1-1-2012

Restricting Supervised Learning: Feature Selection and Feature Space Partition

Xiaofei Nan
University of Mississippi

Restricting Supervised Learning: Feature Selection and Feature Space Partition

Xiaofei Nan

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
Computer Science

University of Mississippi

2012

Abstract

Many supervised learning problems are considered difficult to solve either because of the redundant features or because of the structural complexity of the generative function. Redundant features increase the learning noise and therefore decrease the prediction performance. Additionally, a number of problems in various applications such as bioinformatics or image processing, whose data are sampled in a high dimensional space, suffer the curse of dimensionality, and there are not enough observations to obtain good estimates. Therefore, it is necessary to reduce such features under consideration. Another issue of supervised learning is caused by the complexity of an unknown generative model. To obtain a low variance predictor, linear or other simple functions are normally suggested, but they usually result in high bias. Hence, a possible solution is to partition the feature space into multiple non-overlapping regions such that each region is simple enough to be classified easily. In this dissertation, we proposed several novel techniques for restricting supervised learning problems with respect to either feature selection or feature space partition.

Among different feature selection methods, 1-norm regularization is advocated by many researchers because it incorporates feature selection as part of the learning process. We give special focus here on ranking problems because very little work has been done for ranking using L1 penalty. We present here a 1-norm support vector machine method to simultaneously find a linear ranking function and to perform feature subset selection in ranking problems. Additionally, because ranking is formulated as a classification task when pair-wise data are considered, it increases the computational complexity from linear to quadratic in terms of sample size. We also propose a convex hull reduction method to reduce this impact. The method was tested on one artificial data set and two benchmark real data

sets, concrete compressive strength set and Abalone data set. Theoretically, by tuning the trade-off parameter between the 1-norm penalty and the empirical error, any desired size of feature subset could be achieved, but computing the whole solution path in terms of the trade-off parameter is extremely difficult. Therefore, using 1-norm regularization alone may not end up with a feature subset of small size. We propose a recursive feature selection method based on 1-norm regularization which can handle the multi-class setting effectively and efficiently. The selection is performed iteratively. In each iteration, a linear multi-class classifier is trained using 1-norm regularization, which leads to sparse weight vectors, i.e., many feature weights are exactly zero. Those zero-weight features are eliminated in the next iteration. The selection process has a fast rate of convergence. We tested our method on an earthworm microarray data set and the empirical results demonstrate that the selected features (genes) have very competitive discriminative power.

Feature space partition separates a complex learning problem into multiple non-overlapping simple sub-problems. It is normally implemented in a hierarchical fashion. Different from decision tree, a leaf node of this hierarchical structure does not represent a single decision, but represents a region (sub-problem) that is solvable with respect to linear functions or other simple functions. In our work, we incorporate domain knowledge in the feature space partition process. We consider domain information encoded by discrete or categorical attributes. A discrete or categorical attribute provides a natural partition of the problem domain, and hence divides the original problem into several non-overlapping sub-problems. In this sense, the domain information is useful if the partition simplifies the learning task. However it is not trivial to select the discrete or categorical attribute that maximally simplify the learning task. A naïve approach exhaustively searches all the possible restructured problems. It is computationally prohibitive when the number of discrete or categorical attributes is large. We describe a metric to rank attributes according to their potential to reduce the uncertainty of a classification task. It is quantified as a conditional entropy

achieved using a set of optimal classifiers, each of which is built for a sub-problem defined by the attribute under consideration. To avoid high computational cost, we approximate the solution by the expected minimum conditional entropy with respect to random projections. This approach was tested on three artificial data sets, three cheminformatics data sets, and two leukemia gene expression data sets. Empirical results demonstrate that our method is capable of selecting a proper discrete or categorical attribute to simplify the problem, i.e., the performance of the classifier built for the restructured problem always beats that of the original problem.

Restricting supervised learning is always about building simple learning functions using a limited number of features. Top Selected Pair (TSP) method builds simple classifiers based on very few (for example, two) features with simple arithmetic calculation. However, traditional TSP method only deals with static data. In this dissertation, we propose classification methods for time series data that only depend on a few pairs of features. Based on the different comparison strategies, we developed the following approaches: TSP based on average, TSP based on trend, and TSP based on trend and absolute difference amount. In addition, inspired by the idea of using two features, we propose a time series classification method based on few feature pairs using dynamic time warping and nearest neighbor.

Dedication

I dedicate my dissertation work to my loving parents who have always supported, encouraged and believed in me .

Acknowledgments

I would like to gratefully thank Dr. Yixin Chen for his guidance, comments, patience, and most importantly, his friendship during my study and research at the university of Mississippi. His mentoring was phenomenal in providing a well rounded experience that benefits me building up the the knowledges and skills for my future career. His wisdom, knowledge and commitment to the high standards inspired and motivated me.

I would also like to thank Dr. Dawn Wilkins for her assistance and guidance in my research and career building. For the past three years, I have been offered a lot of good opportunities when I did research in the group leading by Dr. Wilkins and Dr. Chen. I would like to express my thanks to the sparkling academic ideas they provided and collaboration opportunities they suggested. Additionally, I am very grateful for the friendship of all my group members.

I would like to thank the Department of Information and Computer Science at Ole Miss, especially those members of my doctoral committee for their input, valuable discussions and accessibility. Of course, that includes Dr. Xin Dang from Math Department. Thanks very much for all the suggestions and advises they offered. I also owe thanks to Dr. Nan Wang from the university of Southern Mississippi, Dr. Ping Gong from ERDC center and Dr. Robert Doerksen from Department of Medicinal Chemistry of the university of Mississippi for the wonderful collaboration experiences I had in the past year.

Finally, and most importantly, I would like to thank my parents. Their support, encouragement, quiet patience and gratuitous love were the most precious gift I have ever had.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

During the past decades, advances in technology enabled the ability to generate a great amount of data. Unfortunately, as the amount of data increases, the ability to understand and analyze it does not keep up with its growth. Machine learning provides tools to automatically process large quantities of data and achieves good performance in many applications. Supervised learning is the most common type of machine learning task of inferring a function from labelled training data, and has been researched for years. However, the nature of the data, for example high dimensionality with low sample size, or unknown underlying complex generative distribution, makes the learning task difficult. Therefore, we need to restrict such learning problems in a way that the restricted problems could be solved easier than the original problems.

1.1 Supervised Learning

Machine learning is a research area about developing algorithms that learn rules or patterns based on empirical data. Based on whether or not the empirical data has been labelled, it can be categorized as supervised learning, unsupervised learning, or semi-supervised learning. The main task of supervised learning is to infer a function or a model from labelled data. The training data consist of those labelled examples or observations, and each observation includes an input vector consisting multiple features and an output value. The inferred function should capture the characteristics of the unknown underlying data distribution and predict the correct output value for any valid input. Because it is, in general, impossible to

have all possible data or behaviors given (or labelled), the inferred function must generalize from the training data to unseen data.

Supervised learning is used in many areas, for example bioinformatics, cheminformatics where quantitative structure-activity relationship is studied, information retrieval, object recognition in computer vision, spam detection, and speech recognition. According to the differing nature of the output values, supervised learning can be divided into classification, regression and ranking. Classification is the most common task. Its output consists of unordered finite values. For example, spam detector is to predict whether an email is spam or not, and the output values are 1 (spam) and -1 (not spam). The output of regression models is real numbers. For example, a species tissue residue amount could be evaluated based on microarray data, where the tissue residue amount is a real number. Ranking lies between classification and regression – its output has order but no metric structure. For example, a web search engine provides a list of ranked web pages for a query based on relevance, importance and preference.

After many years of development of machine learning, researchers have proposed many learning algorithms. Naïve Bayes [Domingos & Pazzani, 1997] method assumes all features are conditional independent given the class output, and the decision is made by selecting the maximum posteriori based on applying Bayes' theorem. It is robust, fast and scalable to high-dimensional feature spaces and large datasets. However, the strong independence assumption may not be true in practice. Support vector machines [Cortes & Vapnik, 1995] construct one or multiple hyperplanes in a high or even infinite dimensional space for classification, regression or ranking with a tradeoff between a large margin and a small error penalty. Whereas the primal form of support vector machine is stated in a finite dimensional space with linear separation, its dual form provides a mapping into a much higher dimensional space by defining a kernel function to solve nonlinear cases. This approach is used in Chapter 2 with some variations. In contrast to using probabilities or metric-based methods to do prediction, decision tree approaches [Breiman *et al.*, 1984] predict samples based on

a series of questions or decisions. In the tree structures, leaves represents class labels or regression values and branches represent conjunctions of features that lead to those labels or values. Incorporating a stochastic process, random forest [Breiman, 2001] is an ensemble classifier that includes many decision trees, each of which is built on a random selected feature subset. Other approaches include artificial neural network and nearest neighbor classifier.

1.2 Restricting Learning Problem

In a supervised learning problem, a learning model is obtained from a finite number of training samples. The model should catch the characteristics of the seen training data and be able to predict well on unseen samples. For a valid input x, the difference between the predicted output for x and the true output value is called bias in literature. A predictor has high variance for any input x if it predicts differently when trained on different training sets. Generally, a learning model with low bias fits the training data very well. But if the predictor is too flexible and the fit is too good, it normally will lead to high variance. Nevertheless, if the predictor is not flexible, it will cause high bias but probably with low variance. A key issue of many supervised learning algorithms is the ability to adjust the trade-off between bias and variance [Breiman, 1996].

However, in many applications with high dimensionality, the predictor overfits the training set, and therefore results in high variance. Because in a high dimensional space, the samples are distributed sparsely. Learning algorithms based on distance measure could train a model that easily fits the training samples but performs poorly on test samples. That scenario was coined by R. E. Bellman by the term *curse of dimensionality* [Bellman, 1957]. When the dimensionality increases, the volume of the space increases so that the available data become sparse. This sparsity is problematic for methods that require statistical significance. In order to obtain a statistically sound and reliable result, the amount of data to support the result should grow exponentially. For example, Silverman provided

a table illustrating the difficulty of kernel estimation in high dimensions [Silverman, 1986]. The report is provided in Table 1.1.

Table 1.1. The relation between dimensionality and required sample size in kernel estimation

Dimensionality	Required Sample Size
1	4
2	19
5	786
7	10700
10	842000

But, in many applications, high dimensionality is inevitable. For instance, in the cancer prediction problem using microarray data, usually fewer than 100 samples (patients) are available altogether for training and testing, but the number of features (genes) in the raw data ranges from 600 to 60000 [Guyon & Elisseeff, 2003], not mentioning the data generated from the next generation sequencing. Another example is about text processing. Using the bag of words method, a document is represented by a vector of dimension the size of the vocabulary containing word frequency counts. Vocabularies may contain thousands of words. Those applications usually have the nature of "high dimensionality, small sample size", and there are not enough samples for a learning algorithm to train a good predictor.

The solution to restrict a learning problem with high dimension and relatively low sample size is to reduce the dimension size. Dimension reduction is the process of reducing the number of features, and can be divided into feature selection and feature extraction. Feature selection approaches try to find a subset of the original features. This method will be discussed in Chapter 2. Feature selection selects the most discriminative features, and therefore has special usage in biomarker detection, face recognition and other applications. Feature extraction transforms the original input data into a reduced representation set of features. The features in the reduced space are the combinations of all or subsets of the original features. For example, principle component analysis projects the data onto the new basis whose vectors correspond to the largest eigenvectors of the covariance matrix [Pearson,

1901]. This dissertation only focuses on feature selection, not feature extraction.

High dimensionality may result in high variance. High variance also could be observed when the predictor is excessively complex. In that case, overfitting occurs when the model just mimics the behaviors of the training data rather than learning to generalize from trend. Hence, simple models should be used as much as possible. However, sometimes a complex predictor is hard to avoid if the underlying probability distribution is complex. One solution is to partition the feature space into multiple non-overlapping sub-spaces so that each sub-region is simple enough to be handled by a simple model. For example, as shown in Figure 1.1, in a 2 dimensional space, a binary classification task is to separate the dots from rectangles. The true separation function is a trigonometric function represented by a smoothing line in the Figure. But if we partition the 2-D space into four regions based on the three vertical dash lines, each sub-problem generated by the corresponding region is approximately linearly separable. This partition enables the use of simple learning algorithms on hard problems.

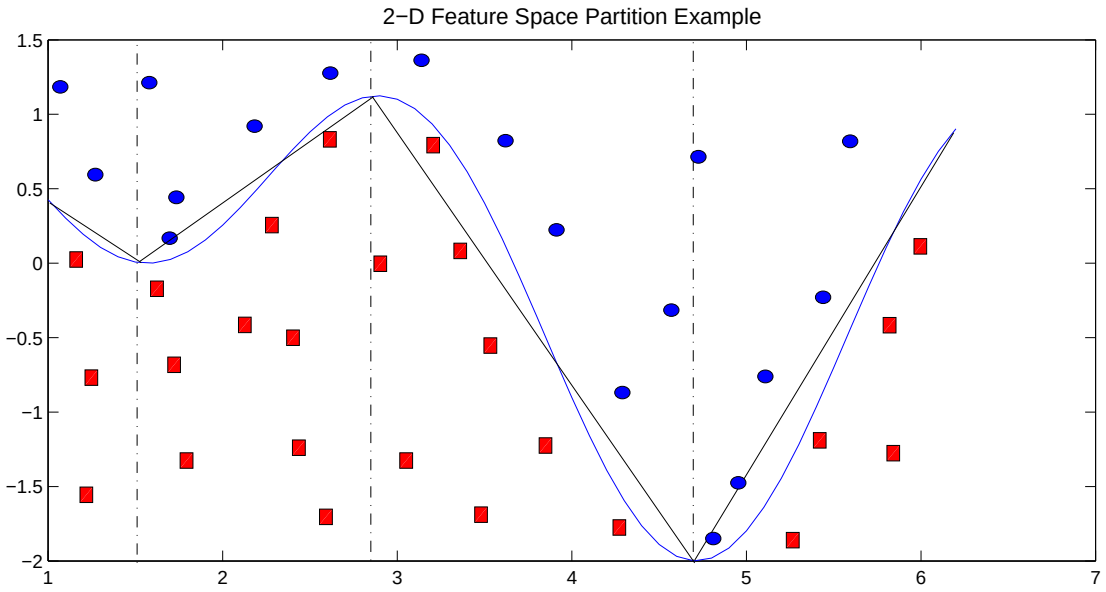

Figure 1.1. A 2-D feature space partition example

1.3 Outline of the Dissertation

The goal of this work is to restrict a supervised learning problem either by feature selection or by feature space partition. This dissertation is organized as follows.

We follow the two main branches to discuss feature selection in Chapter 2 and feature space partition in Chapter 3. The feature selection topic has been researched for years, and it can be categorized into three categories, filter, wrapper and embedded method. The concepts and representative works are described in Section 2.1. Among various feature selection methods, we mainly discuss the approach using 1-norm regularization to embed the feature selection process into the general learning framework, especially when performing classification and regression in Section 2.2. Special focus is given to the ranking problem, because it has been researched the least. We propose a method of ranking with 1-norm regularization using convex hull reduction in Section 2.2.5. A single 1-norm regularization may not satisfy the needs when a small feature subset is required. Therefore, we propose a recursive feature selection framework based on 1-norm regularization in Section 2.3.

Feature space partition is to restructure a complex problem into multiple easy sub-problems. Section 3.1 introduces the motivation and background. Although not much work has been done related with this topic, there are two methods that relate the concept of simplifying a learning problem to feature space partition. One technique is to hierarchically divide a region into two parts by a linear hyperplane using mutual information evaluation [Padmanabhan *et al.*, 1999]. The other is to repeatedly partition the space into hyperparallelepipeds whose faces are perpendicular to feature axes until some stopping criterion is met [Kohn *et al.*, 1996; Singh & Galton, 2003; Kishore *et al.*, 2001; He *et al.*, 2005]. We review these methods in Section 3.2 and Section 3.3. We propose a method of leveraging domain information to restructure prediction problems in Section 3.4.

The work of classification using top scoring feature pairs is listed outside the two branches mentioned above, although strictly speaking, it belongs to feature selection. But it does not match the theme in Chapter 2 where a linear model using 1-norm penalty is advocated. It

will be introduced in Chapter 4. The Top Scoring Pair method on static data is introduced in Sectioin 4.1. And we propose four extensions to the standard TSP to deal with time series data in Section 4.2.

CHAPTER 2

FEATURE SELECTION

In machine learning, feature selection is a technique of seeking the most representative subset of features. In many applications, a data set contains thousands or even more features, though the majority of which may be irrelevant or redundant. Those irrelevant or redundant features often behave like noise in the training process, confusing the learning algorithm and degrading the learning performance. The removal of such features would simplify the learning problem. Feature selection also helps identifying the most discriminant features which normally are meaningful for real applications. For example, when applied to gene expression array analysis, feature selection detects the influential genes by which biological researchers could discriminate normal instances from abnormal ones, and therefore, facilitates further biological research or judgments.

The benefits of feature selections are generalized as Guyon & Elisseeff [2003]:

- Improving prediction performance
- Reducing training time
- Facilitating data visualization and data understanding
- Reducing the measurement and storage requirements

2.1 Filter, Wrapper, and Embedded Method

We only focus on supervised learning in this dissertation. Unsupervised and semi-supervised learning could be found in other literatures [Tseng & Kao, 2005; Liu *et al.*,

2006; Salem *et al.*, 2008]. Feature selection algorithms roughly fall into three categories, wrappers, filters, and embedded methods.

The idea of filter is to implement a fast preprocessing step or auxiliary selection mechanism to reduce the feature set. Filters are normally independent of the choice of the predictor, and has the characters of simplicity and scalability. At first, filter specifically refers to variable ranking which ranks *individual features* according to a metric, and eliminates features that do not exceed a given threshold. Ranking metric could be the correlations between a particular feature and the output vector. Weston et al. [Weston *et al.*, 2003] stated the issue of using correlation criteria for microarray data analysis. The feature ranking also rely on empirical estimates of the mutual information between a single feature and target variabl [Bekkerman *et al.*, 2002]. Another ranking metric is based on feature's individual predictive power, using as a criterion the performance of a classifier built with a single feature. The feature ranking is computational efficient because it only computes feature scores. Nevertheless, this method only focuses on the predictive power of individual features. It is prone to the selection of redundant features. The concept of filter has been extended to feature subset selection, for example, using mutual information between the feature subset and the target variable. Bi et al. [Bi *et al.*, 2003] used a wrapper with linear predictor as a filter and then train a more complex non-linear predictor on the resulting feature subset.

Feature subset selection can be formulated as a search through the space consisting of all possible feature subsets. Given n features, an exhaustive search would require the evaluation of $2^n - 1$ subsets, which is infeasible when n is large. Hence, heuristic searching strategies like A* algorithm or depth first branch and bound may be applied. During the optimal feature subset searching process, some basic properties need to be considered [Blum & Langley, 1997]:

- A starting point in the search space
- An organization of the search

- An evaluation strategy of the selected subset
- A stopping criterion for halting the search

Wrapper methods [Kohavi & John, 1997] search through feature subset space. Each subset is applied to a certain machine learning model and assessed by the learning performance. In these methods, learning models act as black boxes. Sequential algorithms [Kittler, 1978] either start with the full or empty feature set, and proceed by greedily adding or removing features. Sequential backward elimination stars with full set of n features, and considers each of the n subsets of n-1 features by removing each feature once. Form these n subsets, the one giving the highest performance is chosen. The process is then repeated for the set of n-1 remaining features until some termination criterion is fulfilled. Similarly, the sequential forward selection is defined, where the initial state is the empty feature set, and the features are added using a greedy heuristic.

Embedded approaches [Lal *et al.*, 2006] implement feature selection in the process of learning. While wrapper methods search the space of all feature subsets, the searching step in embedded methods is guided by the learning algorithm. This guidance could be obtained from estimating changes in the objective function by adding or removing features. For example, Guyon et al. [Guyon *et al.*, 2002] proposed Support Vector Machine Recursive Feature Elimination (SVM-RFE) algorithm to recursively classify the instances by the SVM classifier and eliminate the feature(s) with the least weight(s). The number of features to be eliminated in each iteration is ad hoc. Moreover, there is no firm conclusion about when to terminate the recursive steps.

Most feature selection in the literature is designed for binary problems. When extended to real-life multiclass tasks, combining several binary classifiers are typically suggested, such as one-versus-all and one-versus-one [Statnikov *et al.*, 2006]. For situations with k classes, one-versus-all approach constructs k binary classifiers, each of which is trained with all the

instances in a certain class as positive and all other examples as negative. It is computationally expensive and has highly imbalanced data for each binary classifier. On the other hand, one-versus-one method constructs $k(k-1)/2$ binary classifiers for all pairs of classes. An instance is predicted for the class with the majority vote. Similar to the one-versus-all approach, the one-versus-one approach has heavy computational burden. Platt et al. [Platt *et al.*, 2000] proposed a directed acyclic graph SVM (DAGSVM) algorithm whose training phase is the same as one-versus-one by solving $k(k-1)/2$ binary problems. However, DAGSVM uses a rooted acyclic graph to make a decision from $k(k-1)/2$ prediction results. Some researchers proposed methods solving multiclass tasks in one step: build a piecewise separation of the k classes in a single optimization. This idea is comparable to the one-versus-all approach. It constructs k classifiers, each of which separates a class from the remaining classes, but all classifiers are obtained by solving one optimization problem. Weston and Watkins [Weston & Watkins, 1999] proposed a formulation of the SVM that enables a multiclass problem. But, solving multiclass problem in one step results in a much larger scale optimization problem. Crammer and Singer [Crammer & Singer, 2001] decomposed the dual problem into multiple optimization problems of reduced size and solved them by a fixed-point algorithm. A comparison of different methods for multiclass SVM was done by Hsu and Lin [Hsu & Lin, 2002].

2.2 1-Norm Regularization Facilitating Feature Selection

A cost function typically comprises two parts, empirical error and a measure of model complexity. The model complexity is usually approximated by a regularizer. 2-norm regularizer, ridge penalty in most literatures, is commonly used, such as in traditional support vector machine. Recently, many researchers proposed replacing 2-norm with 1-norm (i.e. lasso penalty). The lasso penalty was first proposed in [Tibshirani, 1996] for regression problems. Similar to the ridge penalty, the lasso penalty shrinks the feature weights towards zeros, and therefore also benefits from the reduction in weights' variance. But, different from

ridge penalty, when making the trade-off parameter between empirical error and regularizer sufficiently large, lasso penalty will cause some of the feature weights to be **exactly zero**. Zhu et al. [Zhu *et al.*, 2003] argued that the 1-norm regularization yields sparse results, i.e. only a small number of feature weights are nonzero, hence facilitating adaptive feature selection.

The fact that 1-norm favors sparse results could be explained by Figure 2.1 in a 2-D space. Feasible region consists of all the possible solutions (for example, feature weights) that satisfy the constraints in the learning optimization formula. In the feature space, we draw "equipotential contours" on which the data points share the same penalty values. For 2-norm, those contours are circles (hypersphere in a high dimensional space), whereas, 1-norm has diamond shape (hypercubes in a high dimensional space). The penalty value decreases as the the circle or diamond shrinks. Since the objective function of the optimization formula is to minimize the penalty term, the optimal solution is achieved as the intersection point between the feasible region and the equipotential line with least value. Intuitively, it is with greater possibility for a diamond to have that intersection point locating at one axis than for a circle. Therefore, the optimal solution of a learning problem using 1-norm penalty could easily get lots of zeros.

The use of 1-norm was advocated in many applications, such as multi-instance learning [Chen *et al.*, 2006], ranking [Nan *et al.*, 2010a] and boosting [Duchi & Singer, 2009], because of its sparsity-favoring property. Several researchers discussed the multiclass problem based on 1-norm regularization and various loss functions for the empirical error. For example, Friedman et al. [Friedman *et al.*, 2010] introduced 1-norm into multinomial logistic regression which is capable of handling multiclass classification problems. Bi et al. [Bi *et al.*, 2003] chose ϵ-insensitive loss function. Liu and Shen [Liu & Shen, 2006] defined a specific loss function ψ-loss that replaces the convex SVM loss function by a nonconvex function. Other works mainly used hinge loss with different variations [Chapelle & Keerthi, 2008; Szedmak *et al.*, 2004; Wang & Shen, 2006]. In this dissertation, the hinge loss function we

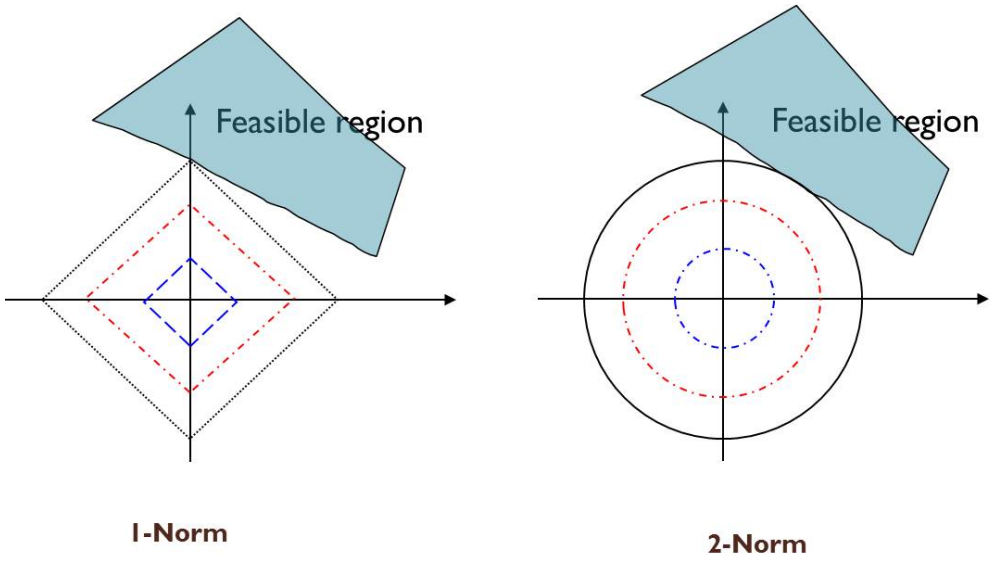

Figure 2.1. Results sparsity comparison between 1-norm and 2-norm

apply is similar to that in [Weston & Watkins, 1999], but has not been used in any 1-norm multiclass work.

