DATA MINING ALGORITHMS FOR CLASSIFICATION OF COMPLEX BIOMEDICAL DATA

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree
DOCTOR OF PHILOSOPHY

---

by
Liang Lan
January, 2013

Examining Committee Members:

Dr. Slobodan Vucetic, Advisory Chair, Department of Computer and Information
Dr. Longin Jan Latecki, Department of Computer and Information
Dr. Zoran Obradovic, Department of Computer and Information
Dr. Adam Davey, External Member, Department of Public Health

ABSTRACT

In my dissertation, I will present my research which contributes to solve the following three open problems from biomedical informatics: (1) Multi-task approaches for microarray classification; (2) Multi-label classification of gene and protein prediction from multi-source biological data; (3) Spatial scan for movement data.

In microarray classification, samples belong to several predefined categories (e.g., cancer vs. control tissues) and the goal is to build a predictor that classifies a new tissue sample based on its microarray measurements. When faced with the small-sample high-dimensional microarray data, most machine learning algorithm would produce an overly complicated model that performs well on training data but poorly on new data. To reduce the risk of over-fitting, feature selection becomes an essential technique in microarray classification. However, standard feature selection algorithms are bound to underperform when the size of the microarray data is particularly small. The best remedy is to borrow strength from external microarray datasets. In this dissertation, I will present two new multi-task feature filter methods which can improve the classification performance by utilizing the external microarray data. The first method is to aggregate the feature selection results from multiple microarray classification tasks. The resulting multi-task feature selection can be shown to improve quality of the selected features and lead to higher classification accuracy. The second method jointly selects a small gene set with maximal discriminative power and minimal redundancy across multiple classification tasks by solving an objective function with integer constraints.

In protein function prediction problem, gene functions are predicted from a predefined set of possible functions (e.g., the functions defined in the Gene Ontology). Gene function prediction is a complex classification problem characterized by the following aspects: (1) a single gene may have multiple functions; (2) the functions are organized in hierarchy; (3) unbalanced training data for each function (much less positive than negative examples); (4) missing class labels; (5) availability of multiple biological data sources, such as microarray data, genome sequence and protein-protein interactions. As participants in the 2011 Critical Assessment of Function Annotation (CAFA) challenge, our team achieved the highest AUC accuracy among 45 groups. In the competition, we gained by focusing on the 5-th aspect of the problem. Thus, in this dissertation, I will discuss several schemes to integrate the prediction scores from multiple data sources and show their results. Interestingly, the experimental results show that a simple averaging integration method is competitive with other state-of-the-art data integration methods.

Original spatial scan algorithm is used for detection of spatial overdensities: discovery of spatial subregions with significantly higher scores according to some density measure. This algorithm is widely used in identifying cluster of disease cases (e.g., identifying environmental risk factors for child leukemia). However, the original spatial scan algorithm only works on static spatial data. In this dissertation, I will propose one possible solution for spatial scan on movement data.

ACKNOWLEDGEMENT

This dissertation concludes my Ph.D. work at Temple University. To earn a Ph.D. degree has been one of the most significant challenges I have faced. Without the support, patience and guidance of the following people, my Ph.D. study would not have been completed. It is to them that I owe my deepest gratitude.

First and foremost, I want to express my sincere gratitude to my Ph.D. advisor Dr. Slobodan Vucetic. This dissertation would not have been a reality without his support. I appreciate all his contributions of time, ideas and funding during my Ph.D. study at Temple. I have benefited greatly from discussions with him and from his expertise in the research area of machine learning, data mining and bioinformatics. His insights and suggestions helped to shape my research skills. His valuable feedback contributed greatly to this dissertation.

I received great help from my Ph.D. committee, Dr. Adam Davey, Dr. Longin Jan Latecki and Dr. Zoran Obradovic. Thanks for their valuable comments which helped to improve the quality of the manuscript.

I gratefully acknowledge the NSF grant IIS-0546155 that made my Ph.D. work possible.

I collaborated and wrote papers with great people. Thank you Zhuang Wang, Kai Zhang, Haidong Shi, Nemanja Djuric, Yuhong Guo, Jun Liu and Fabian Moerchen,

I gratefully thank the students, faculty, and staff in the Department of Computer and Information Sciences. They have contributed immensely to my personal and professional growth at Temple. A special thanks to my friends in Department of

Computer and Information Sciences: Vuk Malbasa, Mihajlo Grbovic, Gregory Johnson, Vladimir Coric, Yi Jia, Lakesh Kansakar, Qiang Lou, Ping Zhang, Xingwei Yang, Shusha Li, Chengliang Wang, Weisi Duan, Norman Zeng, Tianyang Ma, Erkang Chen, Liang Du, Fei Huang, Yunsheng Wang, and forgive me if I forgot someone. It is them that made my life colorful in US.

I would like to express my deepest thanks to my family for their unconditional confidence in me and endless support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Feature Selection in Multi-Task Microarray Data

Microarray technology has the ability to simultaneously measure expression level of thousands of genes for a given biological sample. In microarray classification tasks, samples belong to one of several predefined categories (e.g., cancer vs. control tissues) and the goal is to classify a new tissue sample based on it microarray measurements. Typical microarray classification data set contains a very limited number of labeled examples, range from only a few to several hundred. Building a predictive classification model on such small-sample high-dimensional data sets is a challenging problem that has received an increasing attention from the research community. When faced with the small-sample high-dimensional data, most machine learning algorithms may produce an overly complicated model that perform well on training data but poorly in new data. To reduce the risk of over-fitting, we need to select a small number of genes before we apply classification model to microarray data. So, gene selection, as a special case of a well-known problem named feature selection (Guyon and Elisseeff, 2003) in machine learning community, becomes an essential technique in microarray classification.

The aim of feature selection in microarray data is in two-fold. On one hand, feature selection improves the classifier's generalization ability. On the other hand, the biologists are more interested in identifying biomarkers of disease than building an accurate classification model. So, the second purpose of feature selection is to provide an understandable model that indicates which genes are relevant to the classification

problem. Given a small subset of informative genes, it is easier for biologist to identify their biological relationship with the target diseases and gain deep scientific understanding of the problem.

Depending on how the feature selection process is combined with model learning process, feature section techniques can be organized into three categories: filter methods (Liu and Setiono, 1996), wrapper methods (Kohavi and John, 1997) and embedded methods (Guyon et al., 2002). The filter methods are independent of any learning algorithms. They select features by looking the intrinsic properties of the data. The relevance of each feature to the class label can be measured based on different criteria, such as mutual information (Torkkola, 2003) or statistical test (Golub and Slonim, 1999). Advantages of filter methods are: (1) The relevance score of each feature can be easily and efficiently computed. (2) It can easily scale up to very high-dimensional datasets. (3) It has better generalization power because it is separated from the classifier learning. Wrapper methods are coupled with classifier learning algorithms. In the wrapper methods, various subsets of feature are generated and the goodness of a specific subset of features is evaluated by the accuracy of resulting specific learnt classification model. The subset selection is known as NP-hard, so heuristic method such as forward selection and backward elimination are used to select the best possible gene subset. Compared to feature filter methods, the advantage of wrapper method is that they take into account the feature dependency. However, they can over-fit the train data and are computationally expensive when applied to high dimensional data. The embedded methods (Guyon et al., 2002; Tibshirani, 1995) incorporate feature selection as a part of the classifier training. So,

embedded methods are thus specific to a given learning algorithm, just like wrapper method.

Feature filter methods have much lower complexity and are less prone to over-fitting problem. So, practically, it is widely used in high-dimensional microarray data. The filter methods basically rank the genes based on their correlation with the class label and then select the top ranked genes. The correlation can be measured by statistic test (e.g., t-test), or information theoretic criteria like mutual information. The filter methods are easy to scale up to high dimensional data and can be used in conjunction with any supervising learning algorithm.

However, in many practical microarray classification problems, the size of given microarray data is particularly small (e.g., less than 10). In this case, even the most carefully designed feature selection algorithms are bound to underperform based on such a little amount of information. The best remedy is to borrow strength from external microarray datasets. For example, when learning to discriminate between infiltrative astrocytoma brain tumor and normal brain tissue using labeled microarray data obtained by a single lab, it might be beneficial to explore publicly available microarray data about other brain tumors or even non-brain tumors. In this dissertation, I will present two new multi-task feature filter methods which can improve the classification performance by utilizing the external microarray data. The first algorithm is presented in Chapter 2. This algorithm aggregates the feature selection results from multiple microarray classification tasks. The resulting multi-task feature selection can be shown to improve quality of the selected features and lead to higher classification accuracy. The second algorithm

presenting in Chapter 3 jointly selects a small gene set with maximal discriminative power and minimal redundancy across multiple classification tasks by solving an objective function with integer constraints.

1.2 Protein Function Prediction by Integrating Multiple Data Sources

Protein function prediction is a key challenge in the post-genomic era. Experimental determination of protein functions is accurate, but time-consuming and resource-intensive. With the advent of high-throughput technologies, a variety of large scale datasets are becoming available that can be very useful for protein function predictions, such as protein sequence data, gene expression data and protein-protein interactions data.

In protein function prediction problem, protein functions are predicted from a predefined set of possible functions (e.g., the functions defined in the Gene Ontology). Protein function prediction is a complex classification problem characterized by the following aspects: (1) a single protein may have multiple functions; (2) the functions are organized in hierarchy; (3) unbalanced training data for each function (much less positive than negative examples); (4) missing class labels; (5) availability of multiple biological data sources. The particular problem setting of the first 2 aspects is known as the Hierarchical Multi-Label Classification problem in machine learning research community. However, most of the research of Hierarchical Multi-Label Classification problem (Rouse et al., 2006) in machine learning is focused on the tree hierarchy. In protein function prediction, protein labels defined in Gene Ontology (GO) use more general label

hierarchies: a directed acyclic graph (DAG). Recent works on DAG-structured hierarchies include: Barutcuoglu and Troyanskaya, 2006, Vens et al., 2008 and Bi and Kwok 2011. Chen et al., 2010 proposed a method to address the protein function prediction problem in aspects (3) and (4). Their proposed approach tried to balance the data by enlarging positive samples and sampling the negative samples. As participants in the 2011 Critical Assessment of Function Annotation (CAFA) challenge, our team achieved the highest AUC accuracy among 45 groups. In the competition, we gained by focusing on the 5-th aspect of the problem. Thus, in the Chapter 4 of this dissertation, I will discuss several schemes to integrate the prediction scores from multiple data sources and show results. Interestingly, the experimental results show that a simple averaging integration method is competitive with other data integration methods.

1.3 Spatial Scan for Movement Data

Spatial scan statistics (Kulldorff et al., 1997) is used for detection of spatial overdensities: discovery of spatial subregions with significantly higher scores according to some density measure. This algorithm is widely used in identifying clusters of disease cases, such as identifying environmental risk factors for child leukemia (Kulldorff et al., 1997) and detection of bioterrorist attacks (Neill, 2004). It attempts to find spatial regions whose population has significantly higher disease risk than the background population. It currently has many variants (Kulldorff et al., 2005, Tango and Takahashi, 2005) appropriate for various types of data.

The original spatial scan only uses the populations' residence information as

location information for detecting the high risk region. This is a serious constraint, considering the mobility of populations at various temporal and spatial scales. At a short temporal scale, e.g., a single day, people typically spend significant time outside of their home doing activities such as working, commuting, or shopping. At a longer temporal scale, e.g., a decade or more, people typically change residences multiple times. Spatial scale can range from a person's movement within a home to air travel patterns of global population. Clearly, using only information about the current residence can be misleading because it ignores a multitude of environmental exposures that can occur away from home or which had occurred at previous residences.

In the chapter 5 of this dissertation, I present a novel algorithm which extends the original spatial scan to movement data by modeling the individual's disease risk using the output of logistic regression model. An efficient approach is proposed to compute the new designed test statistic.

1.4 Overviews by Chapters

The rest of the dissertation is organized as follows. Our main contributions are presented in Chapter 2, Chapter 3, Chapter 4 and Chapter 5. Chapter 2 presents a simple but effective multi-task feature selection filter which improves the microarray classification performance when applied in conjunction with both single task and multi-task classifiers. Chapter 3 presents the proposed multi-task feature selection by binary integer programming. The empirical results show the proposed algorithm gets better performance than other state-of-the-art feature selection algorithms. Chapter 4 presents

an algorithm used in CAFA challenge which predicts the protein function by integrating of different data sources. Chapter 5 presents our algorithm extends the original spatial scan to detect the over-density area based on movement data. Chapter 6 concludes the dissertation.

CHAPTER 2

A MULTI-TASK FEATURE SELECTION FILTER

2.1 Problem Definition

Let us define the target data set as $D = \{(x_i, y_i),\ i = 1\ldots N\}$, where x_i is an M-dimensional feature vector for i-th example, y_i is its class label, and N is the number of labeled examples. We can assume that D is a random sample from an underlying distribution defined by feature distribution $p(x)$ and conditional distribution $p(y \mid x)$. The target task is defined as training a classifier $f(x)$ that approximates the conditional distribution $p(y \mid x)$. In single-task learning, only the target data D are used for training.

In multi-task learning, we have access to the additional K auxiliary data sets. Let us denote the k-th auxiliary data set as $D_k = \{(x_i^{(k)}, y_i^{(k)}),\ i = 1\ldots N_k\}$, where $x_i^{(k)}$ and $y_i^{(k)}$ are feature vector and target label for i-th example in D_k, and N_k is the size of D_k. We can assume that D_k is a random sample from feature distribution $p_k(x)$ and conditional distribution $p_k(y \mid x)$. The goal of multi-task learning is to improve accuracy of the target classifier by exploiting the auxiliary data.

Auxiliary data can be particularly useful when the target data set is small. Clearly, the usefulness of an auxiliary data set depends on its similarity to the target data distribution. Interestingly, even when the similarity is only moderate, it has been demonstrated that the auxiliary tasks can boost the classification accuracy on the target task. It is also worth noting that multi-task learning can sometimes lead to negative transfer, where use of auxiliary data decreases accuracy on the target task.

2.2 Overview of Single-Task and Multi-Task Classifiers

In this section, we will give an overview of 3 standard single-task classification algorithms: Penalized Logistic Regression (PLR), Lasso, and Support Vector Machines (SVM). We also outline several previously proposed extensions of these algorithms for multi-task classification

2.2.1 Penalized Logistic Regression

Let us consider for simplicity only binary classification where target variable can have one of two values, $y \in \{0, 1\}$. In logistic regression, the posterior probability of positive class can be written as a logistic sigmoid of a linear function of the feature vector, $p(y = 1 \mid x) = \sigma(w^{\mathrm{T}}x)$, where w is the weight vector to be learned, and the logistic sigmoid is defined as $\sigma(z) = 1/(1+exp(-z))$.

The weight vector w can be estimated by maximizing the log-likelihood l_D of the training data set expressed as

$$\begin{aligned} l_D(w) &= \log p(D \mid w) = \sum_{i=1}^{N} \log p(y_i \mid x_i, w) = \\ &= \sum_{i=1}^{N} y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i)). \end{aligned} \tag{2.1}$$

Therefore, the maximum likelihood estimate for w is given as

$$w_{ML} = \arg\max_w l_D(w). \tag{2.2}$$

To reduce over-fitting, we can introduce a prior distribution $p(w)$ for the weight vector w. By conveniently setting the prior to be multivariate Gaussian $p(w) = N(w \mid \mu, \Sigma)$, we could obtain the maximum a posteriori (MAP) weight estimate as

$$w_{MAP} = \arg\max_w [\log p(D \mid w) + \log p(w)] =$$
$$= \arg\max_w [l_D(w) - \frac{1}{2}(w-\mu)^T \Sigma^{-1}(w-\mu)]. \quad (2.3)$$

The second term is the regularization term that penalizes any deviation of weights w from their desired values μ. Thus the name penalized LR for any training algorithm that estimates $w_{MAP.}$ Evidently, the choice of weight prior hyperparameters μ and Σ is very important for the success of penalized LR.

Single-Task Penalized LR (S_LR)

In single-task scenarios, there are typically two approaches to selection of prior hyper-parameters μ and Σ. The most common is to use the penalized term solely for the purpose of regularization. Typically, one would choose $\mu = 0$ and $\Sigma = \sigma^2 I$, where σ is scalar that determines how strongly the weights w are tied to zero and where I is the identity matrix. As for the choice of σ, it could be determined using cross-validation or by the more sophisticated empirical Bayes procedure. We call the resulting approach *the single-task penalized LR* and denote it as S_LR.

Regardless of its simplicity, the S_LR is regarded as quite successful in over-fitting prevention. However, it is also worth noting that its prior oversimplifies the rich structure among the features. It assumes that the weights are independent of each other and have equal uncertainty. This approach is likely to result in poor performance when the training data are extremely scarce ($N<<M$), as is the case with microarray datasets.

The alternative approach is to select hyper-parameters based on some prior belief about the weight values. This second approach motivates the multi-task penalized LR described

next.

Multi-Task Penalized LR (M_LR)

Recent work in multi-task learning resulted in many algorithms that define the prior hyper-parameters to ensure that the predictors specialized on similar tasks have similar weights (Marx et al., 2008; Raina et al., 2006). One of the simplest, yet effective, approaches of this type is to train a logistic regression model on each of the *K* auxiliary data sets and to use the resulting weight vectors w^k, $k = 1 \ldots K$, to determine the hyper-parameters μ and Σ. In this paper, we applied the approach of (Marx et al., 2008) that calculates μ as the average weight of auxiliary classifiers and Σ as the diagonal matrix whose elements $\Sigma_{j,j} = \sigma_j^2$ are weight variances,

$$\mu = \frac{1}{K}\sum_{k=1}^{K} w^k, \quad \sigma_j^{\,2} = \frac{1}{K-1}\sum_{k=1}^{K}(w_j^k - \mu_j)^2. \tag{2.4}$$

Multi-Task Reweighted LR (RW_LR)

The reweighting approaches for multi-task learning train the target classifier using a mixture of target examples and examples from the auxiliary data sets. Typically, examples from the target data set are given larger weights than those from the auxiliary data sets. The weights for each example can be calculated in various ways. Given the example weights, the logistic regression model is obtained as

$$w_{RW} = \arg\max_{w} [\sum_{i=1}^{N_{tot}} r_i \log p(y_i \mid x_i, w) + \log p(w)], \tag{2.5}$$

where N_{tot} denotes all examples from the target and the auxiliary data sets and r_i the example weights.

The weight for each example r_i can be estimated using the approach proposed by Bickel et al., (2008). In their approach weights of examples are obtained as the outputs of logistic regression model trained to discriminate between target examples (denoted as positive examples) and examples from an auxiliary task (denoted as negative examples). Therefore, large overlap between the distributions of the target and the auxiliary data implies the large weights of the auxiliary examples.

2.2.2 Lasso

Single Task Lasso (S_Lasso)

Lasso (Tibshirani, 1996) is a popular embedded feature selection technique. It does feature selection automatically by solving the l_1-regularized learning problem. If used in conjunction with the logistic regression, the optimal w is obtained by maximizing the regularized log-likelihood

$$w_{S-Lasso} = \arg\max_w [\log p(D \mid w) - \lambda \| w \|_1], \tag{2.6}$$

where λ is a regularization coefficient. This optimization problem can also be explained as regularized logistic regression with Laplacian prior (Krishnapuram et al., 2005; Cawley and Talbot, 2006).

Due to the nature of l_1 penalty, solving this objective problem typically results in a sparse model where most of the weights become zeros. Owing to the sparse solution of Lasso, it becomes a very useful embedded feature selection approach.

Multi-Task Lasso (M_Lasso)

A regularization scheme for multi-task feature selection was proposed by (Obozinski et al., 2009). It aims to find a common set of features that are relevant to all available classification tasks. This is achieved by defining a regularization term as an l_1 sum of the l_2 norms of the feature-specific vectors. The optimization problem for multi-task Lasso can be represented as

$$w_{M-Lasso} = \arg\max_w [\sum_{k=1}^{K} \log p(D^k \mid w^k) - \lambda \sum_{j=1}^{M} \left\| w_j \right\|_2], \tag{2.7}$$

where K is the number of tasks, M is the number of features, w_j is a column vector that represents the coefficients of features j across all K tasks and w^k is a row vector that represents parameters of the learned model for classification task k. The solution is a KxM matrix containing the learned parameter for all tasks. This l_1/l_2 regularization scheme selects feature jointly across all considered classification tasks. The l_2 norms measure the overall relevance of a particular feature while the l_1 sum enforces feature selection. It encourages similar sparsity patterns for all task-specific models. Note that this l_1/l_2 regularization scheme reduces to l_1-regularization if the number of tasks is reduced to one.

2.2.3 Support Vector Machine

Single-Task Support Vector Machine (S_SVM)

Support Vector Machines (Vapnik, 1995) (SVM) are among the most powerful and popular classification algorithms. The SVM is solving the following problem

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} \quad y_i(w^T x_i + b) \geq 1-\xi_i, \xi_i \geq 0, \end{aligned} \tag{2.8}$$

where w is the weight vector, b is the bias term, ξ_i are slack variables, and C is the slack parameter. The resulting predictor has the form

$$f(x) = w^T x + b = \sum_{i=1}^{N}\alpha_i y_i (x_i^T \cdot x) + b, \tag{2.9}$$

where $0 \leq \alpha_i \leq C$ ($\alpha_i \neq 0$ corresponds to the support vector). If needed, this linear classifier can be easily converted to a nonlinear classifier by kernelization, i.e., by replacing the dot product $x_i^T x$ with an appropriately selected kernel function $K(x_i,x)$.

Multi-Task Reweighted Support Vector Machine (RW_SVM)

The reweighting approach introduced in Section 3.1.2 can be used to design a multi-task SVM. Knowing the weight r_i of each example from the target and auxiliary tasks, we use the idea of Fuzzy SVM (Lin and Wang, 2002) to train the multi-task SVM through solving the following optimization problem

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} r_i\xi_i \\ &\text{subject to} \quad y_i(w^T x_i + b) \geq 1-\xi_i, \xi_i \geq 0. \end{aligned} \tag{2.10}$$

The effect of reweighting is equivalent to changing the value of the slack parameter from C for all examples to $C \cdot r_i$. The resulting predictor has the same form as the original SVM, with only difference that $0 \leq \alpha_i \leq C \cdot r_i$. Therefore, examples with smaller weight r_i will have smaller influence on the resulting predictor. To determine weights r_i, we can use the approach by (Bickel et al., 2008).

