Combining pairwise sequence similarity and support vector machines for remote protein homology detection

Li Liao
Central Research & Development
E. I. du Pont de Nemours Company
li.liao@usa.dupont.com

William Stafford Noble*†
Department of Computer Science
Columbia Genome Center
Columbia University
noble@cs.columbia.edu

October 15, 2001

Abstract

One key element in understanding the molecular machinery of the cell is to understand the meaning, or function, of each protein encoded in the genome. A very successful means of inferring the function of a previously unannotated protein is via sequence similarity with one or more proteins whose functions are already known. Currently, one of the most powerful such homology detection methods is the SVM-Fisher method of Jaakkola, Diekhans and Haussler (ISMB 2000). This method combines a generative, profile hidden Markov model (HMM) with a discriminative classification algorithm known as a support vector machine (SVM). The current work presents an alternative method for SVM-based protein classification. The method, SVM-pairwise, uses a pairwise sequence similarity algorithm such as Smith-Waterman in place of the HMM in the SVM-Fisher method. The resulting algorithm, when tested on its ability to recognize previously unseen families from the SCOP database, yields significantly better remote protein homology detection than SVM-Fisher, profile HMMs and PSI-BLAST.

*Corresponding author: 450 Computer Science Building, 1214 Amsterdam Avenue, New York, NY 10027
†Formerly William Noble Grundy: see www.cs.columbia.edu/˜noble/name-change.html

1 Introduction

Protein homology detection is a core problem in computational biology. Detecting subtle sequence similarities among proteins is useful because sequence similarity typically implies homology, which in turn may imply functional similarity. The discovery of a statistically significant similarity between two proteins is frequently used, therefore, to justify inferring a common functional role for the two proteins.

Over the past 25 years, researchers have developed a battery of successively more powerful methods for detecting protein sequence similarities. This development can be broken into four stages. Early methods looked for pairwise similarities between proteins. Among such algorithms, the Smith-Waterman dynamic programming algorithm [24] is among the most accurate, whereas heuristic algorithms such as BLAST [1] and FASTA [22] trade reduced accuracy for improved efficiency. In the second stage, further accuracy was achieved by collecting aggregate statistics from a set of similar sequences and comparing the resulting statistics to a single, unlabeled protein of interest. Profiles [10] and hidden Markov models (HMMs) [18, 4] are two methods for representing these aggregate statistics. These family-based methods allow the comptutational biologist to infer nearly three times as many homologies as a simple pairwise alignment algorithm [21]. In stage three, additional accuracy was gleaned by leveraging the information in large databases of unlabeled protein sequences. Iterative methods such as PSI-BLAST [2] and SAM-T98 [17] improve upon profile-based methods by iteratively collecting homologous sequences from a large database and incorporating the resulting statistics into a central model. All of the resulting statistics, however, are generated from positive examples, i.e., from sequences that are known or posited to be evolutionarily related to one another. In stage four, additional accuracy was gained by modeling the difference between positive and negative examples. Because the homology task requires discriminating between related and unrelated sequences, explicitly modeling the difference between these two sets of sequences yields an extremely powerful method. The SVM-Fisher method [15, 16], which couples an iterative HMM training scheme with a discriminative algorithm known as a support vector machine (SVM) [26, 8], is currently the most accurate known method for detecting remote protein homologies.

This paper presents an SVM-based protein classification method that uses a pairwise sequence similarity algorithm in place of the HMM of the SVM-Fisher method. Both the SVM-Fisher method and the new method, called SVM-pairwise, consist of two steps: converting a given set of proteins into fixed-length vectors, and training an SVM from the vectorized proteins. The two methods differ only in the vectorization step. In the SVM-Fisher method, a protein's vector representation is its gradient with respect to a profile hidden Markov model; in the SVM-pairwise method, the vector is a list of pairwise sequence similarity scores.

The pairwise score representation of a protein offers three primary advantages over the profile HMM gradient representation. First, the pairwise score representation is simpler, since it dispenses with the profile HMM topology and parameterization, including training via expectation-maximization. Second, pairwise scoring does not require a multiple alignment of the training set sequences. For distantly related protein sequences, a profile alignment may not be possible, if for example the sequences contain shuffled domains. Thus, a collection of pairwise alignments allows for the detection of motif- or domain-sized similarities, even when the entire model cannot be easily aligned.

