Analysis of Kernel Based Protein Classification Strategies Using Pairwise Sequence Alignment Measures

Dino Franklin[1], Somdutta Dhir[2], and Sándor Pongor[2]

[1] Federal University of Goiás, Campus de Catalão, 75705-220 Catalão GO, Brazil
[2] International Centre for Genetic Engineering and Biotechnology, Padriciano 99, 34012 Trieste, Italy

Abstract. We evaluated methods of protein classification that use kernels built from BLAST output parameters. Protein sequences were represented as vectors of parameters (e.g. similarity scores) determined with respect to a reference set, and used in Support Vector Machines (SVM) as well as in simple nearest neighbor (1NN) classification. We found, using ROC analysis, that aggregate representations that use aggregate similarities with respect to a few object classes, were as accurate as the full vectorial representations, and that a jury of 6 1NN-based aggregate classifiers performed as well as the best SVM classifiers, while they required much less computational time.

1 Introduction

Automated classification of protein sequences are important issues in computational biology as human annotation is costly and time consuming. Automated protein function assignment methods are usually based on sequence similarity. A classical method of assessing the similarity of two proteins is sequence comparison via Dynamic Programming, but due to the large size of current databases, fast heuristic algorithms, such as BLAST [1] became the method of choice.

Liao and Noble [2] developed an efficient kernel method for detecting remote protein homology based on pairwise sequence comparison [2]. Every protein was represented as a vector of sequence similarity scores calculated with respect to a reference set of sequences (we term this method *full approach*). One can then train an SVM with these vectors, and use it to classify unknown proteins, represented as vectors built from similarity scores calculated with respect to the same reference dataset. Since this approach maps the proteins to high dimensional vectors, it has a high discriminative power at the expense of a relatively high computational cost.

In the so-called *aggregate approach* developed by the ICGEB Group for the SBASE library of protein domain sequences [3], the reference set is first aggregated into groups and a protein is then represented as a vector of similarities with respect to a set of groups. Because of the aggregation, the vectors are much smaller than the vectors used by the *full approach*). The success of this approach depends on how the sequences are grouped, and how the similarities with respect to the groups are defined; however, it potentially allows a substantial reduction in the computation time. This strategy was employed to a combination of various BLAST output parameters (*raw-score, hsp, etc.*) that were used to train SVM classifiers used in the SBASE search engine [3].

In this paper we compare the full vector representation with various levels of aggregate representations. We show that the performance of the full approach can be reached using aggregate strategies, and further improvements are possible employing a jury of Nearest Neighbor (1NN) classifiers that use different BLAST output parameters.

2 Methods

2.1 BLAST Output

BLAST (acronym for Basic Local Alignment Search Tool) is the most widely used alignment algorithm for protein sequence comparison [1]. The fundamental unit of BLAST, the *hsp* (High-scoring Segment Pair), is a pair of sequence segments (taken from the two aligned sequences) whose local alignment score exceeds a cutoff score. *msp* (Maximal-scoring Segment Pair) is the *hsp* with the highest score in a pairwise comparison of two sequences, given a set of *hsp*s and a scoring system. For each *hsp* BLAST also outputs a raw score (*rs*), a bit-score (*bs*) which is independent of the scoring-matrix; and an *e-value* which estimates the number of *hsps* having a *score* S (or higher) to have occurred by chance. In this paper, *msp* and *hsp* are interchangeable because we only consider the best *hsp.*

2.2 Datasets

LN is a protein sequence dataset especially designed for testing classifiers [2]. This dataset of 4,352 protein sequences was obtained from SCOP v.1.53 after removing similar sequences using an e-value threshold of 10^{-25} . The 54 classification tasks were obtained by selecting families with at least 5 members for positive test set and superfamilies with at least 10 members outside the family for positive training set.

SCOP95 is a protein dataset obtained from SCOP database v.1.69 by considering proteins with sequence identity lower than 95%. The 11,944 entries are distributed in 246 classification tasks.

