KERNEL-BASED EMPIRICAL BAYESIAN CLASSIFICATION METHODS WITH APPLICATIONS TO PROTEIN PHOSPHORYLATION AND NON-CODING RNA

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI‘I AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

AUGUST 2014

By
Mark S. Menor

Dissertation Committee:

Kyungim Baek, Chairperson
Guylaine Poisson, Chairperson
Henri Casanova
Scott Robertson
Gernot Presting

Acknowledgments

I would first like to thank my advisors, Dr. Kyungim Baek and Dr. Guylaine Poisson, for all their guidance and support. Their comments and feedback throughout this dissertation research project were invaluable.

I would also like to thank the members of my dissertation committee, Dr. Henri Casanova, Dr. Scott Robertson, and Dr. Gernot Presting, for all their suggestions and advice.

Abstract

With the advancement of high-throughput sequencing technologies, a new era of "big data" biological research has dawned. However, the abundance of biological data presents many challenges in their analysis and it has proven very difficult to extract important information out of the data. One approach to this problem is to use the methods of machine learning.

In this dissertation, we describe novel probabilistic kernel-based learning methods and demonstrate their practical applicability by solving major bioinformatics problems at the transcriptome and proteome levels where the resulting tools are expected to help biologists further elucidate the important information contained in their data.

The proposed binary classification method, the Classification Relevance Units Machine (CRUM), employs the theory of kernel and empirical Bayesian methods to achieve non-linear classification and high generalization. We demonstrate the practical applicability of CRUM by applying it to the prediction of protein phosphorylation sites, which helps explain the mechanisms that control many biochemical processes.

Then we develop an extension of CRUM to solve multiclass problems, called the Multiclass Relevance Units Machine (McRUM). McRUM uses the error correcting output codes framework to decompose a multiclass problem into a set of binary problems. We devise a linear-time algorithm to aggregate the results into the final probabilistic multiclass prediction to allow for predictions in large scale applications. We demonstrate the practical applicability of McRUM through a solution to the identification of mature microRNA (miRNA) and piwi-interacting RNA (piRNA) in small RNA sequencing datasets. This provides biologists a tool to help discover novel miRNA and piRNA to further understand the molecular processes of the organisms they study.

Table of Contents

Acknowledgments . . . iii
Abstract . . . iv
List of Tables . . . vii
List of Figures . . . viii
List of Abbreviations . . . 1
1 Introduction . . . 1
1.1 Background and Motivation . . . 1
1.2 Problem Statement . . . 5
1.3 Research Description . . . 5
1.4 Related Work . . . 6
1.4.1 Protein Phosphorylation Site Prediction . . . 6
1.4.2 Small ncRNA Prediction . . . 8
1.5 Expected Contributions . . . 11
1.6 Structure of Dissertation . . . 11
2 Relevance Units Machines for Classification . . . 16
2.1 Introduction . . . 17
2.2 Model . . . 19
2.3 Assumptions . . . 19
2.4 Learning Algorithm . . . 21
2.5 Experimental Results . . . 23
2.5.1 Ripley's Synthetic Data . . . 24
2.5.2 Benchmark Comparisons . . . 26
2.6 Conclusion . . . 28
3 Probabilistic Prediction of Protein Phosphorylation Sites Using Classification Relevance Units Machines . . . 32
3.1 Introduction . . . 33
3.2 Kernel Machines . . . 35
3.2.1 Support Vector Machine . . . 36
3.2.2 Relevance Vector Machine . . . 36
3.3 Classification Relevance Units Machine . . . 37
3.3.1 Overview of Learning Algorithm . . . 38
3.3.2 Learning Algorithm Assumptions and Derivation . . . 39
3.3.3 Selecting Model Complexity . . . 44

3.4 Experimental Setup . . . 45
3.4.1 Dataset Preparation . . . 45
3.4.2 Performance Assessment Measures . . . 47
3.4.3 Implementation . . . 47
3.5 Results and Discussion . . . 48
3.5.1 Comparison of Learning Methods . . . 48
3.5.2 Evaluation of Protein Site Features . . . 52
3.5.3 Benchmark Comparison . . . 56
3.6 Conclusions . . . 59
4 Multiclass relevance units machine: benchmark evaluation and application to small ncRNA discovery . . . 67
4.1 Background . . . 68
4.2 Methods . . . 70
4.2.1 Classification Relevance Units Machine . . . 70
4.2.2 The Multiclass Classification Problem and Solutions . . . 71
4.2.3 ncRNA dataset preparation and features . . . 78
4.2.4 Performance measures . . . 78
4.3 Results and Discussion . . . 80
4.3.1 Benchmark experiments . . . 80
4.3.2 Small ncRNA experiments . . . 87
4.4 Conclusions . . . 95
5 Prediction of Mature MicroRNA and Piwi-interacting RNA Without a Genome Reference or Precursors . . . 100
5.1 Introduction . . . 101
5.2 Results . . . 103
5.2.1 Feature Selection . . . 103
5.2.2 Kernel Selection . . . 105
5.2.3 Comparison with SVM and piRNApredictor using Hold-out Test Set 106
5.2.4 Comparison with miRPlex using Experimental Datasets . . . 109
5.3 Methods . . . 111
5.3.1 Classification Methods . . . 111
5.3.2 Gaussian Kernels . . . 112
5.3.3 Datasets . . . 113
5.3.4 Features . . . 114
5.4 Conclusion . . . 115
6 Conclusion . . . 119
6.1 Future Work . . . 120

List of Tables

Table		Page
2.1	Benchmarking Results of SVM, RVM, and CRUM	25
3.1	Composition of datatsets	45
3.2	Comparison of predictors on test dataset for serine site prediction	50
3.3	Comparison of predictors on test dataset for threonine site prediction	50
3.4	Comparison of predictors on test dataset for tyrosine site prediction	50
3.5	Three-fold cross-validation performance on phosphoserine (S) prediction	53
3.6	Three-fold cross-validation performance on phosphothreonine (T) prediction	54
3.7	Three-fold cross-validation performance on phosphotyrosine (Y) prediction	54
4.1	McRUM results on wine dataset using 10-fold cross-validation	81
4.2	Prediction time of McRUM on benchmark datasets (in seconds)	83
4.3	McRUM results on iris dataset using 10-fold cross-validation	84
4.4	McRUM results on yeast dataset using 10-fold cross-validation	84
4.5	McRUM results on thyroid training and test datasets	85
4.6	McRUM results on satellite image training and test datasets	86
5.1	Summary of experimental datasets	113

List of Figures

Figure		Page
1.1	Non-linear transformation of data	3
1.2	A multilayer perceptron	4
2.1	Plots of CRUM and RVM on Ripley's data	26
3.1	The CRUM model displayed as a feed-forward network	38
3.2	The CRUM model displayed as a Bayesian network	41
3.3	ROC for phosphoserine (S) prediction by CRUM based on cross-validation performance	55
3.4	ROC for phosphothreonine (T) prediction by CRUM based on cross-validation performance	55
3.5	ROC for phosphoserine (Y) prediction by CRUM based on cross-validation performance	56
3.6	ROC for phosphoserine (S) prediction by CRUM based on benchmark performance	58
3.7	ROC for phosphothreonine (T) prediction by CRUM based on benchmark performance	58
3.8	ROC for phosphoserine (Y) prediction by CRUM based on benchmark performance	59
3.9	Training running-time comparison of CRUM algorithms	61
3.10	Accuracy comparison between empirical Bayes and maximum likelihood	62
4.1	3-fold cross-validation results for miRNA being the positive class	88
4.2	3-fold cross-validation results for piRNA being the positive class	89
4.3	3-fold cross-validation results for non-miRNA and non-piRNA being the positive class	89
4.4	The fraction of unclassified sequences for cross-validation experiment	90
4.5	Evaluation results on independent test set for miRNA being the positive class	91
4.6	Evaluation results on independent test set for piRNA being the positive class	91
4.7	Evaluation results on independent test set for non-miRNA and nonpiRNA being the positive class	92
4.8	The fraction of unclassified sequences for independent test experiment	93

4.9 piRNA prediction results 94
4.10 piRNA prediction results 94

5.1 ROC curves for CFS and all features McRUMs 104
5.2 ROC curves for McRUMs using L_1 and L_2 Gaussian kernels 106
5.3 ROC curves for McRUMs and SVM 107
5.4 Average model size for McRUMs SVM 108
5.5 ROC curves for McRUMs and piRNApredictor 108
5.6 Number of miRNA predictions for McRUM and miRPlex 110

List of Abbreviations

AP	all pairs
BLOSUM	blocks substitution matrix
CFS	correlation-based feature selection
CRUM	classification relevance units machine
DNA	deoxyribonucleic acid
FPR	false positive rate
GBT	generalized Bradley-Terry model
McRUM	multiclass relevance units machine
miRNA	microRNA
MLP	multilayer perceptron
mRNA	messenger RNA
ncRNA	non-coding RNA
OVR	one-versus-rest
piRNA	piwi-interacting RNA
PK	protein kinase
SVM	support vector machine
TPR	true positive rate
RISC	RNA-induced silencing complex
RNA	ribonucleic acid
RNA-Seq	RNA sequencing
ROC	receiver operating characteristic
RUM	relevance units machine
RVM	relevance vector machine

Chapter 1

Introduction

In this dissertation, we describe novel probabilistic kernel-based learning methods and demonstrate their practical applicability by solving major bioinformatics problems at the transcriptome and proteome levels where the resulting tools are expected to help biologists further elucidate the important information contained in their data.

1.1 Background and Motivation

A new era of biological research has dawned with the development of high-throughput sequencing technologies. An explosion of biological data has occurred due to the use of a wide variety of technologies to study the diversity of life, including genome sequencing, RNA sequencing, and mass spectrometry. This has led to an explosion of biological data now residing in many public databases. The abundance of data, however, presents many challenges in their analysis. This has led to a situation some call "data rich, information poor," as it has proven very difficult to extract important information out of the biological data and thus many details of biological mechanisms remain a mystery. In this sense, bioinformatics can be viewed as the sub-field of computer science that provides the tools biologists need to turn their data into information.

The genome of an organism is the entirety of that organism's DNA. According to the Central Dogma of Molecular Biology, segments of DNA are converted into two major types of RNA via a process called transcription. One type of RNA is called non-coding RNA (ncRNA) and they are themselves converted into biochemically functioning agents with a variety of known functions, but the functions of most ncRNA remain unknown to

this day. The other type of RNA is called coding RNA or messenger RNA (mRNA). As the name implies, they transport information to the cell's protein construction machinery. The information contained in the mRNA are the blueprints for proteins that are constructed via a process called translation, the next step of the Central Dogma. Proteins are often considered to be the workhorse of the cell, as they are involved is nearly every biochemical process.

The entire catalog of an organism's RNA and protein is called the transcriptome and proteome, respectively. In principle, according to the Central Dogma, we only need data on an organism's genome, as the organism's transcriptome and proteome is specified by the genome. However, since the mechanisms involved in the Central Dogma are not yet fully understood, data at the transcriptome and proteome level are generated directly using, for example, RNA sequencing (RNA-seq) and mass spectrometry to sequence RNA and protein, respectively. Thus a large abundance of data is being generated by molecular biologists at all levels of the Central Dogma.

One approach to analyzing the abundance of biological data is to use the methods of machine learning. In particular, machine learning methods that solve the classification problem are very important. Classification is the problem of taking an object and assigning it to one of a finite number of classes on the basis of a training set of objects of known class identity. In general, classification has many applications including the identification of people based on photographed faces [27], recognition of words in handwriting [21] or speech [17], and e-mail spam detection [6]. In bioinformatics, classification applications include cancer classification using microarrays [9], prediction of protein localization sites [13], protein fold recognition [8], and identification of the kinase that acts upon a protein phosphorylation site [3]. Due to the broad range of important applications of classification methods, the classification problem has received a huge amount of interest in the machine learning research community.

A common approach to solving the classification problem is to transform the data into a form more easily solved. For example, it is easier to find a linear decision boundary between the two classes in the binary classification problem. Therefore the data in its original state, which is called the observation space, is transformed non-linearly into a new space called the feature space, where the data has been unraveled in such a way that makes

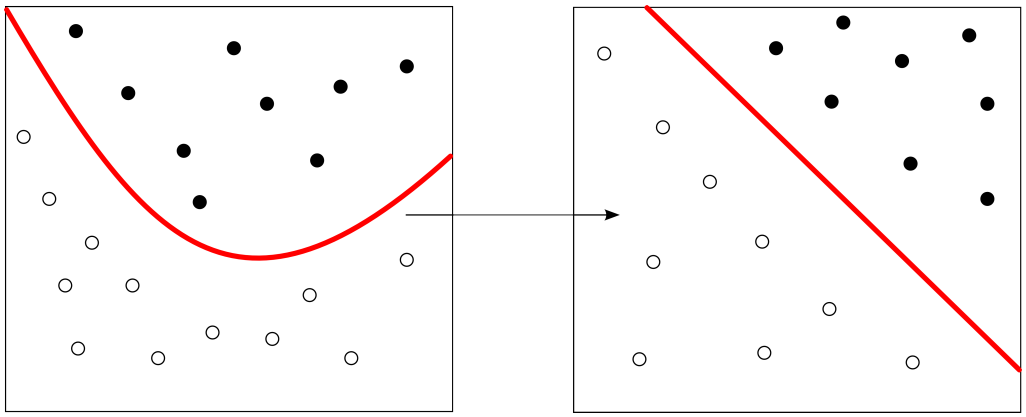


Figure 1.1. Non-linear transformation of data from observation space to a feature space. The transformation is chosen such that a satisfactory linear solution can be found.

finding a satisfactory linear boundary possible, as illustrated in Figure 1.1. However, this non-linear transformation, often to a higher dimensional space, may lead to computationally complex algorithms. This is particularly true if the non-linear transformation is learned from the data, such as with multilayer perceptron (MLP) models [1], as illustrated in Figure 1.2. The first layer is the given object in the observation space, hidden layers represents the non-linear transformation, and the final layer is the computation of the classification of the object in the feature space. Another issue this example illustrates is the difficulty of model selection. How many hidden layers should be used? How many artificial neurons should be used in each layer? These are only a few of the difficult questions that a designer of an MLP faces and often the solution is for the designer to experiment with a variety of configurations to find what works satisfactorily.

A popular solution to the dimensionality issue is to avoid direct computation in the higher dimensional feature space via the so-called kernel trick. The idea is to base the algorithm that determines the unknown parameters of the classifier model, called the learning algorithm, on the kernel function that represents the inner product of the space. In many cases the kernel function can be computed directly without need to compute the non-linear transformation into the feature space. This thereby avoids direct computation in high dimensional spaces and even allows for learning algorithms to use kernel functions that induce infinite-dimensional feature spaces. This also mitigates the designer's difficulty of determining what non-linear transformation is appropriate for their problem. The inner

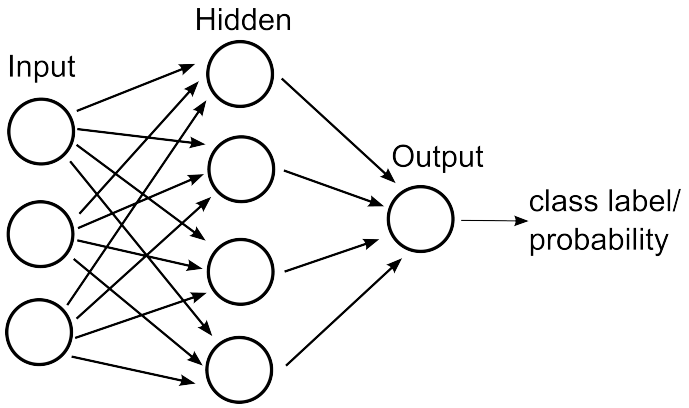


Figure 1.2. A multilayer perceptron with one hidden layer.

product of two vectors in a space is related to the angle between the two vectors, which is like a measure of similarity. Therefore kernel functions are often based on some measure of similarity between two objects. Typically it is not very difficult to design a measure of similarity for the objects being studied and thus designing a kernel function tends to be a simpler task than designing the architecture of the MLP. Methods that use the kernel trick are called kernel methods and have increased in popularity in the past decade due to the Support Vector Machine (SVM) [7].

The SVM model defines a kernel function centered on every observation in the training set and its learning algorithm prunes kernel functions deemed not essential to solving the classification problem. Such a method that prunes out the majority of kernel functions in the model is called a sparse kernel method and a preference for sparse models is motivated by the need to avoid overfitting and by the principle of Occam's razor that states simpler models are preferred. However the SVM is often not sparse enough, leading to classification models that are large in size. Not only is this not desirable from the viewpoint of Occam's razor, but also a large model size leads to longer execution run-times for predictions, which is an issue in large-scale analyses. A second issue is that the SVM predictions are "hard decisions," meaning the SVM simply tells you the predicted class of the given object but not the uncertainty of the prediction. This is an issue as users of classification applications may want to use uncertainty information in order to more effectively make decisions based on the results. This is particularly true in science where a false pos-

itive could lead to costly but unfruitful experiments and in medicine where the decisions may be critical to the survival of a patient.

These issues are addressed by the Relevance Vector Machine (RVM) [23] that has a kernel model similar to the SVM but outputs the probabilities of each class given the object, thereby providing information about the uncertainty of the predictions. Furthermore, the RVM often provides smaller models that are a fraction of the size of equivalent SVM models, leading to faster prediction run-times. However, this comes at the cost of a significantly more computationally complex learning algorithm compared to the SVM due to the complex Bayesian method of pruning the kernel functions. This has been the major obstacle that prevents the RVM from widespread use.

In this dissertation, we propose novel probabilistic kernel-based learning methods similar to the RVM that overcomes the major obstacle, while still maintaining many of the RVM's desirable properties. The practical applicability of the proposed method is shown through some applications to major bioinformatics problems at the transcriptome and proteome levels where the resulting tools are expected to help biologists further elucidate the important information contained in their data.

1.2 Problem Statement

This dissertation examines novel probabilistic kernel-based empirical Bayesian classification methods of small model size and empirically characterizes their performance in terms of both computational complexity and predictive capability. The methods are applied to important bioinformatics problems of identifying certain biochemical regulation mechanisms at the transcriptome and proteome levels of molecular biology.

1.3 Research Description

This dissertation will be presented as a series of published peer-reviewed articles and a manuscript to be submitted for review. A novel probabilistic kernel-based binary classification method called the Classification Relevance Units Machine (CRUM) is specified in Article 1, along with empirical evaluation of its performance compared to SVM

and RVM on benchmarking datasets. However, upon application to a larger dataset in the Article 2, it was found that the supervised learning algorithm specified in Article 1 did not scale satisfactorily. Thus a hybrid unsupervised and supervised learning algorithm for the CRUM is specified to mitigate this problem in Article 2. Using the faster learning algorithm, the problem of protein phosphorylation site prediction is examined.

In Article 3, the binary CRUM algorithm is extended to the general n-class case using the framework of error correcting output codes. This Multiclass Relevance Units Machine (McRUM) is evaluated on benchmarking datasets and preliminarily applied to the problem of detecting particular classes of small ncRNA. In Article 4, we continue investigating the small ncRNA problem to improve upon the preliminary results in Article 3 by further consideration of the biological aspects of small ncRNA and enhancements to McRUM.

1.4 Related Work

This section examines prior research in protein phosphorylation site prediction and small ncRNA prediction.

1.4.1 Protein Phosphorylation Site Prediction

The classical view of the Central Dogma of Molecular Biology states that genetic information flows from DNA to RNA to protein. The process of creating RNA from DNA is called transcription, while the process of creating protein from RNA is called translation. However, as stated previously, not all RNAs code for proteins. Thus they do not follow the classical view of the dogma. Some classes of these non-coding RNA will be further discussed in Section 1.4.2. The class of RNA that code for proteins is called mRNA. In translation, a molecular machine called the ribosome "reads" the amino acid sequence encoded in the mRNA and helps physically assemble the amino acid sequence specified, resulting in a protein.

A post-translational modification of a protein is any modification made to the protein after translation, such as the attachment of further biochemical molecules. These

modifications, which may be reversible, allows the structure and function of the protein to be dynamic. A prominent post-translational modification is phosphorylation, which plays an essential role in a wide range of cellular processes, such as metabolism, immune response, apoptosis, and cellular motility [24]. In phosphorylation, a class of proteins called kinases attach a phosphate group to some designated amino acid, which is called the phosphorylation site, in the target protein. This process is reversible, as a class of proteins called phosphatases can remove the phosphate group from the phosphorylation site. In the simplest case, one of these states, phosphorylated or unphosphorylated, set the protein to be biochemically active, while the other state leaves the protein inactive. The mechanism is more complicated and flexible in the case where a protein has multiple phosphorylation sites.

For example, protein 53 (p53) is a tumor suppressor protein with a multitude of phosphorylation sites and it functions to inhibit cell growth when activated. The key event that leads to the activation of p53 is the phosphorylation of its N-terminal domain that contains numerous phosphorylation sites. Inactivated p53 due to, for example, its mutation or dysfunction of its regulation mechanisms is found in most tumors [16].

As p53 demonstrates, determining which sites of a protein is phosphorylatable is important in the understanding of the regulation of biochemical processes, such as those related to diseases [4]. However, experimental discovery of phosphorylation sites is an expensive and laborious task with many technical challenges [24]. To help mitigate this problem, computational approaches that identify highly probable phosphorylation sites in the abundance of protein sequence data generated by genomics projects, for example, have been an active area of research.

Crystallization studies show that a region of the protein substrate of about 7 to 12 amino acids in length surrounding the acceptor residue contacts the kinase active site [22]. Therefore almost all protein phosphorylation site prediction tools takes the amino acid sequence in a window surrounding a potential site as the primary predictive feature. For example, NetPhos [2] uses a window of length between 9-11 amino acids long, depending on the type of phosphorylation site. NetPhos numerically encodes the window's sequence as a vector using sparse encoding. In sparse encoding, each position in the window's sequence is represented with a 21 bit value that consists of zeros for all but one bit. Thus there

are 21 possible values, representing the 20 different types of amino acids and one value to represent the absence of an amino acid. These numerically encoded windows serve as the input to a MLP for binary classification.

Another example is DISPHOS [15] that employs logistic regression to distinguish phosphorylation sites using information on the predicted 3D structure of the protein. There are four aspects to the 3D structure of a protein: 1) primary structure, the amino acid sequence, 2) secondary structure, local interaction caused by hydrogen bonds, 3) tertiary structure, the overall shape of a protein caused by nonlocal interactions, and 4) quaternary structure, the formation of a protein complex via interaction of multiple proteins. Proteins are not rigid and may dynamically switch to different conformations via post-translation modifications like phosphorylation, for example. In addition to the sparse encoding of the window surrounding a possible phosphorylation site, DISPHOS uses three secondary structure predictors to detect local helix, sheet, and loop structures. Also, some areas of the protein may not settle into a stable 3D structure and remain in an unfolded state. In DISPHOS, these intrinsically disordered regions are computationally predicted using five different predictors and are used as input. Such protein structure information is useful, as phosphorylation sites are frequently contained in disordered protein regions.

However, the use of protein structure predictions is computationally expensive and so a simpler method with at least near comparable results is desirable for large scale analyses, such as making predictions over an organism's whole proteome. In this dissertation, we approach the protein phosphorylation site prediction problem using the CRUM due to its compact and parsimonious models.

1.4.2 Small ncRNA Prediction

ncRNA are functional RNAs that do not code for and are not translated into proteins. The different varieties of ncRNAs have a diverse set of important functions and come in many physical sizes. New classes of ncRNA continue to be discovered [10] and the functions of many classes of ncRNA, such as long ncRNAs, remain largely uncharacterized [12]. An important class of ncRNAs is the small ncRNAs that are roughly 20 to 30 nucleotides long. In this dissertation, we focus on microRNA (miRNA) and piwi-

interacting RNA (piRNA) that are important in post-transcriptional gene regulation. Post-transcriptional regulation refers to the regulation of RNA and includes the blocking of the translation of a mRNA to protein. In contrast to phosphorylation that regulates the activation of a protein, post-transcriptional regulation can prevent the protein from being manufactured in the first place.

The biogenesis of miRNA is relatively well-understood. At the final stages of miRNA biogenesis, a precursor-miRNA (pre-miRNA) is generated that has a distinctive hairpin structure. An enzyme called Dicer cleaves off the loop of the hairpin, leaving the double stranded stem that is historically called the miRNA:miRNA* duplex. The duplex is separated and the mature miRNA strand, in combination with Argonaute proteins, forms the RNA-induced silencing complex (RISC) [14]. There are cases where the miRNA* strand itself is a mature miRNA and forms its own RISC. Using the miRNA sequence as a guide, this RISC binds to mRNA to regulate the mRNA's stability and translation to protein [11]. Therefore miRNAs are implicated in a wide variety of biochemical processes. Unusual changes in miRNA expression can cause dysregulation of important biochemical pathways, implicating the involvement of miRNAs in diseases. Thus the discovery of novel miRNA is an important step in the understanding of these processes.

On the other hand, the biogenesis of piRNA is not fully understood. However it is known that piRNA provides RNA silencing functionality and, in a manner similar to miRNA, the mature piRNA forms a RISC with piwi proteins. They are known to defend against transposons [20], which are DNA sequences that may create mutations harmful to its hosts. Therefore the discovery of novel piRNA is also important.

The discovery of novel miRNA and piRNA is made more possible because of recent advances in high-throughput sequencing technologies that allow biologists to sequence the small RNA content of a tissue or organism. Often small RNA sequencing leads to millions of reads of short length. While it is relatively simple to identify known mature miRNA and piRNA among the reads via comparison to ncRNA databases such as miRBase [18] and NONCODE [5], the discovery of novel mature miRNA and piRNA is non-trivial and most of the available methods are dependent on the availability of the organism's genome sequence and the quality of its annotation. In the worst-case, no genomic information is available and the identification of mature miRNA and piRNA must be based

solely on the RNA reads. This also implies that the longer precursor forms of miRNA and piRNA are not available, as the small RNA sequencing protocol would filter longer RNAs out of the sample. This renders most discovery methods, which rely on the identification of the precursors [14], as not appropriate.

There are a few existing methods that apply to this "no genome" situation. For piRNA, a binary classifier called piRNApredictor has been proposed that uses Fisher's linear discriminant analysis of the nucleotide composition of the reads [26]. Nucleotide composition is analogous to the "bag of words" used in document classification. K-mers, the "words", refer to all possible nucleotide sequences of length K nucleotides. In the case of piRNApredictor, K ranges from one to five. Using only the frequency of K-mers found in the query RNA sequence, piRNApredictor predicts whether or not the query is a piRNA. However, our preliminary analysis suggests that the accuracy of piRNA prediction can be significantly improved upon by using more sophisticated classification methods.

For miRNA, there are two existing approaches: miRMiner [25] and miRPlex [19]. miRMiner relies on evolutionary conservation, which is the idea that related species will have similar sets of miRNAs. The idea of miRMiner is to find potential orthologs of a potential novel miRNA in related species whose genomic information is better characterized. The orthologs are treated as potential pre-miRNA and are folded to see if the distinctive hairpin structure can be formed. In short, the lack of genomic information is overcome by using the genomic information of related species as a proxy. However, the weakness of this approach is the inability to discover weakly- or non-conserved miRNA.

In contrast, miRPlex does not require genomic information for closely related species at all. Instead, it relies on the assumption that both the miRNA and miRNA* of the miRNA:miRNA* duplex would be sequenced. Therefore the existence of a strongly complementary read in the small RNA sequencing dataset would imply that the read is a miRNA. The primary weakness of this approach, however, is that the pairwise folding of millions of reads prohibits the use of this method in a large dataset.

In this dissertation, we propose an approach that addresses these weaknesses. Our approach uses McRUM to predict mature miRNA and piRNA based only on the nucleotide composition of the read, thereby avoiding the conservation assumption and the costly pairwise folding of the reads. Our approach can be considered complementary to miRPlex.

Our method can be used to identify reads that are likely to be miRNAs and this reduced set of reads can be used as a more feasible input for miRPlex, which will provide structural evidence and thus more confidence in the predictions.

1.5 Expected Contributions

In summary, the expected contributions of this dissertation include:

- Novel probabilistic kernel-based binary and multiclass classification methods that leads to significantly more compact and parsimonious models than existing methods. This in turn results in a prediction algorithm of lower computational complexity. This method will be applicable to a wide variety of classification problems, particularly with problems where probabilistic outputs is required and/or where predictions need to be made at a large scale.

- A probabilistic approach to the problem of protein phosphorylation site prediction that uses only sequence information for fast prediction run-times. This will allow biologists to perform predictions at a large scale, such as a whole proteome, to help elucidate biochemical processes.

- A fast probabilistic approach to the problem of detecting mature miRNA and piRNA in small RNA sequencing projects using nucleotide composition data. This will allow biologists to perform predictions on the entire small RNA sequencing results, which are often large, to help them understand how organisms control the translation of proteins and defend themselves from certain mutations.

1.6 Structure of Dissertation

The dissertation is organized into the following chapters:

- Chapter 2 presents Article 1 that precisely describes the CRUM model and supervised learning algorithm for the binary classification problem. The results of experimental evaluations using benchmark datasets are also discussed.