The third and most significant advantage of the pairwise score representation is its use of a negative training set. A profile HMM is trained solely on a collection of positive examples — sequences that are known (or at least believed) to be homologous to one another. The SVM adds to this model the ability to learn from negative examples as well, by discriminating between the two classes. In the SVM-pairwise method, this discriminative advantage is extended throughout the algorithm. The vector space defined by the pairwise scores includes many dimensions (i.e., sequence similarity scores) that are unrelated to the positive training set. These dimensions, if they contain significant similarity scores, can provide important evidence against a protein belonging to the positive class. For example, if a query protein is somewhat similar to sequences in the positive class but very similar to several proteins in the negative class, then the slight similarities to the positive class can safely be ignored. In the absence of these negative examples, the classification of such a sequence would remain in doubt.

The following section describes in more detail the two protein vectorization methods. This section is followed by an experimental comparison of six protein homology detection methods. The methods include the SVM-Fisher [15] and SVM-pairwise methods, two BLAST-based algorithms (PSI-BLAST [2] and Family Pairwise Search [FPS] [12]), a profile HMM method (SAM [18]), and a method (called KNN-pairwise) which is similar to SVM-pairwise but which replaces the SVM classifier with a k-nearest neighbor classifier. We measure the ability of each algorithm to discover previously unseen families from the SCOP database [20], using as training sets all other members of the family's superfamily. The experiments induce a complete ranking of methods, in the following order of performance (least sensitive to most sensitive): FPS, SAM, PSI-BLAST, KNN-pairwise, SVM-Fisher, SVM-pairwise. Thus, for this set of data, the algorithm described here produces the most accurate means of detecting remote homologs among these six methods.

2 Algorithm

The SVM algorithm, which provides the framework of the SVM-Fisher and SVM-pairwise methods, is suprisingly simple. The algorithm addresses the general problem of learning to discriminate between positive and negative members of a given class of n-dimensional vectors. The algorithm operates by mapping the given training set into a possibly high-dimensional feature space and attempting to locate in that space a plane that separates the positive from the negative examples. Having found such a plane, the SVM can then predict the classification of an unlabeled example by mapping it into the feature space and asking on which side of the separating plane the example lies. Much of the SVM's power comes from its criterion for selecting a separating plane when many candidates planes exist: the SVM chooses the plane that maintains a maximum margin from any point in the training set. Statistical learning theory suggests that, for some classes of well-behaved data, the choice of the maximum margin hyperplane will lead to maximal generalization when predicting the classification of previously unseen examples [26]. The SVM algorithm can also be extended to cope with noise in the training set and with multiple classes [8].

One important requirement of the SVM is that the input be a collection of fixed-length vectors. Proteins, of course, are variable-length sequences of amino acids and hence cannot be directly input to the SVM. In the SVM-Fisher method, the HMM provides the necessary means of converting proteins into fixed-length vectors. First, the HMM is trained using the positive members of the training set. Then the gradient vector of any sequence — positive, negative or unlabeled — can be computed with respect to the trained model. Each component of the gradient vector corresponds to one parameter of the HMM. The vector summarizes how different the given sequence is from a typical member of the given protein family. An SVM trained on a collection of positively and negatively labeled protein gradient vectors learns to classify proteins extremely well.

In the current work, we would like to accomplish a similar conversion of a protein from an amino acid sequence into a fixed-length numeric vector. A straightforward method is suggested by the Family Pairwise Search (FPS) algorithm [12, 3]. FPS extends a pairwise sequence comparison algorithm such as Smith-Waterman or BLAST to carry out sequence-versus-family comparisons