The popularity of standard 2-norm support vector machine is based on the concept of kernel, which may create non-linear classifiers by implicitly transforming the original input space to high dimensional feature space. The kernel trick does not apply to 1-norm SVM due to the fact that 1-norm can not be represented by a dot product.

2.2.1 Binary Classification with 1-Norm Regularization

In the cases of binary classification, a support vector machine is formulated as the solution of a quadratic program. Given a set of training data $(x_1,y_1),\ldots,(x_l,y_l)$, where the input $x_i \in \mathbb{R}^n$, and the output $y_i \in \{1,-1\}$ has binary values, the standard SVM with linear kernel is found by solving the following quadratic program:

$$\begin{aligned} \min_{w,b,\xi} \quad & \lambda\|w\|_2^2 + \textstyle\sum_{i=1}^{\ell} \xi_i \\ s.t. \quad & y_i(w^T x_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, i = 1,\ldots,\ell \end{aligned} \tag{2.1}$$

where $\lambda > 0$ is a fixed penalty parameter that specifies the trade-off between empirical misclassification error and the complexity of the classifier. Zhu et al. [Zhu *et al.*, 2003] argued that the 1-norm SVM may have some advantage over the standard 2-norm SVM, especially when there are redundant noise features. Similarly, 1-norm SVM can be formulated by replacing the $\|\cdot\|_2$ regularization operator in (2.1) with $\|\cdot\|_1$ regularization. Hence we have the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \lambda\|w\|_1 + \textstyle\sum_{i=1}^{\ell} \xi_i \\ s.t. \quad & y_i(w^T x_i + b)_T + \xi_i \geq 1 \\ & \xi_i \geq 0, i = 1, \ldots, \ell \end{aligned} \tag{2.2}$$

where $\|w\|_1 = \sum_{j=1}^{n} \|w_i\|$.

The optimization problem (2.2) can be formed as a linear program. We rewrite $w_j = u_j - v_j$ where $u_j, v_j \geq 0$. If either u_j or v_j has to equal zero, then $\mid w_j \mid = u_j + v_j$. Then (2.2) becomes

$$\begin{aligned} \min_{u,v,b,\xi} \quad & \lambda \textstyle\sum_{j=1}^{n}(u_j + v_j) + \sum_{i=1}^{\ell} \xi_i \\ s.t. \quad & y_i[(u - v)^T x_i + b] + \xi_i \geq 1 \\ & u_j, v_j \geq 0, j = 1, \ldots, n \\ & \xi_i \geq 0, i = 1, \ldots, \ell. \end{aligned} \tag{2.3}$$

Solving (2.3) yields solutions equivalent to those obtained by (2.2) because any optimal solution to (2.3) has at least one of the two variables u_j, v_j equal to 0 for all $j = 1, \ldots, n$. Otherwise, assume $u_j > v_j > 0$ without loss of generality, and we can find a better solution by setting $u_j = u_j - v_j$ and $v_j = 0$, which contradicts to the optimality of (u, v).

2.2.2 Multi-class Classification with 1-Norm Regularization

In a k-category classification task, the class label y is coded as $\{1, ..., k\}$.

Given k linear decision functions $f_1, ..., f_k$ where f_c corresponds to class c, each decision function is defined as $f_c(x) = w_c^T x + b_c,\ c = 1, ..., k$, where, the parameters $w_c =$

$[w_{c,1}, ..., w_{c,n}]^T \in \mathbb{R}^n$ and $b_c \in \mathbb{R}$. We consider a winner-takes-all classification rule specified as

$$\Phi(x) = \arg\max_c f_c(x) \ ,$$

which assigns input x to class $\Phi(x)$ with the highest decision value. If the instances are separable, there exist decision functions that satisfy

$$f_{y_i}(x_i) \geq f_c(x_i), c \neq y_i \ .$$

The above inequality is equivalent to

$$(w_{y_i} - w_c)x_i + (b_{y_i} - b_c) \geq 1, c \neq y_i \ .$$

To handle the nonseparable cases, we introduce slack variables into the model, i.e.,

$$(w_{y_i} - w_c)x_i + (b_{y_i} - b_c) \geq 1 - \xi_{y_i,c}, c \neq y_i \ , \tag{2.4}$$

where $\xi_{y_i,c} \geq 0$ is a slack variable.

The learning of the classifier can be formulated as an optimization problem. Our goal is to seek f that minimizes the sum of empirical error and model complexity. Empirical error can be computed as the proportion of nonseparable instances, i.e. errors on training data, or approximated using various loss functions, such as hinge loss, logistic loss. These loss functions were originally defined for binary classes. Extending to multiclass case, there are different variations. For example, Wang & Shen [2006] utilized hinge loss function $\sum_{c \neq y_i} [f_c(x_i) + 1]_+$, where $(\cdot)_+ \equiv \max(\cdot, 0)$. In this paper, we apply the hinge loss function

$$\sum_{c \neq y_i} [1 - (f_{y_i}(x_i) - f_c(x_i))]_+ \ ,$$

which was introduced by Weston and Watkins in [Weston & Watkins, 1999]. This hinge loss function has not been used in the multiclass setting with 1-norm penalty before. And it is equivalent to the slack variable defined in (2.4).

Model complexity is approximated using 1-norm. For the purpose of feature selection, we propose 1-norm multiclass regularization, L1MR, as:

$$\begin{aligned} \min_{w_c, b_c, \xi_{y_i,c}} \quad & \lambda \sum_{c=1}^{k} \|w_c\|_1 + \sum_{i=1}^{l} \sum_{j=1, j\neq y_i}^{k} \xi_{y_i,c} \\ s.t. \quad & (w_{y_i} - w_c)x_i + (b_{y_i} - b_c) \geq 1 - \xi_{y_i,c} \\ & \xi_{y_i,c} \geq 0, \\ & i = 1, \ldots, l, c = 1, \ldots, k, c \neq y_i. \end{aligned} \tag{2.5}$$

where $\sum_{c=1}^{k} \|w_c\|_1 = \sum_{c=1}^{k} \sum_{j=1}^{n} \|w_{c,j}\|$ is a 1-norm penalty, $\sum_{i=1}^{l} \sum_{j=1, j\neq y_i}^{k} \xi_{y_i,c}$ can be viewed as an upper bound on the training error; λ is a cost parameter that controls the training accuracy and sparsity of the solution.

The optimization problem (2.5) can be formed as a linear program by rewriting $w_{c,j} = u_{c,j} - v_{c,j}$ where $u_{c,j}, v_{c,j} \geq 0$. If either $u_{c,j}$ or $v_{c,j}$ has to equal to zero, then $\|w_{c,j}\| = u_{c,j} + v_{c,j}$. Therefore, (2.5) is equivalent to

$$\begin{aligned} \min_{u_c, v_c, b_c, \xi_{y_i,c}} \quad & \lambda \sum_{c=1}^{k} \sum_{j=1}^{n} (u_{c,j} + v_{c,j}) + \sum_{i=1}^{l} \sum_{j=1, j\neq y_i}^{k} \xi_{y_i,c} \\ s.t. \quad & (u_{y_i} - u_c - v_{y_i} + v_c)x_i + (b_{y_i} - b_c) \geq 1 - \xi_{y_i,c} \\ & \xi_{y_i,c} \geq 0, u_{c,j} \geq 0, v_{c,j} \geq 0 \\ & i = 1, \ldots, l, c = 1, \ldots, k, c \neq y_i. \end{aligned} \tag{2.6}$$

The number of slack variables is $(k-1)l$ in L1MR. Motivated by [Crammer & Singer,

2001], we shrink the number of slack variables to l by introducing a new slack variable as

$$\xi_i = (\max_c f_c(x_i) + 1 - f_{y_i}(x_i))_+.$$

Compared with the slack variables in (2.6), which capture gaps between each pair of decision functions, the new slack variable considers the maximum gap. A simplified version of L1MR, SL1MR, is then formulated as,

$$\begin{aligned} \min_{u_c, v_c, b_c, \xi_i} \ & \lambda \sum_{c=1}^{k} \sum_{j=1}^{n} (u_{c,j} + v_{c,j}) + \sum_{i=1}^{l} \xi_i \\ s.t. \ & (u_{y_i} - u_c - v_{y_i} + v_c)x_i + (b_{y_i} - b_c) \geq 1 - \xi_i \\ & \xi_i \geq 0, u_{c,j} \geq 0, v_{c,j} \geq 0 \\ & i = 1, \ldots, l, c = 1, \ldots, k, c \neq y_i. \end{aligned} \tag{2.7}$$

2.2.3 Support Vector Regression with 1-Norm Regularization

Support vector machine was largely developed by Vapnik and co-works at AT&T Bell Laboratories. Also, in regression and time series prediction areas, excellent performances were soon obtained [Müller *et al.*, 1997; Drucker *et al.*, 1997; Stitson *et al.*, 1999; Mattera & Haykin, 1999].

In ϵ-regression [Vapnik, 1995], the goal is to find a function $f(x)$ that has at most ϵ deviation from the true targets for all the training data, and at the same time is as flat as possible. In other words, errors less than ϵ are ignored, but any deviation larger than that threshold can not be accepted. Still, linear discriminant function $f(x) = w^T x + b$ is considered. Depending on the value of ϵ, a solution may not exist. To avoid this, slack variables ξ_i, ξ_i^+ are introduced to cope with otherwise infeasible constrains. Hence, we arrive at the formula stated in [Vapnik, 1995].

$$\begin{aligned} \min_{w,b,\xi,\xi^+} \quad & \lambda\|w\|_2^2 + \sum_{i=1}^{\ell}(\xi_i + \xi_i^+) \\ s.t. \quad & y_i - (w^T x_i + b) \leq \epsilon + \xi_i \\ & (w^T x_i + b) - y_i \leq \epsilon + \xi_i^+ \\ & \xi_i, \xi_i^+ \geq 0, i = 1, \ldots, \ell \end{aligned} \tag{2.8}$$

This corresponds to dealing with a so called ϵ-insensitive loss function $|\xi|_\epsilon$ described by

$$|\xi|_\epsilon = \begin{cases} 0 & \text{if } |\xi| \leq \epsilon; \\ |\xi| - \epsilon & \text{otherwise} \end{cases} \tag{2.9}$$

Replacing the 2-norm penalty with 1-norm penalty, formula (2.8) becomes

$$\begin{aligned} \min_{w,b,\xi,\xi^+} \quad & \lambda\|w\|_1 + \sum_{i=1}^{\ell}(\xi_i + \xi_i^+) \\ s.t. \quad & y_i - (w^T x_i + b) \leq \epsilon + \xi_i \\ & (w^T x_i + b) - y_i \leq \epsilon + \xi_i^+ \\ & \xi_i, \xi_i^+ \geq 0, i = 1, \ldots, \ell \end{aligned} \tag{2.10}$$

Using the same strategies as in 1-norm classification by letting $w_j = u_j - v_j$ where $u_j, v_j \geq 0$, the optimization problem (2.10) can be formed as a linear program as well.

$$\begin{aligned} \min_{u,v,b,\xi,\xi^+} \quad & \lambda \sum_{j=1}^{n}(u_j + v_j) + \sum_{i=1}^{\ell}(\xi_i + \xi_i^+) \\ s.t. \quad & y_i - (u - v)^T x_i - b \leq \epsilon + \xi_i \\ & (u - v)^T x_i + b - y_i \leq \epsilon + \xi_i^+ \\ & u_j, v_j \geq 0, j = 1, \ldots, n \\ & \xi_i, \xi_i^+ \geq 0, i = 1, \ldots, \ell. \end{aligned} \tag{2.11}$$

2.2.4 Lasso Regression

Lasso regression [Tibshirani, 1996] is based on the idea of linear least square regression where the model coefficients are chosen to minimize the residual sum of squares. Lasso regression is defined by the following optimization problem.

$$\begin{aligned} \min_{w,b} \quad & \sum_{i=1}^{\ell}(y_i - b - \sum_{j=1}^{n} w_j x_{ij})^2 \\ s.t. \quad & \sum_{j=1}^{d} |w_j| \leq s \\ & s > 0 \end{aligned} \tag{2.12}$$

In (2.12), the constraints are equivalent to minimizing 1-norm penalty of the feature weights, and hence have the interesting and desirable effect of setting weight coefficients to zero.

The difference between 1-norm support vector regression and lasso regression is the use of different loss function, or in other words, how to measure the training error. Support vector regression uses ϵ-insensitive loss function $|\xi|_\epsilon$ and is described in (2.9). Lasso regression uses square loss, which is defined as

$$l(y, f(x)) = (y - f(x))^2$$

For comparison purposes, we would like to mention hinge loss here as well. It is mainly used in classification problems. We have described the multiclass hinge loss function is Section 2.2.2, but haven't explicitly give the definition for binary classification. For a binary classification task, the hinge loss is defined as

$$l(y, f(x)) = max(0, 1 - yf(x))$$

Figure 2.2 gives an intuitive comparisons among hinge loss, square loss and ϵ-insensitive loss.

Besides the loss functions discussed above, other popular loss functions include absolute loss and logistic loss.

2.2.5 Ranking with 1-Norm Regularization Using Convex Hull Reduction

We give a special focus in this thesis for the ranking problems. Because ranking has not been researched as much as classification and regression, though it has various real

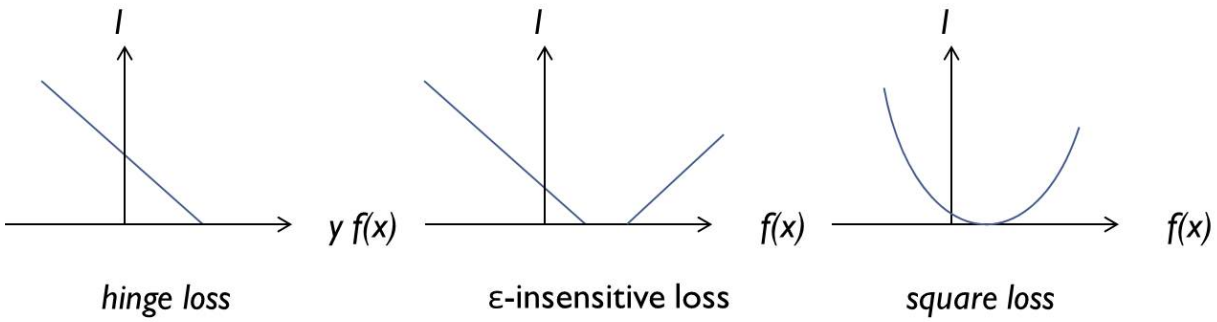

Figure 2.2. Loss function comparison

applications.

Most inductive learning work has concentrated on classification and regression. However, there are many applications that reside in between the two: it is desirable to order objects or determine preferences rather than to classify instances (classification) or to predict ordinal utility values (regression). Ranking problems arise from such applications. For instance, it is common to rank document search results according to their relevance to a query as "perfect match", "relevant" and "non-relevant". A movie recommendation system gives users a list of recommended films where the most potentially wanted films are placed on the top.

One advantage of formulating an application as a ranking problem is that preference judgments may be much easier to obtain than the labels required for classification learning and the values of the dependent variable in regression [Cohen *et al.*, 1999]. For example, in the design of the above movie recommendation system, a user might easily express his or her preference of movie A over movie B to generate a training data set. But it is harder to quantify how much he or she likes these two movies. Therefore, learning to rank is a natural choice for many applications in social science, mathematical economics, and information retrieval where human preferences play an important role.

As described in Section 1.1, ranking does not have a complete metric structure, in particular, the concept of distance is not available in outputs. It hence brings challenges in defining a ranking loss function. A ranking loss function should capture the intuition that

it is more expensive to make mistakes such as placing "perfect match" to "non-relevant" than to "relevant", yet it is difficult to define the distance of any two labels. One approach to tackle this dilemma is to cast a ranking problem as a regression problem that imposes a metric on top of the set of rankings. Balcan et al. [Balcan *et al.*, 2008] proposed and analyzed reductions from ranking to binary classification problems given the significant efforts placed on developing classification algorithms. A commonly used method is to act on pairs of observations, and the loss function measures the probability of misclassification of a randomly drawn pair (x_1, x_2), where the two classes are x_1 being preferred over x_2 and the opposite respectively [Herbrich *et al.*, 2000]. However a limitation of this method is that it increases the computational complexity from linear to quadratic in terms of the number of samples. In this section, we incorporate the idea of convex hull reduction to decrease the computational complexity of ranking. Convex hull decreases the number of constraints that are inherent with pairwise data without compromising the ranking relationships. Its basic idea is to select a subset of representative preference constraints instead of using all preference pairs.

We proposed a ranking algorithm on pair-wise preference constraints. A convex hull reduction algorithm is used to reduce the number of constraints. A linear ranking function is computed by minimizing the sum of ranking error and a 1-norm regularization term. The 1-norm regularization favors sparse solutions, hence selects a subset of features. Experimental results demonstrate good performance on the data sets tested. This work has been published in [Nan *et al.*, 2010a].

2.2.6 Ranking Problem Formulation

The ranking problem is formulated as below. Given observation set $\mathbb{X} \in \mathbb{R}^n$, let $\mathcal{R}$ be a relation on $\mathbb{X} \times \mathbb{X}$ that for any $(x_i, x_j) \in \mathcal{R}$, x_i is ranked higher than x_j. We are interested

in learning a ranking function f to capture this relation, i.e.,

$$f(x_i) > f(x_j), \forall (x_i, x_j) \in \mathcal{R}.$$

If we consider the family of linear ranking functions, $f(x) = w^T x + b$, the given relation $\mathcal{R}$ is linear rankable (similar to linear separable in binary classification) if and only if

$$w^T x_i + b > w^T x_j + b, \forall (x_i, x_j) \in \mathcal{R},$$

which is equivalent to

$$w^T (x_i - x_j) > 0, \forall (x_i, x_j) \in \mathcal{R}.$$

If $\mathcal{R}$ is linear rankable, there exists an n-vector w such that

$$w^T (x_i - x_j) \geq 1, \forall (x_i, x_j) \in \mathcal{R}.$$

For the non-rankable cases, slack variables could be introduced into the model, i.e.,

$$w^T (x_i - x_j) \geq 1 - \xi_{i,j}, \eta_{i,j} \geq 0, \forall (x_i, x_j) \in \mathcal{R}. \tag{2.13}$$

Therefore, the ranking problem has been cast as a classification problem where the loss function acts on pairs of preferences. In order to have some indication that the learning algorithm will generalize well, the loss function should involve both the complexity of the ranking function class and the empirical error. This complexity can be achieved by an informative quantity, for example, VC dimension or an upper bound on VC dimension.

However, it is clear that the number of constraints grows quickly as the size of observation increases, roughly quadratic in the number of training samples, which poses a significant computational burden and even makes the solution infeasible.

2.2.7 Convex Hull Reduction

When there are multiple ranking labels $\{y_1, y_2, \ldots, y_k\}$, let Y_s be the set of all observations with rank label y_s. A preference relation is specified as

$$\mathcal{R} = \{(x_i, x_j) : \forall i, j,\ x_i \in Y_s, x_j \in Y_t, Y_s \prec Y_t\}\ ,$$

where $\prec$ denotes "being preferred to".

The size of $\mathcal{R}$ is $|\mathcal{R}| = \sum_{s,t,Y_s \prec Y_t} |Y_s||Y_t|$. For a ranking problem in Section 2.2.6, there will be $|\mathcal{R}|$ constraints of form (2.13). For almost all ranking problems, $|\mathcal{R}|$ is much larger than the number of ranking labels. In fact, the size of constraints is approximately the quadratic of the observation size, i.e., the number of constraints is $\mathcal{O}(\ell^2)$ where ℓ is the number of training observations. Therefore, the number of constraints is prohibitively large even for a training set of medium size, say 1000.

Our approach to reduce the number of constraints is to select a small subset of representative preference constraints without compromising the ranking performance significantly. These representative constraints are constructed using observations lying on the convex hulls of Y_s $(s = 1, \ldots, k)$.

A convex hull is the smallest convex polygon containing all the points in a set. It captures the shape of a data set. The convex hull computation could be finished in $\mathcal{O}(|Y|lg(|Y|))$ time for each observations set Y. After finding convex hulls for each set $Y_s (s = 1, \ldots, k)$, constraints in (2.13) are reduced to

$$w^T(x_i - x_j) \geq 1 - f_{i,j}, \xi_{i,j} \geq 0, \forall (x_i, x_j)$$
$$\text{that } x_i \in CH(Y_s), x_j \in CH(Y_t), \text{and } Y_s \prec Y_t,$$

where $CH(Y_s)$ denotes the observations on the convex hull of the set Y_s.

After convex hull reduction, the number of constraints shrinks to

$$\mathcal{O}(\sum_{s,t,Y_s \prec Y_t} |CH(Y_s)||CH(Y_t)|),$$

where $|CH(Y_s)|$ is the size of the convex hull of set Y_s. We will show later that in most applications this procedure will reduce the constraint size dramatically.

2.2.8 Ranking with 1-Norm Regularization and Convex Hull Reduction

Combing the results of (2.14) and (2.3), the 1-norm ranking problems could be regularized as

$$\begin{aligned} \min_{u,v,b,\xi} \quad & \lambda \sum_{j=1}^{n}(u_j + v_j) + \sum_{i=1}^{\ell} \xi_{i,j} \\ s.t. \quad & w^T(x_i - x_j) \geq 1 - \xi_{i,j}, \xi_{i,j} \geq 0 \\ & x_i \in CH(Y_s), x_j \in CH(Y_t), Y_s \prec Y_t, \\ & u_j, v_j \geq 0, j = 1, \ldots, n \\ & \eta_{i,j} \geq 0, i = 1, \ldots, \ell. \end{aligned}$$

Slack variables are necessary in the non-rankable applications. But the above formula shows that the size of slack variables is equal to the number of convex hull point pairs, which is another computational burden, especially when the sample points are mostly distributed on the surface of the high dimensional cube and convex hull reduction cannot relieve the computational load a lot.

To relieve the computational load, we not only reduce the number of constraints but also reduce the number of variables in the optimization problem. To fulfill the second purpose, we assign each convex hull point a slack variable. Therefore, the original slack variable for each pair (x_i, x_j) can be represented by the addition of the slack variables of x_i and x_j. And the number of slack variables is dramatically reduced from $|\mathcal{R}| = \sum_{s,t,Y_s \prec Y_t} |Y_s||Y_t|$ to $\sum_s |Y_s|$.

As a result, the ranking function is specified as the solution of the following linear

program:

$$\begin{aligned} \min_{u,v,b,\eta} \quad & \lambda \sum_{j=1}^{n}(u_j + v_j) + \sum_{s,t}(\eta_s + \eta_t) \\ s.t. \quad & (u-v)^T(x_s - x_t) + (\eta_s + \eta_t) \geq 1 \\ & x_s \in CH(Y_s), x_t \in CH(Y_t), Y_s \prec Y_t \\ & u_j, v_j \geq 0, j = 1, \ldots, n \\ & \eta_s, \eta_t \geq 0, s, t = 1, \ldots, n. \end{aligned} \tag{2.14}$$

Moreover, we could further reduce the number of constraints using the transitive property of the preference over ranking labels. For example, we have preference relation denoting as $Y_s \prec Y_t \prec Y_u$. Instead of considering all the preference constraints in the relation $\mathcal{R}_1 = \{(Y_s, Y_t), (Y_s, Y_u), (Y_t, Y_u)\}$, relations $\mathcal{R}_2 = \{(Y_s, Y_t), (Y_t, Y_u)\}$ is sufficient to express $\mathcal{R}_1$ in the sense that the transitive closure of $\mathcal{R}_2$ is $\mathcal{R}_1$. Therefore, the number of constraints after reduction is bounded by $\mathcal{O}(\sum_{s, Y_s \prec Y_{s+1}} |CH(Y_s)||CH(Y_{s+1})|)$.

2.2.9 Kendall τ Correlation Coefficient

Performance metrics are fundamental in assessing any ranking method. Fung et al. [Fung *et al.*, 2006] used generalized Wilcoxon-Mann-Whitney Statistics to measure the probability of any pair of data being ordered correctly. A similar but simpler concept is Kendall τ rank correlation coefficient (simply the Kendall τ), which is a non-parametric statistic used to measure the degree of correspondence between two rankings. It assesses the significance of this correspondence. Considering the occurrence of ties, we use Kendall τ-b variation.

Let a_i and b_i be the rank of observation x_i given by two ranking algorithms, Kendall τ-b rank correlation coefficient for the two ranking algorithms on the given set of observations is defined as

$$\tau = \frac{\sum_{i<j} sgn(a_i - a_j) sgn(b_i - b_j)}{\sqrt{(T_0 - T_1)(T_0 - T_2)}}$$

where $T_0 = n(n-1)/2$, $T_1 = \sum_k t_k(t_k - 1)/2$, and $T_2 = \sum_l u_l(u_l - 1)/2$. The t_k is the number of tied a values in the k-th group, u_l is the number of tied b values in the l-th group, n is the number of observations.

If the two rankings are identical, the Kendall τ-b value is equal to 1. If two rankings are totally opposite, Kendall τ-b equals negative one. It approaches zero when the two rankings are irrelevant.

2.2.10 Experiments and Results

We tested our approach on three data sets: an artificial data set, a concrete compressive strength data set, and an Abalone data set. The artificial data set consists of 2000 samples in two size-balanced groups with dimension 6. Three features out of six are generated according to the uniform distribution on two regions with little overlapping so that the two groups are linear separable. The other three features are random noises with little discriminative power.

The two real data sets are publicly available from UCI machine learning repository [1]. They were used as benchmark data sets for ordinal regression methods in the literature. The dependent variable of the concrete strength data set is continuous. In the Abalone data set, the dependent variable is discrete taking integer values between 1 and 29. The concrete compressive strength data set has 1030 instances and 8 attributes. We discretize the values of continuous dependent variable into five bins of equal size. Observations in the same bin share the same ranking label. Although the dependent variable of the Abalone data set is discrete, considering the fact that most applications do not have such a large number of ranking labels as 29, we grouped the observations into three rank classes. All observations whose dependent variable is smaller than 6 are assigned a ranking label 1. All observations whose dependent variable is greater than 15 are assigned a ranking label 3. All the remaining observations belong to rank class 2. The Abalone data set has 4177 instances and 8 attributes.

We used 5-fold cross validation in all data sets. On the artificial data set, our approach showed perfect ranking performance with Kendall τ coefficient being 1. Moreover, the

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

weights for the three noise features were all 0, which implies that the noise features were rejected by the ranking algorithm. This confirms that 1-norm regularization favors sparse solution and is able to select the most discriminative features under a proper value of the λ parameter.

