BIOINFORMATICS ORIGINAL PAPER Vol. 23 no. 9 2007, pages 1090–1098
doi:10.1093/bioinformatics/btl642

Structural bioinformatics

A structural alignment kernel for protein structures

Jian Qiu[1], Martial Hue[3,*], Asa Ben-Hur[4], Jean-Philippe Vert[3] and William Stafford Noble[1,2]

[1]Department of Genome Sciences, [2]Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA, [3]Center for Computational Biology, Ecole des Mines de Paris, Fontainebleau, France and [4]Department of Computer Science, Colorado State University, Ft. Collins, CO, USA

Received on August 17, 2006; revised on November 22, 2006; accepted on December 15, 2006

Advance Access publication January 18, 2007

Associate Editor: Anna Tramontano

ABSTRACT

Motivation: This work aims to develop computational methods to annotate protein structures in an automated fashion. We employ a support vector machine (SVM) classifier to map from a given class of structures to their corresponding structural (SCOP) or functional (Gene Ontology) annotation. In particular, we build upon recent work describing various *kernels* for protein structures, where a kernel is a similarity function that the classifier uses to compare pairs of structures.

Results: We describe a kernel that is derived in a straightforward fashion from an existing structural alignment program, MAMMOTH. We find in our benchmark experiments that this kernel significantly out-performs a variety of other kernels, including several previously described kernels. Furthermore, in both benchmarks, classifying structures using MAMMOTH alone does not work as well as using an SVM with the MAMMOTH kernel.

Availability: http://noble.gs.washington.edu/proj/3dkernel

Contact: noble@gs.washington.edu

1 INTRODUCTION

Given the large amount of effort currently devoted to determining protein structures, the downstream problem of determining a protein's function from its structure is increasingly important. This structure annotation problem can be formalized as a classification task. If we assume that functional annotations are drawn from a finite list of terms, then the task consists of assigning a subset of these terms to a given structure of unknown function. Note that the classification is both multi-class (i.e. there are many terms) and multi-label (i.e. a given structure can have more than one functional annotation).

Most of the previous research on inferring function from structure focuses on the related problem of inferring remote homology between pairs of structures. These methods can be used to infer the function of a novel structure by simple annotation transfer (Hegyi and Gerstein, 1999; Orengo *et al.*, 2002; Ponomarenko *et al.*, 2005). However, when a novel structure is homologous to multiple structures with varying annotations, then some further analytical method (e.g. clustering and voting) must be applied (Ponomarenko *et al.*, 2005). Furthermore, several analyses have demonstrated a general mismatch between structural clusters and functional annotations (Hegyi and Gerstein, 1999; Orengo *et al.*, 2002). Even highly similar structures may have quite distinct functions, often as a result of gene duplication followed by the evolution of new function.

A slightly more sophisticated approach to the structure annotation problem frames it as a supervised learning task. In this scenario, a classification algorithm learns, on the basis of an annotated training set of structures, a mathematical function that maps from protein structure to one or more annotation terms. The support vector machine (SVM) classifier (Boser *et al.*, 1992) is a strong candidate for application to this problem. SVMs provide state-of-the-art classification performance in a wide variety of application domains, including many applications in computational biology (Noble, 2004).

Recently, two research groups have applied SVMs to the structure annotation problem (Dobson and Doig, 2005; Borgwardt *et al.*, 2005). Both groups analyzed benchmark data sets based upon the top level of the enzyme classification (EC) hierarchy. In each case, the goal of the SVM classifier is to predict the enzyme class of a given unannotated protein structure. The two methods differ in the way that protein structures are represented in the SVM. Dobson and Doig use a vector representation based upon features such as secondary structure content, amino acid propensities, surface properties, etc. In contrast, Borgwardt *et al.* use a representation based upon walks defined on a graph of secondary structural elements. For the EC benchmark, this random walk approach provides better classification accuracy.

This contrast points to the central characteristic of any SVM-based classifier: the most crucial aspect of applying the algorithm is the selection of an appropriate representation of the data. In the SVM literature, this choice is known as the choice of *kernel*. SVMs are members of a large class of algorithms known as *kernel methods* (Schölkopf and Smola, 2002), in which the data representation is accomplished via a positive semidefinite function (the *kernel function*) that defines the similarity between all pairs of objects in the data set. Thus, in the kernel framework, the primary difference between the classifiers built by Dobson and Doig and Borgwardt *et al.* is their choice of kernel function.

*The first two authors contributed equally to this work.

In this work, we propose a simple kernel function that is a straightforward extension of the MAMMOTH structural alignment program (Ortiz *et al.*, 2002). SVMs trained using this kernel function and tested on two benchmarks—superfamily prediction within the structural classification of proteins (SCOP) (Murzin *et al.*, 1995) and prediction of gene ontology (GO) (Gene Ontology Consortium, 2000) terms—perform significantly better than the previously described kernels and better than several other kernels that we tested. Furthermore, for both of these benchmarks, using an SVM with the MAMMOTH kernel provides better performance than using a simple MAMMOTH-based annotation transfer approach.

The technique that we propose here is not specific to the MAMMOTH algorithm: a similar kernel could be derived from any protein structure comparison algorithm that returns a single score for each pair of compared structures. We have shown that, for tasks such as SCOP or GO classification, an SVM trained using this type of kernel yields better classification performance than the underlying structural comparison algorithm alone, and better performance than SVMs trained using kernels derived from explicit vectors of features, secondary structure graphs, contact maps, torsion angles, amino acid sequence or a combination of all of these features.

2 METHODS

2.1 Benchmark data sets

2.1.1 Enyzme classification benchmark For comparison with previous work, we first tested our battery of prediction methods on a benchmark that is derived from the EC hierarchy. This benchmark was created by Dobson and Doig (2005). The top level of the EC hierarchy consists of six enzyme classes, and the benchmark contains 498 PDB structures representing these classes, plus an additional 498 PDB structures of non-enzymes. In each class, no structure contains a domain from the same SCOP superfamily as any other structure. Therefore, it is not possible for domains from the same superfamily to be present more than once within a functional class, but it is possible for domains from the same superfamily to be present in more than one functional class.