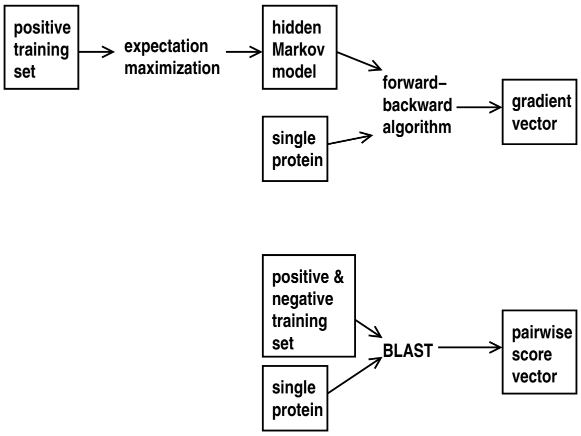


Figure 1: **Schematic diagram of the protein vectorization step in the SVM-Fisher (top) and SVM-pairwise (bottom) algorithms.**

by combining multiple pairwise comparison scores. BLAST-based FPS is efficient and has been shown to perform competitively with HMM methods [12]. In place of an explicit model of the protein family, FPS uses the members of the family. This implicit model provides an easy way to vectorize a given protein: simply store in the vector the pairwise similarity scores with respect to each member of the training set. As in the SVM-Fisher method, the vectorized proteins can then be fed into an SVM. We call this algorithm SVM-pairwise. The difference between the two algorithms is illustrated in Figure 1.

3 Methods

The experiments reported here compare the performance of six algorithms: SVM-pairwise, SVM-Fisher, PSI-BLAST, SAM, FPS, and a simplified version of SVM-pairwise called KNN-pairwise (see Table 2). We assess the recognition performance of each algorithm by testing its ability to classify protein domains into superfamilies in the Structural Classification of Proteins (SCOP) [20] version 1.53. Sequences were selected using the Astral database (astral.stanford.edu [6]), removing similar sequences using an E-value threshold of 10^{-25}. This procedure resulted in 4352 distinct sequences, grouped into families and superfamilies. For each family, the protein domains within the family are considered positive test examples, and the protein domains outside the family but within the same superfamily are taken as positive training examples. The data set yields 54 families containing at least 10 family members (positive test) and 5 superfamily members outside of the family (positive train). Negative examples are taken from outside of the positive sequences' fold, and are randomly split into train and test sets in the same ratio as the positive examples. Details about the various families are listed in Table 1, and the complete data set is available at www.cs.columbia.edu/compbio/svm-pairwise. This experimental setup is similar to that used by Jaakkola *et al.* [15], except for one important difference: in the current experiments, the positive training sets do not include additional protein sequences extracted from a large, unlabeled database. As such, the recognition tasks performed here are more difficult than those in Jaakkola *et al.* In principle, any of the six methods described here could be applied in an iterative framework using an auxiliary database.

The vectorization step of SVM-pairwise uses the Smith-Waterman algorithm as implemented

	Positive set		Negative set			Positive set		Negative set	
ID	Train	Test	Train	Test	ID	Train	Test	Train	Test
1.27.1.1	12	6	2890	1444	2.9.1.4	21	10	2928	1393
1.27.1.2	10	8	2408	1926	3.1.8.1	19	8	3002	1263
1.36.1.2	29	7	3477	839	3.1.8.3	17	10	2686	1579
1.36.1.5	10	26	1199	3117	3.2.1.2	37	16	3002	1297
1.4.1.1	26	23	2256	1994	3.2.1.3	44	9	3569	730
1.4.1.2	41	8	3557	693	3.2.1.4	46	7	3732	567
1.4.1.3	40	9	3470	780	3.2.1.5	46	7	3732	567
1.41.1.2	36	6	3692	615	3.2.1.6	48	5	3894	405
1.41.1.5	17	25	1744	2563	3.2.1.7	48	5	3894	405
1.45.1.2	33	6	3650	663	3.3.1.2	22	7	3280	1043
2.1.1.1	90	31	3102	1068	3.3.1.5	13	16	1938	2385
2.1.1.2	99	22	3412	758	3.32.1.1	42	9	3542	759
2.1.1.3	113	8	3895	275	3.32.1.11	46	5	3880	421
2.1.1.4	88	33	3033	1137	3.32.1.13	43	8	3627	674
2.1.1.5	94	27	3240	930	3.32.1.8	40	11	3374	927
2.28.1.1	18	44	1246	3044	3.42.1.1	29	10	3208	1105
2.28.1.3	56	6	3875	415	3.42.1.5	26	13	2876	1437
2.38.4.1	30	5	3682	613	3.42.1.8	34	5	3761	552
2.38.4.3	24	11	2946	1349	7.3.10.1	11	95	423	3653
2.38.4.5	26	9	3191	1104	7.3.5.2	12	9	2330	1746
2.44.1.2	11	140	307	3894	7.3.6.1	33	9	3203	873
2.5.1.1	13	11	2345	1983	7.3.6.2	16	26	1553	2523
2.5.1.3	14	10	2525	1803	7.3.6.4	37	5	3591	485
2.52.1.2	12	5	3060	1275	7.39.1.2	20	7	3204	1121
2.56.1.2	11	8	2509	1824	7.39.1.3	13	14	2083	2242
2.9.1.2	17	14	2370	1951	7.41.5.1	10	9	2241	2016
2.9.1.3	26	5	3625	696	7.41.5.2	10	9	2241	2016