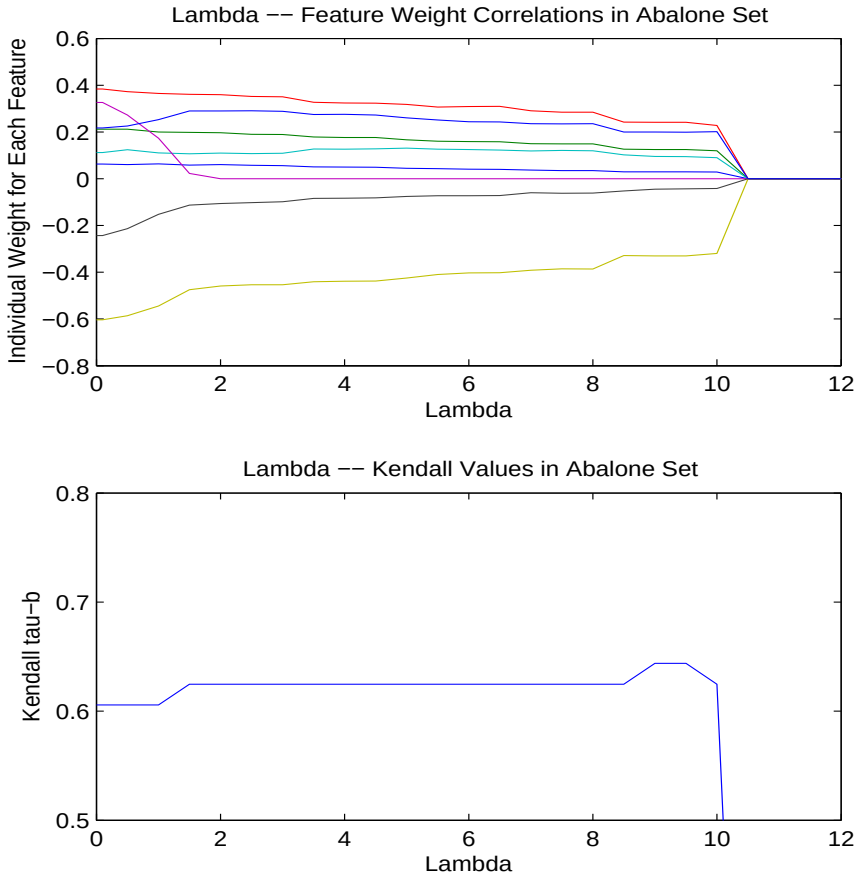

Figure 2.3. The relationship between λ, feature weights (w_j's), and Kendall τ-b rank correlation coefficient on the Abalone data set.

Similar observations were obtained on the other two data sets. The top plots in Figure 2.3 and Figure 2.4 show the feature selection process as the λ parameter in (2.14) varies. The horizontal axis indicates the value of λ. The vertical axis represents the values of the weights. We can see from both figures that as the value of λ increases the absolute values of feature weights decrease. Eventually all the feature weights drop to zero. However, some feature

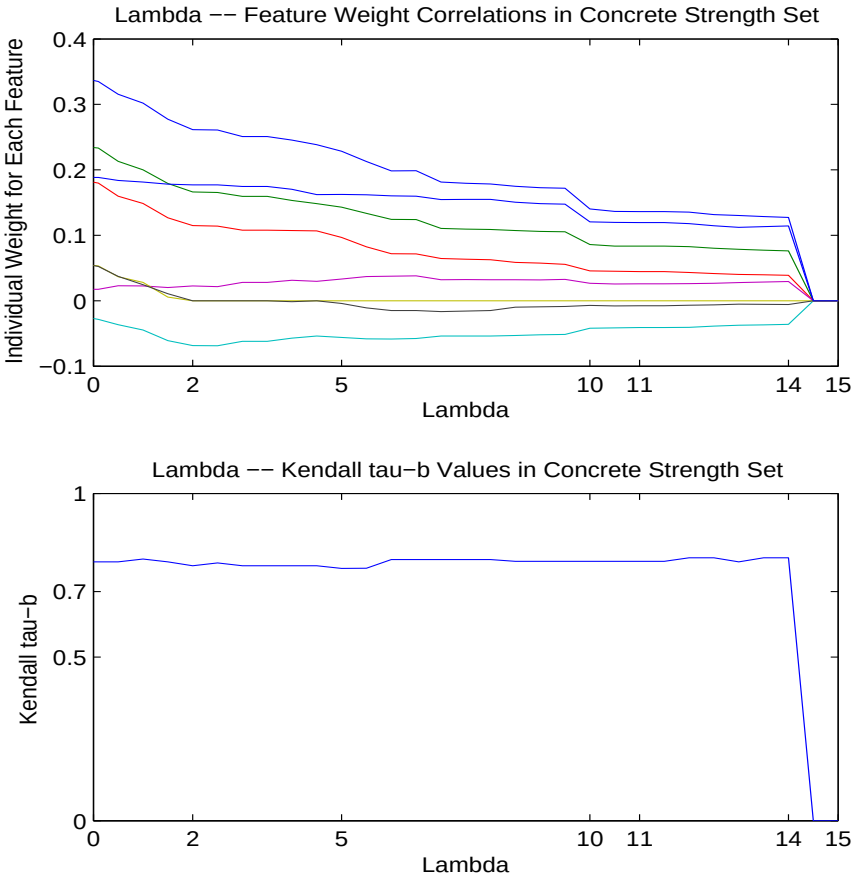

Figure 2.4. The relationship between λ, feature weights (w_j's), and Kendall τ-b rank correlation coefficient on the Concrete Strength data set.

weights decrease more quickly than others. Thus the non-zero-weight features are selected. The bottom plots in Figure 2.3 and Figure 2.4 present the ranking performance as the λ parameter changes. The vertical axis represents the Kendall τ-b value. In the concrete strength data set, 2 feature weights out of 8 drop to zero after λ is equal to 2, which indicates that these two features are "useless" compared with others. The best Kendall τ result, 0.80325, is achieved when λ ranges from 12 to 14. Similar scenario happens in Abalone set. One feature weight drops to zero when λ is greater than 2. It hits the best performance 0.643775 when λ is within the range between 9 and 9.5.

Note that, the reduction ratio is the percentage of the constraints that are removed.

Table 2.1. Number of constraints before and after convex hull reduction.

Data Set	Before Convex Hull Reduction	After Convex Hull Reduction	Reduction ratio
Artificial	640000	174816 ± 2670	72.69%
Concrete Strength	271590 ± 466	178822 ± 4440	34.16%
Abalone	1105700 ± 2556	178060 ± 8833	83.90%

Table 2.2. Convex hull reduction on uniform and normal distributions. There are 1000 observations before the reduction. The right two columns contain the number of observations that are on the convex hull.

Dimension of Feature Space	Normal Distribution	Uniform Distribution
2 Dimension	13	19
3 Dimension	32	72
4 Dimension	85	152
5 Dimension	159	306
6 Dimension	274	496
7 Dimension	358	661
8 Dimension	527	759
9 Dimension	637	852

Next, we look at the effect of convex hull reduction on the number of constraints. In our experiment, the preference pairs are specified as between group observations. We use Quick Hull [2] as the convex hull algorithm. The numbers of preference constraints before and after convex hull reduction are given in Table 2.1. Because 5-fold cross validation is used, we average the number from each run and provide the standard deviation as well. The number of constraint for concrete strength data set is reduced by 34.16%. For the artificial data set and Abalone data set, the reduction ratios are 72.69% and 83.90%, respectively. Although the reduction ratio differs in different data sets, convex hull seems to be an effective option in reducing the constraint set.

However, due to the curse of dimensionality, in a high dimensional space, almost all observations lie on the surface of a convex hull. Therefore we don't expect significant improvements from the proposed ranking algorithm when the feature dimension is large.

Theoretically predicting the number of observations on a convex hull is not an easy task because it depends on the number of observations, the dimension of feature space, and the

[2] `http://www.qhull.org/`

distribution of the observations. Hueter [Hueter, 1999] developed a central limit theorem for the convex hull size in high dimensional spaces. To further investigate how well convex hull reduction works on the ranking problems, we randomly generate 1000 observations with feature dimension varies from 2 to 9, and record the number of observations on the convex hull. Table 2.2 shows that the convex hull size increases with the feature dimension. In a 9-dimensional feature space, the number of observations on the convex hull is close to the size of the whole data set. This is an empirical validation of the curse of dimensionality, which limits the use of convex hull reduction in applications with high feature dimension.

2.3 Recursive Feature Elimination Using 1-Norm Regularization

Feature selection under the framework of 1-norm regularization is achieved by discarding the least significant features, i.e., features with zero weights. The sparsity of the weights is determined by a regularization parameter that controls the trade off between empirical error and model complexity. However, the selection of a proper regularization parameter is a challenging problem. We only know the trend of tuning the parameter to make the number of selected features smaller or larger, but it is difficult to associate a parameter value with a particular feature subset and at the same time achieve a high learning performance, unless the entire regularization path is computed. As 1-norm is non-differentiable (so is hinge loss), calculating the accurate regularization path is difficult (some other loss functions, such as logistic loss, have defined gradients). Even though the regularization path is piecewise linear, path-following methods are slow for large-scale problems. Instead of computing an approximate regularization path, we introduce an iterative 1-norm feature selection framework that selects a small number of features with high performance.

In general, the iterative 1-norm feature selection framework could be used in all learning problems, including classification, regression and ranking. But we demonstrate the method mainly under the scope of multiclass classification, and the extension to regression and ranking is straightforward. In Section 2.2.2, we proposed a multiclass 1-norm regularization

feature selection method, L1MR (Linear 1-norm Multiclass Regularization), and its simple variation SL1MR, that solve a single linear program. Because the 1-norm penalty tends to yield sparse solutions, the proposed formulation has embedded feature selection property. Combined with the idea of recursive feature selection, L1MR and SL1MR identify a small subset of discriminative features effectively and efficiently. The sparsity favoring property of 1-norm regularization enables fast convergence of the iterative feature elimination process. The experimental results show that the iterative elimination process typically completes in less than 10 iterations. Compared with the results obtained from the correlation-based feature selection and information gain methods, L1MR and SL1MR achieve better performance in terms of both the size of selected feature subset and the subset's discriminative power. Compared with support vector machine-recursive feature elimination, our methods have shorter running time and higher accuracy. This work has been published in [Nan *et al.*, 2010b].

2.3.1 Method

It has frequently been observed that 1-norm regularization leads to many feature weights to be zero. This makes it a natural feature selection process, where features with zero weight values should be discarded with no risk. In this section, we call a feature j having zero-weight if all k values, $w_{1,j}, w_{2,j}, \ldots, w_{k,j}$, in the k weight vectors are zero. Otherwise, the corresponding feature has non-zero weight. The purpose of feature selection is to choose a small subset of features and achieve good classification performance. Achieving these two goals simultaneously is difficult.

Although the solutions of the above linear programs are in general sparse, the number of selected features still can be quite large, especially when the dimension of the original feature space is high. By the nature of the 1-norm penalty, increasing the value of parameter λ in (2.6) and (2.7) will reduce the number of non-zero weight features. Figure 2.5 shows the relationship between the number of features selected (non-zero-weight features) and the

parameter λ on the earthworm dataset. It suggests that, by properly tuning the value of λ, different number of features could be selected. Finding a "good" value for λ is a challenging problem. Increasing λ value reduces the size of the selected feature subset, but not necessarily improves the learning performance of the associated model. An accurate solution could be obtained by computing the entire regularization path of L1MR or SL1MR. However, neither of the hinge loss and 1-norm penalty in the multiclass scenario has well-defined gradient [Rosset, 2004], which makes computing the whole solution path extremely difficult.

Inspired by the idea of RFE, we propose a recursive feature elimination approach based on the 1-norm regularization. At each recursive step, a L1MR or SL1MR model is built based on the remaining features from the previous step, and zero-weight features will be eliminated. The recursive process stops when there is no zero-weight feature left. And the final feature subset is chosen based on the associated model with the highest learning accuracy among all the steps. Compared with other embedded feature selection methods, such as SVM-RFE, 1-norm regularized multinomial logistic regression and other single multiclass classifier, the proposed feature selection method has the following properties:

- The recursive process narrows down the feature subset step by step, hence minimizes the number of features selected, which is more aggresive than 1-norm regularized multinomial logistic regression and other single multiclass classifiers.

- The number of feature to be eliminated at each step is automatically determined, i.e., features with zero weights. Whereas in SVM-RFE, the number of features to be eliminated has to be predetermined, e.g. deleting one feature with the least feature weight.

- The recursive process normally involves a smaller number of iterations than SVM-RFE. We have observed that the number of features eliminated at each step varies greatly:

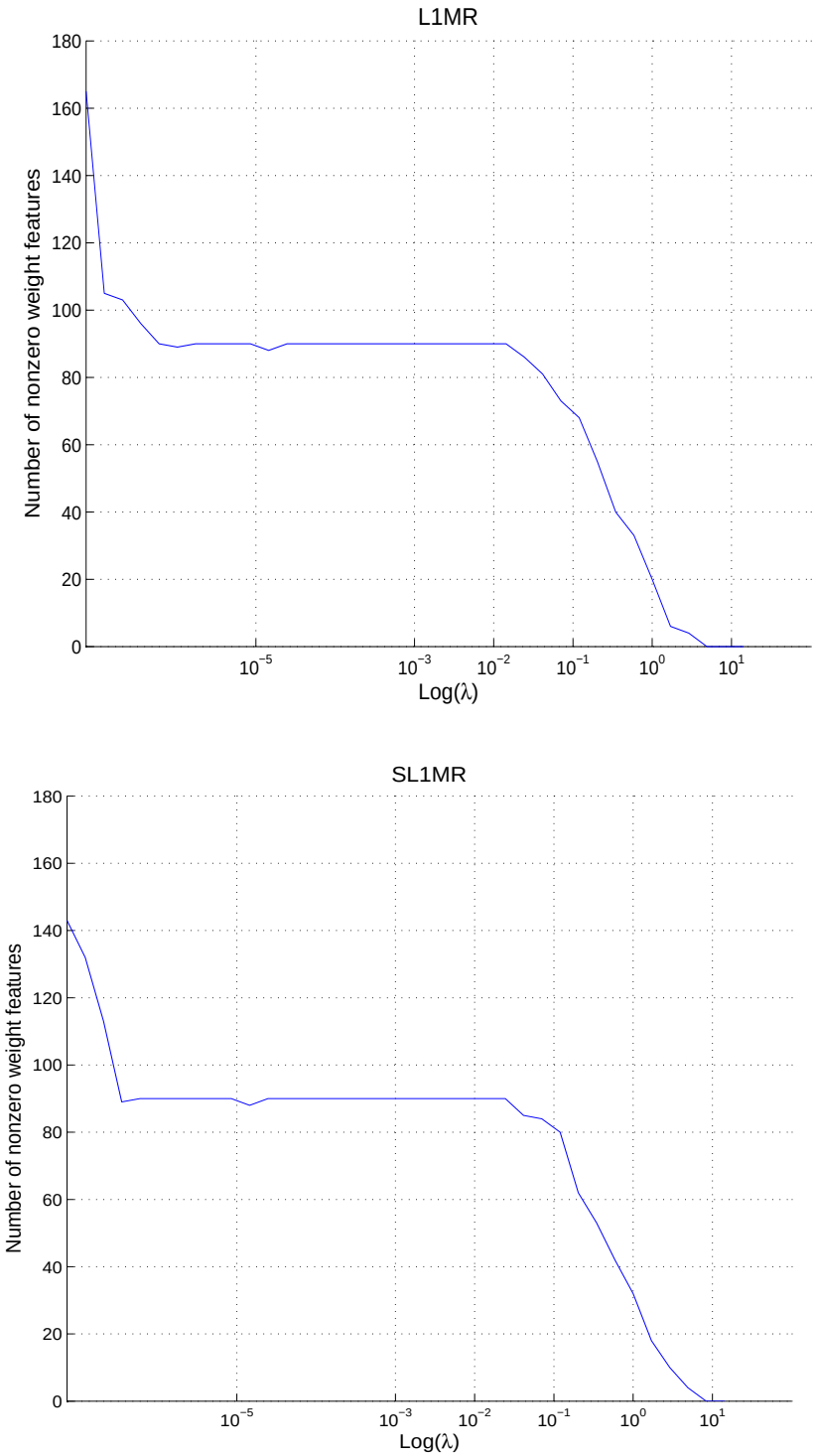

Figure 2.5. The relationship between λ and the number of non-zero-weight features.

dramatic reduction occurs at the first (few) step(s), and shrinks when it comes close to the optimal feature subset. The convergence is typically very fast.

- It has a natural stopping criteria: when the weights of the remaining features are nonzero.

To obtain a stable variable selection result, the instances are resampled multiple times and we choose variables that occur in a large portion of the multiple selection results. This idea is similar as stability selection [Meinshausen & Bühlmann, 2010] in the sense that multiple random sampling reduces the result variance. However, stability selection tests all regularization parameter (cost-parameter) under consideration and our approach includes a parameter selection procedure. As emphasized in [Zhang *et al.*, 2006] and [Ambroise & McLachlan, 2002], feature selection steps external to the resampling procedures may severely bias the evaluation in favor of the learned method due to "information leak". In our method, the feature selection is put inside the resampling. Because the L1MR and SL1MR include the process of tuning a cost-parameter, a double loop [Statnikov *et al.*, 2006] is used: the inner loop is used to select the best parameter, and the outer loop is used for estimating the performance of the learning model built using the previously found best parameter on an independent test set.

The recursive feature selection based on 1-norm procedure is described in detail as follows:

1. Resample data set by k random splits. [3]

2. Recursive feature selection at each split F_j. This step is stated in detail as follows,

   (a) At iteration i, build L1MR or SL1MR model with current S_{i,F_j} features. Initially, S_{1,F_j} consists of all the gene features. This sub-step includes an inner cross validation loop or bootstrap to decide the best parameter with which the learning model yields the best performance on the inner validation set. [4]

[3] 20 random splits is used in our experiment.

[4] 10-fold cross validation was recommended in [Ambroise & McLachlan, 2002] and is used in the inner loop of our experiment.

(b) Remove the zero-weight features from the old feature set and obtain a new one, S_{i+1,F_j}.

(c) Set $i = i + 1$. Repeat from sub-step (a) until S_{i,F_j} does not change (in other words, there are no zero-weight features).

3. For each split, record the optimal feature subset $S^{\star}_{F_j}$ with the maximum accuracy rate $Acc^{\star}_{F_j}$.

4. Aggregate results from different splits.

In the outer loop, each split generates different subsets of features. The workflow of the recursive feature selection framework can be expressed in Figure 2.6.

These subsets usually overlap with each other significantly. Nevertheless, we do need to combine these results. In [Zhang *et al.*, 2006], Zhang et al. utilized a frequency-based selection method that ignores the ordering of features. Because the list of $S^{\star}_{f_j}$ could be viewed as a rank list where gene features are sorted in descending order based on their importance, Borda's method is used in this paper for rank aggregation [Dwork *et al.*, 2001]. Suppose there are k ranking lists. Borda's method first assigns to each feature a k-dimension vector indicating the positional scores of the feature in the k lists. It then sorts the features by the sum of these vectors. A major advantage of using Borda's method is its low computational complexity, linear to the size of the list. Also, this method could be extended to partial lists by apportioning the excess scores equally to all unranked features.

The number of features selected is the average size of the selected subset in each split, i.e.,

$$\left\lfloor N = \frac{\sum_{j=1}^{k} |S_{F_j}|}{k} \right\rfloor, \tag{2.15}$$

with $k = 20$ in our experiment, where $|S_{F_j}|$ is the size of selected feature subset in split F_j. After Borda's method, the top N features are selected from the aggregation ranking list.

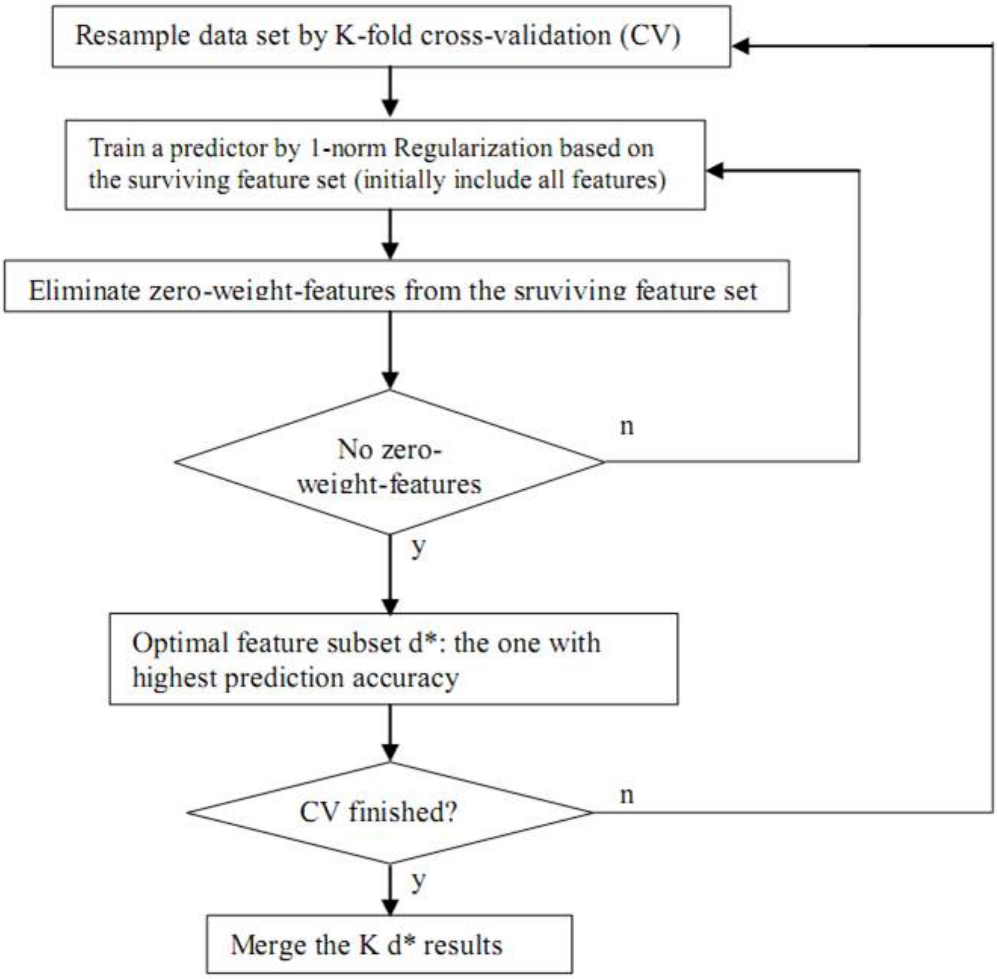

Figure 2.6. The workflow of the recursive feature selection using 1-norm penalty framework

The estimated accuracy is calculated as the average of the k-split accuracies, i.e.

$$\frac{\sum_{j=1}^{k} Acc^{\star}_{F_j}}{k}\ , \tag{2.16}$$

where $Acc^{\star}_{F_j}$ indicates the cross validation accuracy within split F_j.

2.3.2 Experiments and Results

Adult earthworms (*Eisenia. fetida*) were exposed in soil spiked with TNT (0, 6, 12, 24, 48, or 96 mg/kg) or RDX (8, 16, 32, 64, or 128 mg/kg) for 4 or 14 days. The 4-day treatment was repeated with the same TNT concentrations and the RDX concentration

being 2, 4, 8, 16 or 32 mg/kg soil. Each treatment originally had 10 replicate worms with 8-10 survivors at the end of exposure, except the two highest TNT concentrations. At 96 mg TNT/kg, no worms survived in the original 4-day and 14-day exposures, whereas at 48 mg TNT/kg, all 10 worms died in the original 4-day exposure. Total RNA was isolated from the surviving worms as well as the Day 0 worms (worms sampled immediately before experiments). A total of 248 worm RNA samples (8 replicate worms with 31 treatments) were hybridized to a custom-designed oligo array using Agilent's one-colour Low RNA Input Linear Amplification Kit. The array contained 15,208 non-redundant 60-mer probes (GEO platform accession number GPL9420), each targeting a unique *E. fetida* transcript. After hybridization and scanning, gene expression data were acquired using Agilent's Feature Extraction Software (v.9.1.3). In the current study, the 248-array dataset was divided into three worm groups: 32 untreated controls, 96 TNT-treated, and 120 RDX-treated. This dataset has been deposited in NCBI's Gene Expression Omnibus and is accessible through GEO Series accession number GSE18495 [5].

We applied the L1MR and SL1MR on the earthworm microarray gene expression data which is categorized into three classes. L1MR and SL1MR were implemented using Matlab, and 1-norm multiclass regularization was transformed into standard optimization problem format. Then the optimization problem was solved by CPLEX (an optimization software package).

Table 2.3, 2.4, 2.5 and 2.6 show the detailed results using L1MR and SL1MR methods, respectively. The number of iterations is less than ten in each split, which suggests fast convergence of our methods, greatly alleviating the computational burden posed by the traditional RFE method. In all of our tests, the first iteration in both methods always achieves the largest feature elimination. Sometimes, the process achieves an optimal solution after just one iteration. In most scenarios, accuracy follows a bell shaped curve along the selection steps.

[5] http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE18495

Table 2.3. Classification and Feature Selection Results of L1MR (Part I).

Fsize	I1	I2	I3	I4	I5	I6	I7	I8	I9
Split 1	144	122	103	70	63	59	58	57	55
Split 2	53	50	-	-	-	-	-	-	-
Split 3	75	65	63	59	53	52	-	-	-
Split 4	69	63	61	60	56	54	52	-	-
Split 5	144	122	103	70	63	59	58	57	55
Split 6	63	59	57	-	-	-	-	-	-
Split 7	61	59	-	-	-	-	-	-	-
Split 8	78	-	-	-	-	-	-	-	-
Split 9	138	110	97	93	85	83	80	70	67
Split 10	65	58	50	49	-	-	-	-	-
Split 11	121	101	78	76	-	-	-	-	-
Split 12	133	130	112	79	63	59	58	57	55
Split 13	75	67	61	53	49	-	-	-	-
Split 14	144	75	63	59	52	-	-	-	-
Split 15	77	73	69	63	61	-	-	-	-
Split 16	137	117	103	97	83	70	67	-	-
Split 17	63	59	-	-	-	-	-	-	-
Split 18	67	63	61	-	-	-	-	-	-
Split 19	127	103	93	87	83	70	63	-	-
Split 20	69	67	63	59	57	-	-	-	-

Fsize: The number of features selected.
I1-I9: The first iteration to the 9th iteration.

