

Classifying G-protein coupled receptors with support vector machines

This is a preprint of an article to appear in *Bioinformatics* copyright 2001.

Rachel Karchin^{†*} Kevin Karplus[‡] David Haussler^{†§}

[†]Department of Computer Science, University of California, Santa Cruz, CA 95064, USA

[‡]Department of Computer Engineering, University of California, Santa Cruz, CA 95064, USA

[§]Howard Hughes Medical Institute, University of California, Santa Cruz, CA 95064, USA

August 20, 2001

Abstract

Motivation: 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. We discuss the relative merits of various automated methods for recognizing *G-protein coupled receptors (GPCRs)*, a superfamily of cell membrane proteins. GPCRs are found in a wide range of organisms and are central to a cellular signalling network that regulates many basic physiological processes. They are the focus of a significant amount of current pharmaceutical research because they play a key role in many diseases. However, their tertiary structures remain largely unsolved. The methods described in this paper use only primary sequence information to make their predictions. We compare a simple nearest neighbor approach (BLAST), methods based on multiple alignments generated by a statistical profile *hidden Markov model*, and methods, including *support vector machines*, that transform protein sequences into fixed-length *feature vectors*.

Results: The last is the most computationally expensive method, but our experiments show that, for those interested in annotation-quality classification, the results are worth the effort. In two-fold cross-validation experiments testing recognition of GPCR subfamilies that bind a specific ligand (such as a histamine molecule), the errors per sequence at the *minimum error point* (MEP) were 13.7% for multi-class SVMs, 17.1% for our SVMtree method of hierarchical multi-class SVM classification, 25.5% for BLAST, 30% for profile HMMs, and 49% for classification based on nearest neighbor feature vector (kernNN). The percentage of true positives recognized before the first false positive was 65% for both SVM methods, 13% for BLAST, 5% for profile HMMs and 4% for kernNN.

Availability: We have set up a web server for GPCR subfamily classification based on hierarchical multi-class SVMs at

<http://www.soe.ucsc.edu/research/compbio/gpcr-subclass>

By scanning predicted peptides found in the human genome with the SVMtree server, we have identified a large number of genes that encode GPCRs.

A list of our predictions for human GPCRs is available at

http://www.soe.ucsc.edu/research/compbio/gpcr_hg/class_results

We also provide suggested subfamily classification for 18 sequences previously identified as unclassified Class A (rhodopsin-like) GPCRs in GPCRDB (Horn et al., 1998), available at

http://www.soe.ucsc.edu/research/compbio/gpcr/classA_unclassified/

1 Introduction

Support vector machines (SVMs) are a class of statistical learning algorithms whose theoretical basis was first presented by V. Vapnik in 1979 (Vapnik, 1979). During the 1990s, they became extremely popular in the machine-learning community (Cristianini and Shawe-Taylor, 2000). When *SVMs* are applied to the simplest learning problem,

*To whom correspondence should be addressed. Email to: rachelk@soe.ucsc.edu

two-class *pattern recognition*, the 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 our experiments, the data to be classified are protein sequences of the G-protein coupled receptor (GPCR) superfamily (Watson and Arkinstall, 1994). Specifically, we want to recognize small subfamilies of GPCRs that bind the same ligand. This requires extension of the two-class problem to a k -class problem. We have chosen the simplest approach to multi-class SVMs by training k one-to-rest classifiers.

GPCRs were selected because an enormous amount of current pharmaceutical research is aimed at understanding their structure and function (Horn et al., 2000). 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 (Bouvier, 1997). These proteins are very important to understanding human physiology and disease, but their tertiary 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 (2.8Å) (Palczewski et al., 2000).

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 Project and other genome projects (fruitfly, worm, mouse, etc.). A method for highly accurate, annotation-quality, sequence-based prediction of GPCR function has considerable practical value for both research biologists and pharmaceutical companies. We will present the results of a series of controlled experiments to argue that multi-class SVMs are well suited to this problem. Out of all the methods we tested, SVMs had the highest accuracy in recognizing GPCR subfamilies.

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 mathematical tool known as a *kernel function*, which measures the similarity between two examples. Compared to BLAST (Altshul et al., 1990) and hidden Markov models (Durbin et al., 1998), this method is computationally expensive. In our experiments, an initial step is required in which each protein sequence is transformed into a fixed-length feature vector. Next, we train a two-class SVM for each GPCR subfamily. Each of these SVMs learns to distinguish between subfamily members and non-members by making several passes through a training set and using the kernel function to compute a weight for each sequence. The result is a vector of trained weights that will be used to make predictions for new examples (for more detail see Appendix A).

To justify the expense, SVMs must be significantly more accurate than simpler classifiers. We describe results of two-fold cross-validation experiments that compare multi-class SVMs with two popular classification methods: BLAST and profile hidden Markov models. We also compare the SVMs with a nearest-neighbor method based on kernel scores and a fast SVM approximation method that we call SVMtree. We have found that HMMs built with the SAM-T2K protocol (Karplus et al., 2001) are better GPCR discriminators than SVMs at the class (superfamily) level. However, SVMs make significantly fewer errors than other methods when applied to the problem of discriminating subfamilies of GPCRs, particularly when methods are evaluated by their specificity, or performance in the low false-positive range. Detailed results, including ROC plots, for all our experiments are presented in Section 3.

Our method of support vector machine classification is based on the work done by Jaakkola, et. al. on SCOP superfamily discrimination (Jaakkola et al., 2000). However, the present paper applies the method to the very different problem of specific subfamily classification, introduces several new ideas, and addresses difficulties reported by other researchers who attempted to reproduce the results described in the prior work. A web server that implements the SVMtree algorithm for GPCR subfamily classification is available for general use at <http://www.soe.ucsc.edu/research/compbio/gpcr-subclass>.

2 Methods

Although several research groups, including our own, have published papers on the results of support vector classification of proteins, none have addressed the issue of whether all the machinery of SVMs is necessary for such classification. To answer this question, our approach was to systematically test a series of progressively more complex methods and then evaluate accuracy and computational expense of each method.

Since it was not possible to test all existing protein classification methods, we selected a representative method at each level of complexity. To ensure that our comparisons were as fair as possible, methods that involved database

searching (BLAST) were restricted to searching the same datasets used as training and test sets by the model-building methods. We were unable to rectify one built-in inequality when comparing HMMs and SVMs. Because SVMs are trained on both positive and negative examples and HMMs are trained on positive examples only, SVMs inherently contain more information about the positive examples being recognized.

The simplest classification methods work directly with sequence information and classify a *target sequence* according to the annotation of its nearest known neighbor. To find the nearest neighbor, databases of annotated sequences are searched and database sequences ranked by various measures of sequence similarity.

The extremely popular and fast BLAST (Altshul et al., 1990) method uses pairwise local alignments to measure sequence similarity. We selected WU-BLAST (WU-BLAST, 2001) to represent the nearest neighbor sequence approach to classification.

A more sophisticated nearest-neighbor approach builds a library of statistical models for protein classes of interest. The target sequence is scored against all models in the library and classified according to the model that gives the best score to the sequence. We used the SAM-T2K hidden Markov model program (Hughey and Krogh, 1995; Karplus et al., 1998; Karplus et al., 2001) to represent this approach.