Table 1: **SCOP families included in the experiments.** For each family, the numbers of sequences in the positive and negative training and test sets are listed. A version of this table that includes the names of each family (in addition to their SCOP IDs) is available at www.cs.columbia.edu/compbio/svm-pairwise.

Method	Train from
SVM-pairwise	Positives and negatives
SVM-Fisher	Positives and negatives
PSI-BLAST	Positives only
SAM	Positives only
FPS	Positives only
KNN-pairwise	Positives and negatives

Table 2: **Six protein homology detection methods**

on the BioXLP hardware accelerator (www.cgen.com). The feature vector corresponding to protein X is $F_X = f_{x1}, f_{x2}, \ldots, f_{xn}$, where n is the total number of proteins in the training set, and f_{xi} is the logarithm of the p-value of the Smith-Waterman score between sequence X and the ith training set sequence. The default gap opening penalty and extension penalties of 10 and 0.05, respectively, are used.

The SVM implementation employs the optimization algorithm described in [16], and the software is available at www.cs.columbia.edu/compbio/svm. At the heart of the SVM is a kernel function that acts as a similarity score between pairs of input vectors. The base SVM kernel is normalized so that each vector has length 1 in the feature space; i.e.,

$$K(X, Y) = \frac{X \cdot Y}{\sqrt{(X \cdot X)(Y \cdot Y)}}. \tag{1}$$

This kernel $K(\cdot, \cdot)$ is then transformed into a radial basis kernel $\hat{K}(\cdot, \cdot)$, as follows:

$$\hat{K}(X, Y) = e^{-\frac{K(X,X) - 2K(X,Y) + K(Y,Y)}{2\sigma^2}} + 1, \tag{2}$$

where the width σ is the median Euclidean distance (in feature space) from any positive training example to the nearest negative example. The constant 1 is added to the kernel in order to translate the data away from the origin. This translation is necessary because the SVM optimization algorithm we employ requires that the separating hyperplane pass through the origin. An asymmetric soft margin is implemented by adding to the diagonal of the kernel matrix a value $0.02 * \rho$, where ρ is the fraction of training set sequences that have the same label as the current sequence (see [7] for details). The output of the SVM is a discriminant score that is used to rank the members of the test set. The same SVM parameters are used for the SVM-Fisher and SVM-pairwise tests.

Hidden Markov models are trained using the Sequence Alignment and Modeling (SAM) toolkit (www.soe.ucsc.edu/research/compbio/sam.html) [18]. Models are built from unaligned positive training set sequences using the local scoring option ("`-SW 2`"). Following [16], we use a 9-component Dirichlet mixture prior developed by Kevin Karplus (`byst-4.5-0-3.9comp` at www.soe.ucsc.edu/research/compbio/dirichlets). Once a model is obtained, it is straightforward to compare the test sequences to the model by using `hmmscore` (also with the local scoring option). The resulting E-values are used to rank the test set sequences.