We compared L1MR and SL1MR with two well-known statistical methods in the literature – correlation-based feature selection (CFS) and information gain feature selection (IG), and also with two embedded multiclass feature selection methods – SVM-RFE and L1 regularized multinomial logistic regression. CFS [Hall, 2000] offers a heuristic of individual features for predicting the class labels, whereas IG measures the expected reduction in entropy. SVM-RFE [Guyon *et al.*, 2002] trains a multiclass SVM classifier by the method of one-versus-one and eliminates the least-weight-feature at a time. L1 regularized multinomial logistic regression could handle multiclass classification problem [Friedman *et al.*, 2010], and feature selection is implemented by eliminating zero-weight features. SVM-RFE was coded by Matlab and libsvm, a free library for Support Vector Machines [6]. L1 regularized multinomial logistic regression was implemented by a free Matlab package offered by Mark Schmidt [7]. We utilized the CFS and IG packages in Weka directly. CFS was

[6]http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[7]http://www.cs.ubc.ca/ schmidtm/Software/L1General.html

Table 2.4. Classification and Feature Selection Results of L1MR (Part II).

Accuracy	I1	I2	I3	I4	I5	I6	I7	I8	I9
Split 1	74.0%	74.0%	86.0%	90.0%	78.0%	90.0%	86.0%	86.0%	86.0%
Split 2	72.8%	72.8%	-	-	-	-	-	-	-
Split 3	85.3%	89.5%	77.0%	77.0%	81.2%	81.2%	-	-	-
Split 4	78.0%	86.0%	78.0%	86.0%	78.0%	78.0%	82.0%	-	-
Split 5	74.0%	74.0%	86.0%	90.0%	78.0%	90.0%	86.0%	86.0%	86.0%
Split 6	89.5%	93.7%	89.8%	-	-	-	-	-	-
Split 7	82.0%	86.0%	-	-	-	-	-	-	-
Split 8	94.3%	-	-	-	-	-	-	-	-
Split 9	86.0%	98.0%	94.0%	94.0%	94.0%	94.0%	98.0%	98.0%	98.0%
Split 10	62.0%	58.0%	70.0%	58.0%	-	-	-	-	-
Split 11	77.5%	77.5%	77.7%	77.7%	-	-	-	-	-
Split 12	73.3%	75.0%	77.5%	79.0%	81.0%	82.3%	78.0%	86.0%	73.0%
Split 13	74.0%	72.8%	78.0%	89.5%	83.0%	-	-	-	-
Split 14	82.0%	86.0%	78.0%	92.0%	89.0%	-	-	-	-
Split 15	79.2%	82.0%	91.5%	78.0%	82.0%	-	-	-	-
Split 16	69.8%	71.5%	73.0%	78.0%	71.5%	73.3%	72.0%	-	-
Split 17	92.1%	89.0%	-	-	-	-	-	-	-
Split 18	79.0%	84.6%	82.3%	-	-	-	-	-	-
Split 19	81.0%	82.1%	79.5%	77.0%	82.1%	79.0%	79.0%	-	-
Split 20	75.8%	74.0%	79.5%	81.2%	79.5%	-	-	-	-

Accuracy: The classification accuracy.
I1-I9: The first iteration to the 9th iteration.

combined with BestFirst search algorithm and IG was associated with the Ranker search method. Most of the settings are default setting in Weka, except for the number of features to select in IG is set to 64 to match the results of L1MR and SL1MR. In order to make fair comparisons, the same 20 random splits were also carried out in CFS , IG, SVM-RFE, L1 regularized multinomial logistic regression with the data splits identical to those used in L1MR and SL1MR. Because CFS and IG are pure feature selection approaches, to evaluate the learning performances on the selected feature subsets, additional model-building process is needed. An outline of the CFS and IG processes is given below:

1. Use the same setting to divide the data set into 20 splits.

2. For each split, implement CFS and IG feature selection separately. For either CFS or IG, aggregate the 20 feature selection results by Borda's method. CFS selects 60 features after the aggregation step.

Table 2.5. Classification and Feature Selection Results of SL1MR (Part I).

Fsize	I1	I2	I3	I4	I5	I6	I7	I8
Split 1	65	55	54	52	-	-	-	-
Split 2	53	50	-	-	-	-	-	-
Split 3	68	61	58	-	-	-	-	-
Split 4	54	50	-	-	-	-	-	-
Split 5	33	31	30	-	-	-	-	-
Split 6	51	50	-	-	-	-	-	-
Split 7	73	66	63	60	56	54	52	51
Split 8	85	76	73	72	66	64	62	61
Split 9	43	38	-	-	-	-	-	-
Split 10	45	41	39	-	-	-	-	-
Split 11	65	54	52	-	-	-	-	-
Split 12	55	54	50	-	-	-	-	-
Split 13	37	35	-	-	-	-	-	-
Split 14	44	41	-	-	-	-	-	-
Split 15	53	50	44	41	-	-	-	-
Split 16	78	75	73	66	63	56	-	-
Split 17	66	63	59	52	-	-	-	-
Split 18	33	30	-	-	-	-	-	-
Split 19	79	73	70	68	65	60	56	51
Split 20	47	41	37	-	-	-	-	-

Fsize: The number of features selected.
I1-I9: The first iteration to the 9th iteration.

3. Generate a new data set by eliminating any feature that is not on the aggregation list.

4. Train a classifier by L1MR method and estimate the classification accuracy. This learning step implements double loop cross validation as we do in L1MR and SL1MR learning, with the same outer loop data partition.

The comparison results are given in Table 2.7. Because of 20 random splits, accuracy average and standard deviation are provided. From here, we use abbreviations "L1-RMLR" for "L1 regularized multinomial logistic regression". Except for L1-RMLR, other approaches selected small number of features. Though, L1MR and SL1MR achieved higher average accuracies than all the comparison methods, the high standard deviation makes it hard to claim the superiority of our methods. Therefore, we ran a t-test to compare the difference between our method (L1MR or SL1MR) and any comparison method (L1-RMLR, SVM-RFE, CFS, or IG). For each pair, e.g. L1MR and SVM-RFE, the t value was computed based on their learning performances on the 20 random splits. These performance values

Table 2.6. Classification and Feature Selection Results of SL1MR (Part II).

Accuracy	I1	I2	I3	I4	I5	I6	I7	I8
Split 1	94.0%	86.0%	90.0%	90.0%	-	-	-	-
Split 2	72.8%	72.8%	-	-	-	-	-	-
Split 3	85.3%	85.3%	85.3%	-	-	-	-	-
Split 4	86.0%	74.0%	-	-	-	-	-	-
Split 5	90.0%	94.0%	98.0%	-	-	-	-	-
Split 6	89.5%	93.7%	89.8%	-	-	-	-	-
Split 7	90.0%	90.0%	98.0%	90.0%	90.0%	90.0%	86.0%	78.0%
Split 8	90.5%	98.2%	90.5%	86.6%	86.6%	90.5%	94.3%	90.5%
Split 9	94.0%	94.0%	-	-	-	-	-	-
Split 10	70.0%	62.0%	66.0%	-	-	-	-	-
Split 11	92.1%	90.0%	92.1%	-	-	-	-	-
Split 12	88.0%	88.0%	84.5%	-	-	-	-	-
Split 13	90.0%	89.8%	-	-	-	-	-	-
Split 14	86.5%	90.6%	-	-	-	-	-	-
Split 15	77.8%	75.4%	78.0%	77.8%	-	-	-	-
Split 16	79.2%	81.0%	88.0%	83.3%	82.0%	83.3%	-	-
Split 17	94.5%	94.5%	92.0%	92.0%	-	-	-	-
Split 18	88.0%	84.5%	-	-	-	-	-	-
Split 19	86.0%	85.5%	89.8%	90.0%	88.5%	89.8%	86.0%	88.5%
Split 20	90.0%	88.5%	86.4%	-	-	-	-	-

Accuracy: The classification accuracy.
I1-I9: The first iteration to the 9th iteration.

are shown in Figure 2.7. The calculated t values are shown in Table 2.8. According to the t table with $20 - 1 = 38$ degrees of freedom, we see that for the threshold = 0.05 the tabled valued is 2.093, and for threshold = 0.01 the tabled valued is 2.861. Our calculated values are mostly larger than the tabled value at threshold = 0.01 except for the pair of L1MR and SVM-RFE. So we reach the conclusion that SL1MR outperformed all the comparison methods, and L1MR achieved better performance than L1-RMLR, CFS and IG. The difference between L1MR and SVM-RFE is not statistically significant. In all, SL1MR shows the best feature selection result by selecting a small number of features and generating the highest classification accuracy.

Table 2.9 lists the top 20 earthworm genes selected by SL1MR and the position rank of of these gene given by L1MR, CFS and IG. Due to page limit, the remaining 32 genes (ranked between 21th to 52th by SL1MR) are not shown. These 52 genes together are identified as the classifier genes that are highly discriminative in separating the earthworm samples into

Table 2.7. Comparison of L1MR, SL1MR, L1-RMLR, SVM-RFE, CFS, and IG.

Method	Fsize	Accur
L1MR	64	**86.25% ± 7.58%**
SL1M	52	**88.57% ± 7.56%**
L1-RMLR	682	83.72% ± 5.35%
SVM-RFE	35	85.62% ± 5.58%
CFS	64	77.98% ± 5.22%
IG	64	78.16% ± 4.39%

Fsize, Accur, same as Table 2.3 and 2.4
CFS: Correlation-based feature selection.
IG: Information gain feature selection.

Table 2.8. T Values for L1MR/SL1MR and Comparison Method.

	L1-RMLR	SVM-RFE	CFS	IG
L1MR	3.1724	0.7826	10.4234	10.5488
SL1M	6.0317	3.6335	13.2457	13.4653

control, TNT-treated and RDX-treated categories.

Among the top 20 genes, 19 genes (95%) have meaningful annotation with a wide range of biological functions spanning from detoxification (glutathione S-transferase pi and Cadmium-metallothionein) to spermatogenesis (valosine containing peptide-2 or VCP-2) and signal transduction (signal peptides). VCP-2, a gene expressed specifically in the anterior segments of sexually mature earthworms [Suzuki *et al.*, 2005], is identified by three algorithms and ranked in the top 10, suggesting that both TNT and RDX may affect spermatogenesis. Elongation factor 2 or EF2, a putative gene targeted by two probes, TA1-057226 and TA2-006089, is one of the proteins that facilitate the events of translational elongation, the steps in protein synthesis from the formation of the first peptide bond to the formation of the last one [Jorgensen *et al.*, 2006]. This strongly indicates that protein synthesis may be targeted by both TNT and RDX. Nevertheless, more work should be devoted to exploring biological functions and interactions of the identified top classifier genes that may lead or be linked to toxicological effects or biochemical endpoints.

As a continuation of this work, we would like to validate the biological significance of the identified genes because our main purpose was to discover novel gene biomarkers

Table 2.9. Top 20 Genes Selected by SL1MR.

Rank in SL1MR	Probe Name	Rank in L1MR	Rank in L1-RMLR	Rank in SVM-RFE	Rank in CFS	Rank in IG	Target Gene Annotation
1	TA1-153745	6	17	8	-	-	succinate dehydrogenase iron-sulfur protein
2	TA1-057226	2	4	5	23	2	elongation factor-2
3	TA1-058194	4	9	7	-	-	Cytoplasmic
4	TA2-010658	9	2	4	-	-	40S ribosomal protein S8
5	TA2-095861	14	36	19	-	-	tropomyosin
6	TA2-006089	3	1	6	-	35	elongation factor 2
7	TA1-167854	13	57	26	13	-	Cytoplasmic
8	TA1-194525	8	23	3	-	-	valosine containing peptide-2
9	TA1-056351	12	74	20	60	-	Unknown
10	TA1-166421	18	15	13	-	-	40S ribosomal protein S30
11	TA2-164073	17	39	9	-	-	Signal peptide
12	TA1-016697	23	55	16	-	-	peptidase-C13
13	TA1-003758	19	76	-	-	-	MT-EISFO Cadmium-metallothionein
14	TA2-111531	25	130	-	-	-	peptidase-C13
15	TA2-179329	11	49	-	-	-	Cytoplasmic
16	TA2-065306	5	13	33	-	-	Signal peptide
17	TA2-131920	28	61	15	-	-	glutathione S-transferase pi
18	TA1-030037	1	3	1	7	-	Lumbricus rubellus mRNA for 40S ribosomal protein S10
19	TA2-026889	34	155	-	-	-	Rattus norvegicus similar to Finkel-Biskis-Reilly murine
20	TA2-113782	15	86	28	10	-	signal peptide

sensitive and specific to environmental toxicants. In the future, we would also like to perform experiments comparing these algorithms on a wider range of data sets.

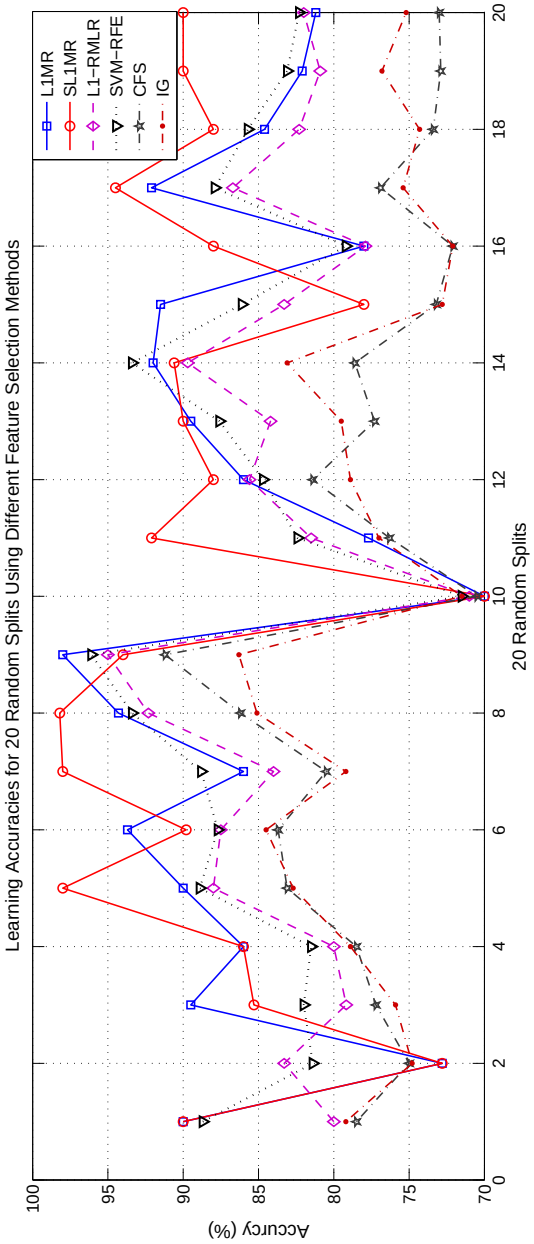

Figure 2.7. Learning Accuracies for 20 Random Splits Using Different Feature Selection Methods.

CHAPTER 3

FEATURE SPACE PARTITION

The basic concept of feature space partitioning is different from that used in the construction of decision tree. Here, the partition is specific for the objective to simplify the learning problem so that each region after partition has a better discriminant ability. Once the partition of feature space is achieved, a simple predictor, for example a linear classifier, is learned on each region.

3.1 Restructuring Problem by Feature Space Partition

In statistical learning, a predictive model is learned from a hypothesis class using a finite number of training samples [Vapnik, 1995]. The distance between the learned model and the target function is often quantified as the generalization error, which can be divided into an approximation term and an estimation term. The former is determined by the capacity of the hypothesis class, while the latter is related to the finite sample size. Loosely speaking, given a finite training set, a complex hypothesis class reduces the approximation error but increases the estimation error. Therefore, for good generalization performance, it is important to find the right tradeoff between the two terms. Along this line, an intuitive solution is to build a simple predictive model with good training performance [Niyogi *et al.*, 1998]. But, many problems have complex underlying data distribution generative function. And using a simple predictor on the whole data set will result in high prediction error. Even worse, the "high dimensionality, small sample size" nature of many applications, such as microarray classification and prediction in cheminoformatics, makes it extremely challenging

to build a good predictive model: a simple model often fails to fit the training data, but a complex model is prone to overfitting. A commonly used strategy to tackle this dilemma is to simplify the problem itself dividing a learning task into several simpler problems, for which building predictive models with good generalization is feasible. The problem dividing is implemented by feature space partitioning so that the feature space is separated into multiple non-overlapping regions each of which represents a simple sub-problem. From a machine learning perspective, the partition actually restructures the original problem into multiple sub-problem with different learning properties.

The concept of dividing feature space has been proposed back in early 1980s, but only served as a classification method then. Because researchers considered the problem of classifier design as partitioning the space into a number of disjoint regions, and the classification then is nothing but the determination of the region to which an unknown sample belongs [Sethi & sarvarayudu, 1982]. Decision tree learning [Breiman *et al.*, 1984] was generated based on that idea. Although, strictly speaking, decision tree learning includes feature space partition when a decision is made by sequentially splitting feature axes. But our purpose here is to use feature space partition to **facilitate not solve** a supervised learning problem.

Not many researches have been explored about restructuring learning problem using feature space partition. A related work has been done by Padmanabhan et al [Padmanabhan *et al.*, 1999]. The work has been patented as well [Bahl *et al.*, 2000]. They described a decision tree based technique that partitions the feature space in a hierarchical fashion where each node in the tree represents a linear hyperplane, and the end result is that the entire feature space is partitioned into non-overlapping regions where each region is bounded by a number of hyperplanes. The criterion for the design of the hyperplanes is the minimization of the average class entropy of the regions. We will introduce this method in section 3.2. Padmanabhan *et al.* [1999] splits the feature space by hyperplane using entropy-based evaluation, and another splitting procedure is to cut the space into hyperparallelepipeds such

that their faces are perpendicular to the feature axes until certain stopping criterion is met. The latter partition procedure was used by Kohn et al [Kohn *et al.*, 1996] and Singh [Singh & Galton, 2003] for class separability measure. Also, hyperparallelepiped parition was used by Kishore et al. [Kishore *et al.*, 2001] to facilitate genetic programming. He et al. [He *et al.*, 2005] discussed a coarse-to-fine strategy to divide the hyperparallelepipeds. The hyperparallelepiped partition will be discussed in section 3.3. Other partitioning strategy includes data similarity measurement. For example, Wang et al. [Wang *et al.*, 2003] associated a partitioned hypersphere with each sample point with the radius equal to the distance to the closest point of the opposite class.

We propose a method of restructuring supervised learning problem using a discrete/categorical attribute. Such attributes naturally divide the original problem into several non-overlapping sub-problems. With a proper choice of the attribute, the complexity of the learning task is reduced, and the prediction performance enhanced. Selecting a proper discrete or categorical attribute that maximally simplifies the learning task is a challenging problem. A naïve approach requires exhaustive searching for the optimal learning model for each possible restructured problem, and hence is computationally prohibitive. We propose a metric to select the categorical attribute based on the estimated expected conditional entropy with respect to random projections. This method can be applied to multi-class and non-linear problems. Experimental results demonstrate the good performance of the proposed approach on several data sets. This work has been published in [Nan *et al.*, 2011] and will be introduced in Section 3.4.

3.2 Hierarchical Linear Hyperplane Partition

This method constructs a binary tree structure to hierarchically partition the data using linear hyperplanes. The hyperplane was computed to partition a multidimensional feature space to maximize the mutual information. Padmanabhan et al [Padmanabhan *et al.*, 1999] showed that designing the hyperplane is equivalent as solving the linear discriminant function

of the data. Therefore, for each node of the tree, a binary classification problem needs to be solved.

The feature space is partitioned in a hierarchical manner using a binary decision tree, starting with all the training data at the root node of tree. Each node of the decision tree partitions the training data at the node into two parts based on which side of a specified hyperplane each training data falls. The process continues either till the data at a node falls below a specified threshold, or till the tree has been grown to a specified depth. The terminal nodes of the tree represent non-overlapping regions of the feature space. As these regions were designed with the objective of minimizing the class entropy, it is expected that each region contains only a small subset of the entire class. For any new input feature vector, starting from the root node, the projection of the new feature vector on the hyperplane is computed. The projection is the compared to a threshold. Depending on whether the value is smaller or greater than the threshold, the left or right child nod of the current node is selected. The process is repeated for the selected child node, and terminates when the child node is a terminal node of the decision tree.

Because each node separates a region into two parts, it is difficult for the method to handle multi-class scenario. Moreover, finding an optimal hyperplane for each node is equivalent as solving a linear discriminant problem, which is not trivial in many cases.

3.3 Hyperparallelepipeds Partition

The feature space partitioning process could be generalized as followings:

Given a training data set, at the beginning, for each feature, its minimum and maximum values are recorded. A hyperplane that is perpendicular the a feature axis and goes though the feature's minimum or maximum data point is built. Those hyperplane intersect with each other and generate a number of hyper-rectangular parallelepipeds (hyperboxes). For example, let x_{1min} and x_{1max} be the minimum and maximum values obtained from the projections of all the training observations on the axis of feature x_1, and there is at least

one hyperbox whose two faces are perpendicular to the feature x_1. A hyperbox at any stage during the feature space partitioning process should first test whether a stopping criterion is satisfied. Assuming none of the stopping condition is met, the hyperbox is partitioned into two or more sub-hyperboxes. Most algorithms perform binary splitting along the the feature that has the largest range (the difference between the maximum and minimum). But more than two splitting was executed in [Valev, 2004]. The splitting point could be chosen at the median of the samples projected on that feature coordinate [Kohn *et al.*, 1996]. The partitioning process is finished when there are no remaining hyperboxes to be split. In other words, all the hyperboxes satisfy at least one of the stopping criteria.

The feature space partitioning stopping criteria are measurements about the "purity" of a hyperbox. If a hyperbox contains data only from one class, then it is totally pure and if it contains data from a number of classes in equal amounts it is then totally impure. The purity measure could be calculated by entropy. Some researchers defined their own purity measurements. For example, Singh et al. [Singh & Galton, 2003] used a degree of separability in a hyperbox, and Wang et al. [Wang *et al.*, 2003] applied a locally defined confidence measure. General stopping criteria are:

- the hyperbox is homogeneous, i.e. it contains samples from a single class;
- the samples in the hyperbox are from linearly separable classes;
- the number of samples in the hyperbox is less than a threshold, and the threshold is associated with the dimension.

The first condition is quite obvious. Any input feature vector drops into such hyperbox will be automatically designated a class label. There's no further learning needed there.

The second criterion is developed to show the sub-problem introduced by the hyperbox is simple enough and no further splitting is required. If the hyperbox has small volume and are separable with nonlinear hypersurfaces, the sections of the nonlinear boundaries may

be approximated as linear, and further splitting is avoid. However, how to check the linear separability is a key issue, because normally, we don't want to solve linear discriminant problem here. Kohn et al. [Kohn *et al.*, 1996] suggested a quick test method. Assuming there are samples from c classes in the hyperbox, the corresponding c centroids are found and we connect those centroids by lines. For each line, samples in the hyperbox are projected onto the line, and if the projections of samples from each class do not overlap with any of the projections of the other classes, then linear separability along that line is satisfied. If such separability condition is satisfied for all lines, the second stopping criterion is satisfied.

The third stopping condition is needed to avoid the occurrence of very small hyperbox. The learning process is hard to be carried out in a region with very few samples.

3.4 Leveraging Domain Information to Restructure Prediction Problem

The use of domain information in problems such as biological prediction has notable effects. There is an abundance of prior work in the field of bioinformatics, machine learning, and pattern recognition. It is beyond the scope of this dissertation to supply a complete review of the respective areas. Nevertheless, a brief synopsis of some of the main findings most related to this thesis will serve to provide a rationale for incorporating domain information in supervised learning.

The simplicity of a learning model is thus essential for the success of statistical modeling. However, the representational power of a simple model family may not be enough to capture the complexity of the target function. In many situations, a complex target function may be decomposed into several pieces, and each can be easily described using simple models. Three binary classification examples are illustrated in Figure 3.1, where red/blue indicates positive/negative class. In example (a), the decision boundary that separates two distinct color regions is a composite of multiple polygonal lines. It suggests the classification problem

in (a) could not be solved by a simple hypothesis class such as a linear or polynomial model. Similarly, in examples (b) and (c), the decision boundary is so complex that neither a linear nor polynomial model can be fitted into these problems. Nevertheless, if the whole area is split into four different sub-regions (as shown in the figure, four quadrants marked from 1 to 4), the problem could be handled by solving each quadrant using a simple model individually. In example (a), the sub-problem defined on each quadrant is linearly separable. Likewise, each quadrant in (b) is suitable for a two-degree polynomial model. A linear model can be viewed as a special case of a two-degree polynomial. Therefore, the four sub-problems in (c) could be solved by a set of two-degree polynomial models. In the three examples, a categorical attribute X_3 provides such partition information.

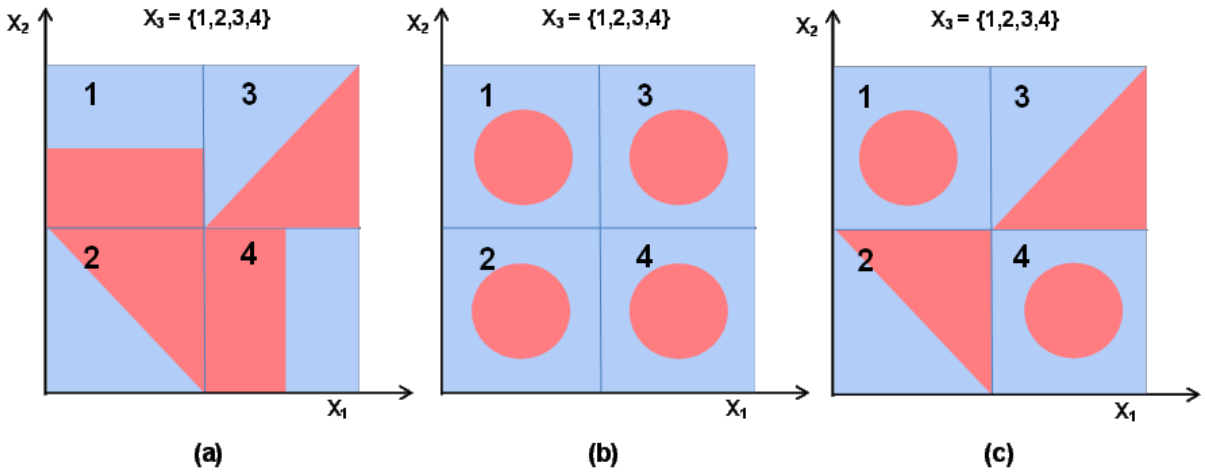

Figure 3.1. Examples of Piece-wise Separable Classification Problems.