Our work with the transformation from protein sequence into Fisher score vector space (described in Appendix A) introduced the possibility of an alternate approach to nearest-neighbor classification. T. Mueller suggested (Mueller, 2000) that the encoding of a protein sequence into a Fisher score vector might contain sufficient information to classify the sequence, and that the additional machinery of support vector machine training is unnecessary. We were interested to see whether nearest-neighbor classification was better in Fisher score vector space than in sequence space, and whether we might get highly accurate classification without using SVMs. For this experiment, both the target sequence and database sequences used in our BLAST experiments were transformed into Fisher score vectors and a radial basis kernel function was used to find the nearest neighbor of each Fisher score vector in the test set. We call this method *kernel nearest neighbor* (kernNN).

We also tested the most computationally expensive method, support vector machine classification of protein sequence feature vectors, using the Fisher score vector transformation (SVM). For these experiments, we built a library of support vector machines for each protein class of interest, and scored each test sequence against the entire library, as in the HMM experiments. A more efficient SVM method (SVMtree) is described in Section 2.2.

2.1 GPCR Superfamily Recognition

Jaakkola et al. report excellent results at superfamily classification with SVMs (Jaakkola et al., 2000). They tested discrimination of SCOP superfamilies with n -fold cross validation, by training on $n - 1$ families within each superfamily of interest and testing on the held-out family.

The experiments compared two-class SVMs trained on Fisher score vectors (FSVs), profile HMMs built with SAM T98, and WU-BLAST run with the Family Pairwise Search algorithm (FPS) (Grundy, 1998).

Our first experiments aimed to duplicate their methods on a similar problem, recognition of the GPCR superfamily. According to the GPCRDB information system (Horn et al., 1998), the superfamily can be subdivided into five major classes (roughly analogous to SCOP families): Class A (receptors related to rhodopsin and the adrenergic receptor), Class B (receptors related to the calcitonin and PTH/PTHrP receptors), Class C (receptors related to the metabotropic receptors), Class D (receptors related to the pheromone receptors), and Class E (receptors related to the cAMP receptors). The classes share $\geq 20\%$ sequence identity over predicted transmembrane helices (Horn et al., 1998).

In data sets suggested by Drs. Gert Vriend and Florence Horn (Vriend, 1999; Horn et al., 1998), the positive examples were 692 sequences from Class A, 56 from Class B, 16 from Class C, 11 from Class D and 3 from Class E. The data sets also included 99 decoy negative examples: 18 archaea rhodopsins and 80 G-protein alpha domains. We added 2425 additional negative examples taken from the SCOP version 1.37 PDB90 domain database. These are easier negative examples than the archaea rhodopsin 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.

The Jaakkola experiments tested recognition of 33 SCOP superfamilies using positive training examples from $n - 1$ of the families in the superfamily and then testing on the n th family (Jaakkola et al., 2000). We used a similar protocol by positive training on GPCRs from Class B, Class C, Class D and Class E, then testing on Class A, positive training on Class A, Class C, Class D and Class E, then testing on Class B, and so forth. As in the Jaakkola paper, the score for each test sequence in a given class was computed by averaging its scores with respect to the models of the other four classes. Results appear in Section 3.

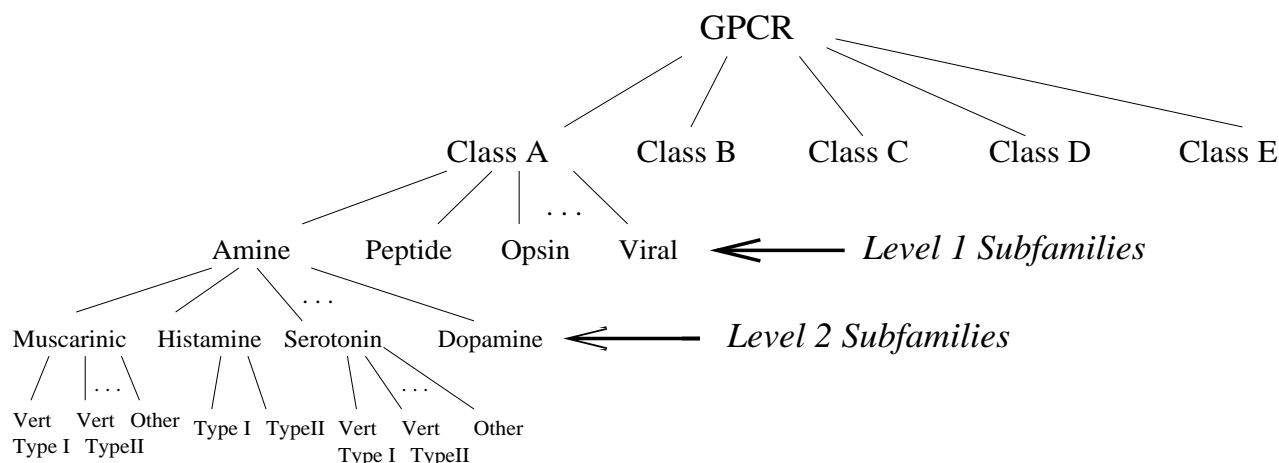


Figure 1: Portion of the GPCR family tree showing the five main classes of GPCRs, a few of the Class A subfamilies, a few of the subfamilies of amine receptors, and a few types of amine receptors. We are interested in discriminating the subfamilies labeled as Level 1 and Level 2. These classifications are taken from the GPCRDB information system (Horn et al., 1998).

2.2 GPCR Subfamily Recognition

Because the main goal of this work is to develop a method to determine GPCR function from sequence information, our main interest is in subfamily rather than superfamily recognition. While superfamily discrimination is best done by methods that can generalize the features shared by a diverse group of examples, subfamily discrimination requires separating examples that may differ only slightly. The problem of recognizing GPCR subfamilies is compounded by the fact that the subfamily classifications in GPCRDB are defined chemically (according to which ligands the receptor binds) rather than by sequence homology. In fact, many subfamilies share strong homology with other subfamilies. Many also contain distinct sub-types (Types) that may be the result of convergent evolution. An example is the Type I and Type II histamine receptors, which both bind the same ligand (a histamine molecule), but do not resemble each other with respect to amino acid sequence in many regions. Finally, many GPCR subfamilies contain few known members, making it difficult to train high-quality models.

The GPCRDB information system (Horn et al., 1998) organizes the GPCR superfamily into a hierarchy of *classes*, *class subfamilies*, *class subfamily subfamilies*, and *types*. A partial view of the GPCR family tree is shown in Figure 1.

We concentrated our experiments on recognition of subfamilies within Class A and Class C. When these datasets were assembled in December 2000, Class A dominated the GPCRDB, accounting for 84% of all full-length sequences in the database. (As of August 2000, 77% of the GPCRDB sequences are Class A members.) Class C was added to our experiments at the request of Dr. Susan Sullivan of NIH (Sullivan, 2000). In Dec. 10, 2000, GPCRDB contained 79 Class A subfamilies. On the GPCR family tree, 15 of these are Class A children and 64 are grandchildren. We will denote the children as *Level 1* subfamilies and the grandchildren as *Level 2* subfamilies. Class C contained 10 subfamilies (4 children and 6 grandchildren). Level 2 subfamilies are further subdivided into types, which are specific both in terms of function and organism class, such as the “Vertebrate Type I muscarinic receptor”. Many types contain only one or two sequences, and others are “grab-bags” of left-overs, which don’t fit into any sub-grouping. For these reasons, we decided to focus our cross-validation experiments on GPCR subfamilies, rather than types.

