JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 10, Number 6, 2003

Pp. 857–868

Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships

LI LIAO[1] and WILLIAM STAFFORD NOBLE[2,3]

ABSTRACT

One key element in understanding the molecular machinery of the cell is to understand the structure and function of each protein encoded in the genome. A very successful means of inferring the structure or function of a previously unannotated protein is via sequence similarity with one or more proteins whose structure or function is already known. Toward this end, we propose a means of representing proteins using pairwise sequence similarity scores. This representation, combined with a discriminative classification algorithm known as the support vector machine (SVM), provides a powerful means of detecting subtle structural and evolutionary relationships among proteins. The algorithm, called SVM-pairwise, when tested on its ability to recognize previously unseen families from the SCOP database, yields significantly better performance than SVM-Fisher, profile HMMs, and PSI-BLAST.

Key words: pairwise sequence comparison, homology, detection, support vector machines.

1. INTRODUCTION

DETECTING SUBTLE PROTEIN SEQUENCE SIMILARITIES is a core problem in computational biology. Sequence similarity typically implies homology, which in turn may imply structural and 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 (Smith and Waterman, 1981) is among the most accurate, whereas heuristic algorithms such as BLAST (Altschul *et al.*, 1990) and FASTA (Pearson, 1985) 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 (Gribskov

[1]Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716.
[2]Department of Genome Sciences, University of Washington, Seattle, WA 98195.
[3]Formerly William Noble Grundy: see www.gs.washington.edu/~noble/name-change.html.

et al., 1990) and hidden Markov models (HMMs) (Krogh *et al.*, 1994; Baldi *et al.*, 1994) 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 (Park *et al.*, 1998).

In stage three, additional accuracy was gleaned by leveraging the information in large databases of unlabeled protein sequences. Iterative methods such as PSI-BLAST (Altschul *et al.*, 1997) and SAM-T98 (Karplus *et al.*, 1998) improved 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 (Jaakkola *et al.*, 1999, 2000), which couples an iterative HMM training scheme with a discriminative algorithm known as a support vector machine (SVM) (Vapnik, 1998; Christianini and Shawe-Taylor, 2000), is one of the most accurate known method for detecting remote protein homologies.

This paper proposes a simple way to represent a protein sequence as a fixed-length vector of real numbers. The resulting vectors can then be used as input to a discriminative learning algorithm. The vector representation is summarized in Fig. 1. The essential idea is that the interesting characteristics of a protein sequence can be effectively captured by asking how similar the protein is to a large collection of other proteins. A given protein is compared to every protein in the collection, which is called the vectorization set of proteins. Thus, for example, if the vectorization set consisted of one protein domain from each of the domain families in the Pfam (Sonnhammer *et al.*, 1997) database, then the resulting vector representation of a protein would indicate which of the Pfam domains appear in the given sequence.

This idea—of representing an object via its similarity to a collection of other objects—is not novel. In support vector machine learning, this type of mapping is known as an empirical feature map (Tsuda, 1999). Such a mapping also underlies the spectral clustering algorithm (Ng *et al.*, 2002), which uses an all-versus-all computation of radial basis function similarity scores as its initial step.

This paper presents an SVM-based protein classification method that uses the 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, computed with respect to all of the sequences in the training set.

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 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)

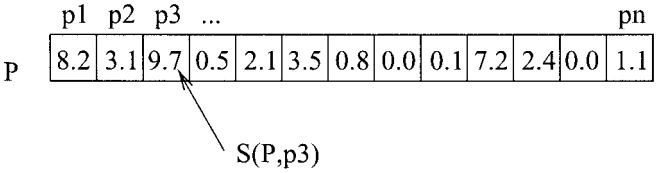

FIG. 1. Representing a protein as a vector of pairwise similarity scores. The protein P is represented as a vector of scores. The score function $S(\cdot, \cdot)$ is computed using a standard sequence comparison algorithm. The vectorization set of proteins $p_1, p_2, p_3, \ldots, p_n$ is selected a priori and is used for every protein being vectorized.

