THESIS

LARGE MARGIN KERNEL METHODS FOR CALMODULIN BINDING PREDICTION

Submitted by

Michael Hamilton

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2010

COLORADO STATE UNIVERSITY

April 2, 2010

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY MICHAEL HAMILTON ENTITLED LARGE MARGIN KERNEL METHODS FOR CALMODULIN BINDING PREDICTION BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Charles Anderson

Hariharan Iyer

Advisor: Asa Ben-Hur

Department Head: Darrell Whitley

ABSTRACT OF THESIS

LARGE MARGIN KERNEL METHODS FOR CALMODULIN BINDING PREDICTION

Protein-protein interactions are involved in nearly all molecular processes of organisms. However direct laboratory techniques for identifying binding partners remain expensive and difficult at the proteome scale. In this work, kernel methods for predicting calmodulin binding partners and calmodulin binding sites are presented. Furthermore, we compare binary and structural support vector machines with multiple kernels defined over protein sequences.

Michael Hamilton
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
Spring 2010

ACKNOWLEDGMENTS

I would like to thank my advisor, Asa Ben-Hur, for his continued support and insight during this work. Moreover, I thank A.S.N. Reddy for bringing this challenging problem to our attention and for sharing insights into the biology. Special thanks to Chuck Anderson, Wim Bohm, and the rest of the Computer Science faculty and fellow graduate students at Colorado State University. And most importantly, I would like to thank my son, Jacob, for his constant patience and motivation throughout my graduate work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Protein-Protein Interactions

Protein-protein interactions (PPIs) are central to nearly all cellular processes. While sequencing technologies have advanced, methods for identifying protein function remain expensive and arduous. Identifying interaction partners is key to annotating proteins, as proteins which interact must be co-located within cells and are likely to be involved in similar biological processes [1].

Recent high-throughput experiments that identify interaction partners have led to advances in reconstructing the interactomes of several model organisms [2]. However, information on binding site location remains scarce. The need to address this gap has led to the development of a variety of methods for identifying interaction sites (see [3] for a review). The majority of binding site prediction methods use structural information on surface residues to predict the likelihood that they belong to a protein-protein interface. Yet, due to the lack of available structural models for most proteins, the applicability of these methods is limited. Therefore it is of interest to develop prediction methods that use features inferred from sequence alone, and the work of Ofran and Rost [4] suggests that it is indeed possible.

The work mentioned above focused on predicting binding sites without reference to the targets of those sites. A more directed approach is to predict binding sites of specific

proteins. However, the paucity of binding site data for individual targets has hampered this effort.

1.2 Calmodulin

Calmodulin is one case where such an approach is currently possible. Because calmodulin is highly conserved across species [5], it is possible to leverage information from different organisms.

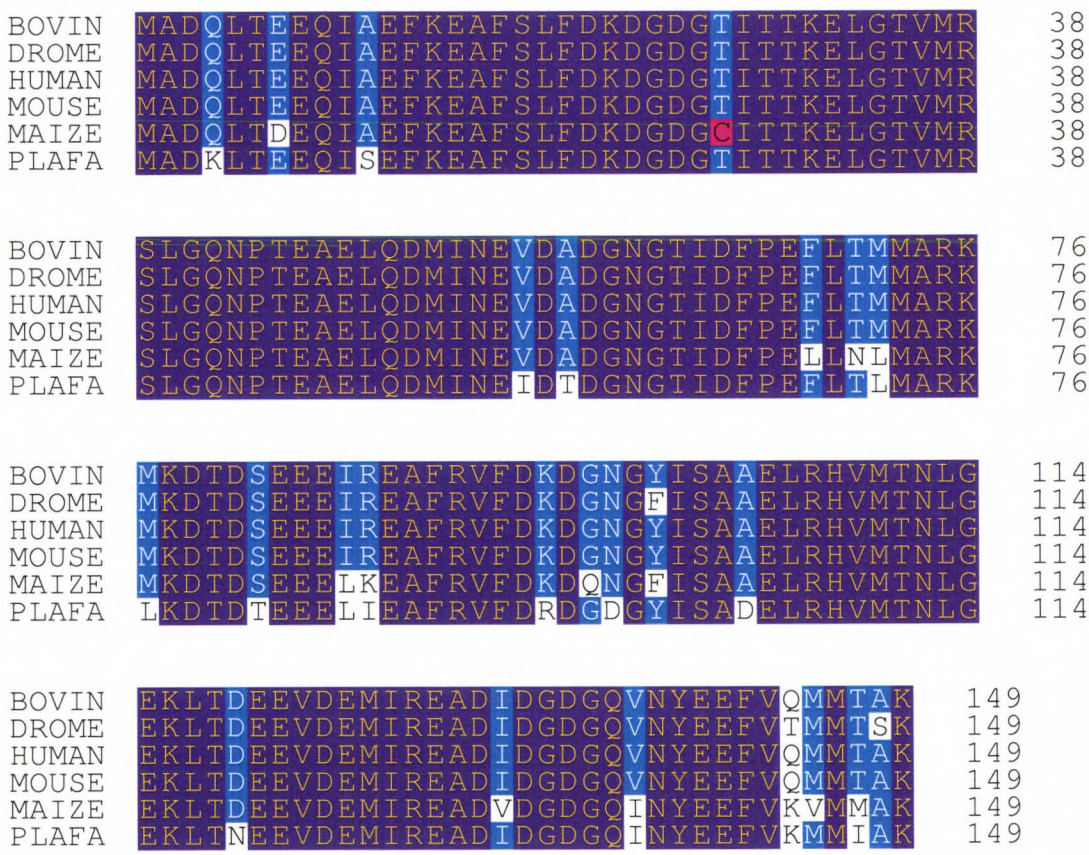


Figure 1.1: Alignment of five eukaryotic calmodulin sequences.

For instance, the multiple sequence alignment of calmodulin sequences, performed using `CLUSTAL W` [6], from cow (`BOVIN`), fruit fly (`DROME`), human, mouse, corn (`MAIZE`), and the malaria parasite (`PLAFA`) is shown in Figure 1.1 and pair-wise sequence similarity and identity is presented in Table 1.1. This high degree of sequence

conservation is rare, given the diversity across these species. And hence, it is likely that interaction partners of calmodulin share a common evolutionary pattern within their binding sites.

Table 1.1: Sequence similarity and identity among 5 eukaryotic calmodulin sequences. Sequence identity is the percentage of positions that have complete conservation. Sequence similarity is computed using substitution rates of amino acids to calculate scores for each position.

	BOVIN	DROME	HUMAN	MOUSE	MAIZE	PLAFA
BOVIN	—	99.3	100.0	100.0	95.9	95.9
DROME	97.9	—	99.3	99.3	95.9	95.9
HUMAN	100.0	97.9	—	100.0	95.9	95.9
MOUSE	100.0	97.9	100.0	—	95.9	95.9
MAIZE	90.6	90.6	90.6	90.6	—	94.6
PLAFA	89.2	87.9	89.2	89.2	86.5	—

% identity

% similarity

Calmodulin, a name derived from calcium modulation, plays a critical role in many biological processes [7]. It is one of a handful of calcium binding proteins that serves a central role in signal transduction: the process of broadcasting an initial signal, in a variety of forms, throughout a cell [8]. Also, calmodulin interacts with a wide variety of proteins in both the presence and absence of calcium [9].

The ability of calmodulin to interact with proteins sharing little functional and sequence similarity has been attributed to its dynamical structure [12]. Figure 1.2 shows three different structural forms of calmodulin. On the left is calmodulin in its native form without calcium [13] and on the right, the structure of calmodulin upon binding four calcium ions [14]. The image in the middle shows calmodulin, colored green, in complex with a myosin light chain kinase fragment, shown in light blue [15]. Most calmodulin binding sites share two key properties: they are contiguous in sequence and

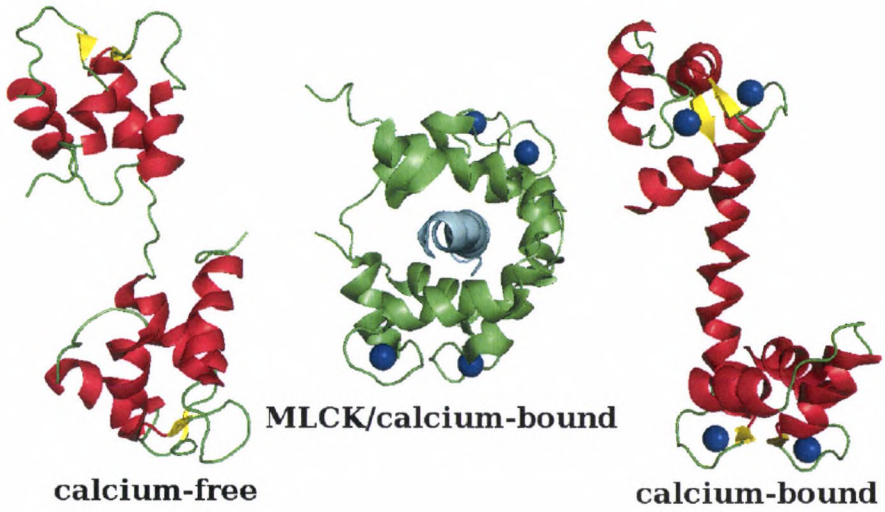


Figure 1.2: Structural dynamics of calmodulin. (PDB IDs: 1DMO, 2BBM, 3CLN [10], image rendered using `PyMOL` [11])

tend to form alpha-helices [16]–factors that will prove useful in predicting interaction sites.

1.3 Calmodulin Binding Prediction

The problem of predicting calmodulin binding sites is an instance of the label-sequence learning problem [17]. Given a protein sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ we want to associate with it a sequence of labels $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ where y_i, indicates whether position i is part of a binding site. A standard approach for solving this problem is the sliding window method [17] which reduces the problem to a two-class classification problem by considering the problem at the amino acid level. Each position in the sequence is predicted to either belong to a binding site or not, independently of its neighbors. At this level one can apply standard classifiers that use features computed on the basis of a fixed-length window of amino acids. This is the approach taken in Radivojac et al. [18], and is the strategy of our binary SVM method.

The second approach presented in this work for calmodulin binding site prediction utilizes the novel methodology of structured outputs learning [19]. Whereas traditional classification methods predict a single discrete class label for a given input, structured outputs methods are able to predict more complex objects and have been applied to a variety of problems such as hierarchical classification [20, 21], natural language parsing [21], and disulfide bridge prediction [22].

It is also worth mentioning methods that predict binding at the level of conserved domains or motifs [23]: methods which aim to explain an observed network of interactions in terms of interactions between these features. Such methods, while using global information on the interaction network, ignore properties of the individual proteins, and only provide a coarse-grained prediction of the binding site location. The proposed methods in this work are much more specific–by training a classifier on examples of interaction partners for a specific protein (calmodulin), it is possible to capture properties of the binding sites.

The rest of this thesis is organized in the following format. Chapter 2 provides the formulations of the binary and structural support vector machine. Chapter 3 details the experiments on calmodulin binding prediction used in this work and illustrates exploratory data analysis over a variety of amino acid features. Results and performance comparisons of the two approaches are explained in Chapter 4 along with a discussion of future work and a summary of this thesis.

Chapter 2

Background

This chapter describes the classifiers used in this work. Section 2.1 reviews the concepts of linear classifiers and then details the formulation of the binary support vector machine. Section 2.2 provides insight into the learning framework for structured outputs and reviews the concept of the structural support vector machine. Finally, Section 2.3 defines the notion of kernel functions and their role within large-margin classifiers.