- Chapter 3 presents Article 2 that precisely describes the hybrid unsupervised and supervised learning algorithm for the CRUM. This revised CRUM is then applied to the important problem of protein phosphorylation site prediction. The results of experimental evaluations of this application are discussed.

- Chapter 4 presents Article 3 that precisely describes the extension of the CRUM method of binary classification to the McRUM method of multiclass classification. The results of experimental evaluations of the McRUM on benchmark datasets are discussed. Furthermore, an application to the detection of mature miRNA and piRNA in small RNA sequencing datasets is described and evaluated experimentally.

- Chapter 5 presents a draft of Article 4. We further investigate the small ncRNA problem in Article 3 to increase the predictive performance of our solution with further consideration of the biology of the problem and further modifications of the McRUM method.

- Chapter 6 concludes the dissertation.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] N. Blom, S. Gammeltoft, and S. Brunak, "Sequence and structure-based prediction of eukaryotic protein phosphorylation sites," *Journal of Molecular Biology*, vol. 294, pp. 1351–1362, 1999.

[3] N. Blom, T. Sicheritz-Ponten, R. Gupta *et al.*, "Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence," *Proteomics*, vol. 4, pp. 1633–1649, 2004.

[4] R. I. Brinkworth, R. A. Breinl, and B. Kobe, "Structural basis and prediction of substrate specificity in protein serine/threonine kinases," *PNAS*, vol. 100, pp. 74–79, 2003.

[5] D. Bu, K. Yu, S. Sun, C. Xie, G. Skogerb, R. Miao, H. Xiao, Q. Liao, H. Luo, G. Zhao, H. Zhao, Z. Liu, C. Liu, R. Chen, and Y. Zhao, "Noncode v3.0: integrative annotation of long noncoding rnas," *Nucleic Acids Research*, 2011.

[6] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 423–430.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, 1995.

[8] C. H. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[9] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[10] E. J. Gardner, Z. F. Nizami, C. C. Talbot, and J. G. Gall, "Stable intronic sequence rna (sisrna), a new class of noncoding rna from the oocyte nucleus of xenopus tropicalis," *Genes & development*, vol. 26, no. 22, pp. 2550–2559, 2012.

[11] A. Grimson, K. K.-H. Farh, W. K. Johnston, P. Garrett-Engele, L. P. Lim, and D. P. Bartel, "Microrna targeting specificity in mammals: determinants beyond seed pairing," *Molecular cell*, vol. 27, no. 1, pp. 91–105, 2007.

[12] M. Guttman and J. L. Rinn, "Modular regulatory principles of large non-coding rnas," *Nature*, vol. 482, no. 7385, pp. 339–346, 2012.

[13] P. Horton and K. Nakai, "Better prediction of protein cellular localization sites with the it k nearest neighbors classifier," in *Ismb*, vol. 5, 1997, pp. 147–152.

[14] Y. Huang, Q. Zou, S. Wang, S. Tang, G. Zhang, and X. Shen, "The discovery approaches and detection methods of micrornas," *Molecular Biology Reports*, vol. 38, pp. 4125–4135, 2011.

[15] L. M. Iakoucheva, P. Radiojac, C. J. Brown *et al.*, "The importance of intrinsic disorder for protein phosphorylation," *Nucleic Acids Research*, vol. 32, pp. 1037–1049, 2004.

[16] L. M. M. Jenkins, S. R. Durell, S. J. Mazur, and E. Appella, "p53 n-terminal phosphorylation: a defining layer of complex regulation," *Carcinogenesis*, vol. 33, no. 8, pp. 1441–1449, 2012.

[17] B.-H. Juang, W. Hou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 5, no. 3, pp. 257–265, 1997.

[18] A. Kozomara and S. Griffiths-Jones, "mirbase: integrating microrna annotation and deep-sequencing data," *Nucleic Acids Research*, vol. 39, no. suppl 1, pp. D152–D157, 2011.

[19] D. Mapleson, S. Moxon, T. Dalmay, and V. Moulton, "Mirplex: A tool for identifying mirnas in high-throughput srna datasets without a genome," *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, vol. 320, no. 1, pp. 47–56, 2013.

[20] K. A. O'Donnell and J. D. Boeke, "Mighty piwis defend the germline against genome intruders," *Cell*, vol. 129, no. 1, pp. 37–44, 2007.

[21] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63–84, 2000.

[22] Z. Songyang, S. Blechner, N. Hoagland *et al.*, "Use of an oriented pepetide library to determine the optimal substrates of protein kinases," *Current Biology*, vol. 4, pp. 973–982, 1994.

[23] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[24] B. Trost and A. Kusalik, "Computational prediction of eukaryotic phosphorylation sites," *Bioinformatics*, vol. 27, no. 21, pp. 2927–2935, 2011.

[25] B. M. Wheeler, A. M. Heimberg, V. N. Moy, E. A. Sperling, T. W. Holstein, S. Heber, and K. J. Peterson, "The deep evolution of metazoan micrornas," *Evolution & development*, vol. 11, no. 1, pp. 50–68, 2009.

[26] Y. Zhang, X. Wang, and L. Kang, "A k-mer scheme to predict pirnas and characterize locust pirnas," *Bioinformatics*, vol. 27, no. 6, pp. 771–776, Mar. 2011.

[27] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng, "Discriminant analysis of principal components for face recognition," in *Face Recognition*. Springer, 1998, pp. 73–85.

Chapter 2

Relevance Units Machines for Classification

Abstract

Classification, a task to assign each input instance to a discrete class label, is a prevailing problem in various areas of study. A great amount of research for developing models for classification has been conducted in machine learning research and recently, kernel-based approaches have drawn considerable attention mainly due to their superiority on generalization and computational efficiency in prediction. In this work, we present a new sparse classification model that integrates the basic theory of a sparse kernel learning model for regression, called relevance units machine, with the generalized linear model. A learning algorithm for the proposed model will be described, followed by experimental analysis comparing its predictive performance on benchmark datasets with that of the support vector machine and relevance vector machine, the two most popular methods for kernel-based classification.

2.1 Introduction

Classification is one of the most studied problems in machine learning research and one that has kept a constant momentum in terms of research interest and theoretical novelty. Classification problems are ubiquitous in various fields of study including bioinformatics. Examples of bioinformatics classification problems are cancer classification using microarrays [11], prediction of protein localization sites [13], protein fold recognition [8], and the prediction of protein phosphorylation sites [2], just to list a few.

The main purpose of classification algorithms is to learn a classifier that assigns each input instance to one of a finite number of class labels. Over the last several decades, many classification algorithms have been proposed, such as decision trees [4, 17, 18], k-nearest neighbors [7], artificial neural networks [25], discriminant analysis [10], Bayesian networks [14, 16], and the SVM [23].

Recently, considerable attention has focused on kernel-based learning approaches in supervised classification where they have shown state-of-the-art performance. A prediction model produced by kernel-based learning is often sparse, depending only on a subset of the kernel basis functions associated with some of the training samples. Of particular interest is the SVM, a sparse linearly-parameterized model, which has demonstrated great successes. The basic idea of the SVM is to find the optimal hyperplane decision boundary between the two data classes. By optimal, it means that the distance from the closest training examples to the separating hyperplane should be the largest possible, i.e. the margin between the two classes is maximized. Therefore, SVM minimizes the risk of overfitting and has potential for better classification accuracy from small training sets compared to other approaches to classification.

However, despite the current popularity and success, a number of significant and practical concerns of the SVM methodology have been identified: 1) The error/margin trade off term has to be selected often through a cross-validation procedure, which is a waste of both computation and data. 2) Although relatively sparse, the number of support vectors required tends to increase linearly to the size of the training set. 3) The prediction is not probabilistic, therefore it makes a hard decision [21].

The RVM, a Bayesian extension of the SVM, is a sparse learning algorithm that does not suffer from the limitations alluded to above [21, 22]. The RVM is capable of generating a fully probabilistic output and it has been shown empirically that, when solving the same problems, the RVM uses fewer relevance vectors than support vectors, making the classifier much sparser than the SVM-based one. This can lead to significant reduction in computational complexity of classification, while maintaining nearly identical performance to, if not better than, that of the SVM [5, 21, 24].

In this paper, we present a new classification model developed based on a sparse kernel regression model called the relevance units machine (RUM), a recent variation of the RVM [12]. While RVM finds relevance vectors from the training samples, analogous to the support vectors in the SVM, those vectors in the RUM are not necessarily from the training data. Those vectors, called relevance units, are in fact learned during the training. The RUM also adopts the Bayesian inference framework to automatically learn all the parameters including the kernel parameters.

Based on the experimental results on regression problems, it has been argued that the RUM can generate an even sparser predictor than the RVM while retaining comparable performance [12]. However, the RUM learning framework has not been introduced for classification, therefore the development of RUM-based classifiers would provide insights for improving existing state-of-the-art classification methods. In this work, we present a RUM based classification model and evaluate its predictive performance through experimental studies with benchmark datasets.

In the next section, we specify a classification model that combines the RUM for regression with the generalized linear model. Assumptions on the parameters and the underlying distributions for prediction and the (hyper)priors are described in Section 2.3, followed by the derivation of the learning algorithm for the proposed model in Section 2.4. Section 2.5 presents experimental studies and analyzes the results of benchmark comparisons with the SVM and RVM. Finally, we conclude our work with discussions on advantages and shortcomings of the current model, and future prospect of overcoming the limitations.

2.2 Model

We assume that the data is a set of N independent input vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^q$, along with their corresponding targets $\mathbf{t} = [t_1, t_2, \ldots, t_N]^T$. Furthermore, we assume that the target values are binary, $t_i \in \{0, 1\}$. We use a model that combines the RUM for regression [12] with the generalized linear model, giving

$$\hat{t}(\mathbf{x}) = f\left(w_0 + \sum_{m=1}^{M} w_m k(\mathbf{x}, \mathbf{u_m}; \boldsymbol{\Theta})\right) \tag{2.2.1}$$

where $1 \leq M \leq N$, $f(\cdot)$ is an activation function, $k(\cdot, \cdot; \boldsymbol{\Theta})$ is a kernel function with a set of parameters $\boldsymbol{\Theta}$, the weights $\mathbf{w} = [w_0, w_2, \ldots, w_M]^T$, and the units $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$ with each $\mathbf{u}_i \in \mathbb{R}^q$. We call this the Classification Relevance Units Machine (CRUM) model.

The problem is to estimate from the data $(X, \mathbf{t})$ appropriate values of $\mathbf{w}$, $\mathbf{U}$, and $\boldsymbol{\Theta}$ to make good predictions $\hat{t}(\mathbf{x})$ for a fixed M. Typically $M \ll N$ to achieve a sparse solution. We use the logistic sigmoid function, $\sigma(a) = (1 + e^{-a})^{-1}$, as the activation function from here on.

2.3 Assumptions

Without loss of generality, we assume $w_0 = 0$ and set $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$. Define

$$y_i = \sum_{m=1}^{M} w_m k(\mathbf{x}_i, \mathbf{u}_m; \boldsymbol{\Theta}) \tag{2.3.1}$$

and the design matrix $\mathbf{K}$ with elements

$$K_{ij} = k(\mathbf{x}_i, \mathbf{u}_j) \tag{2.3.2}$$

for $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}$.

We adopt a Bernoulli distribution for $P(t|\mathbf{x}, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})$ as done with the RVM for classification [21], giving the following likelihood of the data $(X, \mathbf{t})$ given the model:

$$P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) = \prod_{n=1}^{N} \sigma(y_n)^{t_n}[1 - \sigma(y_n)]^{1-t_n} \tag{2.3.3}$$

A maximum likelihood estimate can be obtained by finding the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ that maximizes the likelihood function (3.3.4). However we specify further assumptions to add regularization terms and obtain a maximum a posteriori (MAP) estimate to mitigate overfitting.

We adopt a zero-mean isotropic Gaussian prior distribution over $\mathbf{w}$ to give preference to simple models,

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{2.3.4}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix and $\alpha \in \mathbb{R}^+$ is the hyperparameter controlling the precision of the Gaussian over the weights. Note that this is not the Automatic Relevance Determination prior used in the RVM [21]. Driving w_i values to zero is not a goal since sparsity is instead achieved by selecting a small value for M.

Similarly we specify a Gaussian prior over $\mathbf{U}$ with

$$p(\mathbf{U}|r) = \left(\frac{r}{2\pi}\right)^{Mq/2} \exp\left\{-\frac{r}{2}\mathrm{tr}[\mathbf{U}^T\mathbf{U}]\right\} \tag{2.3.5}$$

where $r \in \mathbb{R}^+$ is the hyperparameter that controls the precision of the prior.

We assume Gamma distributed hyperpriors for the hyperparameters α and r

$$p(\alpha) = \mathrm{Gamma}(\alpha|a, b) \tag{2.3.6}$$

$$p(r|c, d) = \mathrm{Gamma}(r|c, d) \tag{2.3.7}$$

where $\mathrm{Gamma}(x|a, b) = \Gamma(a)^{-1}b^a x^{a-1}e^{-bx}$ and $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$. The a, b, c, and d parameters are fixed to small values to make the hyperpriors non-informative [21].

We obtain the posterior distribution by combining all our assumptions (3.3.4)-(3.3.6),

$$p(\mathbf{t}, \mathbf{U}, \mathbf{w}, \alpha, r|X, \mathbf{\Theta}) = \\ P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{U}|r)p(r)p(\mathbf{w}|\alpha)p(\alpha) \tag{2.3.8}$$

Let $\hat{t}_n = \hat{t}(\mathbf{x}_n)$ and $\mathbf{B} = \mathrm{diag}(\hat{t}_1(1-\hat{t}_1), \hat{t}_2(1-\hat{t}_2), \ldots, \hat{t}_N(1-\hat{t}_N))$. We can marginalize $\mathbf{w}$ from (2.3.8) using a Laplace approximation [1] of the integral, as is done in the RVM, to obtain:

$$\begin{aligned} &p(\mathbf{t}, \mathbf{U}, \alpha, r|X, \mathbf{\Theta}) = \\ &\int P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{w}|\alpha)p(\mathbf{U}|r)p(r)p(\alpha)d\mathbf{w} \\ &\approx (2\pi)^{M/2}|\mathbf{\Sigma}|^{1/2}P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}^*, \mathbf{\Theta})p(\mathbf{w}^*|\alpha)p(\mathbf{U}|r)p(r)p(\alpha) \end{aligned} \tag{2.3.9}$$

where the covariance of the Laplace approximation is

$$\mathbf{\Sigma} = (\mathbf{K}^T\mathbf{B}\mathbf{K} + \alpha\mathbf{I})^{-1} \tag{2.3.10}$$

and its mean $\mathbf{w}^*$ is the $\mathbf{w}$ that maximizes $P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{w}|\alpha)$.

The MAP estimate of the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ are those that maximize (2.3.9).

2.4 Learning Algorithm

We define our objective function as

$$L = -\ln p(\mathbf{t}, \mathbf{U}, \alpha, r|X, \mathbf{\Theta}) \tag{2.4.1}$$

and we use methods from unconstrained optimization for continuous functions to find the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ that minimize L. Typically such methods require computing the gradient of L that we will provide here.

To begin, we need to compute the mean $\mathbf{w}^*$ for the Laplace approximation. This is the $\mathbf{w}$ that maximizes

$$\ln[P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{w}|\alpha)] = \sum_{n=1}^{N}[t_n \ln \hat{t}_n + (1 - t_n) \ln(1 - \hat{t}_n)] - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const} \tag{2.4.2}$$

where $\hat{t}_n$ is the prediction on $\mathbf{x}_n$ using the current model estimates. The first term is the log-likelihood and second is the regularization term. Let $\hat{\mathbf{t}} = [\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N]^T$. Then the vector $\mathbf{w}^*$ can be computed by maximizing (2.4.2) with respect to $\mathbf{w}$ using the Newton-Raphson method with gradient

$$\mathbf{K}^T(\mathbf{t} - \hat{\mathbf{t}}) - \alpha\mathbf{w} \tag{2.4.3}$$

and the Hessian

$$-\mathbf{K}^T\mathbf{B}\mathbf{K} - \alpha\mathbf{I} \tag{2.4.4}$$

Now that we have $\mathbf{w}^*$ we can compute the partial derivatives of L as follows,

$$\frac{\partial L}{\partial \alpha} = -\frac{1}{2}\left(\frac{M}{\alpha} - \mathbf{w}^{*T}\mathbf{w}^* - \text{tr}[\mathbf{\Sigma}]\right) \tag{2.4.5}$$

$$\frac{\partial L}{\partial U_{ij}} = rU_{ij} - \sum_{n=1}^{N}(t_n - \hat{t}_n)w_j^*\frac{\partial k(\mathbf{x}_n, \mathbf{u}_j; \mathbf{\Theta})}{\partial U_{ij}} \tag{2.4.6}$$

$$\frac{\partial L}{\partial r} = \frac{1}{2}\text{tr}[\mathbf{U}^T\mathbf{U}] - \frac{Mq}{2r} \tag{2.4.7}$$

$$\frac{\partial L}{\partial \mathbf{\Theta}} = \sum_{n=1}^{N}(\hat{t}_n - t_n)\text{tr}\left[\mathbf{w}^{*T}\frac{\partial \mathbf{k}(\mathbf{x}_n)}{\partial \mathbf{\Theta}}\right] \tag{2.4.8}$$

where $\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{u}_1; \mathbf{\Theta}), k(\mathbf{x}, \mathbf{u}_2; \mathbf{\Theta}), \ldots, k(\mathbf{x}, \mathbf{u}_M; \mathbf{\Theta})]^T$. Note that the terms in L

associated with $p(\alpha)$ and $p(r)$ were dropped by assuming that they are non-informative. Also the term in L associated with $|\mathbf{\Sigma}|^{1/2}$ was dropped for simplicity in (2.4.6) and (2.4.8). Therefore these are actually the partials of the negative log of (2.3.8) with respect to U_{ij} and $\mathbf{\Theta}$, respectively. We observe good predictive performance despite these simplifications.

We chose to implement this minimization using an alternating variables method where only one of the parameters or hyperparameters ($\mathbf{w}$, α, $\mathbf{U}$, r, or $\mathbf{\Theta}$) is optimized at a time while holding all other parameters and hyperparameters fixed. This method cycles through each parameter and hyperparameter estimator repeatedly until the estimates converge. While this approach leads to slower convergence due to the ignorance of any correlation between the parameters and hyperparameters [9], this simplification allows us to work with smaller Hessian matrices. Also we have observed lower predictive performance with estimates gained using a trust region Newton method [6] over all parameters and hyperparameters simultaneously under the same initial values. Therefore it appears, at least on the datasets we have tested, that the alternating variables method leads to better predictive solutions.

The MAP estimate of $\mathbf{w}$ is $\mathbf{w}^*$, obtained using the Newton-Raphson method. The estimates for α and r are obtained by setting their partial derivatives (2.4.5) and (2.4.7) to zero, giving

$$\alpha^* = \frac{M - \alpha \mathrm{tr}[\mathbf{\Sigma}]}{\mathbf{w}^T \mathbf{w}} \tag{2.4.9}$$

$$r^* = \frac{Mq}{\mathrm{tr}[\mathbf{U}^T \mathbf{U}]} \tag{2.4.10}$$

A trust region Newton method [6] is used to obtain the MAP estimates of $\mathbf{U}$ and $\mathbf{\Theta}$ using (2.4.6) and (2.4.8) to compute the partial derivatives of L with respect to $\mathbf{U}$ and $\mathbf{\Theta}$, respectively, and using finite differencing to approximate the Hessian in both cases.

In summary, the pseudocode for the overall algorithm is as follows:

2.5 Experimental Results

For all experiments we used the Gaussian kernel

Input: Matrix of training data $\mathbf{X}$ and its associated vector of class labels $\mathbf{t}$, kernel function k, number of kernel functions M

$\mathbf{w}, \alpha, \mathbf{U}, r, \boldsymbol{\Theta} \leftarrow$ initial values
while $\mathbf{w}, \alpha, \mathbf{U}, r, \boldsymbol{\Theta}$ have not converged **do**
  Compute design matrix $\mathbf{K}$ using (3.3.3)
  $\mathbf{w} \leftarrow$ Newton-Raphson using (2.4.2)-(2.4.4)
  $\alpha \leftarrow$ Equation (3.3.14)
  $\mathbf{U} \leftarrow$ trust region Newton using (2.4.1) and (2.4.6)
  $r \leftarrow$ Equation (2.4.10)
  $\boldsymbol{\Theta} \leftarrow$ trust region Newton using (2.4.1) and (2.4.8)
end while
return $\mathbf{w}, \mathbf{U}, \boldsymbol{\Theta}$

Algorithm 1: CRUM learning algorithm

$$k(\mathbf{x}, \mathbf{u}; \boldsymbol{\Theta} = \{\sigma\}) = \exp\left\{-\frac{1}{\sigma^2}(\mathbf{x}-\mathbf{u})^T(\mathbf{x}-\mathbf{u})\right\} \tag{2.5.1}$$

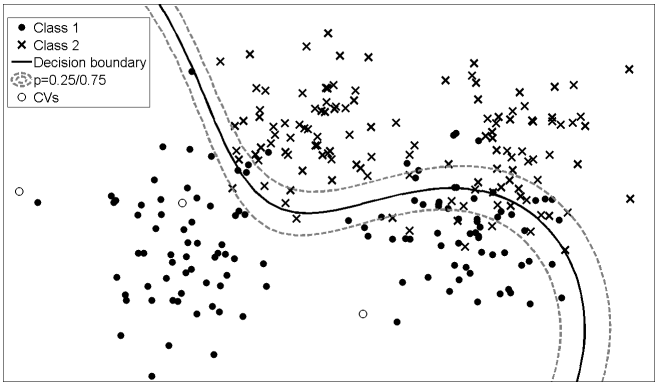
where σ is called the width of the kernel. We will use the term critical vector (CV) to refer to the basis vectors with non-zero w_i terms. The CVs are also called support vectors (SV) in the SVM model ([3,23]), relevance vectors (RV) in the RVM model, and relevance units (RU) in the regression RUM and CRUM models.

2.5.1 Ripley's Synthetic Data

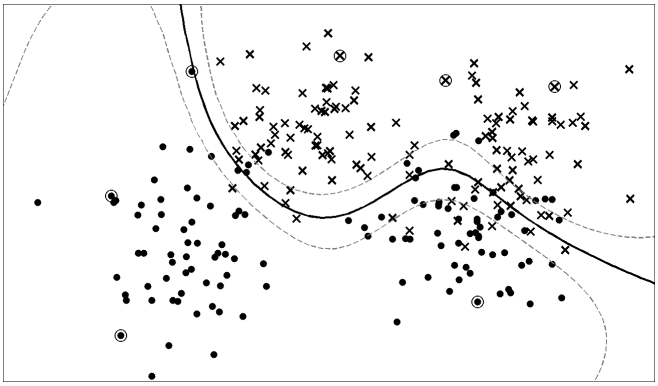
For visualization, we analyze two-dimensional data generated from mixtures of two Gaussians from [20] whose Bayes error is about 8%. For a comparison, both RVM and CRUM models are trained on a set of 250 examples and evaluated on a test set of 1000 examples. A CRUM was estimated with a variety of values of M and it was observed that $M = 3$ achieved the best Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) [1] values. The decision boundary used here is $\hat{t}(\mathbf{x}) = 0.5$. This CRUM model achieves 9.6% error on the test set and is plotted in Figure 2.1a. The RVM model was estimated using a kernel width of 0.5, resulting in 7 CVs and a test error of 9.9%. The RVM

Table 2.1. Benchmarking Results of SVM, RVM, and CRUM

Dataset	N	N_{test}	d	SVM err%	RVM err%	CRUM err%	#SVs	#RVs	#RUs
Pima Diabetes	200	332	8	20.1	19.6	22.0	109	4	2
Banana	400	4900	2	10.9	10.8	13.4	135.2	11.4	7
Breast Cancer	200	77	9	26.9	29.9	28.6	116.7	6.3	2
Titanic	150	2051	3	22.1	23.0	23.4	93.7	65.3	1
Waveform	400	4600	21	10.3	10.9	10.9	146.4	14.6	1
German	700	300	20	22.6	22.2	22.0	411.2	12.5	5
Image	1300	1010	18	3.0	3.9	3.1	166.6	34.6	4



(a) CRUM with $M = 3$



(b) RVM

Figure 2.1. Plots of CRUM and RVM on Ripley's data. The solid curve represents the chosen decision boundary at $\hat{t}(\mathbf{x}) = 0.5$, while the dashed curves represent the boundary at values 0.25 and 0.75. The CRUM and RVM have their 3 and 7 CVs, respectively, indicated by the empty circles.

was generated using the SparseBayes software from `http://www.vectoranomaly.com`. The plotted RVM result is shown in Figure 2.1b. The CRUM achieves slightly better predictive performance with a sparser model than the RVM.

2.5.2 Benchmark Comparisons

We ran a benchmark test to compare the classification results of the CRUM model to the previously published results of the RVM and SVM models on a variety of datasets. All datasets are taken from a collection compiled by Rätsch et al. [19] except the Pima Diabetes dataset which is taken from [20]. In this test, all problems are binary classification

and both RVM and CRUM use a decision boundary at the predicted value of 0.5 in all cases. The benchmark datasets used in the comparison are listed below.

- **Pima Diabetes:** This is a collection of measurements taken from female patients at least 21 years old of Pima Indian heritage, and the problem is to predict the onset of diabetes mellitus.
- **Banana:** A set of two-dimensional samples generated by Rätsch et al. [19].
- **Breast Cancer:** This dataset contains measurements of nine attributes for prediction of the prognosis of breast cancer recurrence.
- **Titanic:** With this dataset, the task is to predict the survival of a passenger of the historical ship Titanic based on some attributes of the individuals.
- **Waveform:** This dataset contains a variety of noisy attributes and the task is to determine the class of the wave.
- **German:** This dataset contains credit data to use to classify a customer as good or bad credit risks.
- **Image:** This is a set of image segmentations described by numeric-valued attributes and the task is to determine what the image segments represent.

Table 2.1 lists the size of training and test datasets, N and N_{test}, data dimension d, the observed error on the independent test dataset, and the number of CVs used in the SVM, RVM, and CRUM models. Unlike the other methods, CRUM requires that we specify M, the number of CVs. Note that what is reported in Table 2.1 is the CRUM model with the best observed AIC, not with the lowest error rate. With the exception of Pima Diabetes, the reported error and number of CVs for the SVM and RVM are averages over 10 training/test splits. However we only report the error and the number of CVs for the CRUM on a single training/test split for computational reasons.

The results indicate that CRUM can achieve sparser (less CVs) solutions than both RVM and SVM with little sacrifice to predictive performance. In particular, there is a significant decrease in the number of CVs in the Titanic, Waveform, and Image datasets

compared to the RVM and SVM. This means the summation in (2.2.1) contains significantly less terms and is therefore more computationally efficient than the equivalent RVM or SVM. This feature is of great importance in running predictions on very large datasets, which is often the case in many biomedical informatics applications.

2.6 Conclusion

The CRUM model has several advantages. The models can be highly sparse and yet lead to very good generalization, as implied by the benchmark comparisons shown in Table 2.1. Additionally, a user can choose to select smaller M to generate more compact models in sacrifice of generalization, if deemed desirable. This is not possible with the RVM, but can be done with the SVM through manipulation of the complexity control parameter. The model is also probabilistic and allows more flexibility than the SVM in choosing an appropriate decision function. The kernel parameters are estimated unlike both the SVM and RVM where they must be specified. An appropriate M still needs to be selected, but this can be done using the AIC or BIC in a process simpler than cross-validation.

While the resulting model is very sparse and leads to a very efficient classifier, the major disadvantage of the CRUM is the computational complexity of the current learning algorithm. For memory complexity, the CRUM requires the storage of an $N \times M$ design matrix and a Hessian matrix of dimension $Mq \times Mq$ in the estimation of $\mathbf{U}$. This means that the space complexity of CRUM is higher than that of the RVM if $Mq > N$. The time complexity of inverting the Hessian matrices used to estimate $\mathbf{w}$ and $\mathbf{U}$, and the approximation of the Hessian itself using finite differencing in the $\mathbf{U}$ estimation are computationally expensive. Potential avenues to reducing the complexity of the algorithm in the future include using more crude but cheaper approximations to the Hessian and the use of other unconstrained optimization algorithms such as L-BFGS [15] that do not explicitly form or store the complete Hessian.

Acknowledgment

This work was supported by U.S. Public Health Services grants P20RR016467 and P20RR018727 from the National Center for Research Resources, National Institutes of Health. The content is solely the responsibility of the authors and do not necessarily represent the official views of the National Center for Research Resources, National Institutes of Health.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] N. Blom, T. Sicheritz-Ponten, R. Gupta *et al.*, "Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence," *Proteomics*, vol. 4, pp. 1633–1649, 2004.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[5] Z. Chen and H. Tang, "Sparse bayesian approach to classification," in *IEEE Proceedings of Networking, Sensing and Control*, 2005, pp. 914–917.