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 protein homology detection methods. The methods include the SVM-Fisher (Jaakkola *et al.*, 1999) and SVM-pairwise methods, two BLAST-based algorithms (PSI-BLAST [Altschul *et al.*, 1997] and Family Pairwise Search [FPS] [Grundy, 1998]), a profile HMM method (SAM [Krogh *et al.*, 1994]), and several variants of the SVM-pairwise algorithm. We measure the ability of each algorithm to discover previously unseen families from the SCOP database (Murzin *et al.*, 1995), using as training sets all other members of the family's superfamily. The experiments indicate that, for this set of data, the algorithm described here produces the most accurate means of detecting remote homologs among these methods.

A preliminary version of this work was described by Liao and Noble (2002). Relative to that paper, the current work includes some new experimental results (Section 4.2) and fixes some errors in the previously reported results. Supplementary data, including the training sets and pairwise similarity scores, as well as experimental results, can be found at *www.cs.columbia.edu/compbio/svm-pairwise*.

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 (Vapnik, 1998). The SVM algorithm can also be extended to cope with noise in the training set and with multiple classes (Christianini and Shawe-Taylor, 2000).

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 standard 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. The sequence similarity score vectorization described in Section 1 is closely related to the Family Pairwise Search (FPS) algorithm (Grundy, 1998; Bailey and Grundy, 1999). FPS extends a pairwise sequence comparison algorithm such as Smith–Waterman or BLAST to carry out sequence-versus-family comparisons by combining multiple pairwise comparison scores. BLAST-based FPS is efficient and has been shown to perform competitively with HMM methods

(Grundy, 1998). 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, and it is illustrated in Fig. 2.

3. METHODS

The primary experiment reported here compares the performance of five algorithms: SVM-pairwise, SVM-Fisher, PSI-BLAST, SAM, and FPS. 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) (Murzin *et al.*, 1995) version 1.53. Sequences were selected using the Astral database (*astral.stanford.edu* [Brenner *et al.*, 2000]), removing similar sequences using an E-value threshold of 10^{-25}. This procedure resulted in 4,352 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.* (1999), 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 seven 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 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 E-value of the Smith–Waterman score between sequence X and the ith training set sequence. The default parameters are used: gap opening penalty and extension penalties of 11 and 1, respectively, and the BLOSUM 62 matrix.

The SVM implementation employs the optimization algorithm described by Jaakkola *et al.* (2000), and a web server and software is available at *svm.sdsc.edu*. 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.,

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

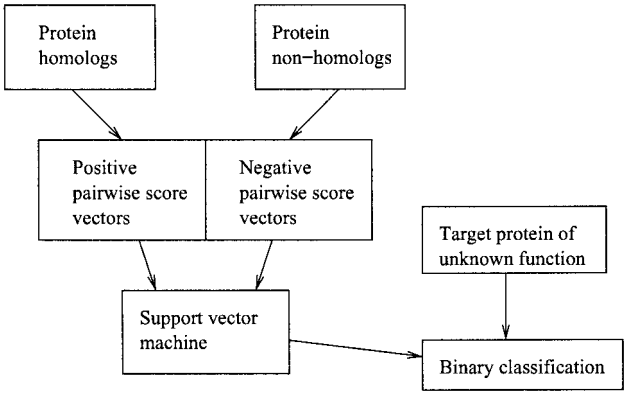

FIG. 2. Schematic diagram of the SVM-pairwise algorithm. Pairwise sequence similarity scores are computed using a standard algorithm such as BLAST or Smith–Waterman. Vectors are computed by comparing each protein to every other protein in the (positive and negative) training set.

TABLE 1. SCOP FAMILIES INCLUDED IN THE EXPERIMENTS[a]

	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

[a]For each family, the numbers of sequences in the positive and negative training and test sets are listed. Full names of each SCOP family are available at *www.cs.columbia.edu/compbio/svm-pairwise*.