The n -fold cross validation method used in our superfamily experiments was problematic for subfamily experiments, due to the large number of subfamilies, many of which are sparsely populated. An n -fold cross-validation split requires n members per subfamily, and it is desirable to have two or more sequences to train a high-quality model so that $2n$ members would be needed. We decided on two-fold cross-validation for the subfamily experiments. Subfamilies that contain only one known member could not be tested, although models were built for them for our web-based classifier.

We designed our subfamily experiment protocol to match the way our methods might actually be used to predict the function of a sequence whose class is unknown. Rather than performing 89 two-class experiments: is it an

amine receptor? is it a chemokine receptor? and so forth, we perform a single multi-class experiment for each test sequence.

In the case of BLAST, a set of training sequences is scored with respect to a query sequence of interest, and the query is classified according to the annotation of the training sequence with the best E-value. The algorithm is implicitly multi-class, since the annotation can be any protein class found in the set.

It makes sense to compare BLAST with model-building methods that are also multi-class, in which the query sequence is scored against a library of models for the classes of interest.

In these experiments, subfamily members are considered positive examples and negative examples are GPCRs from all other subfamilies in the class, GPCRs from the remaining four classes, archaea rhodopsins (GPCR precursors), and G-alpha proteins. We did not include the easy negative examples from SCOP in either the training or test sets.

According to our two-fold cross validation protocol, we randomly partition the data into two non-overlapping sets, arbitrarily named *set0* and *set1*. Because most of our test sequences can be classified with respect to either Level 1 or Level 2, they have two correct subfamily labellings. We tested each method’s Level 1 and Level 2 subfamily recognition separately.

To evaluate BLAST, rather than using FPS (Grundy, 1998), we BLASTed all GPCRs in *set0* against *set1*, and vice versa. In this context, the sequences used as query sequences to BLAST are the test set and the sequences in the BLAST database are equivalent to the training set of a model-building method. The BLAST E-values received by each test sequence were sorted and errors counted by sweeping a threshold over the sorted E-values. At each threshold, all sequences with better scores are positives, and all sequences with worse scores are negatives. Statistics were computed by summing over both test sets.

These sorted lists of BLAST hits contain many sequences from a single subfamily, and consequently we modified our ROC (Receiver Operating Characteristic) analysis to avoid bias against BLAST. As an example, consider the following situation. We classify a test sequence from the amine receptor subfamily and find that there are three sequences in the training set from the viral receptor subfamily with better scores than the best-scoring amine receptor. This would count as three false positive errors, but for the same HMM or SVM experiment, in which the test sequence is scored against a model library, it would count as only one false positive error (viral receptor model score better than amine receptor model score). To eliminate the bias, we count only the best-scoring member of any subfamily in our BLAST ROC analysis. Note that it is still possible to have several false positives from one query if several subfamilies score better than the correct one.

For the HMM experiments, we built a library of HMMs for the 89 subfamilies of Class A and Class C GPCRs according to GPCRDB (Dec. 2000 release). Each subfamily got two models: one trained on subfamily members from *set0* and the other on subfamily members from *set1*. We used SAM-T2K to build the models, setting the homologs parameter to members of the subfamily of interest and not specifying a seed. Then, the sequences in *set0* were scored against the *set1* model library with SAM’s hmmscore program, and vice versa.

Kernel nearest neighbor (kernNN) classification was evaluated with a protocol analogous to the BLAST experiments. All GPCRs in *set0* were scored against *set1*, and vice versa, but with radial basis kernel functions in place of BLAST similarity score. The Fisher score vector transformations were done carefully. Our method requires that kernel scores be computed between Fisher score vectors derived from the same HMM, and we wanted to ensure that test examples were not contaminated with information from the other set. Therefore, each test example from *set0* was transformed with HMMs trained on *set1* and scored against a “training set” of *set1* Fisher score vectors transformed with *set1* HMMs (and vice versa). The HMMs were taken from the subfamily HMM library described previously. ROC analysis was done exactly as in the BLAST experiments.

In our multi-class SVM experiments, two sets of Fisher score vectors (FSVs) were used to train a library of SVMs, producing two SVMs for each of the 89 GPCR subfamilies. As in the kernNN experiments, the *set0* FSV training sets were transformed with *set0* HMMs, and the *set1* FSV training sets were transformed with *set1* HMMs. Likewise, all test examples were first transformed with HMMs from the opposite set and then scored against the SVMs trained on the opposite set.

There were a few idiosyncracies in the subfamily data that are worth mentioning. The test sets for Level 1 and Level 2 experiments are slightly different, because there are Level 1 subfamilies that don’t have any children on the GPCR family tree and Level 2 subfamilies that contain only one member. Therefore, some test sequences can be classified with respect to Level 1 but not Level 2. The test sets were pruned by hand to eliminate such unclassifiable sequences for each experiment. We ended up with 19 testable Level 1 subfamilies (1267 sequences) and 68 testable Level 2 subfamilies (1171 sequences).

| Method | Average errors per sequence at the MEP | Coverage |
|-------------|---|----------|
| SAM-T99 HMM | 0.04% | 98% |
| SVM | 0.22% | 72% |
| FPS BLAST | 6.82% | 3% |

Table 1: Class recognition results of FPS BLAST, profile HMMs built with SAM-T99 and one-to-rest SVMs trained on Fisher score vectors on a test set of 692 Class A GPCRs. The minimum error point (*MEP*) is the score threshold where a classifier makes the fewest errors (both false positives and false negatives). Coverage is the percent of true positives recognized before the first false positive.

Because multi-class SVM is so expensive, we were motivated to design a faster algorithm for SVM classification, which looks only at a subset of possible subfamilies for a test sequence. We can take advantage of the hierarchy of the GPCR family tree by first scoring a test sequence against the five GPCR Classes (with an HMM, the most accurate method for class discrimination). Next, the Level 1 children of the class with the best HMM score are evaluated with a set of one-to-rest SVMs. Finally, the Level 1 subfamily with the largest discriminant has its children evaluated with one-to-rest SVMs, and the test sequence is predicted to belong to the Level 2 subfamily that reports the highest discriminant. We call this method *SVMtree*. Because of the significant performance advantage, we have chosen to implement SVMtree on our GPCR subfamily classification webserver at <http://www.soe.ucsc.edu/research/compbio/gpcr-subclass>.

Results of all our experiments are detailed in Section 3. For those who wish to reproduce these experiments, the datasets described in this section are available at http://www.soe.ucsc.edu/research/compbio/gpcr/subfamily_seqs and http://www.soe.ucsc.edu/research/compbio/gpcr/superfamily_seqs.

