UNIVERSITY of CALIFORNIA
SANTA CRUZ

CLASSIFYING G-PROTEIN COUPLED RECEPTORS WITH SUPPORT VECTOR MACHINES

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Rachel Karchin

June 2000

The thesis of Rachel Karchin is approved:

Professor David Haussler, Chair

Associate Professor Kevin Karplus

Professor Roberto Bogolmoni

Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Classifying G-protein coupled receptors with support vector machines

by

Rachel Karchin

The enormous amount of protein sequence data uncovered by genome research has increased the demand for computer software that can automate the recognition of new proteins. I discuss the relative merits of several automated, state-of-the-art methods for recognizing *G-protein coupled receptors (GPCRs)*, a superfamily of cell membrane signalling proteins. GPCRs are the focus of a significant amount of current pharmaceutical research because they play an important role in many diseases. However, their structures remain largely unsolved. The methods described in this thesis use only primary sequence information to make their predictions. I provide true positive vs. false positive plots for Fisher kernel support vector machines (SVMs), hidden Markov models (HMMs), BLAST and Smith-Waterman on two distinct discrimination problems: recognizing the GPCR superfamily and recognizing medically important sub-families of GPCRs. The SAM-T99 hidden Markov model is best able to recognize the GPCR superfamily. Average percent correct at the minimum error point (*MEP*) on our superfamily test sets is 99.96% for T99, 99.78% for SVM, 96.38% for Smith-Waterman, and 93.18% for BLAST. However, the SAM-T99 HMM is too general to successfully discriminate GPCR subfamilies. A Fisher kernel support vector machine built "on top of" the T99 HMM is best able to recognize the fine distinctions between GPCR subfamilies. The MEP average percent correct on our subfamily test sets is 99.73% for SVM, 99.08% for BLAST, and 97.90% for T99.

Acknowledgements

This thesis would not have been possible without the accomplishments and support of many people. Tommi Jaakkola developed the Fisher kernel and wrote the SVM learning software used in my experiments. David Haussler and Mark Diekhans, together with Tommi, first applied SVM learning to the problem of protein family recognition. Both of them provided valuable advice and input at every stage of this thesis. In addition, Mark shared his Fisher score vector software and his script for BLAST family pairwise search. Bill Grundy developed the family pairwise search algorithm. Florence Horn and Gert Vriend provided all sequence data used in my experiments. Their expertise was crucial to the design of all the experiments.

I also want to thank Nello Cristianini for his explanations of SVM theory, and for letting me read chapters from his wonderful book on SVM learning before the book was published. Kevin Karplus shared tips on his SAM-T9X methods and provided numerous suggestions and ideas. Roberto Boglomoni advised me on the biochemistry of GPCRs and shared his expertise on rhodopsin function and structure. Leslie Grate gave invaluable assistance with my Kestrel Smith-Waterman experiments. Lydia Gregoret allowed me to audit her excellent graduate class on protein chemistry.

My undergraduate advisor, Richard Hughey, encouraged me to pursue an advanced degree, and is also the principal architect of the Kestrel parallel processor. The National Physical Science Consortium (NPSC) awarded me a graduate fellowship and made it possible for me to attend graduate school. Ron Musick, my fellowship sponsor, has been a great source of advice. Carol Mullane and Kristine Kilkenny have been there for me throughout my graduate career, offering both encouragement and the inside scoop

on many matters. Melissa Cline has given generous and invaluable suggestions about every aspect of graduate school life. My parents, Edward and Jean White, are a source of enormous love and support and have maintained an avid, ongoing interest in my work. My husband, Lewis Karchin, has cheerfully put up with my crazy schedule. Finally, the members of the Computational Biology group: Terry Furey, Chuck Sugnet, Christian Barrett, Sugato Basu, and Fan Shen have all shared their knowledge with me and improved the quality of my work and my life.

Chapter 1

Introduction

Support vector machines (SVMs) are a class of statistical learning algorithms whose theoretical basis was first presented by V. Vapnik in 1979 [61]. During the 1990s, they became extremely popular in the machine learning community. In this paper, I apply *SVMs* to the simplest learning problem, two-class *pattern recognition*, in which a learning machine is shown a series of labelled examples from two categories and is trained to distinguish between them. If training is successful, when presented with new examples, it is able to predict their category with a minimal number of errors. In my experiments, the data to be classified are protein sequences of the G-protein coupled receptor (GPCR) family.

GPCRs were selected because an enormous amount of current pharmaceutical research is aimed at understanding their structure and function. They play an important role in human physiology and disease, but their structures remain mostly unsolved. They are difficult to crystallize; NMR spectroscopy can't be used because it requires high concentrations of dissolved proteins and most GPCRs will not dissolve in normal solvents. To date, the structure of only one GPCR has been solved using electron diffraction at medium

resolution [64, 5].

In contrast, the amino-acid sequences of over 1000 GPCRs are known, and an enormous amount of protein sequence information will soon be available as a result of the Human Genome and other genome projects (fruitfly, worm, etc.). A method for accurate sequence-based prediction of GPCR function would be highly valuable to research biologists and to pharmaceutical companies. I will present the results of a series of controlled experiments to argue that SVMs can be used as the foundation of such a method. Out of all the methods I tested for this thesis, only SVMs were able to accurately recognize GPCR subfamilies, groups of GPCRs that bind a common ligand such as a histamine molecule.

As described above, a trained learning machine should be able to make good predictions about the class membership of a previously unseen example. An SVM makes its prediction using a mathemetical tool known as a *kernel function* which measures the similarity between two examples.

Tommi Jaakola and David Haussler have developed a new kind of kernel function that can be used to measure the similarity between two protein sequences [29]. The method is based on building a statistical model of a group of protein sequences, the *hidden Markov model* (HMM), then computing the likelihood that the protein sequence of interest was generated by the model. The HMM is composed of a linear series of *match states* that correspond to conserved amino acid positions in the sequence. Each match state has its own probability distribution over the amino acid alphabet. The HMM also contains probabilities for the transition between each match state i and match state $i + 1$. The posterior probabilities of the states and transitions, conditioned on a particular sequence, form a vector of sufficient statistics for that sequence called a *Fisher score vector*. Jaakola

and Haussler derive a theoretical basis for a distance metric between protein sequences using Fisher score vectors.

Experimental support for the effectiveness of this new method appears in their paper, written in conjunction with Mark Diekhans and presented at ISMB 1999 [28]. Their modified Fisher kernel SVM made significantly fewer classification errors on the same sets of protein sequences from SCOP (version 1.37 PDB90) than two competing methods, *BLAST FPS* (family pairwise search using BLAST) [20] and *SAM Target 98* (SAM-T98), developed by Kevin Karplus at University of California, Santa Cruz. SAM-T98 classifies protein sequences according to their similarity to an HMM for a target protein family [33]. A description of the family pairwise search algorithm appears in Section 3.4.1.

To see if I could get comparable results with the Fisher kernel SVM, I ran a similar series of experiments to recognize the GPCR superfamily and a few selected subfamilies. In addition to SVMs, SAM-T98, and BLAST FPS, I tested SAM-T99 (an upgrade of SAM-T98) and family pairwise search using Smith-Waterman scoring [53]. In almost all of my experiments, the SVM method outperformed BLAST and Smith-Waterman FPS by a wide margin. Smith-Waterman FPS did better than BLAST. I was unable to get good results with the SAM-T98 method used in the ISMB paper. In about half of my experiments, it produced over-generalized models by drawing non-GPCR proteins into the training alignments. The resulting HMMs were not useful in recognizing GPCRs. However, Fisher kernel support vector machines built on top of these overgeneralized models were able to discriminate the GPCR superfamily very well.

When SAM-T99 was released, I created a series of new HMMs, built SVMs on top of them, and repeated my experiments on the same data sets. These new HMMs

did not overgeneralize when discriminating the GPCR superfamily. In fact, they perfectly discriminated the superfamily on most of my test sets, and outperformed the SVMs built on top of them. Currently, SAM-T99 HMMs are the best method for GPCR superfamily recognition known to this author.

However, SAM-T99 models do not work well on the problem of subfamily recognition. SAM-T99 is an iterative model-building algorithm that constructs an initial HMM from a training set of positive examples selected by the user. It uses this model to search a protein database for homologs, appends them to the original training set and creates a multiple sequence alignment to train a second (and third and fourth) HMM. Even though I started this process with a sequence from a GPCR subfamily of interest, the training sets it selected were overgeneralized by the inclusion of many sequences that did not belong to the subfamily. It is not surprising that the resulting models are unable to discriminate the subfamily.

I attempted to improve SAM-T99 subfamily discrimination by restricting the training set to a group of known positive examples. This actually worsened the situation, and resulted in HMMs that made more discrimination errors than the first group of overgeneralized models.

As mentioned above, Fisher kernel support vector machines built on top of SAM-T98 models of the GPCR superfamily were able well discriminate the superfamily, even though the SAM-T98 models could not do this on their own. By building SVMs on top of SAM-T99 models, it is also possible to clean up overgeneralized SAM-T99 subfamily HMMs and very accurately discriminate GPCR subfamilies. I will present detailed results of my experiments recognizing three medically important GPCR sub-families with SVMs

in Section 4.

Chapter 2

Background

2.1 GPCRs

G-protein coupled receptors are a superfamily of cell membrane proteins that share common function and structure, but are not necessarily related by sequence similarity. They are predicted to consist of an initial extra-cellular N-terminal sequence, seven transmembrane alpha-helices which are linked by three sets of alternate intracellular and extracellular loops, and a final intracellular C-terminal sequence [65]. A cartoon version of this structure is shown in Figure 2.1.

Functionally, they play a key role in a cellular signalling network that regulates many basic physiological processes: neurotransmission, cellular metabolism, secretion, cellular differentiation and growth, inflammatory and immune responses, smell, taste and vision [9]. GPCRs are of great interest to pharmaceutical companies as targets of structure-based drug design. Because they are so important to cell function, developing compounds which can augment or inhibit their activity is likely to provide enormous benefits to human

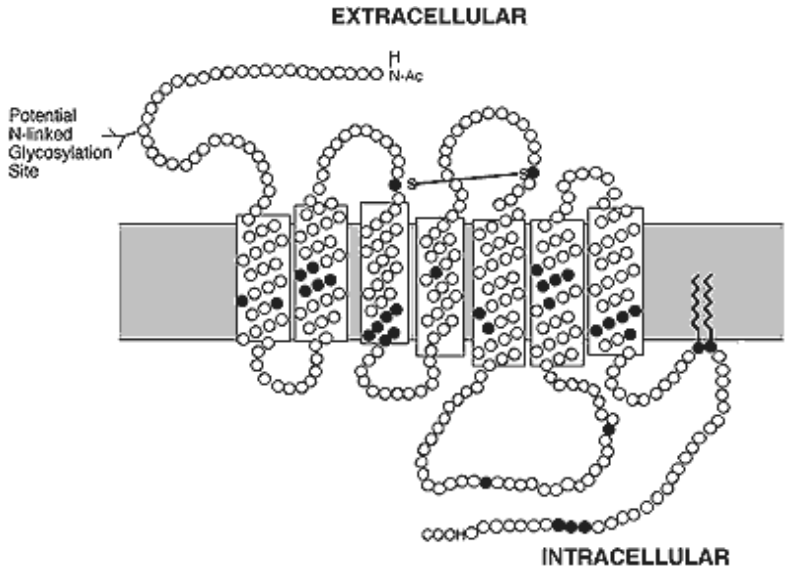


Figure 2.1: A cartoon showing the predicted structure of a member of the GPCR superfamily [46]

health and quality of life [10].

2.1.1 GPCR Structure

As mentioned in the Introduction, attempts to solve GPCR structure with crystallography and NMR have produced very limited results. Electron diffraction has solved the structure of the seven transmembrane helices of *Bacteriorhodopsin* (Brh), a GPCR-like molecule, at high resolution [36] and the seven transmembrane *7TM* helices of one real GPCR (rhodopsin) at medium resolution [34].

Theoretical predictions of GPCR structure based on primary amino acid sequence have been developed through hydropathy analysis [35], distance geometry algorithms [39], homology modeling [59], and specialized hidden Markov models (HMMs) [56].

Domains inside the cell membrane are presumed to be largely composed of hydrophobic amino acids [35]. By plotting the hydrophobicity of amino acids in the primary

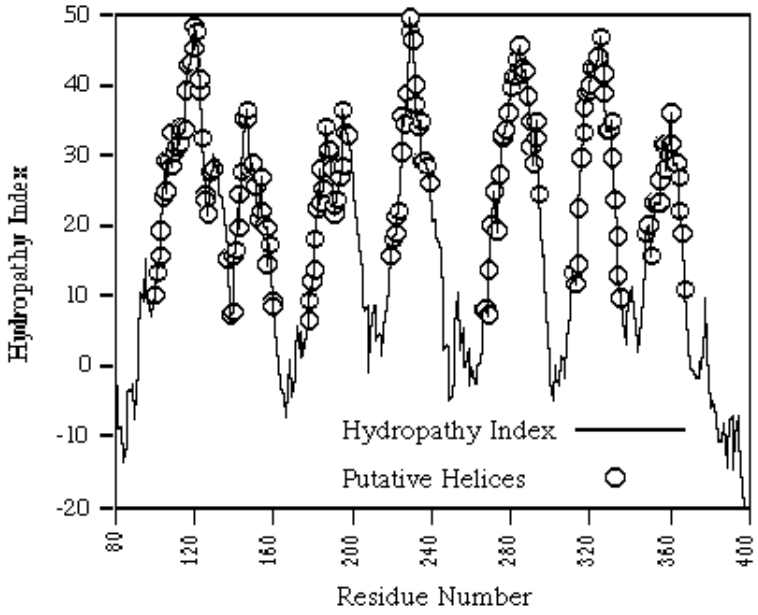


Figure 2.2: A hydropathy plot of the human thrombin receptor showing seven hydrophobic transmembrane regions [35]

sequence of a GPCR, seven distinct peaks can be observed. It is generally accepted that these correspond to the predicted seven transmembrane domains. The hydropathy plot of human thrombin receptor is shown in Figure 2.2.

The Mosberg Lab at the University of Michigan, Ann Arbor has constructed models of 30 different GPCRs, based on distance geometry constraints. Their method is similar to the way that protein structures are calculated from NMR spectroscopy data. For example, to predict the structure of rhodopsin-like GPCRs, they created an average structure based on a sequence alignment of 410 GPCRs from this family. They used the alignment to identify candidate H-bonding pairs between polar side chains in the transmembrane helices. The pairs produced distance constraints which were iteratively refined until each buried polar side-chain from each of the 410 sequences in the alignment participated in at least one hydrogen bond in the final structure [39]. Their prediction for

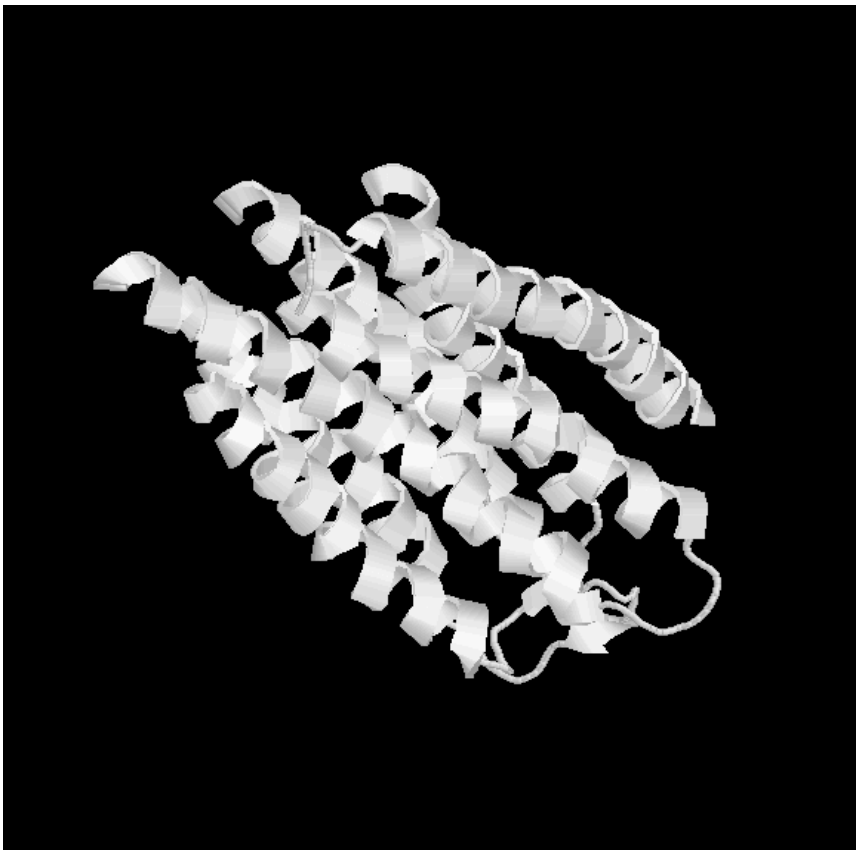


Figure 2.3: Predicted structure for the transmembrane helices of a delta-opioid receptor

the delta-opioid receptor with JH42 is shown in Figure 2.3.

Many researchers have attempted to use homology modeling techniques to predict GPCR structure. This approach takes a known protein structure and uses it as a guide to predict the structure of another protein to which it has a high degree of sequence identity/similarity.

However, until recently, the only available structure of a GPCR "homolog" was Bacteriorhodopsin, and there is no easily detectable sequence homology between Brh and GPCRs [64]. In 1997, the Schertler lab published the structure of rhodopsin [5], providing a greatly improved template from which to predict the structure of other GPCRs.