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 Brown *et al.* [2000] 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*) (Krogh *et al.*, 1994). Models are built from unaligned positive training set sequences using the local scoring option (`-SW 2`). Following Jaakkola *et al.* (2000), 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 by Jaakkola *et al.* (2000), 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 nine-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 (Altschul *et al.*, 1997), 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. The complete positive training set is then aligned using CLUSTALW (Thompson *et al.*, 1994). Using the query sequence and the alignment as inputs, PSI-BLAST is run for one iteration with 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, noniterative methods included in the study.

Family Pairwise Search (Grundy, 1998; Bailey and Grundy, 1999) is another family-based protein homology detection method that is based upon pairwise sequence similarity scores. 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 Smith–Waterman E-values with respect to the positive training set.

Each of 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 (Gribskov and Robinson, 1996). A perfect classifier that puts all the positives at the top of the ranked list will receive an ROC score of 1, and a random classifier will receive an ROC score of 0.5. 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

4.1. Primary experiment

The results of the primary experiment are summarized in Fig. 3. The two graphs rank the five 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 four methods. We assess the statistical significance of differences among methods using a two-tailed signed rank test (Henikoff and Henikoff, 1997; Salzberg, 1997). The resulting p-values are conservatively adjusted using a Bonferroni correction for multiple comparisons. As shown in Table 2, nearly all of the differences apparent in Fig. 3 are statistically significant

