \hat{f}(z) \\ &= \beta_0 + \frac{\beta_1}{n_1} \sum_{y_i=1} r_i \mathcal{K}(z; x_i, r_i), \end{aligned} \tag{6.4}$$

where $\hat{f}(z)$ is the LAGO score of z given parameters K and α. BLAGO given in (6.4) can be viewed as a logistic regression on all kernel components $r_i\mathcal{K}(z; x_i, r_i), \forall x_i \in C_1$ with equal coefficient $\frac{\beta_1}{n_1}$. This model can be easily extended to a more flexible model by assigning different weights to different kernel components as follows,

$$\log \frac{p_{\hat{f}(z)}}{1 - p_{\hat{f}(z)}} = \beta_0 + \frac{1}{n_1} \sum_{x_i \in C_1} \beta_i r_i \mathcal{K}(z; x_i, r_i). \tag{6.5}$$

We call this more complex model BLAGO-II. Our BLAGO approach can be easily extended to fit BLAGO-II. Since the number of predictors of (6.5) equals to the number of class-1 observations which is in general over a hundred; to avoid overfitting, regularization techniques might be necessary.

Two well-known regularization methods in regression context are ridge regression (Hoerl and Kennard, 1970) and the least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996). The LASSO is a shrinkage and selection method for linear regression. It minimizes the residual sum of squares subjected to the sum of the absolute value of the coefficients bounded by a constant. One most attractive feature of the LASSO is that it shrinks the coefficients of those weak predictors to zero and thus produces sparse solutions. This shrinkage not only gives a more accurate prediction but also makes the resulting model much easier to interpret. The computation of the LASSO solutions is a quadratic programming problem with linear inequality constraints, which can be solved by standard numerical analysis algorithm. However, a simple modification of the least angle regression algorithm introduced by Efron *et al.* (2004) provides a more efficient approach to compute the LASSO solutions.

Hastie *et al.* (2001, Chapter 3) pointed out that penalized methods such as ridge regression and the LASSO can usually be cast in a Bayesian framework by thinking of the penalty function as the log-prior, and the penalized loss function as the log-posterior. Therefore, minimizing the penalized loss function amounts to finding the posterior mode. They further shown that ridge regression can also be derived as the posterior mode (or mean) by choosing the prior distribution to be normal; the LASSO can be derived as the mode of a posterior distribution with double exponential as the prior distribution.

One of the most attractive ideas behind BLAGO-II is that we can allow the coefficients β_i to be either significant large or exactly zero via choosing suitable sparsity-promoting priors. To be more specific, we could adopt the double-exponential prior which results in a LASSO-type solution (see Krishnapuram *et al.*, 2005); or we could apply the idea of using a hyper-prior in the same way as Tipping (2001) did for the relevance vector machines.

In order to have a rough idea how BLAGO-II performs, we estimate the coefficients of BLAGO-II (6.5) by fitting a neural network without hidden units and with decay > 0. There are three tuning parameters in this logistic neural network (LNNet) model — K, the

number of class-0 neighbors used to determine the radii r_i's; the global stretching factor α; and the decay parameter of NNet. The optimal values of parameters can be chosen by cross-validation. Note that this LNNet is actually a logistic regression with L_2 penalty, i.e., a ridge logistic regression (Le Cessie *et al.*, 1992).

We applied LNNet to the NCI data. Table 6.2 reports the top 5 optimal values of the three tuning parameters K, α, decay, and their corresponding cross-validated and test-set average precisions (AP). Figure 6.4 shows the contour plots of cross-validated AP evaluated at different values of K and α given the optimal value of decay. According to the APs on the test sets of four random training/test splits of NCI data, BLAGO-II is very promising.

	K	α	Decay	CV AP	Test AP
Split 1	5	1.5	300.00	0.2463	0.2418
	5	1	300.00	0.2462	0.2388
	5	1	266.67	0.2461	0.2385
	5	1	200.00	0.2460	0.2377
	5	1.5	266.67	0.2460	0.2409
Split 2	20	1	300.00	0.2496	0.2630
	20	1	266.67	0.2496	0.2632
	9	1	266.67	0.2493	0.2701
	9	1	300.00	0.2493	0.2701
	9	1	233.33	0.2493	0.2701
Split 3	9	1.5	266.67	0.2407	0.2571
	9	1.5	233.33	0.2406	0.2559
	9	1.5	300.00	0.2406	0.2573
	9	1.5	200.00	0.2403	0.2563
	7	1.5	266.67	0.2399	0.2514
Split 4	1	5	300.00	0.1947	0.2628
	1	5	266.67	0.1940	0.2630
	1	5	233.33	0.1931	0.2628
	1	5	200.00	0.1921	0.2637
	3	3	200.00	0.1916	0.2809

Table 6.2: Top 5 optimal choices of the three tuning parameters—K, α,decay; "CV AP" refers to the cross-validated average precision and "Test AP" refers to the average precision on the test set.