The SVM-Fisher method uses the same, trained HMMs during the vectorization step. As in the Baum-Welch training algorithm for HMMs, the forward and backward matrices are combined to yield a count of observations for each parameter in the HMM. As shown in [16], the counts can be converted into components of a gradient vector $\vec{\mathbf{U}}$ via the following equation:

$$\vec{\mathbf{U}}_{ij} = \frac{E_j(i)}{e_j(i)} - \sum_k E_j(k), \tag{3}$$

where $E_j(i)$ is the number of times that amino acid i is observed in state j, and $e_j(i)$ is the emission probability for amino acid i in state j. Although these gradients can be computed for every HMM parameter, the SVM-Fisher method uses only the gradient components that correspond to emission probabilities in the match states. Furthermore, a more compact gradient vector can be derived using a mixture decomposition of the emission probabilities. The mixture gradient calculation, analogous to Equation 3, is as follows:

$$\vec{\mathbf{U}}_{\ell j} = \sum_{i=1}^{20} E_j(i) \left[\frac{\theta_{i\ell}}{e_j(i)} - 1 \right], \tag{4}$$

where $\theta_{i\ell}$ corresponds to the ith amino acid in the ℓth Dirichlet distribution. These experiments employ the same 9-component Dirichlet mixture mentioned above. For a profile HMM containing m match states, the resulting vector contains $9m$ components. These vectors are then used as input to an SVM, as described above.

For comparison, we also include in the experiments the PSI-BLAST algorithm [2], which is probably the most widely-used protein homology detection algorithm. It is not straightforward to compare PSI-BLAST, which requires as input a single sequence, with methods such as HMMER and SVM-Fisher, which take multiple input sequences. We address this problem by randomly selecting a positive training set sequence to serve as the initial query. PSI-BLAST is run for one iteration on a database consisting only of the remaining positive training set sequences. An extremely high E-value threshold is applied so that all of the training set sequences are included in the resulting profile. This profile is then used for one additional iteration, this time using the test set as a database. The resulting E-values are used to rank the test set sequences. Note that PSI-BLAST is not run on the test set for multiple iterations: this restriction allows a fair comparison with the other, non-iterative methods included in the study.

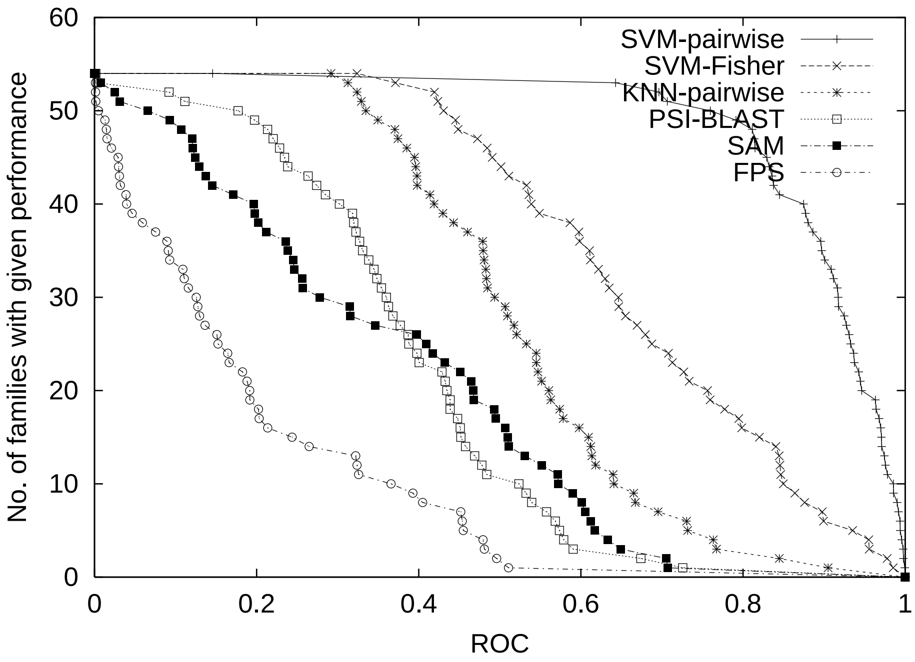
Family Pairwise Seach [12, 3] is another family-based protein homology detection method that is based upon the BLAST algorithm. We include in the study a simple form of FPS, called FPS-minp. This method simply ranks each test set sequence according to the minimum of the BLAST p-values with respect to the positive training set.

Finally, in order to evaluate the utility of the SVM in the SVM-pairwise algorithm, we include a method, KNN-pairwise, that replaces the SVM with a simpler discriminative classifier, the k-nearest neighbor algorithm. The algorithm takes as input the same feature vector as the SVM does in SVM-pairwise. However, rather than classifying a query protein by orienting it with respect to a separating plane, KNN locates the k training set proteins that are nearest to the query protein (using Euclidean distances between vectors). We use a kernel version of k-nearest neighbor, with the same kernel function as in the SVM. The predicted classification is simply the majority classification among these k neighbors. For this study, we use $k = 3$. Sequences are ranked according to the number of distance-weighted votes for the positive class.