2.1 Support Vector Machines

A linear discriminant classifier aims to find a separating hyperplane between two classes. Given data $\{\mathbf{x}_i, y_i\}_{i=1}^N, \mathbf{x}_i \in \Re^p, y_i \in \{1, -1\}$, the objective is to learn a function f, where

$$f_{(\mathbf{w},b)}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2.1}$$

such that the sign of $f_{(\mathbf{w},b)}(\mathbf{x}_i)$ is positive for $y_i = 1$ and negative for $y_i = -1$. The selection of labels, $y_i \in \{-1, 1\}$, allows this to be expressed as

$$\left\{y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 0\right\}_{i=1}^N. \tag{2.2}$$

Figure 2.1 shows a 2-dimensional toy dataset and 3 separating hyperplanes. It is unlikely that the discriminant functions defined by $(\mathbf{w}_1, b_1)$ and $(\mathbf{w}_3, b_3)$ would generalize

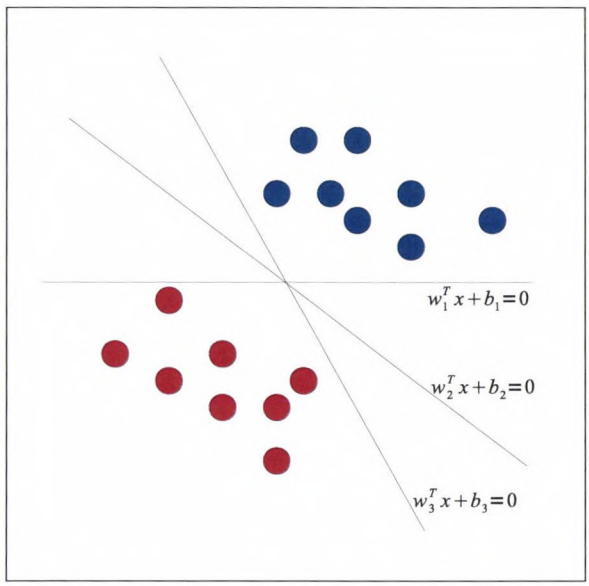


Figure 2.1: Separating hyperplanes for linearly separable data.

well to unseen data as their associated decision boundaries, defined as

$$\mathbf{w}_i^T \mathbf{x} + b_i = 0, \tag{2.3}$$

lie in close proximity to data examples. As it is often the case that data is limited and noisy, one would prefer to choose the classifier determined by $(\mathbf{w}_3, b_3)$ as it is more likely to predict correctly on a novel sample since there is a larger distance between the separating hyperplane and data points. Training data often represent a small sample with respect to the underlying prediction problem, and thus for a classifier to generalize, or predict accurately on unseen data, it is more useful for a classifier to learn a decision boundary that provides the largest separation between classes [24].

This notion of a large separation between classes is the underlying principle for max-margin classifiers, namely support vector machines (SVMs) [24, 25]. The margin is defined as the shortest perpendicular distance from the hyperplane to a data point and SVMs learn the optimal hyperplane: the separating hyperplane that maximizes the margin. Observing that $\mathbf{w}$ is orthogonal to any point on the separating hyperplane and

noting that any data point, $\mathbf{x}$, can be written as a sum of its orthogonal projection onto $\mathbf{w}$, plus some distance, $\gamma_{\mathbf{x}}$, in the direction of $\frac{\mathbf{w}}{||\mathbf{w}||}$, it follows that [26]

$$\gamma_{\mathbf{x}} = \frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{||\mathbf{w}||} = \frac{y_i(\mathbf{w}^T\mathbf{x} + b)}{||\mathbf{w}||}. \tag{2.4}$$

Maximizing Equation (2.4) subject to the constraints defined in Equation (2.2) is not well-defined as there are infinitely many solutions. To see this, note that $\mathbf{w}$ is orthogonal to the decision boundary. Thus an arbitrary rescaling of $\mathbf{w}$ can be offset with a corresponding b. To overcome this, it is necessary to either require $\mathbf{w}$ to be a unit vector or fix the location of the margin boundaries [27]. For this thesis, the focus will be on the latter as the derivation provides a straightforward geometric interpretation and relates well to the structural support vector machine described in Section 2.2.

In order to fix the margin location, the constraints of Equation (2.2) are replaced by

$$\left\{y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1\right\}_{i=1}^{N}, \tag{2.5}$$

which implicitly rescale $\mathbf{w}$ and b. As a result of these reformulated constraints, the margin is now defined as the distance between the hyperplanes $\mathbf{w}^T\mathbf{x} + b = \pm 1$. And referring to Equation (2.4), it is trivial to see that this distance is $\frac{2}{||\mathbf{w}||}$. Thus to maximize the margin, the norm of $\mathbf{w}$ must be minimized. Formally, this can be expressed as the following quadratic optimization problem

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & \left\{y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1\right\}_{i=1}^{N}. \end{aligned} \tag{2.6}$$

This relationship between $||\mathbf{w}||$ and the margin is illustrated in Figure 2.2 where the 2-dimensional toy dataset is revisited. It is worthwhile to note that there may be only a few data points that lie on the margin boundaries. These points that satisfy the equality constraints of Equation (2.5) are named the "support vectors" and their significance will be explored later.

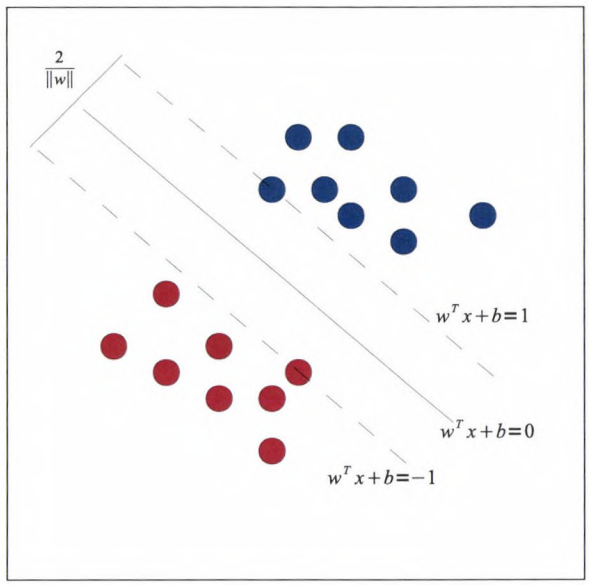


Figure 2.2: Max-margin principle illustrated on the 2-D toy dataset.

The assumption of linear separability between classes is most often infeasible for real-world prediction problems. Thus it is advantageous to allow a subset of the training data to fall within the margin boundaries (margin violations) or even be misclassified. To provide such tolerance, the notion of slack variables is introduced [28]. For each training example $\mathbf{x}_i$, an associated ξ_i is assigned where

$$\xi_i = \begin{cases} 0 & \text{if } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \\ \left|y_i - (\mathbf{w}^T\mathbf{x}_i + b)\right| & \text{otherwise.} \end{cases} \tag{2.7}$$

Thus for $\xi \in (0, 1]$, the slack represents a margin violation and for $\xi > 1$, a misclassification. To account for the slack variables, the optimization problem defined in Equation (2.6) is modified to introduce a penalty term over all accrued violations. The new optimization problem, referred to as the soft-margin formulation, is defined as [28]

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

where C is a positive constant that specifies the amount slack penalty. Hence as C increases, training error decreases at a cost of reducing the size of the margin. To illustrate

this concept, consider the data in Figure 2.3. As the dashed line indicates, the data is linearly separable. However, we may opt to accept a penalty with respect to $\mathbf{x}_j$ and $\mathbf{x}_k$ in order to obtain a larger margin.

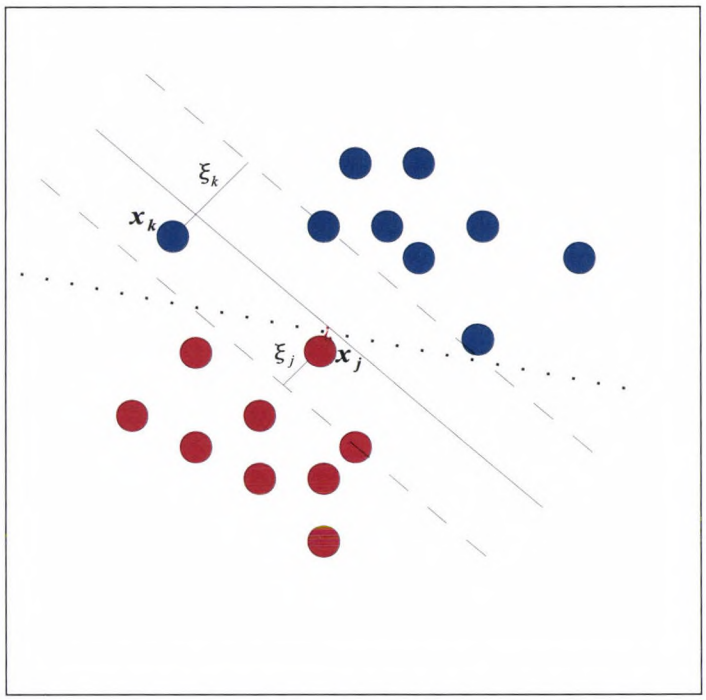


Figure 2.3: Introducing slack variables to form the soft margin SVM.

2.2 Structural Support Vector Machines

Unlike binary prediction problems, structured outputs methods consist of classification problems where the responses are complex objects, such as graphs, trees, and sequences [19, 29]. Formally, structured outputs classifiers learn a function

$$f : \mathcal{X} \rightarrow \mathcal{Y}, \tag{2.9}$$

where $\mathcal{Y}$ can represent a complex output space, consisting of interrelated components. While it may be possible to train classifiers for each element in the structured outputs space individually, such a method would unlikely capture any relationships among the

components of the label. Structural support vector machines (SSVMs) are a natural extension of SVMs that provide a large-margin solution for learning in structured output spaces.

The first step in extending SVMs into predicting structured outputs is to replace the linear discriminant in Equation (2.1) with a linear compatibility function defined as

$$F(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}), \tag{2.10}$$

where $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is the joint feature representation of $\mathbf{x}$ and $\mathbf{y}$. While the definition of the joint feature vectors are problem-specific, representations in this work are of the form

$$\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \psi_1(\mathbf{x}) \\ \psi_2(\mathbf{x}) \\ \vdots \\ \psi_{|\mathbf{y}|}(\mathbf{x}) \end{pmatrix} \tag{2.11}$$

such that each ψ_i extract features from $\mathbf{x}$ that are predictive for the i^{th} element of $\mathbf{y}$. Finally, to infer a label for a given input pattern, $\mathbf{x}$, we select the output that is most compatible with it, namely

$$f(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}). \tag{2.12}$$

The next step in formulating SSVMs is the definition of the margin. This is accomplished by measuring the distance between the joint feature representations of the correct output with the highest-ranked incorrect output for a given training example. As shown in Figure 2.4, the distance can be defined as

$$\gamma_{\mathbf{x}} = F(\mathbf{x}, \mathbf{y}) - \max_{\mathbf{y}_i \in \mathcal{Y} \setminus \mathbf{y}} F(\mathbf{x}, \mathbf{y}_i). \tag{2.13}$$

Figure 2.4 depicts a hypothetical, 2-dimensional joint feature space for a single input pattern. Unlike the SVM, the geometric interpretation of the SSVM hyperplane,

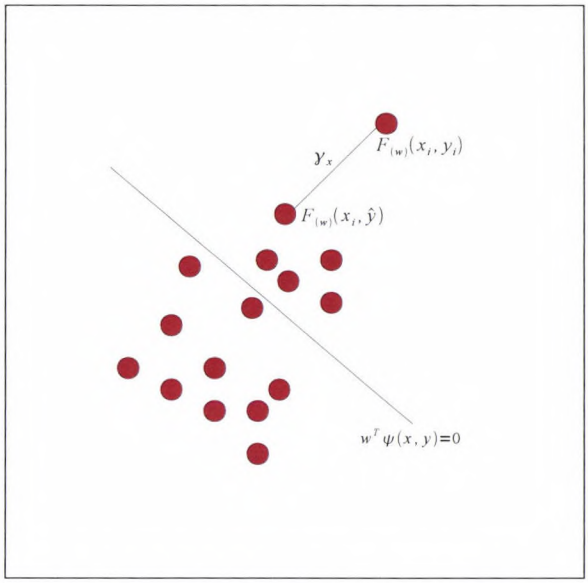


Figure 2.4: Hypothetical, 2-dimensional joint feature space where each point corresponds to the compatibility value of a possible label for a single input pattern. $\hat{\mathbf{y}}$ represents the most compatible, incorrect label for the input.