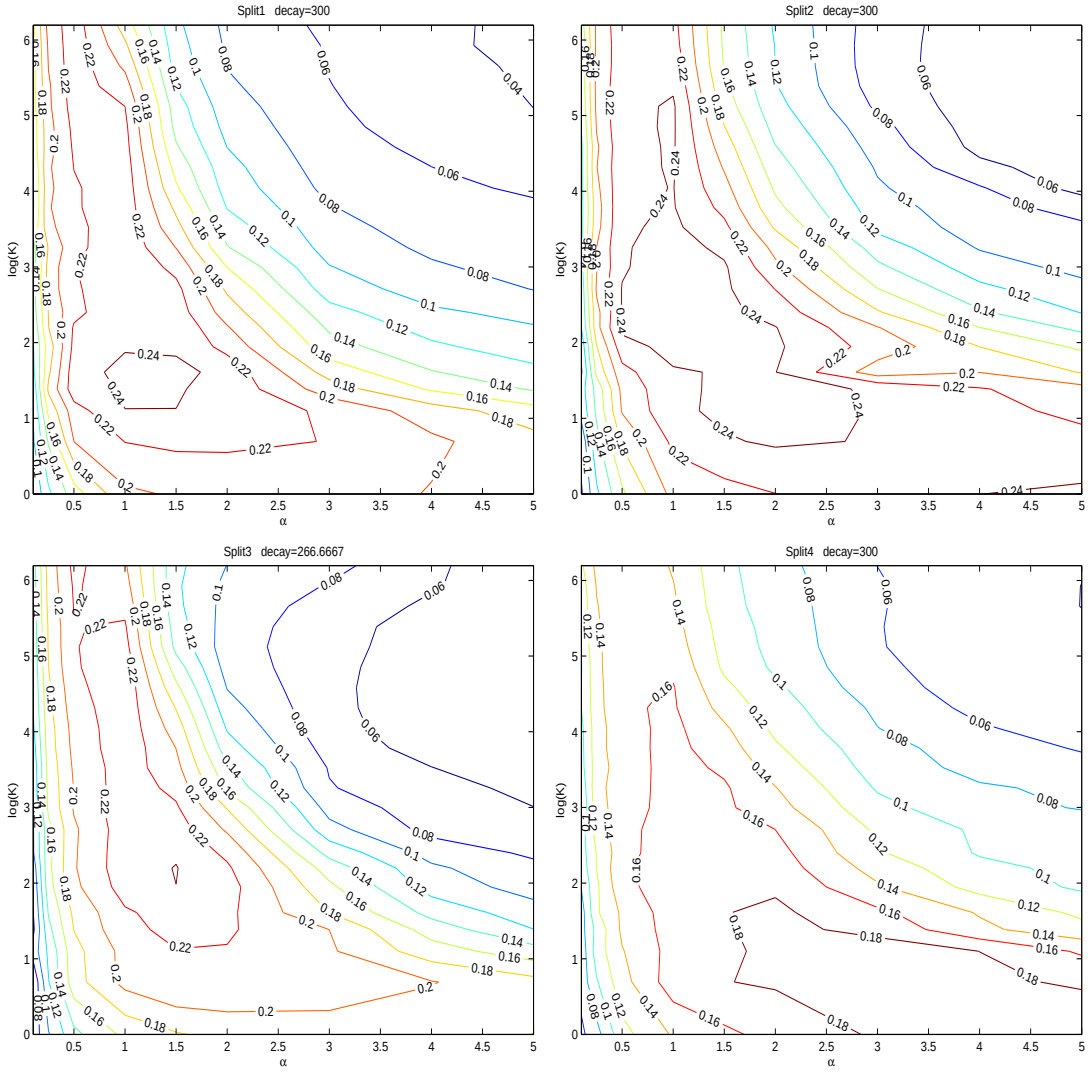

Figure 6.4: Contour plots of test-set average precision for each random split of the NCI data given the optimal value of decay.