2.3 A Simple Multi-Task Feature Selection Filter

In this section we describe our proposed multi-task feature selection filter which is suitable for microarray data. The main property of filters is that they are very efficient and do not require training a classifier. In general, feature x_j, $j = 1 \ldots M$, is deemed useful by a filter if its class-conditional distributions $p(x_j \mid y = 1)$ and $p(x_j \mid y = 0)$ are different. A standard approach to measure the difference is to run a statistical test of the null hypothesis that the two distributions are identical (Radivojac et al., 2004). We first summarize this standard approach in the single-task scenario.

Single-task feature selection filter (S_FS). Let us consider data set $D = \{(x_i, y_i), i = 1 \ldots N\}$ and denote $X_j^+ = \{x_{ij}, y_i = 1\}$ and $X_j^- = \{x_{ij}, y_i = 0\}$ as the values of j-th feature in positive and negative examples, respectively. Let us denote $\theta_j = \theta(X_j^+, X_j^-)$ as a statistic measuring the difference between the two samples. Some examples of possible θ statistics are the difference in sample means, information gain, and χ^2 statistics. In the statistical filter, we ask the question of how likely it is that value θ_j or bigger occurs under the null hypothesis H_0: $p(x_j \mid y = 1) = p(x_j \mid y = 0)$. This is exactly the definition of the p-value of the statistical test. We denote the p-value for j-th feature as $p_j = p(\theta_j^0 \geq \theta_j)$, where θ_j^0 is the statistics under the null hypothesis. The p-values can be obtained analytically for some standard statistics such as the difference of means, while the permutation test can be used for other statistics such as the information gain.

Given the p-values, features can be selected in two ways. In first, all features with p-values below some threshold (e.g. $p_j < 0.05$) are selected. In second, M^* features with the

smallest p-values are selected. In both cases, it is not clear what threshold is appropriate with respect to the resulting classification accuracy and a common practice is to use validation to determine it.

Multi-task feature selection filter (M_FS). When labeled data set D is very small, even feature selection filters can become highly inaccurate. This is because the power of statistical tests to discriminate between useful and irrelevant features drops with the sample size. The consequence is that a large number of irrelevant features would be selected and that sizeable number of relevant features would not, regardless of threshold choice. The remedy proposed here is to use multi-task feature selection filters. The idea is to first determine p-values of features in each auxiliary data set and to integrate those values for target task feature selection.

Let us consider k-th auxiliary data set $D_k = \{(x_i^{(k)}, y_i^{(k)}),\ i = 1 \ldots N_k\}$. We propose to calculate p_{jk} as p-value of j-th feature in D_k. By repeating the procedure for each auxiliary task, each feature could be represented by a K-dimensional vector $Q_j = \{p_{jk},\ k = 1 \ldots K\}$. The following are four simple strategies to determine significance q_j of j-th feature from Q_j: (1) $q_j = min(Q_j)$; (2) $q_j = mean(Q_j)$; (3) $q_j = median(Q_j)$; (4) $q_j = max(Q_j)$. The selected M^* features are those with the smallest q_j values. The proposed strategies have slightly different objectives – $min(Q_j)$ favors features deemed very significant by any of the auxiliary tasks; $mean(Q_j)$ and $median(Q_j)$ prefer features that are most significant in average; and $max(Q_j)$ prefers features significant on all auxiliary tasks.

Related to the choice of M^*, the multi-task filter determines it by validation of single-task classifiers on auxiliary tasks that use single-task filters. This approach is expected to

be more powerful than an arbitrary selection, or doing validation on the target data set with only a few labeled examples.

The proposed filter does not consider the similarity between auxiliary and target tasks and treats each auxiliary task as equally useful for feature selection. In practice, some auxiliary tasks are more similar to the target task than others. Therefore, our multi-task filter provides an option to use only a subset $K^* < K$ of the auxiliary data sets in feature selection. There, only p_{jk} values from the K^* auxiliary data sets are used in Q_j. The practical issue is that the target data set can be too small to aid in detection of unrelated auxiliary tasks.

To resolve this issue, we propose to measure accuracy of single-task auxiliary classifiers on target task examples and to select K^* auxiliary tasks as those with the most accurate classifiers. To measure accuracy, we use the log-likelihood as defined in (2.1). To select the best value of K^* parameter, while avoiding the danger of over-learning from the target data set, we validate K^* on auxiliary data sets.

Kruskal-Wallis test. Among many alternatives, we used the nonparametric Kruskal-Wallis test to determine feature *p*-values. The test is very appropriate for microarray data because it does not require strong distributional assumptions, it works well on small samples, and its *p*-values can be calculated analytically. Given positively and negatively labeled values of *j*-th feature, X_j^+ and X_j^-, Kruskal-Wallis sorts the values and calculates the average rank of positive and negative labels. Then, it calculates test statistic kw_j that becomes large when the average ranks deviate from the expected rank $(N+1)/2$. The p-value of the statistics is calculated easily because kw_j follows the standard χ^2 distribution.

Table 2.1. Data set (cancer:normal cases)

Bladder (11:7)	Lung (20:7)	Prostate (14:9)
Breast (17:5)	Ovary (15:3)	Renal (11:13)
Colon (15:11)	Pancreas (11:10)	Uterus (11:6)

2.4 Empirical Results

2.4.1 Data Description and Experimental Setup

Data sets used to evaluate the proposed algorithms were published in (Ramaswamy et al., 2001) and we downloaded them in the pre-processed form from (Statnikov et al., 2005). They were obtained using Affymetrix microarrays that measured expression of M = 15,009 genes. The original data corresponded to multiple samples from 14 human cancer tissues and 12 normal tissues. From that data, we extracted 9 binary classification data sets by coupling normal and cancer samples from the same tissue type, whenever available. The summary of these 9 data sets in Table 2.1 indicates that all of them are small and that some of them are very imbalanced with just a few samples from normal tissues.

We evaluated several feature selection methods (see Section 2.3) in our experiments: (1) Single-task (S_FS), (2) Multi-task (M_FS), and (3) Random selection (R_FS). In M_FS algorithm, we used the p-value aggregation based on minimum p-value, $q_j = min(Q_j)$. This approach proved slightly better than median and significantly better than maximum in our preliminary studies. The feature selection algorithms were evaluated in conjunction with different classification algorithms (see Section 2.2): (1) Single task

Lasso (S_Lasso); (2) Multi-task Lasso (M_Lasso); (3) Single task SVM (S_SVM); (4) Multi-task reweighted SVM (RW_SVM); (5) Single task penalized LR (S_LR); (6) Multi-task penalized LR (M_LR); (7) Multi-task reweighted LR (RW_LR). In S_LR we used hyperparameter $\sigma = 1$. To train a logistic regression model, we used the conjugate gradient ascent method because it was much faster on high-dimensional microarray data than the more common Newton method (Minka, 2003). For S_Lasso and M_Lasso, we used TL_BBLasso algorithm implemented in UC Berkeley Transfer Learning Toolkit (Rakhlin, 2007). We used $\lambda = 1$ in both the S_Lasso and M_Lasso algorithms. For training the S_SVM and RW_SVM classifiers, we used $C = 1$. For each combination of feature selection and data mining algorithms, we selected one of the 9 data sets as target data and the remaining 8 as the auxiliary data. To explore the influence of training size on transfer learning algorithms, we used $N^+ = \{1, 2, 3, 4, 5\}$ positive and the same number of negative target task examples for training. We used balanced training data to facilitate result interpretation. The remaining examples were used for testing. Due to lack of normal examples, in breast data set we used $N^+ = \{1, 2, 3, 4\}$ and for ovary data set $N^+ = \{1, 2\}$. The reported accuracy was calculated as average between accuracy on positive and negative test examples, $Acc = Acc^+/2 + Acc^-/2$, where $Acc^+ = TP/(TP + FN)$, $Acc^- = TN/(TN + FP)$. For each choice of N^+ we repeated experiments 10 times, each with different randomly selected N^+ positive and $N^- = N^+$ negative examples from the target data. Thus, we built and tested 50 (except for breast and ovary data) classifiers for each feature selection-classifier combination. All experiments were repeated 9 times so that each cancer data set was used as target data set in one set of experiments.

Table 2.2. Win:loss for M^* = 100 vs. other M^*

M^* =	50	500	5,000	15,009
M^* = 100	289:121	216:194	259:141	269:141

Table 2.3. Win:loss for K^* = 5 vs. other K^*

K^* =	0	1	3	8
K^* = 5	241:169	251:159	219:191	216:194

2.4.2 Experimental Results

Choice of number of features M^*. We trained S_LR classifiers using M^* = {50, 100, 500, 5000, M} features selected by S_FS. Following the setup from Section 2.4.1, we trained and tested 410 classifiers for each choice of M^*. Table 2.2 summarizes how many times S_FS + S_LR classifier with M^* = 100 was more accurate than the same classifier with different choice of M^*. M^* = 100 was better than the alternative values.

Choice of number of auxiliary tasks K^*. We determined suitable number of auxiliary tasks for feature selection following the approach explained in Section 2.4.1. In Table 2.3 we compare *win*:*loss* statistics of M_FS + S_LR classifier with K^* = 5 to the same classifier with K^* = {0, 1, 3, 8} (K^* = 0 corresponds to S_FS + S_LR classifier). Although the differences were rather small, K^* = 5 seemed to be the best overall.

Comparing feature selection algorithms. In Figure 2.1 we compare performance of three different feature selection methods, R_FS, S_FS, M_FS, all using 100 selected features. Figure 2.1 a-g shows average accuracies of 7 different classifiers trained with

$N^- = N^+ = 2$ target examples and using 100 selected features. Each of the 9 sets of bars represents a case when a given data set is used as the target task and the remaining data sets as the auxiliary tasks. It could be seen from all seven figures that multi-task feature selection (M_FS) results in superior accuracies. There are only two cases (in S_Lasso experiments) where M_FS was outperformed and several other cases where M_FS performed similar either to S_FS or R_FS. Comparing R_FS and S_FS, it is hard to conclude which one is better. This clearly indicates that 4 training examples are not sufficient for a successful single-task feature selection. On the other hand, the success of M_FS demonstrates usefulness of the idea of borrowing strength from the auxiliary tasks.

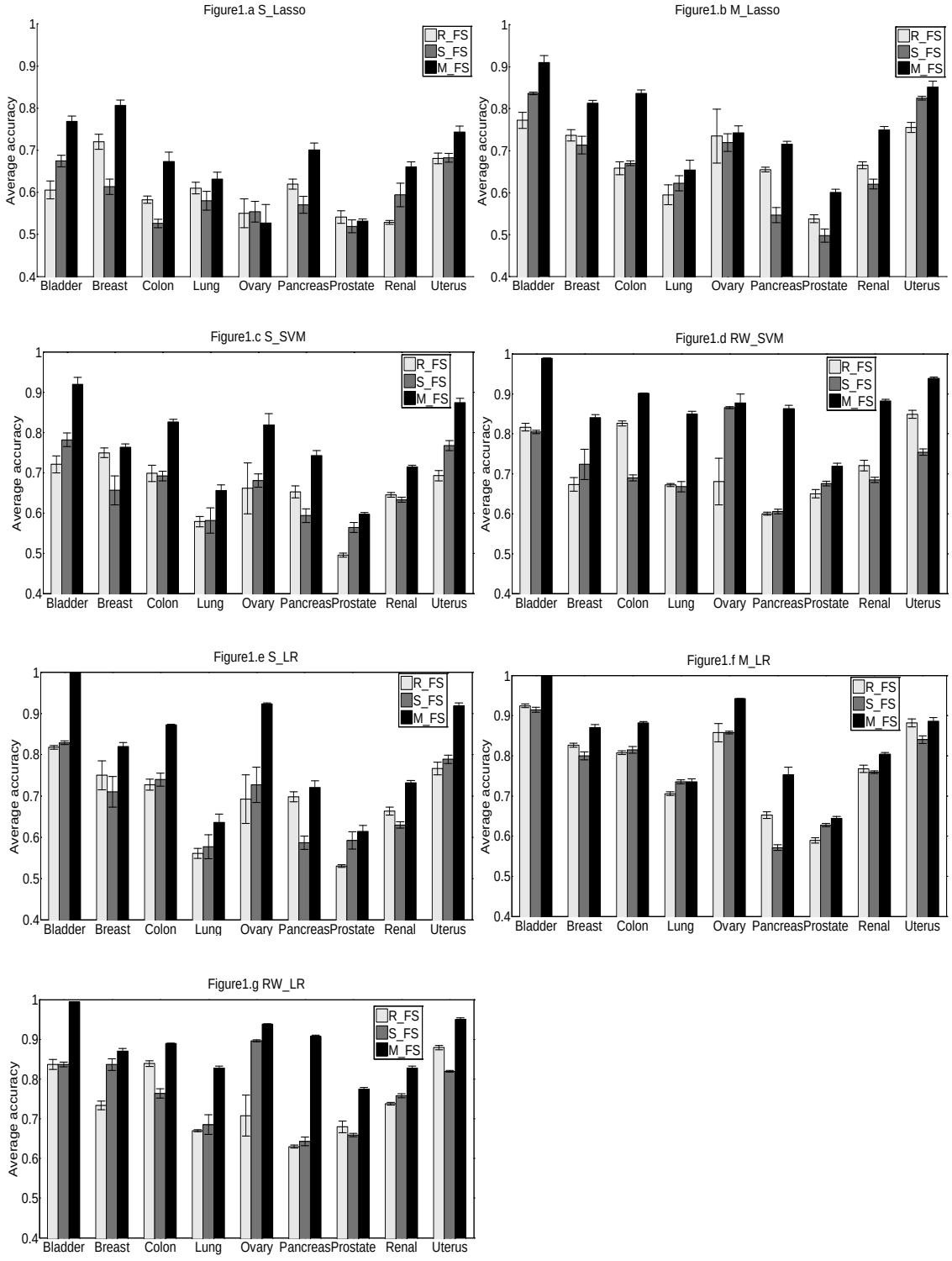

Figure 2.1.a-g. Accuracies of 3 FS algorithms coupled with a) S_Lasso, b) RW_Lasso, c) S_SVM, d) RW_SVM, e) S_LR, f) M_LR and g) RW_LR.

It is worth observing that results in Figure 2.1.b, d, f, and g correspond to different multi-task learning algorithms. Compared the obtained accuracies to the corresponding single-task algorithms, it can be seen that they indeed increase the classification accuracy. While multi-task algorithms are superior to their single-task counterparts, an important conclusion is that the proposed multi-task feature selection filter can further improve their performance. Therefore, the multi-task filter was a very useful preprocessing step for both single and multi-task learning algorithm.

Accuracy vs. training size. In Figure 2.2 we compare accuracies of four representative logistic regression algorithms as a function of the training size on each target task using 100 selected features. S_LR + S_FS is purely single-task algorithm, S_LR + M_FS is combination of single-task classifier and multi-task feature selection, and the remaining two are purely multi-task. It can be seen that S_LR + S_FS was inferior to the other algorithms and that the difference in accuracy was particularly large for small target training sizes. The difference between the 3 multi-task algorithms is rather small, with RW_LR + M_FS being most accurate overall. Interestingly, S_LR + M_FS is a very competitive combination of single-task classifier and multi-task feature selection. When $N^- = N^+ = 5$, the difference between the four algorithms decreased and, in few cases, we could even observe slight negative transfer for the multi-task algorithms. In one particular case (prostate cancer as the target task) S_LR + S_FS was better than S_LR + M_FS. The reason might be that the biological mechanism and biomarkers of prostate cancer are very different from other 8 cancers.

In Figure 2.3, we compared 3 different linear SVM-based classifiers where the

number of selected features was 100. The results are similar to those in Figure 2.3. Overall, RW_SVM + M_FS was the most accurate model while S_SVM + S_FS was the least accurate. The negative transfer also can be seen when prostate cancer is the target task.

In Figure 2.4, we evaluated 3 algorithms based on Lasso, where the number of selected features was 100. It can be concluded that the M_FS filter increases the accuracy of both S_Lasso and M_Lasso. Overall, the M_Lasso + M_FS achieved the best results. Because Lasso results in sparse predictors, the number of final features used in the logistic regression model is actually less than 100. We found that the number of features used in S_Lasso model was in the range between 1 and 15 and that the number of features used in M_Lasso model was in the range between 20 and 40.

Comparing different classification algorithms. In Figure 2.5, we compare performance of three different multi-task learning algorithms with 100 M_FS selected features. RW_LR is the most accurate on 6 data sets, while RW_SVM is the best on the remaining 3 data sets. Both algorithms are significantly more accurate than M_Lasso. The difference between the 3 algorithms decreases with the training size.

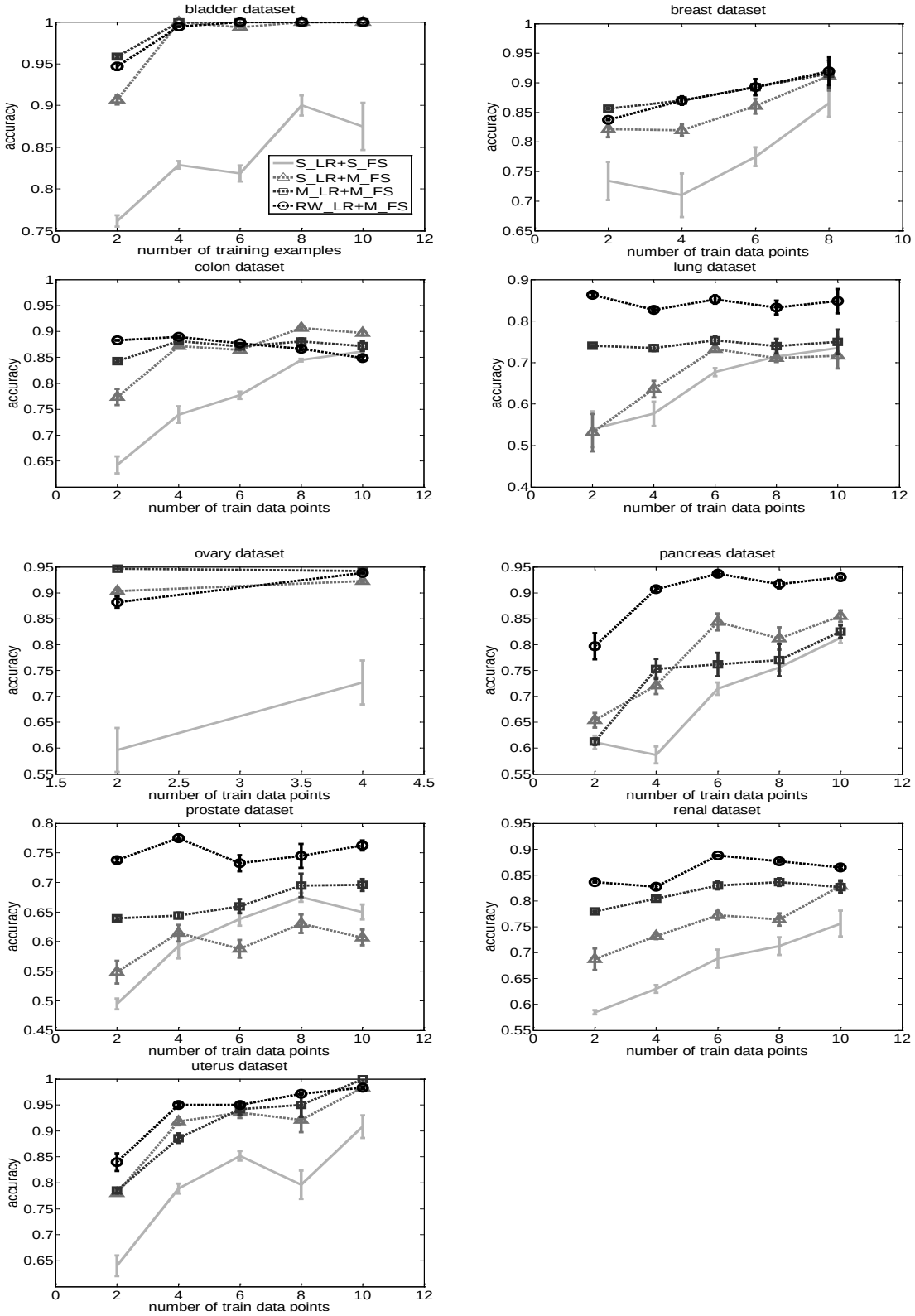

Figure 2.2 Comparison of 4 LR based classifiers on 9 target tasks for varying training sizes.