$\mathbf{w}^T\Psi(\mathbf{x}, \mathbf{y}) = 0$, is not significant. The compatibility function defined in Equation (2.10) merely serves a ranking mechanism for the output space.

As in the case of the SVM, the SSVM margin boundaries are fixed at ± 1 to prevent arbitrary rescaling of $\mathbf{w}$, and slack variables are introduced, leading to the following optimization problem

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}\,||\mathbf{w}||^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & \left\{\mathbf{w}^T\delta\mathbf{\Psi}_i(\mathbf{y}) \geq 1 - \xi_i,\ \forall\,\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y_i}\right\}_{i=1}^{N}, \end{aligned} \tag{2.14}$$

where

$$\delta\mathbf{\Psi}_i(\mathbf{y}) = \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}). \tag{2.15}$$

and C again is a positive constant assigning penalty to margin violations. Each constraint of Equation (2.14), $\mathbf{w}^T\delta\mathbf{\Psi}_i(\mathbf{y}) \geq 1 - \xi_i,\ \forall\,\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y_i}$, ensures that for the input pattern $\mathbf{x}_i$, the distances between every incorrect label $\mathbf{y}$ and the actual label, $\mathbf{y}_i$, is as large as possible within the space defined by the joint feature representation, $\mathbf{\Psi}$.

A standard method of measuring the closeness of a prediction $\mathbf{y}$ with respect to the actual label $\mathbf{y}_i$ is the use of loss functions [21], $\triangle : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$, where $\triangle(\mathbf{y}_i, \mathbf{y}_i) = 0$. In addition, as the similarity between the predictions and the actual label decreases, the loss function should increase accordingly. For structured outputs, the $0/1$ loss,

$$\triangle_{0/1}(\mathbf{y}, \mathbf{y}_i) = \begin{cases} 0 & \text{if } \mathbf{y}_i = \mathbf{y} \\ 1 & \text{otherwise,} \end{cases} \tag{2.16}$$

is too strict, as a predicted label that differs from the actual label by only a few components will have the same loss of a label that has no correct components. As with joint feature representations, loss functions for structured outputs are problem-specific. However, loss functions for label sequence learning are often variations of the hamming distance, defined as the number of non-matching components between the actual and predicted labels [30].

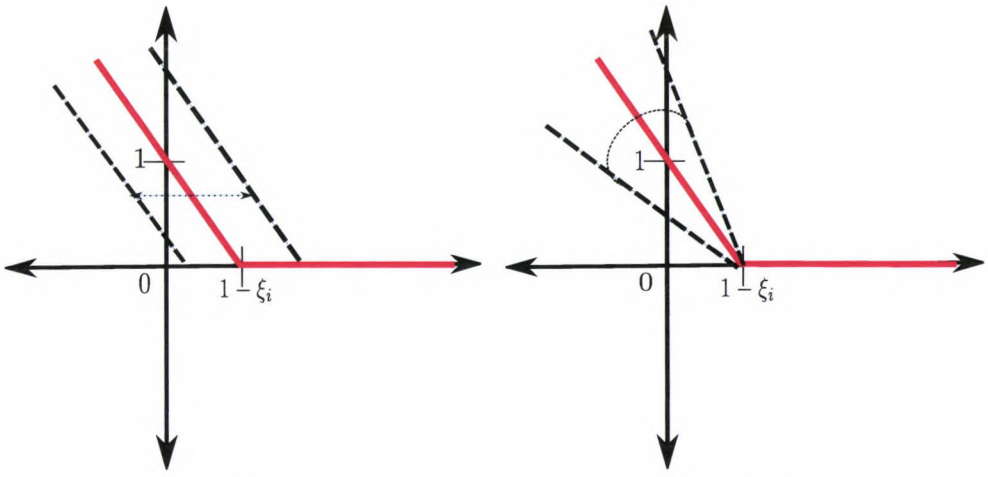
Two separate modifications to the constraints of Equation (2.14) were proposed to incorporate the loss between output labels [21]. The first involves rescaling the margin required for each $\mathbf{y} \in \mathcal{Y}$,

$$\left\{ \mathbf{w}^T \delta\mathbf{\Psi}_i(\mathbf{y}) \geq \triangle(\mathbf{y}, \mathbf{y}_i) - \xi_i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y_i} \right\}_{i=1}^{N}, \tag{2.17}$$

and the other rescales the amount of slack,

$$\left\{ \mathbf{w}^T \delta\mathbf{\Psi}_i(\mathbf{y}) \geq 1 - \frac{\xi_i}{\triangle(\mathbf{y}, \mathbf{y}_i)}, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y_i} \right\}_{i=1}^{N}. \tag{2.18}$$

Figure 2.5 illustrates the differences between these two approaches. The idea behind margin rescaling is that it may be beneficial to the overall optimization problem to reduce the margin for outputs that are similar to the correct label. As shown in Figure 2.5(a), this corresponds to "sliding" the standard hinge loss horizontally with respect to the loss. For slack rescaling, the motivation surrounds the belief that a margin violation of an output that is very different from the correct label should be penalized according to the amount of loss incurred by that output. Referring to Figure 2.5(b), this can be perceived geometrically as adjusting the angle of the hinge.



(a) Margin Rescaling (b) Slack Rescaling

Figure 2.5: Illustrating margin and slack rescaling.

Regardless of rescaling, a central problem of SSVMs is the computation of the argmax function that appears in Equation (2.12). As the size of the output space, $|\mathcal{Y}|$, can be exponential, it is necessary to either constrain the output space to be of polynomial cardinality or derive a polynomial-time algorithm for computing the argmax. Furthermore, cutting plane algorithms have been derived for SSVMs that iteratively add constraints until the desired precision is reached [31]. As it is expected that only a few constraints will be sufficiently violated, these algorithms select the most violated constraint (the most compatible, incorrect output label) for each training example and then perform optimization over this smaller set.

2.3 Dual Formulations and Kernel Functions

It is standard practice to employ the method of Lagrange multipliers to the optimization problems of Equations (2.8) and (2.14) and solve in the dual [32]. For the binary SVM, the Lagrangian is defined as

$$L_{\text{SVM}}(\mathbf{w}, \boldsymbol{\alpha}, b) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}^T \mathbf{x} + b) - 1). \tag{2.19}$$

By differentiating L_{SVM} with respect to $\mathbf{w}$ and setting it equal to zero, it follows that

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i. \tag{2.20}$$

Similarly for the SSVM, $\mathbf{w}$ is of the form

$$\mathbf{w} = \sum_{i=1}^{N} \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{(i,\mathbf{y})} \delta\boldsymbol{\Psi}_i(\mathbf{y}). \tag{2.21}$$

The α's correspond to the Lagrange multipliers and are only non-zero for active constraints. Those data points in the feature space with non-zero values are the support vectors and lie on the margin. Thus the solution to the dual formulation may lead to a sparse collection of the training examples. In this dual form of $\mathbf{w}$, feature vectors for both the SVM and SSVM now appear only as inner products. For instance, substituting into the SVM discriminant, we have

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} \tag{2.22}$$

and for the compatibility function of the SSVM,

$$F_{\boldsymbol{\alpha}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{(i,\mathbf{y})} \delta\boldsymbol{\Psi}_i(\mathbf{y})^T \boldsymbol{\Psi}(\mathbf{x}, \mathbf{y}). \tag{2.23}$$

The inner products of Equations (2.22) and (2.23) can now be replaced with kernel functions defined as $k(\mathbf{x}, \mathbf{x}')$ for binary SVMs and $k((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$ for SSVMs, such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle \tag{2.24}$$

and

$$k((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle (\boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})), (\boldsymbol{\Psi}(\mathbf{x}', \mathbf{y}')) \rangle . \tag{2.25}$$

In other words we can replace the occurrence of the inner product for the original feature vectors with a kernel function, k, so long as k defines an inner product in some feature space defined by the mappings $\boldsymbol{\phi}$ and $\boldsymbol{\Psi}$.

A trivial case is the linear kernel, namely

$$k_{linear}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle \tag{2.26}$$

where in this case, ϕ is the identity mapping.

A useful property of kernels is that their sum is a kernel as well. If ϕ_1 and ϕ_2 are the explicit mappings for kernels k_1 and k_2, respectively, it is straight-forward to verify that $k_1 + k_2$ is kernel by concatenating the feature vectors,

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\ &= \left\langle \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} \phi_1(\mathbf{x}') \\ \phi_2(\mathbf{x}') \end{bmatrix} \right\rangle. \end{aligned} \tag{2.27}$$

Other properties exist for constructing kernels, such as kernel products, linear combinations, and nonlinear mappings such as the polynomial and the Gaussian kernels [33].

Finally, it is important to note that while the SVM optimization problem described in Equation (2.8) is convex, standard optimization algorithms rely on decomposing the problem into a working set and iteratively optimize over these smaller sets. In the extreme case, Sequential Minimal Optimization (SMO) works with pairs of training examples [34]. Popular SVM implementations such as `LIBSVM` [35] and `SVM`^{light} [36] are variations of SMO that mostly differ in the manner they choose the members of the working sets.

Chapter 3

Methods

In this chapter, the details of the experimental set-up for calmodulin-binding prediction at the amino acid and protein levels is presented. Section 3.1 provides an overall picture of the prediction problems and describes the corresponding datasets. In Section 3.2, the sliding window SVM is defined and Section 3.3 details the definition of the SSVMs. Section 3.4 define the kernels for protein sequences used by both binary and SSVM classifiers. Finally, Section 3.5 explains the methods of assessing classifier performance each prediction problem.

3.1 Experiments

The first prediction problem we address is the identification of calmodulin binding sites within known calmodulin binders. Here prediction is at the amino acid level and classification involves identifying which amino acids in a protein are likely involved in the interaction. The Calmodulin Target Database provides a central data repository for calmodulin binding sites, containing data for nearly 200 proteins from a variety of species [37]. From this database, a non-redundant dataset of 153 proteins containing 185 binding sites was provided by Radivojac et al. [18]. This dataset was constructed such that no two proteins share more than 40% sequence identity and no two binding sites, more than 50% identity. This non-redundancy is to prevent bias in assessing classifier per-

formance, as a classifier trained over a subset of protein sequences would likely predict well on another subset containing proteins with similar sequences and thus, would not provide useful insight on the classifier's generalization capabilities.

The other problem addressed in this work is the identification of proteins that interact with calmodulin. This prediction problem is at the protein level and classification addresses the binary response of whether a protein has the potential to interact with calmodulin. In addition to the data with known binding sites, a dataset containing 241 known calmodulin binding proteins from *Arabidopsis* was constructed from the work of Popescu et al. [38]. While these proteins have no associated binding site information, this dataset will be used to evaluate the performance of the classifiers at the protein level.

3.2 Sliding Window SVM Classifiers

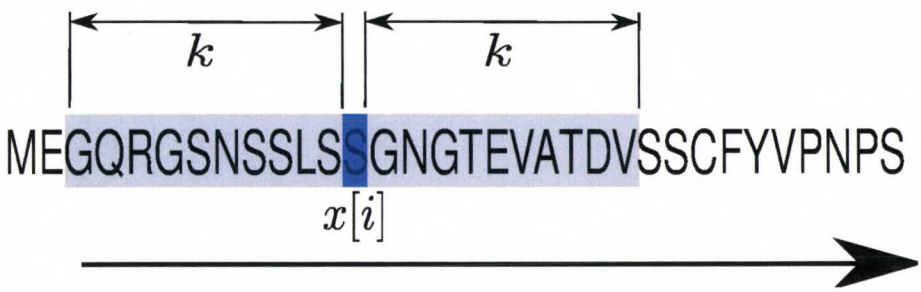


Figure 3.1: Sliding window method of capturing neighboring amino acid features.