6.2.3 Generalize to Multi-class

For statistical detection problems, we consider only the cases of binary response so far. Even it might not be necessary, it is possible to generalize LAGO and BLAGO to multi-class ($J > 2$) applications.

As we know, methods such as tree models, KNN, NNet take care of multi-class response automatically; however, methods such as logistic regression and SVM are originally designed for binary outcomes. Models addressing logistic regression with multi-class response include proportional odds model, continuation-ratio model and adjacent-category model, depending how the odd ratios are specified (Agresti, 1990).

How to effectively extend the binary SVM to multi-class SVM is still an on-going research topic (e.g., Hsu and Lin, 2002). Currently, there are two types of approaches for multi-class SVM. One is to construct and combine several individual binary SVMs; the other one is to solve a multi-class SVM that contains several decision functions in a single optimization step. The first approach includes methods of one-against-rest, one-against-one, and Direct Acyclic Graph (DAG) SVM. The one-against-rest method constructs J SVMs and hence resulting in J decision functions. The ith SVM is trained with all the observations belonging to class i as the positive examples and observations in other classes as negative examples. A new data x is assigned to the class that yields the largest value among the decision functions. The one-against-one approach solves $J(J-1)/2$ pair-wise SVMs. There are different voting strategies to assign the class labels after the $J(J-1)/2$ classifiers have been constructed. A popular voting method is Friedman's Max Wins method (Friedman, 1996) which takes the majority vote of a certain class as the final output. As for DAGSVM (Platt *et al.*, 2000), the training stage is the same as one-against-one by solving $J(J-1)/2$ binary SVMs; while in the testing phrase, it uses a binary directed acyclic graph with $J(J-1)/2$ internal nodes and J leaves. Each node is a binary SVM of ith and jth classes. Given a test point x, starting from the root node, x moves to either the left or the right depending on the value of the decision function at each internal node, and finally reaches to a leaf node indicating its predicted class.

When it comes back to statistical detection, if there are more than one rare targets (like the NCI data), we can simply combine all the observations from rare classes and convert the response to be binary–either background (coded as 0) or target (coded as 1). Furthermore,

multi-class classification methods can be applied to further distinguish the remaining rare targets. After the rare targets are identified, classifying the rare targets should not be a statistical detection problem any more, since the class frequencies of different rare targets are generally balanced.

6.2.4 Categorical Predictors

Like KNN, LAGO is a method based on a distance metric. The performance of LAGO depends on whether the distance metric honestly represents similarity between the observations. So far, LAGO only considers continuous predictors using Euclidean distance. However, categorical variables are unavoidable when we apply LAGO to direct marketing or fraud detection problems. How to define a meaningful distance between two categorical variables is still a challenging problem and how to define the distance between a categorical variable and a continuous variable is an even more challenging issue. Buttrey (1998) proposed a method to handle categorical covariates for KNN. The main idea is to map the set of categories onto the real line by maximizing the ratio of total sum of squares to within-class sum of squares (aggregated over all classes) and then KNN proceeds with Euclidean distance on the mapped data which are real numbers. Continuous variables can be incorporated in such a way that the continuous variables are first categorized by using a linear-spline-like representation and mapped to real line in the same way as categorical variables. For LAGO with mixture of categorical and continuous predictors, we can first map the predictors to real values using Buttrey's approach and then apply LAGO on the transformed data.

Appendix: Proof of Theorem 1

Here we give the proof of Theorem 1, which states:

Let x_0 be a fixed observation from class 1. Suppose that $w_1, w_2, \ldots, w_m$ are iid observations from class 0 that are uniformly distributed around x_0, say on the interval $[x_0 - \frac{1}{2c_0}, x_0 + \frac{1}{2c_0}]$. If r_0 is the average distance between x_0 and its K nearest neighbors from class 0 ($K < m$), then we have

$$\mathrm{E}(r_0) = \frac{K+1}{4(m+1)c_0}.$$

Proof:

Without loss of generality, we assume $x_0 = 0$. Let $u_j = |w_j|$ be the distance between w_j and x_0, and $p(z)$ be the density of $u_j, j = 1, 2, \ldots, m$. Given that the density $p(w) = c_0$ is a constant in a local neighborhood around 0, it is straightforward to show that $p(u) = 2c_0$ and its corresponding cumulative distribution function $F(u) = 2c_0 u$. Therefore, the density of the kth order statistics of $u_1, u_2, \ldots, u_m$, $f_{u_{(k)}}$, is given by

$$f_{u_{(k)}}(u) = \frac{m!}{(k-1)!(m-k)!}(2c_0)(2c_0 u)^{k-1}(1 - 2c_0 u)^{m-k}. \tag{6.6}$$

By (6.6),

$$\begin{aligned}
\mathrm{E}(u_{(k)}) &= \int_0^{1/2c_0} u f_{u_{(k)}}(u)du \\
&= \int_0^{1/2c_0} \frac{m!}{(k-1)!(m-k)!}(2c_0u)^k(1-2c_0u)^{m-k}du \\
&\overset{y=2c_0u}{=} \frac{1}{2c_0}\int_0^1 \frac{m!}{(k-1)!(m-k)!}y^k(1-y)^{m-k}dy \\
&= \frac{1}{2c_0}\int_0^1 \left(\frac{k}{m+1}\right)\frac{(m+1)!}{k!(m-k)!}y^k(1-y)^{m-k}dy \\
&= \frac{k}{2(m+1)c_0}, \qquad (6.7)
\end{aligned}$$

where the last step is due to the fact that

$$\int_0^1 \frac{(m+1)!}{k!(m-k)!}y^k(1-y)^{m-k}dy = 1$$

because the foregoing integrand is the density function of Beta$(k+1, m-k+1)$.

Since

$$r_0 = \frac{1}{K}\sum_{k=1}^{K} u_{(k)}, \qquad (6.8)$$

equation (6.7) implies

$$\begin{aligned}
\mathrm{E}(r_0) &= \frac{1}{K}\sum_{k=1}^{K}\mathrm{E}(u_{(k)}) \\
&= \frac{1}{K}\sum_{k=1}^{K}\frac{k}{2(m+1)c_0} \\
&= \frac{K+1}{4(m+1)c_0}. \qquad (6.9)
\end{aligned}$$

□

Bibliography

[1] Agresti, A. (1990), *Categorical Data Analysis*, John Wiley & Sons, New York.

[2] Albert, J. H. and Chib, S. (1993), Bayesian Analysis of Binary and Polychotomous Response Data, *Journal of the American Statistical Association*, **88**, 669–679.

[3] Besag, J. (1974), Spatial Interaction and the Statistical Analysis of Lattice Systems (with discussion), *Journal of Royal Statistical Society: Series B*, **36**, 192–236.

[4] Besag, J. (1975), Statistical Analysis of Non-lattice Data, *The Statistician*, **24**, 179–195.

[5] Besag, J. (1986), On the Statistical Analysis of Dirty Pictures, *Journal of Royal Statistical Society: Series B*, **48**, 259–302.

[6] Bolton, R. J. and Hand, D. J. (2002), Statistical Fraud Detection: A Review, *Statistical Science*, **17**, 235–255.

[7] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone C. J. (1984), *Classification and Regression Trees*, Chapman & Hall, New York.

[8] Breiman, L. (1996), Bagging Predictors, *Machine Learning*, **24**, 123–140.

[9] Burden, F. R. (1989), Molecular Identification Number for Substructure Searches, *Journal of Chemical Information and Computer Sciences*, **29**, 225–227.

[10] Buttrey, S. E. (1998), Nearest-neighbor Classification with Categorical Variables, *Computational Statistics and Data Analysis*, **28**, 157–169.

[11] Cantor, S. B. and Kattan, M. W. (2000), Determining the Area Under the ROC Curve for a Binary Diagnostic Test, *Medical Decision Making*, **20**, 468–470.

[12] Caruana, R. and Niculescu-Mizi, A. (2004), Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria, *KDD 2004*, 69–78.

[13] Chang, C. C. and Lin, C. J. (2007), *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[14] Chen, S. X. (1996), Empirical Likelihood Confidence Intervals for Nonparametric Density Estimation, *Biometrika*, **83**, 329–341.

[15] Chin, K. K. (1999), *Support Vector Machines Applied to Speech Pattern Classification*, Master Thesis, Department of Engineering, University of Cambridge.

[16] Chipman, H., George, E. I. and McCulloch, R. E. (1998), Bayesian CART Model Search (with discussion), *Journal of the American Statistical Association*, **93**, 935–960.

