BIOINFORMATICS ORIGINAL PAPER Vol. 22 no. 22 2006, pages 2753–2760
doi:10.1093/bioinformatics/btl475

Structural bioinformatics

Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure

Darrin P. Lewis[1], Tony Jebara[1] and William Stafford Noble[2,*]

[1]Department of Computer Science, Columbia University, New York, NY, 10027 and [2]Department of Genome Sciences, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 98195, USA

Received on April 22, 2006; revised on September 1, 2006; accepted on September 3, 2006
Advance Access publication September 11, 2006
Associate Editor: Alfonso Valencia

ABSTRACT

Motivation: Drawing inferences from large, heterogeneous sets of biological data requires a theoretical framework that is capable of representing, e.g. DNA and protein sequences, protein structures, microarray expression data, various types of interaction networks, etc. Recently, a class of algorithms known as kernel methods has emerged as a powerful framework for combining diverse types of data. The support vector machine (SVM) algorithm is the most popular kernel method, due to its theoretical underpinnings and strong empirical performance on a wide variety of classification tasks. Furthermore, several recently described extensions allow the SVM to assign relative weights to various datasets, depending upon their utilities in performing a given classification task.

Results: In this work, we empirically investigate the performance of the SVM on the task of inferring gene functional annotations from a combination of protein sequence and structure data. Our results suggest that the SVM is quite robust to noise in the input datasets. Consequently, in the presence of only two types of data, an SVM trained from an unweighted combination of datasets performs as well or better than a more sophisticated algorithm that assigns weights to individual data types. Indeed, for this simple case, we can demonstrate empirically that no solution is significantly better than the naive, unweighted average of the two datasets. On the other hand, when multiple noisy datasets are included in the experiment, then the naive approach fares worse than the weighted approach. Our results suggest that for many applications, a naive unweighted sum of kernels may be sufficient.

Availability: http://noble.gs.washington.edu/proj/seqstruct

Contact: noble@gs.washington.edu

Supplementary information: Supplementary Data are available at *Bioinformatics* online.

1 INTRODUCTION

It is by now a truism to point out that biological data is being produced at a rapid rate. Less obvious, but equally daunting, is the large variety of types of biological data being produced. Traditional statistical methods that assume Gaussian distributions, or engineering methods that assume vector or matrix input do not obviously generalize to datasets comprised of variable-length strings, vectors of real numbers, trees and networks. Consequently, much recent work has focused on the development of statistical and computational methods that are capable of drawing inferences from large, heterogeneous biological datasets.

Kernel methods (Schölkopf *et al*., 1999) provide a principled means to represent and hence draw inferences from diverse types of data. A kernel method represents a collection of arbitrarily complex data objects by using a so-called kernel function that defines the similarity between any given pair of objects. In practice, this means that a collection of n objects can be sufficiently represented via an $n \times n$ matrix of pairwise kernel values. This kernel matrix, hence, provides a sort of normal form: as long as a valid kernel function can be defined on a given data type, then any such dataset can be represented as a kernel matrix. Kernel methods are algorithms that operate on kernel matrices, rather than on the raw data objects themselves.

By far the best-known kernel method is the support vector machine (SVM) algorithm (Boser *et al*., 1992; Vapnik, 1998; Cristianini and Shawe-Taylor, 2000). The SVM is a supervised classification algorithm that learns by example to discriminate among two or more given classes of data. Within computational biology, SVMs have been applied to an increasing variety of problems, including remote protein homology detection, various types of gene expression analyses, splice site and alternative splicing detection, tandem mass spectrometry analysis, etc. (Noble, 2004).

In order to apply an SVM to a heterogeneous dataset, kernels must be defined for each data type and the kernel matrices must be combined algebraically. For example, Pavlidis *et al*. used this approach to combine microarray gene expression data and phylogenetic profiles in an unweighted fashion, applying mathematical operators to focus the SVM on within-dataset correlations among features while ignoring correlations between datasets (Pavlidis *et al*., 2001, 2002). An unweighted sum of kernels has also been used successfully in the prediction of protein-protein interactions (Ben-Hur and Noble, 2005).

Recently, several research groups have proposed multiple kernel learning (MKL) methods that combine kernels within the SVM algorithm itself (Lanckriet *et al*., 2002; 2004c; Bach *et al*., 2005; Sonnenburg *et al*., 2006a; Jebara, 2004). These methods formulate a single optimization procedure that simultaneously finds the SVM classification solution as well as weights on the individual data types in the heterogeneous set. Lanckriet *et al*. (2002) formulate the problem using semidefinite programming, whereas Sonnenburg *et al*. (2006b) formulate the problem using semi-infinite linear programming. These approaches have been successful in various

*To whom correspondence should be addressed.