[6] T. F. Coleman and Y. Li, "An interior, trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[7] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[8] C. H. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[9] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Wiley, 2000.

[10] J. H. Friedman, "Regularized discriminant analysis," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

[11] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[12] J. Gao and J. Zhang, "Sparse kernel learning and the relevance units machine," in *PAKDD'09*, 2009.

[13] P. Horton and K. Nakai, "Better prediction of protein cellular localization sites with the it k nearest neighbors classifier," in *Ismb*, vol. 5, 1997, pp. 147–152.

[14] F. V. Jensen, *An Introduction to Bayesian Networks*. Secaucus, NJ: Springer-Verlag New York, Inc., 1996.

[15] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Mathematical Programming B*, vol. 45, no. 3, pp. 503–528, 1989.

[16] M. G. Madden, "The performance of bayesian network classifiers constructed using different techniques," in *Proceedings of the ECML/PKDD-03, Workshop on Probabilistic Graphical Models for Classification*, 2003, pp. 59–70.

[17] S. K. Murphy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, pp. 345–389, 1997.

[18] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann, 1993.

[19] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

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

[21] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[22] M. E. Tipping and A. C. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, C. M. Bishop and B. J. Frey, Eds., 2003.

[23] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[24] X.-M. Xu, Y.-F. Mao, J.-N. Xiong, and F.-L. Zhou, "Classification performance comparison between rvm and svm," in *2007 IEEE International Workshop on Anti-counterfeiting, Security, Identification*, 2007, pp. 208–211.

[25] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

Chapter 3

Probabilistic Prediction of Protein Phosphorylation Sites Using Classification Relevance Units Machines

Abstract

Phosphorylation is an important post-translational modification of proteins that is essential to the regulation of many cellular processes. Although most of the phosphorylation sites discovered in protein sequences have been identified experimentally, the in vivo and in vitro discovery of the sites is an expensive, time-consuming and laborious task. Therefore, the development of computational methods for prediction of protein phosphorylation sites has drawn considerable attention. In this work, we present a kernel-based probabilistic Classification Relevance Units Machine (CRUM) for in silico phosphorylation site prediction. In comparison with the popular Support Vector Machine (SVM), CRUM shows comparable predictive performance and yet provides a more parsimonious model. This is

Reprinted, with permission, from M. Menor, K. Baek, and G. Poisson, "Probabilistic prediction of protein phosphorylation sites using classification relevance units machines," *ACM SIGAPP Applied Computing Review*, vol.12, no.4, pp.8-20, 2012. DOI: 10.1145/2432546.2432547. PMCID: PMC3806113.

This work is based on an earlier work: M. Menor, G. Poisson, and K. Baek, "Probabilistic prediction of protein phosphorylation sites using kernel machines," in *The 27th ACM Symposium on Applied Computing (ACM-SAC 2012) Conference Track on Bioinformatics and Computational Systems Biology (BIO)*, Riva Del Garda Congress Center, Italy, March, pp.1393-1398, 2012. DOI: 10.1145/2245276.2231997.

desirable since it leads to a reduction in prediction run-time, which is important in predictions on large-scale data. Furthermore, the CRUM training algorithm has lower run-time and memory complexity and has a simpler parameter selection scheme than the Relevance Vector Machine (RVM) learning algorithm.

To further investigate the viability of using CRUM in phosphorylation site prediction, we construct multiple CRUM predictors using different combinations of three phosphorylation site features BLOSUM encoding, disorder, and amino acid composition. The predictors are evaluated through cross-validation and the results show that CRUM with BLOSUM feature is among the best performing CRUM predictors in both cross-validation and benchmark experiments. A comparative study with existing prediction tools in an independent benchmark experiment suggests possible direction for further improving the predictive performance of CRUM predictors.

3.1 Introduction

Phosphorylation is one of the most widespread post-translational modifications of proteins, and it plays an essential role in the regulation of many cellular processes, such as metabolism, immune response, apoptosis, and cellular motility [39]. Phosphorylation occurs when a protein kinase (PK) enzyme covalently bonds phosphate groups to particular amino acids in a protein [42]. Phosphorylation is reversible through a process called dephosphorylation, where phosphatase enzymes remove the attached phosphate groups. In the simplest case, one of these states, phosphorylated or unphosphorylated, set the protein to be biochemically active, while the other state leaves the protein inactive. Determining which sites of a protein is phosphorylatable is thus important in the understanding of the regulation of biochemical processes, particularly of diseases, making PKs a major target for drug design [7]. For example, a drug called imatinib is used to treat some cancers via inhibiting an associated PK, preventing the cancer's anti-apoptopic processes from being activated via phosphorylation [13].

There are a variety of ways to discover protein phosphorylation sites experimentally, and the mass spectrometry approach has been one of the most popular in recent years [21]. The experimentally verified phosphorylation sites are listed in the annotations in

a variety of protein databases, including Swiss-Prot [5], Phospho-Base [23], and PhosphoSite [17]. Regardless of the method used, however, experimental discovery of phosphorylation sites is an expensive and laborious task with many technical challenges [39]. To help mitigate this problem, computational approaches that identify highly probable phosphorylation sites have been an active area of research for over ten years. As a result, numerous computational techniques have been proposed [39, 42] with the focus on predicting phosphorylatable serine, threonine, and tyrosine, as these are the types of acceptor residues dominating the databases. Often, biologists apply these methods to narrow down the list of possible phosphorylation sites to be experimentally verified in the proteins that they are studying [39].

The majority of existing phosphorylation site prediction tools use methods from supervised learning. These include the SVM used by KinasePhos [41], Musite [14], and PPRED [2], and a variety of neural network methods used by NetPhosK [4] and GANNPhos [36], among others. These tools differ in their choice of site features and encodings, and their choice of learning algorithms. Moreover, these tools can be grouped into two categories: kinase-independent and kinase-specific. In this work, we focus on kinase-independent predictors. A kinase-independent predictor predicts whether a protein site is phosphorylatable or not, regardless of what PK catalyzed the site. The aforementioned PPRED is an example of such a predictor. Another example is DISPHOS [18] that employs logistic regression to distinguish phosphorylatable sites using information on the predicted 3D structure of the protein. Specifically, DISPHOS uses computational disorder predictors to determine whether a protein site is in an intrinsically disordered protein region that does not fold into a stable 3D structure. Such protein structure information is useful, as phosphorylation sites are frequently contained in disordered protein regions [18].

In addition to predicting whether a site is phosphorylatable or not, a kinase-specific predictor also estimates a candidate PK or type of PK that could cause the phosphorylation. This is particularly important for phosphorylation sites discovered using the popular mass spectrometry method, as the method does not identify the PK that catalyzes the site [39]. To identify the PK, the predictor NetPhosK [4] uses Mutilayer Perceptrons on sequence information to predict 17 common PKs. Due to the large number of PKs (estimated to be greater than 500 in mammals) and very little experimental data on the majority

of them, many predictors instead focus on predicting PK groups or families, rather than individual PKs [42]. Predictors like KinasePhos [41] subdivide the large kinase groups into smaller, more specific groups. Using Maximum Dependence Decomposition to subdivide the large PK groups into smaller groups, KinasePhos learns SVMs on sequence information to make predictions from about 60 PK subgroups.

In this study, we propose a kernel-based learning method, the Classification Relevance Units Machine (CRUM), and present its application to prediction of protein phosphorylation sites. CRUM predictors are kinase-independent, and make probabilistic predictions of phosphorylation sites. Experimental results show that CRUM produces a simpler and yet effective prediction system.

In the next section, we briefly describe the two kernel machines for classification that have been extensively studied in the machine learning community SVM and RVM. We note that RVM as well as CRUM have not been previously used to solve the phosphorylation site prediction problem. Detailed description of our newly proposed CRUM learning model and algorithm follows in Section 3.3. Experimental setup including dataset construction and performance assessment measures is described in Section 3.4, which is followed by presentation of results and analysis of three experiments conducted to assess the effectiveness and the viability of using CRUM in phosphorylation site prediction. Finally, we conclude our work with summary and future prospects of overcoming the limitation and further improving predictive performance of CRUM predictors.

3.2 Kernel Machines

We assume that the data is a set of N d-dimensional vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, along with their corresponding target values $\mathbf{t} = [t_1, t_2, \ldots, t_N]^T$, where $(\mathbf{x}_i, t_i)$ are drawn independently and identically from a distribution. Since we are only considering binary classification, $t_i \in \{0, 1\}$ for the CRUM and RVM, and $t_i \in \{-1, 1\}$ for the SVM. The value of t_i is set to 1 if site $\mathbf{x}_i$ is phosphorylatable, otherwise t_i is set to 0 or -1 for CRUM/RVM and SVM, respectively.

3.2.1 Support Vector Machine

The goal of an SVM is to find the hyperplane decision boundary that perfectly separates the two classes in the training dataset with maximum margin [40]. This means the distance from the closest training point to the hyperplane should be maximized. The SVM has been generalized to allow for overlapping class distributions [10] and non-linear decision boundaries via use of kernel functions [6]. A commonly used kernel function, and the one used in this study, is the Gaussian radial basis function (RBF) kernel with parameter $\gamma > 0$:

$$k(\mathbf{x}, \mathbf{x}') = \exp(\gamma \|\mathbf{x} - \mathbf{x}'\|^2). \tag{3.2.1}$$

To use the SVM, a user must specify values for the parameter C that controls the trade-off between minimizing training error and model complexity, and all the kernel parameters; namely, γ in the case of the Gaussian RBF kernel.

Upon completion of training, we can classify a new data point $\mathbf{x}$ using the following formula,

$$\hat{t}(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{3.2.2}$$

where the w_is and b are learned through the SVM training [6] and $\mathrm{sgn}(x)$ is the sign function that evaluates to 1 if $x < 0$ and 1 if $x > 0$. As a consequence of the SVM training, many of the w_is will vanish. Therefore only a subset of the training data is required to make predictions. The training data $\mathbf{x}_i$ whose associated $w_i > 0$ are called support vectors (SVs).

3.2.2 Relevance Vector Machine

Unlike the SVM, the RVM seeks to model the posterior distribution $p(C_+|\mathbf{x})$ that a site $\mathbf{x}$ is phosphorylatable (i.e. is a member of the positive class C_+) using the following model,

$$p(C_+|\mathbf{x}) = \hat{t}(\mathbf{x}) = \sigma\left(\sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{3.2.3}$$

where $\sigma(t) = (1 + e^{-t})^{-1}$, and the w_is and b are learned through the RVM training. Empirically it has been seen that more w_i terms vanish with the RVM than with the SVM, resulting in a sparser, more parsimonious model [37]. Similar to the idea of SVs, the vectors $\mathbf{x}_i$ whose associated $w_i > 0$ are called relevance vectors (RVs). While there has been an addition to the SVM to allow for probabilistic outputs [30], it has been shown that SVMs make poor estimates of the posterior probability due to the nature of its objective function [37]. To use the RVM, the user needs to only specify values for the kernel parameters.

3.3 Classification Relevance Units Machine

We propose a new classification model CRUM, which combines a sparse kernel regression model called the Relevance Units Machine [15] with the logistic regression model [16]. To obtain probabilistic predictions, the CRUM seeks to model the posterior distribution $p(C_+|\mathbf{x})$ that a site $\mathbf{x}$ is phosphorylatable (i.e. is a member of the positive class C_+) using the following model,

$$p(C_+|\mathbf{x}) = \hat{t}(\mathbf{x}) = \sigma\left(\sum_{i=1}^{M} w_i k(\mathbf{x}, \mathbf{u}_i) + b\right) \tag{3.3.1}$$

where M is a positive integer, and the w_is, b, kernel parameters, and the relevance units (RUs) $\mathbf{u}_i$s are learned through the CRUM training. A commonly used kernel function, and the one used in this study, is the Gaussian kernel described in Section 3.2.1. The CRUM can be visualized as a feed-forward network, as shown in Figure 3.1.

Like the RVM, the CRUM provides estimates for the posterior distribution. However, the CRUM training algorithm has lower run-time and memory complexity than the RVM algorithm, requiring the storage of an $N \times M$ rather than an $N \times N$ design matrix [25]. To use the CRUM, the user needs to specify only the model complexity parameter M, typically with $M \ll N$ to achieve a parsimonious model. A simple scheme for selecting M is proposed later in this section to eliminate the need for costly cross-validation for model selection.

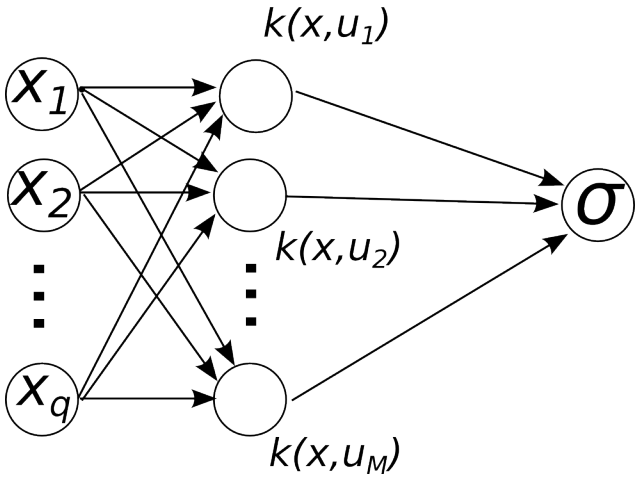


Figure 3.1. The CRUM model displayed as a feed-forward network. The input layer consists of d nodes representing the d-dimensional input vector $\mathbf{x}$. The hidden layer consists of bias value b and the M kernel functions centered on the RUs $\mathbf{u}_i$. Finally, a weighted sum of the hidden layer outputs is computed and transformed into a posterior probability via sigmoid function σ.

3.3.1 Overview of Learning Algorithm

Our original learning algorithm described in [25] implements a regularized supervised learning method using approximate Bayesian techniques to learn the full set of model parameters: the coefficients w_i, the bias b, the kernel parameters, and the RUs $\mathbf{u}_i$. However, this full supervised training algorithm is not practical for training on the large phosphorylation datasets due to the high complexity in learning the RUs and kernel parameters (see Appendix 3A). Instead, we propose to learn the model parameters in two phases.

Phase 1 Clustering: In the first phase, unsupervised learning methods are used to learn the RUs and kernel parameters. For computational simplicity, the k-means clustering algorithm [24] is used to find the M clusters. Since we are using the Gaussian kernel under the Euclidean norm, we use Euclidean distance as the similarity metric for the k-means clustering. Then, the RUs are set to the M cluster centers, and γ is set to $(2r^2)^1$ where r is the maximum distance between cluster centers.

This clustering phase also provides an avenue to select empirically an appropriate value of M. There has been work in the clustering research community for developing methods to determine the number of clusters, such as [35]. There are also clustering al-

gorithms using Bayesian techniques to find the appropriate number of clusters, such as the Variational Bayesian Expectation-Maximization algorithm for Mixture of Gaussians [9]. However, these methods are computationally complex, and we opted to use a simpler method to determine M, as described later. Due to the regularization scheme presented in the next phase, an overestimate of M will not result in overfitting.

Phase 2 Supervised Learning: Like the original CRUM learning algorithm [25], the w_is and b are still learned by using the iteratively reweighted least squares (IRLS) algorithm and the hyperparameter α is learned by using type-II maximum likelihood. The hyperparameter α serves as a regularizer, and prevents overfitting caused by overestimating the model complexity M. What follows is a detailed description of Phase 2 of the CRUM learning algorithm.

3.3.2 Learning Algorithm Assumptions and Derivation

Without loss of generality, we assume $b = 0$ and set $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$ for a fixed positive integer M. Define

$$\hat{t}_i = \sigma\left(\sum_{m=1}^{M} w_m k(\mathbf{x}_i, \mathbf{u}_m)\right) \tag{3.3.2}$$

and the kernel matrix $\mathbf{K}$ with elements

$$K_{ij} = k(\mathbf{x}_i, \mathbf{u}_j) \tag{3.3.3}$$

for $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}$. Here, the kernel function k is the Gaussian kernel with parameter γ under the Euclidean norm in Equation (3.2.1). The RUs $\mathbf{u}_j$s and γ are fixed to the values learned in Phase 1.

Let $\hat{t}_i = \hat{t}(\mathbf{x}_i)$. We adopt a Bernoulli distribution for $P(t|\mathbf{x}, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})$, where $\mathbf{U}$ is the matrix containing all RUs column-wise and $\mathbf{\Theta}$ is the set of all kernel parameters, in this case $\mathbf{\Theta} = \{\gamma\}$. This leads to the following likelihood given the model:

$$P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) = \prod_{n=1}^{N} \hat{t}_n^{t_n}[1 - \hat{t}_n]^{1-t_n}. \tag{3.3.4}$$

To mitigate overfitting by reducing the classifier's complexity, we adopt a zero-mean isotropic Gaussian prior distribution over $\mathbf{w}$ (see Appendix 3B for an empirical example of overfitting mitigation),

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{3.3.5}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix and $\alpha \in \mathbb{R}^+$ is the hyperparameter controlling the precision of the Gaussian over the weights. Finally, we assume a Gamma distributed hyperprior for the hyperparameter α,

$$p(\alpha) = \mathrm{Gamma}(\alpha|a, b) \tag{3.3.6}$$

where $\mathrm{Gamma}(x|a, b) = \Gamma(a)^{-1}b^a x^{a-1}e^{-bx}$ and $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$.

The a and b parameters are fixed to small values to make the hyperpriors non-informative [37].

We obtain the posterior distribution by taking the product of all our assumptions (3.3.4)-(3.3.6),

$$p(\mathbf{t}, \mathbf{w}, \alpha|X, \mathbf{U}, \boldsymbol{\Theta}) = P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \boldsymbol{\Theta})p(\mathbf{w}|\alpha)p(\alpha). \tag{3.3.7}$$

The dependencies between all the variables can be visualized using a directed graph called a Bayesian network, as shown in Figure 3.2.

Taking the negative log of Equation (3.3.7) and dropping all terms not involving w give the following objective function,

$$\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N}[t_n \ln \hat{t}_n + (1 - t_n)\ln(1 - \hat{t}_n)] + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}. \tag{3.3.8}$$

The negative summation term in Equation (3.3.8) is known as the empirical risk under the log loss, and is an empirical measure of the error of the classifier. The last term of the equation is the penalty function, penalizing complex models over simpler ones. It is known that empirical risk is a biased estimate of the true risk and thus the additional penalty term is required to avoid overfitting [32]. Therefore, the parameter α controls the trade-off between empirical error and model complexity. Thus, minimizing the objective

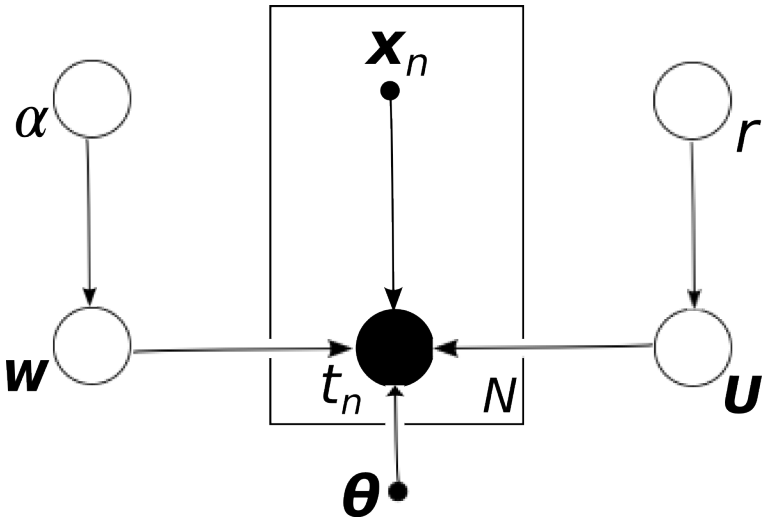


Figure 3.2. The CRUM model displayed as a Bayesian network. Note that the nodes contained in the center rectangle are repeated N times, one for each training data point. When using the Gaussian kernel, $\mathbf{\Theta} = \{\gamma\}$.

function (3.3.8) corresponds to a structural risk minimization [22] and aims to minimize the true error bound. Furthermore, log loss is a margin maximizing loss function [31] and this fact helps to tighten the true error bound to further reduce overfitting [32].

To estimate $\mathbf{w}$ over a fixed α, we minimize Equation (3.3.8), or equivalently maximize Equation (3.3.7), with respect to $\mathbf{w}$. We can compute the $\mathbf{w}$ giving the minimum of Equation (3.3.8), call it $\mathbf{w}^*$, using the Newton-Raphson method, which is also known in this classification context as the IRLS algorithm. The use of IRLS requires the computation of the gradient and Hessian as follows,

$$\nabla\mathcal{L} = \mathbf{K}^T(\hat{\mathbf{t}} - \mathbf{t}) + \alpha\mathbf{w} \tag{3.3.9}$$

$$\mathbf{H} = \mathbf{K}^T\mathbf{B}\mathbf{K} + \alpha\mathbf{I} \tag{3.3.10}$$

where $\mathbf{B} = \mathrm{diag}(\hat{t}_1(1-\hat{t}_1), \hat{t}_2(1-\hat{t}_2), \ldots, \hat{t}_N(1-\hat{t}_N))$ and $\hat{\mathbf{t}} = \left[\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N\right]^T$. The update formula for $\mathbf{w}$ is then

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \lambda\mathbf{H}^{-1}\nabla\mathcal{L} \tag{3.3.11}$$

for an appropriately chosen step size λ. Pseudocode for learning $\mathbf{w}$ is presented in Algorithm 1.

Note that the inverse Hessian computed in line 8 can be done efficiently using the Cholesky decomposition [27]. The inverse Hessian is also used in the learning of α; therefore, in addition to returning the optimal $\mathbf{w}$, an implementation should also return $\mathbf{H}^{-1}$.

To learn α, we use an approximate Bayesian estimate called empirical Bayes (also known as evidence approximation or type-II maximum likelihood). We marginalize $\mathbf{w}$ in the posterior (3.3.7) and use the Laplace approximation [1],

$$\begin{aligned} &p(\mathbf{t}, \alpha | X, \mathbf{U}, \mathbf{\Theta}) = \\ &\int P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) p(\mathbf{w}|\alpha) p(\alpha) d\mathbf{w} \\ &\approx (2\pi)^{M/2} |\mathbf{\Sigma}|^{1/2} P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}^*, \mathbf{\Theta}) p(\mathbf{w}^*|\alpha) p(\alpha) \end{aligned} \tag{3.3.12}$$

where the covariance of the Laplace approximation is

$$\mathbf{\Sigma} = (\mathbf{K}^T \mathbf{B} \mathbf{K} + \alpha \mathbf{I})^{-1} \tag{3.3.13}$$

and its mean $\mathbf{w}^*$ is the $\mathbf{w}$ that maximizes $P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) p(\mathbf{w}|\alpha)$. Note that $\mathbf{w}^*$ and $\mathbf{\Sigma}$ are the optimal $\mathbf{w}$ and the inverse Hessian H^{-1}, respectively, as computed in Algorithm 1.

Taking the derivative of Equation (3.3.12) and setting it to zero give the following iterative estimate for α,

$$\alpha^{new} = \frac{M - \alpha^{old} \mathrm{tr}[\mathbf{\Sigma}]}{\mathbf{w}^{*T} \mathbf{w}^*}. \tag{3.3.14}$$

In summary, the pseudocode for Phase 2 of the CRUM learning algorithm is presented in Algorithm 2.

Note that if we wish, b may be learned by augmenting the weight vector with b, such that $\tilde{\mathbf{w}} = [w_1, \ldots w_M, b]^T$. The kernel matrix must also be augmented to include an additional column of ones. Denoting the column vector of ones with $\mathbf{1}$, we define the augmented kernel matrix as $\tilde{\mathbf{K}} = [\mathbf{K}, \mathbf{1}]$. Then, we use the aforementioned algorithms using $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{K}}$ instead of $\mathbf{w}$ and $\mathbf{K}$, respectively.

Input: Matrix of training data $\mathbf{X}$ and its associated vector of class labels $\mathbf{t}$, kernel function k with parameters Θ, number of kernel functions M, RUs $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M$, and trade-off parameter α

$\mathbf{w} \leftarrow$ initial value
$\lambda_{min} \leftarrow$ a small value (e.g. 2^{-8})
while $\mathbf{w}$ has not converged **do**
  $\mathbf{w}^{old} \leftarrow \mathbf{w}$
  $\mathcal{L}^{old} \leftarrow$ Equation (3.3.8)
  $\mathcal{L} \leftarrow \infty$
  $\nabla\mathcal{L} \leftarrow$ Equation (3.3.9)
  $\mathbf{H} \leftarrow$ Equation (3.3.10)
  $\Delta \leftarrow \mathbf{H}^{-1}\nabla\mathcal{L}$
  $\lambda \leftarrow 1$
  {Comment: Line search to find an appropriate λ value}
  while $\lambda > \lambda_{min}$ and $\mathcal{L} > \mathcal{L}^{old}$ **do**
    $\mathbf{w} \leftarrow \mathbf{w}^{old} - \lambda\Delta$
    $\mathcal{L} \leftarrow$ Equation (3.3.8) using updated $\mathbf{w}$
    $\lambda \leftarrow \lambda/2$
  end while
  {Comment: If line search failed, revert to old $\mathbf{w}$ value}
  if $\lambda < \lambda_{min}$ **then**
    $\mathbf{w} \leftarrow \mathbf{w}^{old}$
  end if
end while
return $\mathbf{w}, \mathbf{H}^{-1}$

Algorithm 2: Learning $\mathbf{w}$

Input: Matrix of training data $\mathbf{X}$ and its associated vector of class labels $\mathbf{t}$, kernel function k with parameters Θ, number of kernel functions M, RUs $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M$

1: $\mathbf{w}, \alpha \leftarrow$ initial values
2: Compute design matrix $\mathbf{K}$ using Equation (3.3.3)
3: **while** $\mathbf{w}$ and α have not converged **do**
4: $\quad \mathbf{w} \leftarrow$ Algorithm 1 using the current value of $\mathbf{w}$ as initial value
5: $\quad \alpha \leftarrow$ Equation (3.3.14)
6: **end while**
7: **return** $\mathbf{w}$

Algorithm 3: CRUM Supervised Learning Algorithm (Phase 2)

3.3.3 Selecting Model Complexity

Both phases of the CRUM learning algorithm require the specification of a fixed parameter M that represents the number of RUs and thus is a model complexity parameter. Cross-validation could be used by repeatedly using the entire learning algorithm on a variety of selected M values and choosing the model with the best performance. Since this is a slow and time consuming process, we propose a more efficient method that exploits only Phase 1 of the learning algorithm and avoids the costly Phase 2.

During Phase 1, we cluster the unlabeled data X into M clusters using k-means clustering. Giving the clustering result a probabilistic interpretation, the AIC and BIC can be computed [28]. Assume all clusters are spherical Gaussian distributions with identical variance, then the maximum likelihood estimate for the variance is

$$\hat{\sigma}^2 = \frac{1}{N-M} \sum_{i=1}^{N} ||\mathbf{x}_i - \boldsymbol{\mu}_{(i)}||^2 \tag{3.3.15}$$

where $\boldsymbol{\mu}_{(i)}$ is the closest cluster center to $\mathbf{x}_i$ using Euclidean distance. The log-likelihood of X given the learned clustering distribution $\mathcal{D}_M$ is then

$$l(X|\mathcal{D}_M) = \sum_{i=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}\hat{\sigma}^d} - \frac{1}{2\hat{\sigma}^2} ||\mathbf{x}_i - \boldsymbol{\mu}_{(i)}||^2 + \log \frac{N_{(i)}}{N} \right) \tag{3.3.16}$$

where $N_{(i)}$ is the number of datapoints belonging to the same cluster as $\mathbf{x}_i$. There are $M + Md$ free parameters in this model: $M - 1$ mixture probabilities, the Md cluster center coordinates, and the variance.

With the log-likehood of the data and the number of free parameters, we can compute the AIC and BIC of the M cluster model using

$$\mathrm{AIC}(M) = 2(Md + M - l(X|\mathcal{D}_M)) \tag{3.3.17}$$

$$\mathrm{BIC}(M) = (Md + M)\log N - 2l(X|\mathcal{D}_M) \tag{3.3.18}$$

where a low AIC or BIC implies a good model. This allows CRUM's M to be selected as the model with the best observed AIC or BIC value. This has clear computational advantages: i) the costly supervised learning algorithm (Algorithm 2) is not invoked and ii) only a single clustering needs to be conducted per M value evaluated, compared to a n-fold cross-validation that would require clustering and supervised training on n different datasets.

3.4 Experimental Setup

3.4.1 Dataset Preparation

Dataset	Type	#Pos	#Neg	Comments
Evaluation	S	9129	10137	Used in the comparison of learning methods
	T	2206	2799	
	Y	1555	1691	
Benchmark Test	S	1255	15459	Test dataset for benchmark experiment
	T	353	9995	
	Y	396	4795	
Benchmark Training	S	8000	8000	Used in 1) cross-validation experiment and 2) benchmark experiment as training dataset
	T	1949	1949	
	Y	1344	1344	

Table 3.1. Composition of datatsets. (#Pos the number of phosphorylation sites (positives), #Neg the number of putative non-phosphorylation sites (negatives))

We prepared three datasets, both of which are derived from the Phospho.ELM database [11, 12] of known, experimentally verified eukaryotic phosphorylatable S, T, and Y sites. The composition of each dataset is summarized in Table 3.3.1.