[17] Chipman, H., George, E. I. and McCulloch, R. E. (2000), Bayesian Treed Models, *Machine Learning*, **48**, 299–320.

[18] Chipman, H., George, E. I. and McCulloch, R. E. (2006), Bayesian Ensemble Learning, *NIPS 2006*.

[19] Cortes, C. and Vapnik, V. (1995), Support Vector Networks, *Machine Learning*, **20**, 273–294.

[20] Cover, T. and Hart, P. (1967), Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, **IT-13**, 21–27.

[21] Cowles, M. K. and Carlin, B. P. (1996), Markov Chain Monte Carlo Convergence Diagnostics : A Comparative Review, *Journal of the American Statistical Association*, **91**, 883–904.

[22] Cristianini, N. and Shawe-Taylor, J. (2000), *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press.

[23] De Bruijn, N. G. (1970), *Asymptotic Methods in Analysis*, Ansterdam, The Netherlands: North-Holland.

[24] Efron, B. and Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Chapman & Hall, New York.

[25] Efron, B., Haste, T., Johnstone, I. and Tibshirani, R. (2004), Least Angle Regression, *The Annals of Statistics*, **32**, 407–499.

[26] Engelmann, B., Hayden, E. and Tasche, D. (2003), Testing Rating Accuracy, *Risk January 2003*, 82–86.

[27] Eubank, R. L. and Speckman, P. L. (1993), Confidence Bands in Nonparametric Regression, *Journal of the American Statistical Association*, **88**, 1287–1301.

[28] Fix, E. and Hodges, J. L. (1951), Discriminatory Analysis–Nonparametric Discrimination: Consistency Properties, Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas.

[29] Friedman, J. (1996), Another Approach to Polychotomous Classification, Technical Report, Dept. of Statistics, Stanford University, Stanford, CA.

[30] Galindo, C., Liang, H., Kauermann, G. and Carrol, R. J. (2001), Bootstrap Confidence Intervals for Local Likelihood, Local Estimating Equations and Varying Coefficient Models, *Statistica Sinica*, **11**, 121–134.

[31] Gamerman, D. and Lopes, H. F. (2006), *Markov Chain Monte Carlo–Stochastic Simulation for Bayesian Inference*, Second Edition, Chapman & Hall.

[32] Gelfand, A. E. and Smith, A. F. M. (1990), Sampling-Based Approaches to Calculating Marginal Densities, *Journal of the American Statistical Association*, **85**, 398–409.

[33] Gelman, A. and Rubin, D. B. (1992), Inference From Iterative Simulation Using Multiple Sequences (with discussion), *Statistical Science*, **7**, 457–511.

[34] Geman, S. and Geman, D. (1984), Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, *IEEE Transitions on Pattern Analysis and Machine Intelligence*, **6**, 721–741.

[35] Geweke, J. (1992), Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments, *Bayesian Statistics 4*, 169–194. Eds: Bernardo, J. M., David, A. P. and Smith, A. F. M., Oxford University Press, Oxford, U.K.

[36] Hall, P. and Owen, A. (1993), Empirical Likelihood Confidence Bands in Density Estimation, *Journal of Computational and Graphic Statistics*, **2**, 273–289.

[37] Hanley, J. A. and McNeil, B. J. (1982), The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve, *Radiology*, **143**, 29–36.

[38] Hanley, J. A. and McNeil, B. J. (1983), A Method of Comparing the Areas under Receiver Operating Characteristic Curves Derived from the Same Cases, *Radiology*, **148**, 839–843.

[39] Härdle, W. and Marron, J. S. (1991), Bootstrap Simultaneous Error Bars for Nonparametric Regression, *The Annals of Statistics*, **19**, 778–796.

[40] Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning*, Springer.

[41] Hastings, W. K. (1970), Monte Carlo Sampling Methods Using Markov Chains and Their Applications, *Biometrika*, **57**, 97–109.

[42] Hawkins, D. M. and Kass, G. V. (1982), Automatic Interaction Detection, *Topic in Applied Multivariate Analysis*, 269–302. Ed: Hawkins, D. M., Cambridge University Press, Cambridge, U.K.

[43] 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`

[44] Hoerl, A. E. and Kennard, R. (1970), Ridge regression: Biased Estimation for Nonorthogonal Problems, *Technometrics*, **12**, 55–67.