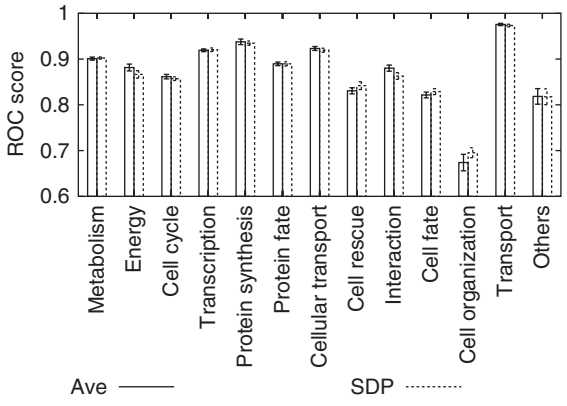

Fig. 1. Comparison of two kernel combination methods for prediction of yeast MIPS classifications. The figure shows, for thirteen one-versus-all classification tasks, the area under the ROC curve from a 5-fold cross-validated SVM experiment. Error bars represent standard error. The results labeled 'SDP' are from Lanckriet *et al*. (2004b) (Supplementary Table 7). The results labeled 'Average' were computed using an unweighted average of the same five kernels. The figure clearly shows that, for each classification task, the SDP and unweighted average approaches perform comparably.

bioinformatics tasks, including yeast protein functional classification (Lanckriet *et al*., 2004a,b), protein structure classification (Borgwardt *et al*., 2005), protein subcellular localization (Zien and Ong, 2006) and alternative splicing recognition (Sonnenburg *et al*., 2006b).

In general, MKL methods that assign weights to individual data types have some practical disadvantages. The SDP approach of Lanckriet *et al*. requires large amounts of memory. This problem was essentially solved by Bach *et al*. (2004a), but the resulting algorithm is still quite slow. For the SDP approach, the running time is $O(K^2n^3)$ for K kernels and n examples (Lanckriet *et al*., 2002). In contrast, the SVM with unweighted combination of kernels requires only $O(n^3)$time. Other MKL methods are faster (Sonnenburg *et al*., 2006c), but all such methods are slower than solving a single SVM. Furthermore, all of these algorithms are more complicated to program than a simple SVM.

The current work aims to answer the question: are MKL methods worth the additional effort, relative to using an unweighted sum-of-kernels? Furthermore, if MKL methods are useful, then we would like to be able to characterize the situations in which they should be used. One motivation for this study is given in Figure 1. The figure compares the published SDP results from (Lanckriet *et al*., 2004b) with results from an SVM trained using an unweighted average of the same kernels. Across all thirteen classes, the two methods perform nearly identically.

In this work, we focus on a particular task: predicting Gene Ontology (GO) terms using a combination of amino acid sequence and protein structural information. This task has intrinsic interest: considering the amount of effort currently being expended on inferring protein structures, it is interesting to quantify the extent to which protein structure improves upon our ability to draw inferences about proteins, relative to inferences drawn from sequence alone. Furthermore, the problem has two characteristics that are useful in the context of this study. First, by restricting ourselves to two kernels, it is possible to explore systematically the space of possible linear combinations. Second, since we know that structure is generally more informative than sequence, we can expect reasonably consistent behavior of our kernel combinations across a wide variety of classification tasks.

Our experiments show, first, that protein structure is more informative than protein sequence. The structure kernel that we employ uses MAMMOTH (Ortiz *et al*., 2002), which is a structural alignment algorithm that considers only the protein backbone, ignoring the side chains that differ among amino acids. Hence, this kernel, by its design, is fairly independent of the sequence kernel. Nonetheless, even without side chain information, the structure kernel provides better recognition performance than the sequence kernel on all 56 GO terms that we considered. These terms come from all three GO hierarchies—molecular function (MF), BP and cellular compartment (CC).

Perhaps more surprisingly, we find that, for this two kernel task, the unweighted average of kernels performs slightly better than a more sophisticated method that assigns weights to the kernels. Indeed, by systematically considering various relative weights, we are able to demonstrate that, for this particular task, no kernel-weighting scheme can perform much better than the simple unweighted sum of kernels.

On the other hand, in a follow-up experiment, we demonstrate that MKL is indeed helpful in some circumstances. Specifically, we consider the case in which additional, noisy kernels are added to the sequence and structure kernels. As we add more noise to the system, the performance of the unweighted average deteriorates. In contrast, the weighted kernel approach learns to down-weight the noise kernels and hence continues to work well.

Finally, in a separate experiment, we investigate the performance of both kernel combination methods in the presence of missing data. In practice, we have sequence information for many more proteins than structure information. Such missing data are common in genome-wide datasets and the probability that any one gene or protein will have missing data increases as we include more data types in a single classification experiment. Hence, it is interesting to ask how well an SVM performs when one of the kernels in the combination contains missing data. In general, however, SVMs do not provide a mechanism for handling missing data. We consider three simple techniques for representing missing examples in a kernel matrix. Our experiments do not definitely show that one of these three techniques is best; however, we do demonstrate that, using any of the three proposed methods for handling missing data and using either a weighted or unweighted kernel combination, the SVM performance degrades fairly gradually as the percentage of missing data in the structure kernel increases.

This empirical study aims to provide guidance to users of SVM classifiers, as well as to suggest avenues for further research. Perhaps our most important practical conclusion is that the simple, unweighted sum of kernels can provide remarkably robust classification performance. Only when used with a relatively large collection of kernels, with some data which were less relevant to the task at hand, did a weighted kernel combination method add value. Our results also suggest caution in interpreting the specific weights assigned to each data type by a weighted kernel approach, since the SVM performance does not vary dramatically as the kernel weights change.

2 ALGORITHMS

We provide here a brief, non-technical overview of the SVM, followed by descriptions of the two methods for handling heterogeneous data. More detail on SVMs and kernel methods can be found in, e.g. (Cristianini and Shawe-Taylor, 2000; Schölkopf *et al*., 1999).