SCOP40mini is a subset of SCOP95, but the sequences are 40% similar at most. Using a strategy similar to [2], 1,357 protein sequences are divided in 55 tasks.

CATH95 is a protein dataset obtained from proteins in CATH v.3.0.0 with sequence identity below 95%. The tasks were derived selecting similarity groups (S) with at least 5 members, and groups with at least 10 members outside S but within the same homology group (H) for the positive set. CATH95 comprehends 165 classification tasks and 11,373 protein sequences.

SCOP40mini and LN are non redundant datasets. The benchmark datasets used in this work are available in:

1) `http://www.cs.columbia.edu/compbio/svm-pairwise` [2];
2) `http://net.icgeb.org/benchmark/` [7]

2.3 Classification Algorithms and Performance Assessment

In this study, we have performed classification simulations using Nearest Neighbor (1NN) and Support Vector Machines (SVM) [4]. 1NN assigns to an object the class of the closest training example in the feature space. SVM is a classical algorithm that computes an optimal hyperplane which separates the positive from the negative training sets. SVM classification is based on the query position regarding this hyperplane. In this work we used the SVM algorithm as implemented in Ref.[9].

Classification performance was characterized using the Receiver Operating Characteristic (ROC curve) using the ROCR package [6]. ROC curve is a plot of true positive rate vs. false positive rate through every prediction threshold value. The Area Under ROC Curve (AUC) is 0.5 for random prediction, and it is 1 when all positive examples have higher predicted values than the negative ones. The Average Area under the ROC curve is an average of AUC values calculated from a number of classification tasks defined on a larger database [2,5,6].

3 Results

3.1 Comparing Various Aggregation Strategies

Suppose a set of proteins is divided into classes, folds, superfamilies, and families as in the diagram presented in Figure 1. A protein can then be represented either as a vector of similarity scores, calculated with respect to all proteins in the set, or as a smaller vector, composed of similarity scores calculated with respect to protein families. Using vectors calculated with respect to families, superfamilies, folds and classes, we obtain smaller and smaller vectors. These vectors can then be used in classifiers using a classical machine learning arrangement as described in [2,7]. In Figure 1, the black squares represent the remote family to be detected.

Figure 2 shows the classifier performance using full pairwise representation for the SCOP40mini dataset. The results show that the classifier performance is usually better if one uses *rs* as kernel, followed by *e-value* and *hsp*. The tests have also shown that classification performance is task and dataset dependent.

The Unary Aggregation is the simplest possible representation which is based exclusively on the positive training set. It does not consider the negative examples when modeling the protein. In a previous paper , we have shown that using the Binary Aggregation, based on separately aggregated positive and negative training sets, considerably improves the performance of a classifier [5]. The most conspicuous result in Figure 4 is the excellent performance of the binary strategy based on the maxima of the positive and negative training sets. For most of the tests, the binary strategy, which is the Ref.[3] approach, has yielded the best AAUCs, sometimes even better than the full pairwise approach.