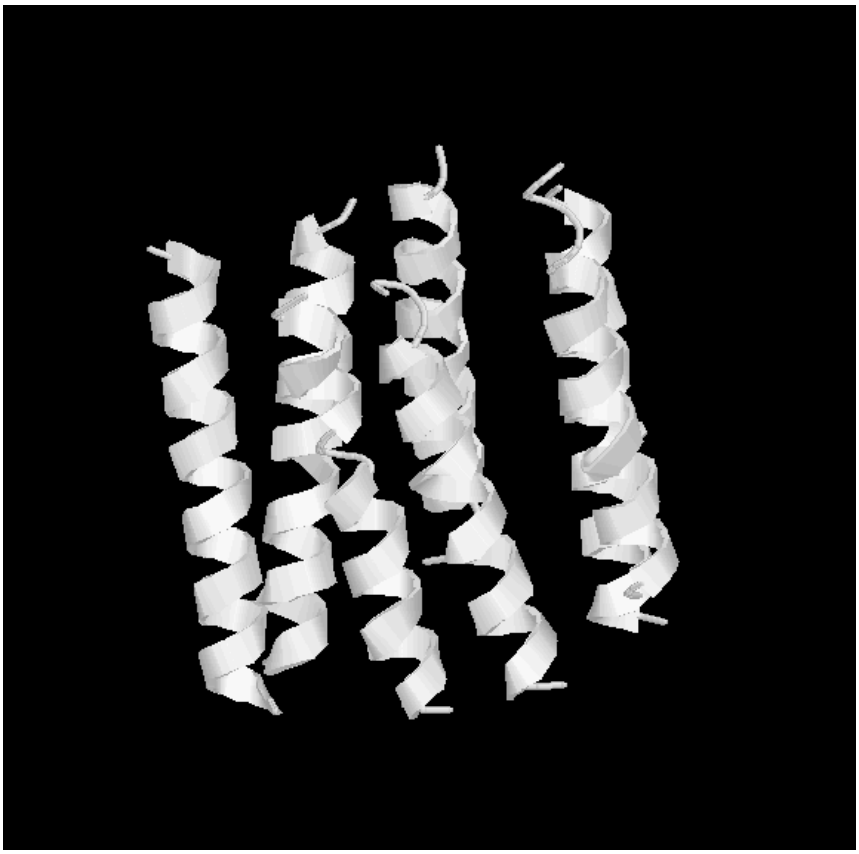


Figure 2.4: Predicted structure for seven helices based on human neuropeptide Y1 receptor

Several GPCR homology modeling servers are accessible on the World Wide Web. The Swiss Model 7TM server at `http://www.expasy.ch/swissmod/` allows the user to submit seven helix sequences and select a template structure. Figure 2.4 shows the results of modeling seven helix sequences, each 25 amino acids in length, on a template of the human neuropeptide Y1 receptor.

2.1.2 Five GPCR Families

According to the G-protein Coupled Receptor Database maintained at EMBL, the superfamily can be divided into five main families whose members share >= 20% sequence identity. (An alternate family classification scheme has been developed by Dr.

Family A	Receptors related to Rhodopsin and the beta2-adrenergic Receptors
Family B	Receptors related to the Calcitonin and PTH/PTHrP Receptors
Family C	Receptors related to the Metabotropic Receptors
Family D	Receptors related to the STE2 and STE3 pheromone Receptors
Family E	Receptors related to the cAMP Receptors

Table 2.1: G-protein coupled receptor families

Frank Kolakowski [58] and Dr. Ken Rice [46], using parsimony and other phylogenetic methods.) Family A (rhodopsin and beta2-adrenergic receptors), Family B (calcitonin and PTH/PTHrP receptors) and Family C (metabotropic receptors) are found in metazoan animals, with Family C only found in mammals. Family D (STE2 and STE3 pheromone receptors) is found in species of fungi and Family E (cAMP receptors) is found in slime mold [25]. Within the families, proteins are further divided into groups which bind common agents on the extracellular side of the membrane. The evolutionary relationship between groups is not known; they may have diverged from a common ancestor or be the result of *convergent evolution* [25], in which functional constraints lead to unrelated proteins from different organisms with the same design.

Because the EMBL and Kolakowski databases are not the same, it was important to confirm that they did not disagree on the labeling of protein sequences included in my training and test sets. Although I did not perform an exhaustive search, I took a random sampling of 50 sequences misclassified by the SVM, and checked their family label on both databases. For every case, the databases were in agreement. However, the Kolakowski database has split Family D into two families of pheremone receptors and labeled them Family D and Family E, and they have renamed EMBL's Family E to be Family F.

2.1.3 Cellular Signalling

GPCRs are organized in three basic domains: a *signal recognizing* domain on the extracellular side of the membrane, a *signal transmission* domain that traverses the cell membrane, and a *signal response and amplification* domain on the intracellular side of the membrane [10].

On the extracellular side, they receive a wide variety of agents: lipid analogues, amino acid derivatives, small peptides, as well as stimuli from light (photons), taste and odor (pheromones). The agent or *ligand* is docked in a binding pocket. For small molecular weight ligands, the docking is done in the transmembrane region of the GPCR. For larger ligands it also includes the extracellular domain [65]. Through a process not yet understood, but which may involve tiny movements in the transmembrane regions, GPCRs transmit a signal, based on the particular ligand, to the intracellular (cytoplasmic) side of the membrane.

Prior to receiving this signal, a G-protein is attached to the intracellular side of the membrane and is bound to the GPCR. Three regions appear to be critical to this binding: a sequence of eight amino acids in the GPCR N-terminus, a twelve amino acid sequence in the C-terminus of the third transmembrane loop, and parts of the second transmembrane loop. While specific G-proteins bind to specific GPCRs [57], it is believed impossible to determine these pairings on the basis of primary amino acid sequence. The interaction appears to depend on the whole tertiary structure of the GPCR [65].

When they receive a signal, G-proteins detach from the GPCR and bind to enzymatic effector proteins lodged in the membrane. The G-proteins function as amplifiers,

Effector/second messenger systems	Some Cellular activities
cGMP-PDE	conversion of light information into electrical nerve activity in rod cells color vision in cone cells
phospholipases	autocrine and paracrine regulation protein kinase C activation ion channel conductance neurotransmitter release smooth muscle contraction synthesis platelet activating factor
adenyl cyclases	gene transcription mitogenesis metabolism growth factors

Table 2.2: Effector/Second Messenger Systems in the Cell

inducing the effectors to produce cascades of secondary messenger molecules that activate other enzymes, creating a diverse range of physiological responses [10]. Effector/second messenger systems include retinal cyclic GMP phosphodiesterases (cGMP-PDE), ion channels (potassium, calcium), and several phospholipases and adenylyl cyclase subtypes. Table 2.2 shows a list of activities controlled by effector/second messenger systems in the cell [65].

Most G-proteins are *heterotrimers* with α, β, and γ components. The generally accepted model is that G-proteins are activated by binding GTP (guanine triphosphate) to their α chain and inactivated when the GTP is *hydrolized* to GDP (guanine diphosphate).

On binding GTP, the G-protein heterotrimer splits into an α subunit and a $\beta\gamma$ heterodimer, both capable of transmitting signals from the GPCR to different effector proteins. This creates a versatile system that allows the cell to respond to a wide variety of extra-cellular signals. In humans the multiple varieties of α, β, and γ subunits allow

Stimulus	GPCR-type	Effector	Physiological Response
Epinephrine	β-adrenergic receptor	adenylate cyclase	glycogen breakdown
Light	rhodopsin	c-GMP phosphdiesterase	visual excitation
IgE-antigen complexes	mast cell IgE receptor	phosphlipase C	histamine secretion in all allergic reactions
Acetylcholine	muscarinic receptor	potassium channel	slowing of pacemaker activity that controls the rate of the heartbeat

Table 2.3: Examples of Extracellular signals, GPCRs, Effector Proteins and Physiological Responses [10]

for hundreds of possible sub-unit combinations and consequently, hundreds of different kinds of G-proteins. Table 2.3, taken from the excellent text by Branden and Tooze [10], shows some examples of extracellular signals and the corresponding GPCR subfamily, intracellular effector protein, and physiological activity.

G-protein activation is followed by a *desensitization* process, during which the amplified signal is turned off by hydrolysis of GTP to GDP. Hydrolysis is a chemical decomposition in which a molecule of water splits a target molecule into new molecules which contain the elements of water. In this case, the resulting molecules are GDP and a single phosphate group.

Although the G-protein itself catalyzes the hydrolysis, it does so very slowly. Acceleration is accomplished by regulatory RGS molecules bound to the G-α subunit [10].

Proper functioning of this "switching system" is essential to the health of an organism. When the system breaks down, resulting diseases include retinal degeneration, precocious puberty, thyroid tumors, hpertension, congestive heart failure, and drug dependence and tolerance [17].

2.2 Support Vector Machines

2.2.1 Statistical Inference and Linear Threshold Functions

To understand support vector machine algorithms, it is useful to appreciate their place in the more general field of statistical learning theory. At the heart of this theory is the problem of *statistical inference*: *"Given a collection of empirical data originating from some functional dependency, infer this dependency."* [62]

Learning machine algorithms are implementations of statistical inference principles. Typically, the machine is presented with a set of *training examples* (x_i, y_i) where the x_i are real-world data instances and the y_i are the desired output [19]. We assume there is a mapping, or *target function* (possibly stochastic) between the data and the output that the machine will learn. The simplest case is known as pattern recognition or classification. In this case, the x_i are instances, each of which is a member of one of k classes. The y_i are labels indicating which class the instance belongs to.

For the two-class pattern recognition problem, $y_i = +1$ or $y_i = -1$. A training example (x_i, y_i) is called *positive* if $y_i = +1$ and negative otherwise. A *decision rule* is a function that takes an instance x and produces a predicted classification label $y = \pm 1$. If each instance x is given a vector $\mathbf{x}$, then the simplest decision rule is a linear threshold function:

$$g(\mathbf{x}) = sgn(< \mathbf{w}, \mathbf{x} > +b) \qquad \text{Linear Threshold Function}$$

where $< \mathbf{w}, \mathbf{x} >$ is the dotproduct of the vectors $\mathbf{w}$ and $\mathbf{x}$. As the machine learns the decision rule, it adjusts the free parameters in $g(\mathbf{x})$ so that $g(\mathbf{x}_i) = y_i$ for as many training examples as possible. (Other learning criteria may be used at this stage as well.)

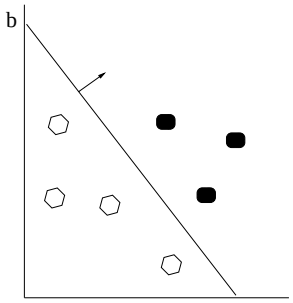


Figure 2.5: A one-dimensional hyperplane separates the classes of white disks and black disks in two dimensions. The weight vector is orthogonal to the hyperplane so that $< \mathbf{w}, \mathbf{x} > +b = 0$ for $\mathbf{x}$ on the hyperplane.

The free parameters are a vector of weights $\mathbf{w}$, and a threshold value b. By the end of the training phase, the parameters found by the machine can be used to apply the decision rule to new instances and correctly predict their class with (hopefully) a minimum of errors.

From a geometric point of view, the machine has constructed a *hyperplane* between the two classes of examples (see Figure 2.5). The weight vector lies orthogonal to this hyperplane so that $< \mathbf{w}, \mathbf{x} > +b = 0$ for $\mathbf{x}$ on the hyperplane. The problem of selecting such a hyperplane is an ill-posed one in that there are multiple possible solutions. For example, in the data set of Figure 2.5, the hyperplane $(\mathbf{w}, b)$ shown could be rotated a bit to the left or right and still correctly classify the disks.

A different kind of data set appears in Figure 2.6. No hyperplane can correctly classify these training examples in two dimensions. We say that such a data set is not *linearly separable* [13].

Each training example $(\mathbf{x}_i, y_i)$ is related to a separating hyperplane by a quantity called the *margin* [13]. The *functional margin* is the product of the example's label and the prediction made by the learning machine.

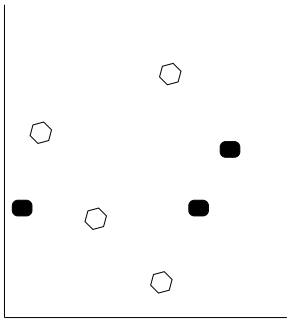


Figure 2.6: A data set that is not linearly separable. No hyperplane can correctly classify the training examples in two dimensions.

$$\gamma_i = y_i(< \mathbf{w}, \mathbf{x}_i > +b) \qquad \text{Functional Margin}$$

If $\gamma_i > 0$, the data point is correctly classified by the hyperplane because the label and the prediction have the same sign, i.e. either $y_i = 1$ and $(< \mathbf{w}, \mathbf{x}_i > +b) \geq 0$ or $y_i = -1$ and $(< \mathbf{w}, \mathbf{x}_i > +b) < 0$.

When the weight vector $\mathbf{w}$ of the functional margin is normalized ($\mathbf{w} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$), we get a second quantity known as the *geometric margin* which measures the distance of the data points from the hyperplane in Euclidean space. The geometric margin of a data point $\mathbf{x}_i$ is shown in Figure 2.7.

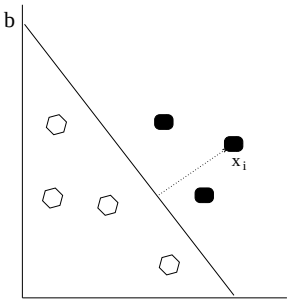


Figure 2.7: The dotted line is the geometric margin of example $\mathbf{x}_i$.

The expression *margin of a training set* is used to refer to the maximum geometric

margin over all possible hyperplanes and appears in Figure 2.8 [13].

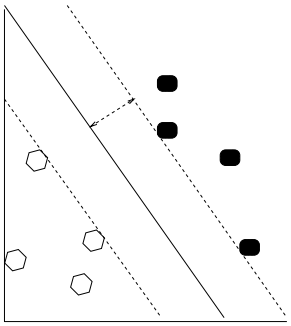


Figure 2.8: The dotted line is the margin of the training set shown.

Unlike the hyperplane in Figure 2.5, the hyperplane defined by the maximum geometric margin is unique. It is known as a *maximal margin hyperplane* [13].

One of the simplest algorithms for learning a linear threshold function $g(\mathbf{x})$ from a set of labeled, linearly separable, training examples is the *perceptron algorithm* [13]. The technique is to intialize $\mathbf{w}$ and b to zero, and then loop through the training examples. On each iteration of the loop, the value of the functional margin

$$y_i(< \mathbf{w}, \mathbf{x}_i > +b)$$

is computed. As described above, if the functional margin is less than zero, the example is misclassified by the linear threshold function whose parameters are the current values of $(\mathbf{w}, b)$. In this situation, the algorithm updates $(\mathbf{w}, b)$:

$$\mathbf{w}_{new} \leftarrow \mathbf{w}_{old} + \eta y_i \mathbf{x}_i \qquad \text{Update Rules} \tag{2.1}$$

$$b_{new} \leftarrow b_{old} + \eta y_i R^2 \tag{2.2}$$

where η is a small fractional value called the *learning rate*, and R is the norm of the largest instance vector $\|\mathbf{x}\|$.

The loop continues until the margin of each training example is non-negative, so that all examples are classified correctly. The perceptron convergence theorem guarantees that this will occur for any linearly separable training set [48]. Because the initial weight vector $\mathbf{w}$ is zero, by the update rule for $\mathbf{w}$ shown in Equation 2.1, when the loop exits, $\mathbf{w}$ will be in the form of:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{2.3}$$

By a simple substitution, $g(\mathbf{x})$ can now be written as [13]:

$$\begin{aligned} g(\mathbf{x}) = &\ \text{sgn}\ (< \mathbf{w},\ \mathbf{x} > + b) \\ = &\ \text{sgn}\ \left(\left\langle \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,\ \mathbf{x} \right\rangle + b \right) \\ = &\ \text{sgn}\ \left(\sum_{i=1}^{l} \alpha_i y_i < \mathbf{x}_i,\ \mathbf{x} > + b \right) \end{aligned} \tag{2.4}$$

The update rule becomes, if $y_i \left(\sum_{i=1}^{l} \alpha_i y_i < \mathbf{x}_i,\ \mathbf{x} > + b \right) < 0$:

$$\alpha_i \leftarrow \alpha_i + 1 \tag{2.5}$$

$$b \leftarrow b + y_i R^2 \tag{2.6}$$

The learning rate η has been dropped because now it only changes the scaling of the hyperplane. For more discussion on this point, see Cristianini and Shawe-Taylor, Chapter 2

[13]. The equivalence of (2.5) and (2.1) can be seen by the following substitution:

$$\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \leftarrow \sum_{j=1}^{l} (\alpha_j + 1) y_j \mathbf{x}_j$$

$$\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \leftarrow \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j + y_j \mathbf{x}_j$$

$$\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \leftarrow \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j + y_i \mathbf{x}_i$$

$$\mathbf{w}_{new} \leftarrow \mathbf{w}_{old} + y_i \mathbf{x}_i \tag{2.7}$$

Notice that in the dual formulation (2.4), the training instances $\mathbf{x}_i$ do not appear individually in either the training loop or the linear threshold function. They always appear in pairs inside a dot product $< \mathbf{x}_i, \mathbf{x} >$. This property will become very important when we consider kernel functions in Section 2.2.3.

2.2.2 Empirical Risk Minimization and Structural Risk Minimization

In order for a learning machine to do useful work, it must be able to make good predictions for new, unlabeled instances. Much of the theoretical work on learning algorithms involves mathematically bounding the number of classification errors a machine will make on new instances, with the assumption that they are generated independently and identically (i.i.d) and drawn from the same unknown but fixed distribution as the training examples [13].

The error of a classification function g in a trained learning machine on distribution $\mathcal{D}$ is the probability that an example randomly generated from $\mathcal{D}$ is misclassified [13]. This quantity is known as the *risk functional*.

$$R(g) = \mathcal{D}\{(\mathbf{x}, y) : g(\mathbf{x}) \neq y\} \qquad \text{Risk Functional} \tag{2.8}$$

A principle of statistical inference known as *empirical risk minimization (ERM)* asserts that the decision rule parameters which minimize error on the training set will minimize the risk functional, i.e. make the fewest mistakes on new instances. Empirical risk is calculated as:

$$R_{emp}(g) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - g(\mathbf{x}_i, \mathbf{w})| \qquad \text{Empirical Risk of Weight Vector w} \tag{2.9}$$

where l is the size of the training set. In this simple classification setting, eq. (2.9) is proportional to the number of disagreements between the prediction made by the decision rule $g(\mathbf{x_i}, \mathbf{w})$ and the label, or true class of the example, y_i on the training examples. Intuition suggests that the size of the training set is important to minimizing the risk functional eq. (2.8) and that a very large training set will result in a smaller risk functional. If the training set is large, the learning machine has a large amount of information available to encode in its decision rule. Furthermore, if a new instance is unlike anything previously seen in a large training set, it must have low probability in $\mathcal{D}$ and will not contribute much to classification error [3].

In the 1970's, the ERM principle was refined by Vapnik [61]. He asserted that, as the number of training examples grows large, the parameters that minimize the empirical risk do not necessarily converge to the best solution (the parameters that would result if we had an infinite number of training examples) [12]. Therefore, the decision rule that minimizes training error on a finite number of examples may not *generalize*, i.e. minimize error on new instances. Vapnik introduced the idea that a finite *VC dimension* is necessary for a learning algorithm to generalize well on new instances.

The VC-dimension is an upper bound on the number of possible decision rules

that can be produced by a learning machine.