Each of the above six methods produces as output a ranking of the test set sequences. To measure the quality of this ranking, we use two different scores: receiver operating characteristic (ROC) scores and the median rate of false positives (RFP). The ROC score is the normalized area under a curve that plots true positives as a function of false positives for varying classification thresholds [11]. A perfect classifier that puts all the positives at the top of the ranked list will receive an ROC score of 1, and for these data, a random classifier will receive an ROC score very close to 0. The median RFP score is the fraction of negative test sequences that score as high or better than the median-scoring positive sequence. RFP scores were used by Jaakkola *et al.* in evaluating the Fisher-SVM method.

4 Results

The results of the experiments are summarized in Figure 2. The two graphs rank the six homology detection methods according to ROC and median RFP scores. In each graph, a higher curve corresponds to more accurate homology detection performance. Using either performance measure, the SVM-pairwise method performs significantly better than the other six methods. We assess the statistical significance of differences among methods using a two-tailed signed rank test [14, 23]. The resulting p-values are conservatively adjusted using a Bonferroni correction for multiple comparisons. As shown in Table 3, nearly all of the differences apparent in Figures 2 are statistically



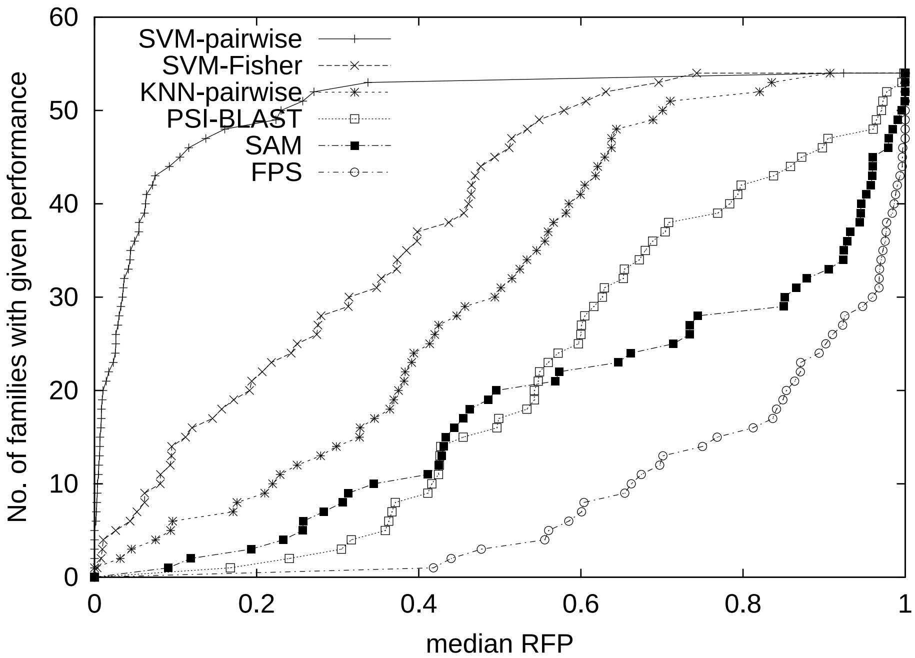


Figure 2: **Relative performance of the six homology detection methods** Each graph plots the total number of families for which a given method exceeds a score threshold. The top graph uses ROC scores, and the bottom graph uses median RFP scores. Each series corresponds to one of the protein homology detection methods shown in Table 2.