2.1.2 SCOP classification benchmark For the SCOP benchmark, we test the ability of a classifier to recognize a novel family within a given superfamily, following the general approach of Jaakkola *et al.* (1999). We extracted a non-redundant data set from SCOP using the ASTRAL database version 1.67 (Brenner *et al.*, 2000) with a 95% sequence identity threshold. In a superfamily of N families, each classifier is trained on $N-1$ families and tested on the held-out family. We require a minimum of 10 proteins for training and 10 proteins for testing the classifier. Therefore, we eliminated any superfamily that did not contain at least one family with 10 or more members and at least 10 additional proteins outside of that family. This filtering resulted in a data set containing 4019 proteins from 397 families and 53 superfamilies; 102 families contain at least 10 members. Note that the held-out family design ensures low sequence similarity between domains in the training and test sets. In the benchmark, the highest sequence similarity between a domain in the training set and that in the test set is 48.28%. Among the 4019 domains in the SCOP benchmark, only 13 domains have BLAST hits with at least 40% sequence identity across a train/test split.

2.1.3 GO term prediction benchmark For the GO term prediction benchmark, we started with a set of 8363 PDB structures, pruned so that no two sequences share $\geq$50% sequence identity (Li *et al.*, 2001). From this set, we selected structures that have GO annotations, downloaded from http://www.ebi.ac.uk/GOA. To ensure that the GO term annotations are not based on sequence or structure similarity inference, we only selected the 1024 structures that are annotated with evidence codes IDA or TAS. For each GO term T, we partitioned the list of proteins into three sets. First, all proteins that are annotated with T are labeled as 'positive.' Next, we traverse from T along all paths to the root of the GO graph. At each GO term along this path, we look for proteins that are assigned to that term and not to any of that term's children. We consider that such proteins might be properly assigned to T, and so we label those proteins as 'uncertain.' Finally, all proteins that are not on the path from T to the root are labeled as 'negative.' For efficiency, we then randomly select a subset of the negative examples, so that the ratio of negatives to positives is 3-to-1.

After this labeling procedure, we eliminated all GO terms with fewer than 30 'positive' proteins. In order to avoid redundancy, we then selected only the most specific of the remaining GO terms, i.e. the leaf nodes of the remaining hierarchy. This procedure yielded a total of 23 GO terms: 11 molecular function terms, eight biological process terms and four cellular component terms.

2.2 Annotation transfer using MAMMOTH

As a baseline method, we use a one-nearest-neighbor classifier based upon MAMMOTH. Say that we are given a test structure with unknown SCOP superfamily or GO term annotation. We compare this test structure to each member of the training set using MAMMOTH, and we identify the training set structure with the smallest MAMMOTH E-value. We then assign to the test structure the same EC label, SCOP superfamily or GO term that is assigned to the identified training set structure. This technique is sometimes referred to as 'annotation transfer', and is commonly performed using sequence comparison algorithms such as PSI-BLAST.

2.3 SVMs and kernel functions

We compare the MAMMOTH nearest neighbor approach with a variety of SVM-based methods. These methods employ different kernel functions, each of which is defined on pairs of protein structures and focuses on different aspects of protein structure. In the following, we describe five structure kernels: the MAMMOTH kernel, two existing kernels (the vector kernel and the TOPS kernel) and kernels based on contact maps and C-α torsion angles. For comparison, we also include a kernel that is based only on sequence (the mismatch kernel) and a kernel that combines all six of the previous kernels.

2.3.1 MAMMOTH kernel Insofar as a kernel function defines the similarity between pairs of objects, the most natural place to begin defining a protein structure kernel is with existing pairwise structure comparison algorithms. Many such algorithms exist, including CE (Shindyalov and Bourne, 1998) DALI (Holm and Sander, 1993) and MAMMOTH (Ortiz *et al.*, 2002). Most of these algorithms attempt to create an alignment between two proteins and then compute a score that reflects the alignment's quality. In this work, we use MAMMOTH (Ortiz *et al.*, 2002), which is efficient and produces high quality alignments.

Unfortunately, the alignment quality score returned by MAMMOTH cannot be used as a kernel function directly, because the score is not positive semidefinite (i.e. for a given set of protein structures, an all-versus-all matrix of MAMMOTH scores will have some negative eigenvalues). We therefore employ the so-called 'empirical kernel map' (Tsuda, 1999) to convert this score to a

kernel: for a given data set of structures $X = x_1, \ldots, x_n$, a structure x_i is represented as an n-dimensional vector, in which the jth entry is the MAMMOTH score between x_i and x_j. The SVM then uses this vector representation directly. This method has been used successfully in the SVM-pairwise method of remote protein homology detection (Liao and Noble, 2002), in which a protein is represented as a vector of log E-values from a pairwise sequence comparison algorithm. In our experiments, we use the log of the E-value returned by MAMMOTH. The resulting MAMMOTH kernel incorporates information about the alignability of a given pair of proteins.

2.3.2 Vector kernel The vector kernel is a reimplementation of the protein structure representation employed by Dobson and Doig (2005). Each protein is described by a vector of 55 features. The first 20 features are amino acid frequencies. The second 20 are, for each type of amino acid, the fraction of that amino acid type's surface area that lies on the surface of the protein. In addition, the total number of residues and total surface area are treated as two additional features. Total surface area and surface area for each residue are computed with NACCESS (http://wolf.bms.umist.ac.uk/naccess). Another two surface-based features are the surface-area-to-volume ratio and the fractal dimension, both computed with MSMS (Sanner *et al*., 1996). Three features represent the secondary structure content of the protein, as the fraction of residues in helix, sheet and coil, respectively. Secondary structure is computed with Stride (Frishman and Argos, 1995). The remaining features take into account the presence of certain cofactors (ATP, FAD, NAD), ions (Ca, Cu, Fe, Mg) and disulphide bonds.

All non-binary feature values v are linearly rescaled according to the following formula: $\hat{v} = (v - v_{\min})/(v_{\max 0.99} - v_{\min})$, where $v_{\min}$ is the minimum value of the given feature, and $v_{max0.99}$ is the 99% quantile of the feature. This quantile-based rescaling avoids the effects of a few outliers with very large values.