3 Results

3.1 GPCR Superfamily Recognition

We compared the classification accuracy of *FPS BLAST* (WU-BLAST with Family Pairwise Search (Grundy, 1998)), profile HMMs built with SAM-T99, and one-to-rest SVMs trained on Fisher score vectors. Results on a test set of 692 Class A GPCRs (and negative examples listed in Section 2.1) are shown in Table 1. Two statistics are shown for each method. *Coverage*, the percent of true positives recognized before the first false positive error, measures the sensitivity of a classifier. Average errors per sequence at the minimum error point (*MEP*) provides a fair comparison of both the sensitivity and selectivity of different methods, each of which may achieve its best performance at a different score threshold. The MEP is the score threshold where a classifier makes the fewest errors of both kinds (false positives plus false negatives).

A ROC plot of the number of false positives vs. coverage for Class A discrimination is shown in Figure 2. Results for the remaining four GPCR classes were similar (ROC plots are available in *Classifying G-Protein Coupled Receptors with Support Vector Machines*, June 2000, at <http://www.cse.ucsc.edu/research/compbio/research.html>) (Karchin, 2000)).

To confirm that profile HMMs discriminate Class A GPCRs with a very low false positive rate (including those not built with SAM software), we also tested Class A recognition with the PFAM (Sonnhammer et al., 1998a) library’s HMM for Class A, *7tm_1*, built with HMMER (<http://hmmer.wustl.edu>). This model was trained on a carefully selected set of Class A sequences, so we were unable to repeat our five-fold cross-validation protocol. (The four SAM-T99 HMMs that we tested for Class A recognition were trained on Class B, Class C, Class D and Class E). When we scored our Class A test set with the *7tm_1* model, it recognized 99.7% of the true positives before the first false positive error. If we consider only sequences which score above the PFAM trusted cutoff for this model, it recognizes 99.3% of the true positives.

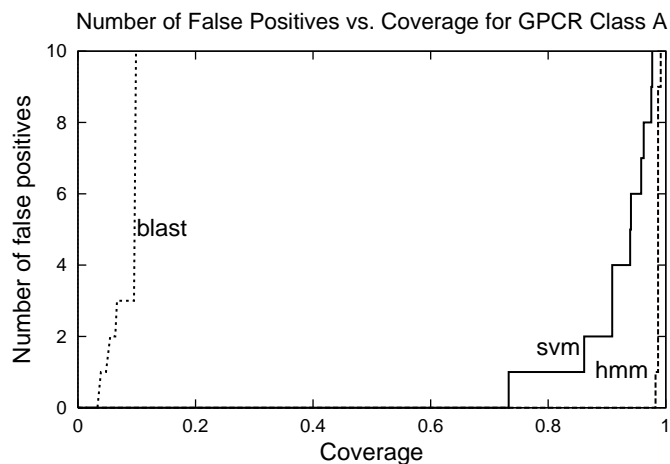


Figure 2: Number of false positives vs. coverage (true positives/692) on recognition of 692 Class A GPCRs by BLAST, SAM-T99 hidden Markov models and two-class support vector machines. Methods are compared by the percentage of correct classifications (true positives) in the low false positive range. Positive training examples used in these experiments were GPCRs from Class B,C,D, and E. The support vector machines were also trained on negative examples of G-protein alpha domains, archaea rhodopsins, and all domains in the SCOP version 1.37 PDB90 database.

3.2 GPCR Level 1 Subfamily Recognition

The results of our two-fold cross validation experiments classifying 1267 GPCR sequences from Class A and Class C into Level 1 subfamilies are shown in Table 2.

A ROC plot of the number of false positives vs. coverage for Level 1 subfamily discrimination is shown in Figure 3.

While SAM-T99 HMMs were the best method for recognition of the GPCR superfamily, the very similar SAM-T2K method did poorly on our subfamily datasets. It is not surprising that HMMs that excel at recognizing the weak similarities between superfamily members are too generalized for good subfamily recognition.

3.3 GPCR Level 2 Subfamily Recognition

The results of our two-fold cross validation experiments classifying 1171 GPCR sequences from Class A and Class C into Level 2 subfamilies appear in Table 3.

| Method | Average errors per sequence at the MEP | Coverage |
|-------------|--|----------|
| SVM | 11.6% | 48% |
| BLAST | 16.7% | 29% |
| SAM-T2K HMM | 30.1% | 6% |
| kernNN | 36.0% | 38% |

Table 2: Level 1 subfamily recognition results of one-to-rest SVMs and SVMtree, BLAST, profile HMMs built with SAM-T2K and kernel nearest neighbor on a test set of 1267 Class A and Class C GPCRs. Because there is not much difference between the SVMtree method and full multiclass SVM for Level 1, we report a single result for both. MEP and coverage are explained in Table 1.

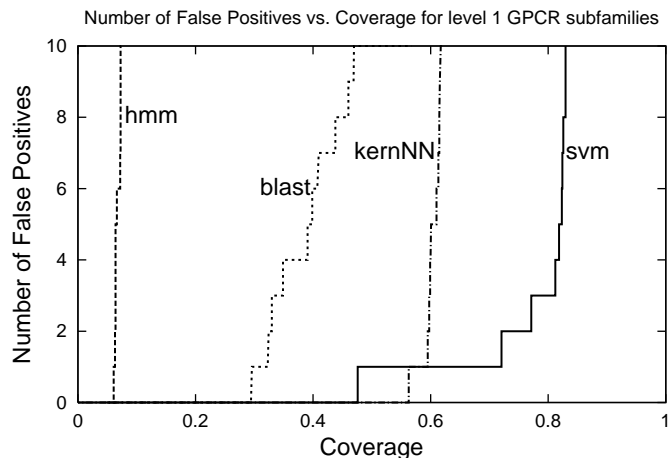


Figure 3: Number of false positives vs. coverage (true positives/1267) on recognition of Level 1 GPCR subfamilies for 1267 GPCR sequences in Class A and Class C. Methods are compared by the percentage of correct classifications (true positives) in the low false positive range. Positive training examples used in these experiments were members of the GPCR Level 1 subfamily being recognized. The support vector machines were also trained on negative examples of GPCRs from Class A and Class C not in the Level 1 subfamily, G-protein alpha domains, and archaea rhodopsins. The fast SVMtree method is not shown since results were almost identical to that of full SVM classification.

| Method | Average errors per sequence at the MEP | Coverage |
|-------------|--|----------|
| SVM | 13.7% | 65.0% |
| SVMtree | 17.1% | 65.0% |
| BLAST | 25.5% | 13.3% |
| SAM-T2K HMM | 30.0% | 5.0% |
| kernNN | 49.0% | 4.0% |

Table 3: Level 2 subfamily recognition results of one-to-rest SVMs, SVMtree, BLAST, profile HMMs built with SAM-T2K and kernel nearest neighbor on a test set of 1171 Class A and Class C GPCRs. MEP and coverage are explained in Table 1.