2.1 Support vector machines

Applying an SVM to a classification problem consists of two phases: training and prediction. During training, the SVM takes as input a dataset in which each example is a fixed-length vector. Furthermore, each example must have an associated binary label. We use '+1' to denote the positive class and '−1' to denote the negative class. If each vector contains m values, then we say that the data resides in an m-dimensional space called the input space.

The SVM training algorithm searches for a plane (or, when $m > 3$, a hyperplane) in the input space that separates the positive from the negative examples. Learning theory suggests that, when many such hyperplanes exist, an optimal procedure selects the hyperplane that is farthest from any training example. This particular hyperplane is known as the maximum margin hyperplane. The problem of selecting, for a given dataset, the maximum margin hyperplane can be formulated and solved efficiently using quadratic programming. This optimization constitutes the SVM training phase.

Having identified this separating hyperplane, the prediction phase takes as input a second dataset of length-m vectors. The goal of this phase is to predict the associated +1/−1 label for each of the test examples. Prediction is accomplished by asking on which side of the separating hyperplane each test example falls.

This description of the SVM algorithm leaves out many details, but should be sufficient for our subsequent discussion. Most notably, we have not described how the SVM works when no separating hyperplane exists. Briefly, this situation is handled in two ways. The first solution involves introducing a so-called 'soft margin,' which allows a subset of the training data to fall on the 'wrong' side of the hyperplane; e.g. a few examples labeled '+1' might lie on the '−1' side of the hyperplane and vice versa. The second solution involves introducing a kernel function, which we describe next.

2.2 Kernel methods

Generically, a kernel function defines the similarity between a given pair of objects. Denoted $K(x,y)$, a large value indicates that x and y are similar and a small value indicates that they are dissimilar. In the context of kernel methods in machine learning, the kernel function must be symmetric and positive semidefinite. The latter means that, for all possible datasets, the matrix of all-versus-all kernel values must have non-negative, real eigenvalues.

The fundamental idea of a kernel method is simple but somewhat subtle. Say that you have a collection of n vectors, each of length m. This data can be written as an $m \times n$ matrix. Given a kernel function $K(\cdot,\cdot)$, we can compute the similarity between all pairs of vectors in the dataset. These kernel values can then be written as an $n \times n$ matrix, called the kernel matrix. For an algorithm to be a kernel method, it must be possible to show that the kernel matrix is a sufficient representation of the data. In other words, if an algorithm is a kernel method, then it should be possible to discard the original data matrix and still run the algorithm, using only the kernel matrix.

The canonical kernel function is the scalar product (a.k.a. the dot product or vector product) $K(x,y) = \sum_i x_i y_i$. Thus, a kernel method is an algorithm that can be written down in such a way that all data vectors appear within a scalar product operation. To 'kernelize' the algorithm, we then simply replace the scalar product operation with the kernel function K.

Substituting the kernel function for the scalar product operation is useful because it is mathematically equivalent to projecting the dataset into a different space. Say that the input space has m dimensions, but we use a quadratic kernel function defined as $K(x,y) = (\sum_i x_i y_i)^2$. In this case, we are implicitly working in a space of $O(m^2)$ dimensions. This higher-dimensional space is called the feature space and in this example, it contains one dimension for every pair of dimensions in the input space. This kernel can thus capture pairwise correlations between input variables. A kernelized version of the SVM algorithm finds the maximum margin hyperplane in the feature space, simply by solving the original optimization problem using a different kernel function.

2.3 Combining kernels

Kernels are useful because they often (though not always) allow the SVM to find a separating hyperplane in a dataset that was previously inseparable. Kernels may also allow us to encode prior knowledge about the data, such as the knowledge that pairwise correlations are important. In the current work, however, we are particularly interested in the kernel function as a way to encode similarities among non-vector and heterogeneous datasets.

First, we note that although the discussion thus far has focused on vector data, a kernel function can be defined for any arbitrarily complex data object. Thus, kernels have been defined for DNA and protein sequences, protein–protein and metabolic networks, phylogenetic trees, etc. (Noble, 2004) As long as our collection of data can be represented as a square kernel matrix, then any kernel method can be applied to the data. The kernel matrix is thus a sort of normal form for representing diverse types of data.

Second, the mathematics of kernels allows us to derive new kernels by combining two or more kernel functions. Many mathematical operations are closed under positive semidefiniteness. The most important such operation is addition: if K_1 and K_2 are both kernel functions, then we can prove that $K(x,y) = K_1(x,y) + K_2(x,y)$is a kernel. This operation is mathematically equivalent to concatenating the vector representations of the two data points in the feature spaces defined by K_1 and K_2. For positive coefficients, a weighted combination of kernels $(\mu_1 K_1(x,y) + \mu_2 K_2(x,y))$ also preserves the kernel property.

This mathematical formalism provides us with a straightforward way to combine heterogeneous data. Given a set of proteins represented as sequences and structures, we compute a kernel matrix K_q from the sequences and a kernel matrix K_r from the structures. The sum of these two kernel matrices is a new kernel that simultaneously represents the protein sequences and structures.