Let F be a set of binary-valued functions defined on a set X and let $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m})$ be a sample of m instances from X. $B_F(\mathbf{x})$ is defined as the number of ways $\{\mathbf{x}_1 \ldots \mathbf{x}_m\}$ can be classified by all the functions $g \in F$. We are counting distinct vectors of the form [3]:

$$(g(\mathbf{x_1}), g(\mathbf{x_2}), \cdots, g(\mathbf{x_m})) \tag{2.10}$$

A few examples of such vectors are:

$$(g_1(\mathbf{x_1}), g_1(\mathbf{x_2}), \cdots, g_1(\mathbf{x_m})) = (-1, 1, \cdots, -1)$$

and

$$(g_2(\mathbf{x_1}), g_2(\mathbf{x_2}), \cdots, g_2(\mathbf{x_m})) = (1, -1, \cdots, -1)$$

We can observe that even if $|F|$ is infinite, the subset of functions $F_{\mathbf{x}}$ on the domain of $\{\mathbf{x}_1 \ldots \mathbf{x}_m\}$ is finite, with cardinality $B_F(\mathbf{x})$. For a sample $\mathbf{x}$ of m examples:

$$B_F(\mathbf{x}) \leq 2^m \tag{2.11}$$

[3] because all vectors in eq. (2.10) are binary sequences of length m [13] and there are only 2^m such sequences.

Next we introduce and briefly describe three related concepts crucial to VC theory.

The growth function $B_F(m)$ is defined to be:

$$B_F(m) = \max\{B_F(\mathbf{x}) : \mathbf{x} \in \mathbf{X}^m\} \tag{2.12}$$

[3] and measures how many possible labellings of m examples can be correctly classified by the functions of F.

A set of functions F is said to *shatter* a sample of m examples $\mathbf{x}$ if F can successfully classify the m examples for all 2^m possible labellings of the examples.

The *VC-dimension* of F is the maximum length of a sample that can be shattered by F. If the maximum does not exist, the VC-dimension of F is infinite [3]. The expressiveness and power of a set of functions F are measured by its VC-dimension.

These concepts may be clarified by a simple example presented in the SVM tutorial written by Burges [12]. If the F are the set of linear threshold functions in two dimensions, the VC dimension of F is three. Each function in F can be represented by an oriented straight line. If we take three points in a two-dimensional space, we can orient the lines to classify the points correctly, no matter how the points are labelled (see Figure 2.10). The VC dimension of F cannot be four because for any four points, there is a labelling of these points that no line can separate. One case of this is shown in Figure 2.10.

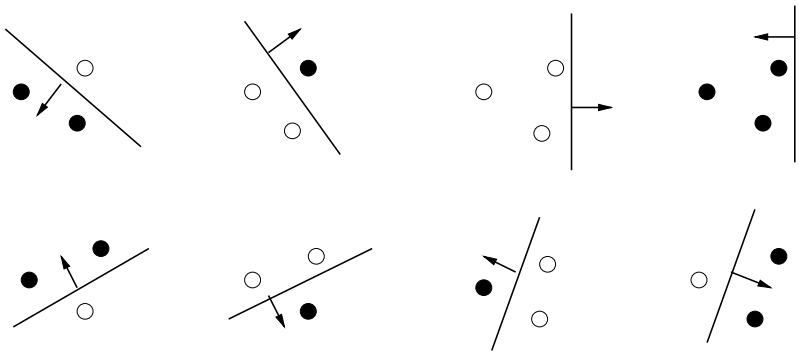


Figure 2.9: A set of lines shattering three points [12]

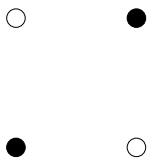


Figure 2.10: Four labeled points that can't be separated by an oriented line in 2-D space

The VC dimension provides an upper bound on the growth function, which in

turn bounds the number of functions available to a particular learning machine. Vapnik's *structural risk minimization (SRM)* theory [62, 50] describes a *structure* of nested subsets of functions,

$$F_1 \subset F_2 \subset \cdots \subset F_n \subset \cdots ,$$

whose VC-dimensions are:

$$h_1 \leq h_2 \leq \cdots \leq h_n \leq \cdots$$

The *structural risk minimization (SRM)* principle [62] minimizes the risk functional by taking a structure of function subsets and choosing the function g from the subset F_i for which the right hand side of eq. (2.13) is minimal [50]. The main result about the SRM is given by eq. (2.13). The two terms in (2.13) are the ERM risk functional which measures training error, and a second *confidence* term proportional to the VC dimension, and inversely proportional to the number of training examples l. (This is intuitively what we expect since a large training set and a small VC dimension are likely to reduce the number of generalization errors.)

Given a random training set of size l, with probability $1 - \delta$:

$$R(g) \leq R_{emp}(g) + sqrt\left\{\frac{h(log(\frac{2l}{h}) + 1) - log(\frac{\delta}{4})}{l}\right\} \qquad \text{Structural Risk Minimization} \tag{2.13}$$

Here h is the VC dimension, g is the decision function chosen by SRM and δ is an arbitrary confidence level, $0 < \delta < 1$.

$R_{emp}(g)$ and the confidence term are in conflict. A complex learning machine with a large F_i set of decision functions and a large VC dimension will have smaller training error; a simpler learning machine with small F_i will have smaller VC dimension and a

smaller confidence term. Structural Risk minimization requires a tradeoff as shown in Figure 2.11 [50].

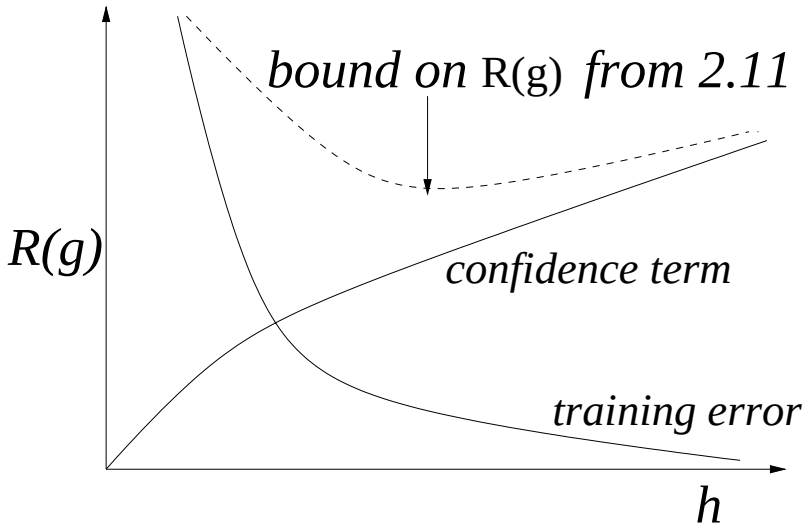


Figure 2.11: Structural risk minimization requires a tradeoff between the training error and the confidence term. The horizontal axis is the VC dimension h and the vertical axis is the risk $R(g)$. The bound on $R(g)$ is shown as a sum of training error and confidence term.[50]

SVM algorithms use structural risk minimization on the structure of separating hyperplanes (normalized linear threshold functions). Each nested subset of separating hyperplanes has a VC dimension h that can be bounded as follows.

If we take R, the radius of the smallest ball B_R that contains points $\mathbf{x_1} \cdots \mathbf{x_r}$, the set of *canonical* (normalized) hyperplane decision functions $\{g_{\mathbf{w},b} : \|\mathbf{w}\| \leq A\}$, where $g_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}\,(< \mathbf{w}, \mathbf{x} > +b)$, has a VC-dimension h such that:

$$h < R^2 A^2 + 1$$

[62, 50]. This result can be understood by considering that $\|\mathbf{w}\|$ and the margin are inversely proportional eq. (2.19). By selecting a subset of decision functions where $\|\mathbf{w}\|$ is appropriately bounded, we have the flexibility to do the following [50]:

- A small VC-dimension can be obtained by setting a large lower bound on the margin (small A).
- For a complex class of problems with many possible labelings of the training data, we can set A to be large and produce a small margin.
- By the SRM principle, high generalization ability requires both small training error and small VC-dimension. This can be achieved by hyperplane decision functions that maximize the margin in addition to accurately separating the training examples.

2.2.3 Higher Dimensions

The pattern recognition decision rules $g(\mathbf{x})$ seen so far are linear functions of the data vectors $\mathbf{x}$ and they work as long as the data sample being classified is linearly separable in the data input space. In practice, this is frequently not the case, and we have the situation shown in Figure 2.6. One of the key insights behind support vector machines is that such a data set can be mapped into a higher dimensional space, where, in many cases, it is linearly separable. The decision rule $g(\mathbf{x})$ to be learned will not be linear in the orignal *input space* of the examples, but it will be linear with respect to the high dimensional space.

Consider the four points shown in Figure 2.10. Although it is not possible to construct a hyperplane that separates the white and black disks in a two-dimensional space, they can be separated in three dimensional space if they are appropriately positioned, even though they would look the same when projected back to a two-dimensional space. Figure 2.12 shows the same four disks from a front and side view. A two-dimensional

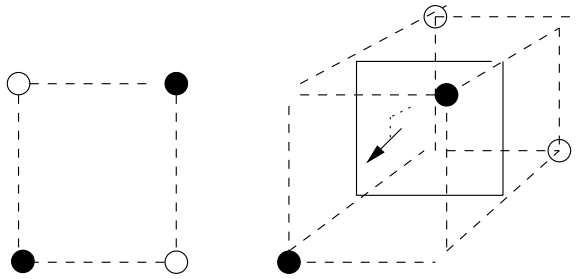


Figure 2.12: Front and side view of four white and black disks in three dimensions, where white and black can be separated by a two-dimensional hyperplane

hyperplane can separate the classes when the points are in three dimensions.

However, there are problems with this approach.

- Working directly in high dimensional space is computationally expensive.
- Although the mapping shown in the toy problem of Figure 2.12 is simple, discovering how to map examples into high dimensional space for real-world problems is very difficult.

In support vector learning the solution is as follows. The examples are mapped to the higher dimensional or *feature space* F with a mapping $\Phi : \mathbf{R}^N \rightarrow F$, but the map is never computed explicitly. The SVM decision rule is updated in a dual form, analogous to the dual form of the perceptron algorithm eq. (2.4). As we saw in the perceptron case, when the α_i parameters were learned inside the training loop, the data (instances) only appeared as part of a dot product $< \mathbf{x_i}, \mathbf{x_j} >$. The key idea is that this learning can take place in feature space, and the instances need only appear as $< \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) >$. Therefore, the mapping into high dimensional space does not have to be computed explicitly. To train the SVM in feature space, we will only need to compute the similarity of the training instances in feature space.

$K(x,y) =< x,\ y >$	Mercer Kernel
$K(x,y) =< x,\ \sigma y >$	Covary Kernel
$K(x,y) = tanh(\kappa < x,\ y > +\theta)$	Sigmoid Kernel
$K(x,y) = e^{\frac{-\|x-y\|^2}{\sigma^2}}$	Radial Basis Kernel
$K(x,y) = (< x,\ y > +1)^d$	Polynomial Kernel

Table 2.4: Some popular kernel functions used in support vector learning.

To do this, we use a "trick" called a *kernel function* K, which encapsulates the similarity between $\Phi(x_i)$ and $\Phi(x_j)$.

$$K(\mathbf{x_i}, \mathbf{x_j}) =< \Phi(x_i),\ \Phi(x_j) > \tag{2.14}$$

By using a kernel function, we avoid the problem of finding $\mathbf{x} \rightarrow \Phi(\mathbf{x})$ and also greatly simplify the computational complexity of the learning algorithm.

By Mercer's Theorem, many kinds of kernel functions can be constructed that have the properties necessary for a mapping from data input space into a potentially infinite-dimensional space. For details on the derivation, see Cristianini and Shawe-Taylor, Chapter 3 [13]. Support vector machines use a variety of these kernel functions: radial basis kernels, polynomial kernels, and sigmoidal kernels [19]. Several kernel functions are shown in Table 2.4. For the work in this thesis, I used a kernel specifically designed for biosequence analysis, the *Fisher kernel*, described in Section 2.2.5.

2.2.4 Optimal Hyperplane

Given a set of example vectors $\mathbf{x}$, the general form of a hyperplane capable of linearly separating the examples is $< \mathbf{w},\ \mathbf{x} > +b = 0$. Vapnik introduced the concept of an *optimal hyperplane* [62]. This is the same as the maximal margin hyperplane shown in Figure 2.8. A way to construct the optimal hyperplane is to find the pair $(\mathbf{w}, b)$ that

satisfies these constraints for all $\mathbf{x}$ in the training set:

$$\begin{aligned} &< \mathbf{w},\, \mathbf{x}_i > +b \geq 1, && \text{if } y_i = 1, \\ &< \mathbf{w},\, \mathbf{x}_i > +b \leq -1, && \text{if } y_i = -1 \end{aligned} \tag{2.15}$$

and the vector $\mathbf{w}$ with the smallest norm

$$\|\mathbf{w}\| =< \mathbf{w},\, \mathbf{w} > \tag{2.16}$$

The constraints eq. (2.15) can be written as

$$y_i < \mathbf{w},\, \mathbf{x}_i > +b \geq 1 \qquad i = 1, \cdots, l \tag{2.17}$$

Vapnik claims the $\mathbf{w}$ that minimizes eq. (2.16) under the constraints eq. (2.17) is related to the vector that forms the optimal hyperplane by

$$\mathbf{w}' = \frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{2.18}$$

and that the margin between the optimal hyperplane and the separated examples is

$$\gamma(\mathbf{w}') = \frac{1}{\|\mathbf{w}\|} \tag{2.19}$$

The proof (by contradiction) is found in his "Statistical Learning Theory" [62].

We can now express the problem of finding the optimal hyperplane for a set of training examples as the minimization of eq. (2.16) subject to eq. (2.17). This is a quadratic optimization problem since $< \mathbf{w},\, \mathbf{w} >$ is quadratic in form.

Using the technique of *Lagrange multipliers*, a Langrangian function can be written for this problem

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} < \mathbf{w},\, \mathbf{w} > \sum_{i=1}^{l} \alpha_i (y_i (< \mathbf{w},\, \mathbf{x_i} > +b) - 1) \tag{2.20}$$

The solution is a saddle point where

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0$$

At the saddle point, $\mathbf{w}$ and b are minimized and the α_i are maximized.

We show below how the α_i can be found by considering a dual formulation. By taking the derivative of L with respect to b and $\mathbf{w}$ and substituting, we get the following equations:

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.21}$$

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x_i} \tag{2.22}$$

Notice that the weight vector is expressed in terms of a sum of training examples, each with a weight α_i and a sign (its y_i label). Using eq. (2.21) and eq. (2.22), we can now

derive a dual form of analagous to the dual form of the perceptron algorithm eq. (2.4).

$$
\begin{aligned}
L(\mathbf{w}, b, \alpha) &= \frac{1}{2} < \mathbf{w},\ \mathbf{w} > \sum_{i=1}^{l} \alpha_i (y_i,\ (< \mathbf{w},\ \mathbf{x_i} > + b) - 1) \\
&= \frac{1}{2} \left\langle \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \right\rangle - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\left\langle \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j,\ \mathbf{x}_i \right\rangle + b \right) - 1 \right] \\
&= \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i,\ \mathbf{x_j} > - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\sum_{j=1}^{l} \alpha_j y_j < \mathbf{x}_i,\ \mathbf{x}_j > + b \right) - 1 \right] \\
&= \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i,\ \mathbf{x_j} > - \sum_{i=1}^{l} \alpha_i y_i \left(\sum_{j=1}^{l} \alpha_j y_j < \mathbf{x}_i,\ \mathbf{x}_j > + b \right) + \sum_{i=1}^{l} \alpha_i \\
&= \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i,\ \mathbf{x_j} > - \sum_{i=1}^{l} \alpha_i y_i \sum_{j=1}^{l} \alpha_j y_j < \mathbf{x}_i,\ \mathbf{x}_j > - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i,\ \mathbf{x_j} > \qquad (2.23)
\end{aligned}
$$

The simplification that produces the last line is a result of the fact that by Equation (2.21):

$$\sum_{i=1}^{l} \alpha_i y_i b = b \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (2.24)$$

The result is that the α_i can be calculated by a maximization of eq. (2.23) subject to the constraints

$$\alpha_i \geq 0, \qquad i = 1, \cdots, l \qquad \text{and} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

In the dual form, the decision rule for the support vector machine is:

$$g(\mathbf{x}) = sgn \left(\sum_{i=1}^{l} y_i \alpha_i < \mathbf{x},\ \mathbf{x_i} > + b \right) \qquad \text{SVM Decision Rule} \qquad (2.25)$$

This is the form of the decision rule that is used in practice by the SVM. As described in Section 2.2.3, if the data is not linearly separable in input space, the simple

dot product between instances $< \mathbf{x}, \mathbf{x_i} >$ can be replaced with another choice of kernel function.

In the experiments described in this thesis, the threshold parameter b was set to zero.

$$g(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i})\right) \qquad \text{SVM Decision Rule with Kernel and b=0} \quad (2.26)$$

The argument of the *sgn* function in eq. (2.26) is called the *discriminant*.

$$g^{'}(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) \qquad \text{Discriminant} \quad (2.27)$$

The discriminant can be used to compute a real-valued score for each test instance, rather than a simple binary prediction $\{-1, 1\}$.

There are many different ways to find the values of the alphas, including the Bunch-Kaufman algorithm, the conjugate gradient algorithm, chunking, and SMO [54]. I used a steepest gradient ascent algorithm developed by Tommi Jaakola [29]. This is a constrained maximization procedure in which the α values are initialized to 0.5 and then iteratively updated as:

$$\alpha_{i_{new}} \leftarrow \frac{1 - y_i \left(\sum_{i=1}^{l} y_j \alpha_{j_{old}} K(\mathbf{x}, \mathbf{x_j})\right) + \alpha_{i_{old}} K(\mathbf{x}_i, \mathbf{x}_i)}{K(\mathbf{x}_i, \mathbf{x}_i)} \quad (2.28)$$

Details of the algorithm, including how to terminate the iterations and heuristics used to constrain the α values to $0 \leq \alpha \leq 1$, can be found in the ISMB paper by Jaakola, Diekhans and Haussler [28].

2.2.5 Fisher Kernel

All the pattern recognition algorithms described in this thesis associate a fixed-length vector $\mathbf{x}$ with the instances X to be classified. We now discuss some specifics as to

how an appropriate vector can be selected for a protein sequence. The Jaakola, Diekhans and Haussler paper [27] was the first to apply support vector learning to the problem of classifying protein sequences. Their method begins with a generative probabilistic model of a sequence, the hidden Markov model, and extracts from it a fixed-length vector of features known as a *Fisher score vector*.

On their own, generative probabilistic models, like the hidden Markov model, have proven to be a very useful tool for classifying protein sequences. To understand Fisher score vectors, it is necessary to know something about hidden Markov models, described here in the context of protein sequence analysis.

The HMM is a statistical model of a collection of protein sequences, usually members of a family or superfamily of interest. HMM learning is an iterative algorithm in which the model parameters are adjusted to reflect the frequencies of amino acids in a training set of positive examples. The resulting model is composed of a series of *match states* that correspond to positions in the training sequences which describe some degree of conserved primary structure. Each match state has a unique probability distribution over the amino acid alphabet that reflects the distribution in the training set at that position. The model also contains transition probabilities between adjacent match states. The term "generative" is used because, after an HMM has been built, we can consider the probability that any particular protein sequence was "generated" by this model.