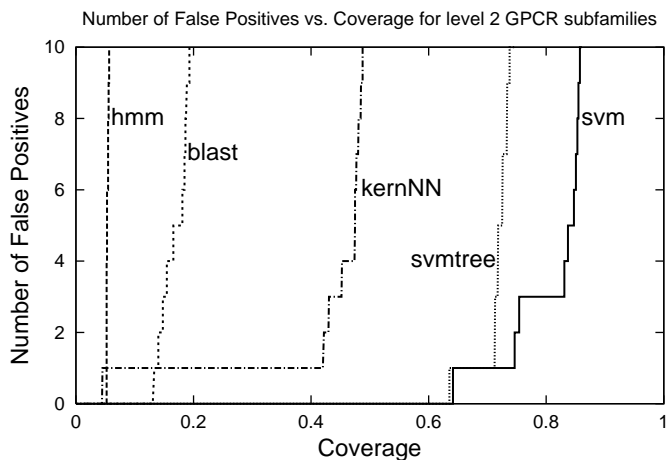


Figure 4: Number of false positives vs. coverage (true positives/1171) on recognition of Level 2 subfamilies for 1171 GPCRs from Class A and Class C. Methods are compared by the percentage of correct classifications (true positives) in the low false positive range. Positive training examples used in these experiments were members of the GPCR Level 2 subfamily being recognized. The support vector machines were also trained on negative examples of GPCRs from Class A and Class C not in the Level 2 subfamily, G-protein alpha domains, and archaea rhodopsins.

A ROC plot of the number of false positives vs. coverage for Level 2 subfamily discrimination is shown in Figure 4.

3.4 Computational Expense

A summary of the time and space expense for each method appears in Table 4. We report single sequence classification time and model library build times for an 800MHz Pentium III PC running RedHat Linux 7.0.

4 Discussion

Although SVMs are not the best method for identifying GPCRs at the class (superfamily) level, they make significantly fewer errors than WU-BLAST and SAM-T2K profile HMMs when applied to the problem of discriminating both Level 1 and Level 2 subfamilies of GPCRs. They make fewer errors of both kinds (false positive and false negative) at the minimum error point (MEP) than the other methods and recognize a significantly larger number of true positives before making the first positive error, as shown in our ROC plots.

SVMs also perform better than simple kernel nearest neighbor classifiers (kernNN), indicating that the SVM algorithm is worth the added computational expense. It does appear that there is an information gain involved in the transformation of a protein sequence into a Fisher score vector, since kernNN makes fewer false positive errors than WU-BLAST in the low false positive range when the two methods are tested on identical data sets. Performance in this range improves further when the SVM computes a smoothed boundary between positive and negative examples in high-dimensional space and uses the boundary to classify a feature vector, as opposed to kernNN's simply classifying feature vectors with the annotation of their nearest neighbor in high-dimensional space.

HMMs, kernNN, and SVMs classify Level 1 and Level 2 with similar accuracy, but WU-BLAST classifies Level 1 much better than Level 2. This may be due to the way BLAST scores sequence similarity only with respect to the most similar regions in a pair of sequences (MSP maximal segment pair scores) (Altshul et al., 1990). Many Level 2 subfamilies with the same Level 1 parent contain lengthy regions of high sequence similarity, making them difficult to discriminate strictly on the basis of MSP scoring. Figure 5 shows a serotonin receptor misclassified as an octopamine receptor by BLAST in our experiments. Both sequences are members of the Level 1 amine receptor subfamily, but different Level 2 subfamilies. Inspection of the alignment shows highly similar regions common to

| | Database and model library requirements | | | | Total build time (hours) | Disk space for build (MB) | Final disk space (MB) | Classification time for one sequence (sec) |
|---------|---|------|----------------------|------|--------------------------|---------------------------|-----------------------|--|
| | GPCRDB sequences | HMMs | Fisher score vectors | SVMs | | | | |
| BLAST | X | | | | 19E-4 | 0.68 | 0.68 | 2 |
| HMM | X | X | | | 9.90 | 16.26 | 16.26 | 10 |
| kernNN | X | X | X | | 15.24 | 7692.78 | 7692.78 | 1047 |
| SVMtree | X | X | X | X | 24.73 | 9014.96 | 1338.44 | 51 |
| SVM | X | X | X | X | 24.73 | 9014.96 | 1338.44 | 173 |

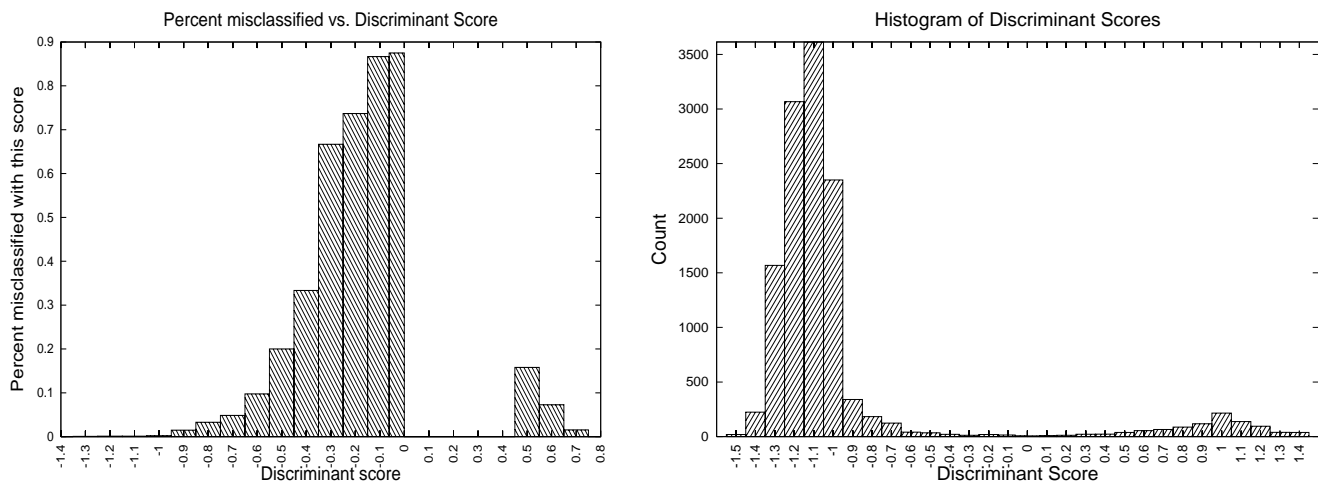
Table 4: Computer time and space expense of methods we tested on GPCR subfamily classification. Two-fold cross-validation testing of the methods required 1418 sequences, 182 HMMs, 252,404 Fisher score vectors and 178 subfamily support vector machines. For BLAST, the sequence data is stored in a special format which takes approximately the same disk space as the raw sequence data. Reported disk space for the library builds is a sum of requirements marked with an X in the columns to the left. All methods except BLAST require an HMM library as part of their final disk space requirements. Note that final disk space for the SVM methods is substantially smaller than kernNN because only the support vectors and their final weights are saved, and our implementation stores them in binary format. The long classification time reported for the kernNN method is due to the fact that it must search through the entire Fisher score vector library (number of sequences times number of HMMs). The SVM method only searches the support vectors (see Appendix A), and SVMtree searches only a subset of the support vectors (see Section 2.2). Library build time and time to classify one sequence into a GPCR subfamily are given for an 800MHz Pentium III PC running RedHat Linux 7.0.

both sequences. Note that the similarities are mainly on the inside end of transmembrane helices and inside the cellular membrane, not outside where ligand recognition must occur.