In the current work, we contrast the simple sum-of-kernels approach with a more sophisticated method that introduces weights on each kernel. Using one of various optimization methods, we can simultaneously find a separating hyperplane and find weights on each individual kernel. The weights are chosen so as to maximize the margin between the two classes. This approach corresponds to learning, e.g. that the sequence kernel is not as informative as the structure kernel for a given classification task. Geometrically, the

kernel weight rescales each dimension of a given kernel. Thus, in the feature space corresponding to $K = K_1 + 2K_2$, each of the dimensions from K_2 is scaled by a factor of 2.

3 METHODS

3.1 GO classes

The GO (Gene Ontology Consortium, 2000) is a diverse catalog of gene (protein) annotations, which includes information about protein function and localization. We define a GO term prediction benchmark by starting with a set of 8363 PDB structures, pruned so that no two sequences share >50% sequence identity (Li *et al*., 2001). Among these proteins, 5325 have GO annotations, downloaded from www.geneontology.org. 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' and ignore them during both training and testing. Finally, all proteins that are not on the path from *T* to the root are labeled as 'negative'.

After this labeling procedure, we eliminated all GO terms with fewer than 100 '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 56 GO terms: 27 MF terms, 22 BP terms and 7 CC terms. All 56 terms are listed in the online supplement.

For each GO term, the number of negative examples far exceeds the number of positive examples. For efficiency, we randomly select a subset of the negative examples so that the ratio of positives to negatives is one-to-one.

3.2 Kernels

To represent protein sequences, we use the mismatch kernel (Leslie *et al*., 2003). This kernel generalizes upon the simpler, spectrum kernel (Leslie *et al*., 2002), which represents a string as a vector of counts of all possible substrings of a fixed length *k*. The mismatch kernel generalizes upon the spectrum kernel by incrementing, for each observed *k*-length string (*k*-mer), the corresponding count as well as the counts of all *k*-mers that differ from the observed *k*-mer by at most *M* mismatches. In this work, we use a mismatch spectrum kernel with $K = 4$ and $M = 1$. The final mismatch spectrum vector has $20^4 = 160\,000$ bins. The mismatch spectrum captures sequence similarity and has been shown to provide good performance in classifying SCOP superfamilies (Leslie *et al*., 2003).

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 (i.e. the E-value) returned by MAMMOTH cannot be used as a kernel function directly, because the score is not positive semidefinite. We therefore employ the so-called 'empirical kernel map' (Tsuda, 1999) to convert this score to a kernel: for a given dataset of structures $\mathrm{X} = x_1, \ldots x_n$, a structure x_i is represented as an *n*-dimensional vector, in which the *j*-th 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, such as Smith-Waterman (Smith and Waterman, 1981). 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.

Prior to summation, each kernel is first centered around the origin in the feature space, and each data point is projected onto the unit sphere using $\hat{K}(x,y) = K(x,y)/\sqrt{K(x,x)K(y,y)}$.

3.3 Experimental framework

All SVM experiments were performed using our own code, which is a combination of C++ and Matlab. To compute weighted kernel combinations we use semidefinite programming, as described in Lanckriet *et al*. (2002). SVMs were tested using 5-fold cross-validation, repeated three times (3×5cv). We use a fixed value of the SVM regularization parameter C = 10. We measure classification performance using the area under the receiver operating characteristic (ROC) curve, which plots the rate of true positives as a function of the rate of false positives for varying classification thresholds. We report means and standard deviations of the area under the ROC curve (the ROC score) with respect to the fifteen 3×5cv splits.

4 RESULTS

We present our results as a series of four experiments. The first experiment is a direct comparison of the unweighted and weighted kernel combination methods across all 56 GO terms in our benchmark. The results show that the structure kernel consistently outperforms the sequence kernel and that the unweighted combination generally performs better than the weighted combination. In the second experiment, we systematically vary the relative kernel weight on a subset of 10 GO terms and we show that, for this task, the unweighted sum-of-kernels performs nearly optimally. The third experiment introduces artificial noise into the dataset. In this scenario, the weighted kernel approach is useful and performs better than the unweighted approach as the amount of noise increases. Finally, in a fourth experiment, we demonstrate the robustness of both the kernel combination methods in the presence of missing data.

4.1 Experiment 1: comparison of kernel combination methods

In this experiment, we performed cross-validated testing of four types of SVMs, using sequence alone, structure alone, an unweighted combination of kernels and a weighted combination of kernels. Table 1 shows a subset of these results. The complete set of results are available in the online supplement. Not surprisingly, the MAMMOTH kernel frequently provides better classification performance: in 55 out of 56 classes, the difference between the mean structure ROC score and the mean sequence ROC score is greater than the sum of the two corresponding standard errors.

An alternate representation of these results, for all 56 terms, is shown in Figure 2A. Qualitatively, the figure shows that the sequence kernel performs far worse than any method that uses the structure kernel. Furthermore, we computed a Wilcoxon signed-rank test between all four pairs of methods. The results yield the following best-to-worst ranking of methods: unweighted sum of kernels, structure kernel alone, weighted sum-of-kernels and sequence kernel alone. In this ranking, the largest (i.e. least significant) *P*-value is 0.007 between the structure kernel alone and the weighted sum-of-kernels. Thus, it appears that combining sequence and structure can be helpful, but only when using the unweighted sum of kernels.