- *Evaluation dataset*: This dataset contains all known sites from 2009 release of Phospho.ELM.
- *Benchmark test dataset*: The independent dataset taken from [2] was used as benchmark test dataset in this study. This dataset is originally from [33], where it was used to assess the performance of existing predictors. The annotations of this dataset are updated based on Release 9.0 of the Phospho.ELM database [11, 12].
- *Benchmark training dataset*: This dataset contains all known phosphorylation sites from the Phospho.ELM (Release 8.1) database minus the overlapping entries with the benchmark test dataset, providing a training dataset independent of the benchmark test dataset. This dataset was also taken from [2].

Positive Datasets: Phosphorylation sites of a specific window size are extracted, resulting in a positive dataset for each type of sites (S, T, and Y). Since independent observations are assumed by the CRUM learning algorithms, each positive training dataset is reduced based on homology. As done in PredPhospho [20], each dataset is individually searched for sites with more than 70% identity using Needleman-Wunsch alignment [26], and only one of the similar sites is kept in the dataset. Homology reduction is not performed on the benchmark test dataset so that all sites are used in the evaluation, including sites at the edges of the protein sequence that do not have sufficient window size.

Negative Datasets: All S, T, and Y sites not annotated as phosphorylatable in the datasets' proteins are assumed to be nonphosphorylatable. This assumption may prove false in some instances, as some of the sites may indeed be undiscovered phosphorylation sites. However, due to the rareness of phosphorylation sites, only a few false negative sites are expected and will only affect our training and evaluation to a small degree [2]. Using the same procedure as the positive training datasets preparation, the negative training datasets are homology reduced. In addition, since the sizes of the negative datasets far exceed those of the positive datasets, only a random sample of the negative sites are chosen in equal size to each positive dataset to improve training [2, 20].

3.4.2 Performance Assessment Measures

The following predictive performance measures on the test datasets are used to assess all predictors: sensitivity (Sn), specificity (Sp), Matthews correlation coefficient (MCC), and accuracy (Ac). By construction, the test datasets are independent of the training datasets and so the test datasets have no bearing on the training or on the determination of the parameters. Thus, the test datasets gives unbiased performance measurements on the generalization ability of the considered predictors. The performance measures are computed as follows

$$Ac = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{3.4.1}$$

$$Sn = \frac{TP}{TP + FN} \times 100\% \tag{3.4.2}$$

$$Sp = \frac{TN}{TN + FP} \times 100\% \tag{3.4.3}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \tag{3.4.4}$$

where the variables are the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). For Ac, Sn, and Sp, the ideal is 100%. As a correlation coefficient, MCC ranges from -1 to $+1$, and the ideal predictor would have a MCC of 1.

3.4.3 Implementation

To conduct the computational experiments, we use the LIBSVM [8] implementation of the SVM that is freely available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. For the RVM, Tipping's MATLAB implementation of the fast RVM learning algorithm described in [38] is used, and is freely available at `www.vectoranomaly.com`. Since the RVM implementation is in MATLAB and a MATLAB interface is available for LIBSVM, we implemented CRUM, and conducted all dataset processing and experiments in the MATLAB environment.

3.5 Results and Discussion

3.5.1 Comparison of Learning Methods

This experiment focuses on the comparison of learning methods where the performance of the CRUM learning method is compared to other popular kernel machines the SVM and the RVM [37] for prediction of phosphorylation sites. In addition, three different model selection schemes for the CRUM are compared.

Parameter Search

All models considered contain parameters that are not learned and thus must be determined by external means. These parameters are C and γ for the SVM, γ for the RVM, and M for the CRUM. A search over the parameter space is conducted and each model is evaluated using cross-validation. The parameters that result in the lowest prediction error according to cross-validation are used to train the final predictor for the comparison. For the T and Y datasets, 10-fold cross-validation is used. Since the S dataset is significantly larger, 3-fold cross-validation is used.

The parameter searches are conducted as follows. For the SVMs, we conduct a course grid search over $C = 2^i$ and $\gamma = 2^j$ where $i, j \in \{-10, -9, \ldots, 9, 10\}$. Based on the cross-validation results, a region is heuristically determined for a more precise parameter search. The parameter searches for RVM and CRUM are conducted in a similar fashion over a single dimension, γ or M, respectively. Also, the use of clustering in CRUM as described in Section 3.3.3 allows M to be selected as the model with the best AIC or BIC value.

Protein Site Features

Crystallization studies show that a region of the protein substrate of about 9 to 12 amino acids in length surrounding the acceptor residue contacts the PK active site [34]. Therefore, the sequence of amino acids surrounding a site in its 3D structure should provide sufficient information to determine whether the site is phosphorylatable. However, since

the 3D structure is not usually known, most predictors settle for only using the amino acid sequence within a window surrounding the site in the linear sequence.

Since CRUM relies mainly on sequence like NetPhos, we adopt the same window sizes that they experimentally verified to have good predictive ability [3]. Specifically, a window size of 9 is used for T and Y sites, while a window size of 11 is used for S sites. Site sequences are represented using the BLOSUM encoding that was shown to achieve better performance than the orthogonal encoding [19]. Each amino acid in the window surrounding the site is represented by a real-valued 20- dimensional vector of substitution scores for each amino acid. A min-max normalized BLOSUM62 matrix is used, where each row of the matrix is the substitution scores of the amino acid against all other amino acids. Using a window size of 11 for S sites, we need to represent each of the 11 amino acids downstream and upstream of the site, resulting in a 440-dimensional vector representing the site. Similarly, T and Y sites are represented by 360-dimensional vectors. Note that center amino acid is always an S, T, or Y, and is thus excluded from the representation.

Predictor Assessment

In addition to the performance assessment measures described in Section 3.4.2, the predictors are also assessed based on their computational complexity. The number of critical vectors (CVs), which are the SVs, RVs, and RUs for the SVM, RVM, and CRUM, respectively, are reported, as this measure gives an indication of the run-time and memory complexity of the resulting predictors, as given in Equations (3.2.2), (3.2.3), and (3.3.1). We also conduct run-time benchmarking trials for the training and prediction algorithms, and report the average of 10 trials. In this case, note that there is a bias toward the SVM implemented in faster compiled C++, whereas both CRUM and RVM are implemented in the slower interpreted language of MATLAB. Due to the presence of multiple local minima in the clustering phase, the CRUM predictors includes 10 replicates of k-means to mitigate this problem and is included in the training run-times reported.

The computer used on the smaller T and Y datasets is a 2.53 GHz Intel Core 2 Duo with 4 GB of memory. Due to the memory requirements for the RVM, we used a PC with 2.83 GHz Intel Core 2 Quad with 8 GB of memory to analyze the large S dataset. The run-time benchmarking trials are run on these computers.

Results

Predictor	Ac (%)	Mcc	Sn (%)	Sp (%)	#CV	Train (s)	Test (s)^2
SVM	75.32	0.51	68.42	81.54	10,536	491.42	32.46
RVM	74.01	0.48	69.52	78.08	76	3312.55	2.03
CRUM-CV	73.97	0.48	69.63	77.89	134	2964.50	2.27
CRUM-BIC	70.34	0.40	63.82	76.21	64	2681.57	1.10
CRUM-AIC	74.13	0.48	68.42	79.27	157	3135.63	2.74

Table 3.2. Comparison of predictors on test dataset for serine site prediction

Predictor	Ac (%)	Mcc	Sn (%)	Sp (%)	#CV	Train (s)	Test (s)^2
SVM	76.55	0.53	60.45	89.25	3031	25.80	2.16
RVM	73.75	0.46	60.90	83.87	50	499.68	0.19
CRUM-CV	74.35	0.48	59.09	86.38	82	129.66	0.25
CRUM-BIC	69.74	0.38	53.64	82.44	20	73.80	0.07
CRUM-AIC	73.74	0.47	57.73	86.38	88	139.56	0.25

Table 3.3. Comparison of predictors on test dataset for threonine site prediction

Predictor	Ac (%)	Mcc	Sn (%)	Sp (%)	#CV	Train (s)	Test (s)^2
SVM	68.52	0.37	64.52	72.19	2339	11.10	1.24
RVM	69.75	0.39	65.16	73.96	16	75.95	0.04
CRUM-CV	70.06	0.40	65.81	73.96	37	53.07	0.07
CRUM-BIC	70.37	0.41	66.45	73.96	16	36.91	0.03
CRUM-AIC	70.37	0.41	66.45	73.96	78	78.31	0.17

Table 3.4. Comparison of predictors on test dataset for tyrosine site prediction

For this experiment, we used the evaluation dataset described in Section 3.4.1. For an unbiased comparison between different models, we need a test dataset. We therefore reserve 10% of each of the positive and negative datasets as the test dataset. The remaining 90% forms the training and validation datasets.

Table 3.2, Table 3.3, and Table 3.4 show the details of the comparative analysis of the predictors on the test dataset for S, T, and Y site predictions, respectively. The CRUM predictor names indicate the method used to determine the model complexity parameter M, which is either "CV" for cross-validation, AIC, or BIC.

In Table 3.2, it is shown that the SVM has the best predictive performance for the S dataset, but the CRUM and RVM are close. The SVM also has the shortest training times, but we note again that the reported times are biased toward the SVM due to its faster compiled implementation compared to the slower interpreted language of MATLAB. However the CRUM and RVM models are significantly more parsimonious with the SVM having at least 67 times more CVs. The effect this complexity has can be seen on the testing times where the RVM and CRUM predictors are at least 9 times faster even with the implementation disadvantage.

With Table 3.3 for the T dataset, it is shown again that the SVM achieves the highest predictive performance. However, the RVM and CRUM predictors perform closely. The CRUM and RVM models are again more parsimonious than the SVM, using between 0.66% and 2.90% of the number of CVs. This leads to CRUM and RVM predictors that are 9 to 31 times faster than the SVM. Furthermore the CRUM training time is about 4 to 7 times faster than the RVM training.

Table 3.4 shows that the CRUM predictors have better predictive performance over the SVM and RVM, though all three methods are close. We again see the CRUM and RVM models achieving more parsimonious models than the SVM model, using between 0.68% to 3.33% of the number of CVs; leading to prediction runtime on the testing data of about 8 to 41 times faster than on the SVM. In this case, the RVM has a fast training time, beating the CRUM-AIC predictor by a few seconds. However, the CRUM BIC predictor's, with comparable predictive performance and complexity, training is about 2 times faster than RVM's and only about 3 times slower than SVM's.

Overall, we see that the SVM offers the best prediction accuracy of the methods considered for S and T prediction and the quickest training times over all datasets. However, this comes at the significant cost that the user does not know the uncertainty of the SVM's prediction, or has an inaccurate measure of uncertainty if Platt's method is used to obtain posterior estimates from SVM outputs [30, 37]. Both the CRUM and RVM provide accurate posterior estimates by virtue of their data likelihood-based objective functions [25,37], while also significantly reducing the run-time and memory complexity of the prediction algorithm, as indicated by the use of less than 5% of the CVs than the SVM.

Arguably, prediction run-time that is more important than training run-time, as training is typically invoked at far less frequency than the prediction algorithm.

In comparison to the RVM, the CRUM achieves comparable predictive performance with a lower cost training algorithm, in both run-time and memory complexity. Furthermore, through the use of AIC and BIC, model selection is simpler and faster with the CRUM. It is known that with a finite sample, BIC underestimates the model complexity, while the AIC overestimates it [16], as confirmed here by the cross-validation model complexity falling between BIC's and AIC's. However the results also show that this is not an issue with the CRUM, as the built-in regularization mitigates the overfitting problem and the resulting complex AIC model exhibits comparable results to the cross-validation model. Therefore the use of AIC is a cheap, viable option for CRUM model selection.

3.5.2 Evaluation of Protein Site Features

The goal of this experiment is to evaluate different phosphorylation site features and their combinations. In addition to the BLOSUM encoded sequence feature described in Section 3.5.1, we consider amino acid composition and predicted disorder regions, which provide 3D structure information. The performance of CRUM predictors based on combinations of these three features is compared via cross-validation to give insight into which feature combinations are important.

Protein Site Features

BLOSUM Encoded Sequence Feature: See Section 3.5.1.

Amino Acid Composition: A statistical analysis of the amino acids surrounding phosphorylation sites revealed characteristics that coincide with the amino acid composition of disordered regions. Both types of sequences are significantly depleted in rigid amino acids and significantly enriched with flexible amino acids [18]. Due to this statistically significant difference in composition, amino acid composition can be used to discriminate a phosphorylatable site from a non-phosphorylatable site. The percent amino acid composition of the site can be represented using a 20-dimensional vector, where each component represents the percent composition of one of the 20 amino acids. The percent composition

is computed over the same window size as the extracted site, including the center amino acid.

Disordered Regions: Phosphorylation sites and other post-translational modifications are observed to be located in disordered regions of proteins frequently, but not exclusively [18]. Intrinsically disordered protein regions are the parts of the protein that do not fold into a stable 3D structure. To maximize the applicability of CRUM, the true identification of disordered regions is not required and is instead predicted using a disorder predictor. The SVM and logistic regression-based VSL2B predictor is used that is based on amino acid composition [29]. VSL2B is applied to the entire protein sequence using an output window length of one, resulting in a probabilistic classification for each amino acid in the sequence being in a disorder region. The disorder classifications of the site and its surrounding amino acids are extracted over the same window sizes as the extracted site sequence. Therefore, 23-dimensional vectors represent S sites, and 19-dimensional vectors represent T and Y sites. Unlike the BLOSUM encoding, the center amino acid is included, as its disorder classification is dependent on the surrounding amino acids.

Results

Features	**Ac(%)**	**Sn(%)**	**Sp(%)**	**MCC**
CRUMp B	75.25	72.33	78.18	0.51
CRUMp D	64.00	78.36	49.64	0.29
CRUMp A	69.08	70.04	68.12	0.38
CRUMp B+D	75.49	73.16	77.82	0.51
CRUMp B+A	75.21	72.35	78.07	0.51
CRUMp D+A	68.37	72.73	64.00	0.37
CRUMp B+D + A	75.60	73.32	77.88	0.51

Table 3.5. Three-fold cross-validation performance on phosphoserine (S) prediction. (Ac - Accuracy, Sn - Sensitivity, Sp - Specificity, MCC - Matthews correlation coefficient; B - BLOSUM encoding feature, D - Disorder feature, A - Amino acid composition feature)

To evaluate the predictive performance of using the CRUM under a variety of different protein site features, we conducted a 3-fold cross-validation experiment using a benchmark training dataset described in Section 3.4.1. The protein features considered are the BLOSUM encoding of the site (B), disorder (D), amino acid composition (A),

Features	Ac(%)	Sn(%)	Sp(%)	MCC
CRUMp B	71.34	66.24	76.45	0.43
CRUMp D	64.91	71.88	57.93	0.3
CRUMp A	67.75	66.44	69.06	0.36
CRUMp B+D	71.73	67.21	76.24	0.44
CRUMp B+A	71.19	66.09	76.30	0.43
CRUMp D+A	68.06	69.06	67.06	0.36
CRUMp B+D+A	71.83	67.21	76.45	0.44

Table 3.6. Three-fold cross-validation performance on phosphothreonine (T) prediction. (Ac - Accuracy, Sn - Sensitivity, Sp - Specificity, MCC - Matthews correlation coefficient; B - BLOSUM encoding feature, D - Disorder feature, A - Amino acid composition feature)

Features	Ac(%)	Sn(%)	Sp(%)	MCC
CRUMp B	66.56	66.07	67.04	0.33
CRUMp D	62.31	66.89	57.74	0.25
CRUMp A	62.65	63.91	61.38	0.25
CRUMp B+D	65.74	68.15	63.32	0.32
CRUMp B+A	66.70	67.11	66.29	0.33
CRUMp D+A	62.69	67.26	58.11	0.25
CRUMp B+D+A	66.00	67.63	64.36	0.32

Table 3.7. Three-fold cross-validation performance on phosphotyrosine (Y) prediction. (Ac - Accuracy, Sn - Sensitivity, Sp - Specificity, MCC - Matthews correlation coefficient; B - BLOSUM encoding feature, D - Disorder feature, A - Amino acid composition feature)

and combinations thereof. Model selection for these CRUM predictors is conducted on the entire benchmark training dataset using the AIC scheme. Table 3.5, Table 3.6, and Table 3.7 give the resulting performance for S, T, and Y sites respectively, using a posterior threshold of 0.5. This means we classify a site as phosphorylatable if the predicted posterior probability is greater than 0.5, otherwise the site is classified as not phosphorylatable.

The results indicate that the CRUM predictors containing the BLOSUM feature show the best performance in terms of both accuracy and Matthews correlation coefficient (MCC) with well balanced sensitivity and specificity. Excluding the BLOSUM feature drops the specificity, and thus overall accuracy and MCC, by including more false positives. This remains true under a variety of different posterior threshold settings, as plotted in Figure 3.3 through Figure 3.5. In the ROC plots, the four CRUM predictors that use the BLOSUM feature (CRUM B, B+A, B+D, and B+D+A) perform roughly identically under

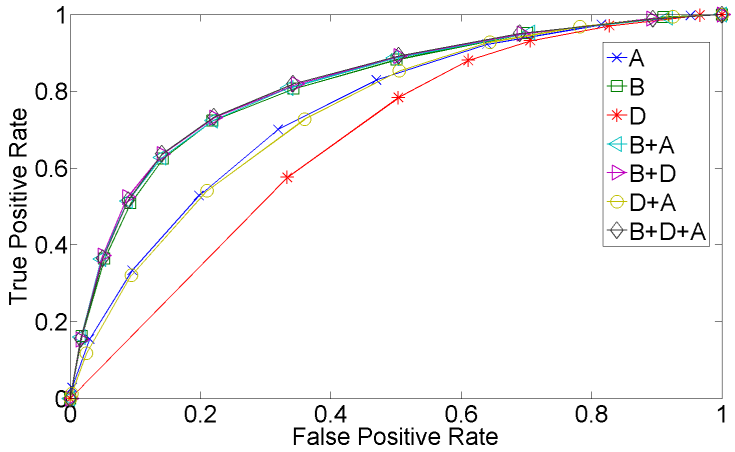


Figure 3.3. ROC for phosphoserine (S) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

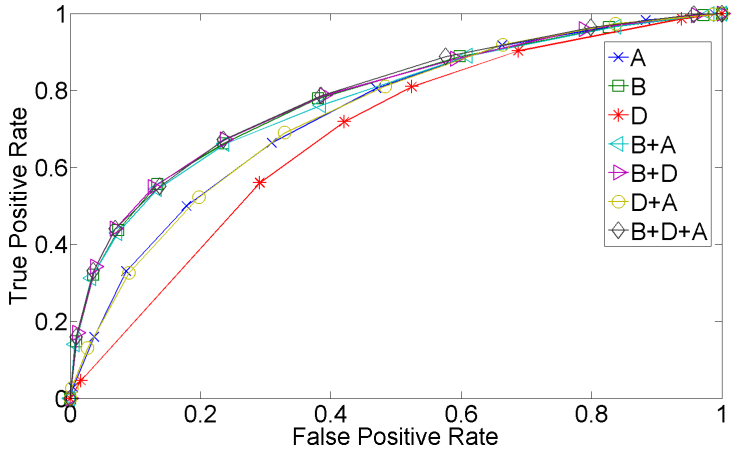


Figure 3.4. ROC for phosphothreonine (T) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

different threshold settings, and are always better or equal to the predictors that exclude the BLOSUM feature. This experiment suggests that using the BLOSUM feature is important to achieve good predictive performance.

It also should be noted that a user can target a desired false positive rate by appropriately selecting the posterior threshold for the CRUM using the ROC plots. No re-

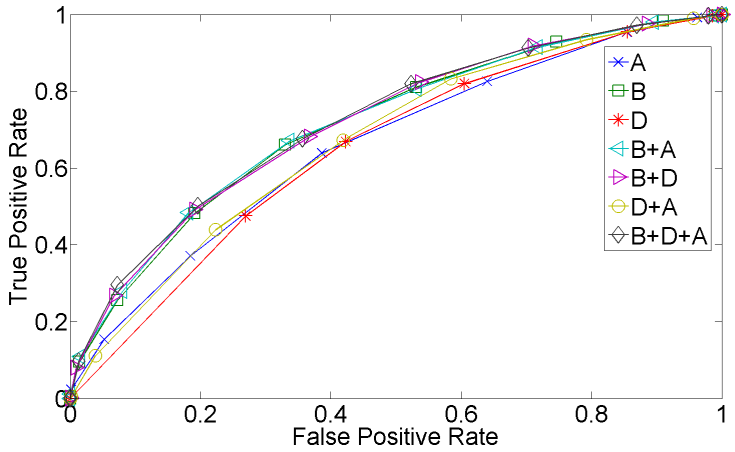


Figure 3.5. ROC for phosphotyrosine (Y) prediction by CRUM based on cross-validation performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

training is needed. This is in contrast with predictors like PPRED [2] that uses the standard hard-decision SVMs with no additional trained logistic regression model on the SVM outputs [30], where the specificity is fixed after training and thus the user has no way to change it without retraining the predictor under different settings of parameters, such as window size or positive to negative training data ratio.

3.5.3 Benchmark Comparison

Finally, the third experiment concentrates on the comparative study with existing predictors where the performance of different CRUM predictor configurations is compared to that of existing predictors on previously published benchmark datasets [2]. For this experiment, the CRUM predictors using different combinations of the features described in Section 3.5.2 are trained on the entire benchmark training dataset described in Section 3.4.1. The trained CRUM predictors' performances are then computed using the benchmark test dataset that is independent of the training set, as described in Section 3.4.1.

For comparison, the performance of NETPHOS [3] and DISPHOS [18] are also computed using the latest versions available on the web. Note that these tools have been updated since their original publication and may have been trained on proteins included in the

benchmark test dataset used in this experiment. Therefore, the performance of NETPHOS and DISPHOS may be biased in favor of them in this comparison.

Protein Features for Sites on Edges of Protein

In previous experiments and in the training of the CRUM predictors in this section, sites that lie on the edges of a protein sequence are omitted because of the insufficient number of amino acids to fill the required window size. However, since NETPHOS and DIPHOS report results for these edges sites, we modify CRUM to do the same when computing predictions.

Edge sites are accommodated by filling in the missing amino acids with the ambiguous amino acid code X. If the site is found near the start of a protein sequence, then the appropriate number of Xs are prepended to the site's sequence to fill the window size. If the site is found near the end of the protein sequence, then the appropriate number of Xs are appended to the site's sequence to fill the window size. These Xs are ignored in the computation of amino acid composition and are given a disorder value of 0.5 for its position in the disorder encoding described in Section 3.5.2.

Results

Figures 3.6, 3.7, and 3.8 show ROC plots for S, T, and Y sites, respectively. Similar to the cross-validation experiment, we see that all the BLOSUM-based CRUM predictors perform comparably with generally greater TPRs compared to the non-BLOSUM-based CRUM predictors (A, D, and D+A) at similar FPR values.

In comparison with NETPHOS and DISPHOS, there is no uniformly better tool in this benchmark test dataset. NETPHOS underperforms DISPHOS and the four BLOSUM-based CRUM predictors uniformly over all FPR values.

For S sites in Figure 3.6, DISPHOS has an advantage in the very small FPR range (less than 0.03) and at around 0.4 - 0.6 FPR, but otherwise the performance is comparable to the simpler BLOSUM-based CRUM predictors. For T sites in Figure 3.7, we again see DISPHOS and BLOSUM-based CRUM predictors being comparable, with the exception of an advantage for DISPHOS at the very low FPR range, again at less than 0.03. However,

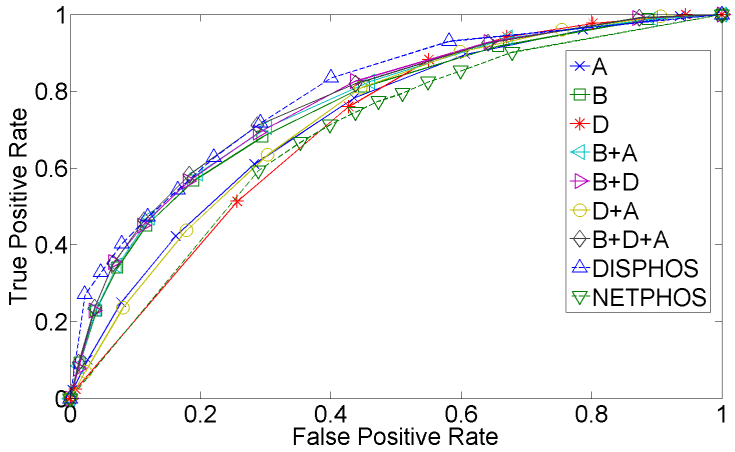


Figure 3.6. ROC for phosphoserine (S) prediction by CRUM based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

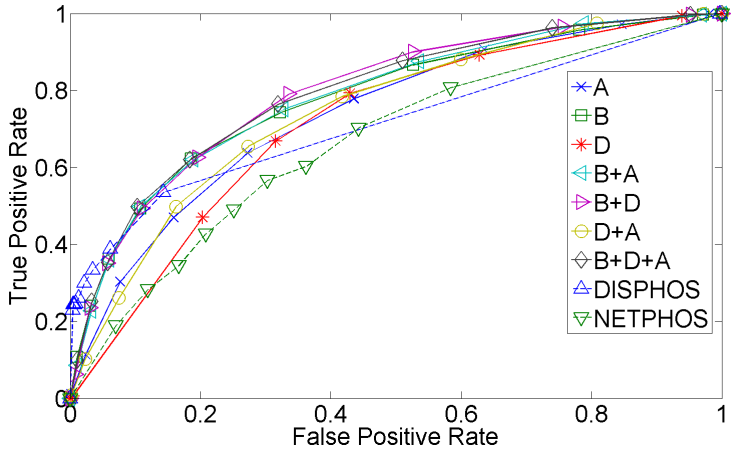


Figure 3.7. ROC for phosphothreonine (T) prediction by CRUM based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

different characteristics are seen for the Y sites in Figure 3.8. Here DISPHOS displays the best performance for FPRs under 0.6, with an increase in TPR of at most 0.3 compared to BLOSUM-based CRUM predictors.

A possible major factor for the underperformance of the CRUM predictors is the high dimensionality of the site representation relative to the sample size, particularly on the

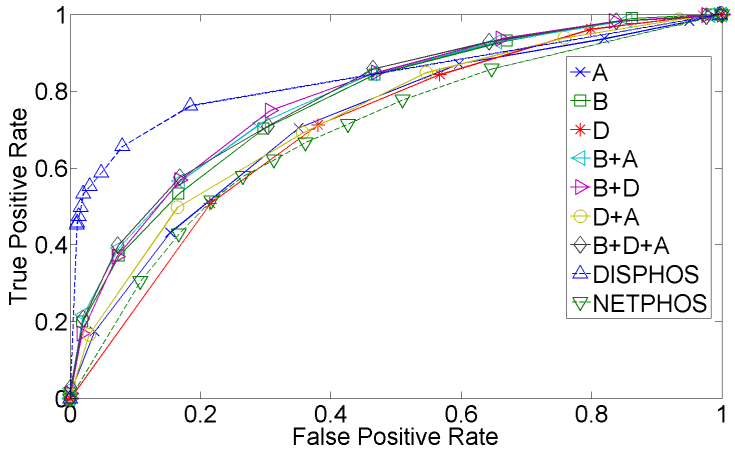


Figure 3.8. ROC for phosphotyrosine (Y) prediction by CRUM based on benchmark performance. The plot is generated using different posterior threshold values ranging from 0 to 1 in increments of 0.1.

Y dataset. The design of DISPHOS' encoding of sites into vectors is more computationally complex, such as the inclusion of the results of several disorder predictors, compared to the settings used here from CRUM. However, the design of DISPHOS used Fisher's permutation test and principal component analysis to reduce the dimensionality [18]. This dimensionality reduction to account for the small sample size may be a major factor in DISPHOS performance on the Y dataset, and will be considered in future work using CRUM.

3.6 Conclusions

In this study, probabilistic predictions of protein phosphorylation sites were proposed using our CRUM model as an alternative to the popular SVM and RVM. The experimental results show that the extremely compact models of CRUM provide a significant reduction to the computational complexity of the prediction algorithm, which is advantageous in conducting predictions at a large scale. We show that CRUM using only the BLOSUM feature, CRUM B, has among the best accuracy and MCC performance compared to other CRUM settings in both crossvalidation and benchmark experiments. The simplicity and efficiency of the BLOSUM feature computation make the data preparation process for using CRUM faster, as disorder predictors do not need to be invoked.

While the BLOSUM-based CRUM predictors are often comparable to the more complex DISPHOS predictor, which uses several disorder and other structural predictors to extract features, on the S and T benchmark datasets, there is a cost to this reduction of computational complexity, as DISPHOS outperforms the different CRUM predictor configurations considered in this study under the Y benchmark dataset.