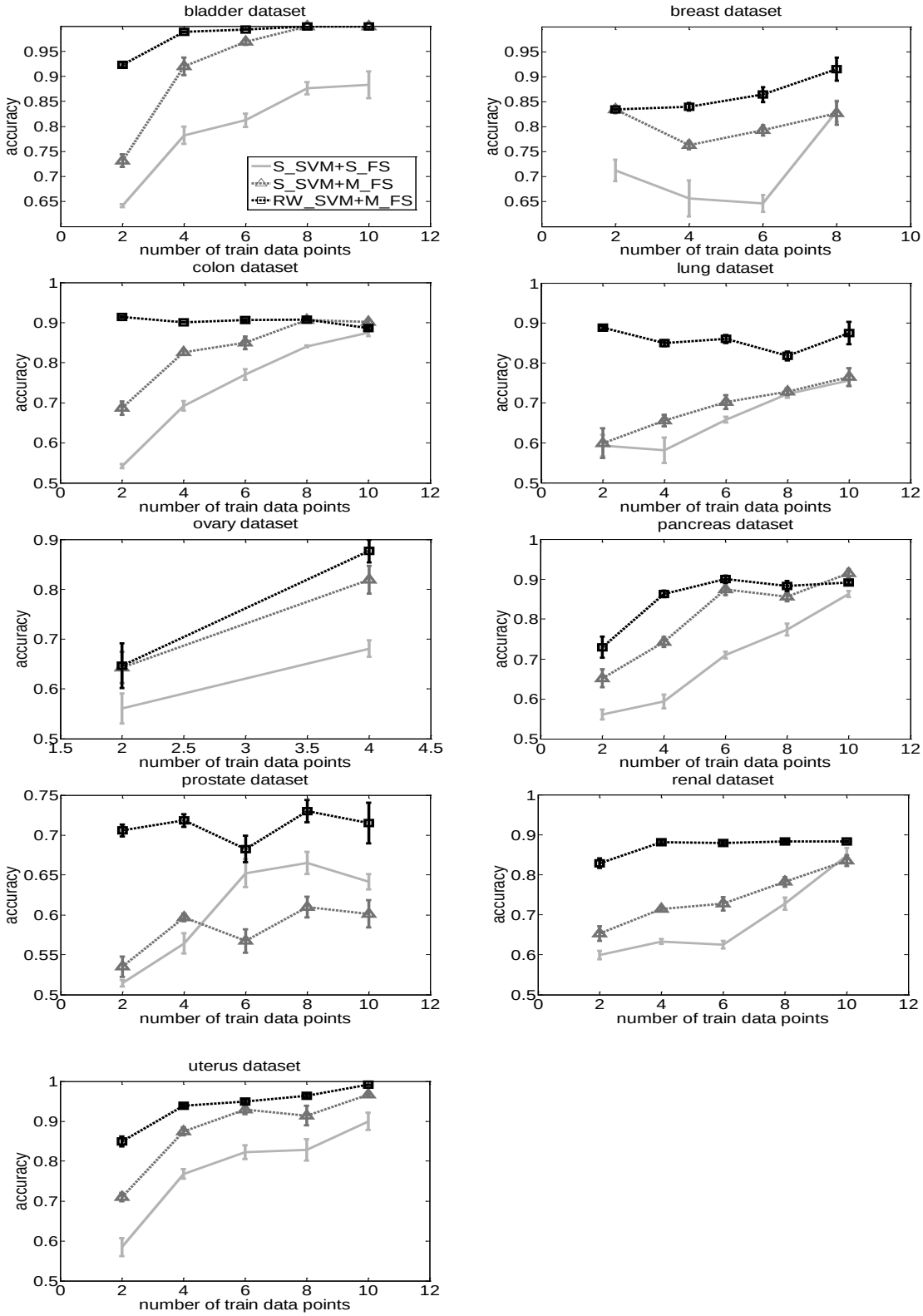

Figure 2.3 Comparison of 3 SVM based classifiers on 9 target tasks for varying training sizes.

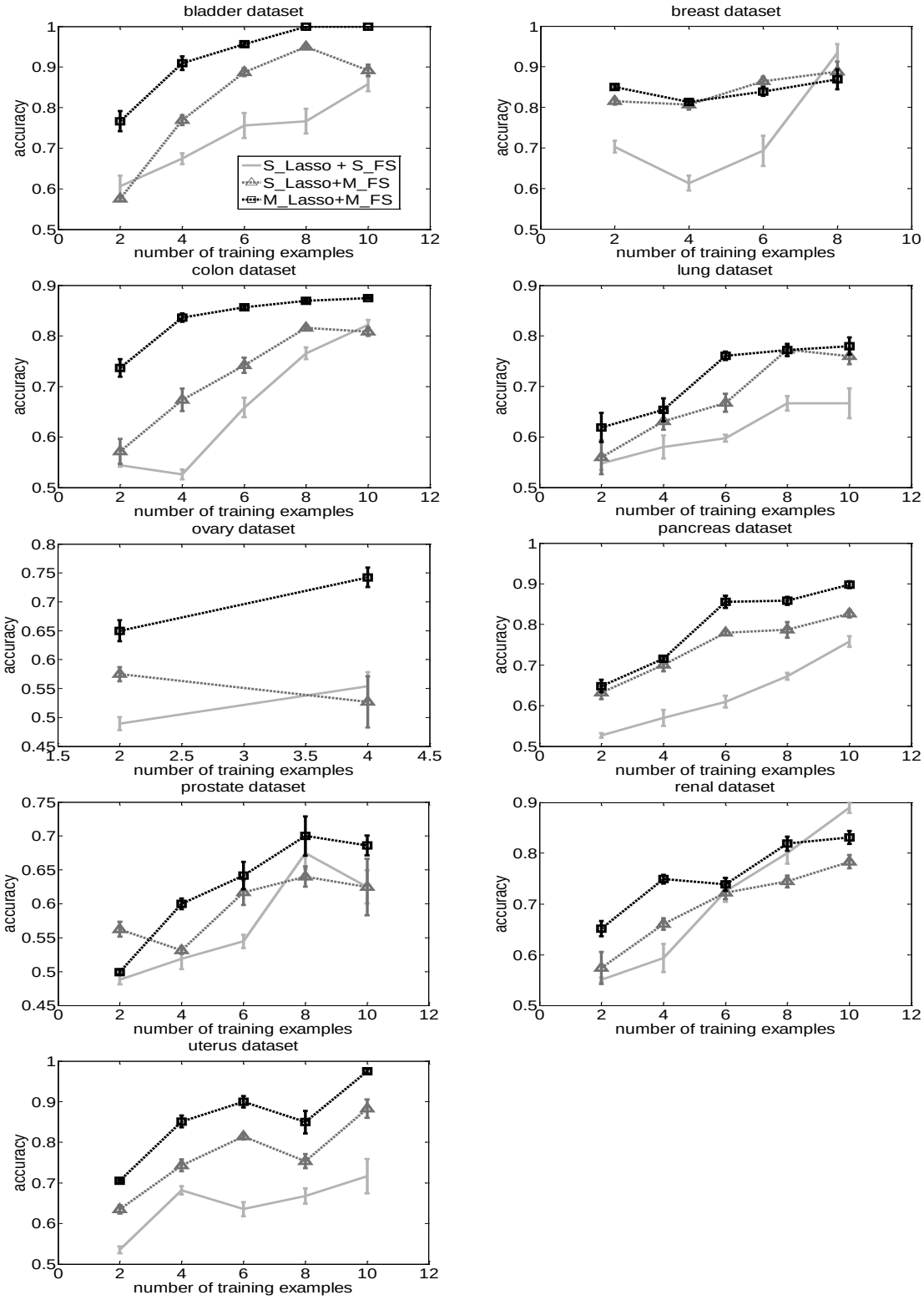

Figure 2.4 Comparison of 3 Lasso based classifiers on 9 target tasks for varying training sizes.

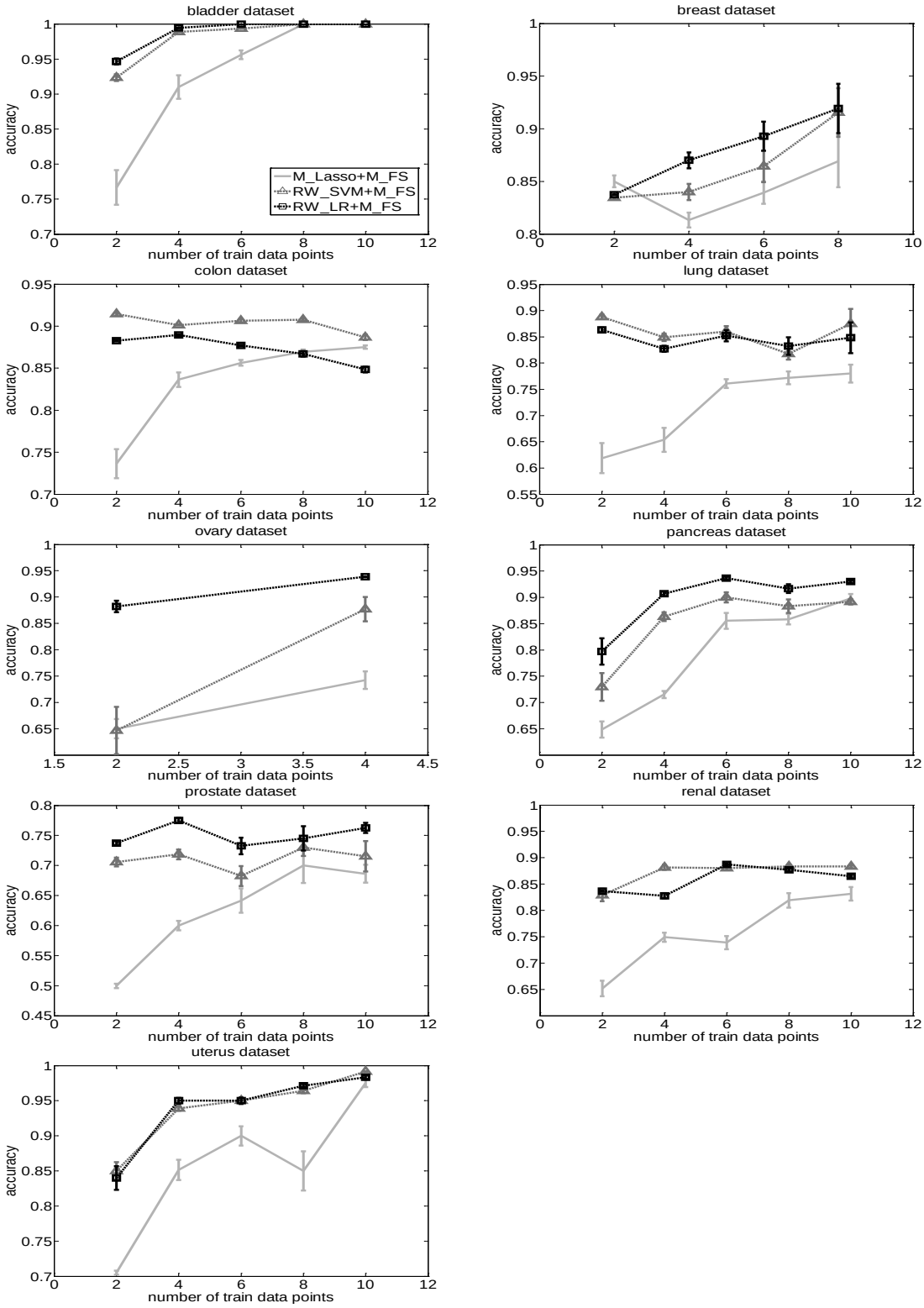

Figure 2.5 Comparison of Multi-Task version of Lasso, SVM, LR classifiers on 9 target tasks for varying training sizes.

2.5 Conclusions

Multi-Task Learning is a very attractive technology for microarray classification because of the availability of rich public repositories of microarray and related data. In this chapter, we presented a multi-task feature selection filter suitable for microarray classification. The filter can be used as a pre-processing step for an arbitrary classification algorithm. Experimental results indicate that the proposed filter boosts accuracy of both single-task and multi-task classifiers. Combination of multi-task feature selection and classification appears particularly successful. We observed that multi-task learning is the most useful when target examples are scarce. Its benefits decrease with data size and could even lead to negative transfer. Despite its limitations, it is evident that multi-task learning can be a useful bioinformatics tool in many biological problems.

CHAPTER 3

MULTI-TASK FEATURE SELECTION BY BINARY INTEGER PROGRAMMING

3.1 Introduction

Traditional filter methods rank the features based on their correlation with the class label and then select the top ranked features. The correlation can be measured by statistic tests (e.g., t-test) or by information-theoretic criteria such as mutual information. The filter methods easily scale up to high dimensional data and can be used in conjunction with any supervised learning algorithm. However, because the traditional filter methods access each feature independently, highly correlated features tend to have similar rankings and tend to be selected jointly. Using redundant features could result in low classification accuracy. As a result, one common improvement for filter methods is to reduce redundancy between selected features. For example, minimal-redundancy-maximal-relevance (mRMR) proposed by Peng et al. (2005) selects the feature set with both maximal relevance to the target class and minimal redundancy among the selected feature set. Because of the high computational cost of considering all possible feature sets, the mRMR algorithm selects features greedily, minimizing their redundancy with features chosen in previous steps and maximizing their relevance to the target class.

A common critique of popular feature selection filters is that they are typically based on relatively simple heuristics. To address this concern, recent research resulted in more principled formulation of feature filters. For example, algorithms proposed in (Lujan et al., 2010) and (Liu et al., 2011) attempt to select the feature subset with

maximal relevance and minimal redundancy by solving a constrained quadratic optimization problem (QP). The objective used by Lujan et al. (2010) is a combination of a quadratic term and a linear term. The redundancy between feature pairs is measured by the quadratic term and the relevance between features and class label is measured by the linear term. The features are ranked based on a weight vector obtained by solving a QP problem. The main limitation of this method is that the relevance between a feature and the class label is measured by either Pearson correlation or mutual information. However, Pearson correlation assumes normal distribution of the measurements, which might not be appropriate to measure correlation between numerical features and binary target. The mutual information requires using discrete variables and is sensitive to discretization. The objective used by Liu et al. (2011) contains only one quadratic term. This quadratic term consists of two parts: one measures feature relevance using mutual information between features and the class label, and another measures feature redundancy using mutual information between each feature pair. However, the square matrix in the proposed quadratic term is not positive semidefinite. Thus, the resulting optimization problem is not convex and could result in poor local optima.

In this chapter, we propose a novel feature filter method to find the feature subset which maximizes the inter-class separability and intra-class tightness, and minimizes the pair-wise correlations between selected features. We formulate the problem as a quadratic programming with binary integer constraints. For high dimensional microarray data, to solve the proposed quadratic programming problem with binary integer constraints requires high time and space cost. Therefore, we relax binary integer constraints and

apply the low rank approximation to the quadratic term in the objective function. The resulting objective function can be efficiently solved to obtain a small subset of features with maximal relevance and minimal redundancy.

In many real-life microarray classification problems, the size of the given microarray dataset is particularly small (e.g., we might have less than 10 labeled high-dimensional examples). In this case, even the most carefully designed feature selection algorithms are bound to underperform. Probably the only remedy is to borrow strength from external microarray datasets. Recent research (Argyriou et al., 2008; Obozinski et al., 2010) illustrates that multi-task feature selection algorithms can improve the classification accuracy. The multi-task feature selection algorithms select the informative features jointly across many different microarray classification data sets. Following this observation, we extend our feature selection algorithm to the multi-task microarray classification setup.

The contributions of this chapter can be summarized as follows.(1) We propose a novel gene filter method which can obtain a feature subset with maximal discriminative power and minimal redundancy; (2) The globally optimal solution can be found efficiently by relaxing the integer constraints and using a low-rank approximation technique; (3) We extend our feature selection method to multi-task classification setting; (4) The experimental results show our algorithms achieve higher accuracy than the existing filter feature selection methods, both in single-task learning and multi-task settings.

3.2 Feature Selection by Binary Integer Programming

Let us denote the training dataset as $D = \{\mathbf{x}_{i,} y_i\}_{i=1}^{N}$, where $\mathbf{x}_i$ is an M dimensional feature vector for the i-th example and y_i is its class label. N is the number of training examples. Our objective is to select a feature subset that is strongly predictive of class label and has low redundancy. We introduce a binary vector $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$ to indicate which features are selected:

$$w_j = \begin{cases} 1 \ \textit{if feature } j \textit{ is selected} \\ 0 \ \textit{if feature } j \textit{ is not selected} \end{cases} \tag{3.1}$$

So, the new feature vector for the i-th example after feature selection can be represented as $\mathbf{g}_i = \mathbf{x}_i \circ \mathbf{w}$, where the symbol $\circ$ denotes the pairwise product. Therefore, $g_{ij} = x_{ij}$, for $w_j = 1$ and $g_{ij} = 0$ for $w_j = 0$. Alternatively, $\mathbf{g}_i$ can be represented as $\mathbf{g}_i = \mathbf{W}\mathbf{x}_i$, where $\mathbf{W}$ is a diagonal matrix and its diagonal is the vector $\mathbf{w}$.

Intuitively, we would like the examples with the same class to be close (intra-class tightness) and the examples from different classes to be far away (inter-class separability) in the spaces defined by selected features. The Euclidean distance between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in the new feature space can be calculated as

$$d_{ij} = \left\| \mathbf{g}_i - \mathbf{g}_j \right\|^2 = \left\| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \right\|^2 = \left\| W\mathbf{x}_i - W\mathbf{x}_j \right\|^2 \tag{3.2}$$

The inter-class separability of the data can be measured by a sum of the pairwise distances between examples with different class labels

$$\sum_{y_i \neq y_j} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2. \tag{3.3}$$

The intra-class tightness of the data can be measured by a sum of the pair-wise distances between examples with the same class label

$$\sum_{y_i=y_j} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2 . \tag{3.4}$$

Therefore, the problem of selecting a feature subset to maximize the intra-class tightness and inter-class separability can be formulated as

$$\begin{aligned} \min_{\mathbf{w}} & \sum_{y_i=y_j} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2 \\ & - \sum_{y_i \neq y_j} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2 . \end{aligned} \tag{3.5}$$

Objective (3. 5) can be rewritten as

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \sum_{j=1}^{N} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2 \; A_{ij} , \tag{3.6}$$

where matrix A is defined as:

$$A_{ij} = \begin{cases} 1 \;\; if \; y_i = y_j \\ -1 \; if \; y_i \neq y_j \end{cases} \tag{3.7}$$

In addition to the objective (3.5) or (3.6), in order to improve the diversity of selected features, we would like to select a feature subset with minimal redundancy. A feature is defined to be redundant if there is another feature highly correlated with it. Let us denote Q as a symmetric positive semidefinite matrix with size $M \times M$, whose element Q_{ij} represents the similarity between feature i and feature j. For example, the similarity between two features can be measured by Pearson correlation or by mutual information. Then, we define a redundancy among the selected set of features represented by vector $\mathbf{w}$ as their average pair-wise similarity $\mathbf{w}^T Q \mathbf{w} / m^2$, where m is the number of selected features. Our objective is to minimize the redundancy defined in such way.

The first contribution of this chapter is to formulate the feature selection task as a new quadratic programming problem subject to binary integer and linear constraints as follows,

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{i=1}^{N}\sum_{j=1}^{N} \| \mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w} \|^2 \; A_{ij} + \lambda \frac{1}{m^2} \mathbf{w}^T Q \mathbf{w} \\ &s.t. \quad w_i \in \{0,1\} \;\; \forall i \\ &\quad \sum_{i=1}^{M} w_i = m. \end{aligned} \tag{3.8}$$

The first term in (3.8), which is a linear term as shown in the following **Proposition 1**, tries to maximize the inter-class separability and intra-class tightness of the data. It describes the discriminative power of the selected feature subset. The second quadratic term is the average pair-wise similarity score between the selected features, which results in reduction of feature redundancy. Parameter λ is introduced to control the tradeoff between feature relevance and feature redundancy. Since Q is a positive semi-definite matrix, the proposed objective function is convex. The first constraint ensures that the resulting vector **w** is binary, while the second constraint ensures that exactly m features are selected. The following proposition establishes that the first term in the objective (3.8) is linear.

Proposition 1. The first term of the objective function (3. 8) can be written as a linear term $\mathbf{c}^T\mathbf{w}$, where **c** is vector of size M with elements $c_i = (X^TLX)_{ii}$, L is the Laplacian matrix of A, defined as $L = D - A$. D is a diagonal degree matrix such that $D_{ii} = \sum_j A_{ij}$. The X is the $N \times M$ feature matrix. Each row in X corresponds to one example. $(X^TLX)_{ii}$ denotes the i-th element in the diagonal of the matrix X^TLX.

Proof. Let us denote W as a diagonal matrix where $W_{ii} = w_i$. Then,

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\|\mathbf{x}_i \circ \mathbf{w} - \mathbf{x}_j \circ \mathbf{w}\|^2 A_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N}\|W\mathbf{x}_i - W\mathbf{x}_j\|^2 A_{ij}$$
$$= \text{trace}(W^T X^T LXW) = \text{trace}(X^T LXWW^T)$$

Because $w_i \in \{0,1\}$, $WW^T = W$. Therefore, trace($X^T LXWW^T$)

$= \sum_{i=1}^{M}(X^T LX)_{ii} W_{ii} = \mathbf{c}^T \mathbf{w}$, where $c_i = (X^T LX)_{ii}$. □

Based on **Proposition 1,** objective (3.8) can be rewritten as the following constrained quadratic optimization problem,

$$\begin{aligned} &\min_{\mathbf{w}} \mathbf{c}^T \mathbf{w} + \lambda \frac{1}{m^2} \mathbf{w}^T Q \mathbf{w} \\ &s.t. \quad w_i \in \{0,1\} \ \forall i \\ &\quad \sum_{i=1}^{M} w_i = m. \end{aligned} \tag{3.9}$$

There are two practical obstacles in solving (3.9): (1) Binary constraint of variable **w**, and (2) feature similarity matrix Q is with size $M \times M$, which implies high computational cost for high dimensional data. In the next two sections, we will first relax the binary constraint, and then we will apply a low-rank approximation to Q. The resulting constrained optimization problem can be solved very efficiently, with linear time with respect to the number of features M.

3.3 Problem Relaxation and Low-Rank Approximation

Problem Relaxation Due to the binary constraint on the indictor vector **w**, it is difficult to solve (3.9) (Liu et al., 2011). To resolve this, we first relax the binary constraint on w

by allowing its elements w_i to be within the range [0, m]. Then, (3.9) could be approximated by

$$\begin{aligned} &\min_{\mathbf{w}} \mathbf{c}^T\mathbf{w} + \lambda \frac{1}{m^2}\mathbf{w}^T Q\mathbf{w} \\ &s.t. \quad w_i \geq 0 \ \forall i \\ &\quad \sum_{i=1}^{M} w_i = m. \end{aligned} \tag{3.10}$$

Now, (3.10) becomes a standard Quadratic Programming (QP) problem. The optimal solution can be obtained by a general QP solver (e.g., MOSEK (Andersen and Anderson, 2000)).

Low-rank Approximation The matrix Q in (3.10) is of size $M \times M$. So, it results in high time and space cost if we work with high dimensional microarray data. Therefore, we would like to avoid the computational bottleneck by using low-rank approximation techniques.