Table 1. Predicting GO terms from sequence or from structure. Each row in the table lists a GO term and description, the ontology from which it comes (MF = molecular function, CC = cellular compartment and BP = biological process), the number of positive examples associated with the term and the mean and standard error ROC scores for 3 × 5cv SVM training using (**1**) the structure kernel, (**2**) the sequence kernel, (**3**) the unweighted sum of both the kernels and (**4**) the weighted sum of the kernel. In the table, terms are sorted by the difference in ROC score between 'Structure' and 'Sequence.' From the entire set of 56 terms that we considered, the table lists only the top 10 and the bottom 5

GO term	Description	Ont	#	Structure	Sequence	Average	SDP
GO:0008168	Methyltransferase activity	MF	108	0.941 ± 0.014	0.709 ± 0.020	0.937 ± 0.016	0.938 ± 0.015
GO:0005506	Iron ion binding	MF	129	0.934 ± 0.008	0.747 ± 0.015	0.927 ± 0.012	0.927 ± 0.012
GO:0006260	DNA replication	BP	109	0.885 ± 0.014	0.707 ± 0.020	0.878 ± 0.016	0.870 ± 0.015
GO:0048037	Cofactor binding	MF	118	0.916 ± 0.015	0.738 ± 0.025	0.911 ± 0.016	0.909 ± 0.016
GO:0046483	Heterocycle metabolism	BP	128	0.949 ± 0.007	0.787 ± 0.011	0.937 ± 0.008	0.940 ± 0.008
GO:0044255	Cellular lipid metabolism	BP	101	0.891 ± 0.012	0.732 ± 0.012	0.874 ± 0.015	0.864 ± 0.013
GO:0016853	Isomerase activity	MF	124	0.855 ± 0.014	0.706 ± 0.029	0.837 ± 0.017	0.810 ± 0.019
GO:0044262	Cellular carbohydrate metabolism	BP	209	0.912 ± 0.007	0.764 ± 0.018	0.908 ± 0.006	0.897 ± 0.006
GO:0009117	Nucleotide metabolism	BP	124	0.892 ± 0.015	0.748 ± 0.016	0.890 ± 0.012	0.880 ± 0.012
GO:0016829	Lyase activity	MF	201	0.935 ± 0.006	0.791 ± 0.013	0.931 ± 0.008	0.926 ± 0.007
GO:0006732	Coenzyme metabolism	BP	119	0.823 ± 0.011	0.781 ± 0.013	0.845 ± 0.011	0.828 ± 0.013
GO:0007242	Intracellular signaling cascade	BP	140	0.898 ± 0.011	0.859 ± 0.014	0.903 ± 0.010	0.900 ± 0.011
GO:0005525	GTP binding	MF	104	0.923 ± 0.008	0.884 ± 0.015	0.931 ± 0.009	0.931 ± 0.009
GO:0004252	Serine-type endopeptidase activity	MF	140	0.937 ± 0.011	0.907 ± 0.012	0.932 ± 0.012	0.931 ± 0.012
GO:0005198	Structural molecule activity	MF	179	0.809 ± 0.010	0.795 ± 0.014	0.828 ± 0.010	0.824 ± 0.011

4.2 Experiment 2: varying relative kernel weight

The previous result—that the weighted sum-of-kernels performs worse than the unweighted sum—is surprising, not least because considerable effort has been expended by various research groups to develop the optimization technology to solve this type of MKL problem. Therefore, we selected a subset of the terms from our benchmark and subjected them to further investigation. Specifically, we selected the 10 terms for which the difference in ROC score between the structure-only and sequence-only SVMs is largest. For each of these terms, we systematically varied the relative kernel weights and performed cross-validated testing of the resulting SVM. The results are shown in Figure 2B.

The most striking aspect of Figure 2B is the qualitative similarity of all ten series in the figure. For each GO term, the performance of the SVM stays roughly the same for all kernel combinations that assign larger weight to the structure kernel. When more weight is assigned to the sequence kernel, the performance degrades gradually, with a rapid degradation only when the structure kernel receives very small weight. This result shows that the SVM is quite robust in the presence of variation in the relative scales of the two datasets.

A second observation is the placement of the green dots, indicating the choice of kernel weights made by the SDP optimization. In nearly every case, the SDP erroneously gives more weight to the sequence kernel. This explains why the weighted kernel combination fares poorly on this benchmark. Apparently, for these tasks, the sequence kernel yields an embedding with a larger margin than does the structure kernel, even though the latter provides better generalization performance. The apparent conflict with Experiment 1—in that case the two kernel combination methods performed equally well, whereas in this case, SDP does nearly uniformly worse—arises because the second experiment focuses on the 10 classes in which the difference between the two kernels is largest.

Finally, it is interesting to consider whether the unweighted sum-of-kernels is optimal. For many of the GO terms in Figure 2B, the highest ROC score is not achieved at a log ratio of 0. However, in every case, the difference between the unweighted sum ROC score and the best possible ROC score is quite small, in the order of 0.01. Furthermore, for any fixed log ratio, there are some terms that perform worse than the unweighted average and some that perform better. Thus, Figure 2B suggests that although an optimal learning procedure might be able to find better kernel weights than the unweighted average, this hypothetical method (1) like the SDP approach, would have to take into consideration not just the kernel matrices but also the GO term labels and (2) would still perform only slightly better than the unweighted sum-of-kernels.

4.3 Experiment 3: multiple noisy kernels

The results from our first two experiments beg the question: why bother with the computational overhead of weighted kernel combinations when the unweighted sum of kernels performs better? In the third experiment, we demonstrate via simulation a situation in which the weighted sum is necessary.