Calmodulin binding sites are contiguous in sequence [16], suggesting the applicability of sliding window classifiers. Given a protein sequence, $\mathbf{x}$, the event of binding at amino acid $x[i]$ is represented as a window of half-length k, that incorporates neighboring amino acid features consisting of $x[i-k+1], x[i-k], \ldots, x[i+k]$. This concept is illustrated in Figure 3.1 where features for predicting the occurrence of a binding site at $x[i]$, represented as the shaded "S" in the figure, are generated by incorporating the highlighted substring `GQRGSNSSLSSGNGTEVATDV`. Creating features for subsequent

amino acids can be viewed as sliding this fixed-sized window across the entire protein sequence. For amino acids that lie at the beginning and end of the sequence, the window is collapsed. For instance, the window associated with the first amino acid will only consist of the amino acids up to position k in the sequence and the last amino acid will produce a window of amino acids containing $x[n-k+1], x[n-k], \ldots, x[n]$, where n is the length of the sequence.

For this work, we choose SVMs as the classifiers used in conjunction with sliding windows. Furthermore, as shown in Figure 3.2, the distribution of binding site lengths suggest a window half-length of $k = 10$, as the average binding site length is 21. The binary SVM is trained over features obtained from the actual binding sites (positive class) and a subset of all sliding window features for non-interacting amino acids (negative class). As features of two neighboring amino acids share all but one amino acid in their associated windows, the set of negative examples used in this work consists of windows corresponding to every 10^{th} amino acid that does not overlap an interaction site. And using a window half length of $k = 10$, each non-binding amino acid is present in at most 3 windows.

Finally, implementation of the binary SVM was performed using the `PyML` machine learning framework that provides data structures, kernels, and SVM optimization wrappers in `Python` [39].

3.3 Structural SVMs

The design of the SSVMs closely resembles the set-up for the binary SVMs. For the structural SVM, the output space for a protein, $\mathbf{x}$, consists of windows for every 10^{th} position in the protein sequence,

$$\mathcal{Y}_{\mathbf{x}} \equiv \left\{ 10y, y = 1, 2, \ldots, \left\lfloor \frac{|\mathbf{x}|}{10} \right\rfloor \right\}. \tag{3.1}$$

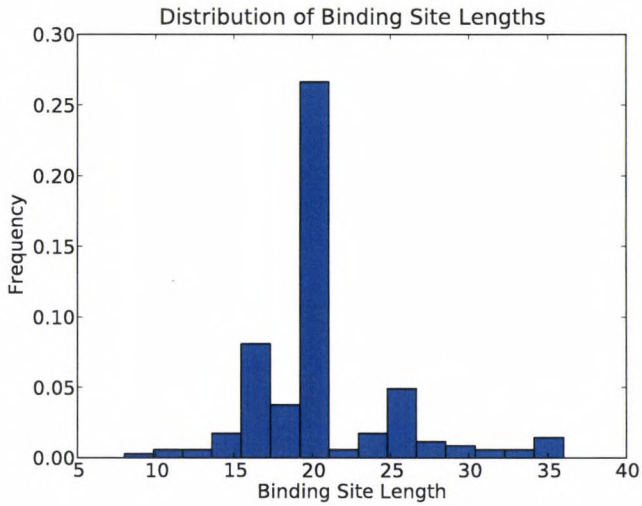


Figure 3.2: Distribution of calmodulin binding site lengths.

In addition, the joint feature representation for a protein $\mathbf{x}$ with a label, y, is defined as the set of features associated with a window centered at position y,

$$\Psi(\mathbf{x}, y) = \boldsymbol{\phi}(x[y-k, y-k+1, \ldots, y+k]), \tag{3.2}$$

where $\boldsymbol{\phi}$ is an explicit feature mapping for a kernel.

Finally, the loss is defined as the proportional overlap of a window $\hat{y}$ with the actual binding site y,

$$\triangle(y, \hat{y}) = \min\left(1, \frac{|c(y) - c(\hat{y})|}{c(y) + k}\right), \tag{3.3}$$

where $c(y)$, $c(\hat{y})$ denote the center of the actual binding site and the center of the predicted window, respectively, and k is the half-length of the window.

Implementation for the structural SVM was performed using `SVM^python`, which provides a `Python` interface to `SVM^struct` [40].

3.4 Kernels for Protein Sequences

This section describes the kernels used for predicting calmodulin binding sites. As the size of the binding sites vary, it is necessary to use kernels that are well-defined for variable length sequences. Along with a formal definition for each kernel, an exploratory data analysis is provided for each explicit feature representation.

3.4.1 p-spectrum Kernel

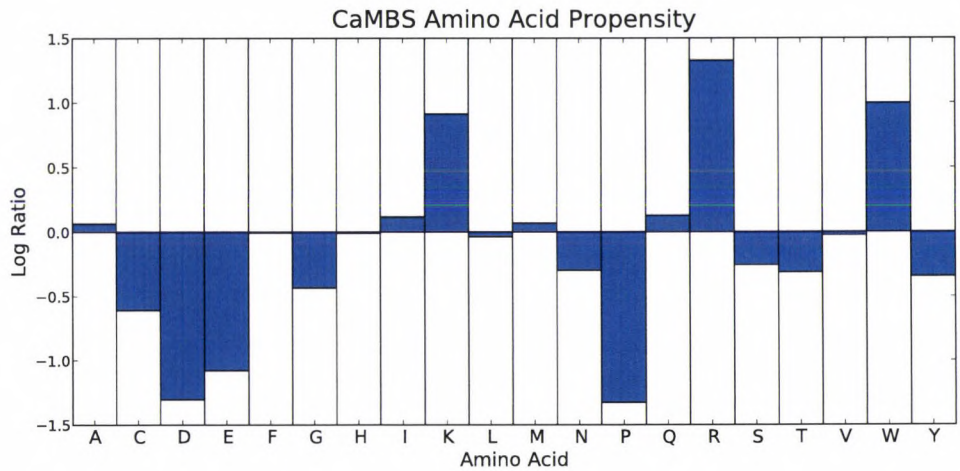


Figure 3.3: log ratios for amino acid propensity in calmodulin binding sites.

Figure 3.3 displays the amino acid propensity of an amino acid occurring in a binding site. Specifically for each amino acid, A, the following ratio is computed

$$\rho(A) = \log_2 \left(\frac{\text{freq}(A)_{\text{binder}}}{\text{freq}(A)_{\text{nonbinder}}} \right) \tag{3.4}$$

where $\text{freq}(A)_{\text{binder}}$ and $\text{freq}(A)_{\text{nonbinder}}$ are the frequencies of the amino acid in binding sites and non-binding regions of proteins, respectively. Thus a value of 1 indicates that a particular amino is twice as likely to occur in a binding site and a value of -1, twice as unlikely. As can be seen by these ratios, there is a clear propensity for a few amino

acids and a general lack of others in binding sites. In order to leverage this amino acid content, we use the p-spectrum kernel [41] for $p = 1$ and $p = 2$.

The p-spectrum of a string over an alphabet Σ counts the occurrences of all p-length substrings. For protein sequences, Σ is the 20-letter amino acid alphabet. Formally, the p-spectrum, $\Phi_p^{\Sigma}(\mathbf{x})$, is defined by

$$\Phi_p^{\Sigma}(\mathbf{x}) = (\phi_u(\mathbf{x}))_{u \in \Sigma^p}, \tag{3.5}$$

where $\phi_u(\mathbf{x})$ is the number of occurrences of u in $\mathbf{x}$. The kernel can now be defined as

$$k_{\text{pspec}}(\mathbf{x}, \mathbf{x}') = \left\langle \Phi_p^{\Sigma}(\mathbf{x}), \Phi_p^{\Sigma}(\mathbf{x}') \right\rangle. \tag{3.6}$$

Although the feature space of the p-spectrum kernel grows exponentially with p, namely $|\Sigma|^p$, the number of fixed-length substrings in a sequence is linear. Specifically there are $n - k + 1$ substrings of length k in a sequence of length n. In addition, it is possible to compute the kernel in linear time by scanning each sequence and building a hash table that maps a substring to its number of occurrences found in the string. The kernel can then be computed by scanning over the set of keys for either protein and accessing the number of occurrences for each substring in expected $O(1)$ time.

Finally, the p-spectrum kernel is normalized using the cosine kernel, $k_{\text{cosine}}(\mathbf{x}, \mathbf{x}')$ where

$$k_{\text{cosine}}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x}) k(\mathbf{x}', \mathbf{x}')}}. \tag{3.7}$$

The cosine kernel is so named as it can be perceived as computing the cosine of the angle between $\boldsymbol{\phi}_k(\mathbf{x}), \boldsymbol{\phi}_k(\mathbf{x}')$ where $\boldsymbol{\phi}_k$ is the explicit feature mapping for kernel k.

3.4.2 Physico-Chemical Kernel

The Amino Acid Index Database provides access to several physico-chemical properties of amino acids [42]. The physico-chemical mapping used in this work represents a

sequence of amino acids as a 60-dimensional feature vector where each component is the average of an individual property across the sequence,

$$\phi_{\text{pchem}}(\mathbf{x}) = \frac{1}{|\mathbf{x}|} \begin{pmatrix} \sum_{i=1}^{|\mathbf{x}|} \phi_1(x[i]) \\ \sum_{i=1}^{|\mathbf{x}|} \phi_2(x[i]) \\ \vdots \\ \sum_{i=1}^{|\mathbf{x}|} \phi_{60}(x[i]) \end{pmatrix}, \tag{3.8}$$

where $\phi_m(x[i])$ is the value associated with the amino acid at position i in the sequence for the m^{th} physico-chemical property. To prevent the magnitude of the physico-chemical properties from affecting classifier training, each feature vector is standardized using a fixed set of means and standard deviations computed by sampling windows over all training data.

To explore the usefulness of the physico-chemical features, a simple feature scoring method was applied to the 60 properties. For each property m, the Golub score [43] was computed,

$$golub(m) = \frac{\mu_{+1}(m) - \mu_{-1}(m)}{\sqrt{\sigma_{+1}(m)^2 + \sigma_{-1}(m)^2}}, \tag{3.9}$$

where $\mu_c(m)$ and $\sigma_c(m)$ are the mean and standard deviation of feature m with respect to class c. A positive value denotes a feature representative of the positive class and negative values are associated with the negative class.

Table 3.1 shows the top ten ranked features with respect to the Golub score. Many of these properties correspond to propensities of forming amphiphilic alpha-helices, which as mentioned in Section 1.2, agrees with the literature on calmodulin binding sites. The flexibility parameter is also interesting as the large, positive value suggests that calmodulin binding sites are more flexible than non-interacting regions. A high degree of flexibility often coincides with a high likelihood that a region is disordered, and as presented

Table 3.1: Top 10 physico-chemical features with respect to the Golub score.

AAIndex ID	Description	Score
KARP850103	Flexibility for two rigid neighbors [45]	0.293
AURR980101	Helix residue frequency N4 [46]	-0.208
FAUJ880108	Localized electrical effect [47]	0.204
AURR980120	Helix residue frequency C4 [46]	0.201
MITS020101	Amphiphilicity index [48]	0.187
CHAM830107	A parameter of charge transfer capability [49]	0.176
RICJ880108	Helix preference at N5 [50]	0.176
CHAM830104	Number of side chain atoms [49]	0.160
RICJ880103	Relative preference value at N-cap [50]	-0.155
RACS820112	Differential geometry parameter [51]	-0.149

in the work by Radivojac et al. [18], several calmodulin binding sites are intrinsically disordered. Other properties that show predictive power include the number of side chain atoms and charge, which are important for any protein-protein interaction [44].