The matrix Q in (3.10) is symmetric positive semi-definite. So, it can be decomposed as $Q = U\Lambda U^T$, where U is a matrix of eigenvectors and Λ is a diagonal matrix with corresponding eigenvalues of Q. By setting $\boldsymbol{\alpha} = \Lambda^{1/2}U^T\mathbf{w}$, it follows that $\mathbf{w} = U\Lambda^{-1/2}\boldsymbol{\alpha}$. Therefore, problem (3.10) can be rewritten as

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathbf{c}^T U\Lambda^{-1/2}\boldsymbol{\alpha} + \lambda \frac{1}{m^2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\ &s.t. \quad U\Lambda^{-1/2}\boldsymbol{\alpha} \geq \mathbf{0} \\ &\quad \mathbf{1}U\Lambda^{-1/2}\boldsymbol{\alpha} = m. \end{aligned} \tag{3.11}$$

Typically, the rank of $\boldsymbol{Q}$ (let us denote it as k) is much smaller than M, $k << M$. Therefore, we can replace the full eigenvector and eigenvalue matrices U and Λ by the

top k eigenvectors and eigenvalues, resulting in an $M \times k$ matrix U_k and a $k \times k$ diagonal matrix Λ_k, without losing much information. Therefore, (3.11) is reformulated as

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathbf{c}^T U_k \Lambda_k^{-1/2} \boldsymbol{\alpha} + \lambda \frac{1}{m^2} \boldsymbol{\alpha}^T \boldsymbol{\alpha} \\ &s.t.\ \ U_k \Lambda_k^{-1/2} \boldsymbol{\alpha} \geq 0 \\ &\qquad \mathbf{1} U_k \Lambda_k^{-1/2} \boldsymbol{\alpha} = m. \end{aligned} \tag{3.12}$$

Since $\boldsymbol{\alpha}$ is a vector with length k, $k<<M$. the QP (3.11) is reduced to a new QP in a k-dimensional space with $M + 1$ constraints. Once the solution $\boldsymbol{\alpha}$ of (3.12) is obtained, the variable $\mathbf{w}$ in original space can be approximated by $\mathbf{w} = U_k \Lambda_k^{-1/2} \boldsymbol{\alpha}$.

Decomposing matrix $\boldsymbol{Q}$ requires $O(M^3)$ time, which is expensive in microarray data where M is large. Next we will show how to efficiently compute the top k eigenvectors and eigenvalues using Nystrom approximation technique (Williams and Seeger, 2000). Nystrom method approximates a $M \times M$ symmetric, positive semi-definite matrix Q by

$$Q \approx E_{Mk} W_{kk}^{-1} E_{Mk}^T, \tag{3.13}$$

where E_{Mk} denotes the sub-matrix of Q created by selecting k of its columns, and W_{kk} is a sub-matrix that corresponds to the intersection of the selected columns and rows. Sampling schemes in Nystrom method include random sampling (Williams and Seeger, 2000), probabilistic sampling (Drineas and Mahoney, 2005), and k-means based sampling (Zhang et al., 2009). We chose the k-means sampling in our experiments because (Zhang et al., 2009) showed that it produces very good low-rank approximations at a relatively low cost. Given (3.13), we can easily obtain the low rank approximation of Q as

$$Q = GG^T \text{ where } G = E_{Mk} W_{kk}^{-1/2}. \quad (3.14)$$

As shown in the following **Proposition 2**, the top k eigenvectors and eigenvalues can be computed in $O(Mk^2)$ time using Nystrom method, which is much more efficient than doing eigen-decomposition of Q, which requires $O(M^3)$ time.

Proposition 2. The top k eigenvectors U_k and the corresponding eigenvectors Λ_k of $Q = GG^T$ can be approximated as $\Lambda_k = \Lambda_G$ and $U_k = GU_G\Lambda_G^{-1/2}$, where U_G and Λ_G are obtained by the eigen-decomposition of $k \times k$ matrix $G^TG = U_G\Lambda_G U_G^T$.

Proof. First, we observe that U_k contains orthonormal columns. $U_k^T U_k = \Lambda_G^{-1/2} U_G^T G^T G U_G \Lambda_G^{-1/2} = \Lambda_G^{-1/2} U_G^T U_G \Lambda_G U_G^T U_G \Lambda_G^{-1/2} = \mathbf{I}$. Next, we observe that $U_k \Lambda_k U_k^T = GU_G\Lambda_G^{-1/2}\Lambda_G\Lambda_G^{-1/2}U_G^T G^T = GG^T = Q$ □

Our proposed feature selection algorithm is summarized in **Algorithm 1**. In the **Algorithm 1**, steps 1 to 5 require O($Mk^2 + k^3$) time. QP in step 6 with k variables has a polynomial time complexity with respect to k. Step 7 requires $O(Mk)$ time. Therefore, overall, the proposed feature selection algorithm is very efficient and it has linear time complexity with the number of features M.

Algorithm 3.1: Single-Task Binary Integer Program Feature Selection

Input: training data $\boldsymbol{X}$, their labels $\mathbf{y}$, regularized parameter λ, number of features m, low-rank parameter k.

Output: m selected features

1. Apply **Proposition 1** to compute the vector $\mathbf{c}$
2. Use k-means to select k landmark features for low-rank approximation of Q
3. Compute E_{Mk} and W_{kk} in (3.11)
4. Obtain low-rank approximation of Q by (3.12)
5. Apply **Proposition 2** to compute the top k eigenvalue Λ_k and eigenvector U_k of Q
6. Obtain $\boldsymbol{\alpha}$ by solving the lower dimensional QP problem (3.10).
7. Obtain $\mathbf{s}$ in original feature space as $\mathbf{w} = U_k \Lambda_k^{-1/2} \boldsymbol{\alpha}$
8. Rank the features according to the weight vector $\mathbf{s}$ and select the top m features

3.4 Multi-Task Feature Selection by Binary Integer Programming

Multi-task learning algorithms have been shown to be able to achieve significantly higher accuracy than single-task learning algorithms both empirically (Obozinski et al., 2010) and theoretically (Ben-David and Schuller, 2003). Motivated by these promising results, in this section, we extend our feature selection algorithm to the multi-task setting. The objective is to select features which are discriminative and non-redundant over multiple microarray data sets.

Let us suppose there are T different but similar classification problems, and denote the training data of the t-th task as $D^t = \{(\mathbf{x}_i^t, y_i^t), i = 1,..., N^t\}$, where N^t is the number of training examples of the t-th task. Argyriou et al. (2008); Obozinski et al. (2010) proposed multi-task feature selection algorithms that use $l_{1,2}$ norm to regularize the linear model coefficients β across K different classification tasks. The $l_{1,2}$ norm regularizer over all βs across K classification tasks could be expressed as $\sum_{j=1}^{M}\sum_{t=1}^{K}\| \beta_t^j \|_2$, where β_t^j is the coefficient of the j-th feature in the t-th task. Due to the l_1 norm on the l_2 norm of group of coefficients of each feature across K tasks, the $l_{1,2}$ norm regularizer selects the same feature subset across K tasks. However, the $l_{1,2}$ norm regularized problem is challenging to solve because the non-smoothness of the $l_{1,2}$ norm. In this section, we would like to show our proposed feature selection can be easily extended to multi-task learning version. The resulting objective optimization problem have the same form as objective (9), which can be solve efficiently as shown in previous section.

Let us denote wt as the binary indicator defined in (3.1) to represent the selected feature subset of the t-th classification task. If we do not consider the relatedness between these K classification task, individual wt could be obtained by applying **Algorithm 1** to different classification tasks. Based on the conclusion given by Argyriou et al. (2008); Obozinski et al. (2010), it would be beneficial to select the same feature subset across K related classification task. In our case, this is can be achieved by setting $\mathbf{w}_t = \mathbf{w}\ \forall\ t$. Therefore, the same feature across K tasks, defined by vector w, can be obtained by solving the following optimization problem,

$$\begin{aligned} &\min_{w} \sum_{j=1}^{T} \mathbf{c}_j^T \mathbf{w} + \lambda \mathbf{w}^T (\sum_{j=1}^{T} Q_j) \mathbf{w} \\ &s.t. \;\; w_i \in \{0,1\} \;\forall i \\ &\quad \sum_{i=1}^{M} w_i = m, \end{aligned} \tag{3.15}$$

where $\mathbf{c}_j$ and Q_j are the linear and quadratic terms of the *QP* (7) corresponding to the *j*-th task. The details about how to compute the $\mathbf{c}_j$ and Q_j are explained in the previous section. The technique of relaxing binary integer constraints and applying low-rank approximation to *Q* introduced in the previous section can be used to solve (3.15) efficiently. The extended multi-task feature selection algorithm is also a feature filter. It can be used in conjunction with any supervised learning algorithm.

3.5 Empirical Results

We compared our proposed feature algorithm with 9 representative feature selection filters. The first 6 are standard feature selection filters: Pearson Correlation (PC), ChiSquare (Liu and Setiono, 1996), GINI, Infogain, Kruskal-Wallis test and RelieF (Kira and Rendell). They rank the features based on different criteria that measure correlation between each feature and class label. The remaining 3 are the state-of-the-art feature selection methods which are able to remove redundant features: mRMR (Peng et al., 2005), QPFS (Lujan et al., 2010) and SASMIF (Liu et al., 2011). The feature similarity for both QPFS and our algorithm was measured by Pearson correlation. For fair comparison, for the SASMIF method we used top *m* ranked features. To balance the effect of feature relevance and feature redundancy, the parameter $\boldsymbol{\lambda}$ in (3.9) was set to $(m^2 M \sum_i c_i) / \sum_{i,j} Q_{ij}$. The low-rank parameter *k* was set to $0.1 \cdot M$, as suggested in (Zhang

et al., 2009). Our algorithm is denoted as *ST-BIP* for single task version and *MT-BIP* for multi-task version.

Given the selected features, we used LIBLINEAR (Fan et al., 2008) to train the linear SVM model. The linear SVM model was chosen because previous studies (Guyon et al., 2002) showed SVM classifier could be very accurate on microarray data. The regularization parameter C of LIBLINEAR was chosen among $\{10^{-3}, 10^{-4}, \dots, 10^{3}\}$. For the experiments in the single-task scenario, we used the nested 5 cross validation to select the optimal regularization parameter. For experiments in multi-task learning scenario, it was too time consuming to use the nested cross-validation to select the regularization parameter. Thus, we simply fixed the regularization parameter to 1 in the multi-task experiments.

Single-Task Feature Selection

In this section, we evaluate our proposed feature selection algorithm for single-task learning using four benchmark microarray gene expression cancer datasets: (1) Colon dataset (U. Alon and Levine, 1999) containing 62 samples, 40 tumor and 22 normal samples; (2) Lung dataset (D.G. Beer, 2002) containing 86 samples coming from 24 patients that died and 62 that survived; (3) Diffuse B-cell Lymphoma (DLBCL) dataset (M.A. Shipp, 2002) containing 77 samples, 58 coming from DLBCL patients and 19 from B-cell lymphoma patients. (4) Myeloma dataset (E. Tian, 2003) containing 173 samples, 137 coming from patients with bone lytic lesions and 36 from control patients. We summarize the characteristics of these datasets in Table 3.1.

Table 3.1 Summary of the Microarray Datasets

	Colon	Lung	DLBCL	Myeloma
#Samples	62(40/22)	86(24/62)	77(58/19)	173(137/36)
#Genes	2000	5469	5469	12558

For each microarray dataset, we randomly selected 20 positive and 20 negative examples (except for choosing 15 positive and 15 negative in DLBCL dataset) as the training set and the rest as the test set. Due to the class imbalance in test sets, we used AUC, the area under the Receiver Operating Characteristic (ROC) curve, to evaluate the performance. The average AUC based on 10 repetitions of experiments on different random splits to training and test set are reported in Table 3.2. We Compared the AUC accuracy of different feature selection algorithms for m = 50, 100, 200, 1000. For each dataset, the best AUC score among all methods was emphasized in bold. As shown in Table 3.2, our proposed method achieved the highest accuracy on Colon and DLBCL datasets. On the Myeloma dataset, it had the highest accuracy when m = 100 and 1000 and had the second highest accuracy when m = 50 and 200. On the Lung dataset, our algorithm was ranked in the upper half of the competing algorithms. The last column in Table 3.2 shows the average AUC score across four different datasets. Our method achieved the highest average AUC scores. The next two successful feature selection algorithms are *Relief* and *QSFS*. The *mRMR* had somewhat lower accuracy, comparable to simple filters such as *PC*, *ChiSquare*, *GINI* and *InfoGain*. *SASMIF* was considerably less accurate, while *KW* was the least successful.

Table 3.2 Average AUC of 10 different feature selection algorithms on 4 different Microarray Datasets

		Colon	**Lung**	**DLBCL**	**Myeloma**	**Average**
m = 50	*PC*	.763±.170	.648±.184	.958±.025	.709±.071	.770
	ChiSquare	.740±.189	.600±.173	.965±.035	.676±.076	.745
	GINI	.742±.183	.586±.167	.966±.034	.666±.096	.740
	InfoGain	.755±.179	.595±.170	.963±.026	.682±.085	.749
	KW	.755±.187	.574±.163	.858±.128	.606±.072	.698
	Relief	.785±.145	**.661±.194**	.966±.027	.677±.082	.772
	mRMR	.748±.182	.651±.219	.948±.067	.695±.093	.761
	SASMIF	.663±.206	.563±.130	.943±.043	.636±.004	.701
	QSFS	.695±.208	.608±.054	.961±.031	**.714±.080**	.745
	ST-BIP	**.828±.082**	.600±.124	**.969±.034**	.710±.110	**.777**
m = 100	*PC*	.753±.176	.607±.122	.963±.025	.708±.062	.758
	ChiSquare	.745±.184	.631±.164	.966±.024	.688±.063	.758
	GINI	.748±.186	.594±.202	.965±.026	.698±.079	.751
	InfoGain	.750±.180	.631±.164	.967±.022	.690±.062	.760
	KW	.727±.188	.570±.206	.879±.113	.624±.071	.700
	Relief	.773±.177	.631±.176	.958±.042	.708±.066	.768
	mRMR	.758±.169	.608±.169	.966±.035	.690±.075	.756
	SASMIF	.785±.131	.611±.213	.950±.035	.647±.072	.748
	QSFS	.777±.173	**.636 ±.113**	.965±.025	.710±.073	.772
	ST-BIP	**.833±.078**	.627±.180	**.975±.033**	**.735±.086**	**.793**
m = 200	*PC*	.760±.164	.632±.120	.973±.018	.704±.059	.767
	ChiSquare	.750±.165	.611±.198	.973±.030	.673±.072	.752
	GINI	.753±.165	.617±.199	.974±.019	.690±.064	.759
	InfoGain	.755±.165	.611±.198	.977±.017	.673±.072	.754
	KW	.735±.219	.571±.199	.878±.145	.637±.036	.705
	relief	.758±.162	.621±.157	.979±.025	**.721±.076**	.770
	mRMR	.755±.155	.585±.169	.974±.027	.668±.068	.746
	SASMIF	.820±.011	.590±.124	.954±.221	.644±.045	.752
	QSFS	.765±.171	**.664±.187**	.974±.025	.687±.052	.773
	ST-BIP	**.833±.080**	.634±.156	**.984±.020**	.706±.106	**.789**
m = 1000	*PC*	.740±.172	.633±.193	.979±.018	.700±.049	.763
	ChiSquare	.743±.174	.606±.121	.974±.028	.676±.060	.750
	GINI	.735±.176	**.645±.152**	.974±.027	.679±.056	.758
	InfoGain	.743±.174	.606±.121	.974±.028	.676±.060	.750
	KW	.722±.198	.568±.184	.941±.051	.652±.037	.721
	Relief	.728±.173	.623±.150	.980±.019	.698±.051	.757
	mRMR	.743±.174	.606±.121	.976±.025	.677±.060	.751
	SASMIF	.763±.149	.587±.176	.952±.038	.669±.054	.743
	QSFS	.745±.175	.624±.163	.980±.017	.690±.047	.760
	ST-BIP	**.828±.063**	.625±.192	**.981±.020**	**.722±.078**	**.789**

Table 3.3 Multi-Task Microarray Datasets (cancer: normal cases)

Bladder	Lung	Prostate	Breast
18(11/7)	27(20/7)	23(14/9)	22(17/5)
Renal	Colon	Pancreas	Uterus
24(11/13)	26(15/11)	21(11/10)	17(11/6)

Table 3.4 Average AUC of 11 different feature selection algorithms on 8 different Microarray Datasets

	Bladder	**Breast**	**Colon**	**Lung**	**Pancreas**	**Prostate**	**Renal**	**Uterus**	***Avg.***
PC	.991	.696	.816	.703	.780	.603	.916	.883	.799
ChiSquare	.969	.625	.749	.669	.789	.636	.741	.908	.761
GINI	.969	.625	.749	.669	.789	.636	.743	.917	.762
Infogain	.969	.625	.749	.669	.789	.636	.741	.908	.761
KW	.903	.621	.907	.750	.876	.626	.870	.913	.808
reliefF	.991	.729	.795	.721	.796	.594	.929	.888	.805
mRMR	.969	.650	.765	.682	.830	.682	.786	.875	.780
SASMIF	.978	.739	.704	.671	.823	.650	.768	.854	.773
QSFS	.991	.693	.817	.700	.788	.600	.916	.883	.799
ST-BIP	.991	.679	**.921**	**.782**	**.882**	.612	**.966**	.910	.843
MT-BIP	**.997**	**.850**	.882	.754	.846	**.715**	.895	**.921**	**.858**

Multi-Task Feature Selection

In this section, we evaluate our proposed feature selection algorithm for multi-task learning. We used 8 cancer-related binary microarray classification datasets published in (Ramaswamy, 2001). The data are summarized in Table 3.3. As shown in Table 3.3, the size of the 8 microarray datasets was very small. The single-task feature selection algorithms are not expected to perform well because there might be insufficient information even when simple feature selection filters are used. In contrast, our multi-task feature selection algorithm is expected to improve the accuracy by borrowing strength across multiple microarray datasets.

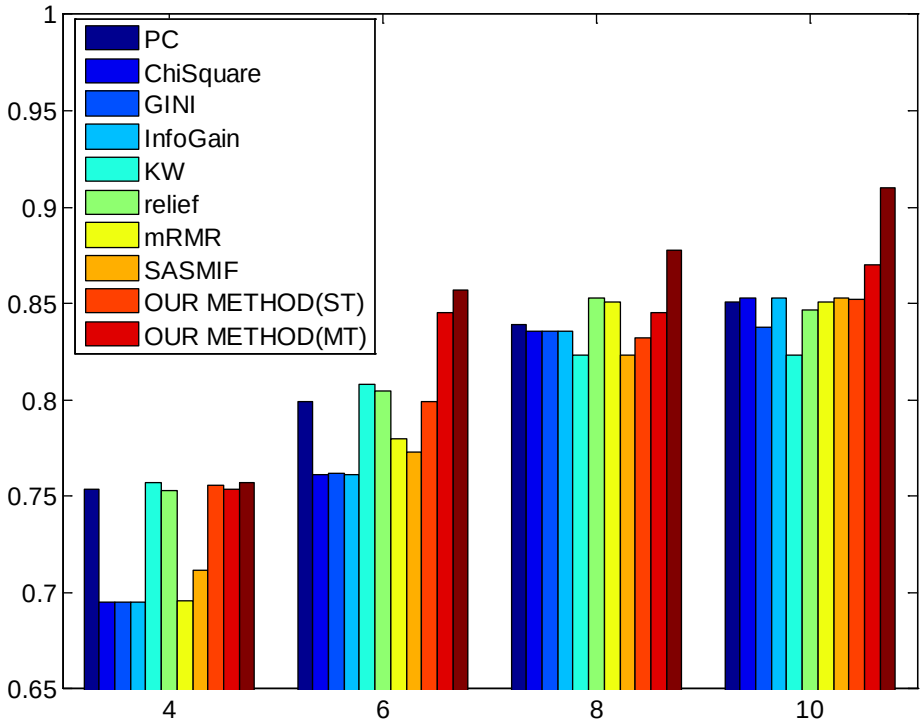

Figure 3.1. Average AUC score of different feature selection algorithms across different train sizes

For each microarray data set, we randomly selected $N^+ = \{2, 3, 4, 5\}$ positive and the same number of negative examples as the training data and used the rest as the test data. We only show the results for $m = 100$ in this section, because it got the best empirical results. The average AUC across these 8 microarray datasets is shown in Figure 1. The results clearly show the multi-task version of our proposed algorithm was the most successful algorithm overall. To gain a deeper understanding about the reason why the multi-task feature selection algorithm obtained better overall accuracy than single-task feature selection algorithms, we show the AUC score of each individual microarray dataset based on $N^+ = 3$ in Table 3.4. We can see that the single task version of our

feature selection algorithm had the highest overall accuracy among other single-task benchmarks, a result consistent with Table 3.2. The multi-task version of our algorithm has higher AUC than its single task version on 4 datasets and its average AUC is about 1.5% higher. In few cases, (e.g. Colon, Lung, Pancreas, Renal datasets) we can also observe the negative transfer, where the accuracy drops. How to prevent negative transfer in multi-task feature selection would be another interest research topic for our future research.

3.6 Conclusions

This chapter proposed a novel feature filter method to select a feature subset with discriminative power and minimal redundancy. The proposed feature selection method is based on quadratic optimization problem with binary integer constraints. It can be solved efficiently by relaxing the binary integer constrains and applying a low-rank approximation to the quadratic term in the objective. Furthermore, we extend our feature selection algorithm to multi-task classification problems. The empirical results on a number of microarray data sets show that in the single task scenario the proposed algorithm results in higher accuracy than the existing feature selection methods. The results also suggest that our multi-task feature selection algorithm can further improve the microarray classification performance.