To carry out this experiment, we created a collection of 'noise' kernels. These are simply copies of the structure kernel, with permuted rows and corresponding columns. We then measured how the performance of our two kernel combination methods changes as the number of noise kernels increases.

The results are shown in Figure 2C. In the figure, green crosses correspond to classifiers trained with no noise kernels. These points were thus generated in our first experiment, described above, using one sequence and one structure kernel. Most of the points fall below the line $y = x$, indicating clearly that the unweighted kernel combination performs better than the weighted sum-of-kernels. However, this result changes as soon as we introduce a single noise kernel. In this case, the weighted sum performs better than

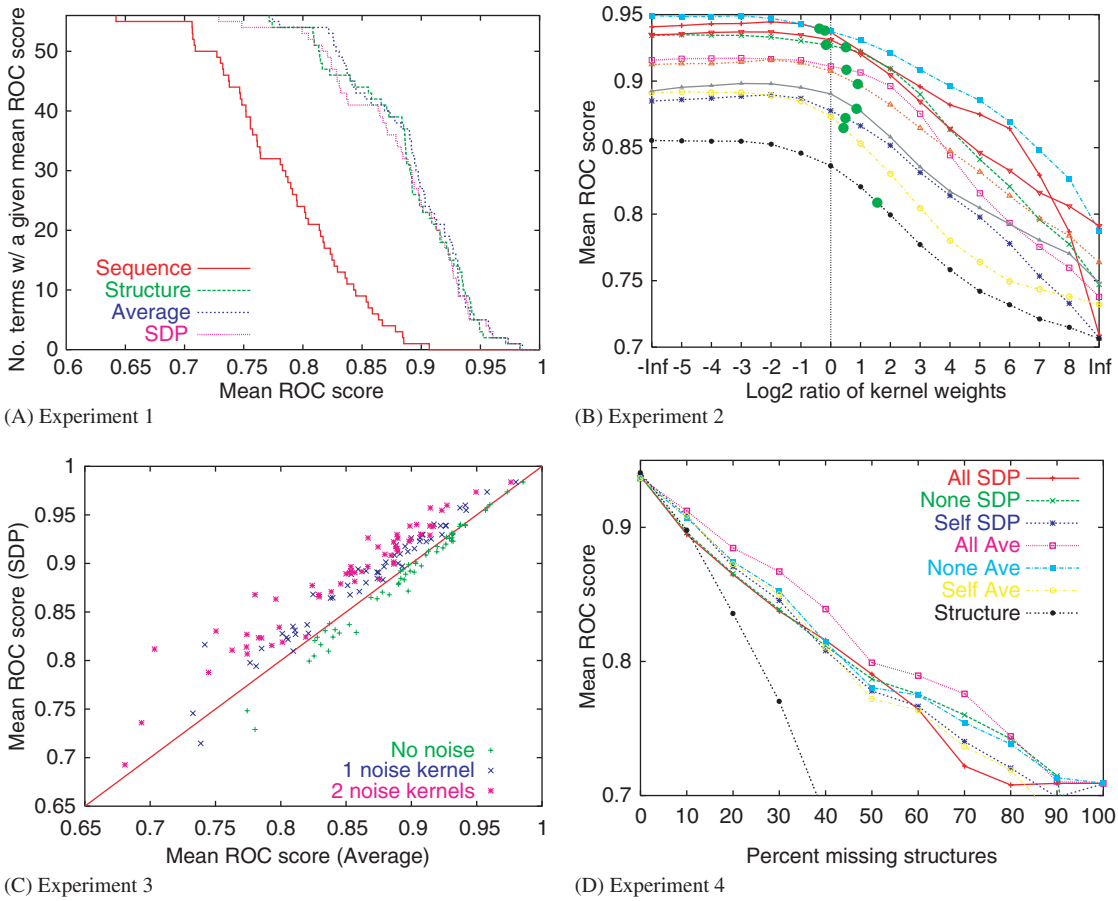

Fig. 2. Summary of four experiments combining sequence and structure kernels. (**A**) Combining kernels: cumulative comparison across 56 GO terms. The figure plots the number of GO terms (*y*-axis) for which a given SVM classifier achieves a specified mean ROC score (*x*-axis). Each series corresponds to an SVM that uses sequence alone, structure alone, or combinations of kernels using an unweighted average or using SDP. (**B**) Varying relative kernel weight. The figure plots mean ROC score as a function of the $\log_2(\mu_q/\mu_r)$, where μ_q and μ_r are the weights assigned to the sequence and structure kernels, respectively. Each series corresponds to one of the GO terms in Table 1. On each series, the large green circle indicates the log ratio of the weights selected by SDP. (**C**) Learning in the presence of noisy kernels. The figure plots, for each GO term, the ROC score achieved by the SDP SVM as a function of the ROC score achieved by the unweighted sum of kernels. Different point types correspond to training using zero, one or two noise kernels, as described in the text. (**D**) Combining kernels with missing data. The figure plots, for a single GO term (GO:00008168—methyltransferase activity), the mean ROC score as a function of the percentage of missing data in the structure kernel. The first six series correspond to the Average and SDP methods, with missing data affinity coded as 'None', 'Self' or 'All'. The final series is from an SVM trained from the structure kernel alone.

the unweighted sum for 55 out of the 56 terms. The effect becomes even more pronounced in the presence of two noise kernels. A Wilcoxon signed-rank test supports these conclusions with extremely small *P*-values.