Attributes like X_3 exist in many applications. For instance, leukemia subtype domain knowledge, which can be encoded as a discrete or categorical attribute, may help the prediction of prognosis. A discrete or categorical attribute provides a natural partition of the problem domain, and hence divides the original problem into several non-overlapping sub-problems. As depicted in Figure 3.2, the original problem is split into multiple sub-problems by one or more discrete or categorical attributes. If the proper attribute is selected in the re-

structuring process, each sub-problem will have a comparably simpler target function. Our approach is fundamentally different from the decision tree approach [Rokach & Maimon, 2008]: first, the tree-like restructuring process is to break up the problem into multiple more easily solvable sub-problems, not to make prediction decisions; second, the splitting criterion we propose here is based on the conditional entropy achieved by a categorical attribute and a hypothesis class, whereas the conditional entropy in decision trees is achieved by an attribute only. The conditional entropy will be discussed in detail later. Also, our method is related to feature selection in the sense that it picks categorical attributes according to a metric. However, it differs from feature selection in that feature selection focuses on the individual discriminant power of an attribute, and our method studies the ability of an attribute to increase the discriminant power of all the rest of the attributes. The categorical attributes selected by our method may or may not be selected by traditional feature selection approaches.

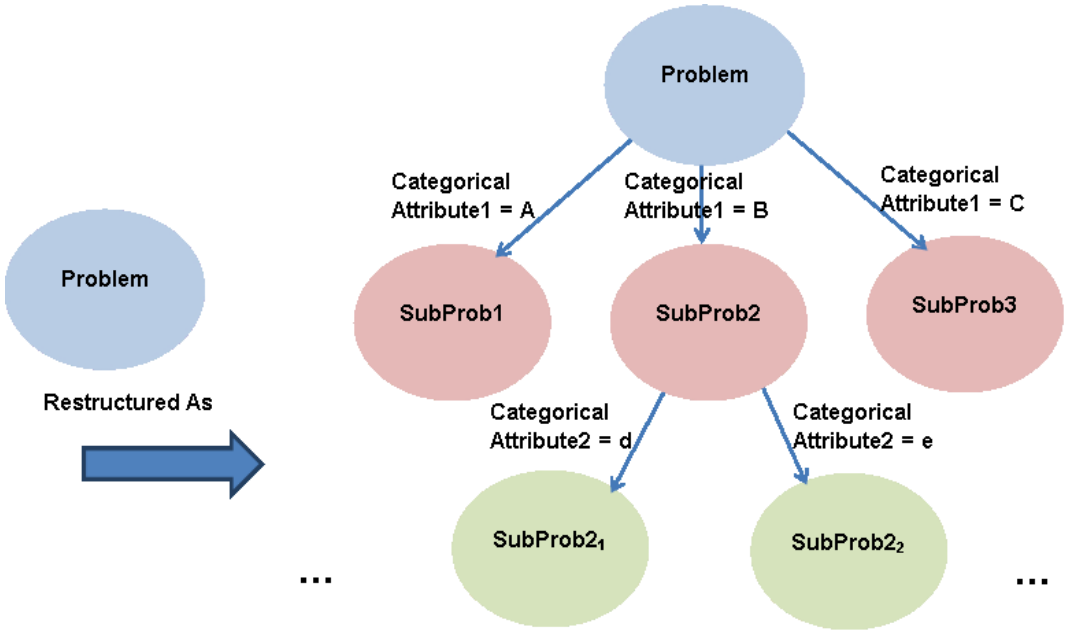

Figure 3.2. Restructuring Learning Problem By One or More Categorical Attribute.

In theory, there's no limit on the number of categorical attributes used in a partition

if an infinite data sample is available. However, in reality, the finite sample size puts a limit on the number of sub-problems good for statistical modeling. In this article, we only consider incorporating one discrete or categorical attribute at a time. Identifying a discrete or categorical attribute that provides a good partition of a problem is nontrivial when the number of discrete or categorical attributes is large. We propose a metric to rank these attributes in section 3.4.2.

In this thesis, we present a novel method that uses domain information encoded by a discrete or categorical attribute to restructure a supervised learning problem. To select the proper discrete/categorical attribute to maximally simplify a classification problem, we propose an attribute selection metric based on conditional entropy achieved by a set of optimal classifiers built for the restructured problem space. As finding the optimal solution is computationally expensive if the number of discrete/categorical attributes is large, an approximate solution is proposed using random projections.

3.4.1 Incorporating Domain Information

Although there is raised awareness about the importance of utilizing domain information, representing it in a general format that can be used by most state-of-the-art algorithms is still an open problem [Witten & Frank, 2005]. Researchers usually focus on one or several types of application-specific domain information. The various ways of utilizing domain information are categorized as following: the choice of attributes or features, generating new examples, incorporating domain knowledge as hints, and incorporating domain knowledge in the learning algorithms [Niyogi *et al.*, 1998].

Use of domain information in the choice of attributes could include adding new attributes that appear in conjunction (or disjunction) with given attributes, or selection of certain attributes satisfying particular criteria. For example, Lustgarten et al. [Lustgarten *et al.*, 2009] used the Empirical Proteomics Ontology Knowledge Bases in a pre-processing step to choose only 5% of candidate biomarkers of disease from high-dimensional proteomic mass

spectra data. The idea of generating new examples with domain information was first proposed by Poggio and Vetter [Poggio & Vetter, 1992]. Later, Niyogi et al. [Niyogi *et al.*, 1998] showed that the method in [Poggio & Vetter, 1992] is mathematically equivalent to a regularization process. Jing and Ng [Jing & Ng, 2010] presented two methods of identifying functional modules from protein-protein interaction (PPI) networks with the aid of Gene Ontology (GO) databases, one of which is to take new protein pairs with high functional relationship extracted from GO and add them into the PPI data. Incorporating domain information as hints has not been explored in biological applications. It was first introduced by Abu-Mostafa [Abu-Mostafa, 1994], where hints were denoted by a set of tests that the target function should satisfy. An adaptive algorithm was also proposed for the resulting constrained optimization.

Incorporating domain information in a learning algorithm has been investigated extensively in the literature. For example, the regularization theory transforms an ill-posed problem into a well-posed problem using prior knowledge of smoothness [Poggio & Girosi, 1990]. Verri and Poggio [Verri & Poggio, 1986] discussed the regularization framework under the context of computer vision. Considering domain knowledge of transform invariance, Simard et al. [Simard *et al.*, 1993] introduced the notion of transformation distance represented as a manifold to substitute for Euclidean distance. Schölkopf et al. [Schölkopf *et al.*, 1998] explored techniques for incorporating transformation invariance in Support Vector Machines (SVM) by constructing appropriate kernel functions. There are a large number of biological applications incorporating domain knowledge via learning algorithms. Ochs reviewed relevant research from the perspective of biological relations among different types of high-throughput data [Ochs, 2010].

Domain information could be perceived of as data extracted from a different view. Therefore, incorporating domain information is related to integration of different data sources [English & Butte, 2007; Berrar *et al.*, 2003]. Altmann et al. [Altmann *et al.*, 2007, 2009] added prediction outcomes from phenotypic models as additional features. English and Butter [En-

glish & Butte, 2007] identified biomarker genes causally associated with obesity from 49 different experiments (microarray, genetics, proteomics and knock-down experiments) with multiple species (human, mouse, and worm), integrated these findings by computing the intersection set, and predicted previously unknown obesity-related genes by the comparison with the standard gene list. Several researchers applied ensemble-learning methods to incorporate learning results from domain information. For instance, Lee and Shatkay [Lee & Shatkay, 2009] ranked potential deleterious effects of single-nucleotide polymorphisms (SNP) by computing the weighted sum of various prediction results from four major biomolecular functions, protein coding, splicing regularization, transcriptional regulation, and post-translational modification, with distinct learning tools.

Domain information could also be treated as constraints in many forms. For instance, Djebbari and Quackenbush [Djebbari & Quackenbush, 2008] deduced prior network structure from the published literature and high-throughput PPI data, and used the deduced seed graph to generate a Bayesian gene-gene interaction network. Similarly, Ulitsky and Shamir [Ulitsky & Shamir, 2009] seeded a graphical model of gene-gene interaction from a PPI database to detect modules of co-expressed genes. In [Jing & Ng, 2010], Gene Ontology information was utilized to construct transitive closure sets from which the PPI network graph could grow. In all these methods, domain information was used to specify constraints on the initial states of a graph.

Domain information could be represented as part of an objective function that needs to be minimized. For example, Tian et al. [Tian *et al.*, 2009] considered the measure of agreement between a proposed hypergraph structure and two domain assumptions, and encoded them by a network-Laplacian constraint and a neighborhood constraint in the penalized objective function. Daemen et al. [Daemen *et al.*, 2008] calculated a kernel from microarray data and another kernel from targeted proteomics domain information, both of which measure the similarity among samples from two angles, and used their sum as the final kernel function to predict the response to cetuximab in rectal cancer patients. Bogojeska et al. [Bogojeska

et al., 2010] predicted the HIV therapy outcomes by setting the model prior parameter from phenotypic domain information. Anjum et al. [Anjum *et al.*, 2009] extracted gene interaction relationships from scientific literature and public databases. Mani et al. [Mani *et al.*, 2008] filtered a gene-gene network by the number of changes in mutual information between gene pairs for lymphoma subtypes.

Domain knowledge has been widely used in Bayesian probability models. Ramakrishnan et al. [Ramakrishnan *et al.*, 2009] computed the Bayesian posterior probability of a gene's presence given not only the gene identification label but also its mRNA concentration. Ucar et al. [Ucar *et al.*, 2009] included ChIP-chip data with motif binding sites, nucleosome occupancy and mRNA expression data within a probabilistic framework for the identification of functional and non-functional DNA binding events with the assumption that different data sources were conditionally independent. In [Werhli & Husmeier, 2008], Werhli and Husmeier measured the similarity between a given network and biological domain knowledge, and by this similarity ratio, the prior distribution of the given network structure is obtained in the form of a Gibbs distribution.

3.4.2 Attribute Selection Metric

A discrete or categorical attribute is viewed as having high potential if it provides a partition that greatly reduces the complexity of the learning task, or in other words, the uncertainty of the classification problem. A hypothesis class, such as the linear function family, is assumed beforehand. Therefore, we quantify the potential using the information gain achieved by a set of optimal classifiers, each of which is built for a sub-problem defined by the discrete or categorical attribute under consideration. Searching for the top ranked attribute with maximum information gain is equivalent to seeking the one with minimum conditional entropy. In a naïve approach, an optimal prediction model is identified by comparing restructured problems using each discrete or categorical attribute. This exhaustive approach is computationally prohibitive when the number of discrete or categorical attributes is large.

We propose to rank attributes using a metric that can be efficiently computed.

In a classification problem, consider a set of l samples $(\mathbf{x}, y)$ from an unknown distribution, $\mathbf{x} \in \mathbb{R}^n$, and y is the class label. In a k-class learning task, y gets a value from $\{1, ..., k\}$; In a binary classification problem, y is either 1 or -1. z represents a discrete or categorical attribute with finite unique values. For simplicity, let's assume z takes values from $\{1, ..., q\}$, which offers a problem partition into q sub-problems, i.e. for all the samples when attribute z takes value i, $i \in 1, ..., q$. Z is the set of all discrete and categorical attributes, $z \in Z$. A hypothesis class M is considered. We will first consider the linear model family. The metric can be generalized to a non-linear hypothesis class using the kernel trick [Vapnik, 1995].

For a binary classification problem, a linear discriminant function is formulated as $f(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + c$, where $\mathbf{w}$ indicates the normal vector of the corresponding hyperplane and c is the offset parameter. For a multi-class task, if the one-vs-one method [Li *et al.*, 2004] is applied, there exists $k(k-1)/2$ linear discriminant functions, each of which separates a pair of classes. Because a categorical attribute z divides the problem into q sub-problems, we define a model m for the whole problem as a set of linear discriminant functions on the q sub-problems: if it is a binary classification problem, m contains q linear discriminant functions; if it is a multi-class problem, m comprises $qk(k-1)/2$ discriminant functions. Model m contains a pair of components $(\mathbf{w}, c)$, where $\mathbf{w}$ is the set of normal vectors of all of the discriminant functions in m, and c contains all of the linear function offset parameters in m.

The most informative attribute under the context discussed above is defined through the following optimization problem:

$$\operatorname*{argmin}_{z \in Z} \inf_{m \in M} H(y|z, m).$$

which is equivalent to

$$\operatorname*{argmin}_{z \in Z} \inf_{(\mathbf{w}, c) \in M} H(y|z, (\mathbf{w}, c)).$$

Note that the conditional entropy used here is fundamentally different from the one

normally applied in decision trees. The traditional conditional entropy $H(y|z)$ refers to the remaining uncertainty of class variable y given that the value of an attribute z is known. The conditional entropy used above is conditional on the information from attribute z and model m. In other words, the proposed method looks one more step ahead than a decision tree about data impurity of sub-problems.

An Approximated Solution

The above optimization problem cannot be solved without knowledge of the probabilistic distribution of data. Sample version solutions may not be useful due to the curse of dimensionality: in high dimension feature spaces, a *finite number* of points may easily be separated by a hypothesis class (an infinitesimal conditional entropy), but the solution is more likely to be overfit than to be a close match to the target function. Taking a different perspective, if a categorical attribute is able to maximally simplify the learning task, the expected impurity value with respect to all possible models within the given hypothesis class should be small. This motivates the following approximation using the expected conditional entropy with respect to a random hyperplane:

$$\underset{z \in Z}{\operatorname{argmin}} E_{\mathbf{w}} \left[\inf_{c} H\left(y|z, (\mathbf{w}, c)\right) \right].$$

The expectation could be estimated by the average over a finite number of trials. Hence, we randomly generate N sets of normal vectors (each set includes q normal vectors for binary-class or $qk(k-1)/2$ for multi-class), search for the corresponding best offset for each normal vector, and calculate the average conditional entropy

$$\underset{z \in Z}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \min_{c_i} H(y|z, (\mathbf{w}_i, c_i)). \tag{3.1}$$

In the i_{th} random projection, $\mathbf{w}_i$ includes all the normal vectors of the linear classifiers, each of which is built on a sub-problem, and c_i does the same for the offsets. According to

the definition of conditional entropy, $H(y|z,(\mathbf{w}_i,c_i))$ in (3.1) is formulated as:

$$\begin{aligned}
&H(y|z,(\mathbf{w}_i,c_i))\\
=&H(y|(z=1,\mathbf{w}_{i1},c_{i1}),...,(z=q,\mathbf{w}_{iq},c_{iq}))\\
=&\sum_{j=1}^{q} p(z=j,\mathbf{w}_{ij},c_{ij})H(y|z=j,\mathbf{w}_{ij},c_{ij})\\
\propto&\sum_{j=1}^{q} p(z=j)H(y|z=j,\mathbf{w}_{ij},c_{ij}).
\end{aligned} \tag{3.2}$$

Probability $p(z=j)$ is approximated by the sub-problem size ratio. The last step of the above derivation is based on the fact that the random projections are independent from the size of the sub-problems.

In a binary classification task, $z=j$ denotes the j_{th} sub-problem, and $(\mathbf{w}_{ij},c_{ij})$ indicates the linear discriminant function of the i_{th} random projection on the j_{th} sub-problem. The discriminant function represented by $(\mathbf{w}_{ij},c_{ij})$ classifies the j_{th} sub-problem into two parts, Ω_{ij}^{+} and Ω_{ij}^{-}:

Ω_{ij}^{+} : { all the samples when attribute z takes value j and satisfying $\mathbf{w}_{ij}^{\intercal}\mathbf{x}+c_{ij}\geq 0$},

Ω_{ij}^{-} : { all the samples when attribute z takes value j and satisfying $\mathbf{w}_{ij}^{\intercal}\mathbf{x}+c_{ij}< 0$}.

$H(y|z=j,\mathbf{w}_{ij},c_{ij})$ in (3.2) quantifies the remaining uncertainty of variable y in the j_{th} sub-problem given the learned partition result $(\Omega_{ij}^{+},\Omega_{ij}^{-})$ defined by the linear discriminant function with parameters $(\mathbf{w}_{ij},c_{ij})$:

$$\begin{aligned}
&H(y|z=j,\mathbf{w}_{ij},c_{ij})\\
=&H(y|(\Omega_{ij}^{+},\Omega_{ij}^{-}))\\
=&-p(\Omega_{ij}^{+})\sum_{y\in\{-1,1\}} p(y|\Omega_{ij}^{+})log(p(y|\Omega_{ij}^{+}))-p(\Omega_{ij}^{-})\sum_{y\in\{-1,1\}} p(y|\Omega_{ij}^{-})log(p(y|\Omega_{ij}^{-})).
\end{aligned} \tag{3.3}$$

In the computation of (3.3), $p(\Omega_{ij}^+) = \frac{|\Omega_{ij}^+|}{|\Omega_{ij}^+|+|\Omega_{ij}^-|}$ and $p(\Omega_{ij}^-) = \frac{|\Omega_{ij}^-|}{|\Omega_{ij}^+|+|\Omega_{ij}^-|}$. $p(y|\Omega_{ij}^+)$ and $p(y|\Omega_{ij}^-)$ are estimated by the proportion of positive/negative samples within Ω_{ij}^+ and Ω_{ij}^-, respectively.

In a multi-class setting, within a sub-problem, instead of two sub-regions (Ω^+, Ω^-), there are q sub-regions $(\Omega^1, ..., \Omega^q)$, each of which is the decision region for a class. All the categorical attributes are ranked according to (3.1).

Extension to Non-linear Models

Our proposed metric could be easily extended to non-linear models using the kernel trick [Vapnik, 1995]. By the dual representation of a linear model, the normal vector is represented as a weighted summation of sample data.

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i.$$

where $\alpha_i \in \mathbb{R}$ is a weight. The linear function is then formulated as:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^\intercal \mathbf{x} + c \\ &= (\sum_{i=1}^{l} \alpha_i \mathbf{x}_i)^\intercal x + c \\ &= \sum_{i=1}^{l} \alpha_i \mathbf{x}_i^\intercal \mathbf{x} + c. \end{aligned}$$

Using the kernel trick, inner product $\mathbf{x}_i^\intercal \mathbf{x}$ can be replaced by a kernel function K. $K(\mathbf{x}_i, \mathbf{x})$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}$ in the reproducing kernel Hilbert space. Therefore, the above linear discriminative function is transformed to,

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + c. \tag{3.4}$$

In our method, given a kernel K, random projections are achieved through α_i.

3.4.3 Experiments and Results

We tested our method on three artificial data sets, three cheminformatics data sets, biological activity data of glycogen synthase kinase-3β inhibitors, cannabinoid receptor subtypes CB1 and CB2 activity data, and CB1/CB2 selectivity data, and two cancer microarray data sets. The random projection was executed 1000 times for each data set.

Three different kernels were applied in this paper: linear, two-degree polynomial and Gaussian. The latter two kernels have one or more parameters. For the two-degree polynomial kernel, we used the default setting as $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^\intercal \mathbf{v})^2$. Choosing a proper parameter γ in the Gaussian kernel $K(\mathbf{u}, \mathbf{v}) = exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$ is not an easy task. This paper focuses on how to select one (or more) categorical or discrete attribute(s) to divide the original problem into multiple simpler sub-problems. Selecting a proper model is not the theme of the work. Therefore, we list three Gaussian kernels using different γ values, 0.01, 1 and 10, to demonstrate that our restructuring process could be extended to non-linear models including the Gaussian kernel.

Many prediction problems have the property of small sample size and high dimensionality, for example, the learning tasks for the three cheminformatics data sets. Simple models under these circumstances are usually preferred. We applied a linear kernel on these three data sets, and analyzed the results from a cheminformaticist's point of view. For the purpose of comparison, two-degree polynomial kernels and Gaussian kernels were also used.

The code was written with Matlab and libsvm package, and can be downloaded from http://cbbg.cs.olemiss.edu/StructureClassifier.zip.

Artificial Data Sets

Three artificial data sets were generated to test our method using both linear and non-linear models. They are shown in Figure 3.1. Each artificial data is generated by four attributes: X_1 and X_2 are continuous attributes, and X_3 and X_4 are categorical attributes. The continuous attributes are uniformly distributed. $X_3 = \{1, 2, 3, 4\}$ denotes four different

smaller square sub-regions. $X_4 = \{1, 2\}$ is a random categorical attribute for the purpose of comparison. In the experiment, we generated 10 sets for Artificial Data 1, 2, and 3, respectively. All 10 sets share the same values of attributes X_1, X_2, and X_3, but X_4 is random. Average results and standard deviations were computed.

The binary class information is coded by two distinct colors. Categorical attribute X_3 provides interesting partitions: the partition in (a) leads to linear classification problems; the partition in (b) and (c) generates nonlinear problems that can be solved using techniques such as SVM with a polynomial kernel. Note that the original problem in (a) is not linear. The original problems in (b) and (c) are nonlinear, and not solvable using a polynomial kernel of degree 2.

Next, we assume linear classifiers in (a) and SVM with a polynomial kernel of degree 2 in (b) and (c). From Tables 3.1, 3.2, and 3.3, we see that the averaged estimated conditional entropy of X_3 is always smaller than that of X_4. Hence X_3 is selected to restructure the problem. Next, we build both linear classifier and degree-2 polynomial SVM models on the original problem (we call it the baseline method), and linear and degree-2 polynomial models on the restructured problems introduced by X_3. Significant improvements in both cross-validation (CV) accuracy and test accuracy are achieved using the partitions provided by X_3. For comparison purposes, models were built on the restructured problem produced by X_4. X_3 outperforms X_4 with a comfortable margin. There is no significant improvement using X_4 than the baseline approaches.

Table 3.1. Experimental Results of Artificial Data 1 (Fig 3.1.(a)) with Linear Model.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	59.6000 ± 3.2042	64.7750 ± 4.0285
X_3	0.7860 ± 0.0044	99.5750 ± 0.2058	96.8607 ± 0.8680
X_4	0.9001 ± 0.0035	61.1250 ± 1.7490	60.4881 ± 2.8090

Table 3.2. Experimental Results of Artificial Data 2 (Fig 3.1.(b)) Using Two-degree Polynomial Kernel.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	71.9750 ± 6.4737	71.0500 ± 7.9292
X_3	0.8980 ± 0.0061	94.1000 ± 0.8350	94.3071 ± 0.9204
X_4	0.9514 ± 0.0043	73.4000 ± 1.4443	73.8682 ± 2.8535

Table 3.3. Experimental Results of Artificial Data 3 (Fig 3.1.(c)) Using Two-degree Polynomial Kernel.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	73.1750 ± 5.7772	71.6025 ± 8.3302
X_3	0.8455 ± 0.0059	96.5500 ± 0.8644	95.3658 ± 1.0224
X_4	0.9328 ± 0.0032	72.8750 ± 1.5601	71.7689 ± 3.5528

Biological Activity Prediction of Glycogen Synthase Kinase-3β Inhibitors

In the first dataset, data samples (IC50) were collected from several publications, with a range from subnanomolar to hundred micromolar. The biological activities have been discretized as binary values: highly active and weakly active, with a cut-off value of 100 nM. The aim is to predict biological activity based on physicochemical properties and other molecular descriptors of the compounds calculated using DragonX software [1]. This data set was divided into 548 training samples and 183 test samples. The attribute set size is 3225, among which 682 are categorical attributes [2].

Using a linear kernel, we ranked the categorical attributes based on their estimated conditional entropies. The top 31 attributes (with smallest estimated conditional entropy) were viewed as candidate attributes for problem partition. We restructured the learning problem according to these candidate attributes separately, and built linear models for each partition. Figure 3.3 shows the experimental results. Among the 31 attributes, there are 17 categorical attributes whose performance beat the baseline approach in terms of both cross-validation accuracy and test accuracy. The detailed performance values and the names of

[1] Which can be found at http://www.talete.mi.it/

[2] Some discrete attributes contain a large number of values. For a fixed sized training set, some regions generated by a partition using such attributes may contain a very small number of samples (many times 1 or 2), and hence are not suitable for training a classifier. So we filtered out attributes with more than 10 unique values.

the attributes are provided in Table 3.4. Compared with linear kernels, the ranking orders of these attributes by two-degree polynomial and Gaussian kernels and their corresponding cross-validation and test accuracies are provided in Table 3.5 as well. For Gaussian kernels, we notice performance improvement for most of the selected attributes under all three tested γ values. The highest performance was achieved when the Bioassay Protocol attribute was selected to restructure the problem. This attribute records the different protocols used during the cheminformatics experiment, and also indicates distinct chemotypes.