The HMM can be used as a classifier, for the family of sequences modeled, by computing the probability that a protein sequence X is generated by a model M. If $P(X|M)$ is high, it is likely that X belongs in the family, otherwise not.

We now consider how to compute the probability $P(X|M)$ that a particular

protein sequence X was generated by a model M. This involves looking at the probability distributions in all of the match states of the model, and at the transition probabilities between states. [1]

$$P(X|M) = P(X|\theta,\tau) = \sum_{s_1,\cdots,s_n} \prod_i P(x_i|s_i,\theta)P(s_i|s_{i-1},\tau) = \sum_{s_1,\cdots,s_n} \prod_i \theta_{x_i|s_i}\tau_{x_i|s_i} \quad (2.29)$$

Here, θ represents the probability distributions in the match states of the model and τ are the transition probabilities. $P(x_i|s_i,\theta)$ is the probability of amino acid x_i in match state s_i, given the probability distributions in M. $P(s_i|s_{i-1},\tau)$ is the probability of landing in match state s_i after being in match state s_{i-1}, while in model M.

The "H" in HMM refers to the fact that the sequence of match states through the model is hidden. This unknown sequence of n states is specified by $\{s_1,\cdots,s_n\}$, and it is necessary to sum over all possible such n-length state sequences. The individual probabilities $P(x_i|s_i,\theta)$ and $P(s_i|s_{i-1},\tau)$ are assumed to be independent, so the probability of a protein sequence, given a particular state sequence, is the product of a chain of these individual probabilities. For computational efficiency, we work with the log of eq. (2.29), also known as the sequence's *log likelihood score*.

$$\log\ (P(X|\theta,\tau)) \quad (2.30)$$

Note that this method does not allow for two sequences to be directly compared to each other. They can only be compared with respect to their relationship to the model. Some potential advantages of support vector classification over HMMs are

[1] The SAM-T99 hidden Markov models used in my experiments actually have two kinds of "amino-acid emitting" states: match states and *insert* states. The insert states are used to handle amino acids which are not part of the conserved primary structure detected in the training set sequences. Details can be found in the biosequence analysis book by Durbin, Eddy, Krogh and Mitchison [16]. Jaakkola lumps match and insert states together in his development of Fisher score vectors [28].

- The similarity of two sequences can be computed directly.
- Because the SVM is trained on both positive and negative examples, it is indirectly able to acquire more information about the class of positive examples.

In their 1998 paper [29], Jaakkola and Haussler observe that to compare two sequences directly, we need to quantify the difference between model M generating sequence X and model M generating sequence Y. This can be done in the *gradient space* of M, the gradient of the log likelihood with respect to the parameters of the model:

$$\frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \log P(X|\theta,\tau) \tag{2.31}$$

where $\theta_{\tilde{x},\tilde{s}}$ makes explicit that each of the individual parameters composing θ gives the probability of an amino acid $\tilde{x}$ in state $\tilde{s}$. With respect to an individual parameter, the gradient of the log likelihood represents the contribution of that parameter in the process of generating the sequence [29].

In an Appendix to the ISMB paper [28], Jaakkola proves that the partial derivative (2.31) is equivalent to

$$\frac{\xi(\tilde{x},\tilde{s})}{\theta_{\tilde{x},\tilde{s}}} - \xi(\tilde{s}) \tag{2.32}$$

where

$$\xi(\tilde{s}) = \sum_{x'} \xi(x', \tilde{s}) \tag{2.33}$$

and

$$\xi(\tilde{x},\tilde{s}) = \sum_{k} E\{\delta_{s_k,\tilde{s}}\delta_{x_k,\tilde{x}}|X,\theta,\tau\} \tag{2.34}$$

In eq. (2.34), the quantity $\xi(\tilde{x}, \tilde{s})$ is the expected value of the product of two Kronecker δ functions of amino acid $\tilde{x}$ and state $\tilde{s}$, taken with respect to a random hidden state sequence of the HMM. The Kronecker δ functions evaluate to one when $s_k = \tilde{s}$ and $x_k = \tilde{x}$, so the only contributions to the sum are for k such that $x_k = \tilde{x}$ and $s_k = \tilde{s}$. This quantity can be calculated by the forward-backward algorithm [16]. It is the same quantity that is used to update the parameters of the HMM during training [16].

The identity in eq. (2.33) is the sum of the expected values of all twenty amino acids, with respect to a particular state $\tilde{s}$ in M. In eq. (2.32), the denominator of the first term is a single instance of the probability $P(\tilde{x}|\tilde{s})$ of amino acid $\tilde{x}$ in state $\tilde{s}$, a parameter of the model.

By using eq. (2.32), we can compute a fixed length *Fisher score vector* for any protein sequence from the parameters of an HMM and use it as an intermediate representation between a sequence and an HMM. The components of the Fisher score vector are indexed by amino acid and state (x, s) and their values are computed by eq. (2.32) [29].

In my experiments, as in those described in the ISMB paper [28], enhanced Fisher score vectors were computed using a set of pre-calculated amino acid distributions known as *mixture priors*. A mixture decomposes the probability of amino acid x in state s into l components.

$$\theta_{x|s} = \sum_{l} c_{l|s} \theta^{(l)}_{x|s} \tag{2.35}$$

The quantities $\theta^{(l)}_{x|s}$ are probabilities that amino acid x in state s is in a given subclass, i.e. small, aromatic, polar, positively charged, large, non-polar, hydrophilic, hydrophobic, and

so forth. The $c_{l|s}$ are mixture coefficients which weight the subclasses and sum to one.

$$\sum_{l} c_{l|s} = 1 \tag{2.36}$$

Both the mixture coefficients and components have been estimated by studying large databases of protein sequences and observing which amino acid distributions are likely to occur at a given position in a protein. The University of California, Santa Cruz Computational Biology group maintains a library of these mixtures [52].

The Fisher score vector for sequence X given model M, has l components per match state in M, and each component represents a subclass of amino acids. In my experiments, a mixture of estimated frequencies for nine subclasses of amino acids, *uprior.9comp* written by Kevin Karplus, was used to compute the components. This mixture is shown in Table 2.5.

In the Fisher score vector, the feature for component l in match state s is given by:

$$f_{l,s}(x) = \frac{\partial}{\partial_{c_{l,s}}} \log P(X|\theta, c, \tau) = \sum_{x} \xi(x,s) \left[\frac{\theta^{(l)}_{x|s}}{\theta_{x|s}} - 1 \right] \tag{2.37}$$

The derivation of the right hand side is analogous to obtaining eq. (2.32). The posterior expected value $\xi(x,s)$ of amino acid x in state s is given by eq. (2.34). It is multiplied by the ratio of the probability $\theta^{(l)}_{x|s}$ that amino acid x is in subclass l and the probability $\theta_{x|s}$ of amino acid x in state s (minus an additional constant term). The resulting quantity is summed over all twenty amino acids x.

When mixture priors are used, the components of the Fisher score vector are indexed by mixture component and state (c,s) and their values are computed by eq. (2.37).

uprior.9comp									
	uprior9.0	uprior9.1	uprior9.2	uprior9.3	uprior9.4	uprior9.5	uprior9.6	uprior9.7	uprior9.8
c	0.182962	0.057607	0.089823	0.079297	0.083183	0.091122	0.115962	0.06604	0.234006
$\|\theta\|$	1.18065	1.35583	6.66436	2.08141	2.08101	2.56819	1.76606	4.98468	0.0995
A	0.270671	0.021465	0.561459	0.070143	0.041103	0.115607	0.093461	0.452171	0.005193
C	0.039848	0.0103	0.045448	0.01114	0.014794	0.037381	0.004737	0.114613	0.004039
D	0.017576	0.011741	0.438366	0.019479	0.00561	0.012414	0.387252	0.06246	0.006722
E	0.016415	0.010883	0.764167	0.094657	0.010216	0.018179	0.347841	0.115702	0.006121
F	0.014268	0.385651	0.087364	0.013162	0.153602	0.051778	0.010822	0.284246	0.003468
G	0.131916	0.016416	0.259114	0.048038	0.007797	0.017255	0.105877	0.140204	0.016931
H	0.012391	0.076196	0.21494	0.077	0.007175	0.004911	0.049776	0.100358	0.003647
I	0.022599	0.035329	0.145928	0.032939	0.299635	0.796882	0.014963	0.55023	0.002184
K	0.020358	0.013921	0.762204	0.576639	0.010849	0.017074	0.094276	0.143995	0.005019
L	0.030727	0.093517	0.24732	0.072293	0.999446	0.285858	0.027761	0.700649	0.00599
M	0.015315	0.022034	0.118662	0.02824	0.210189	0.075811	0.01004	0.27658	0.001473
N	0.048298	0.028593	0.441564	0.080372	0.006127	0.014548	0.187869	0.118569	0.004158
P	0.053803	0.013086	0.174822	0.037661	0.013021	0.015092	0.050018	0.09747	0.009055
Q	0.020662	0.023011	0.53084	0.185037	0.019798	0.011382	0.110039	0.126673	0.00363
R	0.023612	0.018866	0.465529	0.506783	0.014509	0.012696	0.038668	0.143634	0.006583
S	0.216147	0.029156	0.583402	0.073732	0.012049	0.027535	0.119471	0.278983	0.003712
T	0.147226	0.018153	0.445586	0.071587	0.035799	0.088333	0.065802	0.358482	0.00369
V	0.65438	0.0361	0.22705	0.042532	0.180085	0.94434	0.02543	0.66175	0.002967
W	0.003758	0.07177	0.02951	0.011254	0.012744	0.004373	0.003215	0.061533	0.002772
Y	0.009621	0.419641	0.12109	0.028723	0.026466	0.016741	0.018742	0.199373	0.002686

Table 2.5: Parameters of the nine component mixture *uprior.9comp*, developed by Kevin Karplus. The subclasses numbered uprior9.0–uprior9.8 model distributions of small, aromatic, polar, positively charged, large non-polar/aliphatic, isoleucine/valine pairs, hydrophilic, hydrophobic, and highly conserved amino acids. The first row shows the mixture coefficient c for each component. The second row gives the sum of the twenty amino acid frequencies θ_1–θ_{20} listed for the subclass.

As in the ISMB paper [28], I did not use the HMM transition probabilities τ when calculating Fisher score vectors. This reduces the computational complexity of the calculation, and Diekhans reports that the transition probabilities do not improve SVM classification [15].

Fisher score vectors make it is possible to directly compare two protein sequences and to derive kernel functions on protein sequences. The final kernel derived in the paper by Jaakkola and Haussler [29], which I also use in this work, is:

$$K(X,Y) = \exp\left(\frac{-\sum_{l,s}\left(f_{l,s}(X) - f_{l,s}(Y)\right)^2}{2\sigma^2}\right) \tag{2.38}$$

where σ is a width parameter determined as described in the Jaakkola and Haussler paper [29], and X and Y are two protein sequences being compared. For further discussion of *Fisher kernels*, see the Jaakkola and Haussler paper [29].

Chapter 3

Methods

The purpose of this thesis was to find out how well a Fisher kernel support vector machine classifier could handle two kinds of problems:

- Recognizing protein sequences belonging to the GPCR superfamily.
- Recognizing GPCR protein sequences with a specific function, such as binding a histamine molecule.

To get a perspective on the SVM's performance, it was necessary to compare several state-of-the-art methods for identifying protein sequences. For the superfamily classification problem, I repeated each experiment six times, using:

- BLAST scoring in conjunction with the family pairwise search algorithm [20] (see Section 3.4.1)
- Smith-Waterman scoring in conjunction with the family pairwise search algorithm [53] (see Section 3.4.2)

- SAM-T98 HMM
- SAM-T99 HMM
- SVM with Fisher score vectors from SAM-T98 HMM
- SVM with Fisher score vectors from SAM-T99 HMM

The same data sets were used for each method. However, one unavoidable difference is that the SVM is trained on both positive and negative examples, while the BLAST FPS, Smith-Waterman FPS, SAM-T98, and SAM-T99 methods use only positive training examples.

Because all experiments done with SAM-T98 were more successful when SAM-T99 was used, I only report SAM-T99 results in this thesis.

For the subfamily classification problem, I repeated each experiment three times using:

- BLAST/family pairwise search
- SAM-T99 HMM
- SVM with Fisher score vectors from SAM-T99 HMM

3.1 Structure of the Experiments

All SVM experiments were based on the following generic template.

1. Select a protein category of interest.
2. Find a protein sequence known to belong to this category.

3. Use the latest available version of SAM-TXX software to produce a hidden Markov model from the single protein sequence.

4. Find a set of protein sequences that are known to belong to the category of interest. These are the *positive examples.*

5. Find a set of protein sequences that are known not to belong to the category of interest. These are the *negative examples.*

6. For every example, derive a Fisher score vector from the HMM.

7. Partition the Fisher score vectors of the positive examples into *training examples* and *test examples.*

8. Partition the Fisher score vectors of the negative examples into *training examples* and *test examples.*

9. Train the SVM on the positive and negative training examples. In this step, an α coefficient is computed for each training Fisher score vector as described in Section 2.2.4.

10. Test the SVM on the positive and negative test examples and evaluate the rate of misclassifications. Here, the discriminant is used to classify the test examples eq. (2.27).

The template was varied for the series of experiments by a different choice of data set, detailed in Table 3.1 and Table 3.2. Results of the SVM experiments and those for the control methods appear in Section 4.

3.2 GPCR Superfamily Recognition

In the first round of experiments, the protein category of interest was the GPCR superfamily.

As shown in Table 2.1 [25], there are five major families within this superfamily. In a variant of Steps 2 and 3, shown above on the generic experiment template, a sequence was selected at random from each of the five families, and five HMMs were built using SAM-T9X.

3.2.1 Data Sets

The selection of positive examples (Step 4 of the generic template) was provided by Drs. Gert Vriend and Florence Horn of EMBL in Heidelberg, Germany. The positive examples came from all five GPCR families: 692 sequences from Family A (rhodopsin and beta2-adrenergic receptors), 56 from Family B (calcitonin and PTH/PTHrP receptors), 16 from Family C (metabotropic receptors), 11 from Family D (STE2 and STE3 pheromone receptors) and 3 from Family E (cAMP receptors).

They also provided a set of 99 negative examples (Step 5) which are difficult to distinguish from GPCRs: 18 Archebacteriolopsins and 80 G-alpha proteins. I supplemented the negative examples with 2425 additional protein sequences taken from the SCOP version 1.37 PDB90 domain database. These are easier negative examples than the Archebacteriolopsins and G-alpha proteins, because SCOP is heavily weighted towards globular, non-membrane protein domains which are not similar to members of the GPCR superfamily.

To avoid duplicate sequences among the negative examples, it was necessary to check SCOP thoroughly for any overlap with the sequences selected by Vriend and Horn. Although there are no GPCRs in SCOP v. 1.37, I discovered one domain corresponding to a G-alpha protein (1gia) and another to four of the Archebacteriolopsins (1bct). Before any experiments were run, I deleted these domains from the set of negative examples.

Another potential problem was that GPCRs and non-GPCRs might be distinguishable on the basis of sequence length. GPCRs vary in length from several hundred to over 1200 amino acids. Several of the Archebacteriolopsins have as few as 260 amino acids, and there are domain sequences in SCOP with only 150 amino acids. To eliminate length as a possible discrimination feature, I padded every sequence used in the experiments to 1225 amino acids, using the Mark Diekhans program *pad-seqs*. The padding was done with randomly selected amino acids from the distribution of amino acids in SCOP. Although the padding adds noise to the training and test sets, it should not effect the performance of BLAST, Smith-Waterman, HMMs, or SVMs, because all these methods use local alignments to compute the score of a sequence.

3.2.2 Using Multiple HMMs

Fisher score vectors were computed for each padded protein sequence in the positive and negative example sets, using the *get-sam-features* program by Mark Diekhans. As described in Section 2.2.5, only the derivatives with respect to the HMM match state probabilities θ were included. In an expanded version of Step 6 from the generic template, five Fisher score vectors were computed for each sequence, one for each HMM built in Step 1. As a result, each sequence was represented by a score vector from the Rhodopsin/beta2-

adrenergic receptor model, a score vector from the Calicitonin and PTH/PTHrP receptor model, one from the Metabotropic receptor model, and so forth.

I used this variation on Step 6 because receptors from different GPCR families do not share sequence similarity. Consequently, a single HMM cannot be an accurate statistical representation of all GPCRs.

3.2.3 Partitions of Positive and Negative Examples

The partition of positive examples into training and test sets (Step 7 of the generic template) was done using a "leave-one-out" method. When the sequences from Family A (Rhodopsin/beta2-adrenergic receptors) were used as the positive test set, the Family B, Family C, Family D and Family E sequences were the positive training set. When the sequences from Family B (Calicitonin and PTH/PTHrP receptors) were the positive test set, the Family A, Family C, Family D and Family E sequences were the positive training set, and so forth.

To thoroughly evaluate the SVM's classification performance, the partition of negative examples into training and test sets (Step 8 of the generic template) was done in four different ways.

For one experiment, half of the difficult negative examples were in the training set and the other half were in the test set. In contrast, the easy negative examples from SCOP were carefully separated by fold so that the training sets never contained sequences from the same fold as the test set. From a machine learning point of view, the lack of segregation among the difficult negative examples could be considered cheating. When the machine is tested for its generalization abilities, it should be exposed to completely new

examples. By seeing some examples very similar to members of the training set, it may have an unfair advantage.

I evaluated the effect of this unfair advantage with a second partition of negative test examples, putting all of the Archebacteriolopsins in the training set and all of the G-alpha proteins in the test set. For some datasets, the SVM's performance was not noticeably different. In other datasets, the SVM did slightly better when it was allowed to "cheat". Apparently, the unfair advantage is not always used when the SVM classifies Fisher score vectors. This result was also observed in the ISMB paper [28].

Each experiment was repeated with training and test sets swapped, for a total of four partitions of negative examples.

Details of the training and test sets used in my experiments appear in Table 3.1 and Table 3.2.

3.2.4 Training and Testing the SVM

By partitioning and swapping the training examples, as shown in Table 3.1 and Table 3.2, and using five HMMs to generate Fisher score vectors for each example, I trained one-hundred different SVMs to discriminate GPCRs (variation of Step 9 in the generic template). The software for the experiments, written by Tommi Jaakkola, was set with the parameters reported to produce best results in the ISMB paper [1] [28].

As described in Step 10 of the generic template, each SVM was tested on the appropriate set of positive and negative test examples (see Table 3.1 and Table 3.2). The

[1] The *stop* parameter, which controls a convergence threshold for the optimization, was set to 0.00001. The *autovar* parameter, used to calculate the width of the radial basis kernel (σ in Equation 2.38), was set to 5.