[45] Hoeting, J. A., Madigan, D., Raftery, A. D. and Volinsky, C. T. (1999), Bayesian Model Averaging: A Tutorial (with discussion), *Statistical Science*, **14**, 382–417.

[46] Holmes, C. C. and Adams, N. M. (2002), A Probabilistic Nearest Neighbour Method for Statistical Pattern Recognition, *Journal of Royal Statistical Society: Series B*, **64**, 295–306.

[47] Hsu, C. W and Lin, C. J. (2002), A Comparison of Methods for Multiclass Support Vector Machines, *IEEE Transactions on Neural Networks*, **13**, 415–425.

[48] Hsu, C. W., Chang, C. C. and Liu, C. J. (2007), A Practical Guide to Support Vector Classification, Department of Computer Science and Information Engineering, National Taiwan University. Available on the website of LIBSVM at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[49] Jerri, A. J. (1999), *Introduction to Integral Equations with Applications*, , 2nd Edition, John Wiley & Sons, New York.

[50] Joachims, T. (1998), Making Large-scale SVM Learning Practical, *Advances in Kernel Methods — Support Vector Learning*. Eds: Schölkopf, B., Burges, C. J. C., and Smola, A. J., MIT Press, Cambridge, MA.

[51] Kass, R. E., Tierney, L., and Kadane, J. B. (1990), The Validity of Posterior Expansions Based on Laplace's Method, in *Bayesian and Likelihood Methods in Statistics and Econometrics*, 473–488, Eds: Geisser, S., Hodges, J. S., Press, S. J. and Zellner, A., New York: North-Holland.

[52] Kass, R. E. and Raftery, A. (1995), Bayes Factors, *Journal of the American Statistical Association*, **90**, 773–795.

[53] Krishnapuram, B., Carin, L., Figueiredo, M. and Hartemink, A. (2005), Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**, 957–968.

[54] Lam, R. L. H. (2001), *Design and Analysis of Large Chemical Databases for Drug Discovery*, Ph.D. Thesis, Department of Statistics and Actuarial Science, University of Waterloo.

[55] Lam, R. L. H., Welch, W. J., and Young, S. S. (2002), Uniform coverage designs for molecule selection, *Technometrics*, **44**, 99–109.

[56] Le Cessie, S. and Van Houwelingen, J. C. (1992), Ridge Estimators in Logistic Regression, *Applied Statistics*, **41**, 191–201.

[57] Liang, G. and Yu, B. (2003), Maximum Pseudo Likelihood Estimation in Network Tomography, *IEEE Transactions on Signal Processing*, **51**, 2043–2053.

[58] Liu, Q. and Pierce, D. A. (1994), A Note on Gauss-Hermite Quadrature, *Biometrika*, **81**, 624–629.

[59] Mallick, B. K. and Gelfand, A. E. (1994), Generalized Linear Models with Unknown Link Functions, *Biometrika*, **81**, 237–245.

[60] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953), Equations of State Calculations by Fast Computing Machines, *The Journal of Chemical Physics*, **21**, 1087–1092.

[61] Mercer, J. (1909), Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, **209**, 415–446

[62] Meyer, D. (2007), Support Vector Machines–The interface to libsvm in package e1071, Technische Universität Wien, Austria.

[63] Montgomery, D. C. (2000), *Design and Analysis of Experiments*, 5th Edition, John Wiley & Sons, New York.

[64] 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.

[65] Neal, R. M. (1996), *Bayesian Learning for Neural Networks*, Springer-Verlag, New York.

[66] 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.

[67] Platt, J. (1999a), Probabilistic Outputs for Support Vector Machines and Comparision to Regularized Likelihood Methods, *Advances in Large Margin Classifiers*, 61–74, Eds: Smola, A. J., Bartlett, P., Schölkopf, B., and Schuurmans, O., MIT Press, Cambridge, MA.

[68] Platt, J. (1999b), Fast Training of Support Vector Machines Using Sequential Minimal Optimization, *Advances in Kernel Methods — Support Vector Learning*, 185–208. Eds: Schölkopf, B., Burges, C. J. C., and Smola, A. J., MIT Press, Cambridge, MA.

[69] Platt, J., Cristianini, N. and Shawe-Taylor, J. (2000), Large Margin DAG's for Multiclass Classification, *Advances in Neural Information Processing Systems*, **12**, 547–553.