The highest cross-validation performance attribute, nCIR, belongs to the constitutional descriptors. Constitutional descriptors reflect the chemical composition of a compound without the structural information of the connectivity and the geometry. nCIR means the number of circuits, which includes both rings and the larger loop around two or more rings. For instance, naphthalene contains 2 rings and 3 circuits. This attribute could easily distinguish ring-containing structures and linear structures. Many attributes selected have names starting with "F0". They are from the 2D frequency fingerprints, which define the frequency of specific atom pairs at different topological distances from 1 to 10. Among all of the 2D frequency fingerprints, the atom pair "N-N" appeared multiple times. The frequency of this atom pair at different topological distances from 2 to 4 could be used to separate the dataset. Another important atom pair is "N-O", which also appeared multiple times in the list. Both atom pairs contain the nitrogen atom which is highly common in the kinase inhibitor structures, since it plays a key role in the hydrogen bond interactions between the inhibitor and the kinase. Another atom-centered fragment attribute is H-049, which means the atom H attached to any of $C^3(sp^3)$ / $C^2(sp^2)$ / $C^3(sp^2)$ / $C^3(sp)$ groups. The superscripts on the carbons stand for the formal oxidation number and the contents in the parentheses stand for the hybridization state. The hydrogen in an H-049 fragment has negative atomic hydrophobicity and low molecular refractivity [Viswanadhan *et al.*, 1989], so they are less hydrophobic and more hydrophilic. H-049 could be used to separate the

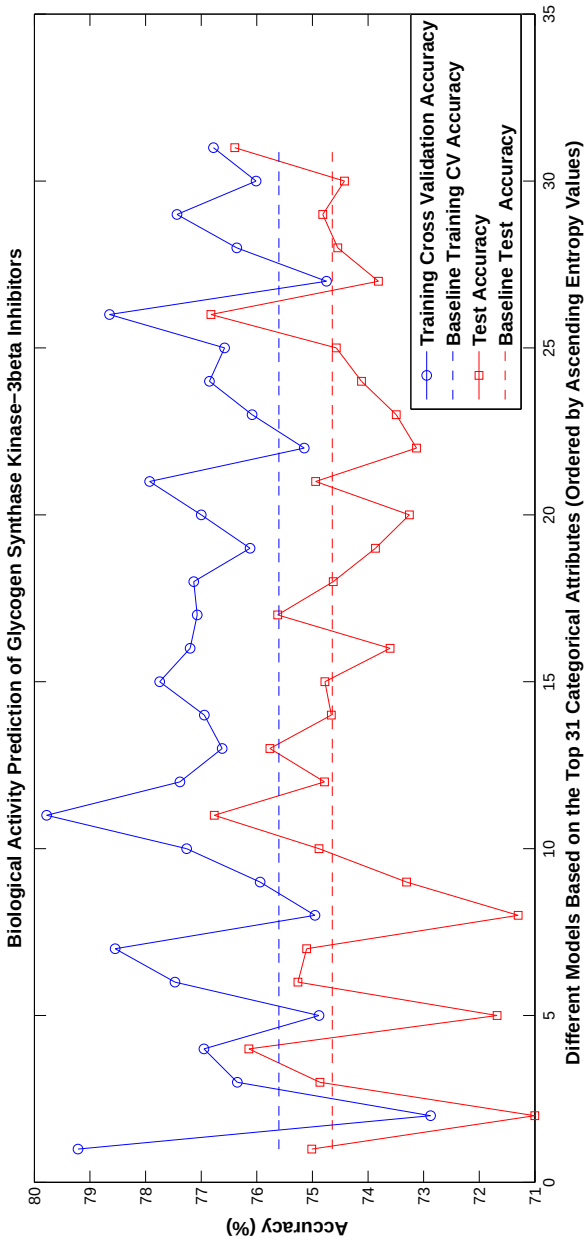

Figure 3.3. Experimental Results for Biological Activity Prediction of Glycogen Synthase Kinase-3β Inhibitors.

database because the kinase inhibitors are usually hydrophilic in order to bind to the protein in the ATP-binding pocket.

Table 3.4. Learning Performance for the Selected Categorical Attributes in Biological Activity Data of Glycogen Synthase Kinase-3β Inhibitors Using Linear Kernel.

	Entropy list order	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	75.60	74.64
nCIR	1	79.21	75.01
F06[N-O]	2	76.35	74.86
H-049	3	76.95	76.14
nN	7	77.38	74.78
F04[N-N]	8	78.55	75.10
Bioassay Protocol	9	79.78	76.76
nHDon	12	77.26	74.88
H-050	13	77.26	74.88
nDB	15	77.74	74.78
F07[C-Br]	16	76.62	75.76
F02[N-O]	22	77.07	75.62
N-075	23	78.65	76.83
F06[C-Br]	25	76.94	74.66
F02[N-N]	26	77.93	74.92
N-074	30	76.78	76.39
F03[N-N]	31	77.44	74.81

Cannabinoid Receptor Subtypes CB1 and CB2 Activity and Selectivity Prediction

These data sets are for cannabinoid receptor subtypes CB1 and CB2. They were also computed from DragonX software, and have 3225 attributes. The second data set is to predict activity and was divided into 645 training samples and 275 test samples. It contains 683 categorical attributes. The third set is to predict selectivity of binding to CB1 vs. CB2 and includes 405 training samples, 135 test samples, and 628 categorical attributes. The experimental results are shown in Figures 3.4 and 3.5, respectively. We ordered the categorical attributes based on their conditional entropy values in ascending order. Note that the model based on the first attribute always performed better than the baseline approach.

The classes and descriptions for the attributes that result in better performance than the baseline approach are listed in Tables 3.6 and 3.8. The learning performance comparison

Table 3.5. Performance Comparison for the Selected Categorical Attributes in Biological Activity Data of Glycogen Synthase Kinase-3β Inhibitors Using Two-degree Polynomial Kernel and Gaussian Kernels.

	Entropy list order				Training CV Accuracy(%)				Test Accuracy(%)			
	Poly	Gausssian			Poly	Gausssian (γ)			Poly	Gausssian (γ)		
		0.01	1	10		0.01	1	10		0.01	1	10
Baseline	–	–	–	–	76.23	73.10	62.74	59.42	74.26	70.69	60.58	57.44
nCIR	3	2	1	1	78.84	75.41	64.48	60.15	74.55	71.23	61.26	58.02
F06[N-O]	2	1	2	2	77.62	73.23	63.34	60.23	73.28	70.49	60.87	56.95
H-049	4	5	4	4	79.75	74.69	65.18	61.03	75.14	71.87	62.76	57.26
nN	1	6	6	7	79.24	74.87	64.77	60.49	75.23	71.04	62.38	57.15
F04[N-N]	7	3	5	6	78.32	74.14	63.14	60.63	74.16	70.02	61.79	57.69
Bioassay Protocol	8	7	3	5	79.15	75.54	65.15	62.25	76.03	72.87	63.76	59.34
nHDon	11	19	18	19	77.63	74.18	63.05	60.02	75.12	71.17	60.34	57.28
H-050	21	7	7	9	76.95	73.57	63.72	60.35	74.34	71.09	59.28	56.94
nDB	13	24	21	25	75.37	73.89	62.83	59.25	73.22	70.18	60.47	56.74
F07[C-Br]	17	12	15	16	77.25	74.58	63.04	60.42	73.96	71.65	61.07	58.15
F02[N-O]	25	16	13	15	76.14	73.87	62.95	58.72	72.87	70.66	60.84	57.35
N-075	20	17	17	21	78.06	74.92	63.74	60.87	75.64	71.29	62.88	59.04
F06[C-Br]	27	26	25	23	75.44	72.05	61.43	58.28	72.76	69.96	60.03	55.74
F02[N-N]	33	30	26	32	77.83	74.15	63.82	60.96	74.56	70.75	61.44	59.45
N-074	29	35	33	34	76.54	73.47	63.95	60.42	74.75	71.03	60.58	57.96
F03[N-N]	36	31	34	37	75.69	74.26	62.65	59.35	73.48	70.33	59.87	57.28

with other non-linear kernels are shown in Table 3.7 and 3.9 respectively. For the CB activity, among the eight features, six of them (F01[N-O], N-076, nArNO2, B01[N-O], N-073 and nN(CO)2) involve nitrogen. This clearly suggests that nitrogen plays a significant role in classifying the active CB ligands. The input data showed that the values of N-076 and nArNO2 for all the active compounds are 0. Hence, it is very likely that any compound with the Ar-NO2 / R–N(–R)–O / RO-NO moiety or a nitro group may not be active. In addition, the majority of the active compounds have F01[N-O] and nN(CO)2 values of 0. Hence, the lack of a N-O or an imide moiety is perhaps a common feature of active CB ligands. Furthermore, the N-073 feature is distributed between 0 and 2 in the active compounds. Hence, the nitrogen atom in the active compounds, if it exists, may appear in the form of Ar2NH / Ar3N / Ar2N-Al / R..N..R. Its role may include acting as a hydrogen bond acceptor, or affecting the polarity of the molecule, which may facilitate the ligand binding. For the CB selectivity problem, two features (nDB and nCconj) involve

Table 3.6. Learning Performance for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Activity Data Using Linear Model.

	Entropy list order	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	85.43	84.36
F01[N-O]	1	86.20	84.37
N-076	4	86.51	85.12
nArNO2	5	86.36	85.07
nCconj	15	87.13	86.37
C-034	16	86.82	86.04
B01[N-O]	17	86.82	84.46
N-073	18	85.89	85.81
nN(CO)2	19	86.05	84.49

double bonds. Both of these address the non-aromatic C=C double bond and the values are primarily distributed between 0 - 6 and 0 - 2, respectively, in the selective compounds. The role of this bond, if it exists, is perhaps to form hydrophobic interactions with the proteins. It is also interesting to note that the nCconj attribute leads to the best test accuracy for both the activity and selectivity datasets. The descriptions of selected categorical attributes can be viewed in Table 3.10 and 3.11.

Leukemia Gene Data

The two leukemia gene data sets used are defined in Yeoh et al. [Yeoh *et al.*, 2002] and Golub et al. [Golub *et al.*, 1999], respectively. We applied a linear classifier, SVM with a two-degree polynomial kernel and Gaussian kernels on these two data sets.

Yeoh's data [3] comprises gene expression data and two additional categorical attributes, Subtype and Protocol. Subtype indicates specific genetic subtypes of Acute lymphoblastic leukemia (ALL), and Protocol means distinct therapies. The entire set contains 201 continuous complete remission (CCR) samples and 32 relapse cases (including 27 Heme relapses and 5 additional relapses). We randomly split the data into training and test sets with 174 and 59 samples, respectively. The original data contains 12627 attributes, which is almost two orders of magnitude larger than the training set size. We used the 58 preselected attributes

[3]Which can be accessed from http://www.stjuderesearch.org/site/data/ALL1

provided in the original paper and two additional categorical attributes to predict prognosis. Tables 3.12, 3.13, and 3.14 show the experimental results using linear, two-degree polynomial and Gaussian kernels, respectively. The subtype categorical attribute has smaller estimated conditional entropy than Protocol, and is thus selected to divide the problem. The learning performances from both the linear model and SVM demonstrate that it is the right choice.

Golub's data set [4] includes gene expression data and four categorical attributes, BM/PM, T/B-cell, FAB, and Gender. A random split was used to separate the whole data set into 54 training samples and 18 test samples. Correlation-based Feature Selection [Hall, 2000] was executed beforehand to decrease the attribute dimension from 7133 to 45. The 45 attributes include two categorical attributes, T/B-cell and FAB. FAB denotes one of the most commonly used classification schemata for Acute Myeloid Leukemia (AML). BM/PM and Gender had been deleted during the feature selection process. The goal is to predict ALL or AML. From Tables 3.15, 3.16 and 3.17, we can see that both T/B-cell and FAB have very small conditional entropy values (it may be because it is an easy learning problem). The T/B-cell categorical attribute was selected to partition the problem.

Discussions

For choosing a proper partition attribute, we could either select the one with the smallest conditional entropy, or the one with the highest training cross-validation accuracy among multiple candidates. The first strategy worked well for all the data sets — while it may not provide the best performing partition, it always outperformed the baseline. The second strategy yielded the best answer for most cases — glycogen synthase kinase-3β inhibitors data is an example — however, it failed on cannabinoid receptor subtypes CB1 and CB2 activity data.

In addition to simplifying the learning problem, the selected categorical attribute may provide additional perspective in unveiling hidden biological information. For example, the

[4]Which can be accessed from http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

attributes chosen from cannabinoid receptor subtypes CB1 and CB2 data sets supply useful information for compound design.

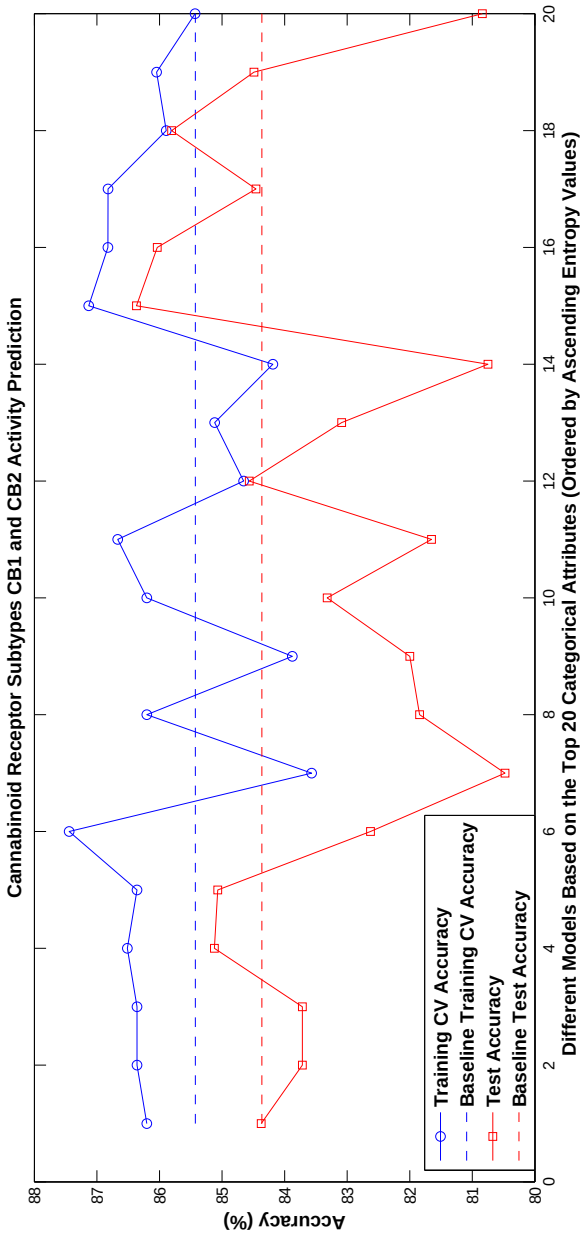

Figure 3.4. Experimental Results for Cannabinoid Receptor Subtypes CB1 and CB2 Activity Prediction.

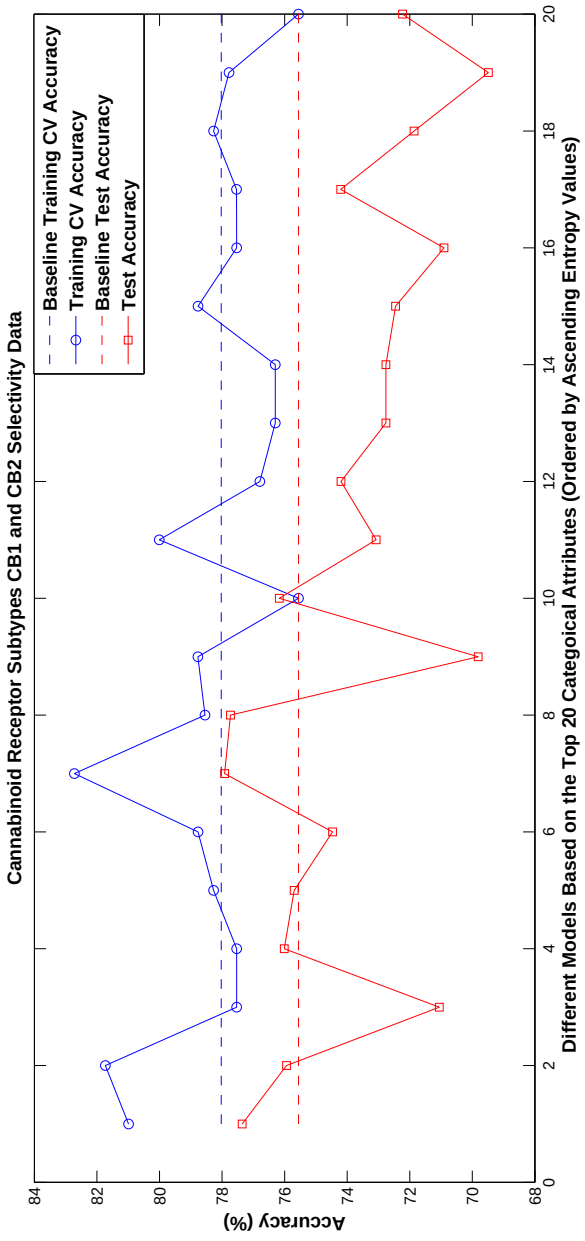

Figure 3.5. Experimental Results for Cannabinoid Receptor Subtypes CB1 and CB2 Selectivity Prediction.

Table 3.7. Performance Comparison for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Activity Data Using Two-degree Polynomial Model and Gaussian Models.

	Entropy list order				Training CV Accuracy(%)				Test Accuracy(%)			
	Poly	Gausssian			Poly	Gausssian			Poly	Gausssian		
		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$
Baseline	–	–	–	–	86.51	75.34	65.21	66.76	85.58	74.35	65.79	65.61
F01[N-O]	1	2	1	1	85.15	76.12	66.16	66.44	85.44	76.14	65.65	66.15
N-076	4	5	4	4	87.50	77.05	66.98	67.33	86.12	76.89	66.34	66.79
nArNO2	6	7	5	5	86.82	75.14	66.78	66.58	85.27	84.35	76.34	64.96
nCconj	16	14	10	12	86.61	77.12	67.03	66.79	83.31	76.72	63.77	65.74
C-034	17	16	11	17	85.98	76.38	66.44	65.89	85.69	75.28	64.59	65.88
B01[N-O]	20	19	19	18	87.21	76.38	66.38	66.66	86.72	76.37	66.29	65.62
N-073	21	20	21	21	84.96	74.79	65.02	65.26	84.15	75.34	64.45	63.71
nN(CO)2	23	24	27	25	86.77	73.72	66.05	64.37	85.78	73.22	63.76	62.96

Table 3.8. Learning Performance for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Selectivity Data Using Linear Model.

	Entropy list order	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	78.02	75.56
O-058	1	80.99	77.35
nDB	2	81.73	75.94
F06[C-Cl]	5	78.27	75.63
nCconj	7	82.72	77.92
C-026	8	78.55	77.73

Table 3.9. Performance Comparison for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Selectivity Data Using Linear Model.

	Entropy list order				Training CV Accuracy(%)				Test Accuracy(%)			
	Poly	Gausssian			Poly	Gausssian			Poly	Gausssian		
		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$		$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$
Baseline	–	–	–	–	76.04	67.15	57.28	57.67	74.89	65.12	54.84	53.33
O-058	2	1	2	2	79.92	70.12	60.34	79.12	65.96	56.34	56.02	53.21
nDB	3	3	4	4	80.05	71.34	61.22	80.36	76.32	67.78	57.67	55.32
F06[C-Cl]	7	8	7	8	79.73	69.96	58.27	79.12	75.12	63.29	54.79	53.29
Cconj	6	7	8	7	78.75	67.54	57.65	77.64	76.07	65.96	55.36	54.34
C-026	9	10	9	11	77.96	68.32	57.34	58.12	75.48	65.32	54.96	53.69

Table 3.10. Descriptions for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Activity Data.

	Attribute Class	Description
F01[N-O]	2D frequency fingerprints	frequency of N-O at topological distance 1
N-076	Atom-centered fragments	Ar-NO2 / R–N(–R)–O / RO-NO
nArNO2	Functional group counts	number of nitro groups (aromatic)
nCconj	Functional group counts	number of non-aromatic conjugated C(*sp2*)
C-034	Atom-centered fragments	R–CR..X
B01[N-O]	2D binary fingerprints	presence/absence of N-O at topological distance 1
N-073	Atom-centered fragments	Ar_2NH / Ar_3N / Ar_2N-Al / R..N..R
nN(CO)2	Functional group counts	number of imides (thio-)-C(=Y1)-N(Y)-C(=Y1)- Y=H or C, Y1= O or S

R represents any group linked through carbon; X represents any electronegative atom (O, N, S, P, Se, halogens); Al and Ar represent aliphatic and aromatic groups, respectively; = represents a double bond; – represents an aromatic bond as in benzene or delocalized bonds such as the N-O bond in a nitro group; .. represents aromatic single bonds as in the C-N bond in pyrrole.

Table 3.11. Descriptions for the Selected Categorical Attributes in Cannabinoid Receptor Subtypes CB1 and CB2 Selectivity Data.

	Attribute Class	Description
O-058	Atom-centered fragments	=O
nDB	Constitutional descriptors	number of double bonds
F06[C-Cl]	2D frequency fingerprints	frequency of C-Cl at topological distance 6
nCconj	Functional group counts	number of non-aromatic conjugated C(*sp2*)
C-026	Atom-centered fragments	R–CX..R

R represents any group linked through carbon; X represents any electronegative atom (O, N, S, P, Se, halogens); Al and Ar represent aliphatic and aromatic groups, respectively; = represents a double bond; – represents an aromatic bond as in benzene or delocalized bonds such as the N-O bond in a nitro group; .. represents aromatic single bonds as in the C-N bond in pyrrole.

Table 3.12. Experimental Results of ALL Prognosis Prediction Using Preselected Attribute Sets and Linear Model.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	85.06	89.83
Subtype	0.3659	89.08	92.20
Protocol	0.5616	85.06	89.96

Table 3.13. Experimental Results of ALL Prognosis Prediction Using Preselected Attribute Sets and Two-degree Polynomial Kernel.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	85.06	89.83
Subtype	0.3638	89.08	92.20
Protocol	0.5630	86.78	87.46

Table 3.14. Experimental Results of ALL Prognosis Prediction Using Preselected Attribute Sets and Gaussian Kernel.

	Conditional Entropy			Training CV Accuracy(%)			Test Accuracy(%)		
	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$
Baseline	–	–	–	85.06	85.06	85.06	89.83	89.83	89.83
Subtype	0.5656	0.5662	0.5662	88.51	88.51	88.51	92.20	92.20	92.20
Protocol	0.3829	0.3835	0.3840	85.06	85.06	85.06	89.96	89.96	89.96

Table 3.15. Experimental Results of ALL/AML Prediction Using Attributes Selected by CFS and Linear Model.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	100.00	99.50
T/B-cell	7.1491e-16	100.00	100.00
FAB	1.1666e-15	100.00	99.70

Table 3.16. Experimental Results of ALL/AML Prediction Using Attributes Selected by CFS and Two-degree Polynomial Kernel.

	Conditional Entropy	Training CV Accuracy(%)	Test Accuracy(%)
Baseline	–	100.00	94.44
T/B-cell	7.1491e-16	100.00	100.00
FAB	1.1666e-15	100.00	100.00

Table 3.17. Experimental Results of ALL/AML Prediction Using Attributes Selected by CFS and Gaussian Kernel.

	Conditional Entropy			Training CV Accuracy(%)			Test Accuracy(%)		
	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$	$\gamma = 0.01$	$\gamma = 1$	$\gamma = 10$
Baseline	–	–	–	68.52	64.81	64.81	66.67	66.67	66.67
Subtype	7.1491e-16	7.1491e-16	7.1491e-16	100.00	100.00	100.00	100.00	100.00	100.00
Protocol	1.1666e-15	1.1666e-15	1.1666e-15	100.00	100.00	100.00	100.00	100.00	100.00

CHAPTER 4

CLASSIFICATION USING TOP SCORING FEATURE PAIRS

"High dimensionality, low sample size" problems, for example, biomedical prediction problems along with high-throughput data generation technology, make the use of complex learning algorithms almost impossible. This dilemma in the statistical analysis of data such as microarray data is well documented in the literature and has been mentioned in previous sections. Another limitation of current methods for classifying gene expression profiles is the "black box" diemma [Geman *et al.*, 2004]. Standard methods in statistical learning and pattern recognition are routinely applied, for instance, decision tree, artificial neural network, and support vector machine. Normally, those predictions are based on nonlinear functions of many expression values. For most biologists, these techniques were developed by researchers in other fields, i.e. computer science, where transparency of the prediction rules is usually not a criterion for success. Whereas in problems such as breast cancer prognosis, scientists are more interested in discovering the cause-effect relationship in disease development. For example, the cause-effect relationship between cancer and a small number of genes is feasible, whereas it is inefficient to have all genes checked. If we would like to go beyond the identification of a mere list of biomarkers, we need to step away from standard statistical or machine learning methods, because it is difficult to extract biologically relevant results from complex models.

Geman et al. [Geman *et al.*, 2004, 2008] has addressed the two problems, small samples

and lack of interpretability, using a method of pairwise feature comparison. They coined the term *Top Scoring Pairs* (TSP) to indicate the algorithm.

TSP builds classifier that only depends on comparisons among selected pairs of features. The classifier is rank-based, and therefore, is invariant to most of the transformations involved in preprocessing and normalization. Every feature pair determines a binary classifier, and the final decision could be made by majority voting.

4.1 Learning the k-TSP Classifier

The Top Scoring Pairs algorithm has been used in biological applications such as classification of human diseases. It discriminates between binary phenotypic states based on one or more transcriptional measurements. TSP evaluates the expression values of all possible pairs of genes in a microarray probe set and chooses gene pairs in which the ordering of expression is most likely to reverse from one phenotype to the other. The prediction of a TSP classifier is only based on the observed ordering of a gene pair or several gene pairs, and there is no parameters needed to be tuned during the training process or assumed beforehand. Such simplicity enables the learning process to generate statistically significant classifiers with a comparatively small number of training instances while avoiding the problem of overfitting. k-TSP indicates that k different top scoring pairs are aggregated by a voting procedure to obtain a combinatoric classifier.

Assume a profile vector consists of d features and l samples $\mathbf{x}_1, ..., \mathbf{x}_l$. The i-th feature, $i \in \{1, ..., d\}$, from the n-th sample is denoted by $x_{i,n}$. Let $(y_1, ..., y_l)$ be the vector of class labels. For simplicity, a binary classification is assumed. For instance, $Y = 1$ refers to positive class and $Y = -1$ indicates negative class. The d features are ordered in descending order within each sample. Let $R_{i,n}$ denote the rank of i-th feature in the n-th observation.

k-TSP exploits a significant difference in the probability of the event $\{R_i < R_j\}$ across all samples from positive class to negative class. The event $\{R_i < R_j\}$ means the rank of feature i is less than the rank of feature j, and is equivalent to the event that the value of

feature i is smaller than that of feature j. The probability of observing the occurrence of such event in each class is computed as

$$p_{ij}(m) = Prob(R_i < R_j | Y = m) = \frac{\sum_{\mathbf{x}_n \in C_m} |\{R_{i,n} < R_{j,n}\}|}{|C_m|}, m = \{1, -1\}, \tag{4.1}$$

where C_m denotes the set of samples in class m and $|C_m|$ is the number of samples in class m. And $|\{R_{i,n} < R_{j,n}\}|$ indicates the occurrence of observing the rank of feature i is less than the rank of feature j in the n-th observation, and it is 1 when the event $\{R_{i,n} < R_{j,n}\}$ occurs or 0 otherwise.