2.3.3 TOPS kernel Borgwardt *et al*., (2005) proposed a kernel based on a representation of a protein structure as a graph whose nodes are secondary structural elements and edges are defined by proximity in the protein structure. Such a representation was first considered in the TOPS diagrams (Westhead *et al*., 1999) from which our implementation of this kernel derives its name. The labeled graph $G = (V, E)$ for a protein structure has a node v for each secondary structural element, and an associated type, where $type(v) \in \{H, E, C\}$, indicating whether the secondary structure element v is a helix, strand or coil, respectively. The graph has an edge (u, v) if the two elements u and v are consecutive in sequence, or have two C-α atoms whose distance is less than some threshold (10 Å).

To compare graphs representing two structures, Borgwardt *et al*. define a kernel that compares equal-length walks on the two graphs. This kernel, in turn, depends on kernels on edges and nodes in the walks. In our implementation, these kernels take into account the properties of the amino acids in a given secondary structural element: average hydrophobicity [quantified by the Kyte-Doolittle index (Kyte and Doolittle, 1982)], average hydrophilicity [quantified by the Hopp–Woods index (Hopp and Woods, 1981)], and fraction of hydrophobic, hydrophilic, positively charged, negatively charged and cysteine residues. The final kernel is a sum of all kernels for walks of different lengths. More details about this kernel are given in the online supplement.

2.3.4 Torsion kernel The torsion kernel represents a protein structure in terms of the torsion angles between adjacent C-α atoms. The computation of the kernel proceeds in three steps. First, we represent each chain of the structure by a sequence of torsion angles of the backbone. In this representation, a torsion angle is associated with each set of four successive C-α atoms, and is defined as the angle between the two successive planes containing three C-α atoms. Second, we discretize each sequence of angles. The angles (0–360) are divided into n_B bins of equal width, which we represent using letters, thereby effectively converting the torsion angles into a string representation of the protein backbone. In the third step, we apply a spectrum kernel (Leslie *et al*., 2002) to this torsion string. The spectrum kernel represents a string as a vector of counts of all possible substrings of a fixed length k. This three-step procedure defines a kernel on pairs of chains, which are extended to multi-chain structures by summing. In this work, we compute four different torsion kernels using two values of n_B (5 and 15) and two values of k (3 and 5). Each feature vector is normalized to unit length, and the resulting vectors are summed to create the final torsion kernel feature vector.

2.3.5 Contact kernel The contact kernel attempts to capture pairwise and multi-body interactions among amino acids in a protein structure. A family of kernels is considered, with two varying parameters: a distance threshold and the number of residues in each interaction (n_C). For each possible set of n_C amino acids, the kernel counts the number of times that those amino acids appear in an n_C-way interaction. Two residues are considered to be in contact if there exists at least one atom from each residue within the predefined distance threshold. However, only residue pairs with a sequence separation of at least three amino acids are considered. More generally, a set of n_C residues are considered to participate in an n_C-body interaction if every pair of the n_C residues are in contact. In the experiments reported here, we use six different kernels: a pairwise interaction kernel with distance thresholds of 3.5, 5 and 6.5 Å, a three-body interaction kernel with distance thresholds of 5 and 6.5 Å and a four-body interaction kernel with distance threshold 6.5 Å. Thus, we compute six separate feature vectors to describe a single protein structure. Each feature vector is first normalized to have unit length, and the six vectors are then concatenated to form a single vector.

2.3.6 Mismatch kernel The mismatch kernel (Leslie *et al*., 2003) is not a structure-based kernel, but instead looks only at the amino acid sequence of the protein. This kernel generalizes upon the spectrum kernel by considering shared k-length strings (k-mers), allowing for mismatches. In this work, we use a mismatch kernel with $k = 4$ and $n = 1$. The final mismatch spectrum vector has $20^4 = 160\,000$ bins and is normalized to unit length. The mismatch spectrum captures sequence similarity, and has been shown to out-perform PSI-BLAST in classifying SCOP superfamilies (Leslie *et al*., 2003).

2.3.7 Sum kernel Finally, we tested the ability of the SVM to synthesize the information from all six of the previously described kernels. We do this by summing the normalized kernels in an unweighted fashion. This is equivalent to concatenating the corresponding feature vectors, yielding a feature space whose dimensionality is equal to the sum of the dimensions of the six individual feature spaces.

2.3.8 Computational complexity The six basic kernels that we tested have different computational complexities. Four of them—vector, torsion, contact and MAMMOTH—can be implemented by representing each structure as an explicit vector in a particular feature space. For the vector, torsion and contact kernels, this computation is fast: for our benchmarks, computing these vector representations and then applying the radial basis kernel requires a few minutes to a few hours. However, for the MAMMOTH kernel, the computation of each MAMMOTH score requires a dynamic programming alignment algorithm that scales as the product of the lengths of the two proteins to be compared. In practice, the computation of the MAMMOTH kernel matrices for our benchmarks required several CPU days. The mismatch and TOPS kernel cannot be written as an inner product

between explicit vector representations. However, the mismatch kernel can be computed efficiently (i.e. in minutes on our data sets) using a mismatch tree data structure. In contrast, computing a single TOPS kernel value scales as the third power of the product of the number of secondary structure elements in the proteins. Like the MAMMOTH kernel, the TOPS kernel computations required several CPU days.

2.4 Experimental framework

All experiments were performed using PyML (http://pyml.sourceforge.org). We trained one-versus-all classifiers for each EC label, SCOP superfamily or GO term. For the SCOP benchmark, each method was tested using the family-based hold-out procedure described above. For the GO benchmark, each method was tested using 5-fold cross-validation, repeated three times (3×5cv).

Prior to running the SVM experiments, each of the kernels described above is first centered and normalized, and then converted to a radial basis function kernel with width 1. For each SVM, an appropriate value of the regularization parameter was selected from $\{0.1, 1, 10, 100\}$ by using a second level of cross-validation within the given training set.