4.4 Experiment 4: kernels with missing examples

Finally, we consider a variant of our experimental design, in which we focus on the problem of missing data. In particular, we are interested in the extent to which we can combine incomplete structural information with complete sequence information. Thus, we simulate randomly deleting varying percentages of the examples from the structure kernel matrix and measure the cross-validated ROC score of the resulting classifier. A pseudocode description of the experimental design is given in the online supplement.

Because SVMs are not designed to handle missing data, it is not obvious a priori how to represent missing examples in the kernel matrix. Therefore we consider three alternative methods for filling in missing kernel entries, all of which preserve positive semidefiniteness (see the Supplementary Data). The first strategy, 'None,' simply replaces the row and column corresponding to a missing entry with all zeroes. This strategy allows the sequence kernel to determine the weight assigned by the SVM to this example, ignoring the structure kernel entirely. The second strategy ('Self') makes each missing example similar only to itself by placing a one on the diagonal of the (normalized) kernel matrix and zeroes elsewhere. Finally, the 'All' strategy makes each missing example similar to every other missing example, but different from all non-missing examples. This is accomplished by placing ones in the kernel matrix between all pairs of missing examples and zeros between missing/non-missing pairs. The effect is to place all missing examples in a single orthogonal dimension in feature space. Effectively, all missing examples are co-located at a single point, infinitely distant from the other data.

Figure 2D shows the results of this experiment for a single GO term. Similar plots for the first ten terms in Table 1 are shown in the online supplement. Strikingly, none of the series deteriorates dramatically as we introduce missing data. Indeed, most methods perform better than the sequence kernel alone even when the structure kernel consists of 50% missing entries. Furthermore, for each of the three strategies for handling missing data, we observed on average a decrease in the weight assigned by SDP as the amount of missing data increases.

Trends among the six methods that we considered—two kernel combination methods and three methods for replacing missing values—are not obvious from the figures. A Wilcoxon signed-rank comparison of the data (see Supplementary Data) yields the same ranking of methods at 10 or 20% missing data: all three unweighted sum methods perform better than all three weighted sum methods, and the best strategy for handling missing data depends upon the kernel combination method. Using an unweighted combination, the best-to-worst ranking is all–none–self. Conversely, using a weighted combination, the corresponding ranking is self–none–all.

In short, this experiment does show convincingly that an SVM can make accurate predictions in the presence of missing data; however, the results are inconclusive with respect to the best method for representing missing examples in the kernel matrices.

5 DISCUSSION

The primary conclusion from this empirical study is that using a weighted sum of kernels in an SVM classifier does not always improve upon the simpler, unweighted sum approach. In particular, we have shown that, for a combination of sequence and structure kernels in the prediction of GO terms, the weighted sum method frequently selects a solution that is worse than the unweighted sum. On the other hand, the weighted sum does appear to improve the SVM's robustness in the presence of noisy or irrelevant kernels. From a practitioner's point of view, these results suggest the value of evaluating individual kernels with respect to any given classification task prior to applying a kernel combination method. In other words, simply collecting a large variety of kernels and applying the resulting combination of kernels to diverse classification tasks is not likely to be as successful as a more directed approach, in which prior knowledge of a particular kernel's relevance guides its inclusion in the training set.

Our results also suggest that the kernel weights assigned by an MKL method may be difficult to interpret. A priori, it is clear that a low weight may be assigned due either to noise in the kernel or redundancy with another kernel in the collection. However, for many of the classification tasks that we considered, a relatively broad range of relative kernel weights often yielded quite similar classification performance. Furthermore, in this particular case, the size of the margin does not correlate well with optimal generalization performance.

With respect to protein classification, it is perhaps not surprising that structure is more informative than sequence. However, our results with respect to missing data suggest that, even when some structure data are not present, an SVM combination of sequence and structure might be valuable.

The lower performance of the weighted kernel combination might be due to overfitting and insufficient training data. Thus, larger datasets might benefit from weighted combination when there is sufficient data to reliably estimate kernel weights. Alternatively, we might group multiple tasks and datasets to more reliably estimate a single setting of the weights on kernels (Jebara, 2004).

Any empirical study necessarily leaves some questions unanswered: one could imagine a variety of modifications, extensions or additional experiments to add to the four we described above. These include, for example, investigating different types or a larger number of kernels, modifying the SVM regularization parameter, systematically adding noise to one or more kernels, etc. While some of these experiments would likely be more informative than others, we believe that our primary message—that combining kernels in a weighted fashion is not always beneficial—would remain unchanged. Further experiments would more precisely define the situations in which weighting kernels is beneficial.

We did, inadvertently, perform one additional experiment that was not reported above, and we believe that the result is instructive. In setting up the experiment comparing kernel combination methods across 56 GO terms, we first observed that the SDP method almost uniformly gave a large weight to the sequence kernel and a small weight to the structure kernel. Investigation of these results showed that the sequence kernel margin was considerably larger simply because we had normalized the kernels (i.e. projected all of the data onto the unit sphere in the feature space) without first centering the data around the origin. In both cases, the data were far from the origin and so projection placed all of the data points onto a very small area on the unit sphere. Centering before normalization, as recommended by Lanckriet *et al*. (2002), leads to much better conditioned matrices. This result illustrates that the weighted kernel method depends upon characteristics of the various kernels in the combination. One avenue for future work would identify characteristics of 'good' kernels and propose methods for creating such kernels.