In the future, as mentioned previously, dimensionality reduction of the protein feature encoding will be considered to improve the performance, particularly with Y sites. In addition, a multi-kernel implementation of CRUM shall be considered where only a single protein feature will be considered in each kernel. Currently in the combined feature CRUM predictors (e.g. B+D+A), all features are concatenated into the one vector and used in the same Gaussian kernel. When combined, it is possible that the lower dimensional features D and A may be overshadowed by the B feature simply due to B's high dimensionality rather than D or A being unimportant, as the above results may suggest. A multi-kernel solution may avoid this issue and the supervised learning algorithm may then be able to exploit all features for improved predictive performance.

Furthermore, we plan to extend the CRUM model and training algorithm for multi-class prediction. This will allow for kinasespecific phosphorylation site prediction, where a prediction for what type of PK phosphorylates the site is also made.

Acknowledgements

This work is supported in part by NIH Grants from the National Institute of General Medical Sciences (P20GM103516 and P20GM103466). The papers contents are solely the responsibility of the authors and do not necessarily represent the official views of the NIH.

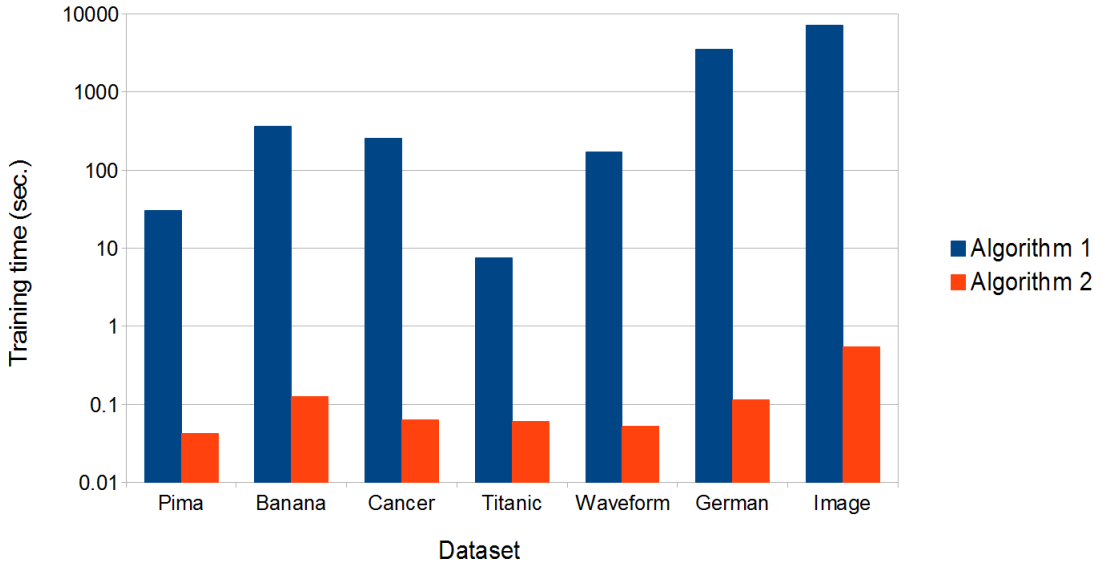


Figure 3.9. Training running-time comparison of CRUM algorithms. Algorithm 1 is the supervised learning algorithm described in Article 1 (Chapter 2) and Algorithm 2 is the hybrid learning algorithm described in Article 2 (Chapter 3).

Appendix 3A: Training Running-time Comparison of CRUM Algorithms

While the CRUM supervised learning algorithm presented in Article 1 (Chapter 2) has it merits in leading to more compact models than the SVM or RVM, the algorithm is not appropriate for problems larger than the benchmark datasets considered. To overcome this issue, the hybrid unsupervised/supervised learning algorithm of Article 2 (Chapter 3) was devised. Figure 3.9 shows the difference in training time between these two algorithms on the benchmark datasets considered in Article 1 (Chapter 2) using the same CRUM model size M. The hybrid learning algorithm displays a two to three orders of magnitude reduction in running time compared to the original supervised learning algorithm. Since the hybrid learning algorithm is more practical, the remaining dissertation focuses on using and building upon the hybrid learning algorithm.

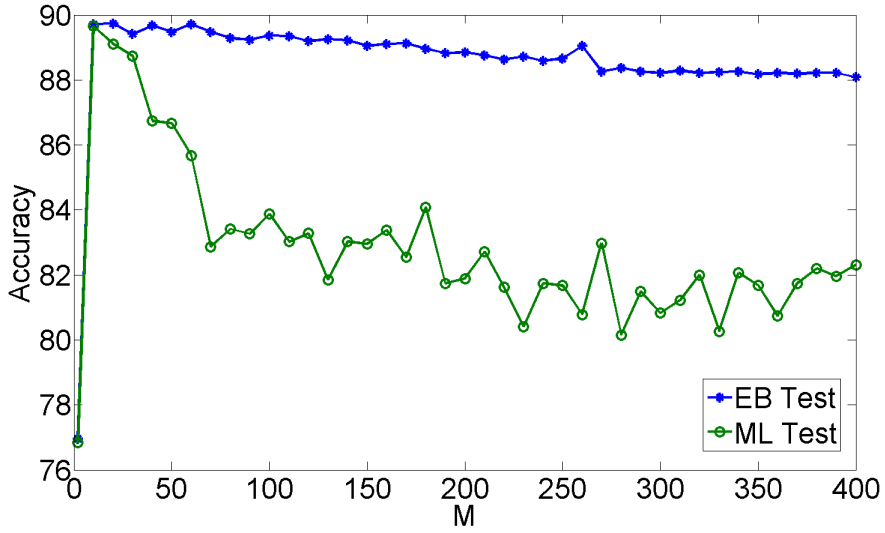


Figure 3.10. Accuracy comparison between empirical Bayes CRUM algorithm (EB Test) and maximum likelihood CRUM algorithm (ML Test) on the wine test dataset.

Appendix 3B: Accuracy Comparison between Empirical Bayes and Maximum Likelihood

To demonstrate the ability of the CRUM learning algorithm's use of empirical Bayes (EB) to mitigate overfitting, we consider a comparison with a CRUM learning algorithm that omits the use of a Gaussian prior on $\mathbf{w}$. Without the prior, this second CRUM learning algorithm would then learn $\mathbf{w}$ using a maximum likelihood (ML) estimator. Both algorithms are used to train a CRUM model on the wine dataset under increasing values of model size M. The resulting models are evaluated using a hold-out test dataset. It is observed in Figure 3.10 that with increasing M, the ML solution quickly overfits as accuracy sharply drops and performance becomes unstable. On the other hand, the EB solution mitigates overfitting and provides a stable solutions with high accuracy. The penalty for choosing a M that is larger than ideal is minimal in comparison to the ML solution. Therefore the use of simple heuristics to choose M is sufficient even though such methods may overestimate M.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] A. K. Biswas, N. Noman, and A. R. Sikder, "Machine learning approach to predict protein phosphorylation sites by incorporating evolutionary information," *BMC Bioinformatics*, vol. 11, no. 273, 2010.

[3] N. Blom, S. Gammeltoft, and S. Brunak, "Sequence and structure-based prediction of eukaryotic protein phosphorylation sites," *Journal of Molecular Biology*, vol. 294, pp. 1351–1362, 1999.

[4] N. Blom, T. Sicheritz-Ponten, R. Gupta *et al.*, "Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence," *Proteomics*, vol. 4, pp. 1633–1649, 2004.

[5] B. Boeckmann, A. Bairoch, R. Apweiler *et al.*, "The swiss-prot protein knowledgebase and its supplement trembl in 2002," *Nucleic Acids Research*, vol. 31, no. 1, pp. 365–370, 2003.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational learning theory*, 1992, pp. 144–152.

[7] R. I. Brinkworth, R. A. Breinl, and B. Kobe, "Structural basis and prediction of substrate specificity in protein serine/threonine kinases," *PNAS*, vol. 100, pp. 74–79, 2003.

[8] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[9] A. Corduneanu and C. M. Bishop, "Varitional bayesian model selection for mixture distributions," in *Artificial Intelligence and Statistics*, T. Jaakkola and T. Richardson, Eds. Morgan Kaufmann, 2001, pp. 27–34.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] F. Diella, C. M. Gould, C. Chica *et al.*, "Phospho.elm: a database of phosphorylation sites - update 2008," *Nucleic Acids Research*, vol. 36, pp. D240–D244, 2008.

[12] H. Dinkel, C. Chica, A. Via *et al.*, "Phospho.elm: a database of phosphorylation sites - update 2011," *Nucleic Acids Research*, vol. 39, pp. D261–D267, 2011.

[13] M. J. Eck and P. W. Manley, "The interplay of structural information and functional studies in kinase drug design: insights from bcr-abl," *Current Opinion in Cell Biology*, vol. 21, no. 2, pp. 288–295, 2009.

[14] J. Gao, J. J. Thelen, A. K. Dunker *et al.*, "Musite, a tool for global prediction of general and kinase-specific phosphorylation sites," *Molecular & Cellular Proteomics*, vol. 9, no. 12, pp. 2586–600, 2010.

[15] J. Gao and J. Zhang, "Sparse kernel learning and the relevance units machine," in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. Springer-Verlag, 2009, pp. 612–619.

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

[17] P. V. Hornbeck, I. Chabra, J. M. Kornhauser *et al.*, "Phosphosite: A bioinformatics resource dedicated to physiological protein phosphorylation," *Proteomics*, vol. 4, no. 6, pp. 1551–61, 2004.

[18] L. M. Iakoucheva, P. Radiojac, C. J. Brown *et al.*, "The importance of intrinsic disorder for protein phosphorylation," *Nucleic Acids Research*, vol. 32, pp. 1037–1049, 2004.

[19] C. R. Ingrell, M. L. Miller, O. N. Jensen *et al.*, "Netphosyeast: prediction of protein phosphorylation sites in yeast," *Bioinformatics*, vol. 23, pp. 895–897, 2007.

[20] J. H. Kim, J. Lee, B. Oh *et al.*, "Prediction of phosphorylation sites using svms," *Bioinformatics*, vol. 20, pp. 3179–3184, 2004.

[21] B. Kobe, T. Kampmann, J. K. Forwood *et al.*, "Substrate specificity of protein kinases and computational prediction of substrates," *Biochimica et Biphysica Acta*, vol. 1754, pp. 200–209, 2005.

[22] V. Koltchinskii, "Rademacher penalties and structural risk minimization," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1902–1914, 2001.

[23] A. Kreegipu, N. Blom, and S. Brunak, "Phosphobase, a database of phosphorylation sites: release 2.0," *Nucleic Acids Research*, vol. 27, no. 1, pp. 237–239, 1999.

[24] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, pp. 281–297.

[25] M. Menor and K. Baek, "Relevance units machine for classification," in *Proceedings of the 4th International Conference on BioMedical Engineering and Informatics*, 2011, pp. 2281–2285.

[26] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[27] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[28] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 2000, pp. 727–734.

[29] K. Peng, P. Radivojac, S. Vucetic *et al.*, "Length-dependent prediction of protein intrinsic disorder," *BMC Bioinformatics*, vol. 7, no. 208, 2006.

[30] J. C. Platt, "Probabilities for sv machines," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Scholkopf *et al.*, Eds. MIT Press, 2000, pp. 61–73.

[31] S. Rosset, J. Zhu, and T. Hastie, "Margin maximizing loss functions," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[32] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[33] A. R. Sikder and A. Y. Zomaya, "Analysis of protein phosphorylation site predictors with an independent dataset," *International Journal of Bioinformatics Research and Applications*, pp. 20–37, 2009.

[34] Z. Songyang, S. Blechner, N. Hoagland *et al.*, "Use of an oriented peptide library to determine the optimal substrates of protein kinases," *Current Biology*, vol. 4, no. 11, pp. 973–82, 1994.

[35] S. Still and W. Bialek, "How man clusters? an information-theoretic perspective," *Neural Computing*, vol. 16, no. 12, pp. 2483–506, 2004.

[36] Y.-R. Tang, Y.-Z. Chen, C. A. Canchaya *et al.*, "Gannphos: a new phosphorylation site predictor based on a genetic algorithm integrated neural network," *Protein Engineering, Design & Selection*, vol. 20, no. 8, pp. 405–412, 2007.

[37] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[38] M. E. Tipping and A. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003, pp. 3–6.

[39] B. Trost and A. Kusalik, "Computational prediction of eukaryotic phosphorylation sites," *Bioinformatics*, vol. 27, no. 21, pp. 2927–2935, 2011.

[40] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, vol. 24, 1963.

[41] Y.-H. Wong, T.-Y. Lee, H.-K. Liang *et al.*, "Kinasephos 2.0: a web server for identifying protein kinase-specific phosphorylation sites based on sequences and coupling patterns," *Nucleic Acids Research*, vol. 35, pp. W588–W594, 2007.

[42] Y. Xue, X. Gao, J. Cao *et al.*, "A summary of computational resources of protein phosphorylation," *Current Protein and Peptide Science*, vol. 11, pp. 485–496, 2008.

Chapter 4

Multiclass relevance units machine: benchmark evaluation and application to small ncRNA discovery

Abstract

Background. Classification is the problem of assigning each input object to one of a finite number of classes. This problem has been extensively studied in machine learning and statistics, and there are numerous applications to bioinformatics as well as many other fields. Building a multiclass classifier has been a challenge, where the direct approach of altering the binary classification algorithm to accommodate more than two classes can be computationally too expensive. Hence the indirect approach of using binary decomposition has been commonly used, in which retrieving the class posterior probabilities from the set of binary posterior probabilities given by the individual binary classifiers has been a major issue.

Reprinted, with permisssion, from M. Menor, K. Baek, and G. Poisson, " Multiclass relevance units machine: Benchmark evaluation and application to small ncRNA discovery," *BMC Genomics*, 14(Suppl 2):S6, 2013. DOI: 10.1186/1471-2164-14-S2-S6. PMCID: PMC3582431.

This work is based on an earlier work: M. Menor, K. Baek, and G. Poisson, " Multiclass relevance units machine: Benchmark evaluation and application to small ncRNA discovery," in *Proceedings of ISCB-Asia/SCCG 2012*, Shenzhen, China, December 17-19, 2012.

Methods. In this work, we present an extension of a recently introduced probabilistic kernel-based learning algorithm called the Classification Relevance Units Machine (CRUM) to the multiclass setting to increase its applicability. The extension is achieved under the error correcting output codes framework. The probabilistic outputs of the binary CRUMs are preserved using a proposed linear-time decoding algorithm, an alternative to the generalized Bradley-Terry (GBT) algorithm whose application to large-scale prediction settings is prohibited by its computational complexity. The resulting classifier is called Multiclass Relevance Units Machine (McRUM).

Results. The evaluation of McRUM on a variety of real small-scale benchmark datasets shows that our proposed Naïve decoding algorithm is computationally more efficient than the GBT algorithm while maintaining a similar level of predictive accuracy. Then a set of experiments on a larger scale dataset for small ncRNA classification have been conducted with Naïve McRUM and compared with the Gaussian and linear SVM. Although McRUMs predictive performance is slightly lower than the Gaussian SVM, the results show that the similar level of true positive rate can be achieved by sacrificing false positive rate slightly. Furthermore, McRUM is computationally more efficient than the SVM, which is an important factor for large-scale analysis.

Conclusions. We have proposed McRUM, a multiclass extension of binary CRUM. McRUM with Naïve decoding algorithm is computationally efficient in run-time and its predictive performance is comparable to the well-known SVM, showing its potential in solving large-scale multiclass problems in bioinformatics and other fields of study.

4.1 Background

The problem of classifying an object to one of a finite number of classes is a heavily studied problem in machine learning and statistics. There are numerous applications in bioinformatics, such as cancer classification using microarrays [11], prediction of protein localization sites [14], protein fold recognition [8], and identification of the kinase that acts upon a protein phosphorylation site [5].

Recently, a novel kernel-based learning algorithm called the CRUM for binary classification was introduced [21]. The CRUM addresses some of the concerns in the use

of the SVM [27], including removing the specification of the error/complexity trade-off parameter by using empirical Bayes methods, generation of more parsimonious models, and providing probabilistic outputs through the estimation of the posterior probability density. Furthermore, the training algorithm is more efficient than that of the RVM [26] that similarly addressed the SVM concerns. The highly compact model the CRUM generates significantly reduces the run-time of the prediction system and hence provides further advantages over the SVM in conducting large-scale data analyses [22], such as Next Generation Sequencing (NGS) data analysis.

In this paper, we extend the CRUM algorithm into the more general multiclass setting, allowing for applications beyond binary classification. This is achieved by decomposing the multiclass problem into a set of binary classification problems using the error correcting output codes (ECOC) [20] framework. To preserve the probabilistic outputs of the binary CRUM into the multiclass setting, the algorithm based on the generalized Bradley-Terry (GBT) model [16] is considered. Since the optimization problem solved by the GBT algorithm can prohibit its use in large-scale classification settings, we also propose a simple linear-time algorithm as an alternative. The details of Multiclass Relevance Units Machine (McRUM) construction based on the binary CRUM are described in the next section.

In this study, McRUM is evaluated on two sets of experiments. First, McRUM is applied to a variety of small-scale datasets from the UCI repository [3] in order to compare the performance of McRUM under different settings by using different decompositions of the multiclass problem into a set of binary classification problems and the use of two different decoding algorithms that aggregate the binary predictions into multiclass predictions.

In the second set of experiments, McRUM is applied to the problem of classifying small noncoding RNAs (ncRNAs) to validate the use of the method on a problem of a larger scale than that of the first set of experiments. This second set of experiments deal with a three-class classification problem, specifically, the identification of sequences from two classes of post-transcriptional gene regulatory ncRNAs – mature microRNA (miRNA) and piwi-interacting RNA (piRNA) – from other ncRNAs. This is of interest to small RNA sequencing projects (under 40 nt) where novel miRNAs and piRNAs can be found amidst a set of unannotated reads. For the miRNAs, it is especially interesting since the

miRNA precursors may not be sequenced in those small ncRNA sequencing project, and thus losing the usual avenue of finding novel miRNAs via identification of their precursors [17]. Furthermore, the predictions with McRUM are based solely on the RNA sequences and no additional genomic information is required, which is ideal for the study of organisms whose genomic information is lacking.

The experimental results on datasets taken from the UCI repository together with the preliminary results on small ncRNAs show that, under certain settings, McRUM can achieve comparable or higher accuracy than previous analyses of these problems Thus the results suggest CRUM's potential in solving multiclass problems in bioinformatics and other fields of study.

4.2 Methods

4.2.1 Classification Relevance Units Machine

The sparse kernel-based binary classification model called the Classification Relevance Units Machine (CRUM) obtains probabilistic predictions [21,22]. Let $\mathcal{X}$ be a set of objects; e.g. $\mathcal{X} \subseteq \mathbb{R}^d$. The CRUM models the posterior distribution $p(C_+|x)$ that an object $x \in \mathcal{X}$ is a member of the positive class C_+ using the following model

$$p(C_+|x) = \hat{t}(x) = \sigma\left(\sum_{i=1}^{M} w_i k(x, u_m) + b\right) \tag{4.2.1}$$

where σ is the sigmoid function, M is a positive integer, $k(,)$ is a kernel function, the weights $w_i \in \mathbb{R}$, the bias $b \in \mathbb{R}$, and the Relevance Units (RUs) $u_i \in \mathcal{X}$. The posterior of the negative class is then $p(C_-|x) = 1 - p(C_+|x)$.

For a given $k(,)$, M, an observed dataset $X = \{x_1, x_2, \ldots, x_N\} \subseteq \mathcal{X}$ and the associated class labels $\{c_1, c_2, \ldots, c_N\}$, the binary CRUM learning algorithm first estimates the kernel parameter(s) and u_is through unsupervised learning, and then learns the values of the w_is, and b through an iterative approach. The CRUM learning algorithm minimizes structural risk under log loss [4, 27] and determines the error/complexity trade-off parameter using an empirical Bayes method. Further details can be found in [21, 22].

4.2.2 The Multiclass Classification Problem and Solutions

Multiclass classification is the generalization of binary classification to an arbitrary number of classes $K > 1$. We denote the set of K classes as $\mathcal{T} = \{C_1, C_2, \ldots, C_K\}$, and want to learn a classifier function $g : \mathcal{X} \to \mathcal{T}$.

There are two major approaches to converting a binary classifier to a multiclass classifier: the direct approach and through the aggregation of multiple binary classifiers.

Direct Approach

In the direct approach, the internals of the binary classifier are changed to reflect the K class situation. For CRUM, this is done by changing the underlying model from the binary sigmoid model to a multiclass softmax model,

$$p(C_k|\mathbf{x}) = \frac{\exp\left(\sum_{m=1}^{M} w_{mk} k(x, u_m) + b_k\right)}{\sum_{j=1}^{K} \exp\left(\sum_{m=1}^{M} w_{mj} k(x, u_m) + b_j\right)} \tag{4.2.2}$$

where the M RUs u_m, $M \times K$ weights w_{mi}, and K biases b_k are to be learned. The RUs can be learned using unsupervised learning on the unlabeled data, as done in the binary case [22]. The K times increase in parameters lead to a K^3 increase in the runtime complexity of the CRUM training algorithm compared to the binary case, due to the inversion of the $(MK + 1) \times (MK + 1)$ Hessian matrix. Similar to the RVM, this may make this method impractical for large problems [26]. Furthermore, related work in softmax regression suggests the need for more elaborate and costly methods for matrix inversion due to ill-conditioning [23].

Likewise, reformulating the SVM for multiclass classification leads to high cost training algorithms [15]. Therefore the second approach of aggregating multiple binary classifiers, which we will discuss next, has been the more popular and practical way to solve the multiclass classification problem.

Decomposition of a multiclass problem into binary classification problems

The idea of the aggregation approach is to decompose the multiclass problem into multiple binary problems that can then be solved with binary classifiers. The most popular

framework for this approach is the method of error correcting output codes (ECOC) [20]. In this framework, the decomposition of a K-class problem into L binary problems is expressed with a coding matrix,

$$\mathbf{M} \in \{0, 1, \Delta\}^{K \times L} \tag{4.2.3}$$

where each column of $\mathbf{M}$ specifies one binary classifier.

For example, the one-versus-rest (OVR) matrix for three classes is a 3×3 identity matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2.4}$$

There are three columns and thus this decomposition will require the training of three binary classifiers. The first binary classifier is trained with the training data belonging to class C_1 as the positive class set and the data belonging to classes C_2 and C_3 as the negative class set. The second binary classifier is trained with the training data belonging to class C_2 as the positive class set and the data belonging to classes C_1 and C_3 as the negative set. The third binary classifier is trained similarly. The name of this decomposition is called one-versus-rest (OVR) because each binary classifier is trained with only one class serving as the positive class and all other classes serving as the negative class. In general, the OVR matrix for K classes is the $K \times K$ identity matrix.

The all-pairs (AP) matrix for three classes is also a 3×3 matrix:

$$\begin{pmatrix} 1 & 1 & \Delta \\ 0 & \Delta & 1 \\ \Delta & 0 & 0 \end{pmatrix} \tag{4.2.5}$$

The Δ symbol denotes omission of the class in the training of the binary classifier. Therefore in this case, the first binary classifier is trained with the training data belonging to class C_1 as the positive class set, data from C_2 as the negative class set, and data from C_3 is omitted. The next two binary classifiers are trained in a similar way. This decomposition is called one-versus-one or all-pairs (AP) as each binary classifier is trained with only a

single class serving as the positive class and another single class as the negative class. Since there are $K(K1)/2$ distinct pairs of classes, the general AP matrix for K classes is a $K \times K(K1)/2$ matrix.

In general any coding matrix $\mathbf{M}$ defined by Equation (3) can be used under the following constraints:

1. All rows and columns are unique
2. No row is solely composed of Δ
3. Each column has at least one 1 and 0

Aggregating the binary outputs

Given a coding matrix $\mathbf{M}$ and the outputs of the L binary classifiers, how do we compute the probability $P(C_k|x)$? Let us first consider the simple case of hard decoding, leading to a hard decision. Assume that the binary classifiers g_i, corresponding to the binary classifier specified by the i-th column of $\mathbf{M}$, return hard decisions where an output of 1 denotes the positive class and 0 denotes the negative class. Then the collective output of the binary classifiers on x can be collected into a row vector $\mathbf{g}(x) = [g_1(x), g_2(x), \ldots, g_L(x)]$. The predicted class that x belongs to is determined by finding the row of M with the smallest distance to $\mathbf{g}(x)$. Let $\mathbf{y}, \mathbf{z} \in \{0, 1, \Delta\}^{1\times L}$. A commonly used measure of distance is a modified Hamming distance [2]:

$$d(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{L} \text{cost}(y_i, z_i) \tag{4.2.6}$$

where

$$\text{cost}(a, b) = \begin{cases} 0 & \text{if } a = b \text{ and } a \neq \Delta \\ 1 & \text{if } a \neq b \text{ and } a, b \neq \Delta \\ 1/2 & \text{if } a \text{ or } b = \Delta \end{cases} \tag{4.2.7}$$

Let $\mathbf{M}(k)$ denote the k-th row of $\mathbf{M}$ and,

$$k^* = \text{argmin}_{k \in \{1,2,\ldots,K\}} d(\mathbf{M}(k), \mathbf{g}(x)) \tag{4.2.8}$$

Then the predicted class of x is C_{k^*}. In the case of the AP coding matrices, this would correspond to choosing the class with the majority vote of the binary classifiers. Note that rows of $\mathbf{M}$ can be interpreted as the unique codewords representing the K classes and that the predicted $\mathbf{g}(x)$ is one of those codewords corrupted by noise. In this context, the above algorithm decodes $\mathbf{g}(x)$ into the closest codeword, thus performing error correction on the corrupted bits and giving the name of this approach to classification, ECOC.

Unfortunately, computing the posterior probabilities $p_k = P(C_k|x)$ for all K classes is more difficult. For general coding matrices, the Generalized Bradley-Terry (GBT) model is used to estimate the posterior probabilities [16]. Let I_i^+ and I_i^- denote the set of positive and negative classes, respectively, used by g_i. Then the output $g_i(x)$ is the probability of the positive class of the i-th binary classification problem. Also let N_i denote the number of training data with classes in $I_i = I_i^+ \cup I_i^-$, and the following quantities:

$$q_i = \sum_{k \in I_i} p_k \tag{4.2.9}$$

$$q_i^+ = \sum_{k \in I_i^+} p_k \tag{4.2.10}$$

$$q_i^- = \sum_{k \in I_i^-} p_k \tag{4.2.11}$$

Given the probabilistic outputs of the binary classifiers $\hat{r}_i = g_i(x)$, the core idea of the GBT model is that

$$\hat{r}_i \approx P(I_i^+|x, I_i) = \frac{q_i^+}{q_i} \tag{4.2.12}$$

Through these relations the posterior probabilities $\mathbf{p} = [p_1, p_2, \ldots, p_K]^T$ can be retrieved. This is done by finding the $\mathbf{p}$ that minimizes the negative log-likelihood,

$$-\sum_{i=1}^{L} N_i(\hat{r}_i \log(q_i^+/q_i) + (1 - \hat{r}_i) \log(q_i^-/q_i)) \tag{4.2.13}$$

under the constraints that each $p_k > 0$ and that they sum to unity. This optimization can be interpreted as the minimization of the weighted Kullback-Leibler divergence between $\hat{r}_i$ and q_i^+/q_i. Huang et al. [16] proposed an iterative algorithm to solve this optimization.

Note that the optimization of Equation (13) must be done for every object x that we want to make a prediction on. This could be too expensive in large-scale prediction applications. Furthermore, the computational complexity of the algorithm is not completely characterized. While Huang et al. [16] provide a proof of convergence under some assumptions, under a general decomposition the algorithm may not converge. In the cases that are known to converge, the speed of convergence is unknown. Therefore, a Naïve approach is proposed.

We make the Naïve assumption that the output of each binary classifier is independent. Under the interpretation of error-correcting codes, the formulation below is a soft-decoding of the observed $\mathbf{g}(x)$ to the codewords in $\mathbf{M}$ under the assumption that bit errors are independent. Then we can compute the class posteriors as simple products of the binary posteriors, as follows

$$P(C_k|x,\mathbf{M}) = \prod_{i=1,\mathbf{M}_{ki}\neq\Delta}^{L} P(I_i^+|x,I_i)^{\mathbf{M}_{ki}} P(I_i^-|x,I_i)^{1-\mathbf{M}_{ki}} \tag{4.2.14}$$

$$\approx \prod_{i=1,\mathbf{M}_{ki}\neq\Delta}^{L} g_i(x)^{\mathbf{M}_{ki}} (1-g_i(x))^{1-\mathbf{M}_{ki}} \tag{4.2.15}$$

where the output of classifiers not trained on data from class C_k are omitted. For example, from the decomposition given in Equation (5), $P(C_2|x,\mathbf{M}) = (1-g_1(x))g_3(x)$. Given the outputs of the binary classifiers, the algorithm is linear in L. In the implementation log of Equation (14) is used for computational stability as shown in Step 4 of Algorithm 2.