For each prediction method, we compute two performance metrics: the area under the ROC curve (ROC) and the area under this curve up to the first 10% of the false positives ($ROC_{10\%}$). The $ROC_{10\%}$ focuses only on the top-ranked examples, which are of most interest in some applications. For GO, each metric is averaged across the fifteen 3×5cv splits.

3 RESULTS

3.1 EC benchmark

For the EC benchmark, we compared the performance of all eight classifiers on seven one-versus-all classification tasks, corresponding to the six enzyme classes and to the enzyme/non-enzyme task. For this experiment, we report mean ROC and $ROC_{10\%}$ scores across 3×5cv. The results, shown in Table 1 and the online supplement, show that the vector kernel yields good performance relative to the other kernels, with the best ROC score in five out of seven cases. The MAMMOTH kernel does not perform best for any class. This result is not surprising, because as previously noted, the EC benchmark is purged so that no class contains two members of the same SCOP superfamily. As such, any kernel based on structural alignment is not well suited to this task.

3.2 SCOP benchmark

We compared the performance of all seven kernels on the SCOP benchmark. The results, shown in Fig. 1A–B, clearly show that the MAMMOTH kernel dominates all of the other kernels. Indeed, the MAMMOTH kernel yields perfect classification performance for approximately half of the SCOP families in the benchmark.

By eye, it is diffult to ascertain which differences in Fig. 1A–B are statistically significant. Therefore, we performed a Wilcoxon signed-rank test on the ROC scores for all pairs of kernels in our experiment. The resulting Bonferroni adjusted, one-tailed *P*-values are given in Table 2. Using either metric, the results show that the MAMMOTH kernel significantly outperforms the sum kernel, which in turn significantly outperforms all other methods. The TOPS, torsion, mismatch, vector and contact kernels provide statistically indistinguishable performance, using either performance metric.

The MAMMOTH kernel is clearly very powerful. To test whether the SVM adds value in this task, we also tested the one-nearest neighbor classifier based upon the MAMMOTH *E*-value. The results from this classifier are labeled 'MAMMOTH NN' in Figure 1A–B. Its performance is significantly worse than the MAMMOTH SVM classifier; hence, on this benchmark, using the SVM does indeed improve upon using MAMMOTH alone.

3.3 GO benchmark

Among the three benchmarks, the GO benchmark is the most diverse, spanning all three sections of the GO. The results from experiments on this benchmark, summarized in Fig. 1C–D, are consistent with the results on the SCOP benchmark. In general, however, due to its small size, the GO benchmark provides less discrimination among methods (Table 2). Using either metric, the GO benchmark divides the methods into two groups: the MAMMOTH, sum, mismatch and MAMMOTH nearest-neighbor methods perform better than the contact, vector, TOPS and torsion kernels. There is some evidence that, among the better-performing group of methods, the MAMMOTH nearest-neighbor is the worst: its performance is statistically indistinguishable from that of the vector kernel by ROC score and from the contact kernel by $ROC_{10\%}$ score.

Table 1. EC benchmark results

ROC	Hydrolase	Isomerase	Ligase	Lyase	Oxidoreductase	Transferase	Enz-NEnz
MAMMOTH NN	0.54 ± 0.02	0.39 ± 0.04	0.63 ± 0.08	0.38 ± 0.04	0.56 ± 0.03	0.50 ± 0.02	0.42 ± 0.01
MAMMOTH	0.57 ± 0.02	0.44 ± 0.01	0.64 ± 0.04	0.49 ± 0.02	0.53 ± 0.01	0.54 ± 0.01	0.61 ± 0.01
Vector	**0.70 ± 0.01**	**0.63 ± 0.02**	0.69 ± 0.01	0.57 ± 0.02	**0.72 ± 0.03**	**0.64 ± 0.00**	0.72 ± 0.01
Torsion	0.65 ± 0.01	0.61 ± 0.00	0.57 ± 0.02	**0.61 ± 0.00**	0.56 ± 0.01	0.58 ± 0.01	0.67 ± 0.01
Contact	0.64 ± 0.00	0.53 ± 0.03	**0.75 ± 0.02**	0.59 ± 0.01	0.70 ± 0.01	0.62 ± 0.01	**0.75 ± 0.01**
Mismatch	0.67 ± 0.01	0.50 ± 0.06	0.60 ± 0.02	0.53 ± 0.02	0.66 ± 0.01	0.57 ± 0.01	0.70 ± 0.00
TOPS	0.60 ± 0.02	0.55 ± 0.01	0.47 ± 0.06	0.53 ± 0.02	0.60 ± 0.00	0.55 ± 0.02	0.68 ± 0.01
Sum	0.69 ± 0.01	0.61 ± 0.00	0.70 ± 0.05	0.60 ± 0.01	0.67 ± 0.00	0.62 ± 0.02	0.73 ± 0.01

The table lists for each kernel and each enzyme class, the mean and standard deviations of the ROC scores for one-versus-all SVM classifiers trained using 3×5cv. The highest score in each column is indicated by boldface type.