CHAPTER 4

PROTEIN FUNCTION PREDICTION BY INTEGERATING DIFFERENT DATA SOURCES

4.1 Background

Determining biological functions of proteins is a key challenge in the post-genomic era. The experimental methods for protein function prediction are time-consuming and resource-intensive. It is infeasible to experimentally determine the functions of all known proteins. For that reason, computational methods that predict biological functions of a protein using known information about its sequence, structure, and functional behavior, are becoming an attractive low-cost alternative. During the recent couple of decades, many computational methods for predicting protein function have been developed, and the 2011 Critical Assessment of Function Annotations (CAFA) has been designed to establish a state of the art in the field.

The sequence alignment-based function inference is the most widely used form of computational function prediction (Friedberg, 2006). These approaches use sequence comparison tools, such as BLAST (Altschul et al., 1997), to search annotated databases for the most similar proteins to the query protein based on sequence and transfer their functions. The bio-logical rationale for sequence comparison is that if two sequences are similar, then they probably evolved from a common ancestor and have similar functions. GOtcha (Martin et al., 2004) is a similar method that takes sequence alignment scores between a query protein and a database of functionally annotated proteins, and overlays them on functional ontology, cumulatively propagating the scores towards the root of the

ontology. Both the BLAST and GOtcha approaches were used as baselines in 2011 CAFA.

Beyond sequence similarity, several computational approaches have been proposed to utilize other types of biological data, such as protein-protein interactions (PPI) and gene expression data. The methods that use PPI data to predict protein functions are based on a simple premise: a protein does not perform its function in isolation; instead, a group of proteins needs to interact in order to perform a certain function. Therefore, the functions of a querying protein can be inferred from its interacting partners. Schwikowski et al., (2000) used a neighbor counting method, where a function is assigned to the querying protein based on the number of its neighbors in the PPI graph which have this function. Hishigaki et al. (2001) extended this method, by considering proteins that could be reached via *n* links, instead of considering only the direct neighbors.

Use of gene expression data for function prediction has been motivated by an observation that co-expressed genes are likely to be functionally related (Zhou et al., 2002; Eisen et al., 1998; Pavlidis et al., 2001). In the seminal work by Eisen et al., (1998) based on the co-expression data, genes were clustered into a number of groups and the functions transferred to all genes in a cluster. Machine learning-based approaches where function prediction is studied as a multi-label classification problem have also been popular. There, a function is predicted from gene expression measurements across several microarrays. For example, in an early work of this type, Brown et al. (2000) applied Support Vector Machines classifier to the task of learning functions from yeast gene expression data. Arguably, each data source captures only one aspect about proteins'

properties. Thus, combining such heterogeneous data can bring a more complete picture about protein function. Recently, several studies showed promising improvements in protein function prediction by integrating multiple types of biological data. Troyanskaya et al. (2003) proposed a Bayesian network model to infer the posterior probability functional linkage between two genes given their functional relationship observed from multiple data sources. Barutcuoglu et al. (2006) integrated different data sources by concatenating all feature vectors from different data sources for a protein into a single feature vector. Mostafavi and Morris (2010) assigned weights to different data sources by solving a constrained linear regression problem, which minimized the least square error between the composite network and the target network constructed from the label vector, on sets of related functional categories. Despite these and related efforts, how to effectively integrate different types of biological data for protein function pre-diction remains a largely open question.

There are several challenges that need to be addressed in future research on multi-source function prediction. The first is that different sources of information may have vastly different coverage. For example, while sequence similarity covers all known proteins, PPI data coverage is significantly smaller, and gene expression similarities are constrained by a specific microarray platform. The second challenge is differences in data quality. For example, PPI can be obtained by a variety of techniques that differ in cost and reliability. A confounding issue is that functional annotations have an uneven coverage biased towards certain types of proteins and functions, and that determination of

protein functions, such as the one provided by Gene Ontology (Ashburner et al., 2000), is a subjective and error-prone process.

In an attempt to address some of the identified challenges and faced with the tight deadline of 2011 CAFA, we focused our attention on the *k*-nearest neighbor approach for function prediction proposed in (Pandey et al., 2009). This is an easy to implement, intuitive, and relatively fast algorithm that searches for *k* nearest neighbors of the query sequence and transfers their functions by weighted averaging, such that nearer neighbors have larger influence to prediction than the farther ones. In this paper, we propose the Multi-Source *k*NN (MS-*k*NN) algorithm able to use multiple sources of protein information. To provide the final prediction, MS-*k*NN uses weighted averaging of the source-specific prediction scores. In the algorithm design, we explored several approaches to determine weights, ranging from averaging to solving a constrained optimization problem. We observe that a query protein does not have to be present in all data sources. For example, we might know the protein's sequence and whether it interacts with other proteins, but not its gene expression (e.g., because its gene is not printed on a microarray, or because microarray data are not available for the host organism). Averaging of the source-specific scores provides a simple mechanism for dealing with potential missing predictions.

In the following, we will discuss evaluation of several prediction approaches prior to CAFA submission deadline, describe how we selected the predictor, summarize and discuss results on CAFA proteins, and propose some directions for the future research.

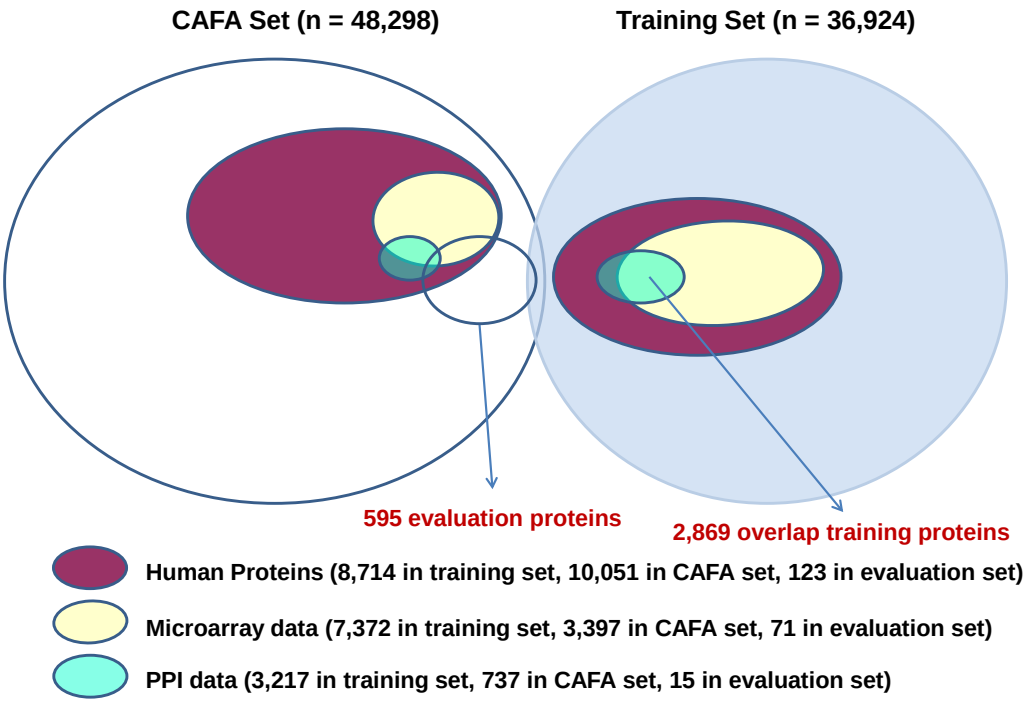

Figure 4. 1. Visual summary of the datasets

4.2 CAFA Challenge and Our Data Sources

At the beginning of the challenge, 48,298 proteins were released by CAFA organizers as the test proteins. In the released test data, the protein names, Uniprot IDs, and sequences were provided. A large majority of these proteins did not have known functional assignments, as defined by the Gene Ontology (GO) annotations in Swiss-Prot database. The organizers did not provide any training data to the participants. Therefore, the participants were free to use any available information about proteins and genes they found suitable. The objective of the assessment was to predict functions of the test proteins. The success was measured by evaluating the prediction accuracy of GO

annotations of the test proteins that became available after the submission deadline.

Data Sources

We considered integration of three different data sources for protein function prediction. These three data sources were: (1) protein sequence data; (2) microarray expression data; and (3) protein-protein interaction data. Particularly, prior to CAFA deadline we focused on human proteins, in order to more easily evaluate and characterize our approaches. Visual summary of the data sets we used is in Figure 4.1.

Protein Sequence Data. By courtesy of Dr. Predrag Radivojac from Indiana University[1], we obtained a data set of GO annotations of 36,924 proteins, as well as their pair-wise sequence similarities (expressed as percent identity), and pair-wise similarities between these proteins and the 48,298 CAFA proteins. These 36,924 proteins with their GO annotations and sequences were used as the training set for function prediction. We note that there were 474 proteins that were present in both training and CAFA data set. These 474 proteins were excluded from the training set during our evaluation.

Microarray Expression Data. We downloaded 392 Affymetrix GPL96 Platform microarray datasets from GEO[2]. The GPL96 is one of the most widely used human microarray platforms. We linked the Affymetrix probe IDs with Uniprot IDs through Entrez. After ID mapping, the microarray data covered 7,372 human proteins in the training set, and 3,397 human proteins in CAFA set. These datasets were already pre-processed by Affymetrix Microarray Suite Version 5.0 and we did not apply any additional pre-processing.

[1] http://montana.informatics.indiana.edu/wtclark/GO_For_Others.zip
[2] http://www.ncbi.nlm.nih.gov/geo/, downloaded in summer of 2009.

Table 4.1. Summary of different data sources

Data source	Training size	CAFA size
Protein sequence similarity	36,924	48,298
Microarray expression	7,372	3,397
Protein-protein interaction	3,217	737

Protein-Protein Interaction Data. For PPI data source, we used physical interactions between human proteins listed in OPHID database[3]. This data source includes 41,457 interactions between 9,141 proteins. After ID mapping, the PPI data source covered 3,217 proteins in the training set, and 737 proteins in CAFA set.

We summarize the information about each data source in Table 4.1.

4.3 Methodology

Below, we describe *k*-nearest neighbor (*k*NN) classifiers we evaluated and used during the competition, as well as the approaches for integration of predictions from multiple data sources.

Baseline kNN Classifier

To calculate a likelihood that protein *p* has function *f*, we used a weighted variant of the kNN algorithm, as proposed in (Pandey et al., 2009). The prediction score of function *f* for protein *p* is calculated as

$$score(p, f) = \sum_{p' \in N_k(p)} sim(p, p') I(f \in functions(p')), \quad (4.1)$$

where $sim(p, p')$ denotes the similarity score between proteins *p* and *p'*, *I* is an indicator function that returns 1 if p' is experimentally annotated with *f* and 0 otherwise, and $N_k(p)$ is the set of the *k* nearest neighbors of *p* according to the metric *sim*. The similarity score

[3]http://www.phenopred.org/

between two proteins $sim(p, p')$ on each of the three data sources we considered was calculated in the following way. For protein sequence data, the similarity score was calculated as percent identity divided by 100. For microarray data, Pearson correlation coefficient, a popular method for measuring the similarity between gene expressions (D'haeseleer, 2005), was used as the similarity score between two proteins. For protein-protein interaction data, the similarity score was set to 1 if two proteins interacted and 0 otherwise. We note that more sophisticated similarity score could be used by considering reliability of PPI information, but we did not pursue it due to the CAFA time limitations.

Lin-Sim *k*-NN Classifier

In protein function prediction problem, a single protein may have multiple functions, and the functions are organized in a hierarchy. Pandey et al., (2009) proposed a method that incorporates contributions not only from the neighboring proteins annotated with the same function, but also from proteins annotated with *similar* functions. Their proposed prediction model is

$$score(p, f) = \sum_{p' \in N_k(p)} sim(p, p')\left(\sum_{f' \in functions(p')} linsim(f', f) \right), \tag{4.2}$$

where $linsim(f', f)$ denotes the similarity score between functions f and f'. The Lin's similarity measure (Lin, 1998) is used to compute the similarity between two concepts in a hierarchy. It is calculated as

$$linsim(f, f') = \frac{2 \times [\log p_{ms}(f, f')]}{\log p(f) + \log p(f')}, \tag{4.3}$$

where f and f' are the Gene Ontology terms. The $p(f)$ denotes the probability of a protein being annotated with function f, which is estimated from the available set of GO

annotations. The joint probability $p_{ms}(f, f')$ is calculated as $p_{ms}(f, f') = \min_{l \in S(f, f')} p(l)$, where $S(f, f')$ is a set of common ancestors of functions f and f'. It is easy to see that $linsim(f, f') = 1$ when $f = f'$, and $limsim(f, f') = 0$ when their minimum subsume is the root of the ontology. Thus, the *limsim* score is always in the [0, 1] range.

Integration of Scores from Multiple Data Sources

By using equations (4.1) or (4.2), we can obtain scores for each protein/function pair (p, f). In particular, we can obtain one score using sequence similarity and one using PPI. Since we used J (=392) microarray data sets, we had one prediction score for each gene expression data set. Given the $J + 2$ scores for each (p, f) pair, an open question was what is the best way to integrate them into a final, single score. We studied the following prediction score integration schemes: (1) averaging; (2) the same weighted averaging for any (p, f) pair; (3) different weighted averaging for different GO term clusters. In the schemes (2) and (3), the weights for different data sources were obtained by solving a convex optimization problem.

(1) Averaging (MS-*k*NN)

Let us denote by $score^{SEQ}(p,f)$ the score obtained from sequence similarity data, by $score^{PPI}(p,f)$ the score obtained from PPI data, and by $score_j^{EXP}(p,f)$ the score obtained by the j-th microarray data set. By averaging, the final score is obtained as

$$score(p, f) = \frac{1}{3} score^{SEQ}(p, f) + \frac{1}{3} score^{PPI}(p, f) + \frac{1}{3J} \sum_{j=1}^{J} score_j^{EXP}(p, f). \qquad (4.4)$$

We call the resulting algorithm the *Multi-Source k-NN* (MS-*k*NN), as this was the final algorithm we used in CAFA.

(2) Weighted averaging (MS-W-*k*NN)

MS-*k*NN assumes that each data source is equally informative, which might not hold in general. We thus considered using weighted averaging of the scores from different sources. For MS-W-*k*NN we learned the weights from training data using a large margin method as follows. Let us assume that we are given m data sources, $\{D_j, j = 1..m\}$, and n proteins $\{x_i, i = 1..n\}$. Each protein is assigned to several functions from the set of k functions. Let Y_i denote the set of functions that protein x_i is assigned to, and $\overline{Y_i}$ the set of functions that protein x_i is not assigned to. Furthermore, let $f(x,y)$ be a vector of length m, whose j-th element is the score of protein x for function y on the data source D_j. Then, a weight vector $\mathbf{w}$, used for averaging of m prediction, is found by minimizing the following optimization problem,

$$\begin{aligned} &\min_{\mathbf{w},\xi} \sum_i \sum_{y\in Y_i,\, \bar{y}\in\overline{Y_i}} \xi_i(y,\bar{y}) \\ &s.t\ \mathbf{w}^T(f(x_i,y) - f(x_i,\bar{y})) \geq -\xi_i(y,\bar{y}), \forall i, y\in Y_i, \bar{y}\in\overline{Y_i} \\ &\quad \xi_i(y,\bar{y}) \geq 0, \forall i, y\in Y_i, \bar{y}\in\overline{Y_i} \\ &\quad \mathbf{w}^T\mathbf{e} = 1; \mathbf{w} \geq 0 \end{aligned} \tag{4.5}$$

where $\mathbf{e}$ is a vector of ones. The resulting convex optimization problem can be solved using standard optimization tools, such as CVX[4]. With the trained weight vector $\mathbf{w}$, the protein-function scores from different data sources can be integrated by taking their weighted average as $\mathbf{w}^T \cdot f(x,y)$.

(3) Cluster-specific weighted averaging (MS-CW-kNN)

Instead of learning a single weight for all GO terms, we can partition functions into clusters and assign cluster specific weights. Since the Gene Ontology is already

[4]http://cvxr.com/cvx/

organized in a hierarchical structure, we can directly use it to cluster the GO terms. In MS-CW-*k*NN, we considered clustering of all GO terms at the root level to MF and BP terms, and at the first level to 7 MF and 25 BP functional clusters.

4.4 Empirical Results before CAFA

There were 2,869 annotated human proteins in the training set that were represented by all 3 data sources. Among them, we randomly selected 1,302 proteins as test set in our pre-CAFA analysis. Given the prediction scores on the 1,302 test proteins for function *f*, the True Positive Rate (TPR) and False Positive Rate (FPR) was calculated at different discrimination thresholds, creating the Receiver Operating Characteristic (ROC) curve. The Area Under the ROC Curve (AUC) was calculated by integrating the ROC curve, which is corresponded to the TermAUC evaluation metric of CAFA. We only considered the GO functions having more than 15 annotated proteins among the 1,567 3-source human proteins left after removing the 1,302 test proteins. This resulted in 122 molecular function (MF) and 546 biological process (BP) GO terms. We used $k = 20$ in all experiments in this section. In the following, we will discuss performance of several proposed function prediction algorithms.

4.4.1 Baseline vs. *Lin-Sim k*-NN Classifier

In this section, we compare the accuracies of the two different prediction algorithms: baseline *k*-NN classifier and *lin-sim* incorporated *k*NN classifiers. The results for *k*-NN using sequence similarity were based on the training set of 1,567 3-source human proteins, remaining after exclusion of 1,302 test proteins. While we had 392

Table 4.2. Comparison of Average TermAUC of Two Different Prediction Algorithms

Data sources	**122 MF terms**		**546 BP terms**	
	k-NN	*lin-sim k*-NN	*k*-NN	*lin-sim k*-NN
Sequence Similarity	0.671	**0.688**	0.557	**0.558**
Microarray	0.555	**0.561**	0.563	**0.578**
PPI	0.574	**0.592**	0.580	**0.611**

different microarray datasets available for this experiment, due to the tight deadline we used only the largest microarray data set (having 221 microarrays) with GEO accession number GSE4475.

The results in Table 4.2 show that the *lin-sim k*-NN classifier had slightly higher accuracy than the baseline *k*-NN. However, to estimate the *lin-sim* between all GO terms and to use them during prediction time is very time-consuming. This includes a need to determine the *lin-sim* function similarity threshold through cross-validation. As a result, we reasoned that the accuracy improvement was not large enough to justify use of *lin-sim k*-NN predictor in CAFA.

4.4.2 Paralogous vs. Orthologous Sequences

In this section, we explore how useful it is to transfer functions from paralogous and orthologous proteins. Paralogous proteins are similar proteins within the same organism that are probably created by duplication and functional divergence. Orthologous proteins are similar proteins across different organisms that are related by speciation. The test set was still the 1,302 human proteins. We used 10 different training sets, after excluding the 1,302 test proteins: (1) 1,567 human training proteins represented by all 3 sources (as in Table 4.2); (2) 7,412 human proteins in training data; (3) all 16,442

Table 4.3a. Average TermAUC based on 5 training sets of different size.

Training Set (Training Size)	122 MF terms	546 BP terms
	TermAUC	TermAUC
(1) HUMAN (1,567)	0.671	0.557
(2) HUMAN (7,412)	0.728	0.609
(3) HUMAN + MOUSE + RAT (16,442)	0.807	0.692
(4) All Mammals (16,754)	0.812	0.696
(5) All Organisms (35,622)	**0.819**	**0.707**

Table 4.3b. Average TermAUC based on 7 training sets with same size

Training Set (Training Size)	122 MF terms	546 BP terms
	TermAUC	TermAUC
(2) HUMAN (7,412)	0.728	0.609
(6) HUMAN + MOUSE + RAT (7,412)	0.771	0.648
(7) All Mammals (7,412)	0.762	0.649
(8) All Organisms (7,412)	0.729	0.628
(9) All Mammals excluding Human(7,412)	**0.779**	**0.659**
(10) All Organisms excluding Human (7,412)	0.721	0.623

proteins from human, mouse, and rat in training data; (4) 16,754 proteins from all mammals in training data; (5) all 35,622 training proteins; (6) randomly selected 7,412 proteins from set (3); (7) randomly selected 7,412 proteins from set (4); (8) randomly selected 7,412 proteins from set (5); (9) randomly selected 7,412 non-human mammal proteins; (10) randomly selected 7,412 non-human proteins. The baseline *k*-NN classifier was used as the prediction model and we used the same GO terms as in Table 4.2.

The average TermAUC accuracies for MF and BP terms are shown in Table 4.3. The results for training set from (6) to (10) are averages of 5 random selections. The results in Table 4.3a show that TermAUC grew with the number of annotated sequences. Interestingly, it was the largest when all available proteins were used, which included evolutionary distant prokaryotes. These results could be explained by the fact that as the

training set of sequences grows, it becomes more likely that truly similar sequences are found among the *k* nearest neighbors of the query sequence. In Table 4.3b, we show Term AUC for training sets of the same size. We observe that the highest accuracy was obtained by using non-human mammal proteins. The lowest accuracies were obtained either by exclusively human training proteins (set 2) or a sample including all proteins (sets 8 and 10). This indicates that the most useful proteins for function prediction are orthologs from closely related organisms. However, by comparing Table 4.3a and 4.3b, it is evident that it is preferable to simply use all available training proteins.

4.4.3 Integrating Predictions from Multiple Data Sources

In this section, we compare the results of single-source *k*NN and the proposed multi-source *k*NN algorithms described in the Methods section. We used the same 1,302 human test proteins for testing as in the previous two subsections. For sequence similarity data, we used training protein sequences from all organisms, because that resulted in the highest accuracy according to Table 4.3. For the microarray expression data, we used all 392 Affymetrix GPL96 microarray data sets. The prediction score was calculated as the average score among the 392 microarray data sets.

The average TermAUC for single and multi-source *k*NN are shown in the Table 4.4. At the level of the single-source predictors, it could be seen that the prediction based on sequence similarity was much more accurate than the microarray and PPI-based predictors on both MF and BP functions. By comparing the microarray-based predictor based on 392 microarrays listed in Table 4.4 with the microarray predictor based on a

Table 4.4. Comparison of AUCs of different methods

Data Source	122 MF terms	546 BP terms
	TermAUC	TermAUC
*k*NN: Sequence similarity	0.819	0.707
*k*NN: PPI	0.574	0.580
*k*NN: Microarray	0.635	0.642
MS-*k*NN	**0.848**	**0.763**
MS-W-*k*NN	0.829	0.758
MS-CW-*k*NN: root level	0.831	0.715
MS-CW-*k*NN: first level	0.851	0.702

single microarray data set listed in Table 4.2, it can be concluded that it was beneficial to combine predictions from multiple microarray data sets.