The above formulation is a generalization to any valid $\mathbf{M}$ of the Resemblance Model for AP decomposition proposed in [12]. Again, the key assumption is the independence of the L binary classifiers, which is highly dependent on the decomposition $\mathbf{M}$. Thus in general, this method is possibly only a crude approximation.

The following pseudocodes summarize the training and prediction processes of McRUM.

Input: $K \times L$ decomposition matrix $\mathbf{M}$, labeled training data

for $i \leftarrow 1$ to L **do**
  $positive\_data \leftarrow$ initialize as empty
  $negative\_data \leftarrow$ initialize as empty
  for $j \leftarrow 1$ to K **do**
    if $\mathbf{M}_{ij} = 1$ **then**
      Add data from class j to $positive\_data$
    else if $\mathbf{M}_{ij} = 0$ **then**
      Add data from class j to $negative\_data$
    end if
  end for
  Set g_i to binary CRUM trained on $positive\_data$ and $negative\_data$ as described in [22]
end for
return $\mathbf{g} = [g_1, g_2, \ldots, g_L]$

Algorithm 4: Training McRUM

Input: $K \times L$ decomposition matrix $\mathbf{M}$, L binary CRUMs g_i, input x

$\mathbf{p} \leftarrow [0, 0, \ldots, 0]$
for $i \leftarrow 1$ to K **do**
  for $j \leftarrow$ to L **do**
    $p_i \leftarrow p_i + \ln \left\{ g_j(x)^{\mathbf{M}_{ij}} (1 - g_j(x))^{(1-\mathbf{M}_{ij})} \right\}$
  end for
end for
$\mathbf{p} = \exp(\mathbf{p}) / \sum \exp(\mathbf{p})$ (normalize to ensure the posterior probabilities sum to 1)
return $\mathbf{p}$

Algorithm 5: Prediction

Optimal Coding Matrix

The next question is whether there is any theory that can help guide us to designing the optimal coding matrix that gives the smallest error. There is, but it is not practically useful. These are some of the properties that would achieve a good ECOC-based classifier [2]:

1. The minimum distance (using Hamming distance, Equation (4.2.6)) between rows of $\mathbf{M}$ should be maximized
2. The number of Δ entries should be minimized
3. The average error of the L binary classifiers should be minimized

All the criteria are at odds with each other. Consider OVR decomposition, Equation (4), again. Since all but one class is considered to be in the negative class, the training data is likely to be imbalanced. To see why this is a problem, let us consider an extreme case where 99% of the training data is negative and only 1% of the data is positive. Then a binary classifier that always predicts the negative class would achieve 1% error. Under the framework of empirical or structural risk minimization, classifier training would tend to converge to his solution as it provides low empirical risk under 0-1 loss. Therefore a large imbalance between the size of the positive and negative sets would bias the classifier against the smaller class. So while OVR does not have any Δ entries, the average error of the binary classifiers could be high.

In the case of the AP decomposition shown in Equation (5), each individual binary classifier only has a single class serving as the positive data and another single class serving as the negative. If the overall training set was balanced between all K classes, then each of the binary classifiers will also be balanced and have good average error. On the other hand, the AP matrix contains many Δ entries, which is a force that increases error. As a side effect, each individual binary classifier will be faster to train compared to OVR, as the amount of data per binary classifier is reduced. This can be overshadowed by the sheer number of classifiers to train if K is large.

Therefore knowing which coding matrix is superior to another a priori is not possible and the choice of coding matrix $\mathbf{M}$ is application-dependent. So we must ex-

perimentally try different matrices to find which one is the best suited to the particular application.

4.2.3 ncRNA dataset preparation and features

The ncRNA dataset is gathered from mirBase's collection of miRNA [18], NONCODE 3.0s collection of piRNA [6], and the remaining ncRNAs in the NONCODE 3.0 database serve as representatives of other ncRNAs. Each of these three sets is individually reduced using CD-HIT [19] to remove sequences with 80% or higher identity. This helps reduce the evolutionary correlations among the data and improves the generalization of the CRUM model that assumes an independent sample. The resulting dataset contains 9,439 miRNAs, 81,147 piRNAs, and 94,809 other ncRNAs.

In the gathered data, miRNAs are observed to be 15 33 nt long and piRNAs are observed to be 16 40 nt long. For the other ncRNAs, the training and evaluation of McRUM does not necessarily use the entire sequence. We chose to use fragments of length 20 nt, which is in the overlapping range of the lengths between miRNAs and piRNAs, so that the fragment has the possibility of being an miRNA or piRNA had the identity of the fragment been unknown. If the other ncRNA sequence is of length longer than 20 nt, we take a random fragment of 20 nt from the sequence instead. Due to the imbalance of the dataset among the three classes, the training set is a sample of the available data. After holding out 20% of the miRNA sequences for an independent test set, we are left with 7,552 miRNAs in the training set. Therefore we sample 7,552 piRNAs and other ncRNAs each to form a balanced 1:1:1 training set. Together with the hold out of 1,887 miRNAs, the remaining 73,595 piRNAs and 87,257 other ncRNAs serve as an independent test set.

Since mature miRNAs and piRNAs lack strong secondary structures, internally McRUM represents each ncRNA using k-mers, for $k = 1, 2, \ldots, 5$. For each value of k, the number of occurrences of each type of k-mer is computed and normalized.

4.2.4 Performance measures

Receiver Operating Characteristic (ROC) curve is a visualization of performance of binary classifiers at various thresholds. On the x-axis is the false positive rate (FPR) and

on the y-axis is the true positive rate (TPR), which is also called sensitivity or recall. These two quantities are calculated as follows,

$$FPR = FP/(FP + TN) \tag{4.2.16}$$

$$TPR = TP/(TP + FN) \tag{4.2.17}$$

where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives. For classification of more than two classes, we can compute ROC curves by considering one class as the positive class and the remaining classes jointly as the negative class. For the small ncRNA experiment, we have three classes. For Figures 4.1 and 4.5, we consider miRNA as the positive class, and piRNA and other ncRNAs jointly as the negative class. Under this setting TP is the count of miRNAs correctly classified, while FP is the number of piRNAs and other ncRNAs classified as miRNAs. FN is the sum of the number of miRNAs incorrectly classified and the number of miRNAs unclassified. Lastly, TN is the sum of the number of piRNA and other ncRNAs correctly classified, the number of piRNAs incorrectly classified as other ncRNAs, the number of other ncRNAs incorrectly classified as piRNA, and the number of piRNAs and other ncRNAs left unclassified. The piRNAs incorrectly classified as other ncRNAs, and vice versa, are counted as true negatives because they are correctly classified into the negative class and not into the positive class, miRNA. Similarly, Figures 4.2 and 4.6 are computed by considering piRNA as the positive class and miRNA and other ncRNAs jointly as the negative class, while Figures 4.3 and 4.7 are computed by considering other ncRNAs as the positive class and miRNA and piRNA jointly as the negative class.

The timing results for the Naïve and GBT decoding algorithms in the benchmark experiments were obtained using MATLAB implementations on a PC with a 2.83 GHz Intel Core 2 Quad processor and 8 GB of memory.

4.3 Results and Discussion

In this section we present two sets of experiments: benchmark experiments and small ncRNA experiments. The purpose of benchmark experiments is to assess performance of McRUM for four different decomposition settings and the two different decoding algorithms. For the experiments, we use a group of datasets from the UCI Machine Learning Repository [3], which is a collection of databases widely used for empirical analysis of algorithms in the machine learning community. The performance of McRUM with different settings is presented in terms of prediction accuracy. The small ncRNA experiments are conducted to validate the use of McRUM on a larger scale problem. Through the benchmark experiments, Naïve decoding algorithm is proven to be much more computationally efficient than the GBT algorithm with its AP and OVR performances being close to the GBT results. Therefore, in the ncRNA experiments, only the Naïve McRUMs with AP and OVR settings are tested and the ROC analysis is used to illustrate how the performance of selected McRUM models changes at various decision thresholds.

For both sets of experiments, we also run the multiclass SVM implemented in LIBSVM [7] to illustrate McRUMs performance relative to SVM-based approaches. The LIBSVM implementation uses the all-pairs decomposition, Platt's method to generate probabilities from the individual binary SVMs, and their own algorithm for aggregating the all-pairs binary results to multiclass posterior probabilities.

4.3.1 Benchmark experiments

For the experiments, we try McRUM on five small datasets from the UCI Machine Learning Repository website [3] using the binary CRUM classifiers with the Gaussian kernel. The kernel width γ and model complexity M is chosen via K-means clustering of the unlabeled training dataset [22]. For a range of values of M, K-means clustering is applied to group the unlabeled training dataset into M clusters and the Akaike Information Criterion (AIC) is computed by giving the clustering results a probabilistic interpretation. The number of clusters with the best AIC value is selected for M. Furthermore, we set $\gamma = (2d^2)^{-1}$, where d is the maximum distance between cluster centers obtained by the K-means clustering with K $= M$. Using K-means clustering to set parameters as described

above has clear computational advantages, which can be critical when training is performed on large datasets. The same γ and M are used for all the individual binary classifiers per dataset.

Throughout the benchmark experiments, we consider the following decompositions: (i) all-pairs (AP), (ii) one-versus-rest (OVR), (iii) random dense, and (iv) random sparse. Random coding matrices M are generated with and without using symbols for the random sparse and random dense cases, respectively. For each random type, 100 random M are generated and the M with the smallest minimum distance among its rows is chosen. Controlling the number of columns in the random sparse matrix, we can aim to create a decomposition that is a compromise between AP and OVR. This is useful should the number of classes K be large and AP impractical, while still retaining some of the reduced training set benefits per binary classifier. The details of the different decompositions are given in the Methods section.

The class label is assigned based on which class has the largest posterior probability, as determined by the Naïve and generalized Bradley-Terry (GBT) decoding algorithms [16]. Since the GBT algorithm is not guaranteed to converge, a maximum of 1000 iterations of the algorithm is imposed. We first examine results using cross-validation.

Wine dataset

	Naïve		GBT	
Decomposition	Train Acc	Test Acc	Train Acc	Test Acc
AllPairs	99.44 (0.20)	97.78 (2.87)	99.44 (0.20)	97.78 (2.87)
OneVsRest	99.44 (0.20)	97.75 (3.92)	99.44 (0.20)	97.75 (3.92)
Dense	84.40 (1.09)	83.63 (6.99)	99.44 (0.20)	98.30 (2.74)
Sparse	91.01 (1.76)	90.00 (6.31)	99.50 (0.26)	97.22 (4.72)

Table 4.1. McRUM results on wine dataset using 10-fold cross-validation. Train/Test Acc is the mean accuracy on the training/test dataset. The standard deviation is shown in the parenthesis. (AP: all-pairs, OVR: one-versus-rest, Dense: random dense, Sparse: random sparse)

The three-class wine dataset contains 178 instances [1]. The objective is to determine the origin of the wines based on 13 physicochemical measurements. The number of binary classifiers for AP and OVR are both three and so we set the dense and sparse

decompositions to also use three classifiers. The mean accuracy and its standard deviation results in Table 4.1 are computed via 10-fold cross-validation. The AP and OVR decompositions give nearly the same results regardless of using the Naïve or GBT algorithms. There is a significant reduction in accuracy for the Naïve algorithm for the random dense and sparse cases, but the GBT results remain close to the Naïve algorithm's AP and OVR performance.

The prediction running-times are also measured for the AP and OVR decompositions where the Naïve and GBT algorithms show comparable predictive performance. As shown in Table 4.2, Naïve algorithm achieves a three orders of magnitude speed-up compared to GBT algorithm for both AP and OVR decomposition.

We observed the mean accuracies of 99.69% (std = 0.33) and 98.89% (std = 2.34) from Gaussian SVM for the training and test set, respectively, which is comparable to the AP and OVR McRUM results of 99.44% (std = 0.20) and 97.78% (std = 2.87). In addition, the best mean accuracy reported for a 10-fold cross-validation using a multiclass RVM for this wine dataset is 96.24% [24], which, considering the standard deviation, is also comparable to the best achieved here using McRUM.

Iris dataset

Table 4.3 gives the 10-fold cross-validation results on the three-class iris dataset, a classic machine learning and statistics benchmark. The problem is to classify flowers of three species of the genus *Iris* based on four physical measurements from a sample of 150 flowers [10]. We see similar results as earlier, where the Naïve and GBT accuracies are nearly the same and the performance of Naïve algorithm drops severely under the random decompositions. Three binary classifiers are used in all cases. As for prediction running-times, as shown in Table 4.2, there is a three order of magnitude speed-up with the Naïve compared to the GBT algorithm with minimal impact to predictive performance, as also seen with the wine dataset results previously.

The mean accuracies on the training and test set we observed from Gaussian SVM are 97.85% (std = 0.65) and 96.67% (std = 3.51). The best mean accuracy reported for a 10-fold cross-validation using a multiclass RVM for this iris dataset is 93.87% [24].

	Naïve		GBT	
	AP	OVR	AP	OVR
wine	0.001868 (0.000206)	0.001662 (0.000164)	7.373405 (1.182947)	1.073139 (0.197278)
iris	0.001409 (0.000139)	0.001376 (0.000141)	6.625228 (0.678363)	2.908828 (0.834858)
yeast	0.040737 (0.000534)	0.0492920 (0.001219)	2722.317544 (117.545388)	2795.766035 (139.314294)
thyroid	0.131689	0.123968	939.692526	179.426632
satellite	0.239550	0.139304	10612.598301	2816.632703

Table 4.2. Prediction time of McRUM on benchmark datasets (in seconds). We provide the prediction time only for AP and OVR cases since their predictive performances are competitive to each other for all benchmark datasets while the performances for random decompositions are much lower than the AP and OVR cases with Naïve algorithm. The prediction time is averaged over 10-fold cross-validation for the first three datasets while it is estimated once for the last two datasets as their explicit partitioning of training and test sets were given. Included in the parenthesis is the standard deviation of the prediction time. (AP: all-pairs, OVR: one-versus-rest)

	Naïve		GBT	
Decomposition	Train Acc	Test Acc	Train Acc	Test Acc
AllPairs	97.56 (0.70)	96.00 (5.62)	97.56 (0.70)	96.00 (5.62)
OneVsRest	97.85 (0.82)	96.67 (5.67)	97.85 (0.82)	96.67 (5.67)
Dense	67.56 (2.88)	68.00 (16.27)	97.70 (0.74)	96.00 (6.44)
Sparse	66.67 (1.16)	66.67 (10.42)	97.78 (0.92)	95.33 (5.49)

Table 4.3. McRUM results on iris dataset using 10-fold cross-validation. Train/Test Acc is the mean accuracy on the training/test dataset. The standard deviation is shown in the parenthesis. (AP: all-pairs, OVR: one-versus-rest, Dense: random dense, Sparse: random sparse)

Again, both multiclass SVM and RVM results are comparable to the best achieved here with McRUM (97.85% (std = 0.82) and 96.67% (std = 5.67)).

Yeast dataset

Decomposition	Naïve Test Acc	GBT Test Acc
AllPairs	59.43 (6.27)	59.10 (6.21)
OneVsRest	58.90 (5.29)	59.51 (4.02)
Dense	3.43 (1.49)	57.75 (3.11)
Sparse	2.02 (1.62)	59.23 (2.79)

Table 4.4. McRUM results on yeast dataset using 10-fold cross-validation. Test Acc is the mean accuracy on the test dataset. The standard deviation is shown in the parenthesis. Note that, due to the large dataset size and the high computational complexity of the GBT algorithm, the mean accuracy on the training partitions cannot be provided. (AP: all-pairs, OVR: one-versus-rest, Dense: random dense, Sparse: random sparse)

Table 4.4 gives the 10-fold cross-validation for the 10-class yeast dataset that contains 1,484 proteins. The goal is to predict the cellular localization site of a protein based on eight sequence-related measures [14]. 18 binary classifiers are used for the random sparse decomposition to achieve a compromise between AP and OVR. For the random dense decomposition, we used 34 classifiers as a sample of the columns of the complete dense decomposition that contains all possible partitions of the data into positive and negative classes. Only the mean accuracy on the test partitions are given, as the dataset is large enough that computing the accuracy of the training partitions is prohibited by the complexity of the GBT algorithm. The extra effort of the GBT algorithm is required in the random

decompositions cases where the Naïve results are dismal. The results for AP and OVR, however, are comparable between Naïve and GBT.

For this dataset, Gaussian SVM shows 60.85% accuracy with standard deviation of 4.08. Best results from AP McRUM (59.43% (std = 6.27)) and OVR McRUM (59.51% (std = 4.02) are again comparable to the SVM results considering the standard deviation. The results from both McRUM and SVM are an improvement over the 56.5% achieved in the dataset's original analysis using PSORT [13] and are comparable to the 59.51% (standard deviation of 5.49) obtained by k-NN with PSORT II on this yeast dataset [14].

The prediction running-times in Table 4.2 show five orders of magnitude speed-up for the AP and OVR McRUMs using the Naïve algorithm over the GBT algorithm. This very significant speed-up is achieved with minimal impact to the predictive performance, as discussed above.

Thyroid disease dataset

	Naïve		GBT	
Decomposition	Train Acc	Test Acc	Train Acc	Test Acc
AllPairs	98.44	97.29	97.93	97.11
OneVsRest	95.47	95.22	95.52	95.04
Dense	92.47	92.71	94.38	93.90
Sparse	24.68	25.50	97.11	96.18

Table 4.5. McRUM results on thyroid training and test datasets. Train/Test Acc is the accuracy on the training/test dataset. Note that an explicit partitioning of the data into training and test sets were provided. Therefore, cross-validation experiment was not performed and, as a result, no information on standard deviation is available. (AP: all-pairs, OVR: one-versus-rest, Dense: random dense, Sparse: random sparse)

The results in Table 4.5 are from the three-class thyroid dataset that has 3,772 and 3,428 training and testing instances, respectively. The problem is to determine if a patient has hypothyroid based on 21 attributes [9, 25]. The number of binary classifiers for AP and OVR are both three and so we set the dense and sparse decompositions to also use three classifiers. Regardless of the posterior decoding algorithm, the AP decomposition performs with the best accuracy. The Naïve and GBT algorithms perform similarly on all but the sparse decomposition. It appears the independent binary classifiers assumption fails

in the sparse case and the Naïve algorithm does not approximate the posteriors for class 2 and 3 as well as the GBT algorithm does. The SVM results we obtained for this dataset are 96.29% and 94.37% for training and test accuracy, respectively. The AP results from McRUM (98.44% and 97.22%) are slightly better.

As shown in Table 4.2, under the AP decomposition, it takes about 0.132 seconds to compute the predictions for the entire test set using the Naïve algorithm. On the other hand, with a maximum of 1,000 iterations per datapoint, the GBT algorithm takes about 934.693 seconds, almost four orders of magnitude longer than the Naïve algorithm for nearly the same predictive accuracy. In the case of OVR, the predictions times for the test set are 0.124 and 179.427 seconds for Naïve and GBT algorithms, respectively. The OVR decomposition appears to be an easier problem for the GBT algorithm to solve than the AP decomposition despite using the same number of binary classifiers.

Landsat satellite (statlog) dataset

	Naïve		GBT	
Decomposition	Train Acc	Test Acc	Train Acc	Test Acc
AllPairs	89.85	88.05	89.76	87.70
OneVsRest	89.56	87.65	89.58	87.50
Dense	10.60	11.85	80.56	77.95
Sparse	23.40	23.50	85.73	84.55

Table 4.6. McRUM results on satellite image training and test datasets. Train/Test Acc is the accuracy on the training/test dataset. Note that an explicit partitioning of the data into training and test sets was provided. Therefore, crossvalidation experiment was not performed and, as a result, no information on standard deviation is available. (AP: all-pairs, OVR: one-versus-rest, Dense: random dense, Sparse: random sparse)

Table 4.6 gives the results on a satellite image (statlog) dataset that has six classes with 4,435 and 2,000 training and testing instances, respectively. The goal is to interpret a scene based on 36 multi-spectral values [9]. Here for both training and test accuracy, both the AP and OVR under Naïve and GBT decoding perform at about the same level. OVR uses 6 binary classifiers, while AP uses 15. To strike a halfway point between the two, the random decompositions use 10 binary classifiers. However, for these random decompositions, the Naïve algorithm is outperformed by GBT. The SVM results we obtained for this

dataset are 98.35% and 91.50% for training and test accuracy, respectively. For training set, SVM is superior to AP and OVR McRUMs (89.85% and 89.76%) and both McRUMs performance on test set (88.05% and 87.70%) is also slightly worse than the SVM.

Table 4.2 shows that, for the test set, it takes about 0.240 and 10,612.598 seconds for the Naïve and GBT algorithms to compute the predictions under the AP decomposition, respectively. For the OVR decomposition, the prediction times are 0.139 and 2,816.633 seconds for Naïve and GBT algorithms, respectively. Again the OVR is the faster of the two decompositions, partly because of the reduced number of binary classifiers.

4.3.2 Small ncRNA experiments

To validate McRUM on a larger scale problem and to explore its use for the task of NGS data analysis, we investigated the classification of mature miRNAs and piRNAs from other ncRNAs. This is a problem of interest in the analysis of small RNA sequencing (RNA-seq) data. Further details of the dataset and sequence features used by McRUM are given in the Methods section. For this experiment, two McRUM models are used for the AP and OVR settings using the Naïve decoding algorithm, and their performance is illustrated relative to the Gaussian and linear multiclass SVMs.

Cross-validation experiments

Figures 4.1 through 4.3 show ROC curves under 3-fold cross-validation experiments. Note that, because ROC curves are defined for binary classification, each ROC curve considers one of the three classes as the positive class, while considering the remaining two classes jointly as the negative class. For example, Figure 4.1 considers miRNA as the positive class and piRNA and other ncRNA as the negative class. The details of the computation of the ROC curves are given in the Methods section. The following results are obtained using the best observed model complexity, $M = 600$, for the underlying binary CRUM models using the Gaussian kernel. The kernel width γ is chosen using K-means clustering of the unlabeled training dataset with the selected M. For multiclass SVM, we trained Gaussian SVMs and linear SVMs where, on average, about 3,725 support vectors were used in each binary Gaussian SVM and about 4,048 for each binary linear SVM.

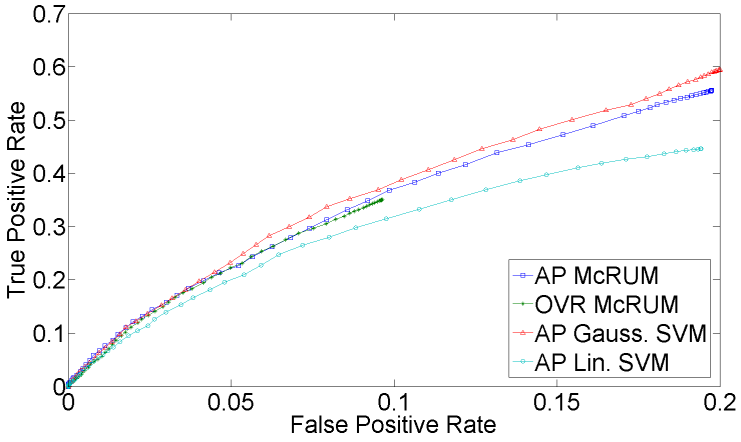


Figure 4.1. 3-fold cross-validation results for miRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

Figure 4.1 suggests that classifying mature miRNA is a very difficult problem, as even at the highest observed FPR (about 0.2), the TPR only approaches 0.6 for both McRUM and SVM. There is no clear superior decomposition for McRUM. However, the OVR McRUM occupies a narrower range of FPRs and cannot achieve as high a TPR as the AP case. Although both versions of McRUM perform slightly better than linear SVM and slightly worse than Gaussian SVM, the performances are all very similar. Based on these results and those given below, it is seen that mature miRNAs often get confused with ncRNAs other than piRNA.

In contrast to miRNA, Figure 4.2 suggests that the classification of piRNA is a much easier problem. The TPR approaches 0.9 even at a low FPR of 0.09 for McRUM and 0.05 for Gaussian SVM. The ROC curves for AP and OVR McRUMs are comparable while Gaussian SVM shows higher performance and linear SVM works very poorly. The poor performance of linear SVM may be due to the non-linear nature of the problem. Although the Gaussian SVM shows superior performance, both AP and OVR McRUMs also achieve similar level of TPR by sacrificing FPR slightly. Furthermore, given more compact models and better computational efficiency of binary CRUM over SVM [21], McRUM can still be a favorable choice for large-scale prediction problems.

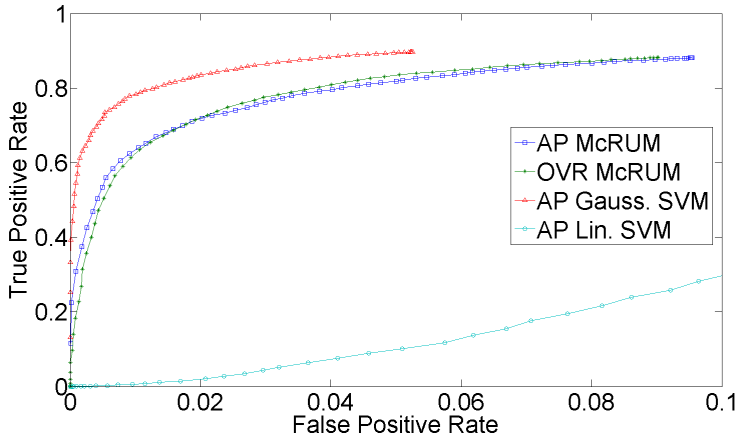


Figure 4.2. 3-fold cross-validation results for piRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

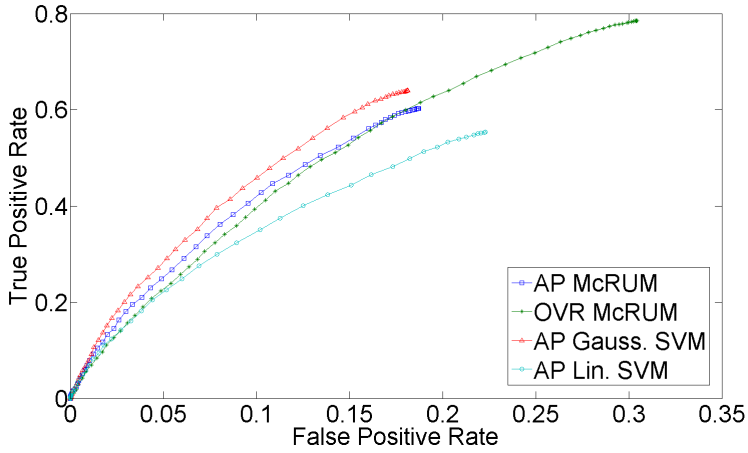


Figure 4.3. 3-fold cross-validation results for non-miRNA and non-piRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

Finally, Figure 4.3 shows that discriminating the class consisting of other ncRNAs is also difficult, but not as difficult as the miRNA case. Again, the performance suffers due to the difficulty of discriminating miRNA from other ncRNAs. While linear SVM

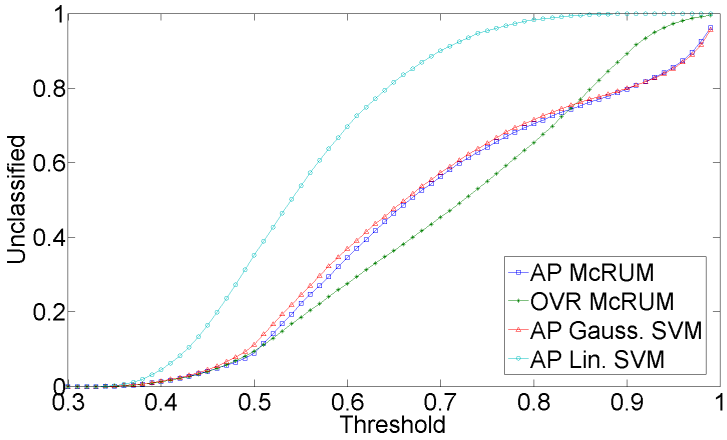


Figure 4.4. The fraction of unclassified sequences for cross-validation experiment. The fraction of the validation set left unclassified for the AP and OVR McRUMs and the Gaussian and linear SVMs at different posterior probability threshold values..

shows poor performance, in the region of overlapping FPRs, the ROC curves for Gaussian SVM and both OVR and AP McRUMs are comparable. However, the OVR McRUM has a wider FPR range, allowing it to achieve a high TPR of about 0.8 at an FPR of about 0.3, that the AP McRUM and Gaussian SVM cannot obtain.

In Figure 4.4 we provide a plot showing the fraction of the validation set left unclassified for the AP and OVR McRUMs and the Gaussian and linear SVMs at different posterior probability threshold values. In summary, linear SVM is uniformly worse than the others, AP McRUM and Gaussian SVM have almost the same fraction of unclassified sequences, and OVR McRUM shows the smallest number of unclassified sequences across a wide range of threshold values.