It is time-consuming to classify a protein sequence with the multi-class SVM we have described, using one-to-rest classifiers for all its potential GPCR subfamilies. A good approximation to this high-quality classification can be achieved with the SVMtree method, which skips classifiers whose parents in the GPCR family hierarchy get poor discriminant scores. As shown in Figure 6(a), SVMtree discriminant scores farthest from zero have the highest confidence. We can also see that SVMtree has an extremely low rate of false positives. Out of 12,601 classification trials in our cross-validation experiments, there were only eleven false positive errors. These include Q89609, which was predicted as a peptide receptor but labeled in GPCRDB as a Class A Orphan, and two sequences which were predicted as rhodopsin vertebrate but labeled as rhodopsin Other (OPSP_COLLI, a pigeon rhodopsin, and Q9W6K3, a chameleon rhodopsin). We believe it is possible that Q89609, which receives a discriminant score in the range of 0.7, may actually be a peptide receptor. The remaining eight false positives are the four members of the rdc receptor subfamily, each of which was misclassified twice in our experiments, both as glucocorticoid-induced and as ebv-induced receptors. All three subfamilies are Class A Orphans, and the glucocorticoid-induced and ebv-induced receptor subfamilies contain only one known member apiece, making it difficult to train our models to discriminate them. Figure 6(b) shows the distribution of all SVMtree discriminant scores reported in our two-way cross-validation experiments. SVMtree does not make any errors for discriminants greater than 0.75 or less than -1.05. A total of 74 sequences received scores between 0.0 and 0.4, and all were correctly classified.

We recently applied our SVMtree webserver to a list of predicted peptides in the human genome produced by Affymetrix and to a set of peptides currently in Swissprot that GPCRDB has not been able to classify beyond general inclusion in Class A. Although we do not have the resources available at our lab to do the wet-lab work (expression and binding assays) that could confirm these predictions, we hope that others will be encouraged to do so.

If we consider all hits getting an E-value of 1.0 or better with respect to our HMMs for Class A, Class B, Class C and Frizzled-smoothened family GPCRs (a threshold where the expected number of false positives is approximately one in twenty-thousand), we predict 275 genes that encode Class A, 52 genes that encode Class B, 41 genes that encode Class C and 10 genes that encode Frizzled-smoothened family GPCRs in the most recent set of Affymetrix-predicted human genes (Kent, 2001). (If a gene gets an E-value better than 1.0 for more than one HMM, we assign it to the HMM that gives it the best E-value.) Out of these 345 putative GPCRs, we have support vector machine



(a) Percent misclassified vs. discriminant score using a fixed threshold of zero. The percentages are calculated as follows. For each bin in the histogram, we sum the false positives (negatively labeled examples that receive discriminant scores greater than zero) and false negatives (positively labeled examples that receive discriminant scores less than or equal to zero) within this discriminant range, and divide by the total number of sequences that fall into the discriminant range.

(b) Histogram of discriminant scores for SVMtree Level 2 subfamily test sets.

Figure 6: Percent misclassified vs. discriminant score for SVMtree Level 2 subfamily predictions and overall distribution of discriminant scores in the test sets for our two-way cross-validation experiments. Note that these results give a lower bound for the accuracy of the webserver classifiers used to make the predictions in Table 5, because the SVMs evaluated here were trained on only one-half of our GPCRDB data (see Section 2.2). The webserver classifiers were trained on all of the GPCRDB data, so they are likely to make fewer errors.

subfamily predictions for genes encoding 183 Class A, 19 Class B, 12 Class C and 7 Frizzled-smoothened family peptides (not counting alternatively spliced variants). A complete list of our class and subfamily predictions for these genes is available at: http://www.soe.ucsc.edu/research/compbio/gpcr_hg/class_results.

Table 5 presents a partial summary of our predictions for the GPCRDB set. A full list is available at: http://www.soe.ucsc.edu/research/compbio/gpcr/classA_unclassified.

In the future, we would like to apply a similar tree-based approach to SVM training. An alternative to building the Level 1 SVMs from scratch would be to retrain the class SVMs, refining the support boundary that has already been computed for each subfamily’s parent. Likewise, the Level 2 SVMs would be built by retraining the Level 1 SVMs. This would have the effect of filtering out easier distinctions between examples higher on the hierarchy prior to SVM training and concentrating training on the most interesting distinctions between examples.

SVMs do not identify the features most responsible for class discrimination, but we believe SVM performance could be further improved by using feature vectors that encode only the most relevant features. We are interested in developing methods that are capable of identifying the key residues involved in ligand binding and extracting features from only these key residues, rather than the full length of each GPCR sequence. To discover the locations of binding residues, we plan to use unsupervised algorithms for learning which features best discriminate GPCR subfamilies, in combination with biological knowledge of each subfamily’s transmembrane topology.

| ID | Level 1 subfamily | discriminant score |
|--------|-------------------|--------------------|
| Q9VEG1 | amine | 0.92495406 |
| Q9VCZ3 | amine | 0.8089386 |
| Q9VN38 | amine | 0.77614653 |
| Q9VG57 | amine | 0.7631041 |
| Q24511 | amine | 0.74767226 |
| Q9VEG2 | amine | 0.50782907 |
| Q9NHA4 | peptide | 0.48891824 |
| Q9VTU7 | rhodopsin | 0.45028937 |
| Q9NHF3 | amine | 0.42113715 |
| Q9WUK7 | rhodopsin | 0.40932548 |
| Q9VRM0 | peptide | 0.40187144 |
| O62059 | peptide | 0.3542946 |
| Q9VE32 | amine | 0.34589463 |
| Q9NSD7 | peptide | 0.3192128 |
| Q9Y344 | rhodopsin | 0.3102843 |
| Q9N324 | peptide | 0.28772765 |
| Q9NJS6 | amine | 0.24319479 |
| Q9Z0T7 | peptide | 0.11519429 |

Table 5: Some of our predictions for unclassified Class A (rhodopsin-like) sequences found in GPCRDB as of May 2001. Each of the sequences was rejected by all Level 2 subfamily classifiers underneath its predicted Level 1 subfamily, indicating that the sequences may belong to novel Level 2 subfamilies. A detailed list of these predictions is available at http://www.soe.ucsc.edu/research/compbio/gpcr/classA_unclassified.

Acknowledgments

This work was supported in part by NSF grant DBI-9808007 and a National Physical Sciences Consortium graduate fellowship. The experiments were run on a 100-node Linux cluster of 800-MHz Dell PC's, provided by Dr. Patrick Mantey and Dr. M.R.C. Greenwood of University of California at Santa Cruz. The paper would not have been possible without the contributions of Drs. Florence Horn and Gert Vriend. Our project began when they provided us with data sets for the superfamily classification experiments, and they generously shared with us the methodology behind the excellent GPCRDB system. We would also like to thank Dr. Tommi Jaakkola, who developed the Fisher score vectors. Special thanks to Mark Diekhans for valuable discussions, as well as the feature extraction software used in our experiments.

References

- Altshul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410.
- Bouvier, M. (1997). Structural and functional aspects of g protein-coupled receptor oligomerization. CSBMCB Merck Frosst Award Address.
- Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press, New York, NY.
- Diekhans, M. (1999). Private communication.
- Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, New York, NY.
- Grundy, W. N. (1998). Family-based homology detection via pairwise sequence comparison. In *Int. Conf. Computational Molecular Biology (RECOMB-98)*, New York. ACM Press.
- Higgins, D. G. and Sharp, P. M. (1988). CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–44.