Let δ_{ij} denote the difference score of gene pair (i, j).

$$\delta_{ij} = |p_{ij}(1) - p_{ij}(-1)|. \tag{4.2}$$

We could compute the difference score for every pair of features $i, j \in \{1, ..., d\}, i \neq j$. And the top k pairs with maximum δ values are selected and are viewed as most informative for classification. The top scoring feature (gene) pairs could be viewed as biomarkers that are associated with susceptibility to disease or a certain target label, and can be used to create genetic networks or pathways of the organism being studied. Also, the classifier built on top of these genes is easily understandable, and hence, it is adequate to suggest an assumption that the increasing or decreasing of certain gene expression values indicates a high risk of disease.

Each of the selected pairs (i, j) defines a classifier. Suppose $p_{ij}(1) > p_{ij}(-1)$. Then, given an observation $\mathbf{x}$, the classifier $f_{ij}(\mathbf{x})$ based on this pair (i, j) is

$$f_{ij}(\mathbf{x}) = \begin{cases} 1, & R_i < R_j, \\ -1, & \text{otherwise.} \end{cases} \tag{4.3}$$

On the other hand, if $p_{ij}(-1) \geq p_{ij}(1)$, then the decision rule is reversed.

For k top scoring gene pairs $(i_1, j_1), ..., (i_k, j_k)$, the discriminant function is defined as

$$f(\mathbf{x}) = \text{sgn}(\sum_{t=1}^{k} f_{i_t,j_t}(\mathbf{x})).$$

4.2 TSP for Time Series Data

Time series data are widely generated in the fields of signal processing, econometrics and mathematical finance. In bioinformatics area, as the capture and analysis of single time microarray expression data becomes routine, researchers and investigators have started examining time series expression data to investigate dynamic disease diagnosis, complex gene regulation schemes and metabolic pathways. Different from a static expression experiment where a snapshot of the expression of genes in various samples is measured, in a time series expression experiment, a temporal process is measured. Time series data have a natural temporal ordering, and exhibit a strong correlation between successive time stamps.

Most time series microarray experiments focused on gene clustering. The challenge is the measurement of similarity between two gene temporal sequences. Among many methods, correlation coefficient-based similarity measures have been widely used. Other approaches have been developed as well. For example, Ramoni et al. [Ramoni *et al.*, 2002] studied clustering time series expression data based on their dynamics which are represented as regressive equations. And Schliep et al. [Schliep *et al.*, 2003] presented a hidden markov model based clustering algorithm for time series expression data. Full reviews of challenges and approaches can be found in [Filkov *et al.*, 2002; Bar-Joseph, 2004; Erdal *et al.*, 2004]. Though we research on instance classification not gene clustering here, the similarity measure definitions can be borrowed to help solve our problems. In the following subsection of *TSP Based on Correlation Coefficient*, correlation coefficient is applied as a measure of pattern difference in a gene pair. And in subsection *Dynamic Time Warping with Nearest Neighbor* some approaches introduced in gene clustering are utilized as distance measures for the purpose of seeking nearest neighbors, even though the similarity measure in gene clustering

is executed between two genes and measures in instance classification are computed for a single gene from different instances.

Predicting instance targets in time series microarray experiments has been addressed in the literatures. Instead of treating expression changes over time points as separate observations, they utilized the information on time dependency of gene expression changes. For example, Parker and Wen [Parker & Wen, 2009] used functional data analysis where the predictor is treated as a set of samples from a basis function rather than simply as a feature vector. Borgwardt et al. [Borgwardt *et al.*, 2006] modeled time series microarray data as a partially observed discrete linear time invariant model. In both works, the whole set of genes (features) was used, and complex computations such as functional principal component analysis or singular value decomposition were involved, which makes the prediction model complicated and lack interpretability.

Inspired by the idea of using very few genes and simple computations, we extend the traditional TSP algorithm and propose classification methods for time series data that only depends on a few pairs of features.

4.2.1 Time Series Microarray Experiment

The time series data used here are earthworm microarray data. The purpose of the biological experimental is to assess the neurotoxic effects of chemicals on non-target organisms. Two chemical compounds are considered. RDX, an explosive compound, and carbaryl, a carbamate insecticide, have shown reversible neurotoxicity on the earthworm *Eisenia fetida*. Gong et al. [Gong *et al.*, 2011] have conducted a time-series gene expression study where earthworms were exposed to a sublethal concentrations of RDX or carbaryl for 6 days, followed by a 7-day recovery to elucidate mechanisms underlying reversible neurotoxicity. Worms sampled at 31 different time points (4 during acclimation, 13 during exposure and 14 during recovery) were measured with 44K gene expression endpoints. They analyzed the time series expression data and derived a collective total of over 6000 differentially expressed

genes.

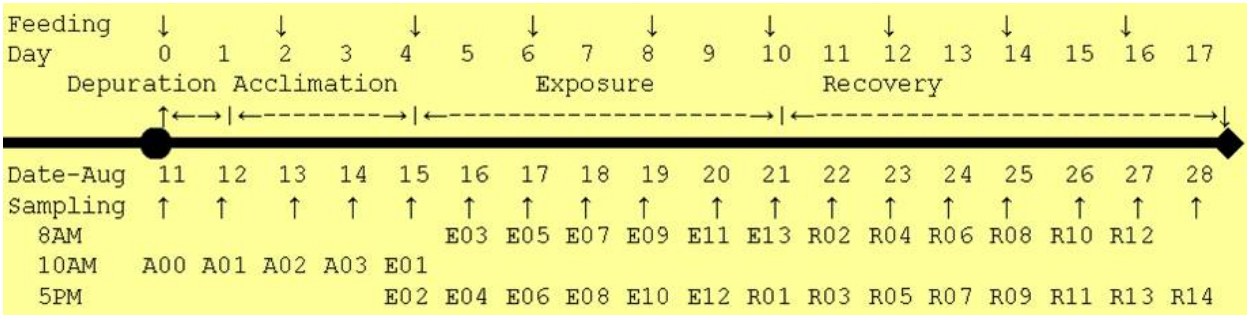

Figure 4.1. Time Series Experimental Design

Figure 4.1 shows the design of this experiment. It last 18 days and consisted of three periods, acclimation (including depuration), exposure, and recovery. The earthworms were sampled once a day during acclimation except the last day. And during exposure and recovery periods, the sampling frequency was fixed at twice a day with the last day in recovery as an exception. Therefore, 4 time points were generated during the acclimation period, 13 during exposure and 14 during recovery. To evaluate the toxicity effects of two chemical compounds, the earthworms were divided into 3 different groups — control, Carbaryl treated and RDX treated. For each group, at a certain time point, 5 to 8 earthworm replicates were sampled. And totally 437 earthworm microarray profiles were obtained. Strictly speaking, this earthworm experiment differs a little from the standard time series microarray experiment where one living organism is measured and sampled at various time points. But in this experiment, an earthworm is dried and grounded to obtain its gene profile. Therefore, the microarray data on different time points were read from different earthworms. However, the experiment was well designed and carried out in a strict way so that such individual deviation could be ignored.

The goal is to use earthworm time series microarray data to predict a period/compound (for example, RDX-exposure) combination, select most informative genes, and be able to use the derived patterns of these genes to interpret chemical neurotoxic effects on earthworms.

4.2.2 Methods and Results

Inspired by the Top Scoring Pair algorithm, we propose several classification approaches that use one or several gene pairs to predict target labels. Different from simple value comparison in static data, the key issue of time series data is how to incorporate temporal information into the TSP framework.

Assume time series data set $\mathbf{x}_1, ..., \mathbf{x}_l$ consists of d features, l samples, and z time points. The i-th feature, $i \in \{1, ..., d\}$, from the n-th sample, denoted as $\mathbf{x}_{i,n}$, comprises z measurements each of which is denoted by x_{i,n_t}, $t \in 1, ..., z$. And $(y_1, .., y_l)$ is the vector of class labels. The sample profile and its class label are regarded as random variables, denoted by $\mathbf{X}$ and Y respectively. And $\mathbf{X}_i$ represents the i-th feature variable.

TSP Based on Average Comparison

The most basic idea is to generalize the time sequence values into one single number, and then the traditional TSP algorithm could be applied. A simple method is to get an average. We first compute the average gene expression values and apply the TSP algorithm described previously in Section 4.1. The only difference is the event $\{R_i < R_j\}$ means the average value of feature i is smaller than that of feature j.

In this approach, the time sequence information is ignored, and samples of different time points are treated equally. The classifier only depends on the average expression comparisons among selected pairs of genes.

TSP Based on Trend

In order to incorporate the temporal information, we consider the variations between every adjacent part of time points. And such variation is represented as a trend. A trend is defined as an increase (+) or a decrease (-) between a pair of adjacent time points. Equality is treated as an increase. A trend pattern q of a sequence with z time points includes $z - 1$ increases and/or decreases, $q \in Q$ where Q is the whole set of all trend patterns.

Instead of considering the ordering difference of a pair of features, the changing of trend pattern is taken into account. If for a given class, a feature exhibits a certain trend pattern while another feature does not, and such observation is unlikely to happen in the other class, the classification will be completed using these two features and the trend pattern. The metric of measuring the occurrence of such an event in each class is computed as

$$\begin{aligned} P_{ij}^{q}(m) &= Prob(\mathbf{X}_i \sim q | Y = m) - Prob(\mathbf{X}_j \sim q | Y = m) \\ &= \frac{\sum_{\mathbf{x}_n \in C_m} |\{\mathbf{x}_{i,n} \sim q\}| - \sum_{\mathbf{x}_n \in C_m} |\{\mathbf{x}_{j,n} \sim q\}|}{|C_m|}. \end{aligned} \tag{4.4}$$

where $m = \{1, -1\}$, C_m denotes the set of samples in class m and $|C_m|$ is the number of samples in class m. Symbol $\sim$ means following a certain trend pattern. $|\{\mathbf{x}_{j,n} \sim q\}|$ indicates the occurrence of observing the i-th feature time sequence in the n-th sample follows the trend pattern q, and has value 1 when such event occurs or 0 otherwise. Note that, $P_{ij}^{q}(m)$ could have negative value when the trend pattern q is more prominent in the j-th feature than in the i-th. It is not a probability, but still serves as a measure of trend pattern change.

Let δ_{ij}^{q} denote the difference score of gene pair (i, j) for a trend pattern q.

$$\delta_{ij}^{q} = |P_{ij}^{q}(1) - P_{ij}^{q}(-1)|.$$

The feature pair is selected based on the maximum difference score among all the trend patterns. Let δ_{ij} denotes this score.

$$\delta_{ij} = \max_{q} \delta_{ij}^{q}.$$

Every feature pair will be evaluated based on their δ value. Pairs of features with high values are viewed as most informative for classification. And the top k pairs with maximum δ values are chosen. The corresponding pattern q for each top pair is also recorded.

With the i-th and j-th features being chosen with trend pattern q, we suppose $P_{ij}^{q}(1) > P_{ij}^{q}(-1)$. Then, given an observation $\mathbf{x}$, the classifier based on the pair (i, j) and trend pattern q is

$$f_{ij}(\mathbf{x}) = \begin{cases} 1, & \mathbf{X}_i \sim q \text{ and } \mathbf{X}_j \not\sim q, \\ -1, & \text{otherwise}, \end{cases} \tag{4.5}$$

where symbol $\not\sim$ means not belonging to the pattern. On the other hand, if $P_{ij}^{q}(-1) \geq P_{ij}^{q}(1)$, then the decision rule is reversed.

For k top scoring gene pairs $(i_1, j_1), ..., (i_k, j_k)$, the discriminant function is defined as

$$f(\mathbf{x}) = \text{sgn}(\sum_{t=1}^{k} f_{i_t, j_t}(\mathbf{x})).$$

The size of trend pattern set Q could grow exponentially as the number of time points increases. With z time points, the possible number of trend pattern is 2^{z-1}. In order to control the size of pattern set, for the earthworm microarray time series data, only the head, middle and tail time points are considered. Therefore, there are only 4 trends patterns. Even though we sacrifice the accuracy because some time points are not considered, the computational complexity decreases significantly and the classifier is robust against noise fluctuations.

This classification algorithm only utilizes the trend between adjacent time points, and it will fail when the classification depends on feature value changes. For example, in the earthworm experiment, values of many genes in a chemical compound treated group increase or decrease with different value changes. Some of these changes is due to experimental error, and normally have small change value. If we only consider the trend pattern, it will be hard to reduce such experimental error.

TSP Based on Trend and Absolute Difference Amount

The absolute difference amount is defined as the sum of absolute differences of adjacent time points. Let Δ denote the difference amount. For the i-th feature of the n-th sample

with trend pattern q, the difference amount is

$$\Delta_{i,n}^{q}(m) = \sum_{t=1}^{z-1} |x_{i,n\_t+1} - x_{i,n\_t}|, \mathbf{x}_n \in C_m \mathbf{x}_{i,n} \sim q,$$

where $m = \{1, -1\}$ and $\mathbf{x}_{i,n} \sim q$ indicates that only the samples whose i-th feature has the trend pattern q are included in the computation.

The overall difference amount within the class for the i-th feature and trend pattern q is the sum of difference $\Delta_{i,n}^{q}(m)$ with respect to samples that belong to the class and follow pattern q. If there are no samples satisfying the criteria, the overall difference is 0.

$$\Delta_{i}^{q}(m) = \begin{cases} \frac{1}{|C_m|+N_{i,m}^{q}} \sum_n \Delta_{i,n}^{q}(m), & \mathbf{x}_n \in C_m, \mathbf{x}_{i,n} \sim q, \\ 0, & \mathbf{x}_n \in C_m, \nexists \mathbf{x}_{i,n} : \mathbf{x}_{i,n} \sim q, \end{cases} \tag{4.6}$$

where $m = \{1, -1\}$, $\nexists \mathbf{x}_{i,n} : \mathbf{x}_{i,n} \sim q$ means there is no sample in class the i-th feature of which follows trend pattern q, and $N_{i,m}^{q}$ denotes the number of samples that belongs to the class C_m and the i-th feature of which has the pattern q. The denominator adds $|C_m|$ to reduce the effect that there are only a few samples following the pattern but having high difference values.

The observation of the i-th feature having larger difference over the j-th within a class for a certain trend pattern implies either the i-th feature has trend pattern q while the j-th does not, or the pattern is exhibited in both features but the i-th feature shows the trend more clearly. The probability of observing the occurrence of such an event in each class is computed as

$$P_{i,j}^{q}(m) = Prob(\Delta_{i}^{q}(m) < \Delta_{j}^{q}(m)). \tag{4.7}$$

Let δ_{ij} denote the difference score of feature pair (i, j).

$$\delta_{ij}^{q} = |P_{ij}^{q}(1) - P_{ij}^{q}(-1)|.$$

The feature pair is selected based on the maximum difference score among all the trend patterns. Let δ_{ij} denotes this score.

$$\delta_{ij} = \max_q \delta_{ij}^q.$$

The difference score is computed for each pair of features. Pairs with high scores are viewed as most informative. It is easy to show that maximizing the score is equivalent to minimizing error rate of the classifier whose decision is made based on the comparison of difference values between the two features. And the top k pairs with maximum δ values are chosen and their corresponding trend pattern q are recorded as well. Suppose $P_{ij}^q(1) > P_{ij}^q(-1)$. Then, given an observation $\mathbf{x}$, the classifier $f_{ij}(\mathbf{x})$ based on this pair (i, j) is

$$f_{ij}(\mathbf{x}) = \begin{cases} 1, & \Delta_i^q < \Delta_j^q, \\ -1, & \text{otherwise}, \end{cases} \tag{4.8}$$

For a single observation $\mathbf{x}$, $\Delta_i^q(m) = \sum_{t=1}^{z-1} |x_{i\_t+1} - x_{i\_t}|$ when $\mathbf{x_i} \sim q$, $m = \{1, -1\}$. Note that, $\Delta_j^q(1)$ or $\Delta_j^q(1)$ has a value of 0 when the j-th feature does not follow the trend pattern q. Therefore, the classifier gives consideration of two scenarios: one feature follows the pattern while the other does not; both features have the pattern but one is more apparent than the other.

On the other hand, if $P_{ij}(-1) \geq P_{ij}(1)$, then the decision rule is reversed.

For k top scoring gene pairs $(i_1, j_1), ..., (i_k, j_k)$, the discriminant function is defined as

$$f(\mathbf{x}) = \text{sgn}(\sum_{t=1}^{k} f_{i_t, j_t}(\mathbf{x})).$$

Dynamic Time Warping with Nearest Neighbor

The previous methods did not fully utilize the time information, and therefore, some meaningful "spikes" on certain time points have been ignored. In theory, if there exists

a metric of measuring the distances between two time sequences, and an event using the measure is more likely to happen in one class than in the other, then, the TSP framework can be applied. However, though there are various distance measure of time series sequences, it is hard to define such an event associated with the measure. For example, we could use correlation coefficient for evaluating the correlations between two sequences, but it is difficult to define an event by which a classifier could be built. The key idea of this chapter is to use few features to build a simple classifier. In order to choose the most informative feature pairs, instead of computing the probability difference of the occurrence of an event from one class to the other, we use a simple classifier, such as nearest neighbor, to evaluate the performance of a feature pair. And once the top k feature pairs are selected, the simple classifier based on these features will be the final predictor.

There are many distance measures available for time series data. For example, Pearson correlation coefficient is typically used, especially in time series data clustering. It measures the strength of linear dependence between two sequences. Even though phase shift, Pearson correlation coefficient could not provide an accurate measure, due to its usually suggesting a coarse correlation. Also, an edge function was proposed in [Filkov *et al.*, 2002] to pick up strong local signals as opposed to global similarity.

Dynamic time warping [Sakoe & Chiba, 1978] has been used in many disciplines, such as signal processing, for a period of time. It earned its popularity by being extremely efficient as the time series similarity measure which minimizes the effects of shifting and distortion in order to detect similar shapes with different phases. Recently, it has been applied to the area of bioinformatics. Most of the works focused on the comparison of genes [Sheehy *et al.*, 2009; Böck *et al.*, 2011]. We use dynamic time warping to measure the similarity between the time sequences of a certain feature from two different samples.

Given two samples $\mathbf{u}$ and $\mathbf{v}$. For the feature pair (i, j), we compute the distance of these two samples as

$$d_{ij}(\mathbf{u}, \mathbf{v}) = \sqrt{DTW^2(\mathbf{u_i}, \mathbf{v_i}) + DTW^2(\mathbf{u_j}, \mathbf{v_j})}, \tag{4.9}$$

where $DTW(\mathbf{u_i}, \mathbf{v_i})$ represents the dynamic time warping distance of time sequence $\mathbf{u_i}$ and $\mathbf{v_i}$.

Suppose $\mathbf{u}$ is from the training set, and $\mathbf{v}$ is an observation to be classified. We use the simple nearest neighbor approach to do the prediction. Given a pair of features (i, j), the closest training sample of $\mathbf{v}$ is found based on the distance metric introduced above, and $\mathbf{v}$ shares the class label with its nearest neighbor.

The entire steps of TSP dynamic time wrapping with nearest neighbor methods are:

- First, the original training set is divided into a training set and a validation set;
- For each feature pair (i, j), every sample $\mathbf{v}$ in the validation set is classified based on nearest neighbor search in the two dimensional space constructed using Equation 4.9, and the performance of the entire validation set is recorded;
- The top k feature pairs with highest classification performances on the validation set are selected as $(i_1, j_1), ..., (i_k, j_k)$. The discriminant function is $f(\mathbf{x}) = sgn(\sum_{t=1}^{k} f_{i_t,j_t}(\mathbf{x}))$, where $f_{i_t,j_t}(\mathbf{x})$ is the nearest neighbor classifier generated using feature pair (i_t, j_t).

Results

We treated different periods (acclimation, exposure, and recovery) and treatments (control, carbaryl, and RDX) separately. There are five group combinations (control, carbaryl-exposure, carbaryl-recovery, RDX-exposure, RDX-recovery), and thus five classes in the prediction problem. One-vs-one method was used to deal with this multi-class scenario. The experimental data cannot be used directly on the treatment-period classification task, because first, each class has a different number of time points, and second, measurements at different time points are based on different individual earthworms. Thus, pre-processing steps are needed to generate time series data that meets the basic requirements of a classification task.

Each time point under a treatment-period combination has 5 to 8 replicates. Because these adult earthworm replicates were raised in the same environment, sampled and processed by the same procedures, it is reasonable to ignore the individual difference of replicates. To generate time sequence data, the measurements of replicates of adjacent time points within a class were concatenated randomly. The concatenation followed the rule that each replicate measurement should be used at least once. For example, among 5 replicates at time E01 (as shown in Figure 4.1), we randomly chose the first sample E01-01, and the third sample E02-03 at time E02 is chosen next. The measures of E01-01 and E02-03 will be concatenated. After the generation of the time sequence data, the measurements of E01-01 and E02-03 must be used at least once. We randomly generated time series data because the whole set of possible combinations of replicates from different time points will be large. For the earthworm data set of 13 time points with 5 replicates at each time point, the number of total possible combinations is $5^{13} = 1220703125$.

Another issue is the different number of time points of different classes. The class of control group has a total number of 31 time points, whereas two exposure classes have 13 and two recovery classes have 14. To maintain a consistent length for time sequences, we need to trim the numbers of time points of control and recovery classes. Because in general there is little change between the measurements of R13 and R14, the data from these two time points are combined and treated as being sampled from a single time point. And for the control class, in theory, the data should have a stable value. However, in reality, they fluctuate according to the time. Therefore, for the sake of comparison with exposure and recovery classes, the control class consists of sequences from E01 to E13, sequences from R01 to the combination of R13 and R14, and sequences randomly sampled among all the time points. The first two sets of sequences have the larger proportion of 80%, while the remaining 20% is from the last set.

After the pre-processing, every class has 13 time points. Each compound-treated class (no matter the exposure or recovery period) has 250 samples, and the control group has

500 samples. We randomly split the data set into training and test at ratio 4:1. For the dynamic time warping with nearest neighbor approach, the original training data is again divided into training and validation sets at a ratio 4:1.

There are $\binom{6000}{2} = 1799700$ possible gene pairs that could be generated from the 6000 genes. We preselected statistically significant genes using BRB ArrayTools v4.2.0 [1], and obtained 1192 significant gene candidates, and 500 most insignificant genes. For the TSP based approaches (including TSP based on average, TSP based on trend, TSP based on trend and difference amount), the classifier achieves a good performance if one gene is strongly active to the compounds while the other is inactive or behaviors reversely. So the significant genes and insignificant genes were paired. And we also added some pairs made only from the top 500 significant genes. and the total number of gene pairs is $1192 \times 500 + \binom{500}{2} = 720750$. For the dynamic time warping with nearest neighbor method, we only included the significant genes. And the number of possible gene pairs is $\binom{1192}{2} = 709836$.

In the TSP based approaches, we selected the top 10 gene pairs (10-TSP). The experiment was run by 20 times, and we computed the average accuracy and standard deviation for the four methods.

Table 4.1. Experimental Results of Time Series Earthworm Data

	TSP on Average	TSP on Trend	TSP on Trend and Difference	DTW with Nearest Neighbor
Accuracy	0.3978 ± 0.0368	0.4877 ± 0.0421	0.5027 ± 0.0437	0.6793 ± 0.04176
Top Scored Gene Pair	TA2-030964 TA1-195066	TA1-048554 TA1-033438	TA1-222050 TA2-055248	TA1-011325 TA2-124905
2nd Scored Gene Pair	TA1-094629 TA1-224931	TA2-058110 TA1-006918	TA2-164381 TA2-075017	TA1-118468 TA1-233321
3rd Scored Gene Pair	TA1-011325 TA1-126553	TA2-206312 TA1-084735	TA1-048554 TA2-180988	TA2-202612 TA1-038165

This is a classification problem with 5 classes. A random guess will achieve 20% accuracy in theory. From Table 4.1, all the four methods exhibited good classification ability even

[1] `http://linus.nci.nih.gov/brb`

though only 10 pairs of genes were involved in the prediction. The approach of dynamic time warping with nearest neighbor has the highest performance. TSP based approaches have lower performances but better interpretability than DTW with nearest neighbor. The top scored gene pairs may help to discover the underlying toxicity mechanism of chemical compounds.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

The dissertation was set out to explore the concept of restricting supervised learning problems from two different perspectives, feature selection and feature space partition.

Many supervised learning problems have either high dimensionality or the structural complexity of the generative function. High dimensionality arises in various applications such as computational biology, where redundant features decrease the prediction performance and may result in high variance no matter which learning algorithm is used. Feature selection identifies the most informative feature subset and decreases dimensionality. Another issue of supervised learning is that some problems have complex probabilistic distribution. A simple model, for example a linear classifier, may not be able to fit the data, even though simple models are generally advocated because they usually lead to low variance results. Feature space partition is a technique that splits the feature space into multiple non-overlapping regions such that a simple model can be learned for each region.

There are many methods of feature selection. Using 1-norm regularization is one of the most popular approaches. It embeds feature selection as part of the learning process. We apply 1-norm regularization into the ranking problem because there is little work that has been done about 1-norm ranking. In our method, a ranking problem is cast as a classification task where pair-wise preferences are considered. This casting increases the computational complexity from linear to quadratic in terms of sample size. We propose a method of ranking with 1-norm regularization using convex hull reduction. This method was tested on artificial data sets and two benchmark data sets, concrete compressive strength set and Abalone data

set. We showed the effect of significant sample size reduction by using convex hull reduction, and also demonstrated the feature selection results based on 1-norm regularization.

1-norm regularization has the benefit of feature selection. Theoretically, by tuning the tradeoff parameter between the empirical error and 1-norm regularizer, various feature subsets with different size could be obtained. But it requires the computing of the whole regularization path, which is extremely difficult due to the nondifferentiable nature of 1-norm penalty. To achieve a feature subset with the minimum size and high performance, we propose a 1-norm recursive feature elimination framework. This framework has the following advantages over other recursive feature selection methods: first, it automatically determines the number of features to be eliminated in each iteration; second, it has a natural iteration stopping criterion. We tested the method on an earthworm microarray data set. It outperformed many classic feature selection approaches in terms of accuracy and the size of selected feature subset.