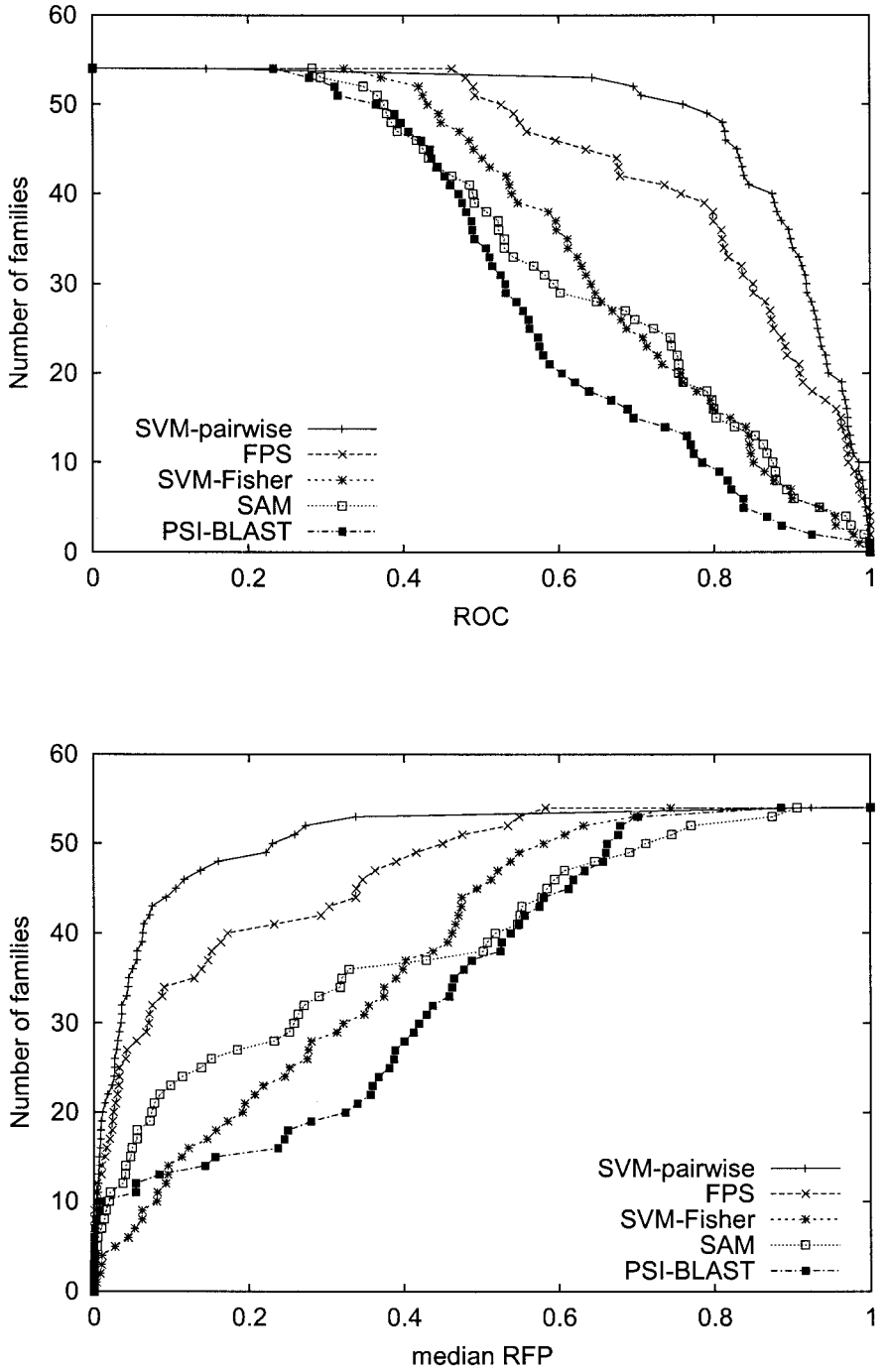

FIG. 3. Relative performance of the five homology detection methods. Each graph plots the total number of families for which a given method exceeds a score threshold. The **top** graph uses ROC scores, and the **bottom** graph uses median RFP scores. Each series corresponds to one protein homology detection method.

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

Some of these results agree with previous assessments. For example, the relative performance of SVM-Fisher and SAM agrees with the results given by Jaakkola *et al.* (1999), although in that work the difference

TABLE 2. STATISTICAL SIGNIFICANCE OF DIFFERENCES BETWEEN PAIRS OF HOMOLOGY DETECTION METHODS[a]

	FPS	*SVM-Fisher*	*SAM*	*PSI-BLAST*
SVM-pairwise	6.3e-03	3.3e-08	2.2e-08	1.0e-08
FPS		1.5e-05	5.4e-06	5.4e-09
SVM-Fisher			—	1.7e-02
SAM				—

[a]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.

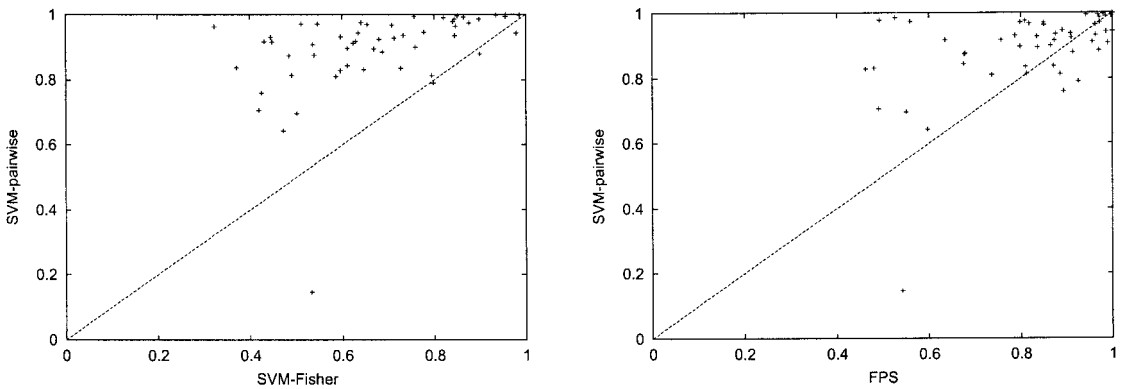

FIG. 4. Family-by-family comparison of SVM-pairwise with SVM-Fisher and FPS method. Each point on the graph corresponds to one of the SCOP superfamilies listed in Table 1. The y-axis in each plot is the ROC score achieved by the SVM-pairwise method. The x-axes in the two plots correspond to ROC scores from SVM-Fisher and FPS.

was more pronounced. Similarly, Park *et al.* (1998) showed that SAM significantly out-performs PSI-BLAST. Here, the difference between the two methods is not as pronounced.