We observe that by averaging scores from all 3 data sources using MS-*k*NN TermAUC increased by 0.03 on MF and 0.06 on BP functions as compared to using sequence similarity only. This result is very interesting considering superiority *k*NN accuracy using sequence similarity as compared to the ones using PPI and microarray data. Such result clearly indicates that integration of multiple data sources can be beneficial for protein function prediction. Accuracies of the weighted versions of MS-*k*NN were not as high as its basic version. This was a somewhat unexpected result. Upon a more careful study of the optimization problem stated in (4.5), we concluded that the issue lies in the interpretation of zero labels. Formulation (4.5) assumes that $Y_{ij} = 0$ means that *i*-th protein does not have *j*-th function. However, $Y_{ij} = 0$ often means that the function is not known and not accounting for this results in reduced accuracy. We think that this is a valuable insight that might be helpful in design of future predictors of protein function.

It might be somewhat surprising that MS-*k*NN is able to improve prediction scores using sequence similarity with seemingly inferior prediction scores coming from PPI and microarray data. In order to understand why two seemingly inferior predictors can help the superior one, in Table 4.5 we show prediction scores and ranks of 7 test proteins annotated with function GO:0044106 (cellular amine metabolic process) obtained by 3 single-source predictors and by their averaging. We note that the predicted scores from each individual data source ranged from [0, 20] because we set the parameter k in k-NN to 20. We can see that TermAUC obtained with sequence data (0.829) was much larger than with microarray data (0.613) and PPI data (0.564). However, integrating these 3 sources improved the TermAUC to 0.938. For the first 5 proteins listed in Table 4.5 (NOS1_HUMAN, NOS3_HUMAN, OAZ1_HUMAN, OAZ2_HUMAN and SYK_HUMAN), we can see that they were ranked very high by the sequence similarity-based predictor. The addition of prediction scores from other two sources resulted in a slight decrease in their rank. For the last two proteins in Table 4.5, the sequence similarity-based predictor gave score 0, indicating that none of their $k = 20$ nearest neighbors were annotated with GO:0044106. Microarray-based score of PEPD_HUMAN was relatively small, but it was sufficient to improve its ranking near the top one third. In case of PON1_HUMAN, it had the top ranking based on PPI data and a very high ranking based on microarray data, such that it was ranked 18th in aggregate.

Table 4.5. Prediction score and rank for test proteins annotated by GO:0044106

Proteins	Microarray (AUC: 0.6127)	PPI (AUC:0.5641)	Sequence (AUC: 0.8285)	Average (AUC: 0.9379)
SYK_HUMAN	0.14(1203)	0 (NaN)	2.17 (2)	0.77 (3)
NOS3_HUMAN	0.23 (212)	0 (NaN)	1.95 (3)	0.73 (6)
NOS1_HUMAN	0.29 (19)	0 (NaN)	1.92 (4)	0.74 (5)
OAZ2_HUMAN	0.17 (882)	0 (NaN)	1.80 (6)	0.66 (8)
OAZ1_HUMAN	0.18 (820)	0 (NaN)	1.63 (7)	0.60 (9)
PEPD_HUMAN	0.22 (340)	0 (NaN)	0 (NaN)	0.07 (544)
PON1_HUMAN	0.26 (66)	1 (3)	0 (NaN)	0.42 (18)

4.5 CAFA Results

Algorithm Selected for CAFA

By considering the results presented above, we observed that *lin-sim k*-NN classifier improves prediction performance only slightly, while it is computationally costly and sensitive to the *lin-sim* threshold choice. Therefore, due to the time constraints of the competition, we decided not to use the *lin-sim* approach for score calculation. We used MS-kNN as our predictor because, as it can be seen from Table 4.4, it was more accurate than single-source kNN and both simpler and more accurate than other MS-kNN algorithms we studied. A given CAFA protein could be represented in one, two or three sources. If a data source was not available for a test protein, the score for that source in MS-kNN was set to zero. In this way, scores of proteins represented by multiple sources were biased upwards, reflecting increased prediction confidence.

For CAFA assessment we provided predictions for all 48,298 CAFA proteins and for all GO terms (8,728 MF terms and 18,982 BP terms). One of the rules of the competition was that, for the final submission, one protein cannot be associated with

more than 1,000 GO terms. Thus, we sorted the prediction scores for each protein and submitted the top 1,000 GO terms with the corresponding prediction scores. We note that for vast majority of CAFA proteins (44,471 out of 48,298) we only had sequence information available.

CAFA Proteins Used for Testing

Only 595 of the CAFA proteins were experimentally annotated after the submission deadline, and they were used to evaluate the prediction accuracy. Of these 595 proteins, 366 proteins were associated with MF functions and 436 with BP functions. In the evaluation set, there were 2,786 new MF annotations and 11,075 new BP annotations. Among the 595 proteins, only 10 were covered by all 3 data sources, while 66 were covered by 2 of the 3 data sources. For the remaining 519 proteins we only had sequence information.

Baseline Predictors

The CAFA organizers used the following 3 baseline algorithms for comparison with the submitted predictions.

(1) **Priors**. Prediction score of every protein for a given GO term was the same and was calculated as the probability of that GO term occurring in Swissprot. This approach made it more likely for a protein to be annotated with a more common GO term.

(2) **BLAST**. To obtain prediction score for annotation of a target protein with a GO term, the protein's sequence was compared with all protein sequences annotated with this GO term using BLAST. The sequence identity of the most similar protein was used as the prediction score.

(3) **Gotcha**. Using the same BLAST output as (2), Gotcha prediction score was calculated as the sum of the negative logarithms of the E-value of the alignments between the target protein and all proteins associated with the given term.

Evaluation Measures.

The CAFA organizers used 4 different evaluation methods; 3 of them were *protein-centric* and one was *function-centric*. In this paper, we report only on the AUC results for simplicity of analysis. In protein-centric evaluation methods, the prediction scores of each protein across all available GO functions are sorted. Then, AUC is calculated for each protein. In function-centric evaluation method, for each function, the prediction scores of all proteins associated with this particular function are sorted and AUC is calculated for each GO term.

*Threshold***.** At each threshold, precision and recall are calculated and reported as averages across all proteins. If a particular score for a term/protein pair is above the given threshold, then the annotation at that threshold is propagated towards the root of the ontology.

*Top N***.** For a particular protein, the scores are first sorted. Then, for the highest 20 scores, precision and recall are calculated. If there is a tie between more than one term, all such terms are used to calculate AUC.

*Weighted threshold***.** For each threshold, weighted precision and recall are calculated based on the information content of each term. The information content of a GO term is calculated from the January 2011 version of Swissprot, as the negative log of the

frequency of the term among proteins annotated with at least one experimental evidence code.

*TermAUC***.** For any GO term associated with 25 or more proteins from all organisms, precision and recall are calculated at each threshold after propagating annotations for testing proteins. We note that we used TermAUC during the design of our MS-kNN algorithm.

CAFA Results

The AUC scores for MF terms based on different evaluation methods are shown in Table 4.6. The AUC scores for BP terms based on different evaluation methods are shown in Table 4.7. The reported TermAUC accuracies are average AUC accuracy on 11 MF and 25 BP GO terms. We provide results for sequence-based kNN and MS-kNN. The results over the 4 different AUC accuracies show that MS-*k*NN worked the best overall on MF prediction. Particularly, the improvements in Threshold and Weighted threshold AUC were quite large. For the TermAUC, sequence-based kNN and MS-kNN had similar results. This could be explained by the fact that for 519 of the 595 test proteins we used only sequence information. In BP predictions, MS-kNN was also overall the most accurate, although in was not the most accurate on any of the 4 accuracy measures. For a further insight, TermAUC of the 11 MF terms based on different prediction methods are compared in Figure 4.2. We can see that MS-kNN was better than BLAST on all but one MF term. It was better than Gotcha on 7 out of the 11 MF terms.

Table 4.6. AUC scores for MF terms

Algorithm	Threshold	Top n	Weighted threshold	TermAUC
Prior	0.867	0.742	0.795	0.500
BLAST	0.794	0.779	0.734	0.634
Gotcha	0.786	0.774	0.728	0.665
k-NN (1 source)	0.814	0.780	0.747	**0.702**
MS-*k*NN (3 sources)	**0.883**	**0.784**	**0.819**	0.701

Table 4.7. AUC score for BP terms

Algorithm	Threshold	Top n	Weighted threshold	TermAUC
Prior	**0.898**	0.630	**0.822**	0.500
BLAST	0.771	0.633	0.697	0.648
Gotcha	0.748	0.637	0.677	**0.666**
k-NN (1 source)	0.811	**0.642**	0.724	0.651
MS-*k*NN (3 sources)	0.893	0.636	0.818	0.650

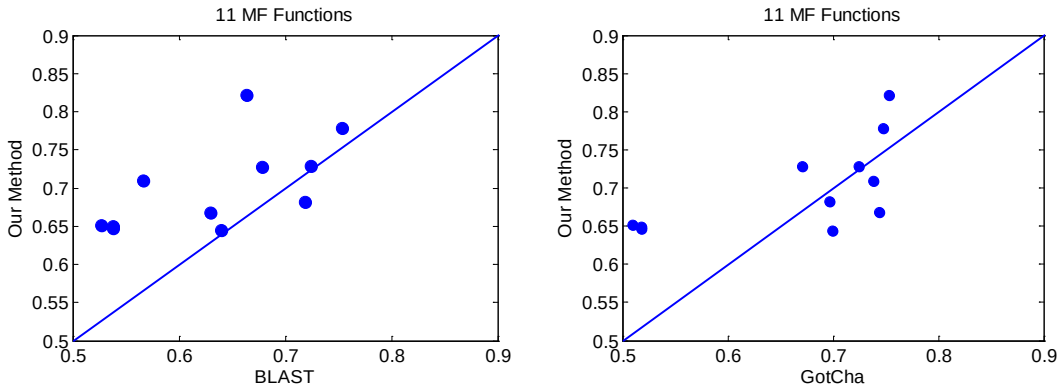

Figure 4.2. AUC comparison on 11 MF functions

Table 4.8. Comparison of AUC scores on 8 test proteins based on MF terms

Algorithm	Threshold	Top n	Weighted threshold
k-NN (1 source)	0.853	0.740	0.768
MS-*k*NN (3 sources)	0.949	0.845	0.910

Table 4.9. Comparison of AUC scores on 8 test proteins based on BP terms

Algorithm	Threshold	Top n	Weighted threshold
k-NN (1 source)	0.798	0.526	0.696
MS-*k*NN (3 sources)	0.920	0.526	0.846

Post-CAFA Analysis

We performed several additional experiments to get a better insight into the proposed algorithms and to explore some alternatives. Among the 595 test proteins, 66 proteins were in 2 data sources, and only 10 proteins in all 3 data sources. Among these 10 proteins, 8 of them were annotated with MF terms and 8 of them with BP terms. We studied results on these 10 proteins in more detail. While the results in Tables 4.8 and 4.9 should not be interpreted in terms of statistical significance due to small sample size (for that, we point a reader to Table 4.4, they provide an insight into improved accuracy of MS-*k*NN as compared to similarity-based *k*NN on 595 CAFA proteins. Because we had only 8 proteins for evaluation for both MF terms and BP terms, the TermAUC accuracies for GO terms were not reliable and are not shown. As can be seen, AUC of MS-*k*NN was much larger than that of similarity-based *k*NN on all accuracies except Top n AUC on BP terms. We note that these results are consistent with those reported in Table 4.4, that were obtained on 1,302 test proteins.

4.6 Conclusion

The protein function prediction is a complex problem. In this paper, we focused

on the question of how to integrate multiple data sources to improve the prediction accuracy. We discussed and evaluated several different integration schemes in this paper. Our pre-CAFA and CAFA results strongly indicate that integrating information from multiple data sources could improve protein function prediction accuracy. At the level of sequence similarity-based predictions, we observed that it is beneficial to consider all available annotated proteins, regardless how evolutionary distant they are from a query protein. Considering the time limitations associated with the tight deadline for submission of the CAFA predictions, our strategy to use the simple and efficient *k*-NN algorithm, coupled with simple integration of prediction scores from multiple data sources, proved to be reasonable.

There are certainly many avenues for future improvements of function predictions. A straightforward one is to include as many available sources of structural and functional protein information. For example, in CAFA, we used only microarray data from a single, albeit commonly used, human microarray platform. Information beyond microarray data and protein-protein interaction data, such as chromosomal neighborhood of a gene, mutations, role in various diseases, or protein structure, could certainly be valuable. Based on our experience during CAFA, we think that further advances in statistical approaches for function prediction are needed. Particularly, we would like to point to two open problems we believe could lead to significant advances in protein function prediction. One is related with the observation that a lack of a protein's annotation with a certain GO term should not be treated as negative evidence, but rather as a missing label. As a consequence, it would be advisable to treat function prediction problem as a one

class classification, instead of binary classification. Another is the problem of sampling bias, created by the fact that available annotations are not a random sample from the protein/term space. Developing methods that are robust to sampling bias or the ones that could correct its negative effects should be one of the priorities of future research in the field.

CHAPTER 5

SPATIAL SCAN FOR MOVEMENT DATA

5.1 Background

Disease mapping methods are used to understand the geographic variability in disease risk by studying the association between the occurrence of disease and the locations of individuals in the populations. It is an essential tool in modern epidemiology, because location servers as a proxy for lifestyle, social and environmental factors that may be unobserved or unavailable for study. Disease maps have served as a hypotheses generating tool, allowing investigators to draw inferences about disease etiology and inform allocation of public health resources.

There are two major approaches for disease mapping. The first method named disease clustering aims to find spatial regions whose population has significantly higher disease risk than the background populations. This widely used approach stems from Kulldorff's spatial scan statistics (Kulldorff et al., 1997). This statistic is widely used in identifying cluster of disease cases, such as identifying environmental risk factors for child leukemia (Kulldorff et al., 1997) and detection of bioterrorist attacks (Neill et al., 2006). It currently has many variants (Kulldorff et al., 2005, Tango and Takahashi, 2005) that can be used for various types of data. Spatial scan received attention in data mining community from the perspective of computational efficiency (Neill et al., 2004; Neill et al., 2005). The second method for disease mapping aims to determine the actual spatial map of disease risk. This approach typically relies on computationally expensive Bayesian Hierarchical Modeling (Banerjee et al., 2003; Mollie 1996) to exploit spatial

correlation in disease risk. Several Bayesian spatial models have been proposed for disease mapping (Best et al., 2005). While the output of disease clustering and disease risk estimation is different, both methods require the same input – information about location and disease status of individuals from the population under study.

The existing disease mapping methods only uses the populations' residence information as location information for detecting high risk region. However, in case of environmental diseases (e.g., caner, asthma), the causative exposures may occurs in other places which are far away from the individual's current residence. For example, if the true high risk areas are nonresidential locations visited by many people, such as airport or a business district, then the existing disease mapping approaches are unable to find these regions because they only consider the residential information.

In recent years, several positive developments for the epidemiology of mobile populations occurred. One is related to the recent interest in 'life course' approach to health (Pickles et al., 2007), which emphasizes the significance of timing in associations between physical (e.g., chemical, sun exposure) and social (e.g. poverty, employment) exposures and chronic diseases. Q-statistic (Jacquez et al., 2005; Jacquez et al., 2007) and M-statistic (Manjourides and Pagano 2011) are recently proposed methods to analyze residential history and disease risk. These two methods are heuristically motivated by spatial scan and use a strong assumption that the studied population has similar moving trajectories.

Until recently, the main obstacle in using movement data for disease mapping was a lack of technology to collect such data. However, the almost ubiquitous presence of

mobile and smart phones, as well as the emergence of geocoded databases about residential histories, make it is possible to obtain detailed and accurate information about mobility of human population at an unprecedented scale with low-cost. This development offers a significant opportunity to improve the quality of disease surveillance using movement data.

In this chapter, I present a novel approach which extends the original spatial scan to movement data by modeling the individual's disease risk using classical logistic regression model. Given the information about the movement of individuals and their health status, we assume that the probability of each individual being sick is the logistic function of the weighted sum of the time spent at visited locations of this person. We have designed a new test statistic working on movement data based on the likelihood ratio. This proposed test can be used to detect cluster of any size, located anywhere in the study region. Due to the computational bottleneck of computing the statistic and the significance testing by Monte Carlo replication, I also present an efficient algorithm to compute the spatial scan statistic for movement data. It can be shown that the computational requirements of our proposed method are reasonable even for a large population.

5.1 Problem Definition

Let us assume we are given a spatial region inhabited by N individuals and consisting of L locations. We denote the disease status of the i-th individual as $y_i = 1$ if he or she is sick, and $y_i = 0$ otherwise. Each individual could be expressed as $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots,$

x_{iL}], where x_{il} is the fraction of total time the i-th individual spent at location l ($\sum_{l=1}^{L} x_{ij} = 1$). We denote $\mathbf{r} = [r_1, r_2, \ldots, r_L]$ as a vector of spatial risks, where r_l is the disease risk of the l-th location. Our objective is to find a sub-region with the highest disease risks.

5.2 Methodology

Original Spatial Scan

The original spatial scan (Kulldorff et al., 1997) only works on residential data. So, in the setting of the original spatial scan, $\mathbf{x}_{il} = 1$ if location l is the i-th individual's residence, otherwise $\mathbf{x}_{il} = 0$. In the original spatial scan, the study region is partitioned into cells, and a pair (c_l, p_l) can be easily obtained by summarizing input data X, where c_l is the number of people residing in the l-th cell with a disease or a symptom, and p_l is the number of people residing in the cell. For any considered subregrion l, spatial scan statistic computes the log of the likelihood ratio of two possibilities: that the disease risk in region l is higher than the outside region, and that the disease risk is identical inside and outside the region. With the sufficient statistic, (c_l, p_l) for the l-th region and (c_{tot}, p_{tot}) for the entire grid, we compute the score of this region as $S_l = c_l \log(c_l/p_l) + (c_{tot} - c_l)\log((c_{tot} - c_l)/(p_{tot} - p_l)) - c_{tot}\log(c_{tot}/p_{tot})$, if $c_l/p_l > c_{tot}/p_{tot}$, and 0 otherwise. Kulldorf et al., proved that the spatial scan statistic is individually the most powerful for finding a significant region of elevated disease risk: it is more likely to detect the overdensity than any other test statistic.

Spatial Scan for mobile populations

Let us denote **r** as a spatial field representing disease risks and assume the disease risk for each *i*-th individual equals $\rho_i = \frac{1}{1+\exp(-\mathbf{r}^T\mathbf{x}_i)}$, which represents the probability of the *i*-th individual being sick. Let us denote by r_{in} the risk inside a candidate sub-region *R* and r_{out} the risk outside the sub-region *R*. We use $x_{i,in}$ as the fraction of time spent by the *i*-th individual within sub-region *R* and $x_{i,out}$ as the fraction of time spent outside sub-region *R*. The risk for *i*-th individual can be expressed as $\rho_i = \frac{1}{1+\exp(-(r_{in}x_{i,in} + r_{out}x_{i,out}))}$. The alternative hypothesis is $\boldsymbol{H_1}$: $r_{in} > r_{out}$, stating that the risk within *R* is higher than the background risk. The null hypothesis is $\boldsymbol{H_0}$: $r_{in} = r_{out}$, stating that disease risks are equal within and outside *R*.

We use the likelihood ratio test. Let us express the likelihood function as

$$L(S, r_{in}, r_{out}) = \prod_{i=1}^{N} \rho_i^{y_i} (1-\rho_i)^{1-y_i}, \tag{5.1}$$

and the likelihood ratio as

$$S_R = \frac{\sup_{r_{in}>r_{out}} L(R, r_{in}, r_{out})}{\sup_{r_{in}=r_{out}} L(R, r_{in}, r_{out})}, \tag{5.2}$$

when $r_{in} = r_{out}$, we can write the likelihood as

$$L(R, r_{in} = r_{out}) = \prod_{i=1}^{N} \rho^{y_i} (1-\rho)^{(1-y_i)} = \rho^C (1-\rho)^{N-C}], \text{ where } \rho = \frac{1}{1+\exp(-r_{in})}.$$

The denominator in equation (5.2) then becomes

$$\sup[_{r_{in}=r_{out}} L(S, r_{in}, r_{out})] = \frac{C^C (N-C)^{N-C}}{N^N} \overset{def}{=} L_0, \tag{5.3}$$

because the maximal likelihood is obtained when $\rho = C/N$, where C is the number of sick individuals in total population N. Therefore, L_0 depends only on the total number of sick individuals C, not on their spatial distribution, and is a constant.

Now, we would like to find the value of numerator in (5.2). For a given fixed region R, we maximize over all possible $r_{out} \le r_{in}$. Now, let us denote $\mathbf{x}_i = [x_{i,in}\ x_{i,out}]$, where $x_{i,in}$ as the fraction of time spent by the i-th individual within sub-region R and $x_{i,out}$ as the fraction of time spent outside sub-region R. $\mathbf{r} = [r_{in},\ r_{out}]$ and rewrite $\rho_i = \frac{1}{1+\exp(-\mathbf{r}^T \mathbf{x}_i)}$. Instead of maximizing (5.1), we can maximize the log-likelihood:

$$\begin{aligned} \max \log L(S, r_{in}, r_{out}) &= \sum_{i=1}^{N} [y_i \log(\rho_i) + (1-y_i)\log(1-\rho_i)] \\ s.t.\ r_{in} &> r_{out} \end{aligned} \tag{5.4}$$

The gradient of the objective is

$$\mathbf{g} = \sum_{i=1}^{N} [(y_i - \rho_i)\mathbf{x}_i], \tag{5.5}$$

and the Hessian of the objective is

$$H = -\sum_{i=1}^{N} [\rho_i (1-\rho_i)\mathbf{x}_i \mathbf{x}_i^T]. \tag{5.6}$$

Note the Hessian matrix is non-positive definite, which means the objective is concave and a unique global optimal solution can be obtained. The Newton method updates the parameter $\mathbf{r}$ as :

$$\mathbf{r}^{new} = \mathbf{r}^{old} - (H)^{-1}\mathbf{g}. \tag{5.7}$$

The Hessian matrix is with size 2 × 2 allowing efficient learning.