	Training Set	
Data Set ID#	Positive	Negative
1	Family B,C,D,E	9 Archebacteriolopsins 40 G-α proteins 723 SCOP all β fold 248 SCOP small protein fold 55 SCOP multi-domain fold 30 SCOP membrane, cell surface, peptide fold 47 SCOP peptide and fragment fold 4 SCOP designed protein fold
2	Family A,C,D,E	same as above
3	Family A,B,D,E	same as above
4	Family A,B,C,E	same as above
5	Family A,B,C,D	same as above
6,7,8,9,10	same as test sets for Datasets 1,2,3,4,5	
11	Family B,C,D,E	18 Archebacteriolopsins 408 SCOP all α fold 488 SCOP parallel β fold2 422 SCOP anti-parallel β fold
12	Family A,C,D,E	same as above
13	Family A,B,D,E	same as above
14	Family A,B,C,E	same as above
15	Family A,B,C,D	same as above
16,17,18,19,20	same as test sets for Datasets 11,12,13,14,15	

Table 3.1: Twenty partitions of positive and negative examples into training sets for GPCR superfamily recognition. Test sets are shown in Table 3.2.

	Test Set	
Data Set ID#	Positive	Negative
1	Family A	9 Archebacteriolopsins 41 G-α proteins 408 SCOP all α fold 488 SCOP parallel β fold 422 SCOP anti-parallel β fold
2	Family B	same as above
3	Family C	same as above
4	Family D	same as above
5	Family E	same as above
6,7,8,9,10	same as train sets for Datasets 1,2,3,4,5	
11	Family A	81 G-α proteins 723 SCOP all β fold 48 SCOP small protein fold 55 SCOP multi-domain fold 30 SCOP membrane, cell surface, peptide fold 47 SCOP peptide and fragment fold 4 SCOP designed protein fold
12	Family B	same as above
13	Family C	same as above
14	Family D	same as above
15	Family E	same as above
16,17,18,19,20	same as train sets for Datasets 11,12,13,14,15	

Table 3.2: Twenty partitions of positive and negative examples into test sets for GPCR superfamily recognition. Train sets are shown in Table 3.1.

multiple-HMM approach resulted in each test sequence being scored by five different SVMs, and classification performance was evaluated for each one. I also generated an average score for each sequence. The results shown in Section 4 measure the SVM classification errors on these average scores. Averages were computed using only four scores from the models that were not in the family of the positive test sequences. (For example, when the positive test set was Family A, the final score of each test sequence was the average of the scores computed by four SVMs, and these SVMs were built on top of HMMs for Families B,C,D, and E.)

3.3 Recognizing GPCR sequences with specific function

3.3.1 Selected Subfamilies

I designed recognition experiments for three sub-families of the *amine receptors*, a sub-family of GPCR Family A. These are the histamine receptors, muscarinic (acetycholine) receptors, and serotonin receptors. The selection was motivated by the suggestions of Dr. Florence Horn [23]. With respect to current medical knowledge, the natural ligands of amine receptors have the greatest therapeutic value of any GPCR subfamily. Drugs designed to mimic/augment receptor activity after binding an amine (*amine agonists*) and also to counteract/inhibit this activity (*amine antagonists*) have been developed in the treatment of: glaucoma and tachycardia (muscarinic agonists), migraine (serotonin antagonists), and allgeries (histamine antagonists) [24].

3.3.2 Datasets

The sub-family experiments were based on the datasets shown in Table 3.3 and Table 3.4.

Each amine subfamily can be subdivided into *Types*. In Datasets 21, 22, and 23 from Table 3.3 and Table 3.4, there is no overlap between Types in the positive training and test sets. For example, in Dataset 21 all positive training examples are histamine Type I receptors and all positive test examples are histamine Type II receptors.

Interestingly, none of the tested methods was able to effectively discriminate histamine Type II after being trained on Type I, and vice versa. Inspection of a multiple alignment of both types (see Figure 3.1) reveals that there is not much similarity between the two types of histamine receptors. This may be an instance of convergent evolution, where two different kinds of receptor molecules independently evolved to bind histamine ligands. When the experiment was modified so that the positive training set included both Type I and Type II histamines (Dataset 27 in Table 3.3 and Table 3.4), the SVM was able to perfectly discriminate histamine receptors.

The negative training and test examples for this series of experiments can be visualized as a hierarchy of nested subsets as in Figure 3.2. The non-GPCR families of archebacteriolopsins and G-alpha protein domains lie outside the hierarchy. The outermost ring of negative examples is taken from GPCR Families B,C,D, and E. These are the GPCR examples with the least similarity to histamine, muscarinic, and serotonin receptors. The next ring of examples is from Family A, but not in the amine receptor subfamily. In the datasets, these have been split into the peptide receptors, and all remaining subfamilies of

	Training Set	
Data Set ID#	Positive	Negative
21	5 histamine Type I	15 muscarinic receptors 48 serotonin receptors 204 peptide receptors 56 GPCR Family B 18 archebacteriolopsins
22	9 acetycholine vertebrate Types I,II,III	10 histamine receptors 48 serotonin receptors 204 peptide receptors 56 GPCR Family B 18 archebacteriolopsins
23	20 serotonin vertebrate Type I	10 histamine receptors 15 muscarinic receptors 204 peptide receptors 56 GPCR Family B 18 archebacteriolopsins
24,25,26	same as test sets for Datasets 21,22,23	
27	3 histamine Type I 2 histamine Type II	15 muscarinic receptors 48 serotonin receptors 204 peptide receptors 56 GPCR Family B 18 archebacteriolopsins
28	9 sertonin vertebrate Type I, 4 Type II, 1 Type IV, 2 Type V, 1 Type VI, 2 Type VII, 1 insect Type I, 1 insect Type II, 4 other	10 histamine receptors 15 muscarinic receptors 204 peptide receptors 56 GPCR Family B 18 archebacteriolopsins
29,30	same as test sets for Datasets 27,28	

Table 3.3: Partitions of positive and negative examples into training sets for amine subfamily experiments. Test sets are shown in Table 3.4.

	Test Set	
Data Set ID#	Positive	Negative
21	5 histamine Type II	73 amine receptors 302 GPCR Family A non-amine 30 GPCR Family C,D,E 81 G-α proteins
22	6 acetycholine vertebrate Types IV,V, insect I, other	same as above
23	28 serotonin vertebrate Types II-VII, insect I,II, other	same as above
24,25,26	same as train sets from Datasets 21,22,23	
27	2 histamine Type I 3 histamine Type II	73 amine receptors 302 GPCR Family A non-amine 30 GPCR Family C,D,E 81 G-α proteins
28	11 serotonin vertebrate Type I, 4 Type II, 1 Type IV, 2 Type V, 1 Type VI, 2 Type VII 3 other	same as above
29,30	same as train sets from Datasets 27,28	

Table 3.4: Partitions of positive and negative examples into test sets for amine subfamily experiments. Training sets are shown in Table 3.3.

```
                  250        260        270        280          290
                   |          |          |          |           |
HH1R_BOVIN|P30546 AKKPGKESPWEVLKRKPKDTGGGPVLKPPSQEP.KeVTSPGVFSQEkeeKDG
HH1R_CAVPO|P31389 TRRMGKESPWEDPKRCSKDASGVHTPMPSSQHLvD.MPCAAVLSED...EGG
HH1R_HUMAN|P35367 AKKPGKESPWEVLKRKPKDAGGGSVLKSPSQTP.KeMKSPVVFSQE...DDR
HH1R_MOUSE|P70174 AKKPGKESPWGVQTRPSRDPTGGLDQKSTSADP.K.VTSPTVFSQE...GER
HH1R_RAT|P31390   AKKPGRESPWGVLKRPSRDPSVGLDQKSTSEDP.K.MTSPTVFSQE...GER
HH2R_CANFA|P17124 AKRIHHMGSWKAATIGEHKA-------------.T.VTLAAVMGA-...---
HH2R_CAVPO|P47747 ARRINHIGSWKAATIREHKA-------------.T.VTLAAVMGA-...---
HH2R_HUMAN|P25021 AKRINHISSWKAATIREHKA-------------.T.VTLAAVMGA-...---
HH2R_MOUSE|P97292 AKRINHISSWKAATIREHKA-------------.T.VTLAAVMGA-...---
HH2R_RAT|P25102   AKRINHISSWKAATIREHKA-------------.T.VTLAAVMGA-...---
```

Figure 3.1: Portion of a SAM-T99 multiple alignment of histamine receptors Type I (HH1R) and Type II (HH2R). Members of each type are highly homologous, but Type I and Type II are very different from each other. This may be an example of convergent evolution.

non-amine Family A receptors. The innermost ring of negative examples are sub-families of amine receptors different from the sub-family being recognized in a particular experiment. (Strictly speaking, this is the "sub-sub-family" level of specificity.) For instance, when serotonin receptors are being recognized, the amine negatives include muscarinic receptors, histamine receptors, adrenoceptors, dopamine, and octopamine receptors.

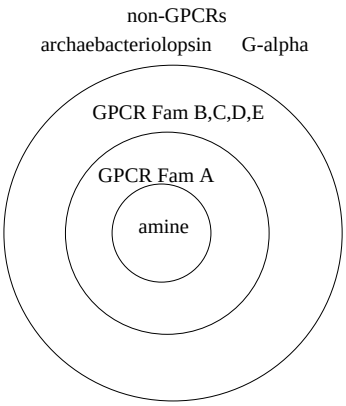


Figure 3.2: Nested hierarchy of negative examples used in histamine, muscarinic, and serotonin recognition experiments.

To further clarify the relationships between the GPCR superfamily, families, subfamilies and types, I have included a tree diagram of the hierarchy, Figure 3.3, showing only the portion of the tree relevant to my amine subfamily experiments.

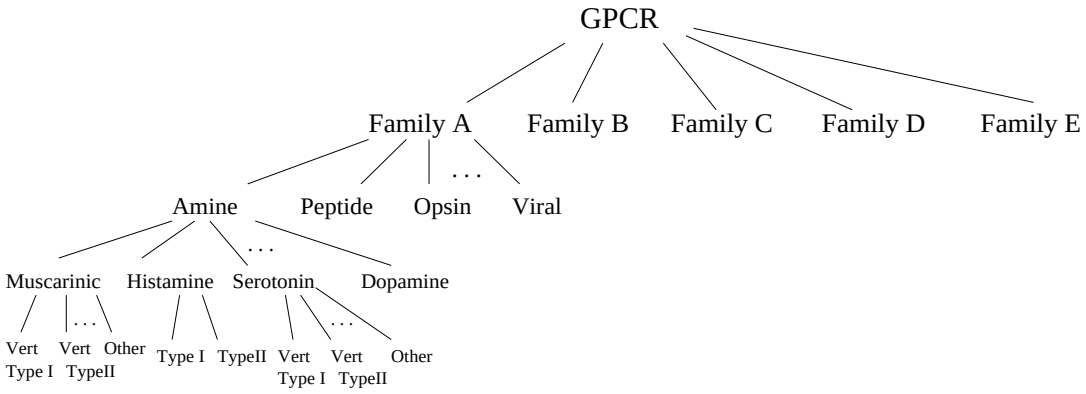


Figure 3.3: Portion of the GPCR family tree relevant to amine subfamily experiments on histamine, muscarinic, and serotonin recognition.

3.4 Control Experiments

For the problem of recognizing sequences of the GPCR superfamily, I compared the performance of support vector machines with several state-of-the-art methods: SAM-T99, BLAST family pairwise search, and Smith-Waterman family pairwise search.

3.4.1 Family Pairwise Search with BLAST

BLAST (Basic Local Alignment Search Tool) is a fast homolog finder popular among research biologists. The algorithm takes a query sequence and searches a target database for sequences containing the longest segments similar to those in the query. BLAST reports an *E-value* score (expected number of false positives with this good a match) for each sequence in the database, which measures the statistical significance of the similarities. The best possible quality match is given an E-value of 0. Poor quality matches, predicted by BLAST to be random similarities, have large E-values.

When used in conjunction with the *family pairwise search algorithm (FPS)*, BLAST can be used as a sequence classifier. The method is described in a paper by Bill Grundy [20], and was used as a control in the ISMB paper [28]. I used *WU-blastp version 2.0a16* and *blast-score*, a script written by Mark Diekhans, to implement the algorithm.

For family pairwise search, the database is queried with a set of query sequences, known to be homologous (i.e. query sequences are positive examples). These are equivalent to the positive training set in the SVM and HMM methods. The target database must contain the positive and negative test examples to be classified, but can contain other sequences as well.

First, the query sequences are considered in relation to the test examples. For each query sequence, BLAST searches for similar contiguous regions in each test example and reports an E-value for the test example. Then the test examples are considered in relation to the query sequences. For each test example, we find the query sequence with the best E-value. This is the query sequence that BLAST considers the most similar to the test example. The query sequence's E-value becomes the *score* assigned to the test example.

The scores of the positive and negative test examples are sorted and statistically evaluated as described in Section 3.5. Note that the positive test examples and the positive training examples are taken from the class of proteins we wish to recognize, so the positive test examples should have greater similarity to the queries than do the negative test examples. For perfect separation of positive and negative test examples, the minimum E-value scores calculated for all positive test examples must be smaller than the those for the negative test examples.

These experiments were done on the data sets shown in Table 3.1 and Table 3.2, with the negatives removed from the training set. This was unavoidable since the method uses only positive training examples.

3.4.2 Family Pairwise Search with Smith-Waterman

Smith-Waterman is a sequence alignment and scoring algorithm that is slower (in most implementations), but more sensitive than BLAST. Like BLAST, it scores a query sequence against a target database, however a complete alignment is done for each comparison.

For protein sequence alignments, scores are computed by residue-to-residue comparisons using scoring matrices based on the evolutionary distance between amino acids. The algorithm allows for subsequence matching, reflecting the biological reality that, due to mutations, homologs may contain regions of low similarity in between functionally important regions of high similarity.

Smith-Waterman is based on a recursive equation, implemented with a dynamic programming matrix whose rows and columns are indexed by the two sequences being compared. Details of the recursion and the algorithm can be found in the paper by Smith and Waterman [53].

A blazingly fast Smith-Waterman implementation is available at U.C. Santa Cruz in the lab of Professor Richard Hughey. It runs on his Kestrel SIMD parallel processing board. Kestrel Smith-Waterman takes a query sequence and computes both an alignment score and an E-value for every sequence in a target database.

I was able to repeat the methodology described in Section 3.4.1 using Kestrel Smith-Waterman in the place of BLAST. As in the BLAST experiments, the E-value was used as a sequence score.

The target databases were translated into a specially encoded Kestrel format using a script written by Leslie Grate. The family pairwise search algorithm was executed with *sw_fps*, a script written by myself.

3.4.3 SAM Target-99

SAM-T99 is a state-of-the-art algorithm to build a hidden Markov model for a group of related protein sequences. Given a single *seed sequence* or a multiple alignment of

training sequences, it builds the model using an iterative method. First, it does a BLAST search of a database (the default is the non-redundant protein database *nr*) and selects a "pool" of potential homologs of the sequence. The strongest matches from the pool are the training set for an initial HMM, which generates a multiple alignment of the strong matches. This HMM is used to search the pool for weaker homologs, which are added to the training set. The appended training set trains a second HMM, and so forth. The program executes four rounds of database searching and model building. [32]. I built one SAM-T99 model for each GPCR family, starting with a single, randomly chosen family member as the seed sequence. The positive and negative test sets shown in Table 3.2 were scored with respect to the five models by another program from the SAM suite, *hmmscore*, using the defaults (EM-scoring and geometric mean *null model* [7].) As in the SVM experiments, I averaged the scores from the four models that were not in the family of the positive test sequences.

The *hmmscore* program has recently been upgraded to use a *reverse null model* which improves the discrimination performance of hidden Markov models [31]. It is very likely that SAM-T99 would do better if reverse null model scoring was used on our test sets. In future experiments, I plan to switch to this new scoring method.

3.5 Statistical Analysis

An ideal classification method would balance *sensitivity*, the ability to recognize a true positive example and *specificity*, the ability to reject a false positive. In practice, most classifiers make errors in both directions. The true positives are positive test exam-

ples, correctly identified by the classifier as positive. The false positives are negative test examples that the classifier incorrectly identifies as positive. (False negatives are positive test examples that the classifier incorrectly identifies as negative.)

In the analysis that follows, "positive" and "negative" refer to the labelling of a sequence, and indicate whether it is a member of the class of sequences to be recognized in a particular experiment. For the GPCR recognition experiments, a positive sequence is a GPCR and a negative sequence is a non-GPCR, as defined in Table 3.1 and Table 3.2. For recognition of GPCR histamine receptors, a positive sequence is a GPCR histamine receptor and a negative sequence is any sequence that is not a GPCR histamine receptor, as defined in Table 3.3 and Table 3.4.

Each classification method was evaluated using a sorted list of test example scores. The method was to "sweep" the threshold through the range of score values in the list, from best to worst. During the first pass through the sorted list, the threshold is the score of the first example on the list; in the second pass, the threshold is the score of the second example on the list, and so forth.

Several counts are maintained for each example in the test set. For each positive example, we count the number of positive test sequences in the list with better scores as *true positives*, and the number of negative test sequences in the list with better scores as *false positives*. For negative examples, we count the number of positive test sequences that score worse as *false negatives*.

From these counts, it is possible to compute a rate of false positives or RFP and a rate of true positives RTP for each positive sequence.

$\frac{\text{number of false positives}}{\text{total number of negative test seqs}}$	Rate of False Positives
$\frac{\text{number of true positives}}{\text{total number of positive test seqs}}$	Rate of True Positives

Three kinds of summary statistics were computed for each experiment:

- the *minimum error point* (MEP). At one threshold value, the number of false positives plus the number of false negatives is at a minimum. This sum is the minimum total number of errors.
- the *maximum rate of false positives*. At one threshold value, all positive test sequences are recognized and as few negatives as possible, subject to this requirement. The maximum RFP is the RFP at this threshold.
- the *median rate of false positives*. At one threshold value, half of the positive examples have RFPs that are larger and half have RFPs that are smaller. The median RFP is the RFP at this threshold.

As described in the ISMB paper [28], the maximum RFP may be misleading, because of the stringent requirement that all true positives must be recognized. A few difficult-to-recognize positive test examples can result in a large maximum RFP, when in fact, all other test examples have been classified correctly. The median RFP is a more forgiving statistic, because a low median RFP only requires that at least half of the positive test examples can be easily recognized by the method.

Chapter 4

Results

During the course of my experiments, I tried several variations on the hidden Markov model and support vector machine methods.

Prior to the release of SAM-T99, I worked with an earlier version of the program known as SAM-T98. I also did a series of experiments in which SAM-T99's homolog search space was restricted to a carefully chosen set of positive training examples. I tested "general" SAM-T99 models which "self-select" their training sets, as described in Section 3.4.3. Finally, for sub-family recognition only, I tested a new "beta" option of SAM-T99 called *family* which optimizes recognition of small groups of related sequences.