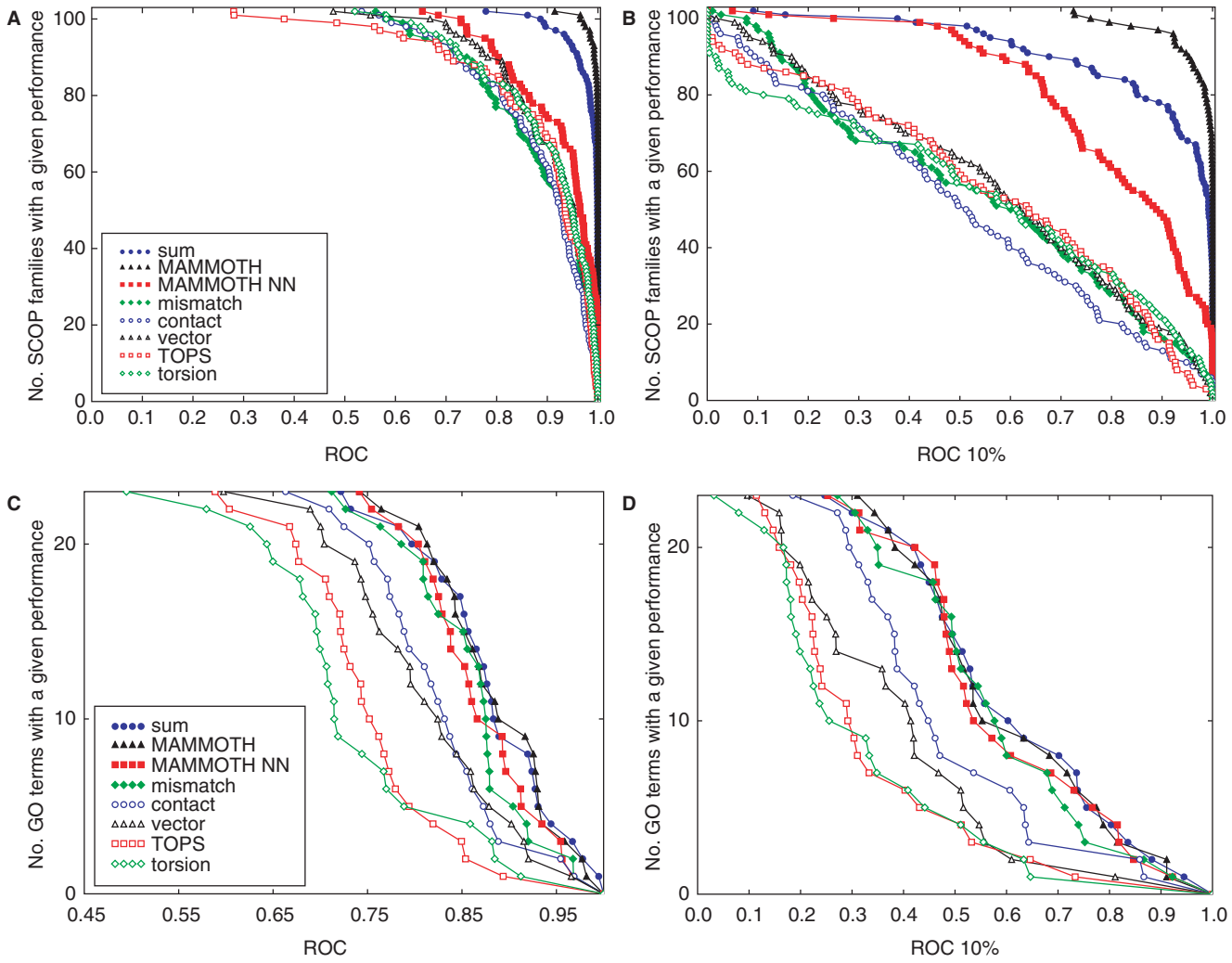

Fig. 1. SCOP and GO benchmark results. Each figure plots on the *y*-axis the number of SCOP families (panels A and B) or GO terms (panels C and D) for which a given SVM classifier achieves a specified ROC or $ROC_{10\%}$ score (*x*-axis). Each series corresponds to a different kernel, with the exception of 'MAMMOTH NN', which is a nearest-neighbor classifier.

Table 3 compares the performance of the MAMMOTH kernel and all of the other kernels on individual GO terms within the benchmark. Overall, the MAMMOTH kernel yields the best performance for 16 of the 23 GO terms. By either performance metric, the MAMMOTH kernel generally performs best for molecular function GO terms, with the exception of transmembrane receptor activity. On the other hand, not surprisingly, the MAMMOTH kernel does a poor job of predicting protein localization. None of the cellular component GO terms appears in the group of terms for which the MAMMOTH kernel improves most over the mismatch kernel, and cellular component GO terms are enriched in the group of terms where MAMMOTH performs worse than other kernels.

Among the GO molecular function terms for which MAMMOTH performs well, quite a few relate to catalytic enzyme activity. For instance, oxidoreductase and two types of hydrolases are among the top 10 GO terms exhibiting the most improvements relative to the mismatch kernel for both ROC and $ROC_{10\%}$. Surprisingly, however, the MAMMOTH kernel performs poorly on the EC benchmark in the classification of ligase, lyase, oxidoreductase, transferase, isomerase and hydrolase. The likely explanation for this apparent discrepancy lies in the method by which the EC benchmark was constructed by Dobson and Doig. The benchmark specifically subtracts structural information, by disallowing two proteins with domains in the same SCOP superfamily to appear in a single enzyme class.

In order to identify which GO terms the MAMMOTH and mismatch kernels have the most trouble with, we investigated the annotations of the top-ranked false positives assigned by these two kernels for each GO term. For each test set in the cross-validation, we examined the top *n* predictions, where *n* is the total number of positive examples in the

Table 2. Pairwise comparison of kernels for the SCOP and GO benchmarks

SCOP	ROC							$ROC_{10\%}$						
	Sum	M-NN	Vec	Tor	TOPS	Mis	Con	Sum	M-NN	TOPS	Vec	Tor	Mis	Con
MAMMOTH	6.1e-05	0	0	0	0	0	0	9.8e-07	0	0	0	0	0	0
Sum		7e-08	0	0	0	0	0		0.0022	0	0	0	0	0
MAMMOTH NN			—	—	—	0.0063	0.03			2e-07	3.4e-07	2.5e-07	0	0
Vector				—	—	—	0.02					—	—	—
Torsion					—	—	—						—	—
TOPS						—	—				—	—	—	—
Mismatch							—							—

GO	ROC							$ROC_{10\%}$						
	MAM	Mis	M-NN	Con	Vec	TOPS	Tor	MAM	Sum	M-NN	Con	Vec	TOPS	Tor
Sum	—	—	—	0.00076	0.00076	0.00076	0.00076			—	0.0013	0.00076	0.00076	0.00076
MAMMOTH		—	—	0.0017	0.00099	0.00076	0.00076		—	—	0.032	0.0015	0.00086	0.00086
Mismatch			—	0.021	0.021	0.00076	0.00076	—	—	—	0.0045	0.0011	0.00086	0.00099
MAMMOTH NN				0.029	—	0.0011	0.00076				—	0.0045	0.0019	0.0017
Contact					—	0.0013	0.00086					—	0.0024	0.0019
Vector						0.013	0.0083						—	—
TOPS							—							—

The table lists Bonferroni adjusted, one-tailed *P*-values from a Wilcoxon signed rank test performed on ROC and $ROC_{10\%}$ scores from all superfamilies in the SCOP benchmark (top) and all terms in the GO benchmark (bottom). Only *P*-values <0.05 are reported. A significant *P*-value in the table indicates that the kernel in the corresponding row outperforms the kernel in the corresponding column. A dash indicates that the median value for the row kernel exceeds the median value for the column kernel, but that the difference is not significant.

test set. A perfect classifier will assign no false positives within this set. For each GO term prediction, we calculated the fraction of highly ranked false positives that are annotated with another GO term. We compute this fraction with respect to all five cross-validation folds, and then average across the three repetitions. Figure 2 shows the resulting confusion matrix for the mismatch and MAMMOTH kernels (panels A and B, respectively). In order to show contrast, panel C shows the difference between the matrices in A and B.