A second area for future work is the development of alternate kernel combination methods. For example, given the relatively robust performance of SVMs with respect to gradations in relative kernel weight, we believe that a combinatorial method which finds binary kernel weights might be very successful. Unfortunately, it is not obvious how to perform efficient optimization on discrete kernel weights.

ACKNOWLEDGEMENTS

This project was supported by National Science Foundation grants CCR-0312690, IIS-0347499 and IIS-0093302 and by National Institutes of Health grant R33 HG003070.

Conflict of Interest: none declared.

REFERENCES

Bach,F.R. *et al*. (2004a) Multiple kernel learning, conic duality and the smo algorithm. In *Proceedings 21st International Conference on Machine Learning (ICML),* Banff, Canada, pp. 6–13.

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

Ben-Hur,A. and Noble,W.S. (2005) Kernel methods for predicting protein–protein interactions. *Bioinformatics*, **21** (suppl. 1), i38–i46.

Borgwardt,K.M. *et al*. (2005) Protein function prediction via graph kernels. *Bioinformatics*, **210** (Suppl. 1), i47–i56.

Boser,B.E., Guyon,I.M. and Vapnik,V.N. (1992) A training algorithm for optimal margin classifiers. In Haussler,D. (ed.), *In proceedings of the 5th Annual ACM Workshop on COLT*. ACM Press, Pittsburgh, PA, pp. 144–152.

Cristianini,N. and Shawe-Taylor,J. (2000) *An Introduction to Support Vector Machines*. 1st edn. Cambridge UP, Cambridge, UK.

Gene Ontology Consortium (2000) Gene ontology: tool for the unification of biology. *Nat. Genet*., **250,** 25–29.

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

Jebara,T. (2004) Multi-task feature and kernel selection for SVMs. In *Proceedings of the International Conference on Machine Learning*. Banff, Canada, pp. 55–63.

Lanckriet,G.R.G. *et al*. (2002) Learning the kernel matrix with semi-definite programming. In Sammut,C. and Hoffman,A. (eds), In *Proceedings of the 19th International Conference on Machine Learning,* Morgan Kauffman, Sydney, Australia, 323–330.

Lanckriet,G.R.G. *et al*. (2004a) A statistical framework for genomic data fusion. *Bioinformatics*, **200**, 2626–2635.

Lanckriet,G.R.G. *et al*. (2004b) Kernel-based data fusion and its application to protein function prediction in yeast. In Altman,R.B., Dunker,A.K., Hunter,L., Jung,T.A. and Klein,T.E. (eds), In *Proceedings of the Pacific Symposium on Biocomputing*. World Scientific, Hawaii, USA, pp. 300–311.

Gert,R.G. *et al*. (2004c) Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res*., **5**, 27–72.

Leslie,C. Eskin,E. and Noble,W.S. (2002) The spectrum kernel: a string kernel for SVM protein classification. In Altman,R.B., Dunker,A.K., Hunter,L., Lauderdale,K. and Klein,T.E. (eds), *In Proceedings of the Pacific Symposium on Biocomputing*. World Scientific, New Jersey, pp. 564–575.

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

Li,W. *et al*. (2001) Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, **170**, 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, 225–232.

Noble,W.S. (2004) Support vector machine applications in computational biology. In Schoelkopf,B., Tsuda,K. and Vert,J.-P. (eds), *Kernel methods in computational biology*. MIT Press, Cambridge, MA, pp. 71–92.

Ong,C.S. *et al*. (2005) Learning the kernel with hyperkernels. *J. Mach. Learn. Res*., **6**, 1043–1071.

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., Westen,P., Cai,J. and Grundy,W.N. (2001) Gene functional classification from heterogeneous data. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology,* ACM Press, NY, USA, 242–248.

Pavlidis,P. (2002) Learning gene functional classifications from multiple data types. *J. Comput. Biol*., **90**, 401–411.

Schölkopf,B., Burges,C.J.C. and Smola,A.J. (eds) (1999) *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA.

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.

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

Sonnenburg,S., Rätsch,G. and Schafer,C. (2006a) A general and efficient multiple kernel learning algorithm. In Weiss,Y., Schoölkopf,B. and Platt,J. (eds), *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA.

Sonnenburg,S. *et al*. (2006b) Learning interpretable SVMs for biological sequence classification. *BMC Bioinformatics*, **70** (Suppl. 1), S1–S9.

Sonnenburg,S. *et al*. (2006c) Large scale multiple kernel learning. *J. Mach. Learn. Res*., To appear.

Tsuda,K. (1999) Support vector classification with asymmetric kernel function. In Verleysen,M. (ed.), In *Proceedings ESANN*. Bruges, Belgium, pp. 83–188.

Vapnik,V.N. (1998) *Statistical Learning Theory. Adaptive and Learning Systems for Signal Processing, Communications, and Control*. John Wiley & Sons, New York, USA.

Zien,A. and Ong,C.S. (2006) An automated combination of sequence motif kernels for predicting protein subcellular localization. *Technical Report TR-146*. Max-Planck Institute for Biological Cybernetics, Tübingen, Germany.