The Fisher kernel support vector machine method is built on top of a hidden Markov model, as desribed in Section 3.1. Variations on this SVM method included SVM on top of SAM-T98 models, SVM on top of SAM-T99 models (where the T99 training set was a carefully chosen set or subset of positive training examples) and SVM on top of general SAM-T99 models.

Without exception, I observed these results:

- SAM-T98 models don't discriminate as well as general SAM-T99 models. This is not surprising because T99 is an upgraded, improved version of T98.
- SAM-T99 models trained in a restricted mode don't discriminate as well as general SAM-T99 models.
- For the subfamily experiments, the SAM-T99 models built with the "family" option (which is still a "beta" feature of the program) did not discriminate significantly better than those built with general SAM-T99 models.
- SVMs on top of SAM-T98 don't discriminate as well as SVMs on top of general SAM-T99 models.
- SVMs on top of SAM-T99 models trained in a restricted mode don't discriminate as well as SVMs on top of general SAM-T99 models.

To present the clearest picture of how BLAST, Smith-Waterman, hidden Markov models and support vector machine methods compare to each other, the results tallied in this section only include the best performing version of each method: general SAM-T99 models and the SVMs built on top of them. For the sub-family experiments, I also include the results of the "family" option.

Here is a brief overview of my results:

- To discrminate the GPCR superfamily, the best method is a general SAM-T99 model. These HMMs are able to perfectly recognize Family A, which represents approximately 90% of known GPCRs [24]. They also perfectly recognize Family B and Family E, and outperform the other methods on Family C. On Family D, the

T99/SVM combination performs better, but only by a tiny amount.

- General SAM-T99 models do not well discrminate GPCR sub-families. For sub-family recognition, the best method is a Fisher kernel SVM built on top of a general SAM-T99 model.

The sub-family discrimination problem requires recognition of fine distinctions between sequences, a task for which the SVM is well suited. The alignments built by SAM-T99 (both general and family option versions) were overgeneralized, i.e. they contained many sequences which were not members of the subfamily to be recognized by the model. However, when a Fisher kernel SVM is built on top of an overgeneralized HMM, it uses Fisher score vectors of both positive examples (e.g. histamines) and negative examples (e.g. non-histamines), and this helps prevent it from overgeneralizing.

The superfamily discrmination problem requires recognition of sequences from five different families which share very little sequence homlogy (only 15% average pairwise similarity [24]). A general SAM-T99 HMM is the preferred tool for classifying GPCRs vs. non-GPCRs. Even though an SVM built on top of one of these models can also well discriminate the superfamily, it is consistently less accurate and a waste of the additional time and computer cycles required.

4.1 Superfamily Recognition

An overview of superfamily recognition results appears in Figure 4.1, Figure 4.2 and Figure 4.3. The Figure shows the relative performance of

- BLAST

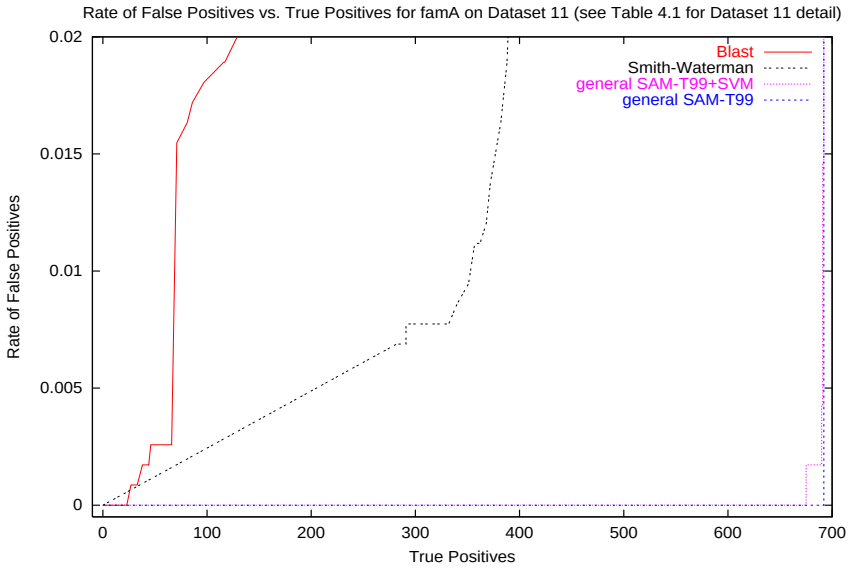
- Smith-Waterman
- SAM-T99 general HMM
- SVM with Fisher score vectors from SAM-T99 general HMM

on five datasets designed to test each method's ability to discriminate a single GPCR Family from non-GPCRs. These five datasets (numbered 11,12,13,14 and 15 in Table 3.1) and Table 3.2) were the most difficult to discriminate because there are no sequences in the training sets that are similar to the sequences in the test sets. (The datasets numbered 1,2,3,4 and 5 contain negative test examples similar to negative training examples, giving the SVM an advantage.)

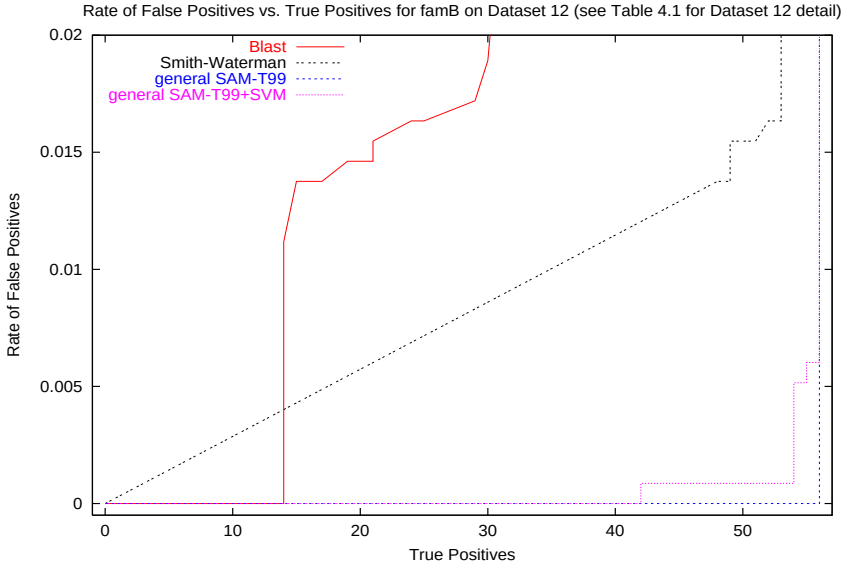
I have chosen to show only results from Datasets 11,12,13,14 and 15 only, because results from the three other groups of datasets (Datasets 1-5, 6-10 and 16-20) are almost exactly the same.

Rates of false positives greater than 0.02 are not shown. Even though a method is effectively useful with an error rate above this threshold, blowing up the plots this much allows for a detailed comparison of the methods.

Table 4.1 presents a numeric overview of the statistics for my superfamily recognition experiments.

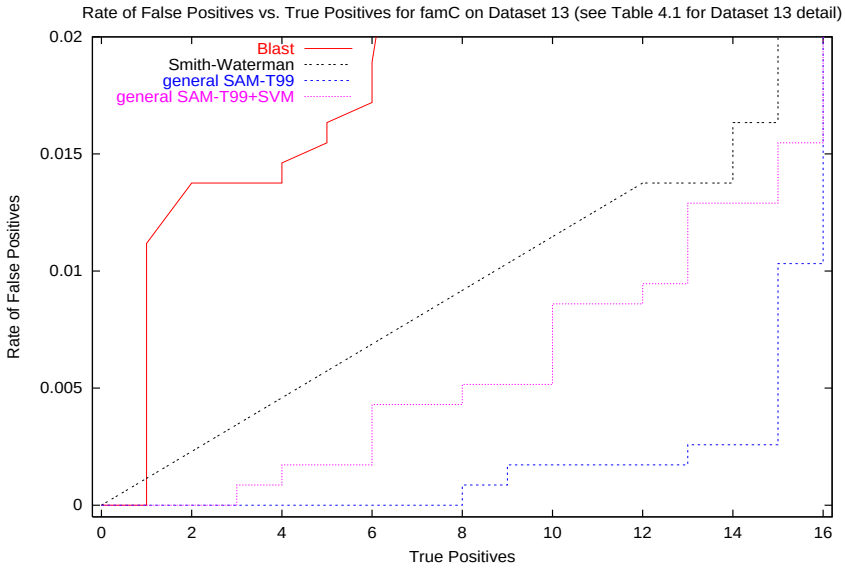


Family A. Dataset 11 in Table 3.1 and Table 3.2

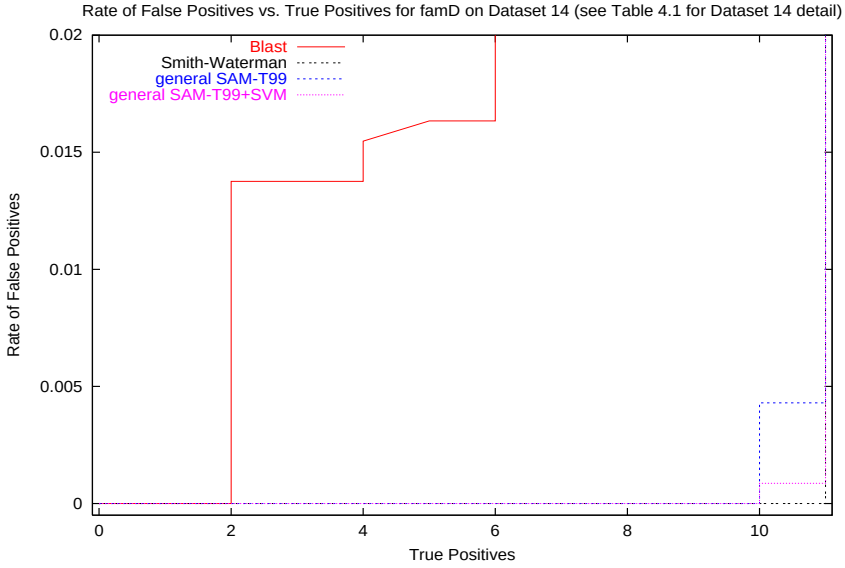


Family B. Dataset 12 in Table 3.1 and Table 3.2

Figure 4.1: Number of true positives recognized vs. the rate of false positives for recognition of GPCR Families A and B vs non-GPCRs. These are "strict" datasets where no negative test example is of the same type as a negative training example. Rates of false positives greater than 0.02 are not shown. If the curve for a given method is not visible, its discrimination performance was perfect and it is being covered by the curve of another method. The extended straight lines of the Smith Waterman curve are the result of interpolation.

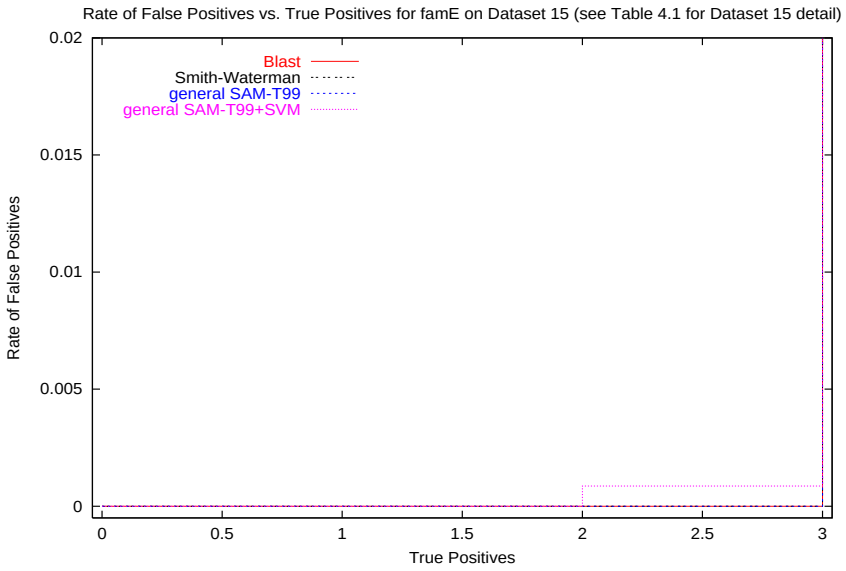


Family C. Dataset 13 in Table 3.1 and Table 3.2



Family D. Dataset 14 in Table 3.1 and Table 3.2

Figure 4.2: Number of true positives recognized vs. the rate of false positives for recognition of GPCR Families C and D vs non-GPCRs. These are "strict" datasets where no negative test example is of the same type as a negative training example. Rates of false positives greater than 0.02 are not shown. If the curve for a given method is not visible, its discrimination performance was perfect and it is being covered by the curve of another method. The extended straight lines of the Smith Waterman curve are the result of interpolation.



Family E. Dataset 15 in Table 3.1 and Table 3.2

Figure 4.3: Number of true positives recognized vs. the rate of false positives for recognition of GPCR Family E vs non-GPCRs. These is a "strict" dataset where no negative test example is of the same type as a negative training example. Rates of false positives greater than 0.02 are not shown. If the curve for a given method is not visible, its discrimination performance was perfect and it is being covered by the curve of another method.

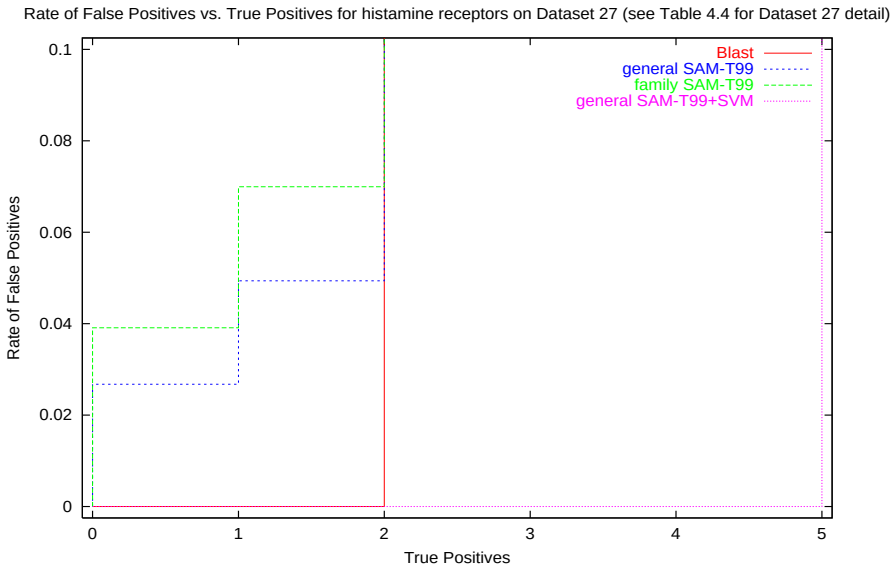
Test Fam	Method	MEP					Med RFP	Max RFP
		Errs	NFN	NTN	NFP	NTP		
A	BLAST	546	403	1045	143	289	0.2109	0.9925
A	SW	322	293	1159	29	399	0.0142	0.9967
A	gen SAM-T99 HMM	0	0	1188	0	692	0	0
A	SVM+gen T99	4	2	1186	2	690	0	0.0146
B	BLAST	44	41	1185	3	15	0.0184	0.3799
B	SW	9	9	1188	0	47	0	0.1364
B	gen SAM-T99 HMM	0	0	1188	0	56	0	0
B	SVM+gen T99	0	0	1188	0	56	0	0
C	BLAST	16	16	1188	0	0	0.0326	0.2987
C	SW	4	4	1188	0	12	0	0.1498
C	gen SAM-T99 HMM	4	1	1185	3	15	0.0009	0.0103
C	SVM+gen T99	10	9	1187	1	7	0.0052	0.0198
D	BLAST	10	10	1188	0	1	0.0569	0.5213
D	SW	0	0	1188	0	11	0	0
D	gen SAM-T99 HMM	1	1	1188	0	10	0	0.0043
D	SVM+gen T99	1	0–1	1187–1188	1–0	10–11	0	0.0009
E	BLAST	0	1	1188	0	2	0.0008	0.0008
E	SW	0	0	1188	0	3	0	0
E	gen SAM-T99 HMM	0	0	1188	0	3	0	0
E	SVM+gen T99	0	0	1188	0	3	0	0

Table 4.1: Comparison of the five methods on Datasets 11-15 from Table 3.1 and Table 3.2. These are "strict" datasets where no negative test example is of the same type as a negative training example. Plots of the false positive rate for these datasets appear in Figure 4.1, Figure 4.2 and Figure 4.3. The minimum error point (MEP) is the threshold where the method makes the fewest errors of both kinds, false positives and false negatives. The number of false negatives (NFN), number of true negatives (NTN), number of false positives (NFP) and number of true positives (NTP) at the minimum error point are shown. Errs is the total number of errors (NFN+NFP) at the minimum error point. The median rate of false positives and maximum rate of false positives for the experiments are also shown.

4.2 Subfamily Recognition

4.2.1 Histamine Receptors

As described in Section 3.3.2, none of the tested methods were able to recognize Type I histamine receptors after being trained on Type II histamine receptors, and vice versa. Figure 4.2.1 shows the performance of BLAST, SAM-T99 and the SVM on a training set containing both Type I and Type II histamines (Dataset 27 in Table 3.3 and Table 3.4). When the training and test sets were swapped (Dataset 29 in Table 3.3 and Table 3.4), there were no significant differences in the results.



Histamine receptors. Dataset 27 in Table 3.3 and Table 3.4

Figure 4.4: When histamine receptors of both Type I and Type II are included in the positive training set, the SVM perfectly discriminates the test set of five histamine receptors. BLAST perfectly discriminates two out of five histamine receptors. The SAM-T99 models overgeneralize and do not well discriminate this test set. Rates of false positives above 0.1 are not shown.