- Horn, F., Mokrane, M., Weare, J., and Vriend, G. (2000). *Genomics and Proteomics: Functional and Computational Aspects*, chapter G-protein coupled receptors or the power of data, pages 191–214. Kluwer Academic / Plenum Publishers, Norwell, MA.
- Horn, F., Weare, J., Beukers, M. W., Hörsch, S., Bairoch, A., Chen, W., Edvardsen, O., Campagne, F., and Vriend, G. (1998). Gpcrdb: an information system for g protein-coupled receptors. *Nucleic Acids Res*, 26(1):277–281.
- Hughey, R. and Krogh, A. (1995). 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.
- Jaakkola, T., Diekhans, M., and Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Jour. Comp. Biol.*, 7(1/2).
- Jaakkola, T. and Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, San Mateo, CA. Morgan Kaufmann Publishers.
- Karchin, R. (2000). Classifying g-protein coupled receptors with support vector machines. Master's thesis, University of California, Computer Science, UC Santa Cruz, CA 95064. <http://www.cse.ucsc.edu/research/compbio/research.html>.
- Karplus, K. (1996). Dirichlet mixture site. <http://www.cse.ucsc.edu/research/compbio/dirichlets/index.html>.
- Karplus, K., Barrett, C., and Hughey, R. (1998). Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856.
- Karplus, K., Karchin, R., Barrett, C., Tu, S., Cline, M., Diekhans, M., Grate, L., Casper, J., and Hughey, R. (2001). What is the value added by human intervention in protein structure prediction. *Prot: Struct, Funct, and Genetics Suppl*.
- Kent, J. (2001). Human Genome Browser WWW site. <http://genome.ucsc.edu/>.
- Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531.
- Mueller, T. (2000). Private communication.
- Palczewski, K., Kumasaka, T., Hori, T., Behnke, C., Motoshima, H., Fox, B., Trong, I. L., Teller, D., Okada, T., Stenkamp, R., Yamamoto, M., and Miyano, M. (2000). Crystal structure of rhodopsin: A g protein-coupled receptor. *Science*, 289:739–745.
- Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press.
- Platt, J. C., Cristianini, N., and Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553. MIT Press.
- Sjölander, K., Karplus, K., Brown, M. P., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. (1996). Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *CABIOS*, 12(4):327–345.
- Sonnhammer, E. L. L., Eddy, S. R., Birney, E., Bateman, A., and Durbin, R. (1998a). Pfam: multiple sequence alignments and hmm-profiles of protein domains. *NAR*, 26(1):320–322.
- Sonnhammer, E. L. L., von Heijne, G., and Krogh, A. (1998b). A hidden markov model for predicting transmembrane helices in protein sequences. In et. al, J. G., editor, *Proc. Sixth Int. Conf. on Intelligent Systems for Molecular Biology*, pages 175–182, Menlo Park. AAAI Press.
- Sullivan, S. (2000). Private communication.
- Vapnik, V. N. (1979). *Estimation of Dependencies Based on Empirical Data*. Nauka, Birmingham, AL.
- Vriend, G. (1999). Private communication.
- Watson, S. and Arkininstall, S. (1994). *The G-protein Linked Receptor FactsBook*. Academic Press, Harcourt Brace & Company, Burlington, MA.
- Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway University of London.
- WU-BLAST (2001). WU-BLAST WWW archives. <http://blast.wustl.edu/>.
- Yeang, C. H., Ramaswamy, S., Tamayo, P., Mukherjee, S., Rifkin, R. M., Angelo, M., Reich, M., Lander, E., Mesirov, J., and Golub, T. (2001). Molecular classification of multiple tumor types. *Bioinformatics Suppl*, 17(1):S316–S322.

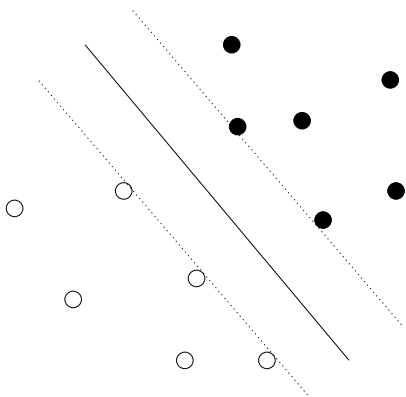


Figure 7: A hyperplane separating two classes of examples. The support vectors are the examples adjacent to the hyperplane.

Appendix A Two-Class Support Vector Machines

The SVM is a supervised learning algorithm that is trained to discriminate examples from different classes. We will consider first the two-class situation. During training, the machine is shown a set of labeled *training examples* (\vec{x}_i, y_i) where the \vec{x}_i are vector representations of the examples and the y_i are class labels (1 and -1). The closeness between two examples is computed by a *kernel function* $K(\vec{x}, \vec{y})$, a generalization of the inner product $\langle \vec{x}, \vec{y} \rangle$. The algorithm does several rounds of all-against-all kernel scoring, and computes a final weight α for each training example that measures its importance as a discriminator between the two classes.

There are many known algorithms for training a support vector machine. We used a steepest gradient ascent algorithm developed by Tommi Jaakkola (Jaakkola and Haussler, 1998). This is a constrained maximization procedure in which each of l training examples is assigned an initial α weight of 0.5 and then iteratively updated as

$$\alpha_{i_{new}} \leftarrow f \left(\frac{1 - y_i \left(\sum_{j=1}^l y_j \alpha_{j_{old}} K(\vec{x}_i, \vec{x}_j) \right) + \alpha_{i_{old}} K(\vec{x}_i, \vec{x}_i)}{K(\vec{x}_i, \vec{x}_i)} \right) \quad (1)$$

where

$$f(z) = \begin{cases} 0, & z < 0 \\ z, & 0 \leq z \leq 1 \\ 1, & z > 1 \end{cases}$$

In our implementation, we speed up convergence by dropping an example \vec{x}_i from the training set if α_i stays at zero for two consecutive iterations.

The constraints imposed by the clipping function $f(\cdot)$ relax the requirement that the two classes of examples be perfectly separated by a linear boundary during training, a technique known as using a *soft margin* (Cristianini and Shawe-Taylor, 2000).

Geometrically, an example vector can be plotted as a point in an n -dimensional space (where n is the number of components in the vector). The weight vector learned during training defines the *hyperplane* with the widest possible margin between the two classes of training examples, with largest weights assigned to those examples closest to the hyperplane.

Mathematically, the hyperplane is the zeros of a linear function called the *discriminant*:

$$g(\vec{x}, \vec{x}_i, \alpha_i, y_i) = \sum_{i=1}^l y_i \alpha_i K(\vec{x}_i, \vec{x}) \quad (2)$$

The discriminant can be used to predict the class of a new, previously unseen example by summing over the weighted kernel scores of the new example and the l labelled examples from the training set. A positive discriminant score predicts that the new example is in the positive class and vice versa. As the discriminant's distance from zero increases, so does the confidence of the prediction. It is possible to estimate the statistical significance of these scores, but an unbiased calibration requires more positive examples than we have in the GPCR subfamily dataset (Platt, 1999).