	SVM-Fisher	KNN-pairwise	PSI-BLAST	SAM	FPS
SVM-pairwise	5.0e-08	7.2e-09	3.3e-09	6.5e-09	3.3e-09
SVM-Fisher		2.0e-05	2.4e-08	1.0e-05	3.3e-09
KNN-pairwise			6.4e-08	1.6e-03	6.5e-09
PSI-BLAST				—	8.2e-07
SAM					6.2e-06

Table 3: **Statistical significance of differences between pairs of homology detection methods.** Each entry in the table is the p-value given by a two-tailed signed rank test comparing paired ROC scores from two methods for each of the 54 families. The p-values have been (conservatively) adjusted for multiple comparisons using a Bonferonni adjustment. An entry in the table indicates that the method listed in the current row performs significantly better than the method listed in the current column. A "—" indicates that the p-value is greater than 0.05. The statistics for median RFP scores are similar.

significant at a threshold of 0.05. The resulting induced performance ranking of methods is SVM-pairwise, SVM-Fisher, KNN-pairwise, PSI-BLAST, SAM, FPS. Only the difference between PSI-BLAST and SAM is not statistically significant.

Many of these results agree with previous assessments. For example, the relative performance of SVM-Fisher and SAM agrees with the results given in [15], as does the relatively poor performance of the FPS algorithm on this task. This latter result is probably due to the difficulty of the recognition task. A previous assessment [12], which found FPS to be competitive with profile HMMs, tested both algorithms on much less remote homologies. The FPS algorithm can be improved by using Smith-Waterman p-values, rather than BLAST, and by computing p-values for sequence-to-family comparisons [3]. However, we do not expect these improvements to make the algorithm competitive with the best algorithms in this experiment.

One surprise in Figure 2 is the relative ranking of SAM and PSI-BLAST: in previous work, SAM significantly out-performs PSI-BLAST [21]. This difference may have several explanations. First, we may have improperly used the SAM software, setting parameters differently than an expert would. In order to reduce this possibility, we repeated the experiment above using CLUSTALW [25] to align the sequences and HMMER [9] to build models and score them. The resulting ROC and median RFP scores are very similar to the scores produced by SAM (data not shown): the two sets of scores are not statistically significantly different from one another nor from PSI-BLAST scores. Second, the benefit of using SAM may be more improved in the context of an iterated search, as was used in [21]. A third explanation for the improvement in PSI-BLAST's performance is just that: the PSI-BLAST algorithm has been improved considerably in the last several years, and it may now perform as well as SAM, at least in this experimental paradigm.

The placement of the KNN-pairwise algorithm above PSI-BLAST and below SVM-Fisher is significant in several respects. On the one hand, this result shows that the pairwise similarity score representation brings considerable power to the method, resulting in a state-of-the-art classification method using only a very simple classification algorithm. On the other hand, the result also shows the utility of the SVM algorithm, since both SVM-based methods perform better than the KNN-based method. It would certainly be possible to improve our k-nearest neighbor implementation, using for example a generalization such as Parzen windows [5]. We have no reason to suspect, however, that such an improvement would yield better performance than the SVM-pairwise method.

The most significant result from our experiments is the top-ranking performance of the SVM-

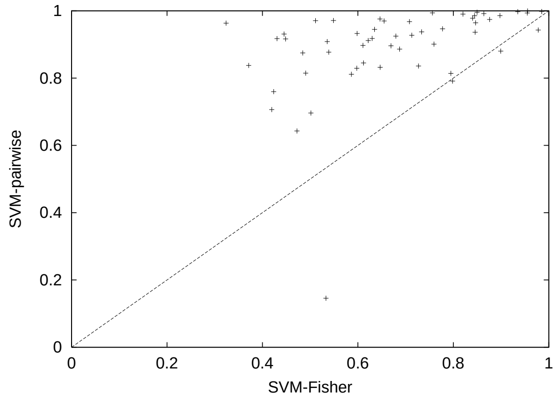


Figure 3: **Family-by-family comparison of Fisher-SVM and SVM-pairwise** Each point on the graph corresponds to one of the SCOP superfamilies listed in Table 1. The axes are ROC scores achieved by the two primary methods compared in this study: SVM-Fisher and SVM-pairwise.

pairwise method. This result is further illustrated in Figure 3, which shows a family-by-family comparison of the 54 ROC scores computed for each method. The SVM-pairwise method scores higher than the SVM-Fisher method on nearly every family. The one outlier is family 2.44.1.2, which has a relatively small training set. Family-by-family results from each of the six methods are available at www.cs.columbia.edu/compbio/svm-pairwise.