One surprise in Fig. 3 is the relatively strong performance of the FPS algorithm on this task.[1] Previous assessments have shown FPS to outperform HMMs for close homologies (Grundy, 1998) but to perform worse than HMMs in recognizing remote relationships (Jaakkola *et al.*, 1999). These results suggest that, when only a small number of similar sequences are available, the FPS algorithm does a good job of exploiting those sequences.

The relative performance of the SVM-pairwise method is further illustrated in Fig. 4, which shows a family-by-family comparison of the 54 ROC scores computed for SVM-pairwise versus SVM-Fisher and FPS. The SVM-pairwise method scores higher than both other methods on nearly every family. The one outlier is family 2.44.1.2 (eukaryotic proteases), which has a relatively small training set. Family-by-family results from each of the seven methods are available at *www.cs.columbia.edu/compbio/svm-pairwise.*

4.2. Lesion studies

To understand how brains work, scientists perform lesion studies, removing a portion of the brain and examining the subsequent behavior of the subject. We are interested in understanding why the SVM-pairwise algorithm works as well as it does. Accordingly, we have performed a series of experiments with modified versions of the original algorithm.

[1]The FPS results reported in the earlier version of this paper were incorrect, due to a software bug.

First, in order to evaluate the necessity for an $O(n^2)$ sequence comparison algorithm, we repeated the above experiment using BLAST log p-values in place of the Smith–Waterman E-values in SVM-pairwise. The results, shown in Fig. 5(A), show intermediate performance between SVM-pairwise and the FPS algorithm. The signed-rank test indicates a significant difference between the two variants of SVM-pairwise ($p = 0.011$), but not between BLAST-based SVM-pairwise and FPS. This result is encouraging, because the modified SVM-pairwise algorithm is much more efficient than the original version and provides performance that is nearly as good.