A lot of problems have complex data generative models. Applying simple learning models will result in high bias. We propose a feature space partition method that divides the space based on domain information in a hierarchical fashion. It is assumed that domain knowledge is encoded by discrete or categorical attributes. Such an attribute provides a natural partition of the problem domain and therefore separates the original problem into several non-overlapping sub-problems. A discrete or categorical attribute is useful if the partition simplifies the learning task. It is time consuming to select such an attribute if an exhaustive method is used to search all the possible restructured problems, especially when there are lots of discrete or categorical attributes available. We use a metric to rank attributes according to their potential to reduce the uncertainty of a classification task. It is quantified as a conditional entropy given an attribute and a set of optimal classifier, each of which is built for a sub-problem defined by the attribute under consideration. We approximate the solution by the expected minimum conditional entropy with respect to random projections. The method was tested on three artificial data sets, three cheminformatics data

sets, and two leukemia gene expression data sets. Results showed that our method selected a proper categorical attribute and the learning performance of the restructured problem is higher than the original problem.

Simple learning models are always suggested for supervised learning. Besides linear models, another method is introduced in the dissertation, Top Scoring Pair (TSP). It does classification only based on the comparison of two features. Traditional TSP only deals with static data. We propose the following approaches for time series data using the TSP framework: TSP based on average comparison, TSP based on trend, and TSP based on variance amount. They all provide simple metrics for selecting the most discriminative feature pair and build the simple models. TSP based methods have the advantage of good interpretability power. Based on the idea of using only two features, we also propose the method of dynamic time warping with nearest neighbor based on two features. These four approaches were tested on a time series earthworm microarray data set, where five classes were considered. And the result showed the best performance was achieved on the last method.

There are a lot of works could be done in the future that extend the idea of learning with a feature pair. A classifier built on two features is normally a weak learner. Therefore, boosting is a natural choice that consists of iteratively learning weak classifiers and aggregating them to build a strong classifier. Each weak learner may select a different feature pair, and each feature pair represents a classification rule. Hence, the aggregation of multiple TSP classifiers is equivalent to the combination of multiple decision rules, which is comparable with decision tree method. We may be able to borrow decision tree related concepts and methods, and apply them on TSP based approaches in the future. Also, when many feature pairs are considered, it may be required to reduce the feature size. And 1-norm regularization is a good option because it has embedded feature selection ability. Another direction of future works is about how to define interpretability. We talk about this word a lot but never quantify it. If we are able to measure the interpretability of a model, we can achieve a

classifier with balance between the learning performance and model interpretability. And this framework will be very useful for researchers who want to understand more about the patterns or rules indicated by a classifier instead of treating it as a black box. It also can help incorporate domain knowledge in the classifier.

Bibliography

Bibliography

Abu-Mostafa, Y. S. (1994) "Learning from Hints." *Journal of Complexity*, Vol. 10, pp. 165–178.

Altmann, A.; Beerenwinkel, N.; Sing, T.; Savenkov, I.; Däumer, M.; Kaiser, R.; Rhee, S.; Fessel, W. J.; Shafer, R. W.; & Lengauer, T. (2007) "Improved Prediction of Response to Antiretroviral Combination Therapy Using the Genetic Barrier to Drug Resistance." *Antiviral Therapy*, Vol. 12(2), pp. 169–178.

Altmann, A.; Sing, T.; Vermeiren, H.; Winters, B.; Craenenbroeck, E. V.; Borght, K. V.; Rhee, S.; Shafer, R. W.; Schülter, E.; Kaiser, R.; Peres, Y.; Sönnerborg, A.; Fessel, W. J.; Incardona, F.; Zazzi, M.; Bacheler, L.; Vlijmen, H. V.; & Lengauer, T. (2009) "Advantages of Predicted Phenotypes and Statistical Learning Models in Inferring Virological Response to Antiretroviral Therapy from HIV Genotype." *Antiviral Therapy*, Vol. 14(2), pp. 273–283.

Ambroise, C. & McLachlan, G. J. (2002) "Selection Bias in Gene Extraction on the Basis of Microarray Gene-Expression Data." *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 99, pp. 6562–6566.

Anjum, S.; Doucet, A.; & Holmes, C. C. (2009) "A Boosting Approach to Structure Learning of Graphs with and withour Prior Knowledge." *Bioinformatics*, Vol. 25(22), pp. 2929–2936.

Bahl, L. R.; deSouza, P. V.; Nahamoo, D.; & Padmanabhan, M. (2000) "System and Method for Partitioning the Feature Space of A Classifier in A Pattern Classification System.".

Balcan, M.; Bansal, N.; Beygelzimer, A.; Coppersmith, D.; Langford, J.; & Snorkin, G. B. (2008) "Robust Reductions from Ranking to Classification." *Machine Learning*, Vol. 72(1-2), pp. 139–153.

Bar-Joseph, Z. (2004) "Analyzing Time Series Gene Expression Data." *Bioinformatics*, Vol. 20(16), pp. 2493–2503.

Bekkerman, R.; EI-Yaniv, R.; Tishby, N.; & Winter, Y. (2002) "Distributional Word Clusters vs. Words for Text Categorization." *Journal of Machine Learning Research*, Vol. 1, pp. 1–48.

Bellman, R. E. (1957) *Dynamic Programming.* Princeton University Press.

Berrar, D. P.; Sturgeon, B.; Bradbury, I.; & Dubitzky, W. (2003) "Microarray Data Integration and Machine Learning Techniques for Lung Cancer Survival Prediction." In *Proceedings of the the International Conference of Critical Assessment of Microarray Data Analysis*, pages 43–54.

Bi, J.; Bennett, K. P.; Embrechts, M.; Breneman, C.; & Song, M. (2003) "Dimensionality Reduction via Sparse Support Vector Machines." *Journal of Machine Learning Research*, Vol. 3, pp. 1229–1243.

Blum, A. & Langley, P. (1997) "Selection of Relevant Features and Examples in Machine Learning." *Artificial Intelligence*, Vol. 97, pp. 245–271.

Böck, M.; Schmitt, C.; & Kramer, S. (2011) "A Study of Dynamic Time Warping for the Inference of Gene Regulatory Relationships." In *Proceedings of the 5th International Workshop on Machine Learning in Systems Biology*, pages 6–9.

Bogojeska, J.; Bickel, S.; Altmann, A.; & Lengauer, T. (2010) "Dealing with Sparse Data in Predicting Outcomes of HIV Combination Therapies." *Bioinformatics*, Vol. 26(17), pp. 2085–2092.

Borgwardt, K. M.; Vishwanathan, S. V. N.; & Kriegel, H. (2006) "Class Prediction from Time Series Gene Expression Profiles Using Dynamical Systems Kernels." In *Pacific Symposium on Biocomputing 11*, pages 547–558.

Breiman, L. (1996) "Heuristics of Instability and Stabilization in Model Selection." *Annals of Statistics*, Vol. 24(6), pp. 2350–2382.

Breiman, L. (2001) "Random Forest." *Machine Learning*, Vol. 45(1), pp. 5–32.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; & Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth & Brooks/Cole Advanced Books & Software ISBN 978-0412048418.

Chapelle, O. & Keerthi, S. S. (2008) "Multi-class Feature Selection with Support Vector Machine." *Proceedings of the American Statistical Association*, Vol. .

Chen, Y.; Bi, J.; & Wang, J. Z. (2006) "MILES: Multiple-instance Learning via Embedded Instance Selection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(12), pp. 1931–1946.

Cohen, W. W.; Schapire, R. E.; & Singer, Y. (1999) "Learning to Order Things." *Journal of Artificial Intelligence Research*, Vol. 10(1), pp. 243–270.

Cortes, C. & Vapnik, V. (1995) "Support Vector Networks." *Machine Learning*, Vol. 20(3), pp. 273–297.

Crammer, K. & Singer, Y. (2001) "On the Algorithm Implementation of Multiclass Kernel-based Vector Machines." *Journal of Machine Learning Research*, Vol. 2, pp. 265–292.

Daemen, A.; Gevaert, O.; Bie, T. D.; Debucquoy, A.; Machiels, J.; Moor, B. D.; & Haustermans, K. (2008) "Integrating Microarray and Proteomics Data to Predict the Response on Cetuximab in Patients with Rectal Cancer." *Pacific Symposium on Biocomputing*, Vol. 25, pp. 322–330.

Djebbari, A. & Quackenbush, J. (2008) "Seeded Bayesian Networks: Constructing Genetic Networks from Microarray Data." *BMC Systems Biology*, Vol. 2, pp. 57.

Domingos, P. & Pazzani, M. (1997) "On the Optimality of the Simple Bayesian Classifier under Zero-one Loss." *Machine Learning*, Vol. 29, pp. 103–137.

Drucker, H.; Burges, C. J. C.; Kaufman, L.; Smola, A.; & Vapnik, V. (1997) *Advances in Neural Information Processing System 9.*, chapter Support Vector Regression Machine, pages 155–161 MIT Press.

Duchi, J. & Singer, Y. (2009) "Boosting with Structural Sparsity." *Proceedings of the 26th Annual International Conference on Machine Learning*, Vol. pages 297–304.

Dwork, C.; Kumar, R.; Naor, M.; & Sivakumar, D. (2001) "Rank Aggregation Methods for the Web." *Proceedings of the 10th international conference on World Wid Web*, Vol. pages 613–622.

English, S. B. & Butte, A. J. (2007) "Evaluation and Integration of 49 Genome-wide Experiments and the Prediction of Previously unknown Obesity-related Genes." *Bioinformatics*, Vol. 23(21), pp. 2910–2917.

Erdal, S.; Ozturk, O.; Armbruster, D.; Ferhatosmanoglu, H.; & Ray, W. C. (2004) "A Time Series Analysis of Microarray Data." In *Proceeding of the 4th IEEE Symposium on Bioinformatics and Bioengineering*, pages 336–342.

Filkov, V.; Skiena, S.; & Zhi, J. (2002) "Analysis Techniques for Microarray Time-Series Data." *Journal of Computational Biology*, Vol. 9, pp. 317–330.

Friedman, J.; Hastie, T.; & Tibshirani, R. (2010) "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Lightwave Technology of Statistical Software*, Vol. 33(1), pp. 1–22.

Fung, G.; Rosales, R.; & Krishnapuram, B. (2006) "Learning Rankings via Convex Hull Separation." *Advances in Neural Information Processing Systems*, Vol. 18, pp. 395–402.

Geman, D.; Afsari, B.; Tan, A. C.; & Naiman, D. (2008) "Microarray Classification from Several Two-gen Expression Comparison." *Proceedings of International Conference on Machine Learning and Applications*, Vol. 11-13, pp. 583–585.

Geman, D.; d'Avignon, C.; Naiman, D. Q.; & Winslow, R. L. (2004) "Classifying Gene Expression Profiles from Pairwise mRNA Comparison." *Statistical Applications in Genetics and Molecular Biology*, Vol. 3, pp. Article 19.

Golub, T.; Slonim, D.; Tamayo, P.; Huard, C.; Gaasenbeek, M.; Mesirov, J.; Coller, H.; Loh, M.; Downing, J.; Caligiuri, M.; Bloomfield, C.; & Lander, E. (1999) "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression." *Science*, Vol. 286(5439), pp. 531–537.

Gong, P.; Loh, P. R.; Barker, N. D.; Tucker, G.; Wang, N.; Zhang, C.; Escalon, B. L.; Berger, B.; & Perkins, E. J. (2011) "Building Quantitative Prediction Models for Tissue Residue of Two Explosives Compounds in Earthworms from Microarray Gene Expression Data." *Environ Sci Technol*, Vol. 46(1), pp. 19–26.

Guyon, I. & Elisseeff, A. (2003) "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research*, Vol. 3, pp. 1157–1182.

Guyon, I.; Weston, J.; Barnhill, S.; & Vapnik, V. (2002) "Gene Selection for Cancer Classification Using Support Vector Machine." *Machine Learning*, Vol. 46(1-3), pp. 389–422.

Hall, M. (2000) "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning." In *Proceedings of the 17th International Conference on Mahince Learning*, pages 359–366.

He, Y.; Zhang, B.; & Li, J. (2005) "A New Multiresolution Classification Model based on Partitioning of Feature Space." *Proceedings for 2005 IEEE International Conference on Granular Computing*, Vol. 2, pp. 462–467.

Herbrich, R.; Graepel, T.; & Obermayer, K. (2000) *Advances in Large Margin Classifiers.*, chapter Large Margin Rank Boundaries for Ordinal Regression, pages 115–132 MIT Press, Cambridge, MA.

Hsu, C. & Lin, C. (2002) "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Transactions on Neural Networks*, Vol. 13, pp. 415–425.

Hueter, I. (1999) "Limit Theorems for the Convex Hull of Random Points in Higher Dimensions." *Transactions of American Mathematical Society*, Vol. 351(11), pp. 4337–4363.

Jing, L. & Ng, M. K. (2010) "Prior Knowledge Based Mining Functional Modules from Yeast PPI Netwoks with Gene Ontology." *BMC Bioinformatics*, Vol. 11(Supplement: 11), pp. 1–19.

Jorgensen, R.; Merrill, A.; & Andersen, G. R. (2006) "The Life and Death of Translation Elongation Factor 2." *Biochem Soc Trans*, Vol. 34, pp. 1–6.

Kishore, J. K.; Patnaik, L. M.; mani, V.; & k. Agrawal, V. (2001) "Genetic Programming Based Pattern Classification with Feature Space Partitioning." *Information Sciences*, Vol. 131(1-4), pp. 65–86.

Kittler, J. (1978) *Pattern Recognition and Signal Processing.*, chapter Feature Set Search Algorithms, pages 41–60 Sijthoff and Noordhoff.

Kohavi, R. & John, G. (1997) "Wrappers for Feature Selection." *Artificial Intelligence*, Vol. 97(1-2), pp. 273–324.

Kohn, A. F.; Nakano, L. G. M.; & Silva, M. O. E. (1996) "A Class Discriminability Measure Based on Feature Space Partitioning." *Pattern Recogonition*, Vol. 29(5), pp. 873–887.

Lal, T. N.; Chapelle, O.; Weston, J.; & Elisseeff, A. (2006) *Embedded Methods.* Springer-Verlag.

Lee, P. H. & Shatkay, H. (2009) "An Intergrative Scoring System for Ranking SNPs by their potential deleterious effects." *Bioinformatics*, Vol. 25(8), pp. 1048–1055.

Li, T.; Zhang, C.; & Ogihara, M. (2004) "A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression." *Bio*, Vol. 20(15), pp. 2429–2437.

Liu, B.; Wan, C.; & Wang, L. (2006) "An Efficient Semi-unsupervised Gene Selection Method via Spectral Biclustering." *IEEE Transactions on Nano-Bioscience*, Vol. 5(2), pp. 110–114.

Liu, Y. & Shen, X. (2006) "Multicategory Psi-learning." *Journal of the American Statistical Association*, Vol. 101(474), pp. 500–509.

Lustgarten, J. L.; Visweswaran, S.; Bowser, R.; Hogan, W.; & Gopalakrishnan, V. (2009) "Knowledge-based Variable Selection for Learning Rules from Proteomic Data." *BMC Bioinformatics*, Vol. 10(Supplement: 9), pp. 1–7.

Mani, K. M.; Lefebvre, C.; Wang, K.; Lim, W. K.; Basso, K.; Dalla-Favera, R.; & Califano, A. (2008) "A Systems Biology Approach to Prediction of Oncogenes and Molecular Perturbation Targets in B-cell Lymphomas." *Molecular System Biology*, Vol. 4(Article No. 169).

Mattera, D. & Haykin, S. (1999) *Advances in Kernel Methods – Support Vector Learning.*, chapter Support Vector Machines for Dynamic Reconstruction of A Chaotic System, pages 211–242 MIT Press.

Meinshausen, N. & Bühlmann, P. (2010) "Stability Selection." *Journal of the Royal Statistical Society*, Vol. Series B 72, pp. 417–473.

Müller, K. R.; Smola, A.; Rätsch, G.; Schölkopf, B.; Kohlmorgen, J.; & Vapnik, V. (1997) *Artificial Neural Networks ICANN'97.*, chapter Predicting Time Series with Support Vector Machine, pages 999–1004.

Nan, X.; Chen, Y.; Dang, X.; & Wilkins, D. (2010a) "Learning to Rank Using 1-norm Regularization and Convex Hull Reduction." *Proceeding of ACM Southeast 2010*, Vol. page 5 pages.

Nan, X.; Fu, G.; Zhao, Z.; Liu, S.; Patel, R. Y.; Liu, H.; Daga, P. R.; Doerksen, R. J.; Dang, X.; Chen, Y.; & Wilkins, D. (2011) "Leveraging Domain Information to Restructure Biological Prediction." *BMC Bioinformatics*, Vol. 12, pp. 15 pages.

Nan, X.; Wang, N.; Gong, P.; Zhang, C.; Chen, Y.; & Wilkins, D. (2010b) "Gene Selectioin Using 1-Norm Regularization for Multi-Class Microarray Data." *Proceeding of IEEE International Conference on Bioinformatics & Biomedicine*, Vol. pages 520–524.

Niyogi, P.; Girosi, F.; & Poggio, T. (1998) "Incorporating Prior Information in Machine Learning by Creating Virtual Examples." *Proceedings of the IEEE*, Vol. 86(11), pp. 2196–2209.

Ochs, M. F. (2010) "Knowledge-based Data Analysis Comes of Age." *Briefings in Bioinformatics*, Vol. 11(1), pp. 30–39.

Padmanabhan, M.; Bahl, L. R.; & Nahamoo, D. (1999) "Partitioning the Feature Space of a Classifier with Linear Hyperplanes." *IEEE Transactions on Speech and Audio Processing*, Vol. 7(3), pp. 282–288.

Parker, B. J. & Wen, J. (2009) "Predicting microRNA Targets in Time-Series Microarray Experiments via Functional Data Analysis." *BMC Bioinformatics*, Vol. 10, pp. S32.

Pearson, K. (1901) "On Lines and Planes of Closest Fit to System of Points in Space." *Philosophical Magazine*, Vol. 2(6), pp. 559–572.

Platt, J. C.; Cristianini, N.; & Taylor, J. S. (2000) "Large Margin DAGs for Multiclass Classification." *Advances in Neural Information Processing Systems*, Vol. pages 547–553.

Poggio, T. & Girosi, F. (1990) "Networks for Approximation and Learning." *Proceedings of the IEEE*, Vol. 78(9), pp. 1481–1497.

Poggio, T. & Vetter, T. (1992) "Recognition and Structure from One 2D Model View: Observations on Prototypes, Object Classes and Symmetrics." *A.I. Memo No. 1347*, Vol. .

Ramakrishnan, S. R.; Vogel, C.; Prince, J. T.; Li, Z.; Penalva, L. O.; Myers, M.; Marcotte, E. M.; Miranker, D. P.; & Wang, R. (2009) "Integrating Shotgun

Proteomics and mRNA Expression Data to Improve Protein Identification." *Bioinformatics*, Vol. 25(11), pp. 1397–1403.

Ramoni, M.; Sebastiani, P.; & Kohane, I. (2002) "Cluster Analysis of Gene Expressioin Dynamics." In *Proceedings of National Academy of Sciences*, volume 99, pages 9121–9126.

Rokach, L. & Maimon, O. (2008) *Data Mining with Decision Trees: Theory and Applicatioins*. World Scientific Publishing Co. Pte. Ltd.

Rosset, S. (2004) "Tracing Curved Regularized Optimization Solution Paths." *Advances in Neural Information Processing Systems*, Vol. 17.

Sakoe, H. & Chiba, S. (1978) "Dynamic Programming Algorithm optimization for Spoken Word Recognition." *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 26(1), pp. 43–49.

Salem, S.; Jack, L.; & Nandi, A. (2008) "Investigation of Self-organizing Oscillator Networks for Use in Clustering Microarray Data." *IEEE Trans. NanoBioscience*, Vol. 7, pp. 65–79.

Schliep, A.; Schonhuth, A.; & Steinhoff, C. (2003) "Using Hidden Markov Model to Analyze Gene Expression Time Course Data." *Bioinformatics*, Vol. 19, pp. 1264–1272.

Schölkopf, B.; Simard, P.; Smola, A.; & Vapnik, V. (1998) "Prior Knowledge in Support Vector Kernels." *Advances in Neural Information Processing Systems*, Vol. 10, pp. 640–646.

Sethi, I. K. & sarvarayudu, G. P. R. (1982) "Hierarchical Classifier Design Using Mutual Information." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4(4), pp. 441–445.

Sheehy, S. P.; Huang, S.; & Parker, K. K. (2009) "Time-warped Comparison of Gene Expression in Adaptive and Maladaptive Cardiac Hypertrophy." *Circ Cardiovasc Genet*, Vol. 2(2), pp. 116–124.

Silverman, B. W. (1986) *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.

Simard, P.; LeCun, Y.; & Denker, J. S. (1993) "Efficient Pattern Recognition Using a New Transformation Distance." *Proceedings of Advances in Neural Information Processing Systems*, Vol. 5, pp. 50–58.

Singh, S. & Galton, A. P. (2003) "Multiresolution Estimates of Classification Complexity." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25(12), pp. 1534–1539.

Statnikov, A.; Aliferis, C. F.; Tsamardinos, I.; Hardin, D.; & levy, S. (2006) "A Comprehensive Evaluation of Multicategory Classification Methods for Microarray Gene Expression Cancer Diagnosis." *Bioinformatics*, Vol. 21(5), pp. 631–643.

Stitson, M.; Gammerman, A.; Vapnik, V.; Vovk, V.; Watkins, C.; & Weston, J. (1999) *Advances in Kernel Methods – Support Vector Learning.*, chapter Support Vector Regression with ANOVA Decomposition Kernels, pages 285–292 MIT Press.

Suzuki, T.; Honda, M.; Matsumoto, S.; Sturzenbaum, S.; & Gamou, S. (2005) "Valosine-containing Proteins (VCP) in An Annelid: Identification of A Novel Spermatogenesis Related Factor." *Gene*, Vol. 362, pp. 11–18.

Szedmak, S.; Shawe-Taylor, J.; Saunders, C. J.; & Hardoon, D. R. (2004) "Multiclass Classification by L1 Norm Support Vector Machine." *Pattern Recognition and Machine Learning in Computer Vision Workshop*, Vol. Grenoble, France.

Tian, Z.; Hwang, T. H.; & Kuang, R. (2009) "A Hypergraph-based Learning Algorithm for Classifying Gene Expression and ArrayCGH Data with Prior Knowledge." *Bioinformatics*, Vol. 25(21), pp. 2831–2838.

Tibshirani, R. (1996) "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 58, pp. 267–288.

Tseng, V. S. & Kao, C. P. (2005) "Efficiently Mining Gene Expression Data via A Novel Parameterless Clustering Method." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 2, pp. 355–365.

Ucar, D.; Beyer, A.; Parthasarathy, S.; & Workman, C. T. (2009) "Predicting Functionality of Protein-DNA Interactions by Integrating Diverse Evidence." *Bioinformatics*, Vol. 25(12), pp. 137–144.

Ulitsky, I. & Shamir, R. (2009) "Identifying Functional Modules Using Expression Profiles and Confidence-scored Protein Interactions." *Bioinformatics*, Vol. 25(9), pp. 1158–1164.

Valev, V. (2004) "Supervised Pattern Recognition by Parallel Feature Partitioning." *Pattern Recognition*, Vol. 37(3), pp. 463–467.

Vapnik, V. N. (1995) *The Nature of Statistical Learning Theory.* Springer-Verlag New York.

Verri, A. & Poggio, T. (1986) "Regularization Theory and Shape Constraints." *A.I. Memo No. 916*, Vol. .

Viswanadhan, V. N.; Ghose, A. K.; Revankar, G. R.; & Robins, R. K. (1989) "Atomic Physicochemical Parameters for Three Dimensional Structure Directed Quantitative Structure-activity Relationships. 4. Additional Parameters for Hydrophobic and

Dispersive Interactions and Their Application for An Automated Superposition of Certain Naturally Occurring Nucleoside Antibiotics." *J. Chem. inf. Comput. Sci.*, Vol. 29, pp. 163–172.

Wang, J.; Neskovic, P.; & Cooper, L. N. (2003) "Partitioning A Feature Space Using A Locally Defined Confidence Measure." *Joint 13th International Conference on Artificial Neural Networks and 10th International Conference on Neural Information Processing*, Vol. Istanbul.

Wang, L. & Shen, X. (2006) "Multi-category Support Vector Machines, Feature Selection and Solution Path." *Statistica Sinica*, Vol. 16, pp. 617–633.

Werhli, A. V. & Husmeier, D. (2008) "Gene Regulatory Network Reconstruction By Bayesian Integration of Prior Knowledge And/OR Different Experimental Conditions." *Journal of Bioinformatics and Computational Biology*, Vol. 6(3), pp. 543–572.

Weston, J.; Elisseff, A.; Schölkopf, B.; & Tipping, M. (2003) "Use of the Zero Norm with Linear Models and Kernel Methods." *Journal of Machine Learning Research*, Vol. 3, pp. 1439–1461.

Weston, J. & Watkins, C. (1999) "Multi-class Support Vector Machines." *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, Vol. Bruges.

Witten, I. H. & Frank, E. (2005) "Incorporating Domain Knowledge." In *Data mining: Practical Machine Learning Tools and Techniques*, pages 349–351. D. Cerra, 2nd edition.

Yeoh, E.; Ross, M.; Shurtleff, S.; Williams, W.; Patel, D.; Mahfouz, R.; Behm, F.; Raimondi, S.; Relling, M.; Patel, A.; Cheng, C.; Campana, D.; Wilkins, D.; Zhou, X.; Li, J.; Liu, H.; Pui, C.; Evans, W.; C, C. N.; Wong, L.; & Downing, J. (2002) "Classification, Subtype Discovery, and Prediction of Outcome in Pediatric Acute Lymphoblastic Leukemia by Gene Expression Profiling." *Cancer Cell*, Vol. 1(2), pp. 133–143.

Zhang, X.; Lu, X.; Shi, Q.; Xu, X.; Leung, H. E.; Harris, L. N.; Iglehart, J. D.; Miron, A.; Liu, J. S.; & Wong, W. H. (2006) "Recursive SVM Feature Selection and Sample Classification for Mass-spectrometry and Microarray Data." *BMC Bioinformatics*, Vol. 7, pp. 197–210.

Zhu, J.; Rosset, S.; Hastie, T.; & Tibshirani, R. (2003) "1-norm Support Vector Machines." Stanford University, Technical report.