First, we note that the two matrices look qualitatively similar. This indicates that the two kernels tend to make similar false positive predictions. In this analysis, both kernels tend to perform poorly in predicting cellular component terms. Both kernels also make a similar pattern of mistakes among a subset of the biological process terms. The GO term 'nucleus' has 149 positives, much larger than any other GO term. As a result, quite a few GO term predictions have a large fraction of false positives annotated with the GO term 'nucleus.'

Due to GO term dependencies that are not represented in the ontology, the results shown in Fig. 2 occasionally overestimate false positive rates. For example, in both the mismatch and MAMMOTH kernel results, the term 'purine nucleotide binding' has the largest fraction of false positives annotated with 'protein amino acid phosphorylation' and 'protein–tyrosine kinase activity'. Because kinases require ATP to phosphorylate their targets, these protein most likely bind purine nucleotides. It is not surprising, therefore, that both kernels make predictions of protein–tyrosine kinases or proteins participating in amino acid phosphorylation as purine nucleotide binding proteins. Thus, it seems likely that these proteins represent true positives instead of false positives.

In a similar case, both the mismatch and MAMMOTH kernels tend to incorrectly assign the term 'signal transduction' to proteins participating in protein amino acid phophorylation or transport. Because signal transduction cascades often involve phosphorylation of downstream proteins, it is not surprising that both kernels predict proteins in phosphorylation processes as signal transduction proteins. In addition, signal transduction and transport processes are often coupled, with the latter regulated by the former. One example is the protein structure 1j2j, which is an ADP-ribosylation factor (ARF1) binding GGA1. ARF1 is a small G protein regulating membrane traffic. GGA1 is a Golgi-localized, gamma adaptin ear-containing, ARF-binding protein that is involved in protein sorting between the trans-Golgi network and the lysosome. Due to incomplete GO annotation, it is possible that some transport proteins are also part of a signal transduction process, although they are not annotated in our data set.

A third example of dependencies not captured by our data set is illustrated by the false positives associated with 'biosynthesis'. These false positives are distributed across multiple GO terms, especially in the case of the mismatch kernel. This indicates the involvement of multiple functions

Table 3. GO benchmark results

GO term		MAMMOTH	Mismatch	Contact	Vector	Random walk	Torsion	Sum	MAMMOTH NN
GO:0005215	MF transporter activity	**0.842**	0.712	0.757	0.782	0.602	0.643	0.848*	0.780
GO:0016788	MF hydrolase activity, acting on ester bonds	**0.820***	0.727	0.709	0.597	0.667	0.495	0.732	0.820
GO:0004713	MF protein-tyrosine kinase activity	**0.935***	0.867	0.817	0.743	0.722	0.789	0.883	0.857
GO:0005125	MF cytokine activity	**0.982***	0.920	0.855	0.902	0.773	0.714	0.978	0.956
GO:0005509	MF calcium ion binding	**0.926**	0.870	0.861	0.920	0.724	0.770	0.944*	0.896
GO:0004175	MF endopeptidase activity	**0.928***	0.873	0.889	0.796	0.780	0.859	0.920	0.913
GO:0016491	MF oxidoreductase activity	**0.931***	0.879	0.831	0.879	0.705	0.695	0.931	0.893
GO:0003700	MF transcription factor activity	**0.871***	0.825	0.824	0.829	0.820	0.708	0.865	0.829
GO:0009058	BP biosynthesis	**0.804**	0.763	0.771	0.762	0.762	0.715	0.821*	0.741
GO:0006810	BP transport	**0.843**	0.809	0.773	0.700	0.590	0.650	0.797	0.866*
GO:0006351	BP transcription, DNA-dependent	**0.885***	0.856	0.783	0.844	0.752	0.678	0.881	0.838
GO:0006468	BP protein amino acid phosphorylation	**0.917**	0.904	0.788	0.756	0.730	0.700	0.924	0.935*
GO:0006508	BP proteolysis and peptidolysis	**0.930**	0.918	0.880	0.859	0.854	0.882	0.927	0.954*
GO:0045449	BP regulation of transcription	**0.861***	0.851	0.810	0.824	0.792	0.681	0.852	0.811
GO:0008236	MF serine-type peptidase activity	**0.977**	0.968	0.969	0.966	0.894	0.912	0.995*	0.967
GO:0005634	CC nucleus	**0.888**	0.879	0.844	0.795	0.721	0.744	0.889	0.893*
GO:0017076	MF purine nucleotide binding	0.813	**0.814**	0.725	0.748	0.677	0.626	0.828*	0.804
GO:0016021	CC integral to membrane	0.868	**0.875***	0.794	0.737	0.709	0.696	0.857	0.853
GO:0007596	BP blood coagulation	0.959	**0.968***	0.955	0.916	0.849	0.885	0.967	0.912
GO:0004888	MF transmembrane receptor activity	0.852	**0.876**	0.873	0.863	0.769	0.767	0.876*	0.825
GO:0005737	CC cytoplasm	0.834	**0.877***	0.837	0.810	0.744	0.719	0.873	0.860
GO:0007165	BP signal transduction	0.742	**0.786***	0.663	0.689	0.675	0.579	0.722	0.754
GO:0043234	CC protein complex	0.765	**0.809**	0.752	0.704	0.737	0.707	0.783	0.838*

The table lists, in each row, a GO term and the corresponding mean ROC scores generated by the various SVM kernels and by the MAMMOTH nearest neighbor classifier. Values in boldface are the maximum among the six individual kernels. Asterisked values are the maximum among all eight methods. Rows are sorted by the difference between MAMMOTH and mismatch SVM mean ROC scores. A similar table for $ROC_{10\%}$ is in the online supplement.

in the biosynthesis process. For instance, biosynthesis requires an energy source and hence ATP-binding proteins. It also often involves oxidoreductase activity and electron transport.