3.4.3 PSI-BLAST Kernel

As mentioned in Section 1.2, calmodulin is highly conserved across a variety of species. And as it has been shown that interacting proteins often co-evolve [52], it is likely that that calmodulin interaction sites share similar evolutionary histories. To represent the evolutionary history of an amino acid sequence, position-specific scoring profiles are built for each protein using `PSI-BLAST` [53].

`BLAST` is a fast database search tool for nucleotide and protein sequences [54] that uses a statistic, called the E-value, that represents the number of matches expected by chance between a query sequence and a sequence within a database. Thus a small value corresponds to a significant match.

`PSI-BLAST` is an iterative search tool that first applies `BLAST` between a query sequence and a given database. Using a subset of the original database, chosen based on their E-values, a multiple sequence alignment (MSA) is constructed. From this MSA,

a position specific scoring matrix (PSSM) is computed that contains the frequencies of amino acids occurring in each position of the alignment. For a protein sequence $\mathbf{x}$ of length n,

$$\text{PSSM}(\mathbf{x}) = \begin{bmatrix} p_1(x[1]) & p_1(x[2]) & \dots & p_1(x[n]) \\ p_2(x[1]) & p_2(x[2]) & \dots & p_2(x[n]) \\ \vdots & \vdots & \ddots & \vdots \\ p_{20}(x[1]) & p_{20}(x[2]) & \dots & p_{20}(x[n]) \end{bmatrix}, \tag{3.10}$$

where $p_m(x[i])$ is the level of conservation of amino acid m for the i^{th} position of $\mathbf{x}$.

For this work, PSSMs for all training and test sequences were generated using `PSI-BLAST` against the non-redundant, UniRef50 database [55]. This database can be viewed as the protein space over all sequenced proteins that share no more than 50% sequence identity. Five iterations of `PSI-BLAST` were conducted and the final PSSM generated was used for constructing the features of the kernel.

Using these PSSMs, we define a feature mapping that collapses a PSSM into a length-400 feature vector where the features represent the average level of conservation of an amino acid with respect to all amino acids,

$$\phi_{\text{PSI}}(\mathbf{x}) = \frac{1}{|\mathbf{x}|} \left(\sum_{i=1}^{|\mathbf{x}|} \delta_m(x[i]) p_s(x[i]) \right)_{m,s=1}^{20}, \tag{3.11}$$

where $\delta_m(x[i])$ is the Kronecker delta function,

$$\delta_m(x[i]) = \begin{cases} 1 & \text{if } m = x[i] \\ 0 & \text{otherwise} \end{cases}. \tag{3.12}$$

Thus each feature, $\phi_{\text{PSI}}(\mathbf{x})_{m,s}$, corresponds to the average level of conservation of the amino acid s, over columns of the PSSM where an amino acid m is encountered in $\mathbf{x}$. The kernel is then defined as

$$k_{\text{PSI}}(\mathbf{x}, \mathbf{x}') = \langle \phi_{\text{PSI}}(\mathbf{x}), \phi_{\text{PSI}}(\mathbf{x}') \rangle, \tag{3.13}$$

and as these features are sparse, normalization is performed by applying the cosine kernel defined in Equation (3.7).

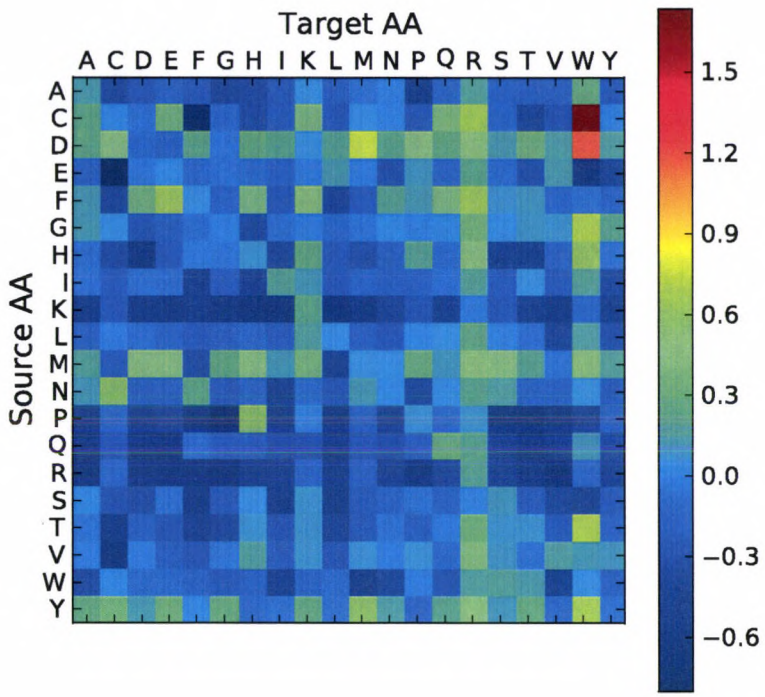


Figure 3.4: Amino acid substitution propensities in calmodulin binding sites.

The averages of conservation were computed using Equation (3.11) for both binding and non binding regions in calmodulin interaction partners. Figure 3.4 shows the log ratios over these averages. Colder colors across rows indicate more conservation in calmodulin binding sites while warmer colors can be perceived as faster rates of evolution. Notable differences include the conservation of arginine (R), lysine (K), and more than a 2-fold decrease in conservation from cysteine (C) to tryptophan (W). There is evidence through synthetic peptides where substituting a tryptophan residue with a cysteine residue within a calmodulin binding site reduces the affinity of the interaction [56].

Hence this lack of conservation may suggest evidence of the converse.

3.5 Classifier Performance Assessment

3.5.1 Leave-One-Protein-Out Cross Validation

Following Radivojac et al. [18], a leave-one-protein-out cross validation (LOPOCV) was carried out to assess the generalization performance of both the binary and structural SVMs. For each protein in the training set, a classifier was trained on the remaining proteins, and the prediction on the test protein was noted. This procedure was repeated until every protein was involved in testing.

3.5.2 ROC Analysis

To quantify performance, Receiver Operating Characteristic (ROC) analysis was conducted on the decision values (binary SVM), defined in Equation (2.1) and compatibility function values (structural SVM), defined in Equation (2.10). For the remainder of this section, the term discriminant value will be used to encompass both terms for simplicity.

ROC analysis is a popular method for assessing classifier performance [57]. Each point in an ROC curve corresponds to the true positive rate (*TPR*) with respect to the false positive rate (*FPR*) for a given threshold of the discriminant values. *TPR* is the fraction of examples from the positive class that have a corresponding discriminant value that is at least as large as the threshold whereas *FPR* is the fraction of examples from the negative class that have values at least as large as the threshold.

As an ideal classifier would systematically rank a positive example higher than a negative example, the corresponding ROC curve would plot increasing values of *TPR* while *FPR* remained fixed at 0; then the *TPR* would remain fixed at 1 while the *FPR* increased. For a random classifier, the *TPR* and the *FPR* would increase at the same rate. Figure 3.5 illustrates the concept of an ideal and random classifier.

As classifiers often produce decision values with different distributions, ROC curves are useful for comparing performance as they represent the ability to rank a positive example higher than a negative example. A classifier with an associated ROC curve that is consistently higher than another implies superior performance.

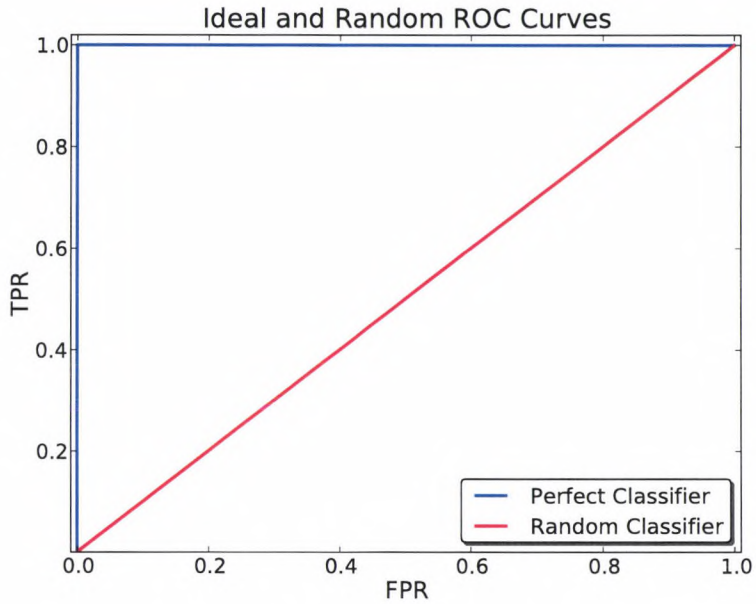


Figure 3.5: Illustrating the concept of the ideal and random classifier in ROC space.

Our ROC curves are generated by averaging the ROC curves produced for each left-out protein in a LOPOCV experiment. This averaging is performed using the vertical averaging technique suggested by Fawcett [58]. For fixed intervals along the *FPR* axis, the corresponding *TPR* values are averaged over all ROC curves with respect to the selected *FPR*s. In the cases where the selected *FPR*s did not coincide with the actual *FPR*s, the corresponding *TPR*s were interpolated by using the two neighboring (*FPR*, *TPR*) points.

The area under the ROC curve (AUC) provides a useful summary of performance for a classifier. Considering Figure 3.5, any classifier with an AUC > 0.5 has performance

that is better than random and an AUC $= 1$ is considered perfect. Another useful feature is that AUC can be viewed as the probability of a classifier scoring a randomly selected positive example higher than a random negative example [58].

In addition to the ROC and AUC, it is often useful to compute the ROC_{50} and the AUC_{50}, that computes the curve and area up until 50 negative examples are encountered [58]. By fixing the tolerance of false positives, it allows one to quantify the performance of a classifier for the highest-scored examples.

3.5.3 Model Selection

For each classifier, the slack penalty parameter, C, is selected using LOPOCV for values $C \in \{10^{-5}, 10^{-4}, \ldots, 10^{4}\}$. By using the AUC as the performance criterion, the C with the highest score is used for overall classifier assessment.

3.5.4 Gene Ontology

To determine the quality of the predictions at the protein level, a Gene Ontology (GO) analysis is performed on a set of highly-ranked proteins. GO provides a set of annotation terms that describe gene function using a hierarchical structure [59]. Three major hierarchies, cellular component, molecular function, and biological process, are maintained for genes that have annotated functional roles.

A popular GO analysis is identifying GO terms that are significantly overrepresented in a set of highly-ranked genes with respect to a background distribution of GO terms [60]. For this work, we are interested in determining if highly-ranked proteins, in terms of the maximum classifier score for an amino acid, from *Arabidopsis* contain overrepresented GO terms that are associated with proteins likely to interact with calmodulin.

Identification of significant terms is performed using `GOrilla` which computes a probabilistic score for each term found in the set of highly-ranked genes with respect

to the set of background genes [61, 62]. Using a threshold determined from the characteristics of the ROC curve for the best-performing classifier, proteins that have corresponding classifier scores greater than this threshold are given as the target class while the remainder are used as the background class.

Chapter 4

Results

In this chapter, we summarize classifier performance at the amino acid and protein levels for both binary and structural SVM in Sections 4.1 and 4.2, respectively. In addition, we perform an analysis of overrepresented Gene Ontology terms for highly-scored proteins in *Arabidopsis* in Section 4.3. In Section 4.4, a summary of this thesis and a discussion on future work is given.

4.1 Binding Site Prediction

This section provides the results for prediction of calmodulin binding sites. We performed LOPOCV experiments for SVMs and SSVMs using five kernels: 1-spectrum, 2-spectrum, Physico-Chemical, PSI-BLAST, and sum of the Physico-Chemical and PSI-BLAST kernels. All reported results are for a linear kernel over that kernel's feature space. In preliminary experiments, classifier performance did not improve with the use of a polynomial or Gaussian kernel.