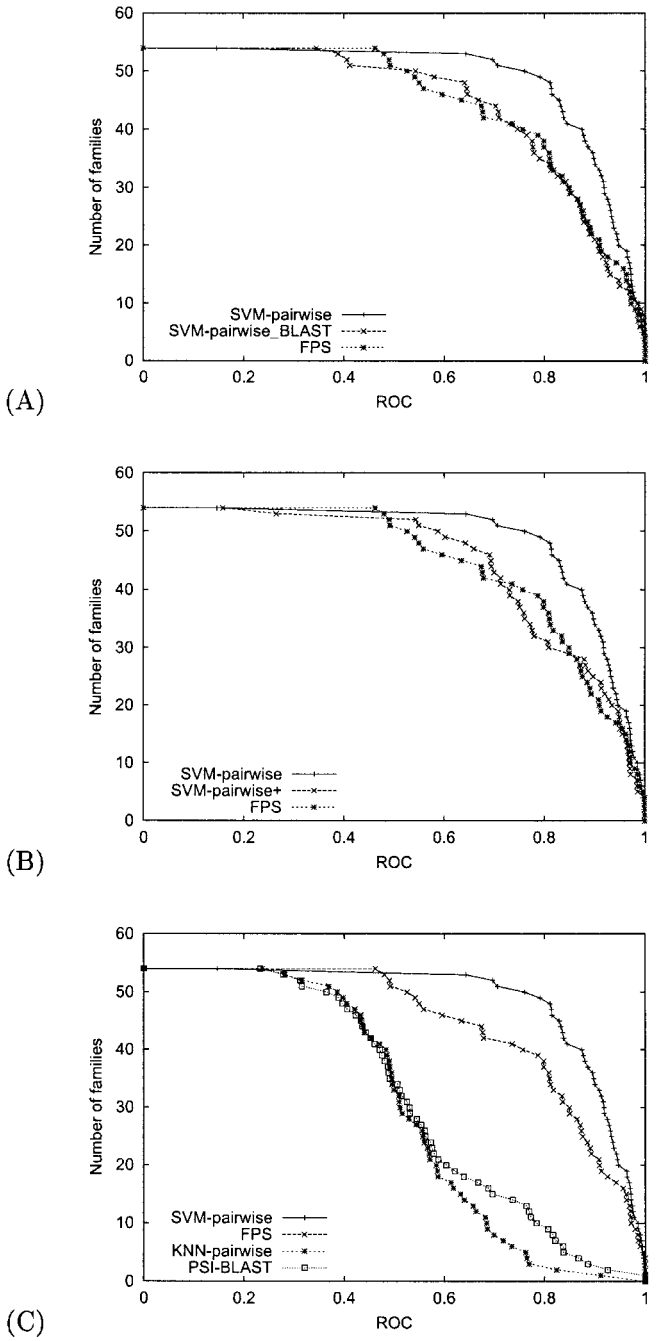

FIG. 5. Testing variants of the SVM-pairwise algorithm. (**A**) Replacing Smith-Waterman with the BLAST algorithm. (**B**) Removing negative examples from the vectorization set. (**C**) Replacing the SVM with the k-nearest neighbor algorithm.

Second, in order to evaluate the benefit provided by the negative elements in the pairwise score vector, we tested a version of SVM-pairwise in which the negative training set is not used during the creation of the score vectors. In this method, called SVM-pairwise$^+$, the negative examples are still used during the training of the SVM. The results in Fig. 5(B) are quite similar to the results from the previous modification: SVM-pairwise$^+$ performs significantly worse than the original algorithm ($p = 0.0005$) and not significantly better than FPS. Nonetheless, with respect to the range of available algorithms, the SVM-pairwise$^+$ algorithm performs quite well. This result implies that the power of SVM-pairwise does not lie primarily in the use of the negative training set during vectorization. Given the large size of the negative training set, SVM-pairwise$^+$ is considerably faster than SVM-pairwise and therefore provides a quite powerful, efficient alternative.

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.

As shown in Fig. 5(C), KNN-pairwise performs relatively poorly. The difference betwen KNN-pairwise and PSI-BLAST is not statistically significant. This result 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 (Bishop, 1995). We have no reason to suspect, however, that such an improvement would yield better performance than the SVM-pairwise method.

5. DISCUSSION

We have shown that, for the datasets used here, the SVM-pairwise method yields significantly improved remote homology detection relative to a number of existing, state-of-the-art algorithms. The key components of the algorithm include its discriminative approach, its use of a large-margin SVM classifer, and its straightforward means of converting a protein into vector form. We believe that this vectorization is particularly significant. Algorithms such as BLAST and the Smith–Waterman have undergone two decades of empirical optimization in the field of bioinformatics. Thus, considerable prior knowledge is implicitly incorporated into the pairwise sequence similarity scores and hence into the SVM-pairwise vector representation.

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. The BLAST-based version of SVM-pairwise reduces the running time to $O(n^2m)$.

There is, however, an important sense in which the SVM-pairwise algorithm is incomplete. In addition to learning from the given sequences, state-of-the-art algorithms such as PSI-BLAST and SAM-T99 pull in additional training set examples from a large, unannotated database of protein sequences. SVM-pairwise has not yet been implemented in an iterative framework. Therefore, in order to produce a fair comparison, the experiments reported here prevent any of the algorithms from iterating. Clearly, to make SVM-pairwise a usable algorithm in practice, unlabeled data must be included. This extension will be the subject of future work.