In Figure 4.2.1, the SVM is able to perfectly discriminate the five histamine positive test examples. BLAST perfectly recognizes two positive test examples. Both

```
                      200          210        220        230
                        |            |          |          |
HH1R_HUMAN|P35367     NFYL...PTLLMLWFYAKIYKAVRQHCQHRELINRSLPSFSE
HH1R_HUMAN|P35367     NFYL...PTLLMLWFYAKIYKAVRQHCQHRELINRSLPSFSE
HH1R_BOVIN|P30546     NFYL...PTLLMLWFYAKIYKAVRQHCQHRELINGSFPSFSD
HH1R_RAT|P31390       NFYL...PTLLMLWFYVKIYKAVRRHCQHRQLTNGSLPSFSE
HH1R_MOUSE|P70174     NFYL...PTLLMLWFYVKIYNGVRRHCQHPQLTNGSLPTFLE
HH1R__CAVPO|P31389    NFYL...PTLLMLWFYIRIYKAVRRHCQHRQLINSSLPSFSE
HH2R_RAT|P25102       TFYL...PLLIMCVTYYRIFKIAREQAKRINHISSWK-----
HH2R_MOUSE|P97292     TFYL...PLLIMCVTYYRIFKIAREQAKRINHISSWK-----
HH2R_HUMAN|P25021     TFYL...PLLIMCITYYRIFKVARDQAKRINHISSWK-----
HH2R_CANFA|P17124     TFYL...PLLVMCITYYRIFKIARDQAKRIHHMGSWK-----
HH2R_CAVPO|P47747     TFYL...PLLIMCITYFRIFKIAREQARRINHIGSWK-----
AA1R_CAVPO|P47745     FFVWvlpPLLLMVLIYLEVFYLIRKQLSKKVSASS-------
AA1R_RAT|P25099       FFVWvlpPLLLMVLIYLEVFYLIRKQLNKKVSASS-------
AA3R_HUMAN|P33765     FLTWifiPLVVMCAIYLDIFYIIRNKLS--------------
AA2B_CHICK|O13076     FFGCvllPLIIMLGIYIKIFMVACKQLHQIELMGNSR-----
```

Figure 4.5: Portion of a SAM-T99 multiple alignment of Type I and Type II histamine receptors(HH1R and HH2R) with four adenosine receptors (AA1R, AA3R, AA2B) mistakenly identified as histamine by BLAST. Inspection of positions 202-229 shows that the adenosine receptors contain a chunk very similar to the histamine receptors.

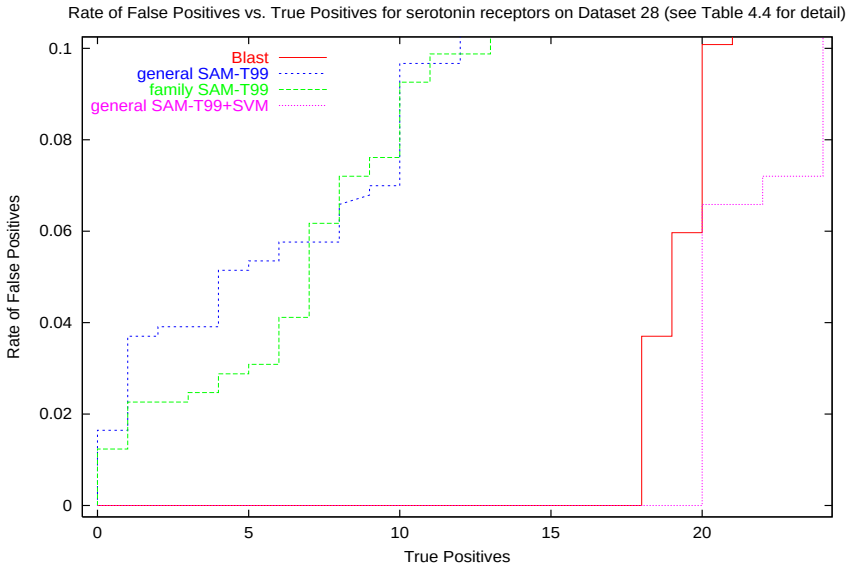
general and family SAM-T99 models overgeneralize, incorporating non-histamines into their training alignments, and are unable to discriminate the histamine receptors.

To further investigate why BLAST did not do well on this test set, I created a multiple alignment of ten histamine receptors and four Family A (adenosine) receptors which BLAST mistakenly recognized as histamine. Inspection shows that the adenosine receptors contain contiguous regions which are very similar to the histamines (see Figure 4.5). Because it bases its similarity scores on local alignments, BLAST is confused by this situation. The Fisher kernel SVM uses a global similarity measure, and is able to perfectly discriminate between adenosine and histamine receptors.

4.2.2 Serotonin Receptors

As in the histamine case, when the serotonin receptor training set was strictly segregated from the test set by Type (Dataset 23 in Table 3.3 and Table 3.4), the methods tested were unable to discriminate serotonin receptors.

However, when Types of serotonin receptors were mixed in the training set and test set (Dataset 28 from Table 3.3 and Table 3.4), performance was much better. Results are shown in Figure 4.2.2.



Serotonin receptors. Dataset 28 in Table 3.3 and Table 3.4

Figure 4.6: When mixed Types of serotonin receptors are included in the positive training set, the SVM perfectly recognizes 20 out of 28 positive test examples. BLAST perfectly discriminates 17 out of 28 positive test examples. The SAM-T99 models overgeneralize and do not well discriminate this test set. Rates of false positives above 0.1 are not shown.

BLAST performs relatively well on this test set, perfectly recognizing 17 out of the 28 positive test examples. However, the SVM perfectly recognizes 20 of the positive test examples. As in the histamine case, when the training and test sets were swapped

Test subfam	Method	MEP					Med RFP	Max RFP
		Errs	NFN	NTN	NFP	NTP		
Histamine	BLAST	3	3	486	0	2	0.5638	0.8313
Histamine	Gen SAM-T99	5	5	486	0	0	0.2140	0.2428
Histamine	Fam SAM-T99	5	5	486	0	0	0.1934	0.1996
Histamine	SVM+Gen T99	0	0	486	0	5	0	0
Serotonin	BLAST	6	6	486	0	18	0	0.8765
Serotonin	Gen SAM-T99	24	24	486	0	0	0.1091	0.2922
Serotonin	Fam SAM-T99	24	24	486	0	0	0.0988	0.2963
Serotonin	SVM+Gen T99	4	4	486	0	20	0	0.0720

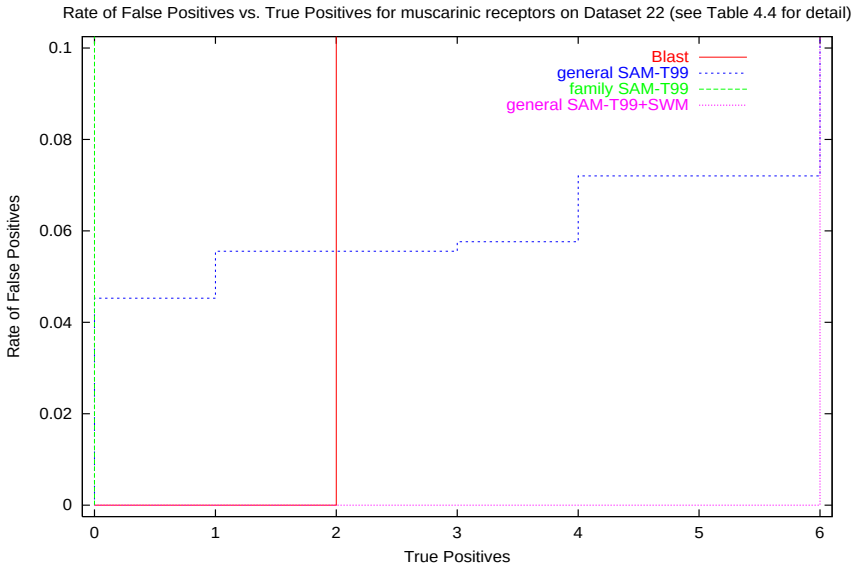
Table 4.2: Comparison of methods on Datasets 27 and 28 from Table 3.3 and Table 3.4. These are "mixed" datasets in which positive training examples and positive test examples are of the same subtype. The minimum error point (MEP) is the threshold where the method makes the fewest errors of both kinds, false positives and false negatives. The number of false negatives (NFN), number of true negatives (NTN), number of false positives (NFP) and number of true positives (NTP) at the minimum error point are shown. Errs is the total number of errors (NFN+NFP) at the minimum error point. The median rate of false positives and maximum rate of false positives for the experiments are also shown.

(Dataset 30 in Table 3.3 and Table 3.4), there were no significant differences in the results.

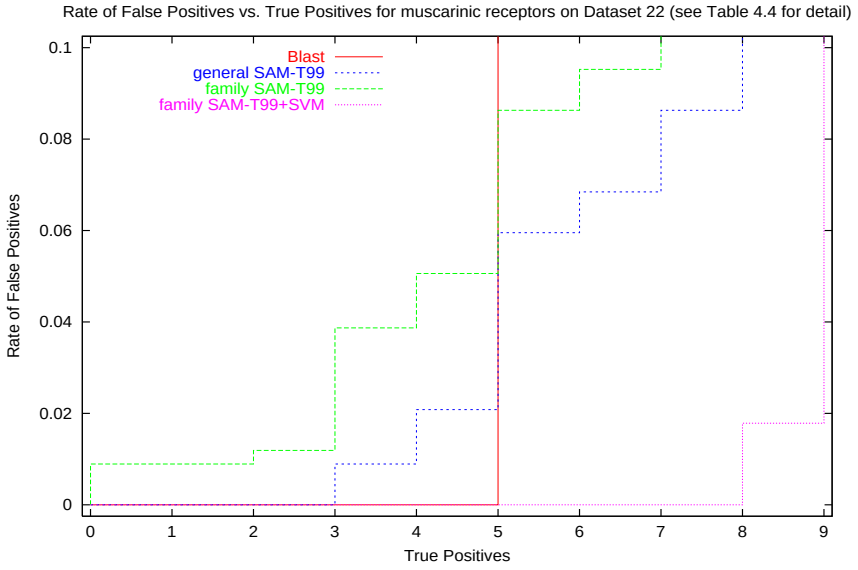
4.2.3 Muscarinic Receptors

Unlike the other two subfamilies tested, the various Types of muscarinic receptors have a close evolutionary relationship [23], and the Fisher kernel SVM was able to well discriminate the two datasets in Table 3.3 and Table 3.4 in which the training and test sets were strictly segregated by Type (Datasets 22 and 25). Results are shown in Figure 4.2.3.

The SVM perfectly recognized the six positive test examples in Dataset 22. BLAST perfectly recognized two positive test examples. When the training and test sets were swapped, the SVM perfectly recognized eight out of nine positive test examples, and BLAST perfectly recognized five positive test examples. The SAM-T99 models overgeneralize on these test sets and do not well discriminate the positive test examples.



Muscarinic receptors. Dataset 22 in Table 3.3 and Table 3.4



Muscarinic receptors. Dataset 25 in Table 3.3 and Table 3.4

Figure 4.7: Recognition of muscarinic receptors on Datasets 22 and 25 from Table 3.3 and Table 3.4. The SVM perfectly recognizes all six positive test examples in Dataset 22 and eight out of nine positive test examples in Dataset 25. BLAST perfectly recognizes two positive test examples in Dataset 22 and five positive test examples in Dataset 22. The SAM-T99 models overgeneralize and do not well discriminate this test set. The best performing SAM-T99 model on this test set is the general model on Dataset 25 which perfectly recognizes three positive test examples. Rates of false positives above 0.1 are not shown.

Test subfam	Method	MEP					Med RFP	Max RFP
		Errs	NFN	NTN	NFP	NTP		
Muscarinic	BLAST	5	5	486	0	1	0.6502	0.8416
Muscarinic	Gen SAM-T99	6	6	486	0	0	0.0576	0.0720
Muscarinic	Fam SAM-T99	6	6	486	0	0	0.5885	0.6008
Muscarinic	SVM+Gen T99	0	0	486	0	6	0	0

Table 4.3: Comparison of methods on Dataset 22 from Table 3.3 and Table 3.4. This is a "strict" dataset where no positive test example is of the same sub-type as a positive training example.

Test subfam	Method	MEP					Med RFP	Max RFP
		Errs	NFN	NTN	NFP	NTP		
Muscarinic	BLAST	3	3	336	0	6	0	0.4018
Muscarinic	Gen SAM-T99	6	6	336	0	3	0.0595	0.1994
Muscarinic	Fam SAM-T99	9	9	336	0	0	0.0863	0.4048
Muscarinic	SVM+Gen T99	1	1	336	0	8	0	0.0179

Table 4.4: Comparison of methods on Dataset 25 from Table 3.3 and Table 3.4. This is a "strict" dataset where no positive test example is of the same sub-type as a positive training example.

Chapter 5

Software

The following software was used in the experiments.

- A protein sequence padding program *pad-seqs* written by Mark Diekhans. This program padded each sequence to a desired length by random selection of amino acids from the distribution in SCOP version 1.37 PDB90.
- A program written by Mark Diekhans that constructs a Fisher score vector for a sequence from a hidden Markov model, *get-sam-features*.[1] The hidden Markov model must be in SAM format [26]. SAM is a sequence alignment and modelling program available from the Computational Biology group at University of California, Santa Cruz.
- Experimental SVM classification software written by Tommi Jaakola.
- A script written by Mark Diekhans that uses the family pairwise search algorithm [20] and BLAST to discriminate between two classes of protein sequences.

[1] The program was set to use Smith-Waterman scoring, which computes fully local alignments when summing over all paths of the HMM.

- Scripts written by myself to convert the output of the SVM software and the FPS BLAST script to files of sequence scores in SAM format. Having a common "score file" format for all results simplified statistical analysis.

- A script written by myself to calculate all statistical information described in the "Statistical Analysis" section from these score files.

- A script written by myself to implement the family pairwise search algorithm [20] on Kestrel Smith-Waterman scoring.

- Gnuplot was used to create graphical plots of the statistical results.

Chapter 6

Related Research

The scope of this thesis includes both protein superfamily (fold) recognition and family/subfamily recognition. Recognition is based on primary sequence information.

6.1 Classification by Fold

Other methods to classify superfamilies by primary sequence information include threading (aligning a new protein sequence to the structure of a known protein),[11] [22] [18] [49] [1], multiple sequence alignment and profile HMMs [55], and iterative profile search [2]. The SMART server is a multiple alignment tool explicitly for protein signaling domains [51].

There are also sophisticated classifiers like CATH which use secondary and tertiary information in addition to primary sequence information. CATH clusters proteins by class (secondary structure content), architecture (secondary structure orientation), topology (number and connections of secondary structures) and homologous superfamily (pro-

teins with similar structure and function) [38].

Research specific to the GPCR superfamily includes the GPCRDb's three-way classification system (by families, ligand, and species [58]) and phylogenetic classification [58, 46, 25]. The GCRDB group at EMBL also uses multiple alignments and 3-D homology modeling to classify GPCR families. (Note: the similarly named GPCRDb and GCRDB are two different GPCR databases.)

In the machine learning area, Net-ID Inc., a commercial enterprise of Dr. Pierre Baldi, is collaborating with SmithKline Beecham to build a library of GPCR HMM models. In Finland, Dr. Tommi Rintala has used genetic algorithms to predict GPCR-ligand binding from primary sequence information [47].

6.2 Classification by Family/Subfamily

A motif-recognition method known as *fingerprinting* has been implemented on the GPCR Pattern Recognition server. They have built 120 signatures (groups of motifs) that characterize conserved regions in GPCR subfamilies. The server can be used to predict subfamily membership by matching a query sequence with a known signature or against all signature candidates [4].

6.2.1 Classification by Ligand

Most prediction research on protein-ligand interaction is based on finding a good structural alignment between the ligand and a potential docking site on the protein. Algorithms to do so are described in a series of papers by Rarey and colleagues [43, 41, 42, 40, 45, 44].

The *LISSA* program [37] is a tool designed to search PDB (Brookhaven Protein Data Bank) for ligands, using a set of atomic bonding constraints. A similar, more recent tool is *ReLiBase* which searches PDB for ligand/protein interactions. It is capable of finding all ligands for a particular protein family [21]. Both LISSA and ReLiBase use structural rather than primary sequence information.

6.2.2 Classification by Active Site

I was able to find only one publically available database that specializes in protein-active site interaction, PROMISE [14]. To date, no membrane proteins are listed in this database. There is also a researcher at the University of Sheffield, U.K., Robert Gaizaukas, developing a structured database of protein active site data using Natural Language Processing technology. His work is not yet published.

The Sarnoff corporation in New Jersey is developing a proprietary database of protein binding pockets. They also claim they are about to release a collection of tools to predict GPCR structure, based on a recent breakthroughs in understanding the GPCR activation process. Details of their research are not available.

Chapter 7

Future Work

7.1 Short Term

7.1.1 Subfamily Classification

Initial results for SVM recognition of three subfamilies of Family A: serotonin receptors, histamine receptors, and muscarinic receptors, are very promising.

The GCRDB lists 24 sub-families of Family A, 13 sub-families of Family B, and 3 sub-families of Family C. Family D and Family E do not have subfamilies. According to Dr. Florence Horn, the most interesting recognition problems lie on the sub-sub-family level [23]. Classification on this resolution level includes the six sub-families of the Family A sub-family of *amine receptors*, the 21 sub-families of the Family A sub-family of *peptide receptors*, and the 3 subfamilies of the Family A sub-family of *hormone protein receptors*. There are 57 such sub-sub-families of Family A in the GCRDB.

Because there are so many of these sub-types, I plan to prioritize my experiments to focus on the classifications that are most interesting to biologists. To eliminate the

possibility of "controversial" labellings of GPCRs, the labellings of the sub-families will be taken from the annotations in the SWISS-PROT protein sequence data bank [6].

It will also be interesting to find out if reverse null model scoring significantly improves SAM-T99 subfamily recognition.

7.1.2 Object-oriented rewrite of SVM software

The Computational Biology group at University of California Santa Cruz currently has two versions of SVM software available: an experimental platform written by Tommi Jaakola, used in the experiments in this thesis, and a package written by Bill Grundy. Both versions are procedural, rather than object-oriented designs. The group needs an extensible, object-oriented version of the code that allows the user to choose:

- an optimization algorithm (steepest gradient ascent, chunking, SMO),
- a discriminant rule,
- a kernel function (linear, radial basis, polynomial),
- a normalization method

without having to rewrite the entire program.

I have written an initial version in Java. Upon completion of this thesis, I plan to migrate my experiments to this new software. This will allow me the freedom to experiment with various heurstics, faster algorithms, and feature extraction methods (such as wavelet transformation), described in Section 7.2.5.

7.2 Medium Term

7.2.1 Phylogenetic Classification

The SVM classification method outperforms FPS BLAST and Smith-Waterman by a wide margin. It also improves on the performance of SAM-T99 HMMs for subfamily data sets. To further strengthen the case for SVM classification, it would be useful to run a series of experiments that evaluate the performance of a phylogenetic classification method on the data sets described in this thesis. Possible phylogenetic classifiers are *phytree* and Bill Bruno's *Weighbor* software.

7.2.2 Classifying GPCRs by Ligand and G-protein

As described in the Background section of this thesis, GPCRs bind to ligands on the extracellular side of the cell membrane. On the cytoplasmic side, they bind G-proteins. Both kinds of binding are selective and characterize the function of a particular GPCR.

Several synthetic peptides bind specifically to loop regions and compete with G-protein binding. The sequences of these peptides would be an excellent source of negative examples for the SVM [8].

The results described in Section 4 have shown that SVMs can successfully classify GPCRs by super-family. I am currently doing experiments to verify that the method also works on sub-family classification. Because sub-families are largely organized around common ligands, good sub-family classification implies good classification by ligand. It is even possible that the method can be extended to recognize GPCRs by their binding G-proteins.

If subsequent experiments are successful, SVM classification will be a powerful tool in scanning the human genome for new GPCRs, and rapidly predicting the function of newly discovered molecules.