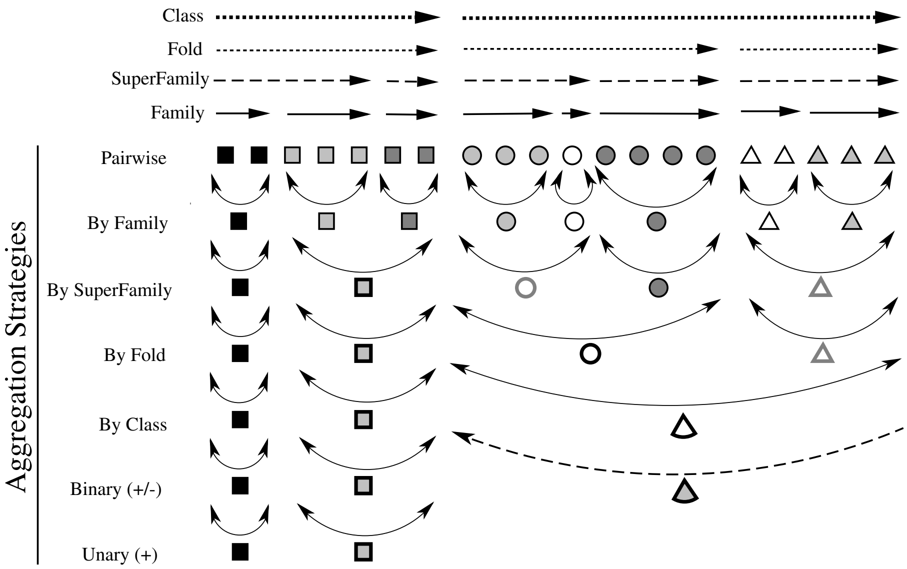

Fig. 1. Diagram of possible aggregations of the protein sequence similarities representation based on SCOP for superfamily classification. On the top, the arrows show the classes, folds, superfamilies, and families division. The proteins similarities are represented by geometric figures related to their folds, and shaded according to their families. The aggregations (maximum or average) are represented by arrowed arcs. On the left side, the aggregation criteria are shown. See text for further details.

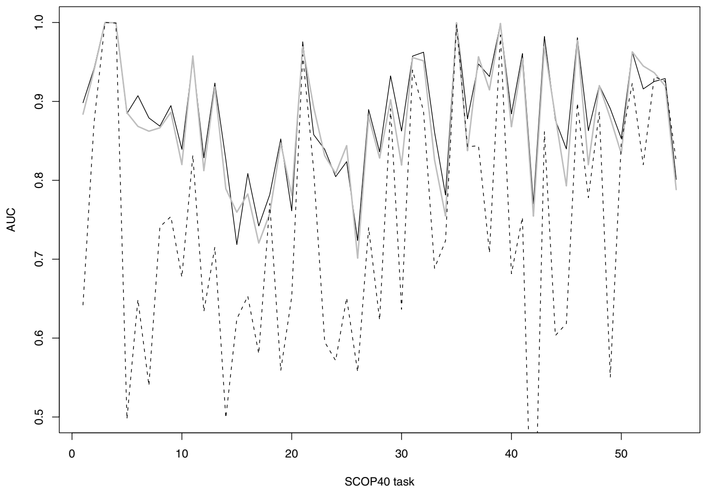

Fig. 2. AUC for every task of the SCOP40mini dataset for three BLAST output parameters: *rs* (solid line), *hsp* (gray line) and *e-value* (dashed line). The tests were done using full pairwise representation [1] and an SVM with linear kernel. The parameters were optimized by simple grid search.

3.2 Correlation and Influence on Classification Performance of BLAST Output Parameters

The utility of various BLAST output parameters (*raw-score, bit-score, e-value, hsp-length* and *bit-score*) as well as different aggregation strategies (maximum, average) for building kernel functions was estimated via correlation analysis and classification tests on several benchmark datasets, using the 1NN algorithm.

The results in Table 1 show that *bit-score* and *e-value* are the BLAST output parameters that apparently contain most of the information. The classification performance is generally better when aggregates are based on the maximal similarity instead of average similarity. In most of tests, *raw-score* and *bit-score* have yielded the best AUCs. This is in fact expected as they are linearly related.

The tendency of AAUCs is in good agreement with the correlation analysis. Table 2 shows the average correlation between BLAST similarity measures for the LN dataset. It is interesting to note that some features are quite correlated, e.g. for bs_{max} and rs_{max}, the correlation is 0.99217. A priori, AAUCs results suggest that the higher the mutual information, the more similar are their classification performances. This analysis can be particularly relevant for feature selection in a multiple input approach.

Table 3 shows the classification performances of 1NN on different kernel spaces obtained from combined BLAST output parameters, e.g. $rs/len(query)$, raw-score divided by the length of the query. In some cases, a feature that has a high AUC in maximum aggregation, has a low AUC using average aggregation and vice-versa. This is because the positive and negative training sets are mapped to 2 scalars.