AUC and AUC_{50} values for each experiment are provided in Tables 4.1 and 4.2. Our first observation is that there is a slight advantage for the binary SVM when looking at the AUC values. However, in terms of AUC_{50}, there is some indication that SSVMs perform better and suggests that SSVMs may provide more accurate predictions for highly-ranked predictions. In addition, both margin and slack rescaling provide com-

Table 4.1: Classifier performance at the amino acid level for binary and structural SVMs computed using LOPOCV for the 1-spectrum and 2-spectrum (1-spec, 2-spec), Physico-Chemical (pchem), PSI-BLAST (psi), and sum of Physico-Chemical and PSI-BLAST kernels (pchem+psi). Reported values are AUC scores.

Kernel	Binary SVM	SSVM (slack)	SSVM (margin)
1-spec	0.87	0.87	0.87
2-spec	0.87	0.86	0.86
pchem	0.87	0.86	0.86
psi	0.88	0.88	0.88
pchem+psi	0.88	0.88	0.87

Table 4.2: AUC_{50} values for the kernels and classifiers from Table 4.1

Kernel	Binary SVM	SSVM (slack)	SSVM (margin)
1-spec	0.62	0.62	0.64
2-spec	0.57	0.62	0.57
pchem	0.58	0.58	0.58
psi	0.61	0.64	0.64
pchem+psi	0.63	0.63	0.63

petitive results in terms of both AUC and AUC_{50} scores.

With respect to the kernels, the PSI-BLAST and sum of Physico-Chemical and PSI-BLAST kernels perform better or as good as the rest, highlighting the importance of evolution within calmodulin binding sites. A surprise is that the 1-spectrum does as well as the PSI-BLAST kernel in terms of AUC_{50} whereas the 2-spectrum performs worse. Finally, the addition of kernels provided no improvement in performance and the sum of Physico-Chemical and PSI-BLAST kernels was the highest performing sum of kernels over all possible pairs.

Figure 4.1 shows the ROC curves for each kernel using a binary SVM. Overall, the 2-spectrum was outperformed by all kernels whereas the PSI-BLAST kernel exhibits dominance. For the SSVM, similar rankings of kernels are observed. Figure 4.2 shows

the ROC curves with respect to SSVMs using slack rescaling and Figure 4.3, margin rescaling. For margin rescaling, the levels of dominance between kernels are more pronounced.

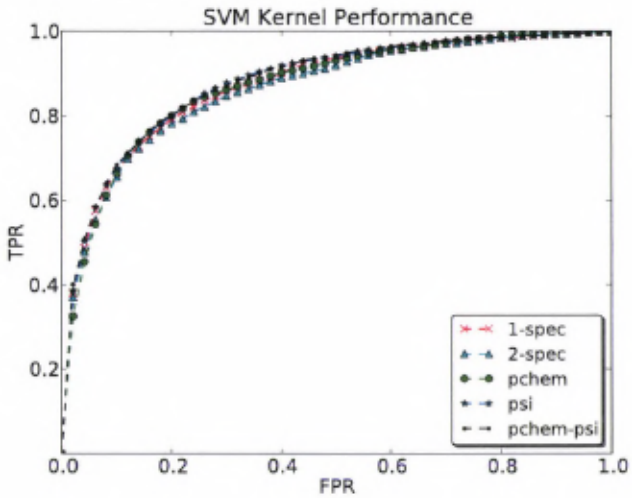


Figure 4.1: ROC curves for SVM classifiers with different kernels. Abbreviations in the legend follow the notation in Table 4.1.

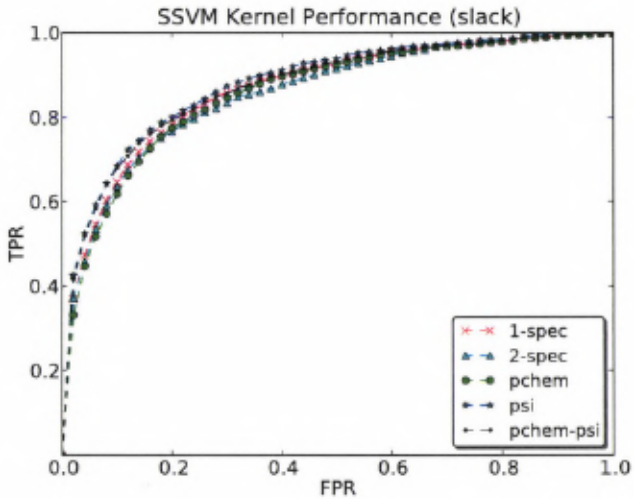


Figure 4.2: ROC curves for SSVM classifiers with different kernels using slack rescaling.

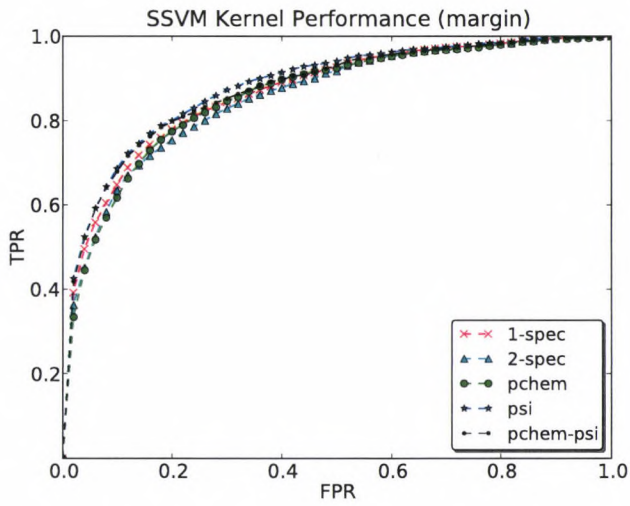


Figure 4.3: ROC curves for SSVM classifiers with different kernels using margin rescaling.

Figure 4.4 displays the ROC curves for each kernel separately for all classifiers, thus giving a means of comparing the classifiers with respect to a specific kernel. Again, performance is competitive among the classifiers with the exception of the Physico-Chemical kernel, where the binary SVM shows dominance for small *FPR*s.

In Table 4.3, the slack penalty, C, and the fraction of examples selected as support vectors (SVF), are presented for the best-performing classifier in each category. For SSVMs, the small value of C for the Physico-Chemical kernel implies a large amount of slack and thus providing evidence that the dense features of the physico-chemical properties are more difficult to separate.

As the output space for the SSVM was chosen to be roughly equivalent to the number of training examples for the SVM, it is possible to compare the sparsity of the classifiers, in terms of the number of support vectors, or equivalently, the number or examples (binary SVMs) or labels (SSVMs) with an associated, non-zero α in the dual formulation. And as shown in Table 4.3, the solutions for SSVMs are highly-sparse.

To compare the solutions learned by both the binary and SSVMs, a summary of

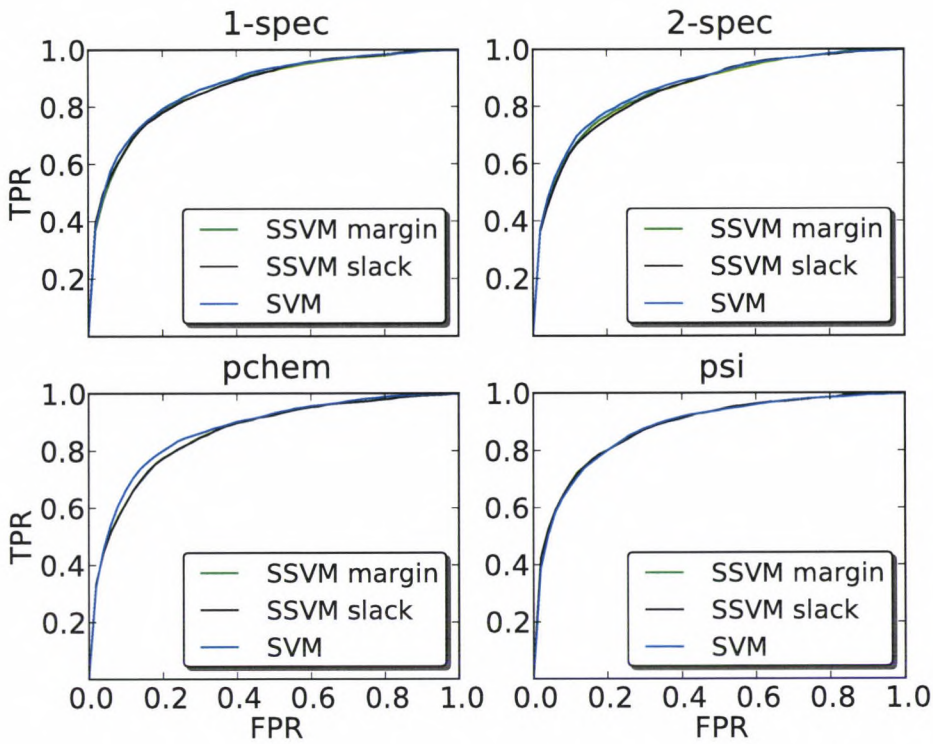


Figure 4.4: Comparative ROC analysis over classifiers with respect to a specific kernel.

Table 4.3: Slack penalties and support vector fractions (SVF)

Kernel	**Binary SVM**		**SSVM (slack)**		**SSVM (margin)**	
	C	SVF	C	SVF	C	SVF
1-spec	10^{0}	0.49	10^{-4}	0.01	10^{3}	0.02
2-spec	10^{-2}	0.99	10^{-1}	0.03	10^{-4}	0.01
pchem	10^{3}	0.48	10^{-4}	0.03	10^{-4}	0.01
psi	10^{-1}	0.92	10^{1}	0.02	10^{1}	0.02
pchem+psi	10^{0}	0.97	10^{2}	0.02	10^{-1}	0.01

classifier weights is provided. In Figure 4.5, the weights for the 1-spectrum for both classifiers are shown. The high positive weights of arginine (R), lysine (K), and tryptophan (W) and negative weights of aspartic acid (D), glutamic acid (E), and proline (P) agree with the propensities shown in Figure 3.3. However, there are clear differences

in weights of other amino acids between the classifiers, such as methionine (M) and threonine (T), indicating that the solutions to the classifiers are different.

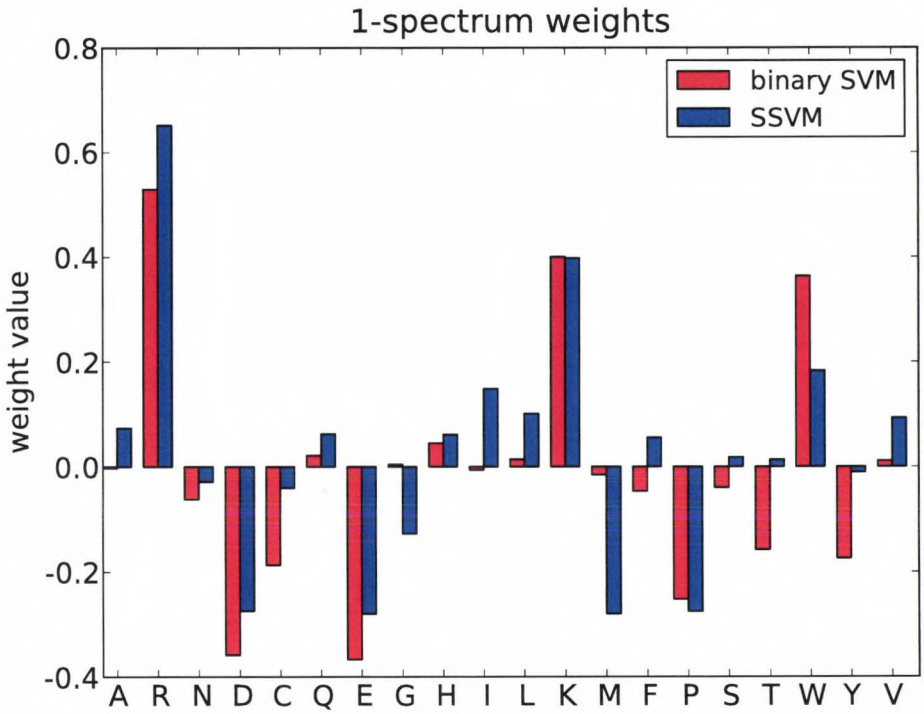


Figure 4.5: 1-spectrum weights for the binary and SSVM classifiers. Each bar represents a specific classifier's weight for the amino acids listed along the bottom of the figure.