Independent test experiments

Further evaluation of McRUM and the multiclass SVMs on a larger, independent dataset was also conducted and the ROC curves are given in Figures 4.5 through 4.7. McRUMs are trained with $M = 600$ and SVMs are trained with about 5,410 support vectors on average for each binary Gaussian SVM and about 6,034 for each binary linear SVM using the entire training dataset used in the cross-validation experiment. Figure 4.5 reiterates that the miRNA case is very difficult, showing results similar to the cross-validation

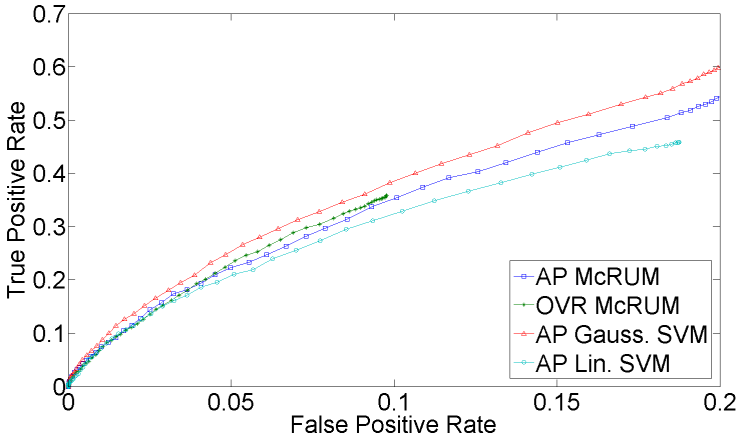


Figure 4.5. Evaluation results on independent test set for miRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

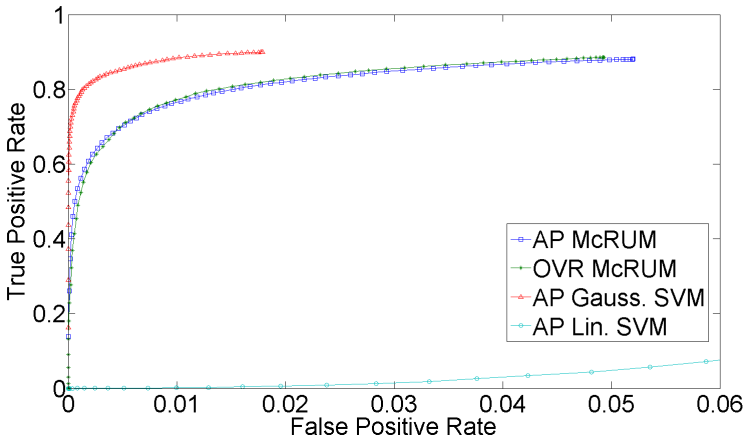


Figure 4.6. Evaluation results on independent test set for piRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

experiment. For the piRNA case shown in Figure 4.6, we again see very good performance from Gaussian SVM and both the AP and OVR McRUMs, and extremely poor performance from linear SVM. Note that the scale of the FPR axis is very small. As described above, AP

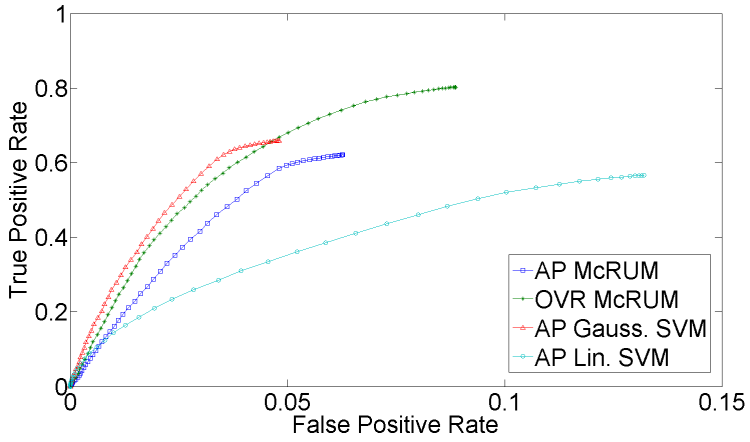


Figure 4.7. Evaluation results on independent test set for non-miRNA and nonpiRNA being the positive class. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01 for the two McRUM models (AP and OVR settings) with the Naïve decoding algorithm, and for the Gaussian and linear SVMs with all-pairs decomposition.

and OVR McRUMs can achieve the similar level of TPR as Gaussian SVM by sacrificing FPR slightly, about 0.03. In addition, McRUMs are computationally more efficient than the multiclass SVMs since the binary CRUM is less expensive computationally than the binary SVM [21]. To compute the class posterior probabilities of the entire test dataset, the SVMs take about 2.5 to 3 times longer than McRUMs even though the SVM implementation is in faster compiled C++, whereas McRUM is implemented in MATLAB, a slower interpreted language. (OVR McRUM takes 3,203.35 sec, AP McRUM 3,429.91 sec, Gaussian SVM 11,015.29 sec, and linear SVM 8,444.61 sec.) Lastly in Figure 4.7, we see the remaining ncRNAs showing good results with Gaussian SVM and the OVR McRUM that are slightly better than the AP McRUM.

In Figure 4.8, we provide a plot showing the fraction of the test set left unclassified for the AP and OVR McRUMs and the Gaussian and linear SVMs at different posterior probability threshold values. The results are very similar to those obtained for the cross-validation experiments shown in Additional file 1.

Recently, a Fisher Linear Discriminant (FLD) based classifier called piRNApredictior has been proposed for binary piRNA classification by Zhang et al. [29]. We have downloaded the script from their website [28] and trained it with the training dataset used

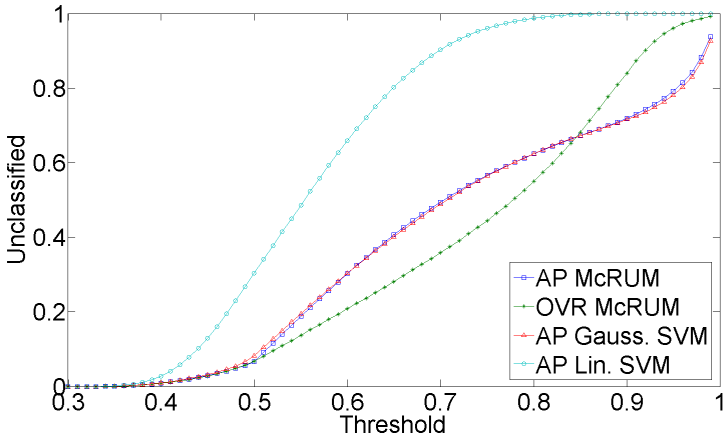


Figure 4.8. The fraction of unclassified sequences for independent test experiment. The fraction of the test set left unclassified for the AP and OVR McRUMs and the Gaussian and linear SVMs at different posterior probability threshold values.

in [29] (FLD_1) and also with our own training dataset (FLD_2). Then the resulting classifiers were evaluated on our test dataset. For the prediction step, both FLD-based classifiers constrain the input sequence being at least 25 nt. We removed this constraint in the script as our test dataset contains many ncRNAs (both piRNAs and non-piRNAs) shorter than 25 nt.

The ROC curves generated from observed results of the FLD-based classifiers are presented in Figures 4.9 and 4.10 along with the ROC curves for McRUM and SVM results. Both AP and OVR McRUMs and Gaussian SVM clearly show superior predictive performance over the FLD-based classifiers. One reason for the lower performance of FLD-based classifiers can be the possible nonlinear boundary between the classes of ncRNAs under the features considered. The nonlinearity of the problem is also evident from the extremely poor performance of linear SVM.

Note that about 99% of the sequences in the positive training dataset for FLD_1 are from NONCODE 2.0s collection of piRNA. Our positive test dataset is gathered from a later version of NONCODE database and, as a result, 98.57% of the sequences in our positive test dataset are already included in the positive training dataset used for FLD_1. Therefore the prediction results may be biased in favor of FLD_1. Then, FLD_2 showing better performance than FLD_1 may seem contradictory when the training set used for

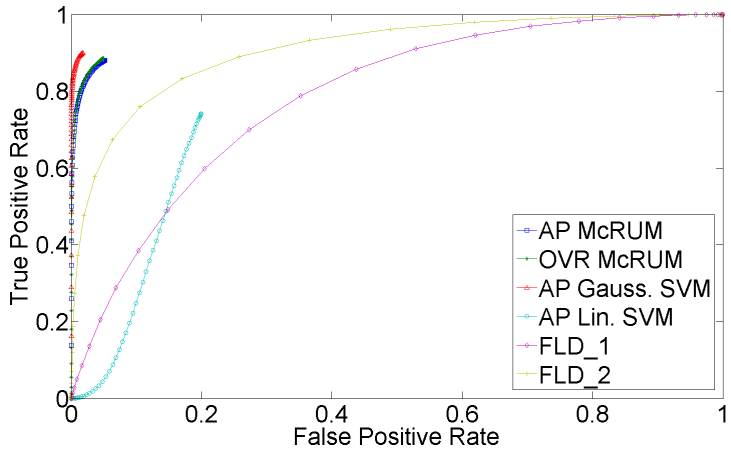


Figure 4.9. piRNA prediction results. The ROC curves for FLD-based classifiers are generated from observed prediction results on our test dataset with the threshold varying from -0.2 to 0.46 in increments of 0.001. (-0.2 and 0.46 are the lower and upper bounds of the observed predicted values of FLD-based classifiers.) FLD_1 classifier was trained with the dataset used in [29] while FLD_2 used our training dataset. McRUM and SVM results are the same as shown in Figure 4.6.

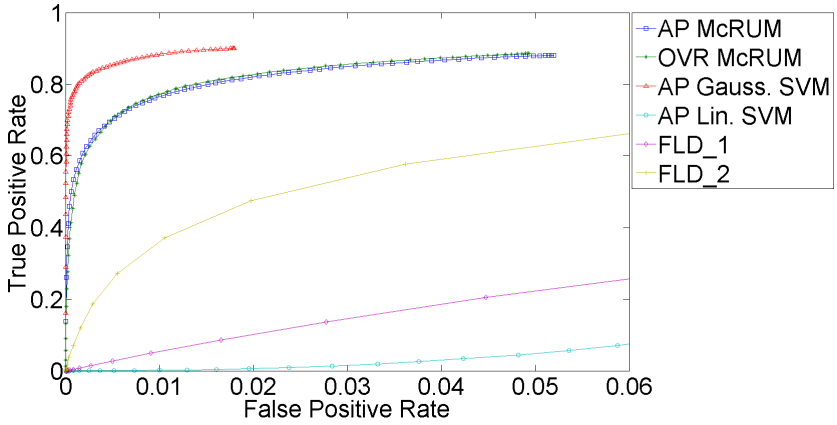


Figure 4.10. piRNA prediction results. The ROC curves in Figure 4.9 are zoomed in to match the FPR range shown in Figure 4.6. The ROC curves for FLD-based classifiers are generated from observed prediction results on our test dataset with the threshold varying from -0.2 to 0.46 in increments of 0.001. (-0.2 and 0.46 are the lower and upper bounds of the observed predicted values of FLD-based classifiers.) FLD_1 classifier was trained with the dataset used in [29] while FLD_2 used our training dataset. McRUM and SVM results are the same as shown in Figure 4.6.

FLD_2 is independent of the test set. It can be because FLD_1 is not specifically trained on the ncRNAs shorter than 25 nt. The training dataset for FLD_1 contains about 4.67% ncRNAs shorter than 25 nt while our training dataset used for FLD_2 contains 66.41% sequences shorter than 25 nt. In the test dataset, 55.93% of the sequences in the test dataset are shorter than 25 nt, for which correct prediction can be hard for FLD_1.

4.4 Conclusions

In this study, the binary CRUM model is generalized to the multiclass setting via ECOC framework. The probabilistic nature of the binary CRUM is preserved using either the GBT or the proposed linear-time decoding algorithms. The proposed linear-time algorithm allows for efficient application to large-scale prediction settings, where the GBT algorithm's complexity is prohibitive, while still maintaining comparable predictive performance under certain decompositions of the given multiclass problems, as evidenced by the benchmark experiments. The applicability of McRUM to larger scale problems is demonstrated by an analysis of small ncRNA sequences. The results demonstrate that McRUM can be an advantageous solution to resolve multiclass problems especially when applied to large datasets.

The preliminary results on small ncRNA classification presented in this paper demonstrate that McRUM has potential in addressing the problem of classifying small ncRNAs. In this study, we restricted the length of the other ncRNA fragments to be maximum of 20 nt, but we plan to conduct further experiments with various lengths of fragments. We also plan to include short byproducts of small RNA biogenesis, such as miRNA*, in the class of other ncRNAs. In the future, we will also extend the current study by including other classes of small ncRNAs and optimizing the use of McRUM for large-scale datasets such as those generated by NGS sequencing projects. Features other than the simple k-mers will be considered to improve the predictive performance, especially for classifying the mature miRNAs. Finally, the interesting preliminary results obtained by the multiclass Gaussian SVM on the problem of small ncRNA classification show that it could be an advantageous alternative to McRUM on smaller datasets and thus we intend to develop in tandem both classifiers for further experiments. The resulting small ncRNA classifiers will

be integrated into a combined prediction tool that will offer both the multiclass SVM and McRUM options providing more alternative choices to users.

Competing interests

The authors declare that there are no competing interests.

Authors' contributions

MM implemented McRUM and performed the experiments. All authors participated in the design of the study, development of CRUM and the data analyses. KB and GP supervised and coordinated the whole research work. All authors have read, revised and approved the final manuscript.

Acknowledgements

This work is supported in part by NIH Grants from the National Institute of General Medical Sciences (P20GM103516 and P20GM103466). The paper's contents are solely the responsibility of the authors and do not necessarily represent the official views of the NIH.

Bibliography

[1] S. Aeberhard, D. Coomans, and O. De Vel, "Comparison of classifiers in high dimensional settings," *Dept. Math. Statist., James Cook Univ., North Queensland, Australia, Tech. Rep*, no. 92-02, 1992.

[2] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2000.

[3] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[4] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, pp. 463–482, 2002.

[5] N. Blom, T. Sicheritz-Ponten, R. Gupta *et al.*, "Prediction of post-translational glycosylation and phosphorylation of proteins from the amino acid sequence," *Proteomics*, vol. 4, pp. 1633–1649, 2004.

[6] D. Bu, K. Yu, S. Sun, C. Xie, G. Skogerb, R. Miao, H. Xiao, Q. Liao, H. Luo, G. Zhao, H. Zhao, Z. Liu, C. Liu, R. Chen, and Y. Zhao, "Noncode v3.0: integrative annotation of long noncoding rnas," *Nucleic Acids Research*, 2011.

[7] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[8] C. H. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[9] W. Duch. Datasets used for classification: comparison of results. [Online]. Available: http://www.is.umk.pl/projects/datasets.html

[10] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[11] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[12] T. Hamaura, H. Mizutani, and B. Irie, "A multiclass classification method based on multiple pairwise classifiers," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 809–813.

[13] P. Horton and K. Nakai, "A probabilistic classification system for predicting the cellular localization sites of proteins," in *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1996, pp. 109–115.

[14] ——, "Better prediction of protein cellular localization sites with the it k nearest neighbors classifier," in *Ismb*, vol. 5, 1997, pp. 147–152.

[15] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions in Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[16] T. Huang, R. C. Weng, and C. Lin, "Generalized bradley-terry models and multi-class probability estimates," *Journal of Machine Learning Research*, vol. 7, pp. 85–115, 2006.

[17] Y. Huang, Q. Zou, S. Wang, S. Tang, G. Zhang, and X. Shen, "The discovery approaches and detection methods of micrornas," *Molecular Biology Reports*, vol. 38, pp. 4125–4135, 2011.

[18] A. Kozomara and S. Griffiths-Jones, "mirbase: integrating microrna annotation and deep-sequencing data," *Nucleic Acids Research*, vol. 39, no. suppl 1, pp. D152–D157, 2011.

[19] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[20] A. C. Lorena, A. C. P. L. F. de Carvalho, and J. M. P. Gama, "A review on the combination of binary classifiers in multiclass problems," *Artificial Intelligence Review*, vol. 30, pp. 19–37, 2008.

[21] M. Menor and K. Baek, "Relevance units machine for classification," in *Proceedings of the 4th International Conference on BioMedical Engineering and Informatics*, 2011, pp. 2281–2285.

[22] M. Menor, G. Poisson, and K. Baek, "Probabilistic prediction of protein phosphorylation sites using kernel machines," in *Proceedings of the 27th ACM Symposium on Applied Computing*, 2012, pp. 1393–1398.

[23] I. T. Nabney, "Efficient training of rbf networks for classification," *International Journal of Neural Systems*, vol. 14, no. 3, pp. 1–8, 2004.

[24] I. Psorakis, T. Damoulas, and M. A. Girolani, "Multiclass relevance vector machines: sparsity and accuracy," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1588–1598, 2010.

[25] J. R. Quinlan, P. Compton, K. Horn, and L. Lazarus, "Inductive knowledge acquisition: a case study," in *Proceedings of the Second Australian Conference on Applications of expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1987, pp. 137–156.

[26] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[27] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[28] Y. Zhang, X. Wang, and L. Kang. pirnapredictor: a tool for pirna prediction. [Online]. Available: http://59.79.168.90/piRNA/index.php

[29] ——, "A k-mer scheme to predict pirnas and characterize locust pirnas," *Bioinformatics*, vol. 27, no. 6, pp. 771–776, Mar. 2011.

Chapter 5

Prediction of Mature MicroRNA and Piwi-interacting RNA Without a Genome Reference or Precursors

Abstract

MicroRNA (miRNA) and piwi-interacting RNA (piRNA) are two classes of small RNA that are important in post-transcriptional regulation and are implicated in many important biological processes, such as those involved in diseases. The discovery of novel miRNA and piRNA is an important step in the understanding of these processes. This has been made more possible due to recent advances in high-throughput sequencing technologies that allow biologists to sequence the small RNA content of a tissue or organism. Often small RNA sequencing leads to millions of reads of short length. Finding novel mature miRNA and piRNA is non-trivial and most of the available methods are dependent on the availability of the organism's genome sequence and the quality of its annotation. In this study, we consider the worst-case scenario: no genomic information is available and the identification of mature miRNA and piRNA must be based solely on the short RNA reads. This implies that the longer precursor forms of miRNA and piRNA are not available, as the small RNA sequencing protocol would filter longer RNAs out of the sample. This renders most discovery methods, which rely on the identification of the precursors, inappropriate.

We propose an approach that addresses the weaknesses of existing methods. The method relies primarily on nucleotide composition of the read, thereby avoiding the need of reference genomes of related species and the computationally expensive pairwise folding of the reads required by existing methods. Using Multiclass Relevance Units Machine (McRUM), an empirical Bayesian kernel method for classification, compact models suitable for large-scale analyses are built using databases of known mature miRNA and piRNA. We find that the usage of a L_1-based Gaussian kernel can double the true positive rate compared to the standard L_2-based Gaussian kernel. Our approach can increase the true positive rate by at most 60% compared to piRNApredictor based on the analysis of a hold-out testset. Using experimental data, we show that our approach can detect about an order of magnitude or more known miRNAs than the mature miRNA predictor miRPlex.

5.1 Introduction

Non-coding RNAs (ncRNAs) are functional RNA transcripts that do not translate into proteins. The small ncRNA class consists of ncRNAs of length about 20 to 30 nt. Two important types of small ncRNA are microRNA (miRNA) and Piwi-interacting RNA (piRNA), as they play important roles in molecular biology.

The mature form of miRNA regulates messenger RNAs (mRNAs) in plants and animals as a form of post-transcriptional regulation of genes. This is accomplished by the binding of the mature miRNA to the target mRNA transcript, typically within the 3' UTR [19]. Therefore miRNA play key roles in the regulation of cellular processes and their dysregulation is implicated in many diseases such as cancer [6].

Mature miRNA arises from a complex biogenesis. The canonical pathway to a mature miRNA involves the transcription of a long primary RNA transcript (pri-miRNA). The Drosha enzyme cleaves the pri-miRNA, resulting in a shorter hairpin structure called the precursor miRNA (pre-miRNA). Alternative non-canonical biogenesis pathways are known to produce pre-miRNA without Drosha. For example, mirtrons are miRNAs formed by the introns of a host protein coding gene [4]. In any case, the pre-miRNA is further cleaved into two segments by the enzyme Dicer. The now liberated double-stranded stem of the pre-miRNA is historically called the miRNA/miRNA* duplex. The duplex is later

separated and both strands may possibly lead to functional mature miRNA by recruitment into a RNA-induced silencing complex (RISC) via Argonaute proteins.

On the other hand, the most highly expressed and most diverse small ncRNAs in animals are the piRNAs [17, 25]. Mature piRNAs form RNA-protein complexes with piwi proteins. The piRNA complexes are linked to epigenetic and post-transcriptional silencing of retrotransposons, particularly in the germ line cells, and with tumorigenesis [24]. However recent evidence suggests further roles for piRNA beyond transposon silencing, such as piRNAs that target genes involved with establishing memory in neurons [22]. Many details of piRNA, such as the target silencing mechanisms and aspects of its biogenesis, remain open questions [16]. Therefore piRNAs are important and continuing research will further establish their role in cellular processes.

High-throughput small RNA sequencing technologies have been developed to help in the study of small ncRNAs. For example, small RNA sequencing technologies were used to implicate piRNAs in anti-viral defense in mosquitoes [10]. A variety of methods have been developed to annotate the small RNA reads that require genomic information, such as miRCat [20] for miRNA identification. Nevertheless, annotation of small ncRNA datasets remains a challenge, as for example studies on locust small RNA datasets leave 30-40% sequences unannotated. Furthermore, the requirement of genomic information prevents the majority of methods from being used on non-model organisms where whole-genome sequencing and annotation is a non-trivial hurdle.

For miRNA, there are two existing methods for predicting mature miRNA from small RNA sequencing without a genome reference: MiRMiner [27] and miRPlex [18]. MiRMiner relies on evolutionary conservation and uses the genomic information from other species as a substitute for direct genome reference. Thus MiRMiner requires genome references reasonably close to the target species. On the other hand, miRPlex exploits knowledge of miRNA biogenesis by predicting possible miRNA/miRNA* duplexes via RNA folding algorithms. This avoids the use of any genome references. However, the use of miRPlex on the full small RNA datasets that have millions of reads would require powerful parallel computing infrastructure due to the very large number of possible pairs that need to be evaluated. To reduce the number of duplexes, miRPlex primarily filters the

sequences considered by abundance. This can lead to the omission of miRNAs that were not highly expressed.

For piRNA, to the best of our knowledge, there exists only one method for predicting mature piRNA from small RNA sequencing data, piRNApredictor [28]. piRNApredictor is based purely on nucleotide composition and does not require reference genomes nor sequencing of precursor forms. However, preliminary results in Article 3 (Chapter 4) suggest the predictive performance of piRNApredictor can be significantly improved upon.

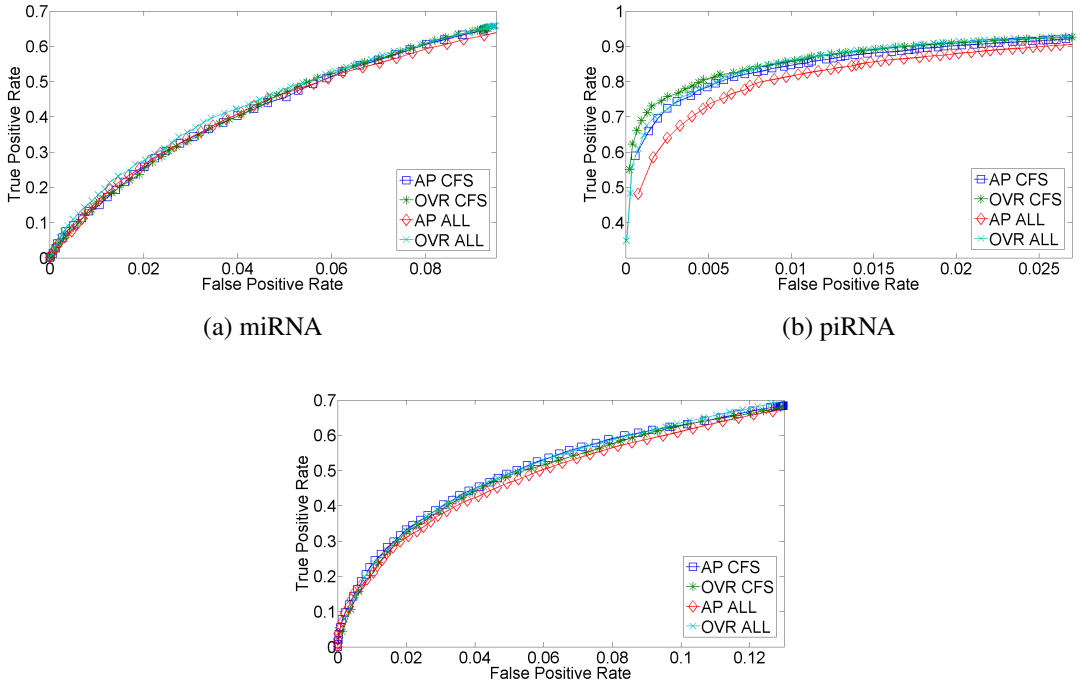
In this study, we propose an approach that addresses the weaknesses of existing methods and our previous approach in Article 3 (Chapter 4). The method will rely primarily on nucleotide composition of the read, thereby avoiding the need of reference genomes and the computationally expensive pairwise folding of the reads required by existing methods. We use Multiclass Relevance Units Machine (McRUM), an empirical Bayesian kernel method for classification, to achieve compact models appropriate for large-scale analyses. The results suggest a significant improvement over our previous efforts in Article 3 (Chapter 4) and to existing methods.

5.2 Results

5.2.1 Feature Selection

Since mature miRNAs and piRNAs lack strong secondary structures, we analyze the nucleotide composition of the ncRNAs. Our previous analysis in Article 3 (Chapter 4) suggests this composition approach suffices for identifying piRNAs but is not optimal in the identification of miRNAs. To improve the performance with regards to miRNAs, we consider additional miRNA-centric features such as the first nucleotide of the sequence, the composition of the seed region, and the A/U composition. We use Correlation-based Feature Selection (CFS) [8] to select a subset of features on which to build classifier models. The core idea of CFS is that the best subsets of features are those that are highly predictive but minimally correlated to each other.

In total we consider 1389 features, including the 1364 unique k-mers, for k = 1 to 5, that represent nucleotide composition. For the CFS McRUMs, CFS is used to select a



(a) miRNA

(b) piRNA

(c) Other ncRNA

Figure 5.1. 3-fold cross-validation ROC curve for CFS and all features (ALL) McRUMs using L_1 Gaussian kernel with all-pairs (AP) or one-vs-rest (OVR) decompositions. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01.

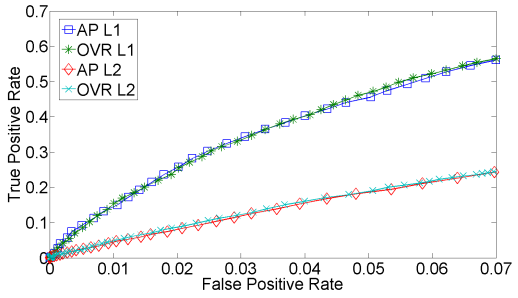
subset of features to build a McRUM for each fold of cross-validation. CFS selects about 142-161 features per fold, which is about a ten fold decrease from using all features. When applied to the entire training dataset, CFS selects 154 features. In all cases, among those selected by CFS are the four binary features representing A, C, G, and U that mark the identity of the first nucleotide. This is not surprising as both miRNA and piRNA are biased toward starting with a U base. Also selected is the AU score [7], as both piRNA and miRNA tend to have higher scores. This too is not surprising for miRNA because miRNA targets are known to have high AU content [7]. Lastly, the frequency of the 2-mer CG in the potential seed region is selected. Interestingly, this feature distinguishes piRNA rather than miRNA, as piRNAs are more biased toward lower scores than other ncRNAs.

Figure 5.1 illustrates the predictive performance of McRUMs with L_1 Gaussian kernel using all features or CFS-selected features. Two popular decompositions of a multi-

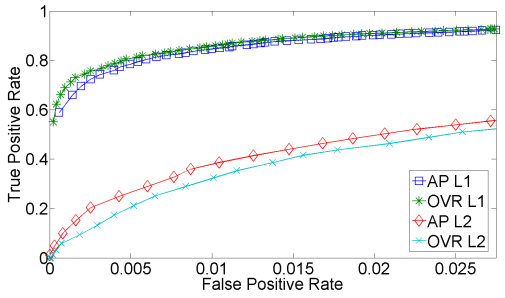
class problem to a set of binary class problems are considered: all-pairs (AP) and one-vs-rest (OVR). In the AP decomposition, a binary classifier is created for every pair of classes: 1) miRNA vs. piRNA, 2) miRNA vs. other ncRNA, and 3) piRNA vs. other ncRNA. On the other hand the OVR decomposition creates a binary classifier for each class against all others: 1) miRNA vs. piRNA & other ncRNA, 2) piRNA vs. miRNA & other ncRNA, and 3) other ncRNA vs. miRNA & piRNA. The performance is measured using 3-fold cross-validation. Figure 5.1b shows some advantage of using the OVR decomposition over AP. However, there is little difference between the models that use all features and CFS-selected features. This is surprising as the order of magnitude reduction of features and thus dimensionality was expected to improve predictive performance due to the relative contrast theory of Aggarwal [1] and since CFS should not be eliminating vital features. Nevertheless, the CFS analysis did provide insight into which features are actually necessary for prediction.