7.2.3 Feature extraction methods

The Fisher score vector represenation of a protein sequence is an ordered collection of features, derived from the parameters of a hidden Markov model. As described in Section 2.2.5, I did not use all the parameters of the HMM to construct my Fisher score vectors, only the match state probabilities.

Thus, it would be possible to construct even larger Fisher score vectors by deriving more features from the HMM transition probabilities. This would certainly make the process of SVM learning more computationally expensive, however it would be interesting to see whether the resulting classification performance would improve.

Another possibility is to use fewer features, and to try and identify the features that are most important. We may be able to measure feature importance using known biological facts.

For example, the amino acids involved in the extracellular ligand-binding pocket are known for many GPCRs [8]. By constructing Fisher score vectors from only those HMM match states that correspond to these amino acids, it may be possible to build a better Fisher kernel SVM. Likewise, for the case of classifying GPCRs by the G-protein bound on the cytoplasmic side of the membrane, feature extraction could be restricted to the positions of amino acids known to be involved in this interaction.

7.2.4 Weighting the kernel coefficients

Kevin Karplus has suggested using a modified optimization algorithm which computes a weight for each pair of examples in the training set, based on the kernel score of the two examples $K(x_i, x_j)$.

We consider the case of a radial basis kernel which will construct a circular hyperplane around the positive examples as shown in Figure 7.1.

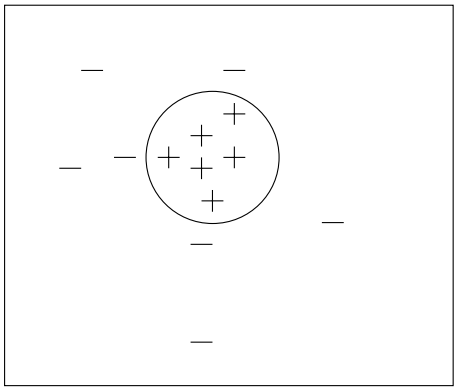


Figure 7.1: A radial basis kernel constructs a circular hyperplane to separate positive and negative examples.

Since the kernel score is inversely proportional to the distance between two examples, a large kernel score implies the examples are a small distance apart and a small kernel score implies that they are a large distance apart.

Intuitively, to construct the maximal margin hyperplane, we are most interested in only two cases:

- x_i and x_j have different labels ($y_i = 1$ and $y_j = -1$) and $K(x_i, x_j)$ is large.
- x_i and x_j are both positive examples ($y_i = y_j = 1$) and $K(x_i, x_j)$ is small.

The new algorithm would optimize the *alphas* by maximizing the difference

between positive/negative pairs and minimizing the difference between positive/positive pairs.

By giving high weight to features that are different in the positive/negative pairs and similar in positive/positve pairs, and giving low weight to features that are the same in positive/negative pairs and different positive/positive pairs, it may be possible to mathematically discover the most important features.

7.2.5 Improving SVM Classification with Wavelets

Another feature extraction method involves application of signal processing techniques to the Fisher score vectors.

If there are N match states in an HMM, and nine components per match state in the Fisher score vector (see Section 2.2.5), each component is a single point in an N point time series. The score vectors compose nine such N point time series. If a sequence fits the HMM poorly at match state x, the derivative of the likelihood of the sequence at that point is large in magnitude, and one or more of the components will have a large value at x. This produces a spike at position x in one (or several) of the nine time series. As an example, if the model favors a hydrophobic amino acid at position x and the sequence being scored has a hydrophilic amino acid at x, the Fisher score vector component for hydrophobic amino acids will take on a large value.

An interesting issue is whether signal processing of these nine time series can produce SVM input vectors that are easier to classify. Wavelet transformations of time series have been applied to neural network inputs and produced a significant improvement in the network's classification abilities [60, 63, 30].

Wavelets

A wavelet transformation is a change of basis that represents a time series (signal) as a weighted sum of two kinds of basis functions: *wavelets* and *scaling functions*. All the basis functions in the transformation are scaled and shifted versions of a single *mother* wavelet and *father* scaling function. The weight or *coefficient* of each basis function measures its similarity to a piece of the time series.

Wavelets have become extremely popular in the signal processing community because of this property of localization. Fourier transform coefficients measure the similarity of an entire time series to a wiggly complex exponential basis function. Wavelet coefficients measure similarity with respect to a tiny neighborhood around a single point in the time series. They can also measure similarity to the entire time series, because of the flexibility provided by scaling and shifting the basis functions.

I applied a wavelet decomposition to all Fisher score vectors computed from the Family A (rhodopsin and beta2-adrenergic receptor) model on Data Set ID #6 shown in Table 3.1 and Table 3.2. The method was to construct nine N-point time series from the N nine-component Fisher score vectors, where N was the number of match states in the Family A model. The significance of these time series is described Section 7.2.5.

Each time series was decomposed to $log_2 N$ resolution levels using the Daubechies 10 wavelet, and the wavelet coefficients from all resolution levels were concatenated. Finally, the concatenated coefficients were used to reassemble transformed versions of N nine-component Fisher score vectors.

The median rate of false positives for the Family A model on the above data set

was 0.0025. With wavelet transformation, this error rate decreased to 0.0016.

Due to the enormous size of Fisher score vectors in these experiments, a thorough evaluation of this method will require very efficient software, and a computing environment where on the order of 100 GBytes storage is available. (A file of Fisher score vectors often exceeds 200 MBytes for a nine-component vector and a model with 300-400 match states.) Another alternative is software able to transform the score vectors in memory alone, without resorting to disk writes.

Chapter 8

Conclusion

Both hidden Markov models and support vector machines are powerful tools to discriminate GPCRs on the basis of primary sequence information. Experiments on a series of 20 superfamily data sets and 6 subfamily datasets show that a general SAM-T99 hidden Markov model makes fewer classification errors than any other method when recognizing the GPCR superfamily. A Fisher kernel support vector machine makes the least classification errors when recognizing three medically interesting GPCR subfamilies. The popular BLAST method consistently makes the most errors on these discrimination problems. For certain data sets, Smith-Waterman scoring provides excellent classification performance with respect to the time and computer cycles required.

The SVM does not directly classify protein sequences. Rather, it learns to classify Fisher score vectors that correspond to protein sequences. The score vector is derived from a hidden Markov model of the protein category being discriminated and describes the gradient of the likelihood of a sequence, with respect to the parameters of the model.

The HMMs used in my experiments were built with two versions of the SAM-T9X

program, a smart algorithm that takes an initial user-selected training set and appends the set with its own choice of protein sequences. In approximately one-half of my experiments, the plain vanilla HMMs built with SAM-T98 were not able to well discriminate the GPCR superfamily. Upon inspection, it turned out that the the algorithm had included many non-GPCRs in its training alignments.

When Fisher score vectors were extracted from these overgeneralized models and presented to an SVM as training and test examples, the SVM algorithm was able to successfully discriminate members of the GPCR superfamily. However when SAM-T98 was upgraded to SAM-T99, I built a new series of T99 HMMs. These models proved to be the best superfamily classifiers in their own right. Fisher kernel support vector machines built on top of them not only failed to improve on their performance, they acually did worse.

SAM-T99 models were not so successful at discriminating the GPCR subfamilies. In all of my subfamily experiments, these models incorporated sequences from outside the targeted subfamily into their training alignments. Once again, when I extracted Fisher score vectors from the overgeneralized HMMs, Fisher kernel SVMs were able to discriminate the three tested sub-families extremely well.

Due to rapid progress in genetic research, approximately one new GPCR is being discovered every week. In the near future, it is likely that sequence information will be available for several thousand, as opposed to one thousand GPCRs. Because the ligands that bind to most of these new GPCRs are unknown, they are in a growing category of GPCRs known as orphans.

The current consensus is that the GPCR/ligand affinities are not detectable by

primary sequence information. Most research directed at discovering these relationships is based on solving the structures of new molecules, either through improved wet lab techniques or computerized molecular modeling.

If SVMs can be shown to successfully classify sequences into GPCR sub-families, it is likely that the technique can be expanded to automate ligand recognition on the basis of primary sequence information. The method would be inexpensive both financially and computationally, compared to wet lab techniques and sophisticated molecular modeling on supercomputer hardware.

While not eliminating the need for wet labs or structural modeling, an SVM recognition method may be able to provide a valuable filter for new molecules and greatly speed up the time between discovery of a GPCR and development of a drug targetted at augmenting or inhibiting its function.

Bibliography

[1] N. N. Alexandrov, R. Nussinov, and R. M. Zimmer. Fast protein fold recognition via sequence to structure alignment and contact capacity potentials. In L. Hunter and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 53–72. World Scientific Publishing Co., 1995.

[2] S. F. Altschul, T. L. Madden, A. A. Schaffer, 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*, 25:3389–3402, 1997.

[3] M. Anthony. Probabilistic Analysis of Learning in Artificial Neural Networks: the PAC Model and its Variants. *Neural Computing Surveys*, 1, 1997.

[4] T. K. Attwood and M. E. Beck. PRINTS - A protein motif finger-print database. *Protein Engineering*, 7(7):841–848, 1994.

[5] J. M. Baldwin, G. F. X. Schertler, and V. M. Unger. An alpha-carbon template for the transmembrane helices in the rhodopsin family of G-protein-coupled receptors. *Journal of Molecular Biology*, 272:144–164, 1997.

[6] A. Barioch and R. Apweiler. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1998. *Nucleic Acids Research*, 26:38–42, 1998.

[7] Christian Barrett, Richard Hughey, and Kevin Karplus. Scoring hidden Markov models. *CABIOS*, 13(2):191–199, 1997.

[8] R. Bogolmoni. Private communication, 2000.

[9] M. Bouvier. Structural and functional aspects of G protein-coupled receptor oligomerization. CSBMCB Merck Frosst Award Address, 1997.

[10] C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, 2nd edition, 1999.

[11] S. H. Bryand and S. F. Altschul. Statistics of sequence-structure threading. *Curr. Opin. Struct. Biol.*, 5:236–244, 1998.

[12] C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition.* Kluwer Academic Publishers, 1998.

[13] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[14] K. N. Degtyarenko, A. C. T. North, and J. B. C. Findlay. PROMISE: a database of bioinorganic motifs. *Nucleic Acids Res.*, 27:233–236, 1999.

[15] M. Diekhans. Private communication, 1999.

[16] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[17] S. S. G. Ferguson. Using green fluorescent protein to understand the mechanisms of G-protein-coupled receptor regulation. *Brazilian J Med Biol Res*, 31(11):1471–1477, Nov 1998.

[18] D. Fischer and D. Eisenberg. Fold Recognition Using Sequence-Derived Predictions. *Protein Science*, 5:947–955, 1996.

[19] M. Giamporcaro. An Introduction to Support Vector Learning. unpublished article, June 1999.

[20] Willian N. Grundy. Family-based homology detection via pairwise sequence comparison. In *Int. Conf. Computational Molecular Biology (RECOMB-98)*, New York, 1998. ACM Press.

[21] M. Hendlich, F. Rippman, and G. Barnickel. ReLiBase. `http://www2.ebi.ac.uk:8081/home.html`.

[22] L. Holm and C. Sander. The FSSP database of structurally aligned protein fold families. *Nucl. Acids Res.*, 22:3600–3609, 1994.

[23] F. Horn. Private communication, 2000.

[24] F. Horn, M. Mokrane, J. Weare, and G. Vriend. G protein-coupled receptors, or the power of data. courtesy of Dr. F. Horn.

[25] F. Horn, J. Weare, M. W. Beukers, S. Horsch, A. Bairoch, W. Chen, O. Edvardsen, F. Campagne, and G. Vriend. GPCRDB: an information system for G protein-coupled receptors. *Nucleic Acids Res*, 26(1):277–281, 1998.

[26] R. Hughey and A. Krogh. SAM: Sequence alignment and modeling software system. Technical Report UCSC-CRL-95-7, University of California, Santa Cruz, Computer Engineering, UC Santa Cruz, CA 95064, 1995.

[27] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies, 1998. Unpublished, available from `http://www.cse.ucsc.edu/research/compbio/research.html`.

[28] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, Aug 1999.

[29] T. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In *Advances in Neural Information Processing Systems 11*, San Mateo, CA, 1998. Morgan Kauffmann Publishers.

[30] G. S. Kapogiannopoulos. Character Recognition Using a Biorthogonal Discrete Wavelet Transform. *Proceedings of the SPIE Conference on Signal and Image Processing IV*, 2825:384 – 393, August 1996.

[31] K. Karplus. Private communication, 2000.

[32] Kevin Karplus, Christian Barrett, and Melissa Cline. Getting the most out of hidden Markov models. In *ISMB99 tutorial*, Heidelberg, Germany, August 1999. `http://www.cse.ucsc.edu/research/compbio//ismb99.tutorial.html`.

[33] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov Models for detecting Remote Protein Homologies. *Bioinformatics*, 14(10):846–856, 1998.

[34] A. Krebs, C. Villa, P. C. Edwards, and G. F. X. Schertler. Characterization of an improved two-dimensional p22121 crystal from bovine rhodopsin. *Journal of Molecular Biology*, 282:991–1003, 1998.

[35] Pete Ludovice Lab. Simulation of G protein-coupled receptors. `http://www.chemse.gatech.edu/ pete/pjl/proj/gpcr/gpcr.html`.

[36] H. Luecke, B. Schobert, H. T. Richter, J. P. Cartailler, and J. K. Lanyi. Structure of Bacteriorhodopsin at 1.55 Angstrom Resolution. *J.Mol.Biol*, 291, 1999.

[37] S. L. Moodie and J. M. Thornton. A study into the effects of protein binding on nucleotide conformation. *Nucleic Acids Research*, 21(6):1369–1380, 1993.

[38] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. CATH- A Hierarchic Classification of Protein Domain Structures. *Structure*, 5(8):1093–1108, 1997.

[39] I. D. Pogozheva, A. L. Lomize, and H. I. Mosberg. Modeling of the d-opioid receptor transmembrane a-bundle. In P. Kaumaya and R. S. Hodges, editors, *Peptides: Chemistry, Structure, and Biology, Proceedings of the1 4th American Peptide Symposium*, pages 350–351. Mayflower Scientific, 1996.

[40] M. Rarey, B. Kramer, and T. Lengauer. Placement of medium-sized molecular fragments into active sites of proteins. *Journal of Computer-Aided Molecular Design*, 10:41–54, 1996.

[41] M. Rarey, B. Kramer, and T. Lengauer. CASP-2 experiences with docking flexible ligands using FlexX. *PROTEINS: Structure, Functions, and Genetics, Suppl*, 1(1):221–225, 1997.

[42] M. Rarey, B. Kramer, and T. Lengauer. Multiple automatic base selection: Protein-ligand docking based on incremental construction without manual intervention. *Journal of Computer-Aided Molecular Design*, 11:369–384, 1997.

[43] M. Rarey, B. Kramer, and T. Lengauer. Docking of hydrophobic ligands with interaction-based matching algorithms. Bioinformatics, in press, 1999.

[44] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. Predicting Receptor-Ligand Interactions by an Incremental Construction Algorithm. *Journal of Molecular Biology*, 261(3):470–489, 1996.

[45] M. Rarey, S. Wefing, and T. Lengauer. Time-Efficient Docking of Flexible Ligands into Active Sites of Proteins. In C. Rawlings et. al., editor, *Proceedings of the Third International Conference on Intellegent Systems in Molecular Biology*, pages 41–54, 1996.

[46] K. Rice. Evolution and Classification of G-protein Coupled Receptors. `http://phylofarm.bio.upenn.edu/ rice/receptor/receptor.html`.

[47] T. Rintala. Ligand Recognition with Genetic Algorithms. In J. T. Alander, editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and Applications*, Aug 1996. `http://www.uwasa.fi/cs/publications/2NWGA/node122.html`.

[48] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.

[49] B. Rost. Threading One-dimensional Predictions Into Three-dimensional Structures. In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, and S. Wodak, editors, *The third international conference on Intelligent Systems for Molecular Biology (ISMB)*, volume 5, pages 947–955, Cambridge, U.K., Jul 1996. AAAI Press.

[50] B. Schölkopf. *Support Vector Learning*. PhD thesis, Technische Universitat Berlin, 1997.

[51] J. Schultz, F. Milpetz, P. Bork, and C. P. Ponting. SMART, a simple modular architecture research tool: Identification of signalling domains. *Proc. Natl. Acad. Sci. USA*, 95:5857–5864, 1998.

[52] K. Sjölander, K. Karplus, M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haus-

sler. Dirichlet Mixtures: A Method for Improving Detection of Weak but Significant Protein Sequence Homology. *CABIOS*, 12(4):327–345, August 1996.

[53] T. F. Smith and M. S. Waterman. Comparison of bio-sequences. *Adv. Appl. Math*, 2:482–489, 1981.

[54] B. Scholkopf & C. Burges & A. J. Smola, editor. *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1999.

[55] E. L. L. Sonnhammer, S. R. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Research*, 26(1):320–322, Jan 1998.

[56] E. L. L. Sonnhammer, G. von Heijne, and A. Krogh. A hidden Markov model for predicting transmembrane helices in protein sequences. In J. Glasgow et. al, editor, *Proc. Sixth Int. Conf. on Intelligent Systems for Molecular Biology*, pages 175–182. AAAI Press, 1998.

[57] A. M. Spiegel. *Inborn Errors of Signal Transduction: Mutations in G Proteins and G Protein-coupled Receptors as a Cause of Disease*. Journal of Inherited Metabolic Disease, 1997.

[58] GCRDb staff. Introduction to G protein-coupled receptor families. GCRDb, the G-protein coupled receptor database. `http://www.gcrdb.uthscsa.edu/GCRFam.html`.

[59] D. Strahs and H. Weinstein. Structure-Function Relationships in the delta, kappa and mu Opoid Receptors: Modeling and Molecular Dynamics Simulations of the Receptor Structures. Department of Physiology and Biophysics, Mount Sinai School of Medicine, New York N.Y.

[60] C. M. Johnson & E. W. Page & G. A. Tagliarini. Signal Classification Using Wavelets and Neural Networks. *Proceedings of the SPIE Conference on Wavelet Applications III*, 2762:202 – 207, April 1996.

[61] V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Nauka, 1979.

[62] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

[63] M. A. Shaikh & B. Tian & M. R. Azimi-Sadjadi & K. E. Eis & T. H. VonderHaar. An Automatic Neural Network-Based Cloud Detection/Classification Scheme Using Multispectral and Textural Features. *Proceedings of the SPIE Conference on Algorithms for Multispectral and Hyperspectral Imagery II*, 2758:51 – 61, April 1996.

[64] G. Vriend. Molecular Modeling of GPCR's. *7TM*, 5, 1995.

[65] S. Watson and S. Arkinstall. *The G-protein Linked Receptor FactsBook*. Academic Press, Harcourt Brace & Company, 1994.