Table 4.4 lists the top 10 features of the Physico-Chemical kernel according to the absolute value of the classifier weight for the binary SVM. Those associated with the SSVM are listed in Table 4.5. Features with a † correspond to features highly-ranked by the Golub score (see Table 3.1). Again there are features ranked highly for both classifiers, namely those associated with alpha-helices, electrical charge, and flexibility.

Table 4.4: Top 10 physico-chemical properties for the binary SVM. A † indicates a top 10 Golub score.

AAIndex ID	Description	weight
KARP850103†	Flexibility parameter for two rigid neighbors	0.475
RICJ880103†	Relative preference value at N-cap	-0.271
FAUJ880108†	Localized electrical effect	0.255
SUEM840102	Zimm-Bragg parameter sigma x 1.0E4	-0.226
RICJ880108†	Helix preference at N5	0.201
PALJ810113	Frequency of turn in all-alpha class	0.192
RACS820101	Differential geometry parameter	0.185
FAUJ880107	N.M.R. chemical shift of alpha-carbon	-0.181
QIAN880117	Weights for beta-sheet	0.170
SNEP660101	Chemical structure	0.168

Table 4.5: Top 10 physico-chemical properties for the structural SVM. A † indicates a top 10 Golub score.

AAIndex ID	Description	weight
KARP850103†	Flexibility parameter for two rigid neighbors	0.392
FAUJ880108†	Localized electrical effect	0.375
WOLS870103	Structure property of non-coded amino acids	-0.271
DIGM050101	Hydrostatic pressure asymmetry	-0.255
JOND750102	pK (-COOH)	-0.247
PONP800105	Surrounding hydrophobicity in beta-sheet	-0.237
OOBM850103	Optimized transfer energy parameter	0.229
AURR980101†	Helix residue frequency N4	-0.207
RICJ880103†	Relative preference value at N-cap	-0.170
AURR980120†	Helix residue frequency C	0.169

4.2 Interaction Prediction

To assess the accuracy for classifiers at the protein level, the classifiers trained at the amino acid level were applied to the *Arabidopsis* proteome (27,379 proteins). Each protein was assigned a score which is the maximum decision value over all amino acid scores.

Table 4.6: AUC at the protein level for binary and structural SVMs

Kernel	SVM	SSVM
1-spec	0.71	0.70
2-spec	0.71	0.68
pchem	0.64	0.66
psi	0.74	0.69
pchem+psi	0.73	0.69

Using the known calmodulin binders in *Arabidopsis*, described in Section 3.1, as the positive class and the remaining proteins as the negative class, ROC scores for each classifier were computed and the results are summarized in Table 4.6. Figures 4.6 and 4.7 feature the ROC curves for the binary SVM and SSVM classifiers. Here, the results show that binary SVMs performed better, with the exception of the Physico-Chemical kernel.

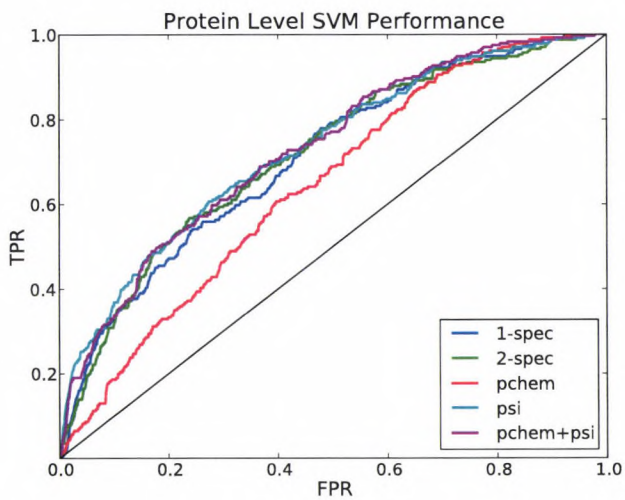


Figure 4.6: SVM performance at the protein level.

Another observation is that the Physico-Chemical kernel performed worse in comparison to the other kernels for both classifiers. This may be due to the fact that binding

sites, in general, share common physical and chemical properties and that classifiers using this kernel may be predicting amino acids involved in interaction sites unrelated to calmodulin. For instance, a binding site must occur on the surface of a protein and measures of hydrophobicity and electrostatic potential are predictive for identifying binding sites [63].

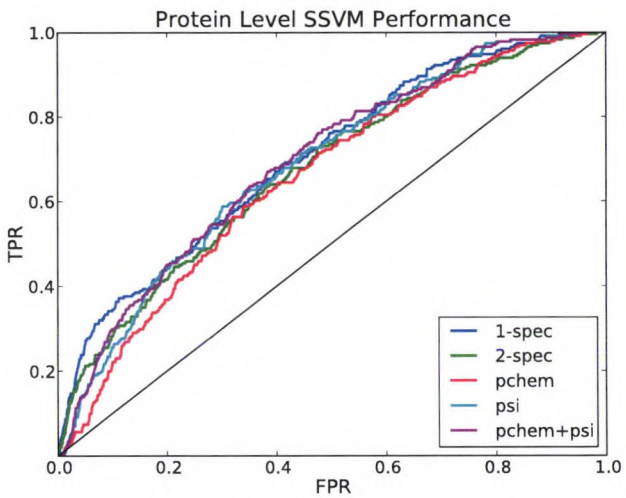


Figure 4.7: SSVM performance at the protein level.

4.3 Gene Ontology Terms Analysis

Using the binary SVM with the PSI-BLAST kernel, a threshold was chosen to select highly-ranked proteins such that the set would contain as few false positives as possible. The ROC curve for the classifier is shown in Figure 4.8 along with a hand-selected threshold. With respect to this threshold, only a little more than 21.4% of the known calmodulin binders have decision values greater than the threshold; however, only 2.6% of these proteins are expected to be false positives.

Selecting this threshold results in a dataset of 747 proteins, of which 53 are known to interact with calmodulin and 57 were not shown to interact according to the experi-

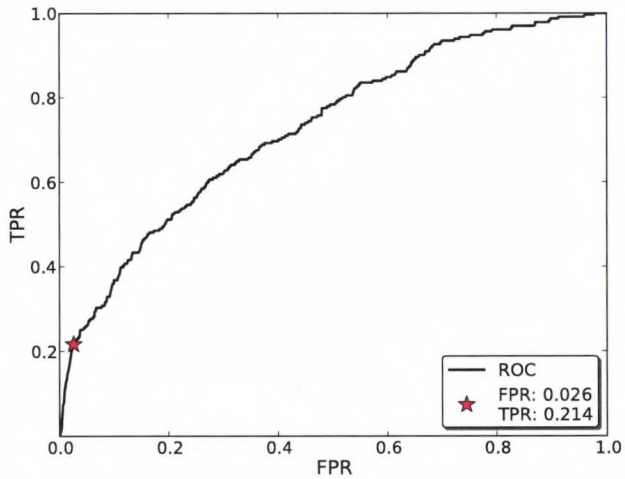


Figure 4.8: Selecting a threshold for highly-ranked proteins in *Arabidopsis*

ments by Popescu et al. [38]. Of the remaining 638 proteins, 398 have Gene Ontology annotations. Using this set as the target class and the remaining proteins not known to interact with calmodulin as the background class, significant over-representation of Gene Ontology terms was found in all three namespaces of annotation.

Table 4.7 shows the most significant GO terms related to biological processes. A number of these terms correspond to regulation of transcription. And as indicated in work of Bouche et al. [64], calmodulin is known to regulate a variety of transcription factors. In addition, the occurrence of phosphorylation related terms are most likely associated with the presence of kinases, and it is well known that calmodulin interacts with this class of proteins [65, 66].

In Table 4.8, molecular functions with significant over-representation are shown. Again, GO terms corresponding to transcription factor and kinase activity are abundant along with proteins involved in ATP functions. Experiments by Harper et al [67] and Bussemer et al. [68] have identified interactions of such proteins with calmodulin in *Arabidopsis*.

Table 4.7: Overrepresented GO terms with respect to biological processes

GO ID	**Description**	p**-value**
GO:0045449	regulation of transcription	2.79E-18
GO:0019219	regulation of nucleic processes	5.54E-18
GO:0051171	regulation of nitrogen compound metabolic process	6.35E-18
GO:0010556	regulation of macromolecule biosynthetic process	8.89E-18
GO:0009889	regulation of biosynthetic process	1.02E-17
GO:0031326	regulation of cellular biosynthetic process	1.02E-17
GO:0010468	regulation of gene expression	1.33E-17
GO:0080090	regulation of primary metabolic process	2.24E-17
GO:0031323	regulation of cellular metabolic process	2.91E-17
GO:0060255	regulation of macromolecule metabolic process	3.31E-17
GO:0019222	regulation of metabolic process	6.66E-17
GO:0044260	cellular macromolecule metabolic process	1.14E-11
GO:0065007	biological regulation	6.11E-11
GO:0050794	regulation of cellular process	3.4E-10
GO:0044267	cellular protein metabolic process	4.65E-10
GO:0050789	regulation of biological process	1.02E-9
GO:0043170	macromolecule metabolic process	1.45E-8
GO:0044237	cellular metabolic process	2.26E-7
GO:0019538	protein metabolic process	3.68E-7
GO:0009733	response to auxin stimulus	4.45E-7
GO:0006468	protein amino acid phosphorylation	4.91E-7
GO:0016310	phosphorylation	6.36E-7
GO:0006796	phosphate metabolic process	1.25E-6
GO:0006793	phosphorus metabolic process	1.25E-6
GO:0043687	post-translational protein modification	4.74E-6
GO:0006412	translation	6.22E-6

And finally, cellular component terms are summarized in Table 4.9. The over-representation of the nucleus component provides additional evidence to the argument that several of these proteins likely regulate transcription. In addition, the occurrence of the membrane terms may correspond to proteins involved in signal transduction.

Table 4.8: Overrepresented GO terms with respect to molecular function.

GO ID	**Description**	p**-value**
GO:0003676	nucleic acid binding	2.84E-25
GO:0003677	DNA binding	4.1E-22
GO:0003700	transcription factor activity	3.86E-21
GO:0030528	transcription regulator activity	8.8E-19
GO:0005488	binding	4.95E-9
GO:0042623	ATPase activity, coupled	2.41E-8
GO:0004386	helicase activity	2.99E-7
GO:0017111	nucleoside-triphosphatase activity	1.52E-6
GO:0003735	structural constituent of ribosome	1.8E-6
GO:0016462	pyrophosphatase activity	1.83E-6
GO:0016818	hydrolase activity, phosphorus-containing anhydrides	2.11E-6
GO:0016817	hydrolase activity, acting on acid anhydrides	2.64E-6
GO:0016887	ATPase activity	4.36E-6
GO:0008026	ATP-dependent helicase activity	6.31E-6
GO:0070035	purine NTP-dependent helicase activity	6.31E-6
GO:0016301	kinase activity	1.77E-5
GO:0005198	structural molecule activity	4.55E-5

4.4 Conclusion and Future Work

In this work, a comparison of binary and structural SVMs has been presented on the problem of calmodulin binding prediction and identifying calmodulin binding sites. Overall, the performance for both classes of predictors was competitive; however, the binary SVM shows better performance at binding prediction.

The results on predicting calmodulin binding activity show promise, and as indicated by the GO term analysis, it is likely that novel calmodulin binders can be identified with these classifiers. In addition, while Radivojac et al. [18] reported an amino acid level AUC of 0.89, the post-processing step of excluding potential false positives prevents a direct comparison of performance with this work impossible, and makes their results overly optimistic. We were able to obtain nearly that level of performance without

Table 4.9: Overrepresented GO terms with respect to cellular component.