Now, let us consider the constraint $r_{out} \le r_{in}$. We are only interested on the regions R where $r_{in} > r_{out}$. After solving (5.7), if we get a solution where $r_{in} < r_{out}$, we will set the solution to be $r_{in} = r_{out}$ and the corresponding likelihood ratio to 1.

Therefore, we can get the log-likelihood ratio for sub-region R as:

$$S_R^{mobile} = \begin{cases} \log \frac{\max_{\mathbf{r}} L(\mathbf{r})}{L_0} = \log \max_{\mathbf{r}} L(\mathbf{r}) - \log L_0 & if\ r_{in} > r_{out} \\ 0 & if\ r_{in} \le r_{out} \end{cases} \quad (5.8)$$

5.3 Efficient Spatial Scan Algorithm

The Most Risky Cluster

We would like to pick a sub-region with the highest log-likelihood ratio. This maximum value defines the test statistic as

$$\lambda = \max_R S_R^{mobile} \quad (5.9)$$

Let denote the grid size as $K \times K$. To obtain the value of λ, we need to compute S_R^{mobile} for all squares with size $m = 1, \ldots, K$. For any size m, there are $(K - m + 1)^2$ regions. So there are $O(K^3)$ regions to examine. Training the logistic regression model takes O(N) time, where the N is the size of population. Therefore, the naïve implementation requires $O(K^3N)$ time to compute λ.

Monte Carlo-based Hypothesis Testing

The distribution of λ depends on the underlying inhomogeneous population distribution and it has no simple analytic form. So, the only principal approach is to use the Monte Carlo method to sample from the exact distribution of λ. In each Monte Carlo replication, we need to calculate λ on a randomized population where sick labels are

randomly shuffled. Let us denote the number of replications as *B*. The total time complexity becomes $O(BK^3N)$. Usually, *B* is typically set to a large number (i.e., *B* = 1000). Therefore, to detect a significant cluster using naïve implementation could be extremely computationally expensive. To remedy this, we propose an algorithm to speed up the process.

Speeding Up by Discretization

Calculating the Hessian matrix (5.6) requires O(*N*) time. This process can be slow if the number of individuals in a population is large. Here, we propose a data compression technique to reduce the training size. The $\mathbf{x}_{i,in}$ is within range [0, 1], we divide the range with *M* bins. The examples with the same discretized value $\mathbf{x}_{i,in}$ and label y_i are grouped together. After discretization, the new dataset can be represented as $\{x_b, c_b^+, c_b^-\}_{b=1}^M$, where $\mathbf{x}_b$ is corresponding discretized value of the *b*-th bin, and c_b^+ and c_b^- is count of examples in discretized bin *b* of positive and negative labels respectively. Note that *M* could be orders of magnitude smaller than *N*. Therefore, (5.4), (5.5), (5.6), (5.7) can be rewritten as weighted logistic regression,

$$\begin{aligned} \max \log L(S, r_{in}, r_{out}) &= \sum_{i=1}^{M} [c_i (y_i \log(\rho_i) + (1 - y_i) \log(1 - \rho_i))] \\ s.t\ r_{out} &\le r_{in}, \end{aligned} \tag{5.10}$$

$$\mathbf{g} = \sum_{i=1}^{M} [c_i (y_i - \rho_i) \mathbf{x}_i], \tag{5.11}$$

$$H = -\sum_{i=1}^{M} [c_i \rho_i (1 - \rho_i) \mathbf{x}_i \mathbf{x}_i^T], \tag{5.12}$$

$$\mathbf{r}^{new} = \mathbf{r}^{old} - (H)^{-1} \mathbf{g}, \tag{5.13}$$

Therefore, the time complexity to solve weighted logistic regression is $O(M)$. In our experimental section, we will show that setting M to 100 is enough to get a satisfyingly accurate solution. Therefore, assuming one time $O(N)$ effort for discretization, the time for calculating λ using weighted logistic regression becomes $O(N) + O(K^3)$. $O(K^3)$ is expected to dominate $O(N)$ in real life scenarios. Thus, in the following subsections, we will disregard the $O(N)$ cost.

Speed-up by pruning

We need to find the statistic λ for each replication and each replication requires $O(K^3)$ time. This is the key step that requires the most computing time. However, the number of sick persons is typically small. Due to the sparsity structure of the movement data, the number of locations that are visited by sick persons is also typically small. Therefore, most of the time, it is not crucial to explore all possible squares. For example, for a given subregion with size $m \times m$ in the grid, as shown in Figure 5.1, let us use *subarea1* to denote the first row and column and use *subarea2* to denote the last row and column. If *subarea1* or *subarea2* are not visited by any sick person, then we do not need to compute the log-likelihood ratio of this larger square $(m + 1) \times (m + 1)$, as it cannot have larger score than the upper left $m \times m$ square or the bottom right $m \times m$ square.

Because of the sparsity of movement data, this pruning procedure could reduce computational time.

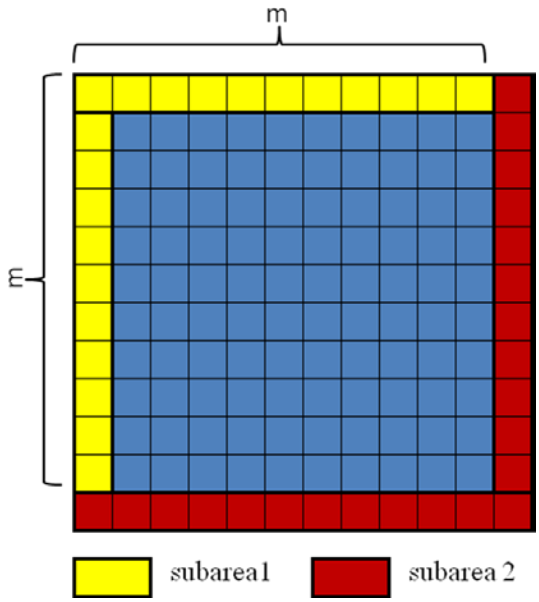

Figure 5.1. Speed up by pruning

5.4 Experimental Setting and Results

EpiSims Data

In order to evaluate the proposed spatial scan algorithm and to compare usefulness of residential and movement data in detecting significant overdensity clusters, we used EpiSims dataset from Network Dynamics and Simulation Science Laboratory. This synthetic data set contains the activities of 1,601,329 peoples over 240,090 locations. It has been designed to realistically simulate behavior of the population of Portland, OR, at the level of individual people. This dataset contains information about the movement of individuals, the types of their activities, and social contacts.

We first processed the original EpiSims data into a regular 256 × 256 grid. Each cell represents a location. In Figure 5.2, we show the total time spent on each location (in log scale) in the 256 × 256 grid. As shown in these two figures, the distributions of density of residential and movement data are very similar.

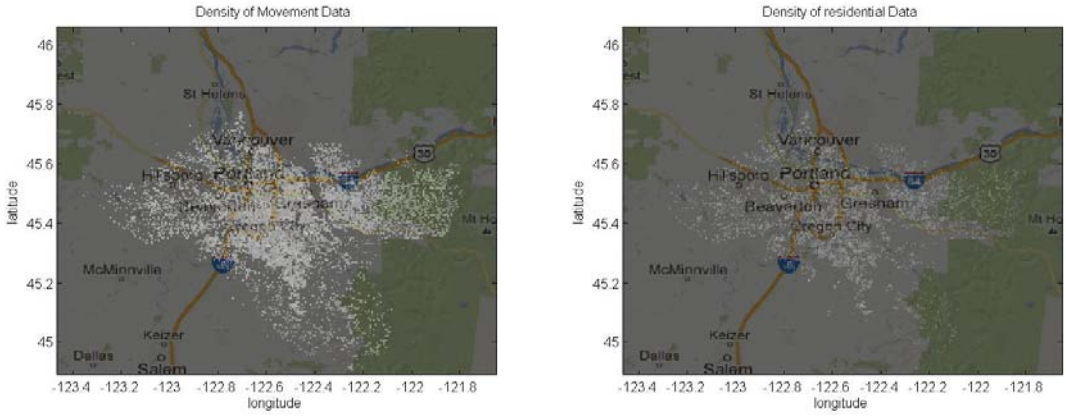

Figure 5. 2. Density of residential (left) and movement (right) data for EpiSims Portland, OR.

Data Generation

Considering the distribution of the movement density, we focused on the region with between blocks 51 and 200 on *X*-axis and between blocks 37 and 186 on *Y*-axis from the original 256 × 256 grid. Then we transformed the resulting 150 × 150 grid into a coarser 50 × 50 grid. The fraction of total time spent on this coarser 50 × 50 grid was 95.5%. In the following experiments, we picked several square regions as our true risk regions. We specify r_{in} value within a selected high risk region and r_{out} for the remaining locations. We selected r_{in} to be larger than r_{out}. To generate the target y_i for *i*-th individual, we first computed the probability $\rho_i = \frac{1}{1+\exp(-\mathbf{r}^T \mathbf{x}_i)}$. Then the labels $y_i \in \{0, 1\}$ were generated

by this probability. In the following experiments, we evaluate the performance of the proposed algorithm on movement data and residential data.

Experiments: Scenario 1

In our first experiment, we set $\boldsymbol{r}_{in}$ = log (199) and $\boldsymbol{r}_{\text{out}}$ = log (999) such that an individual spending all time inside the region would have disease probability ρ_i = 0.005, while an individual spending all time outside would have disease probability ρ_i = 0.001. In this setting, we randomly sampled N = 100,000 people. Then we used our proposed mobile spatial scan to detect the most significant cluster. In our scenario 1, we used a square (denoted the red solid square in Figure 5.3) with size 3 $\times$ 3 centered on "*Milwaukie business industrial*" as the true risk region. This location was chosen because both residential time and travel time spent in this region were larger than on other locations. The detected highest risk regions based on movement data and residential data are shown in Figure 5.3. The detected cluster (denoted as the red dotted square in Figure 5.3) based on movement data was centered within the true risk region. The size of this detected square was 6 $\times$ 6. The resulting value for the test statistic λ was 12.17. If only static residential data was used, the detected region was the black dotted square as shown in Figure 5.3. The size of this square was 11 $\times$ 11 and the resulting value for the test statistic λ of this detected square was 5.87. The Figure 5.4 shows the distribution of the computed statistics based on 100 repetitions based on shuffled movement data and static data. The detected region based on movement data was significant (p value was 0.00) while the detected cluster using the static data was not significant (p value was 0.39). So, using residential data spatial scan did not detect any significant region with high disease

risk. However, using the movement data we were able to detect a significant over-density cluster around the true risk region.

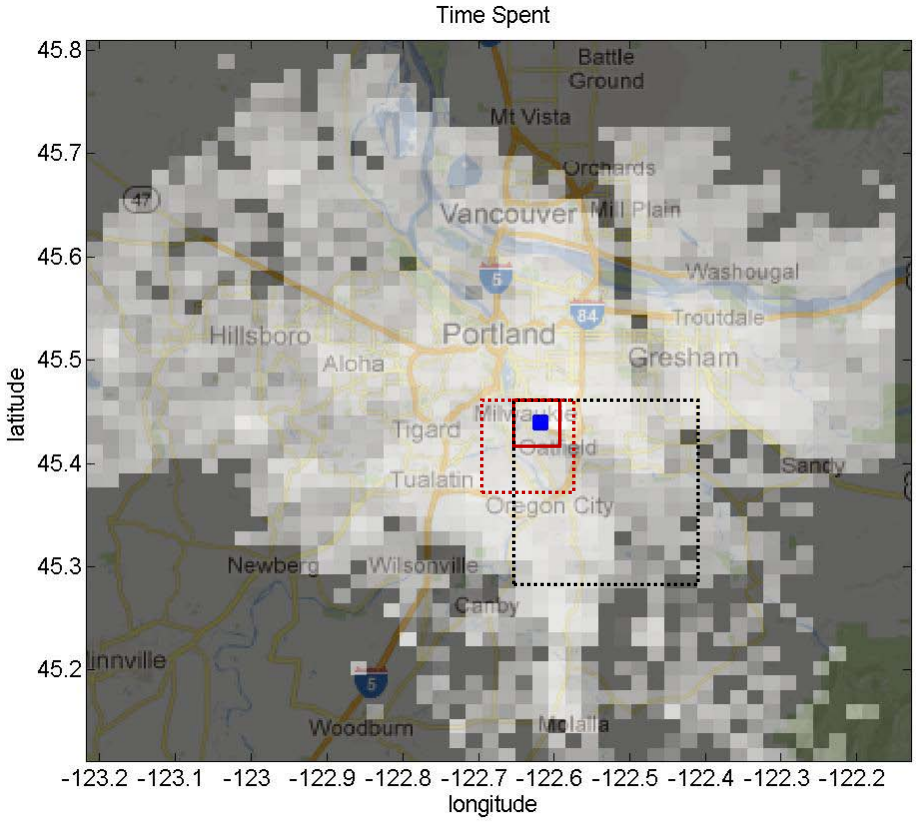

Figure 5.3. Detected region for Scenario 1.

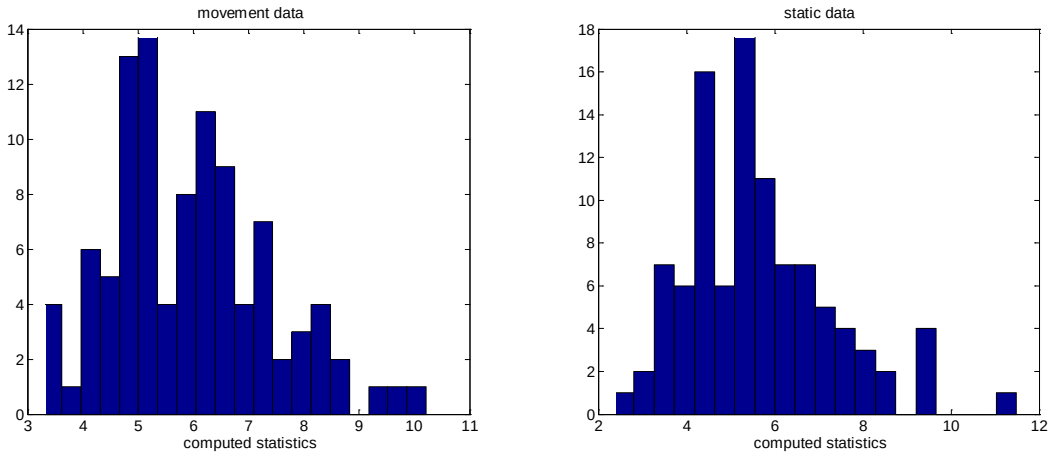

Figure 5.4. The distribution of the λ statistics on shuffled data for (a) movement data and (b) static residential data

Experiments: Scenario 2

In our second experiment, we chose Portland international airport as the true risk region. This region was chosen because it was an example of the regions in which a few people resided but which are visited by many people. Therefore, only using residential information is not expected to lead to detection of the high risk region. In this scenario, we tested our spatial scan algorithm under several different choices of the risk and size of the high risk region.

Setting 1. In the first case, we set r_{in} = log (199) (i.e. ρ_i = 0.005) and r_{out} = log (999) (i.e. ρ_i = 0.001). We randomly sampled N = 100,000 people from the whole population. In this setting, we used a square (denoted red dotted square in Figure 5.3) with size 3 × 3 centered on Portland international airport as the true risk region. Compared to scenario 1, we only changed the location of true risk region. The detected regions based on movement data (dotted red square) and static data (dotted black square)

are shown in Figure 5.5. The detected region based on movement data was around the true risk region with significance level at 0.22. The detected region based on static data was far from the true risk region and its significant level was 0.49. In this case, none of these two algorithms returned any significant risky region. The reason was that both the risk factor and the size of true risk region were very small. Then the number of sick people would be very small (around 0.1% of the population) in the data. The difference between r_{in} and r_{out} is difficult to be detected in this setting. So, the *p*-value based on Monte Carlo testing is large. However, it should be observed that in this setting the most risky cluster detected based on movement data was still around the true risk region, while the best region found using residential data was elsewhere. In the following two setting, we show the results with higher risk factor and larger region size.

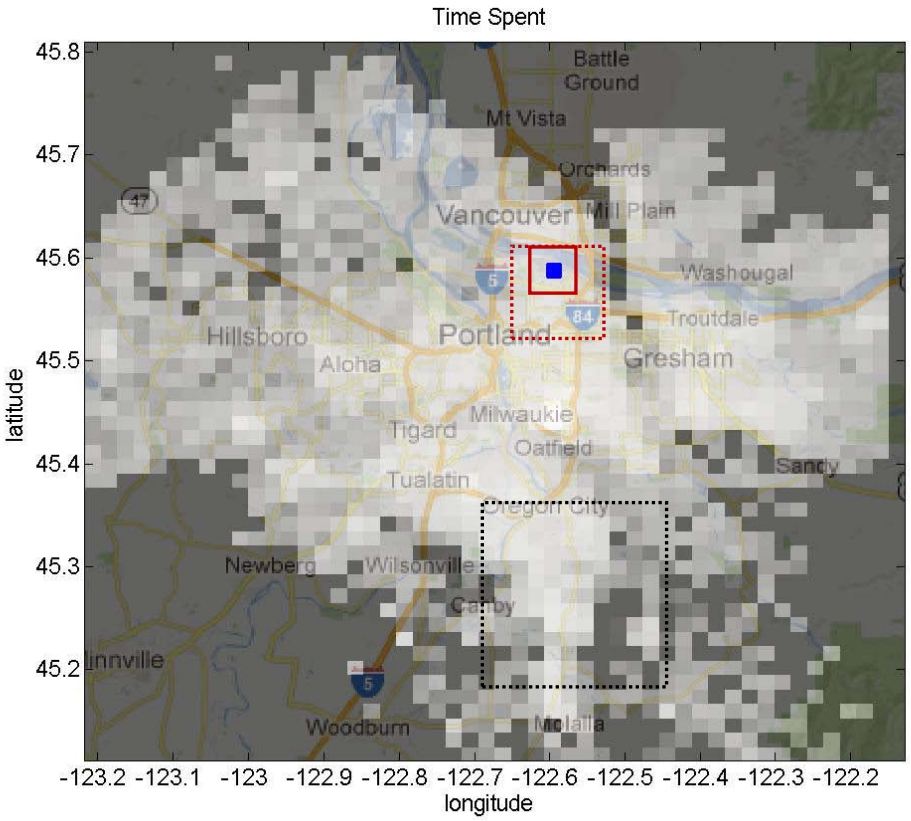

Figure 5.5. detected region for scenario 2 on setting 1.

Setting 2. In our second case, we increased the r_{in} from r_{in} = log(199) (i.e. ρ_i = 0.005) to r_{in} = log (99) (i.e. ρ_i = 0.010). The r_{out} was still fixed at log (999) (i.e. ρ_i = 0.001). In this case, the risk factor difference between r_{in} and r_{out} was larger than in the previous setting. The detected regions based on movement data and static data are shown in Figure 5.6. The detected region based on movement data was around the true risk region at significance level at 0.02. The detected region based on static data did not overlap the true risk region and its significance level was 0.42. The result clearly shows using movement data could result in better performance than only using residential data.

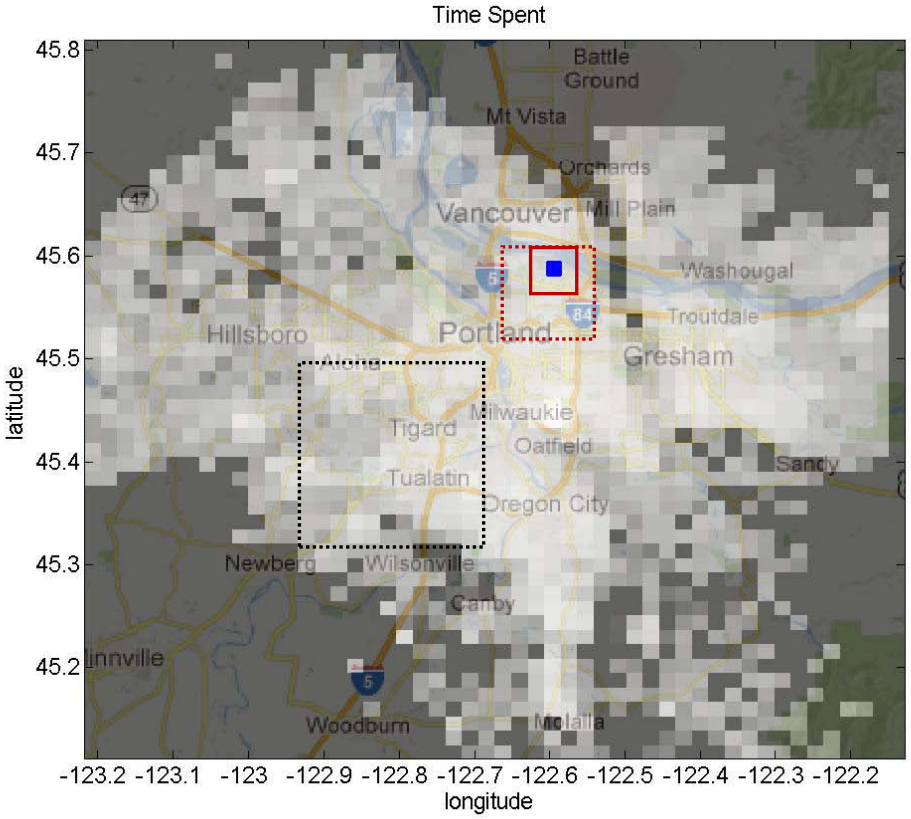

Figure 5.6. detected region for scenario 2 on setting 2.