We can observe in Figure 7 that the hyperplane is completely defined by the training examples with the largest α values, which lie adjacent to it. These are known as *support vectors*. All other examples have α values of 0 when the algorithm converges. Conveniently, this simplifies computation of the discriminant when we want to classify a new example.

To derive vectors for our examples, we use a method developed by Tommi Jaakkola and David Haussler (Jaakkola and Haussler, 1998). The method is based on building a statistical model of a group of protein sequences, the *hidden Markov model* (HMM) (Durbin et al., 1998), then computing the gradient of the log likelihood that a protein sequence of interest was generated by the model. Because we consider the possibility that the encoding of a protein sequence into a feature vector is more important than the algorithm used to classify feature vectors, and because this transformation is critical for those who wish to reproduce our experiments, we will go into the *Fisher score vector* encoding in some detail.

To calculate the likelihood that a sequence X is a member of the group of sequences modeled by the hidden Markov model M , we use eq. (3) to compute the posterior probability of X given M .

$$P(X|M) = P(X|\theta, \tau) = \sum_{s_1, \dots, s_n} \prod_i P(x_i|s_i, \theta) P(s_i|s_{i-1}, \tau) = \sum_{s_1, \dots, s_n} \prod_i \theta_{x_i|s_i} \tau_{s_{i-1}|s_i} \quad (3)$$

Here, θ represents the probability distributions in the character emitting (match and insert (Krogh et al., 1994)) 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 emitting state s_i , given the probability distributions in M . $P(s_i|s_{i-1}, \tau)$ is the probability of landing in emitting state s_i after being in emitting state s_{i-1} , while in model M .

Because the sequence of n emitting states $\{s_1, \dots, s_n\}$ through the model is hidden, we sum over all possible such state sequences to compute the posterior probability. This quantity can be viewed as the probability that X was generated by M .

To avoid computational problems due to floating point underflow, we use a simple transformation from multiplication of probabilities $P(X|\theta, \tau)$ to summation of *log likelihoods* $\log P(X|\theta, \tau)$.

The difference between model M generating two sequences X and Y can be quantified in the *gradient space* of M . The gradient of the log likelihood of X with respect to the parameters of the model is the vector of partial derivatives:

$$\frac{\partial}{\partial \theta_{\tilde{x}, \tilde{s}}} \log P(X|\theta, \tau) \quad (4)$$

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 (Jaakkola and Haussler, 1998).

Each component of the Fisher score vector is computed by Fisher sufficient statistics:

$$f_s(x) = \frac{\partial}{\partial \theta_{\tilde{x}, \tilde{s}}} \log P(X|\theta, \tau) = \frac{\xi(\tilde{x}, \tilde{s})}{\theta_{\tilde{x}, \tilde{s}}} - \xi(\tilde{s}) \quad (5)$$

where $\xi(\tilde{x}, \tilde{s})$ is the expected number of times we visit state s and generate amino acid x , and $\xi(\tilde{s})$ is the expected number of times we visit state s , as we traverse all paths through the model. These quantities are easily computed using the standard forward-backward algorithm (Jaakkola and Haussler, 1998). By using eq. (5), 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 will be indexed by amino acid and state (x, s) (Jaakkola and Haussler, 1998). We construct Fisher score vectors by taking partial derivatives with respect to the match state emission parameters of the HMM, ignoring transition parameters and emissions from the insert states. This heuristic was used for the experiments reported in Jaakkola et. al. (Jaakkola et al., 2000) and resulted in fewer errors than taking partial derivatives with respect to all the HMM parameters (Diekhans, 1999). Consequently, each Fisher score vector has twenty components for each match state in the hidden Markov model. For example, a typical member of the histamine receptor subfamily is approximately 500 residues long. Our HMM for this subfamily contains 483 match states, so each Fisher score vector computed with respect to this HMM will contain 9660 floating point components (77.3 KB). The training set for the histamine receptor SVM has 2487 example vectors, requiring 192 MB of disk space. Two-hundred and fifty such training sets (one per GPCR subfamily) would require 48 GB.

To deal with this problem, we used a feature reduction technique (Jaakkola et al., 2000), in which enhanced Fisher score vectors were computed using a set of pre-calculated amino acid distributions known as *mixture priors* (Sjölander et al., 1996). A mixture decomposes the probability of amino acid x in state s into l components.

$$\theta_{x|s} = \sum_l c_{l|s} \theta_{x|s}^{(l)} \quad (6)$$

The quantities $\theta_{x|s}^{(l)}$ 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 that weight the subclasses

and sum to one. In these enhanced Fisher score vectors, the feature for component l in match state s is given by

$$f_{l,s}(X) = \frac{\partial}{\partial c_{l,s}} \log P(X|\theta, c, \tau) = \sum_x \xi(x, s) \left[\frac{\theta_{x|s}^{(l)}}{\theta_{x|s}} - 1 \right] \quad (7)$$

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. (7). We used a nine-component mixture, *uprior.9comp* (Karplus, 1996), reducing the number of components in each Fisher score vector by approximately one-half. Although the selection of mixture components is somewhat arbitrary and the mixture coefficients are never explicitly computed, this technique makes manipulation of Fisher score vectors significantly more tractable in terms of computer memory and disk space. We have observed no decrease in discrimination performance due to the feature reduction, a result also reported by Jaakkola, et. al. (Diekhans, 1999).

In all of our classification experiments involving feature vectors, we first transformed each protein sequence in our data sets into a $9n$ -component Fisher score vector, where n is the number of match states in the HMM. The closeness between a pair of Fisher score vectors was then measured with the following radial basis kernel function.

$$K(X, Y) = \exp \left(\frac{-\sum_{l,s} (f_{l,s}(X) - f_{l,s}(Y))^2}{2\sigma^2} \right) \quad (8)$$

To get good discrimination and generalization performance from the kernel function, the width parameter σ must match the scale of the examples being classified. We set σ to the median of Euclidean distances between the feature vectors of positive examples and their nearest negative neighbors in the training set, a heuristic approach that has produced good results (Jaakkola et al., 2000).

Appendix B Multi-Class Support Vector Machines

Three approaches have been proposed for k -class support vector machines: training k “one-against-the-rest” classifiers, training $O(k^2)$ “one-against-one” classifiers, in which a two-class SVM is trained on all possible pairs of classes, and training a single SVM, which constructs a discriminant that considers all k classes at once (Weston and Watkins, 1998).

Our implementation of one-to-rest multi-class SVMs involves training a two-class SVM for each of k GPCR subfamilies of interest. The predicted class $h(\vec{x})$ for each example \vec{x} in the test set is the class with the largest discriminant computed by any of the SVMs. For a labeled training set of l examples (\vec{x}_i, y_i) , which we will denote as X , we have

$$h(\vec{x}) = \arg \max_k g(\vec{x}, X^k, \alpha^k) \quad (9)$$

$$= \arg \max_k \sum_{i=1}^l y_i^k \alpha_i^k K_k(\vec{x}_i, \vec{x}) \quad (10)$$

There are many other ways to do multi-class support vector machines, but no consensus as to which one is best (Platt et al., 2000). Our choice was motivated by the idiosyncracies of our data set (few positive examples and many negative ones, large number of classes, high-dimensional feature vectors), the lower complexity of one-to-rest SVMs and reported lower error rates of one-to-rest SVMs (Yeang et al., 2001).