In other cases, the SVM simply confuses GO terms that are closely related. For example, the mismatch kernel tends to assign the term 'hydrolase activity, acting on ester bonds' to endopeptidases and to serine-type peptidases. Because both types of peptidases are hydrolases, the mismatch kernel probably learns to predict hydrolase activity instead of the more specific hydrolase activity acting on ester bonds. Interestingly, only the mismatch kernel makes this type of mistake.

Finally, the reason for some types of false positives is unclear. The most striking example is in the prediction of the GO term 'hydrolase activity, acting on ester bonds': in this case, the MAMMOTH kernel SVM assigns the largest fraction of false positives to proteins annotated with 'transcription factor activity'. The mismatch kernel also predicts some transcription factors as positives.

The difference between the mismatch and MAMMOTH confusion matrices is shown in Fig. 2C. As mentioned above, we find that the types of false positive assignments are generally similar between the two sets of results. The two methods differ in the extent, but not the pattern of false positives. We could find no satisfying explanation for the few cases in which the mismatch and MAMMOTH differed. For example, when predicting 'regulation of transcription' the mismatch kernel assigns many false positives that have cytokine activity, which is not obviously related to transcriptional regulation.

4 DISCUSSION

We have compared the performance of five protein structure kernels, a sequence kernel and the unweighted sum of all six

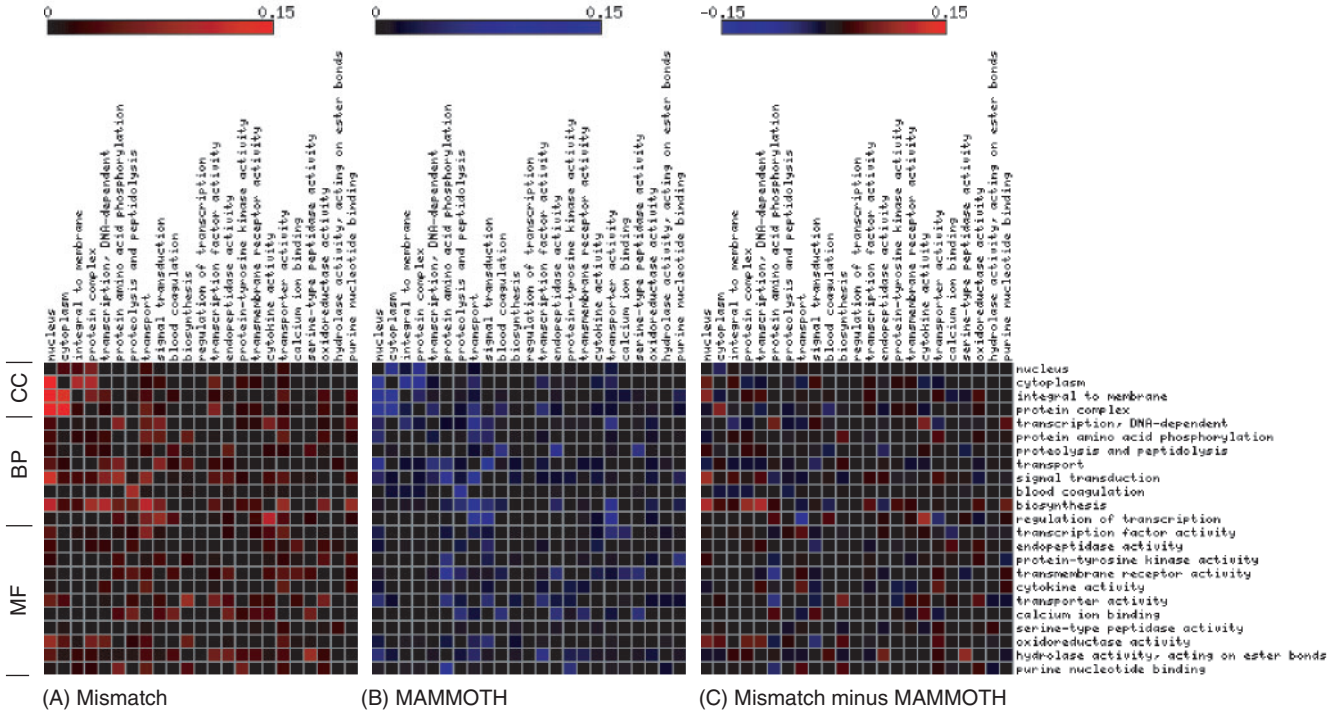

Fig. 2. Confusion matrices for mismatch and MAMMOTH kernels. Within each matrix, the value displayed in row *i* and column *j* is the average fraction of false positives annotated with GO term *j* when predicting GO term *i*. Panel A is for predictions made using the mismatch kernel; panel B is for the MAMMOTH kernel, and panel C shows the difference between panels A and B. The heat maps were generated using matrix2png (Pavlidis and Noble, 2003).

kernels. Among these, the MAMMOTH kernel is clearly the most powerful, yielding highly accurate classification performance across the SCOP and GO benchmarks.

We have reported results on the EC benchmark primarily because others have used this benchmark (Dobson and Doig, 2005; Borgwardt *et al*., 2005). However, we do not emphasize the EC results because we believe that the benchmark is flawed. In the benchmark, two SCOP domains from the same superfamily are not allowed to occur in the same EC class but are allowed to occur in different EC classes. This setup removes most of the information that a sequence- or structure-based method can use. In fact, all of the methods that we tested yield poor performance on the EC benchmark, with ROC scores rarely exceeding 0.7. The vector kernel yields slightly better performance presumably because it includes additional informative features; for example, the presence of FAD is indicative of oxidoreductase activity.