GO ID	**Description**	p**-value**
GO:0005634	nucleus	4.75E-13
GO:0043226	organelle	3.76E-8
GO:0043229	intracellular organelle	3.76E-8
GO:0044445	cytosolic part	9.22E-8
GO:0033279	ribosomal subunit	1.49E-7
GO:0044424	intracellular part	8.06E-7
GO:0043231	intracellular membrane-bounded organelle	1.72E-6
GO:0043227	membrane-bounded organelle	1.72E-6
GO:0022625	cytosolic large ribosomal subunit	9.45E-6
GO:0030529	ribonucleoprotein complex	9.88E-6
GO:0005622	intracellular	1.22E-5

removing negative examples during testing. Moreover, their method generated features from multiple windows for an amino acid while the methods presented in this thesis considered a single window.

An extension of this work that may prove successful is applying Multiple Kernel Learning [69] where a linear combination of kernels is used. A simple sum of kernels in this work did not provide any improvement in performance; therefore a weighting associated with each kernel could lead to improved results.

Another extension is the inclusion of global features for a protein, such as GO annotations, overall sequence similarity with known binders, and protein domains. These may be incorporated in both methods at the binding site level as fixed features for a given protein, or used at the protein level for identifying calmodulin binding proteins within a proteome.

Finally, as some protein examples contain multiple calmodulin binding sites, it is likely that proteins used in this work are partially labeled. This may suggest that we are indeed predicting some binding sites correctly, and therefore it is probable that the performance of the classifiers is underestimated.

REFERENCES

[1] O. Keskin, A. Gursoy, B. Ma, and R. Nussinov. Principles of protein-protein interactions: what are the preferred ways for proteins to interact? *Chemical Reviews*, 108:1225–1244, April 2008.

[2] J. F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, N. Klitgord, C. Simon, M. Boxem, S. Milstein, J. Rosenberg, D. S. Goldberg, L. V. Zhang, S. L. Wong, G. Franklin, S. Li, J. S. Albala, J. Lim, C. Fraughton, E. Llamosas, S. Cevik, C. Bex, P. Lamesch, R. S. Sikorski, J. Vandenhaute, H. Y. Zoghbi, A. Smolyar, S. Bosak, R. Sequerra, L. Doucette-Stamm, M. E. Cusick, D. E. Hill, F. P. Roth, and M. Vidal. Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, 437:1173–1178, 2005.

[3] Huan-Xiang Zhou and Sanbo Qin. Interaction-site prediction for protein complexes: a critical assessment. *Bioinformatics*, 23(17):2203–2209, June 2007.

[4] Yanay Ofran and Burkhard Rost. Isis: interaction sites identified from sequence. *Bioinformatics*, 23(2):e13–e16, 2007.

[5] P.K. Jena, A.S. Reddy, and B.W. Poovaiah. Molecular cloning and sequencing of a cDNA for plant calmodulin: signal-induced changes in the expression of calmodulin. *Proceedings of the National Academy of Sciences U.S.A*, 86:3644–3648, 1989.

[6] J. Thompson, D. Higgins, and T. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[7] WY Cheung. Calmodulin plays a pivotal role in cellular regulation. *Science*, 207(4426):19–27, 1980.

[8] Linda Van Eldik and D. Watterson, editors. *Calmodulin and Signal Transduction*. Academic Press, April 1998.

[9] Allen R. Rhoads and Felix Friedberg. Sequence motifs for calmodulin recognition. *The FASEB Journal: Official Publication of the Federation of American Societies for Experimental Biology*, 11(5):331–340, April 1997.

[10] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[11] W. L. Delano. *The PyMOL Molecular Graphics System*. DeLano Scientific, Palo Alto, CA, USA, 2002. Software available at `http://www.pymol.org`.

[12] C. Yang, G. S. Jas, and K. Kuczera. Structure and dynamics of calcium-activated calmodulin in solution. *Journal of Biomolecular Structure and Dynamics*, 19(2):247–271, October 2001.

[13] Mingjie Zhang, Toshiyuki Tanaka, and Mitsuhiko Ikura. Calcium-induced conformational transition revealed by the solution structure of apo calmodulin. *Nature Structural & Molecular Biology*, 2(9):758–767, 1995.

[14] Y. S. Babu, C. E. Bugg, and W. J. Cook. Structure of calmodulin refined at 2.2 a resolution. *Journal of Molecular Biology*, 204(1):191–204, November 1988.

[15] M. Ikura, G. M. Clore, A. M. Gronenborn, G. Zhu, C. B. Klee, and A. Bax. Solution structure of a calmodulin-target peptide complex by multidimensional nmr. *Science*, 256(5057):632–638, May 1992.

[16] K.T. O'Neil and W.F. DeGrado. How calmodulin binds its targets: sequence independent recognition of amphiphilic alpha-helices. *Trends in Biochemical Sciences*, 15(2):59–64, 1990.

[17] Thomas G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30, London, UK, 2002. Springer-Verlag.

[18] Predrag Radivojac, Slobodan Vucetic, Timothy R. O'Connor, Vladimir N. Uversky, Zoran Obradovic, and A. Keith Dunker. Calmodulin signaling: Analysis and prediction of a disorder-dependent molecular recognition. *Proteins: Structure, Function, and Bioinformatics*, 63(2):398–410, 2006.

[19] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 3–10. AAAI Press, 2003.

[20] Artem Sokolov and Asa Ben-Hur. GOstruct: utilizing the structure of the Gene Ontology for accurate prediction of protein function. 8th Annual International Conference on Computational System Bioinformatics (CSB2009), 2009.

[21] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, September 2005.

[22] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Twenty Second International Conference on Machine Learning (ICML05)*, 2005.

[23] Haidong Wang, Eran Segal, Asa Ben-Hur, Qian-Ru Li, Marc Vidal, and Daphne Koller. Insite: a computational method for identifying protein-protein interaction binding sites on a proteome-wide scale. *Genome Biology*, 8(9):R192, 2007.

[24] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[25] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[26] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004.

[27] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[28] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[29] Gükhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

[30] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 26(2):147–160, 1950.

[31] T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[32] P. Wolfe. A duality theorem for nonlinear programming. *Quarterly of Applied Mathematics*, 19(3), 1961.

[33] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004.

[34] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[35] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[36] Thorsten Joachims. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT Press, Cambridge, MA, 1999.

[37] Kyoko L. Yap, Justin Kim, Kevin Truong, Marc Sherman, Tao Yuan, and Mitsuhiko Ikura. Calmodulin target database. *Journal of Structural and Functional Genomics*, 1(1):8–14, March 2000.

[38] Sorina C. Popescu, George V. Popescu, Shawn Bachan, Zimei Zhang, Montrell Seay, Mark Gerstein, Michael Snyder, and S. P. Dinesh-Kumar. Differential binding of calmodulin-related proteins to their targets revealed through high-density *Arabidopsis* protein microarrays. *PNAS*, 104(11):4730–4735, March 2007.

[39] Asa Ben-Hur. *PyML - machine learning in Python*, 2009. Software available at `http://pyml.sourceforge.net/`.

[40] Thomas Finley. *SVM^{python}*, 2007. Software available at `http://svmlight.joachims.org/svm_struct.html`.

[41] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[42] Shuichi Kawashima, Hiroyuki Ogata, and Minoru Kanehisa. AAindex: Amino acid index database. *Nucleic Acids Research*, 27(1):368–369, 1999.

[43] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, October 1999.

[44] S. Jones and J. M. Thornton. Principles of protein-protein interactions. *Proceedings of the National Academy of Sciences U.S.A*, 93:13–20, January 1996.

[45] P. A. Karplus and G. E. Schulz. Prediction of chain flexibility in proteins. *Naturwissenschaften*, 72(4):212–213, April 1985.

[46] R. Aurora and G. D. Rose. Helix capping. *Protein science : a publication of the Protein Society*, 7(1):21–38, January 1998.

[47] JL Fauchère, M Charton, LB Kier, A Verloop, and V Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *International Journal of Peptide and Protein Research*, 32(4):269–278, October 1998.

[48] S. Mitaku, T. Hirokawa, and T. Tsuji. Amphiphilicity index of polar amino acids as an aid in the characterization of amino acid preference at membrane-water interfaces. *Bioinformatics*, 18:608–616, April 2002.

[49] M. Charton and B. I. Charton. The dependence of the Chou-Fasman parameters on amino acid side chain structure. *Journal of Theoretical Biology*, 102:121–134, May 1983.

[50] J. S. Richardson and D. C. Richardson. Amino acid preferences for specific locations at the ends of alpha helices. *Science*, 240(4859):1648–1652, June 1988.

[51] S. Rackovsky and H. A. Scheraga. Differential geometry and polymer conformation. 4. Conformational and nucleation properties of individual amino acids. *Macromolecules*, 15:1340–1346, 1982.

[52] Chern-Sing Goh, Andrew A. Bogan, Marcin Joachimiak, Dirk Walther, and Fred E. Cohen. Co-evolution of proteins with their interaction partners. *Journal of Molecular Biology*, 299(2):283 – 293, 2000.

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

[54] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[55] Baris E. Suzek, Hongzhan Huang, Peter B. McGarvey, Raja Mazumder, and Cathy H. Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[56] Thomas Vorherr, Peter James, Joachim Krebs, Agnes Enyedi, Daniel J. McCormick, John T. Penniston, and Ernesto Carafoli. Interaction of calmodulin with the calmodulin binding domain of the plasma membrane calcium pump. *Biochemistry*, 29(2):355–365, 1990.

[57] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[58] Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories Palo Alto, 2004.

[59] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M.

Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25:25–29, May 2000.

[60] P. Khatri and S. Drăghici. Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*, 21:3587–3595, September 2005.

[61] E. Eden, R. Navon, I. Steinfeld, D. Lipson, and Z. Yakhini. GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. *BMC Bioinformatics*, 10:48, 2009.

[62] E. Eden, D. Lipson, S. Yogev, and Z. Yakhini. Discovering motifs in ranked lists of DNA sequences. *PLoS Computational Biology*, 3:e39, Mar 2007.

[63] Tapan Patel, Manoj Pillay, Rahul Jawa, and Li Liao. Information of binding sites improves prediction of protein-protein interaction. In *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*, pages 205–212, Washington, DC, USA, 2006. IEEE Computer Society.

[64] N. Bouche, A. Scharlat, W. Snedden, D. Bouchez, and H. Fromm. A novel family of calmodulin-binding transcription activators in multicellular organisms. *Journal of Biological Chemistry*, 277:21851–21861, Jun 2002.

[65] David Blumenthal, Koji Takio, Arthur Edelman, Harry Charbonneau, Koiti Titani, Kenneth Walsh, and Edwin Krebs. Identification of the calmodulin-binding domain of skeletal muscle myosin light chain kinase. *Biochemistry*, 1985.

[66] M. Charpenteau, K. Jaworski, B. C. Ramirez, A. Tretyn, R. Ranjeva, and B. Ranty. A receptor-like kinase from Arabidopsis thaliana is a calmodulin-binding protein. *Biochemical Journal*, 379:841–848, May 2004.

[67] J. F. Harper, B. Hong, I. Hwang, H. Q. Guo, R. Stoddard, J. F. Huang, M. G. Palmgren, H. Sze, and M. L. Evans. A novel calmodulin-regulated Ca2+-ATPase (ACA2) from Arabidopsis with an N-terminal autoinhibitory domain. *Journal of Biological Chemistry*, 273:1099–1106, Jan 1998.

[68] J. Bussemer, F. Chigri, and U. C. Vothknecht. Arabidopsis ATPase family gene 1-like protein 1 is a calmodulin-binding AAA+-ATPase with a dual localization in chloroplasts and mitochondria. *FEBS Journal*, 276:3870–3880, Jul 2009.

[69] Sören Sonnenburg, Bernhard Schlkopf Bernhard, P. Bennett, and Emilio Parradohernndez. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:2006, 2006.