Setting 3. In the third case, we increased the size of the true region to 15 × 15. All other parameters were the same as in **Setting 1.** Under this setting, the detected regions both based on movement data and static data were around the true risk region and with significant levels at 0.00. The true risk region and detected regions are shown in Figure 5.7. The solid red square is the true risk region. The detected region based on movement data was exactly the same as the true region. So, in Figure 5.7, the dotted red square (detected based on movement data) was overlapped with the solid red square (true risk region). Using only residential data can also detected the correct cluster (dotted black square with size 14 × 14).

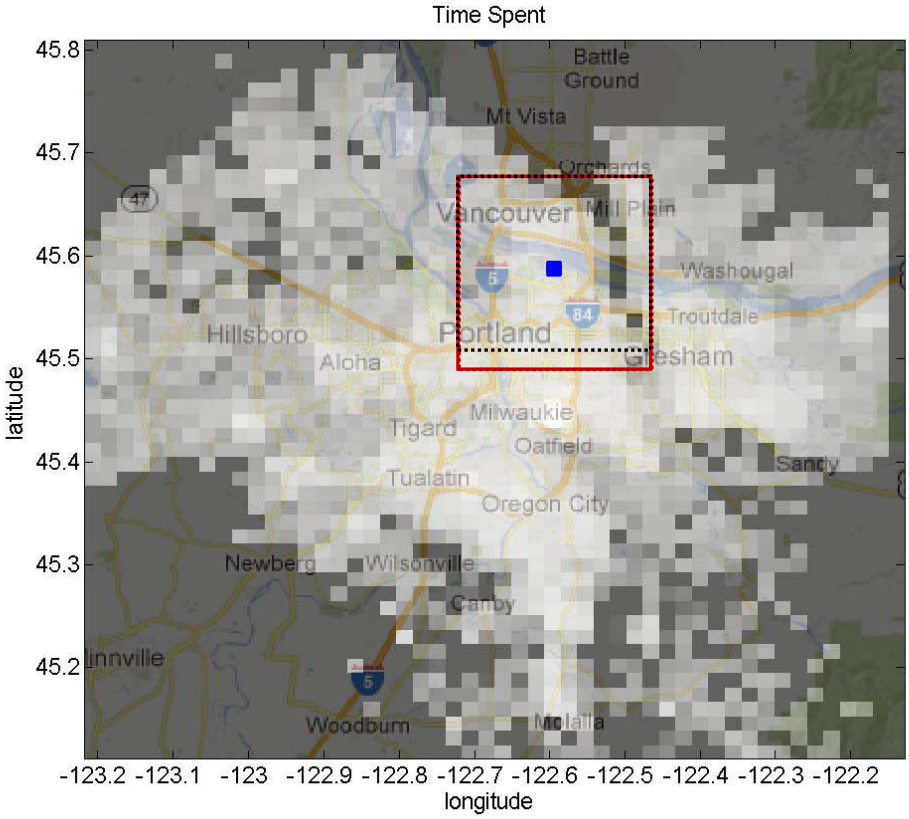

Figure 5.7. detected region for scenario 2 on setting 3.

Computing Time vs. Spatial Resolutions

In this section, we would like to check the required computing time based on different resolutions. We studied the cases when the resolutions are 150 × 150, 75 × 75, 50 × 50, 30 × 30 and 10 ×10. The required time was shown in Figure 5.8. As expected, the computing time is cubic with respect to the resolution size.

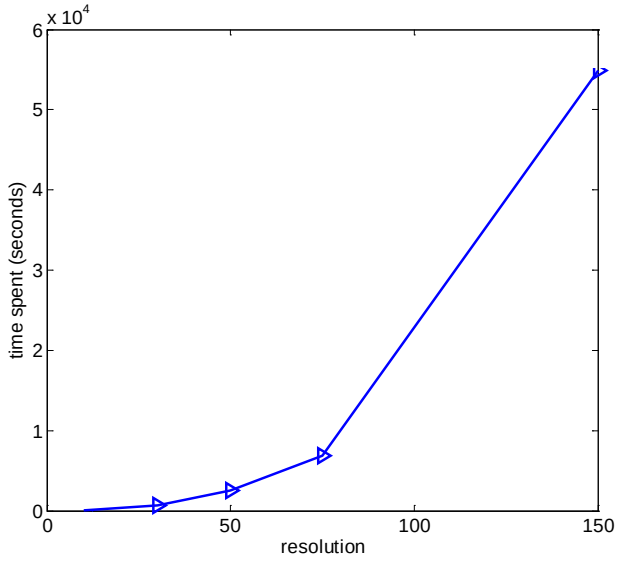

Figure 5.8 The running time of proposed spatial scan for different resolutions

Data Discretization

As discussed in section 5.3, we used data discretization technique to speed up the training time of logistic regression models. In this section, we would like to check how the discretization step affects the solution accuracy and the running time. Let us denote the optimal solution obtained from (5.4) as $\mathbf{r}_{opt}$ and the approximated solution using discretization from (5.10) as $\mathbf{r}_{appr}$, we used $||\mathbf{r}_{opt} - \mathbf{r}_{appr}||_2/||\mathbf{r}_{opt}||_2$ to denote the solution difference, where $||\bullet||_2$ denotes the l_2 norm. In our experimental setting, we increased the number of bins from M = 10 to 100,000. As shown in left subfigure of Figure 5.9, we got very accurate approximate solution when the number of bins to 100. By increasing the number of bins from 100 to 100,000, the accuracy of log-likelihood estimation improved only slightly (0.03%). However, the running time increased linearly as shown in right

subfigure in Figure 5.7. Therefore, by setting the number of bins to 100, we could get a good tradeoff between solution accuracy and running time.

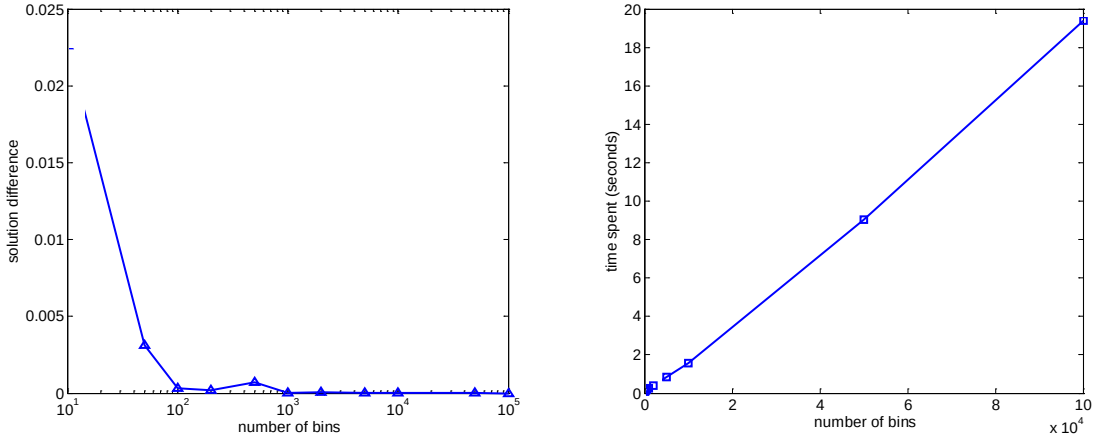

Figure 5.9 Solution difference (left) and time spent (right) based on different number of bins.

5.5 Conclusion

In this chapter, I presented the proposed method which extends the original spatial scan to movement data. Due to the computational bottleneck of computing the statistic and the significance testing by Monte Carlo replication, an efficient algorithm to compute the spatial scan statistic was proposed. The required computational time is acceptable even for a large population and fine spatial grid resolution. We have performed several experiments to check the difference between using movement and residential data. The experiments clearly show that, if the true risk regions are the locations where a few people resided but many people visited, the movement data is much more useful than residential data. This novel algorithm is very useful for disease monitoring, especially for the environmental diseases (e.g., caner, asthma) where the causative exposures may occurs in the other places which are far away from the individual's current residence. In

the future, we would like to further improve the computational efficiency to enable our algorithm to run for even bigger spatial resolution. Another interesting direction is to explore the difference between the original spatial scan and the proposed test statistic when only residential data are used.

CHAPTER 6

CONCLUSION

In this dissertation, I present a several novel data mining algorithms for classification of complex biomedical data. The multi-task feature selection algorithm presented in chapter 2 can use rich public repositories of microarray and related data to improve the classification of target microarray classification problem having small train data. In chapter 3, I presented a novel feature filter method to select a feature subset with discriminative power and minimal redundancy. This proposed feature filter method is based on quadratic optimization problem with binary integer constraints. I also presented techniques to solve the problem efficiently. This proposed feature selection method can was extended to multi-task classification problem. In chapter 4, I describe a new algorithm for protein function prediction by integrating multiple data sources. Several useful insights based on the experimental results were discussed and presented in chapter 4. In chapter 5, I presented a novel algorithm to extend the original spatial scan to detect significant cluster from movement data. The experimental results showed that the movement data are very useful, which can be used to improve the accuracy of disease monitoring.

REFERENCES CITED

Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ (1999). “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays,” The Proceedings of the National Academy of Sciences USA 96(12):6745-6750.

Altschul, S. F., Madden, T. L., Schffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” Nucleic Acids Research 25, 17, 3389–3402.

Andersen ED, Andersen KD. (2000). “The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm,” High Perf Optimization, 197-232.

Ando, R. and Zhang, T. (2005). “A framework for learning predictive structures from multiple tasks and unlabeled data,” Journal of Machine Learning Research, 6, pp. 1817–1853.

Argyriou, A., Evgeniou, T. and Pontil, M. (2008). “Convex multi-task feature Learning,” Machine Learning, 73(3), pp. 243-272.

Ashburner M, Ball CA, Blake JA, et al.: (2000). “Gene Ontology tool for the unification of biology, The gene ontology consortium,” Nature Genetics, 25: 25-29.

Banerjee S., Carlin B.P., Gelfand A.E., (2003). “Hierarchical Modeling and Analysis for Spatial Data,” Chapman & Hall, London, UK.

Barutcuoglu Z, Schapire RE, Troyanskaya OZ (2006). “Hierarchical multi-label prediction of gene function,” Bioinformatics, 22(7): 830-836.

Beer DG, Kardia SL, Huang CC, Giordano TJ, Levin AM, Misek DE, Lin L, Chen G, Gharib TG, Thomas DG et al (2002). “Gene-expression profiles predict survival of patients with lung adenocarcinoma,” Nat Med, 8(8):816-824.

Best, N., Richardson, S. and Thomson, A., (2005). “A comparison of Bayesian models for disease mapping,” Statistical Methods in Medical Research, vol. 14, pp. 35-59.

Bickel, S., Bogojeska, J., Lengauer, T. and Scheffer, T. (2008). “Multi-task learning for HIV therapy screening,” Proceedings of 25th International Conference on Machine Learning, pp. 56-63, Helsinki, Finland.

Bickel, S., Sawade, C. and Scheffer, T. (2009). “Transfer learning by distribution matching for targeted advertising,” Advances in neural information processing systems, pp. 145–152.

Brown M, Grundy W, Lin D, et al., (2000). “Knowledge-based analysis of microar-ray gene expression data by using support vector machines,” Proc Natl Acad Sci USA 2000, 97: 262-267.

Caruana, R. (1997) “Multitask learning,” Machine Learning, 28, pp. 41–75.

Cawley, G. C., and Talbot N.L.C. (2006). “Gene selection in cancer classification using sparse logistic regression with Bayesian regularization,” Bioinformatics, 22(19), pp. 2348-2355.

Chen, Y., Li, Z., Wang, X., Feng, J and Hu, X. (2010). “Predicting gene function using few positive examples and unlabeled ones,” BMC Genomics, 11 (Suppl 2): S11.

D'haeseleer P (2005): “How does gene expression clustering work?” Nat Biotech, 23: 1499-1501.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). “Least angle regression,” Annual Statistics, 32, pp. 407–499.

Eisen MB, Spellman PT, Browndagger PO, Botstein D (1998). “Cluster analysis and display of genome-wide expression patterns,” Proc Natl Acad Sci USA, 95: 14863-14868.

Friedberg I (2006): “Automated protein function prediction—the genomic challenge,” Brief Bioinform, 7: 225-242.

Golub, T.R., Slonim, D.K. et al, (1999). “Molecular classification of cancer: class discovery and class prediction by gene expression monitoring,” Science, 286, 531-537.

Guyon,I. and Elisseeff,A., (2003). “An introduction to variable and feature selection,” Journal of Machine Learning Research 3, 1157–1182.

Guyon,I., et al. (2002). “Gene selection for cancer classification using support vector machines,” Mach. Learn., 46 389–422.

Hall, M. A. (2000). “Correlation-based feature selection for discrete and numeric class machine learning,” In Proceedings of the International Conference on Machine Learning, pages 359–366.

Harris MA, Clark J, Ireland A, Lomax J, Ashburner M, Foulger R, Eilbeck K, Lewis S, Marshall B, Mungall C, Richter J, Rubin GM, Blake JA, Bult C, Dolan M, Drabkin H, Eppig JT, Hill DP, Ni L, Ringwald M et al, Gene Ontology Consortium (2004). “The Gene Ontology (GO) database and informatics resource,” Nucleic Acids Res 32 (database issue): D258–D261.

Hishigaki H, Nakai K, Ono T, Tanigami A, Takagi T (2001). “Assessment of prediction accuracy of protein function from protein–protein interaction data,” Yeast, 18: 523-531.

Jacquez, G. M., Kaufmann, A., Meliker, J., Goovaerts, P., AvRuskin, G., Nriagu, J., (2005). “Global, local and focused geographic clustering for case-control data with residential histories” Environ Health, Vol.4.

Jacquez, G.M., Meliker J., Kaufmann, A., (2007). “In search of induction and latency periods: space-time interaction accounting for residential mobility, risk factors and covariates” Int J Health Geogr, vol. 6, pp. 35

Kanehisa M., Goto S., Kawashima S., Okuno Y. and Hattori M. (2004). “The KEGG resource for deciphering the genome,” Nucleic Acids Res: D 277-80.

Kira, K., Rendell, L. A. (1992), “A practical approach to feature selection,” Machine Learning, 249–256.

Krishnapuram, B., Carin, L., Figueiredo, M. and Hartemink, A. (2005). “Sparse multinomial logistic regression: Fast algorithms and generalization bounds,” Machine Intelligence, 27(6), pp. 957–968.

Kulldorf, M., (1997). “A spatial scan statistic,” Communications in Statistics - Theory and Methods, vol. 26, pp. 1481-1496.

Kulldorf, M., Heffernan, R., Hartman, J., Assuncao, R., Mostashari, F.A, (2005) "Space-time permutation scan statistic for disease outbreak detection," PLos Med., Vol. 2, pp. 59

Lan, L., Djuric, N., Guo, Y., Vucetic, S., (2011). “Protein Function Prediction by Integrating Different Data Sources,” Automated Function Prediction SIG 2011 featuring the CAFA Challenge: Critical Assessment of Function Annotations (AFP/CAFA 2011), Vienna, Austria.

Lan, L., Vucetic, S., (2009). “A Multi-Task Feature Selection Filter for Microarray Classification,” IEEE Int’l Conf. on Bioinformatics and Biomedicine (BIBM), Washington, D.C.

Lan, L., Vucetic, S., (2011). “Improving Accuracy of Microarray Classification by a Simple Multi-Task Feature Selection Filter,” International Journal of Data Mining and Bioinformatics, Vol. 5 (2), pp. 189-208.

Li, C. and Li, H. (2008). “Network-constrained regularization and variable selection for analysis of genomic data,” Bioinformatics, 24, pp. 1175–1182.

Liao, X., Xue, Y. and Carin, L., (2005). “Logistic regression with an auxiliary data source,” Proceedings of the 22nd International Conference on Machine Learning, pages 505–512, Bonn, Germany.

Lin D (1998). “An Information-Theoretic Definition of Similarity,” In Proceeding of International Conference on Machine Learning: 296-304.

Lin, C. F. and Wang, S. D. (2002). “Fuzzy support vector machine,” IEEE Transactions on Neural Networks, 13, pp. 464–471.

Liu, H. and Setiono, R.,(1996). “A probabilistic approach to feature selection - a filter solution,” In Proceedings of the Thirteenth International Conference on Machine Learning, pages 319–327.

Liu, S., Liu, H., Latecki, J. L., Yan, S., Xu, C. and Lu, H.,(2011). “Size Adaptive Selection of Most Informative Features,” Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco.

Manjourides J., Pagano M., (2011). “Improving the power of the chronic disease surveillance by incorporating residential history,” Statistical Medicine, vol. 30 pp. 22-23

Martin DMA, Berriman M, Barton GJ (2004). “GOtcha: a new method for predic-tion of protein function assessed by the annotation of seven genomes,” BMC Bioinformatics, 5: 178.

Marx, Z., Rosenstein, M.T., Dietterich,T.G. and Kaelbling, L.K. (2008). “Two algorithms for transfer learning,” In Inductive Transfer: 10 years later.

Minka, T.P. (2003). “A comparison of numerical optimizers for logistic regression,” Technical report.

Mollié A, (1996). “Bayesian mapping of disease’. In Markov chain Monte Carlo in practice,” W. R. Gilks, S. Richardson, D. J. Spiegelhalter eds. 359-379, Chapman and Hall, London, UK.

Mostafavi S, and Morris Q (2010). “Fast Integration of heterogeneous data sources for predicting gene function with limited annotation,” Bioinformatics 2010, 26(14): 1759 - 1765.

Neil, D.B., Moore, A. W., (2004). "Rapid detection of significant spatial clusters," In Proc. 10th ACM SIGKDD conf. on knowledge discovery and data mining, pp. 256-265.

Obozinski, G., Taskar, B. and Jordan. M. (2009). “Joint covariate selection and joint

subspace selection for multiple classification problems," Statistics and Computing, pp 1-22.

P. Drineas, M.W. Mahoney (2005). "On the Nystrm Method for Approximating a Gram Matrix for Improved Kernel-Based Learning," Journal of Machine Learning Research, Volume 6.

Pandey G, Myers CL, Kumar V (2009). "Incorporating functional inter-relationships into protein function prediction algorithms," BMC Bioinformatics 10: 142.

Pavlidis P, Weston J, Cai J, Grundy WN (2001). "Gene functional classification from heterogeneous data," In Proceeding of the fifth annual international con-ference on Computational Molecular Biology: 2001; Montreal, Quebec, Can-ada. ACM Press; 2001: 249-255.

Peng, H., Long, F., and Ding, C., (2005). "Feature selection based on mutual information: criteria of max-dependency max-relevance, and min-redundancy," IEEE Trans. Pattern Anal. Mach. Intell, 27:1226–1238.

Pickles, A., Maughan, B., Wadsworth, M. E. J., (2007). "Epidemiological methods in life course research," New York: Oxford University Press.

R. Kohavi and G. H. John., (1997). "Wrappers for feature subset selection," Artificial Intelligence, 97(1-2): 273–324.

Radivojac, P., Obradovic, Z., Dunker, AK and Vucetic, S. (2004). "Characterization of permutation tests for feature selection," Proc. 15th European Conference on Machine Learning, pp. 334-346, Pisa, Italy.

Raina, R., Ng A.Y. and Koller, D. (2006). "Constructing informative priors using transfer learning," Proc. 23rd International Conference on Machine Learning, pp.713-720, Pittsburgh, PA.

Rakhlin, A. (2007). "Transfer learning toolkit," http://multitask.cs.berkeley.edu.

Ramaswamy, et al., (2001). "Multiclass cancer diagnosis using tumor gene expression signatures," Proceedings of the National Academy of Sciences, 98, pp. 15149–54.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J.(2006). "Kernel-based learning of hierarchical multilabel classification models," Journal of Machine Learning Research, 7:1601–1626.

Schwikowski B, Uetz P, Fields S (2000): "A network of protein–protein interactions in yeast," Nat Biotechnol, 18: 1257-1261.

Shipp MA, Ross KN, Tamayo P, Weng AP, Kutok JL, Aguiar RCT, Gaasenbeek M, Angelo M, Reich M, Pinkus GS, et al (2002). “Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning,” Nature Medicine, 8(1):68-74.

Statnikov, A., Aliferis, C., Tsamardinos, I., Hardin, D., and Levy, S., (2005). “A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis,” Bioinformatics, 21, pp. 631–643.

Tai, F. and Pan, W. (2007). “Incorporating prior knowledge of predictors into penalized classifiers with multiple penalty terms,” Bioinformatics, 23, pp. 1775–82.

Tango, T., Takahashi, K., (2005). "A flexibly shaped spatial scan statistic for detecting clusters," International Journal of Health Geographics, vol. 4, pp. 11.

Tibshirani, R. (1996). “Regression shrinkage and selection via the lasso,” Journal of Royal Statistical Society: Series B, 58, pp. 267–288.

Torkkola K.,(2003). “Feature extraction by non-parametric mutual information maximization.” JMLR, 3: 1415–1438.

Troyanskaya OG, Dolinski K, Owen AB, Altman RB, Botstein D (2003). “A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae),” Proc Natl Aca Sci USA, 100(14): 8348-8353.

Vapnik, V. (1995) “The Nature of Statistical Learning Theory,” New York: Springer.

Weston, J., Elisseef, A., Schlkopf, B. and Tipping, M., (2003). “Use of the zero-norm with linear models and kernel methods,” Journal of Machine Learning Research, 3, pp. 1439–1461.

Williams, C., and Seeger, M., (2000). “The effect of the input density distribution on kernel-based classifiers.” Proceedings of the 17th International Conference on Machine Learning, (pp.1159–1166).

Wu, P. and Dietterich, T. (2004). “Improving SVM accuracy by training on auxiliary data sources,” Proceedings of the 21st International Conference on Machine Learning, pp. 871-878, Morgan Kaufmann.

Zhang, K., Kwok, J.T., (2009) “Prototype vector machine for large scale Semi-supervised Learning,” International Conference on Machine Learning

Zhou X, Kao MC, Wong WH (2002) “Transitive functional annotation by shortest-path analysis of gene expression data” Proc Natl Acad Sci USA, 99: 12783-12788.