5.2.2 Kernel Selection

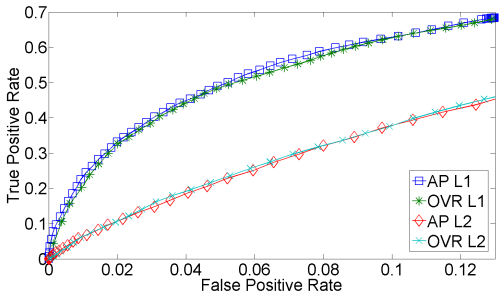
We compare the use of L_1 and the standard L_2 Gaussian kernels with 3-fold cross-validation using CFS-selected features. Figure 5.2 shows the significant increase of performance from using L_1 rather than the standard L_2 Gaussian kernel. For example, there is an increase to about 0.9 true positive rate (TPR) using L_1 from about 0.4 using L_2 at 0.01 false positive rate (FPR) for piRNA prediction (Figure 5.2b). Aggarwal et al. [1] studied the behavior of distance metrics in high dimensional clustering that explains the results we observe. They define a concept called the *relative contrast* that measures how well you can discriminate between the nearest and furthest neighbor when using a particular distance metric. If relative contrast is too low, all points appear to be equally close by, which would be disastrous in classification where the locality of datapoints is crucial. They go on to show that relative contrast using L_1 distance decays at a slower rate than L_2 distance as dimension increases and therefore L_1 distance remains a viable metric at larger dimensions than L_2 distance. These theoretical results agree with our observations that the standard L_2 Gaussian kernel evaluations in our problem are all near unity, indicating low relative contrast as all points in the dataset appear to be equally distant from each



(a) miRNA



(b) piRNA



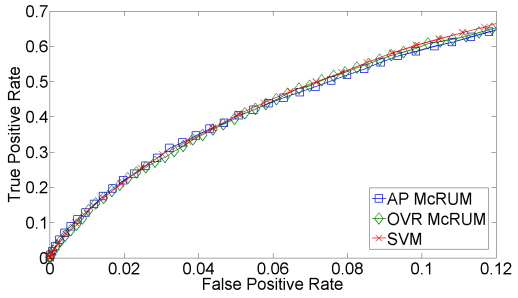
(c) Other ncRNA

Figure 5.2. 3-fold cross-validation ROC curves for McRUMs using L_1 and L_2 Gaussian kernels with all-pairs (AP) or one-vs-rest (OVR) decompositions. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01.

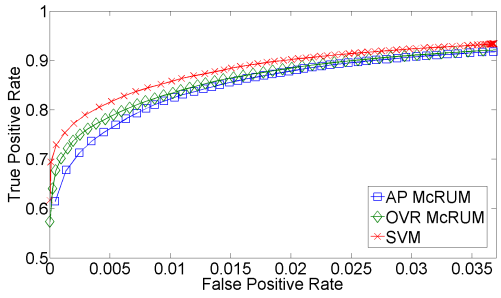
other. The increased relative contrast offered by using L_1 distance allows McRUM to more easily distinguish the datapoints belonging to different classes and thus improves predictive performance.

5.2.3 Comparison with SVM and piRNApredictor using Hold-out Test Set

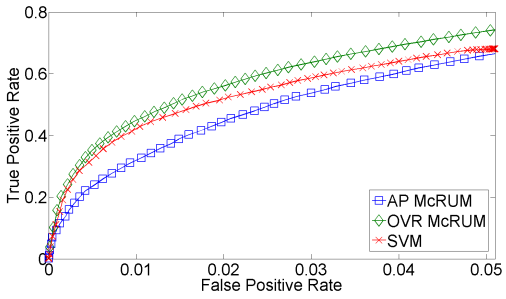
We use a hold-out test set to compare the predictive performance of McRUMs to a SVM-based solution using the L_1 Gaussian kernel and the CFS-selected features and the existing piRNApredictor software [28]. We implement the L_1 Gaussian kernel in the machine learning library Weka [9] and use Weka's implementation of sequential minimal optimization to train a SVM with logistic models. Weka's implementation of multiclass



(a) miRNA



(b) piRNA



(c) Other ncRNA

Figure 5.3. Test set ROC curves for McRUMs and SVM using L_1 Gaussian kernel. McRUMs use all-pairs (AP) or one-vs-rest (OVR) decompositions. The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01.

SVM uses the AP decomposition to binary classification problems. The predictive performance results are given in the ROC curves in Figure 5.3, which shows no clear superior method. Performance is nearly identical for miRNAs (Figure 5.3a), the SVM has a small advantage for piRNAs (Figure 5.3b), and the OVR McRUM has the advantage for other ncRNAs. The primary advantage of using McRUM models over the SVM model is the smaller models generated by McRUM, as illustrated in Figure 5.4. The SVM and both McRUM models are composed of three binary classification models. On average, a binary classification model of the SVM uses about 6 times more terms (support vectors) in its linear model than McRUMs. Therefore, McRUM models are able to produce predictions at 6 times the rate of the SVM, leading to faster RNA-seq analysis run-times.

The predictive performance of McRUMs for the piRNA case compared to the software piRNApredictor [28] is given in Figure 5.5. piRNApredictor uses Fisher's Linear

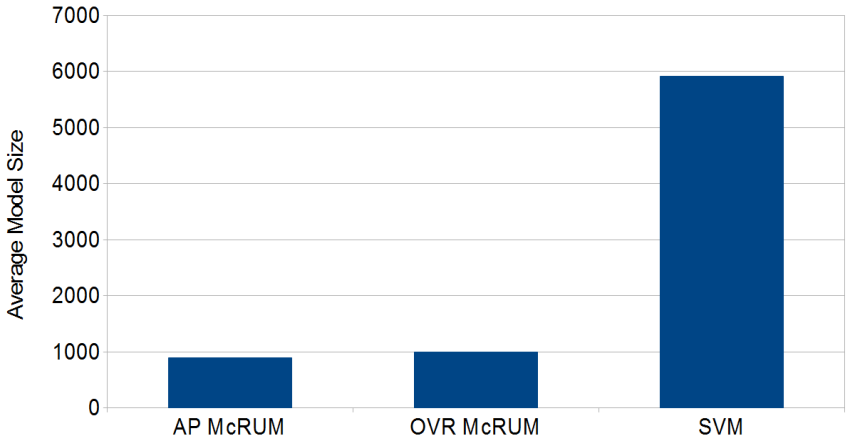


Figure 5.4. The average size (number of relevance units or support vectors) of the binary classification models for all-pairs (AP) & one-vs-rest (OVR) McRUMs and SVM.

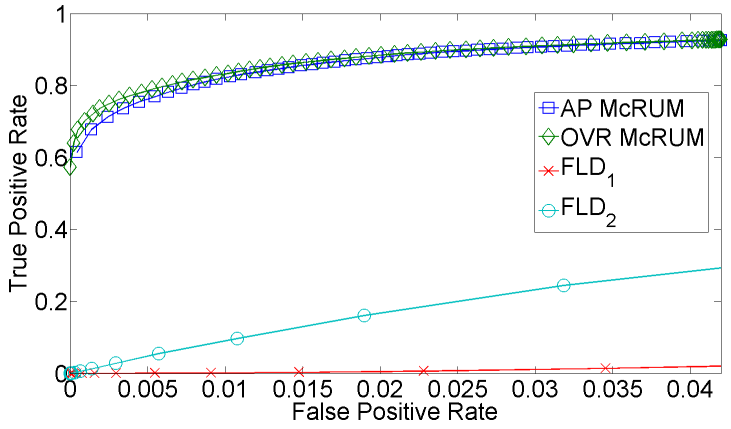


Figure 5.5. Test set ROC curves for all-pairs (AP) & one-vs-rest (OVR) McRUMs and piRNApredictors - the original piRNApredictor (FLD_1) and a retrained piRNApredictor (FLD_2). The ROC curves are generated from observed FPR and TPR under varying posterior probability thresholds from 0.30 to 0.99 in increments of 0.01.

Discriminant (FLD) classifier and makes predictions based on k-mer composition. Two variations of piRNApredictor are examined: the original provided by the authors and one built on our training dataset. Both McRUMs significantly outperform the piRNApredictors. One reason for the low performance of piRNApredictor is its use of a FLD's linear model of the decision boundary between piRNA and other ncRNAs. We also observed poor performance using a linear SVM in our preliminary analysis in Article 3 (Chapter 4). Also

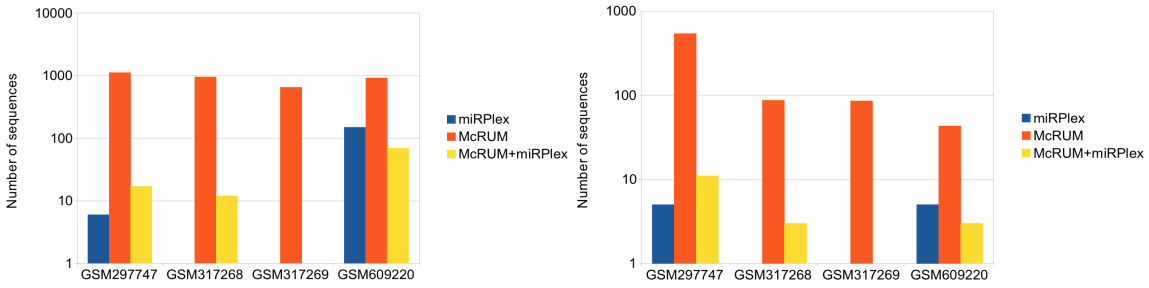
noted in our preliminary analysis was the small representation of ncRNA sequences under 25 nt long in piRNApredictor's training dataset. Only 4.67% of piRNApredictor's training dataset were ncRNAs under 25 nt long. This weak representation of short ncRNA may be the reason why the piRNApredictor trained on our dataset outperforms the original.

5.2.4 Comparison with miRPlex using Experimental Datasets

We compare miRPlex [18] and our McRUMs using experimental RNA-seq datasets since miRPlex requires information on sequence abundance and pairing sequences for the miRNA duplex prediction. These additional information are not provided in our training or hold-out test sets. Therefore, we consider the following four datasets: *Caenorhabditis elegans* (GSM297747), *Drosophila melanogaster* (GSM609220), and the gregarious (GSM317268) and solitary (GSM317269) phases of *Locusta migratoria*. While the ground truth is not known for experimental datasets, we are able to gain some insight on how miRPlex and McRUM perform individually and in conjunction with each other.

The primary disadvantage of miRPlex is its sheer computational complexity, as it requires invoking a RNA folding routine for every pair of sequences in RNA-seq dataset that may contain millions of sequences. To make miRPlex practical, the dataset must be filtered down to a smaller subset in some way. miRPlex does this by allowing the user to specify the minimum abundance of the sequences to be considered. This removes lowly expressed sequences from the dataset, which can be an issue should the miRNA of interest occur at low expression levels. We also consider the use of McRUM as a filter for miRPlex, as an alternative to abundance filtering.

Due to computational constraints, we use a minimum abundance of 25 reads per million (RPM) for GSM297747 and GSM609220 datasets and 40 RPM for GSM317268 and GSM317269 that have a larger number of unique sequences with high expression. For the McRUM results, we use a minimum posterior probability of 0.9 using an OVR-based McRUM model and minimum abundance of 2 RPM to reduce noise. We also use the resulting McRUM outputs as the input to miRPlex with no abundance filter. This will allow miRPlex to pick out the McRUM results that have reasonable duplex pairings. The results are summarized in Figure 5.6a.



(a) Predictions

(b) Matches to known miRNA

Figure 5.6. (a) Number of miRNA predictions for McRUM and miRPlex. (b) Number of miRNA predictions matching a known mature miRNA.

Due to the high number of unique sequences in the locust datasets, we were forced to use a high minimum abundance of 40 RPM. This is too stringent for miRPlex and it is unable to detect any miRNAs. Using McRUM as the filter instead results in the detection of more miRNAs for two of the four datasets. MiRPlex still detects no miRNAs for GSM317269, while it detects less than half the amount of the abundance-filtered miRPlex for GSM609220. On the other hand, McRUM by itself detects about 1-2 orders of magnitude more miRNAs than the standard miRPlex.

To get a sense of the performance of these predictions, we look at how many of these predicted miRNAs closely match to known miRNAs using a BLAST search against miRBase's collection of mature miRNAs. We use NCBI's blastn-short that is optimized for sequences less than 50 nucleotides [12]. A challenge with using BLAST on short sequences is that perfect or near perfect matches result in large e-values relative to analyses with long sequences. A perfect match of length 22 nt, roughly the length of a mature miRNA, only gets an e-value of 2e-7. While only having a single mismatch the e-value jumps up to 5e-5 and two mismatches gives an e-value of 2.8. This is troublesome because of the existence of isomiRs, which are variants of the canonical sequences given in miRBase. These isomiR variants include nucleotide substitutions, additions, and trimming of the 5' and/or 3' end of the miRNA. We choose an e-value threshold of 5e-5 to allow for a single mismatch in a 22 nt alignment in order to get some sensitivity to isomiRs while still not preventing too many coincidental matches. While this analysis still remains rough and does not account for novel miRNAs, it does give a sense of how many of these predictions are reasonably close

to known miRNAs. The results are summarized in Figure 5.6b and illustrates similar trends to that of Figure 5.6a. This shows that McRUM by itself can detect known miRNAs better than miRPlex using abundance or McRUM filtering by about an order of magnitude or more. This possibly comes with a number of false positives, but these can be reduced using a higher minimum posterior probability or by usage of miRPlex on McRUM's predictions.

One additional potential reason for miRPlex under-performing is the aforementioned existence of isomiRs. MiRPlex cannot account for the miRNAs being further modified after splitting apart from the duplex phase. The isomiR variants of the miRNAs may no longer be able to form a thermodynamically favorable duplex and thus miRPlex would not be able to detect these miRNAs. This is less of an issue with McRUM models since the models do not consider the duplex and instead rely upon other miRNA features such as nucleotide biases.

5.3 Methods

5.3.1 Classification Methods

We use and compare the following two kernel-based classification methods: multiclass relevance units machine (McRUM) from Article 3 (Chapter 4) and the support vector machine (SVM) [26] using the Gaussian kernel. The SVM is a popular method due to its high accuracy via margin-maximizing framework. Probabilistic outputs will be generated from the SVM models by training a logistic regression model on the trained SVM's outputs [21], as implemented in Weka [9]. Cross-validation is used to select the SVM parameters, the soft margin parameter C and the Gaussian kernel width.

McRUM is a new method that provides probabilistic outputs, high accuracy through its empirical Bayes foundations that mitigate overfitting, and more parsimonious models than the SVM. McRUM decomposes an n-class classification problem into a set of binary classification problems that can be solved independently. In this study, we consider the popular and effective all-pairs (AP) and one-vs-rest (OVR) decompositions. A binary classifier is generated for every pair of classes in the AP decomposition, resulting in $n(n-1)/2$ binary classifiers. For each class in the OVR decomposition, the selected class is compared

to the aggregation of all remaining classes. Therefore the OVR decomposition consists of n binary classifiers.

Each binary classification problem is solved using the Classification Relevance Units Machine (CRUM) described in Article 2 (Chapter 3) where cluster centers are used as the centers for the Gaussian kernels. For this study, the same number of clusters is used for all CRUMs and is chosen using cross-validation. An empirical Bayesian algorithm determines the weight of each Gaussian kernel. A linear-time algorithm aggregates the set of binary predictions from the trained CRUM models to produce the prediction for the original n-class problem.

5.3.2 Gaussian Kernels

In our previous analysis in Article 3 (Chapter 4), we used the standard Gaussian radial basis function kernel for both McRUM and SVM,

$$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2)$$

where γ controls the growth of the kernel and the squared Euclidean distance of the two points $\mathbf{x}$ and $\mathbf{x}'$ is considered. The use of Euclidean (L_2) distance as its metric is not necessarily the optimal choice depending on the characteristics of the dataset, such as its dimensionality and sparseness. This is very well known in the clustering literature [1], but these results are not prevalently used in the classification literature. This is perhaps due to the technical difficulty that using a different metric in the Gaussian kernel may lead to an invalid kernel function and the fact that the popular SVM packages, LIBSVM [5], SVM^{light} [11], and Weka [9], do not provide the built-in option to use different metrics in their Gaussian kernel implementations. However, it is known that the Gaussian kernel using Manhattan (L_1) distance leads to a valid kernel [23]. We investigate the use the L_1 distance-based Gaussian kernel over the standard L_2 distance-based version with the L_1 Gaussian kernel in a d-dimensional real vector space defined as,

$$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma \|\mathbf{x} - \mathbf{x}'\|_1)$$

where

$$\|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{i=1}^{d} |x_i - x'_i|.$$

5.3.3 Datasets

The small ncRNA dataset is gathered from miRBase's collection of miRNA [14] and NONCODE collection of piRNA [3]. To represent other small RNAs that may be present in small RNA sequencing projects, we use *Mus musculus* GSM314553 small RNA dataset from NCBI GEO database [2]. The mouse dataset is from a Dicer knockout mouse, which should not contain any mature miRNA due to Dicer's important role in miRNA biogenesis. Any known mature miRNA and piRNA in this dataset will be filtered out. This should provide a more realistic representation of other small RNAs than our previous effort that used random short fragments of other ncRNAs from the NONCODE database. Each of these three sets are individually reduced using CD-HIT [15] to remove sequences with 80% or higher identity. This helps reduce the evolutionary correlations among the data and improves the generalization of McRUM and SVM models that assumes an independent sample.

After redundancy reduction, the dataset consists of 13,891 miRNA, 81,147 piRNA, and 190,761 other ncRNA sequences. We randomly set aside 20% of the miRNA sequences for the hold-out test set and the remaining 80% for the training set. We randomly select equal number of piRNA and other ncRNA sequences for training to keep the training sets balanced. All remaining piRNA and other ncRNA sequences are used in the hold-out test set.

Dataset	Species	Library size	# unique seq.	# filtered seq.
GSM297747	*Caenorhabditis elegans*	5 million	371,937	1353
GSM317268	*Locusta migratoria*	1 million	228,526	2091
GSM317269	*Locusta migratoria*	2 million	429,041	1668
GSM609220	*Drosophila melanogaster*	14 million	78,236	1349

Table 5.1. Summary of experimental datasets. Sequences are filtered based on minimum RPM. We use a minimum of 25 RPM for GSM297747 and GSM609220, and 40 RPM for GSM317268 and GSM317269.

For comparison with miRPlex, four experimental RNA-seq datasets are used. Information about these datasets is summarized in Table 5.1.

5.3.4 Features

Since mature miRNAs and piRNAs lack strong secondary structures, we will represent each ncRNA using k-mers, for k = 1, 2,..., 5. For each value of k, the number of occurrences of each type of k-mer is computed and normalized. If we view each ncRNA sequence as a text document, this k-mer approach is analogous to the "bag of words" approach in document classification. Our previous analysis in Article 3 (Chapter 4) suggests this k-mer approach suffices for identifying piRNAs but is poor in identifying miRNAs.

To improve the performance with regards to miRNAs, we consider additional features. It is known that miRNAs have a strong bias toward the U nucleotide in the first position of their sequences [18]. We therefore propose to use four binary features for the four possible RNA nucleotides, A, C, G, and U, to mark the identity of the first nucleotide of the read. It is also known that the target mRNAs of a miRNA need to be physically accessible, particularly to the first eight positions of the miRNA that is called the seed region [13]. Accessibility is correlated with the nucleotide composition of the mRNA target region and thus the miRNA's nucleotide composition. The composition of the entire miRNA is already captured by the k-mers features, but to emphasize the importance of the seed region, we propose the addition of separate k-mer features computed over the seed region only. Lastly, we also consider the AU Score proposed in [7] as another feature related to accessibility. The AU Score is a measure of AU content weighed by the distance of the A/U occurrence to the seed region.

Using k = 1 and 2 only for the seed region k-mer composition features, we have a grand total of 1389 features. Since this is high dimensional, we may be able to achieve improved results through feature selection. We use CFS [8] to select a subset of features to build classifier models on. The core idea of CFS is that the best subset of features are those that are highly correlated to the class and minimally correlated to each other. In other words, CFS seeks to find features that are maximally relevant and minimally redundant. To prevent overfitting through feature selection, we use CFS in the cross-validation context.

5.4 Conclusion

We proposed an approach to the prediction of mature miRNA and piRNA that relies primarily on nucleotide composition of the read. This approach avoids the need of reference genomes of related species and the computationally expensive pairwise folding of the reads required by existing methods. Using the McRUM classification method, compact models suitable for large-scale analyses are built using databases of known mature miRNA and piRNA. Cross-validation results show that the usage of a L_1-based Gaussian kernel can double the true positive rate compared to the standard L_2-based Gaussian kernel. Analysis of the the hold-out test set shows our approach can increase the true positive rate for piRNA by at most 60% compared to piRNApredictor. Finally, we show that our approach can detect at about an order of magnitude or more known miRNAs than miRPlex using experimental datasets.

Bibliography

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," *Lecture notes in computer science*, pp. 420–434, 2001.

[2] T. Barrett, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, M. Holko *et al.*, "Ncbi geo: archive for functional genomics data setsupdate," *Nucleic acids research*, vol. 41, no. D1, pp. D991–D995, 2013.

[3] D. Bu, K. Yu, S. Sun, C. Xie, G. Skogerb, R. Miao, H. Xiao, Q. Liao, H. Luo, G. Zhao, H. Zhao, Z. Liu, C. Liu, R. Chen, and Y. Zhao, "Noncode v3.0: integrative annotation of long noncoding rnas," *Nucleic Acids Research*, 2011.

[4] L. Castellano and J. Stebbing, "Deep sequencing of small rnas identifies canonical and non-canonical mirna and endogenous sirnas in mammalian somatic tissues," *Nucleic acids research*, vol. 41, no. 5, pp. 3339–3351, 2013.

[5] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[6] G. Di Leva and C. M. Croce, "mirna profiling of cancer," *Current opinion in genetics & development*, 2013.

[7] A. Grimson, K. K.-H. Farh, W. K. Johnston, P. Garrett-Engele, L. P. Lim, and D. P. Bartel, "Microrna targeting specificity in mammals: determinants beyond seed pairing," *Molecular cell*, vol. 27, no. 1, pp. 91–105, 2007.

[8] M. Hall, "Feature subset selection: a correlation based filter approach," in *Proc Fourth International Conference on Neural Information Processing and Intelligent Information Systems, Dunedin, New Zealand, 1997*, 1997.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[10] A. M. Hess, A. N. Prasad, A. Ptitsyn, G. D. Ebel, K. E. Olson, C. Barbacioru, C. Monighetti, and C. L. Campbell, "Small rna profiling of dengue virus-mosquito interactions implicates the piwi rna pathway in anti-viral defense," *BMC microbiology*, vol. 11, no. 1, p. 45, 2011.

[11] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in kernel methods*. MIT Press, 1999, pp. 169–184.

[12] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and T. L. Madden, "Ncbi blast: a better web interface," *Nucleic acids research*, vol. 36, no. suppl 2, pp. W5–W9, 2008.

[13] M. Kertesz, N. Iovino, U. Unnerstall, U. Gaul, and E. Segal, "The role of site accessibility in microrna target recognition," *Nature genetics*, vol. 39, no. 10, pp. 1278–1284, 2007.

[14] A. Kozomara and S. Griffiths-Jones, "mirbase: integrating microrna annotation and deep-sequencing data," *Nucleic Acids Research*, vol. 39, no. suppl 1, pp. D152–D157, 2011.

[15] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[16] M. Luteijn and R. Ketting, "Piwi-interacting rnas: from generation to transgenerational epigenetics." *Nature reviews. Genetics*, vol. 14, no. 8, pp. 523–534, 2013.

[17] S. R. Mani and C. E. Juliano, "Untangling the web: The diverse functions of the piwi/pirna pathway," *Molecular reproduction and development*, vol. 80, no. 8, pp. 632–664, 2013.

[18] D. Mapleson, S. Moxon, T. Dalmay, and V. Moulton, "Mirplex: A tool for identifying mirnas in high-throughput srna datasets without a genome," *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, vol. 320, no. 1, pp. 47–56, 2013.

[19] H. Min and S. Yoon, "Got target?: computational methods for microrna target prediction and their extension," *Experimental & molecular medicine*, vol. 42, no. 4, pp. 233–244, 2010.

[20] S. Moxon, F. Schwach, T. Dalmay, D. MacLean, D. J. Studholme, and V. Moulton, "A toolkit for analysing large-scale plant small rna datasets," *Bioinformatics*, vol. 24, no. 19, pp. 2252–2253, 2008.

[21] J. C. Platt, "Probabilities for sv machines," in *Advances in Large Margin Classifiers*, A. J. Smola, P. L. Bartlett, B. Scholkopf *et al.*, Eds. MIT Press, 2000, pp. 61–73.

[22] P. Rajasethupathy, I. Antonov, R. Sheridan, S. Frey, C. Sander, T. Tuschl, and E. R. Kandel, "A role for neuronal pirnas in the epigenetic control of memory-related synaptic plasticity," *Cell*, vol. 149, no. 3, pp. 693–707, 2012.

[23] B. Ribeiro, "On the evaluation of minkovsky kernel for svms." *Neural Parallel & Scientific Comp.*, vol. 13, no. 1, pp. 77–90, 2005.

[24] K. Sato and M. C. Siomi, "Piwi-interacting rnas: biological functions and biogenesis," *Essays in Biochemistry*, vol. 54, no. 1, pp. 39–52, 2013.

[25] A. G. Seto, R. E. Kingston, and N. C. Lau, "The coming of age for piwi proteins," *Molecular cell*, vol. 26, no. 5, pp. 603–609, 2007.

[26] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[27] B. M. Wheeler, A. M. Heimberg, V. N. Moy, E. A. Sperling, T. W. Holstein, S. Heber, and K. J. Peterson, "The deep evolution of metazoan micrornas," *Evolution & development*, vol. 11, no. 1, pp. 50–68, 2009.

[28] Y. Zhang, X. Wang, and L. Kang, "A k-mer scheme to predict pirnas and characterize locust pirnas," *Bioinformatics*, vol. 27, no. 6, pp. 771–776, Mar. 2011.

Chapter 6

Conclusion

In this dissertation, we developed novel probabilistic classification methods and demonstrated their practical applicability by solving major bioinformatics problems at the transcriptome and proteome levels where the resulting tools are expected to help biologists further elucidate the important information contained in their data.

The proposed non-linear binary classification method, CRUM, achieves high generalization through the utilization of the theory of kernel and empirical Bayesian methods. The practical applicability of CRUM is demonstrated by its application to an important problem in molecular biology, the prediction of protein phosphorylation sites. Predictions of these sites help explain the mechanisms that control many biochemical processes.

Then we developed an extension of CRUM from binary to multiclass problems with McRUM. McRUM decomposes a multiclass problem into a set of binary problems by using the error correcting output codes framework. CRUM can then be used to solve the individual binary problems independently. A linear-time algorithm to aggregate the results of the individual CRUMs into the final probabilistic multiclass prediction is devised to allow for predictions in large scale applications. The practical applicability of McRUM is demonstrated through a solution to the identification of mature miRNA and piRNA, which are implicated in post-transcriptional regulation, in small RNA sequencing datasets without the use of genomic references. This provides a tool to help biologists discover novel miRNA and piRNA and further understand the molecular processes of the organisms they study, including non-model organisms whose genomic information may be lacking.

In this new era of big data biological research, the abundance of data has presented many challenges in their analysis and has led to a situation some call "data rich, information poor." It is expected that the research developed in this dissertation will help alleviate this problem by providing methods to extract important information out of data in order to elucidate details of biological mechanisms that are helpful for basic science and the development of new diagnostic and therapeutic tools.

6.1 Future Work

While this dissertation focused on bioinformatics applications, the diverse sample of benchmarking datasets processed using CRUM and McRUM shows the general applicability of these methods to classification problems of many domains. A future direction is more in-depth studies into areas such as computer vision, natural language processing, and speech recognition using CRUM or McRUM. Sentiment analysis is an example problem from natural language processing that involves determining the attitude of the writer with respect to the topic being discussed in the text. This can involve the classification of the polarity of the message as expressing a positive, negative or neutral opinion. Sentiment analyzers work best if trained specifically for the the domain of application due to the differences in lexicon used in different topic areas (e.g. finance or movie reviews) or different media (e.g. editorial articles or short social media messages). The real-time stream of text generated by social media is an interesting source of data, as social media platforms have become a central site where the public can express their opinions. This can allow for real-time sentiment analysis to track the public's opinion regarding a product, company, or political candidate. This would provide the ability to analyze the relationship between the expressed public sentiment and emerging events, and possibly allow for intervention to sway public sentiment to a desired state. The development of online learning algorithms for CRUM and McRUM to efficiently learn with streams of data would facilitate such real-time learning applications.