ACKNOWLEDGMENTS

We thank Mark Diekhans for providing access to detailed results from prior work as well as software for computing Fisher gradient vectors. We also thank Timothy Bailey and Jason Weston 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. The BioXLP resource was provided by the National Biomedical Computation Research at the San Diego Supercomputer Center. NBCR is funded by the National Center for Research Resources (P41 RR08605-07).

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. A basic local alignment search tool. *J. Mol. Biol.* 215, 403–410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search prorams. *Nucl. Acids Res.* 25, 3389–3402.

Bailey, T.L., and Grundy, W.N. 1999. Classifying proteins by family using the product of correlated *p*-values. In Istrail, S., Pevzner, P., and Waterman, M., eds., *Proc. 3rd Ann. Int. Conf. Comp. Mol. Biol.* 10–14. ACM.

Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M.A. 1994. Hidden Markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA* 91(3), 1059–1063.

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

Brenner, S.E., Koehl, P., and Levitt, M. 2000. The ASTRAL compendium for sequence and structure analysis. *Nucl. Acids Res.* 28, 254–256.

Brown, M.P.S., Grundy, W.N., Lin, D, Cristianini, N., Sugnet, C., Furey, Jr., T.S., Ares, M., and Haussler, D. 2000. Knowledge-based anlaysis of microarray gene expression data using support vector machines. *Proc. Natl. Acad. Sci. USA* 97(1), 262–267.

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

Gribskov, M., Lüthy, R., and Eisenberg, D. 1990. Profile analysis. *Methods Enzymol.* 183, 146–159.

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

Grundy, W.N. 1998. Family-based homology detection via pairwise sequence comparison. *Proc. 2nd Ann. Int. Conf. Computational Molecular Biology*, 94–100.

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

Jaakkola, T., Diekhans, M., and Haussler, D. 1999. Using the Fisher kernel method to detect remote protein homologies. *Proc. 7th Int. Conf. Intelligent Systems for Molecular Biology*, 149–158.

Jaakkola, T., Diekhans, M., and Haussler, D. 2000. A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.* 7(1–2), 95–114.

Karplus, K., Barrett, C., and Hughey, R. 1998. Hidden Markov models for detecting remote protein homologies. *Bioinformatics* 14(10), 846–856.

Krogh, A., Brown, M., Mian, I., Sjolander, K., and Haussler, D. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235, 1501–1531.

Liao, L., and Noble, W.S. 2002. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proc. 6th Ann. Int. Conf. Computational Molecular Biology*, 225–232.

Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C. 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536–540.

Ng, A.Y., Jordan, M., and Weiss, Y. 2002. On spectral clustering: Analysis and an algorithm, *in* Dietterich, T.G., Becker, S., and Ghahramani, Z., eds., *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA.

Park, J., Karplus, K., Barrett, C., Hughey, R., Haussler, D., Hubbard, T., and Chothia, C. 1998. Sequence comparison using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.* 284(4), 1201–1210.

Pearson, W.R. 1985. Rapid and sensitive sequence comparisons with FASTP and FASTA. *Methods Enzymol.* 183, 63–98.

Salzberg, S.L. 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery* 1, 317–328.

Smith, T., and Waterman, M. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.

Sonnhammer, E., Eddy, S., and Durbin, R. 1997. Pfam: A comprehensive database of protein domain families based on seed alignments. *Proteins* 28(3), 405–420.

Thompson, J.D., Higgins, D.G., and Gibson, T.J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* 22(22), 4673-4680.

Tsuda, K. 1999. Support vector classification with asymmetric kernel function. *Proc. ESANN*, 183–188.

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

Address correspondence to:
William Stafford Noble
Health Sciences Center
Box 357730
1705 NE Pacific Street
Seattle, WA 98195

E-mail: noble@gs.washington.edu