An important caveat for the comparison of the structure-based and sequence-based kernels is that, for the purposes of this study, we have restricted the mismatch kernel to using only sequences from proteins for which the 3D structure is known. In practice, for the GO benchmark, many protein sequences are available for which no structure is known. We expect that the performance of the mismatch kernel, or any other sequence-based kernel, would improve dramatically if we used all available sequence data.

For the classification tasks that we investigated, the simple combination of kernels provided by the sum kernel does not improve upon the best single-kernel learning method. This result might arise because the difference in performance between the best-performing MAMMOTH kernel and the other kernels is so large. In the future we will investigate more sophisticated ways to optimize linear combinations of kernels for each problem (Lanckriet *et al*., 2004; Bach *et al*., 2004). On the other hand, the observation that some kernels provide good performance on a subset of GO terms suggests a simple way of leveraging our repertoire of kernels: rather than performing kernel summation, we can choose the best kernel for a particular term using cross-validation.

One question raised by the current study is why the TOPS kernel performs so poorly. This kernel was intended to essentially replicate the random walks kernel of Borgwardt *et al*. However, the published results for this kernel on a variant of the EC benchmark (Borgwardt *et al*., 2005) suggest that the random walks kernel out-performs the vector kernel. We initially intended to include the random walks kernel in this study, using code provided by Borgwardt *et al*. However, that kernel is extremely costly to compute, and computing the kernel matrices for our benchmarks would require more computational resources than are currently available to us.

In conclusion, we have described a straightforward method to derive a protein structure kernel from an existing structure alignment algorithm, and we have demonstrated that, for the SCOP and GO benchmarks used here, this kernel outperforms a variety of alternative protein structure kernels.

ACKNOWLEDGEMENTS

The authors thank Ora Furman for useful discussions and Charles E. Grant for help with preparation of the GO benchmark. This work was funded by National Institutes of Health awards NHGRI R33 HG003070 and NCRR P41 RR11823.

Conflict of Interest: none declared.

REFERENCES

Bach, F. R. *et al*. (2004). Computing regularization paths for learning multiple kernels. In Saul, L. K. *et al*. (eds) *Advances in Neural Information Processing Systems*, Cambridge, MA, MIT Press.

Borgwardt, K.M. *et al*. (2005) Protein function prediction via graph kernels. *Bioinformatics*, **21**(Suppl. 1), i47–i56.

Boser, B. E. *et al*. (1992) A training algorithm for optimal margin classifiers. In Haussler, D. (ed.), *5th Annual ACM Workshop on COLT*, Pittsburgh, PA, ACM Press, pp. 144–152.

Brenner, S. E. *et al*. (2000) The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, **28**, 254–256.

Dobson, P. D. and Doig, A. J. (2005) Predicting enzyme class from protein structure without alignments. *J. Mol. Biol.*, **345**, 187–199.

Frishman, D. and Argos, P. (1995) Stride: knowledge-based protein secondary structure assignment. *Proteins: Struct. Funct. Genet.*, **23**, 566–579.

Gene Ontology Consortium. (2000) Gene ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–9.

Hegyi, H. and Gerstein, M. (1999) The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *J. Mol. Biol.*, **288**, 147–164.

Holm, L. and Sander, C. (1993) Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, **233**, 123–138.

Hopp, T. P. and Woods, K. R. (1981) Prediction of protein antigenic determinants from amino acid sequences. *Proc. Natl. Acad. Sci.* USA, **78**, 3824–3828.

Jaakkola, T. *et al*. (1999) Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, AAAI Press, pp. 149–158.

Kyte, J. and Doolittle, R. F. (1982) A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*, **157**, 105–132.

Lanckriet, G. R. G. *et al*. (2004) A statistical framework for genomic data fusion. *Bioinformatics*, **20**, 2626–2635.

Leslie, C. *et al*. (2002) The spectrum kernel: A string kernel for SVM protein classification. In Altman, R. B. *et al*. (eds) *Proceedings of the Pacific Symposium on Biocomputing*, pp. 564–575, New Jersey, World Scientific.

Leslie, C. *et al*. (2003) Mismatch string kernels for SVM protein classification. In Becker, S. *et al*. *Advances in Neural Information Processing Systems*, Cambridge, MA, MIT Press, pp. 1441–1448.

Li, W. *et al*. (2001). Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, **17**, 282–283.

Liao, L. and Noble W. S. (2002) Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology*, Washington, DC, April 18–21, pp. 225–232.

Murzin, A. G. *et al*. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.

Noble, W. S. (2004) Support vector machine applications in computational biology. In Schoelkopf, B. *et al*. (eds) *Kernel Methods in Computational Biology*, Cambridge, MA, MIT Press, pp. 71–92.

Orengo, C. A. *et al*. (2002) The CATH protein family database: a resource for structural and functional annotation of genomes. *Proteomics*, **2**, 11–21.

Ortiz, A. R. *et al*. (2002) MAMMOTH (Matching molecular models obtained from theory): an automated method for model comparison. *Protein Sci.*, **11**, 2606–2621.

Pavlidis, P. and Noble, W. S. (2003) Matrix2png: a utility for visualizing matrix data. *Bioinformatics*, **19**(2), 295–296.

Ponomarenko, J. V. *et al*. (2005) Assigning new GO annotations to Protein Data Bank sequences by combining structure and sequence homology. *Proteins: Structure, Function and Bioinformatics*, **58**, 855–865.

Sanner, M. F. *et al*. (1996) Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, **38**, 305–320.

Schölkopf B. and Smola, A. (2002) *Learning with Kernels*. Cambridge, MA, MIT Press.

Shindyalov, I. N. and Bourne, P. E. (1998) Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Eng.*, **11**, 739–747.

Tsuda, K. (1999) Support vector classification with asymmetric kernel function. In Verleysen, M. (ed.) *Proceedings ESANN*, pp. 183–188.

Westhead, D.R. *et al*. (1999) Protein structural topology: automated analysis, diagrammatic representation and database searching. *Protein Sci.*, **8**, 897–904.