5 Discussion

We have shown that the SVM-pairwise method yields significantly improved remote homology detection relative to a number of existing, state-of-the-art algorithms. Like the SVM-Fisher algorithm, SVM-pairwise exploits a negative training set to yield more accurate predictions. Unlike SVM-Fisher, SVM-pairwise extends this discriminative component into the vectorization step. We hypothesize that the inclusion of negative examples in the vectorization step, along with the relaxation of a requirement for a multiple alignment of the training set sequences, explains the excellent performance of this algorithm.

One significant characteristic of any homology detection algorithm is its computational efficiency. In this respect, the SVM-pairwise algorithm is not significantly better than SVM-Fisher. Both algorithms include an SVM optimization, which is roughly $O(n^2)$, where n is the number of training set examples. The vectorization step of SVM-Fisher requires training a profile HMM and computing the gradient vectors. The gradient computation dominates, with a running time of $O(nmp)$, where m is the length of the longest training set sequence, and p is the number of HMM parameters. In contrast, the vectorization step of SVM-pairwise involves computing n^2 pairwise scores. Using Smith-Waterman, each computation is $O(m^2)$, yielding a total running time of $O(n^2m^2)$. Thus, assuming that $m \approx p$, the SVM-pairwise vectorization takes approximately n times as long as the SVM-Fisher vectorization.

There are several ways to speed up the SVM-pairwise vectorization. Most obviously, it should be possible to carry out the vectorization using a linear time approximation of Smith-Waterman, such as BLAST. This modification would immediately remove a factor of m from the running time, although the change would presumably decrease the accuracy of the algorithm. A second approach would be to use an explicit "vectorization set" of proteins for creating the feature vectors. In the current implementation, SVM-pairwise compares each training and test set sequence to every sequence in the training set. There is no reason, however, that the columns of the vector matrix must correspond to the training set sequences. A relatively small collection of widely distributed sequences (or even a library of profile HMMs) might provide a powerful, concise vector signature of any given protein.

A different approach for combining pairwise similarity scores with an SVM is to build the similarity score directly into the SVM. Several authors have derived kernel functions that allow direct comparison of strings [27, 13, 19]. These methods are appealing for protein homology detection because they obviate the need for an explicit vectorization step. A direct comparison of these methods with SVM-pairwise will be the subject of future research.

Acknowledgments: We thank Mark Diekhans for providing access to detailed results from their prior work as well as software for computing Fisher gradient vectors. We also thank Timothy Bailey for helpful discussion, and Darrin Lewis for his implementation of the k-nearest neighbor algorithm. This work was supported by an Award in Bioinformatics from the PhRMA Foundation, and by National Science Foundation grants DBI-0078523 and ISI-0093302.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[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] T. L. Bailey and W. N. Grundy. Classifying proteins by family using the product of correlated p-values. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology*, pages 10–14. ACM, April 1999.

[4] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the United States of America*, 91(3):1059–1063, 1994.

[5] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford UP, Oxford, UK, 1995.

[6] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

[7] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, Jr. M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*, 97(1):262–267, 2000.

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

[9] S. R. Eddy. Multiple alignment using hidden Markov models. In C. Rawlings, editor, *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 114–120. AAAI Press, 1995.

[10] M. Gribskov, R. Lüthy, and D. Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1990.

[11] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

[12] W. N. Grundy. Family-based homology detection via pairwise sequence comparison. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 94–100. ACM, 1998.

[13] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, Santa Cruz, CA, July 1999.

[14] S. Henikoff and J. G. Henikoff. Embedding strategies for effective use of information from multiple sequence alignments. *Protein Science*, 6(3):698–705, 1997.

[15] 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*, pages 149–158, Menlo Park, CA, 1999. AAAI Press.

[16] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2):95–114, 2000.

[17] K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–56, 1998.

[18] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[19] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, 2002. To appear.

[20] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[21] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210, 1998.

[22] W. R. Pearson. Rapid and sensitive sequence comparisions with FASTP and FASTA. *Methods in Enzymology*, 183:63–98, 1985.

[23] S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:371–328, 1997.

[24] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

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

[26] V. N. Vapnik. *Statistical Learning Theory.* Adaptive and learning systems for signal processing, communications, and control. Wiley, New York, 1998.

[27] C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. Bartlett, B. Schölkopf, and C. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT Press, 1999.