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

[71] Raftery, A. E. and Lewis, S. (1992), How Many Iterations in the Gibbs Sampler?, *Bayesian Statistics 4*, 763–773. Eds: Bernardo, J. M., David, A. P. and Smith, A. F. M., Oxford University Press, Oxford, U.K.

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

[73] Robins, G., Pattison, P., Kalish, Y. and Lusher, D. (2007), An Introduction to Exponential Random Graph (P^*) Models for Social Networks, *Social Networks*, **29**, 173–191.

[74] Rusinko III, A. and Lipkus, A. H. (1993), unpublished results obtained at Chemical Abstracts Service, Columbus OH.

[75] Schölkopf, B., Sung, K. K., Burges, C. J. C., Girosi, F., Niyogi, P., Poggio, T. and Vapnik, V. (1997), Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers, *IEEE Transactions on Signal Processing*, **45**, 2758–2765.

[76] Singhal, A. (2001), Modern Information Retrieval: A Brief Overview, *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, **24**, 35–43.

[77] Snijders, T. A. B. (2002), Markov Chain Monte Carlo Estimation of Exponential Random Graph Model, *Journal of Social Structure*, **3**(2).

[78] Stanton, D. T. (1999), Evaluation and Use of BCUT Descriptors in QSAR and QSPR Studies, *Journal of Chemical Information and Computer Sciences*, **39**, 11–20.

[79] Strauss, D. and Ikeda, M. (1990), Pseudolikelihood Estimation for Social Networks, *Journal of the American Statistical Association*, **85**, 204–212.

[80] Thomas, G. B. and Finney, R. L. (1996), *Calculus and Analytic Geometry*, 8th Edition, Reading, MA: Addison-Wesley.

[81] Tibshirani, R. J. (1996), Regression Shrinkage and Selection via the LASSO, *Journal of Royal Statistical Society: Series B*, **58**, 267–288.

[82] Tierney, L. and Kadane, J. B. (1986), Accurate Approximations for Posterior Moments and Marginal Densities, *Journal of the American Statistical Association*, **81**, 82–86.

[83] Tierney, L., Kass, R. and Kadane, J. B. (1989), Fully Exponential Laplace Approximations to Expections and Variances of Nonpositive Functions, *Journal of the American Statistical Association*, **84**, 710–716.

[84] Tipping, M. E. (2001), Sparse Bayesian Learning and the Relevance Vector Machine, *Journal of Machine Learning Research*, **1**, 211–244.

[85] Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer.

[86] Veropoulos, K., Campbell, C. and Cristianini, N. (1999), Controlling the Sensitivity of Support Vector Machines, in *Proceedings of the International Joint Conference on Artificial Intelligence*.

[87] Wang, Y. (2005), *Statistical Models for High Thoughput Screening Drug Discovery Data*, Ph.D. Thesis, Department of Statistics and Actuarial Science, University of Waterloo.

[88] Wasserman, S. S. and Robins, G. L. (2005), An Introduction to Random Graphs, Dependence Graphs, and p^*, in *Models and Methods in Social Network Analysis*, 148–161. Eds: Carrrington, J. S. P. and Wasserman, S. S., Cambridge University Press, Cambridge.

[89] Welch, W. J. (2002), *Computational Exploration of Data: Course Notes Fall 2002*, Department of Statistics and Actuarial Science, University of Waterloo.

[90] Zadrozny, B. and Elkan, C. (2001a), Learning and Making Decisions When Costs and Probabilities are Both Unknown, *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 204–213, ACM Press.

[91] Zadrozny, B. and Elkan, C. (2001b), Obtaining Calibrated Probability Estimates From Decision Trees and Naive Bayesian Classifiers, *Proceedings of the Eighteenth International Conference on Machine Mining*, 609–616, Morgan Kaufmann Publishers, San Francisco, CA.

[92] Zadrozny, B. and Elkan, C. (2002), Transforming Classifier Scores into Accurate Multiclass Probability Estimates, *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, 694–699, ACM Press.

[93] Zeger, S. L. and Karim, M. R. (1991), Generalized Linear Models With Random Effects; A Gibbs Sampling Approach, *Journal of the American Statistical Association*, **86**, 79–86.

[94] Zhu, M., Su, W. and Chipman, H. A. (2006), LAGO: A Computationally Efficient Approach for Statistical Detection, *Technometrics*, **48**, 193–205.