Finally, Figure 3 shows the performance (AAUCs) of SVM classifiers with a linear kernel using several single and combined BLAST output parameters as kernels and different representation strategies for the SCOP40mini dataset.

Table 1. Average Area Under Curve (AAUC) for 1NN classification based on Single BLAST output parameters

Aggregation Strategy: Maximum

			BLAST output	
Dataset	*hsp*	*raw-score* (*rs*)	*bit-score* (*bs*)	*e-value* (*ev*)
LN (54)*	0.7840	0.7992	0.7992	0.7917
SCOP95 (250)	0.6971	0.6986	0.6986	0.6974
CATH95 (165)	0.9786	0.9789	0.9789	0.9787

Aggregation Strategy: Average

			BLAST output	
Dataset	*hsp*	*raw-score* (*rs*)	*bit-score* (*bs*)	*e-value* (*ev*)
LN (54)	0.7640	0.7903	0.7902	0.7325
SCOP95 (250)	0.6936	0.6967	0.6967	0.6933
CATH95 (165)	0.9770	0.9788	0.9788	0.9755

* *Values in parenthesis mean quantity of tasks in the dataset.*

Table 2. Average of Correlation between BLAST output parameters for the LN dataset

	hsp_{max}	rs_{max}	bs_{max}	ev_{max}	hsp_{avg}	rs_{avg}	bs_{avg}	$Avg(log(ev))$
hsp_{max}	1							
rs_{max}	0.7253	1						
bs_{max}	0.702	0.9945	1					
ev_{max}	-0.6105	-0.8868	-0.8952	1				
hsp_{avg}	0.8758	0.7864	0.7808	-0.7029	1			
rs_{avg}	0.6214	0.9371	0.9632	-0.8981	0.7532	1		
bs_{avg}	0.6037	0.9235	0.9554	-0.8876	0.7370	0.9975	1	
$Avg(log(ev))$	-0.5421	-0.8696	-0.9146	0.8461	-0.6800	-0.9703	-0.9849	1

Table 3. Average Area Under Curve (AAUC) using 1NN and different kernel functions from combined BLAST output parameters for maximum or average aggregates

Combined BLAST output	**LN (54)**	**SCOP95 (250)**	**CATH95 (165)**
NSD *	0.7774	0.6975	0.9761
$Max(hsp/len(query))$	0.7086	0.6934	0.9784
$Max(hsp/len(subject))$	0.7828	0.6976	0.9788
$Avg(hsp/len(query))$	0.6900	0.6892	0.9770
$Avg(hsp/len(subject))$	0.7802	0.6946	0.9773
$Max(rs/len(query))$	0.5909	0.6872	0.9768
$Max(rs/len(subject))$	0.7744	0.6976	0.9786
$Max(rs/rs(query))$	0.5935	0.6873	0.9769
$Max(rs/rs(subject))$	0.7738	0.6976	0.9786
$Avg(rs/len(query))$	0.8135	0.6850	0.9736
$Avg(rs/len(subject))$	0.7942	0.6949	0.9782
$Avg(rs/rs(query))$	0.8187	0.6850	0.9737

** NSD means number of neighbors. Max and Avg mean maximal and average aggregation. len() is for the sequence length of subject or query.*

Based on the results presented in Table 3 and Figure 3, we selected the best performing features for building combined classifiers. Among these, (NSD) defined as the number of significant similarities (similarity score above a certain threshold) is particularly interesting as is is not well correlated with the other parameters $Max(hsp/len(subject))$, $Avg(rs/len(query))$, etc.

Based on these results, we have trained SVM classifier that inputs 6 different BLAST output parameters as features. Although an optimal feature set is dataset dependent, the results in Figure 5 show that using a set of features can improve the classification performance as compared with an individual feature, but not significantly. Especially, the performance improvement is only a bit better than using rs_{max} alone.

3.3 Complexity Analysis

The total time for running a protein classification using a machine learning (ML) algorithm consists of training time and test time. The training time has three

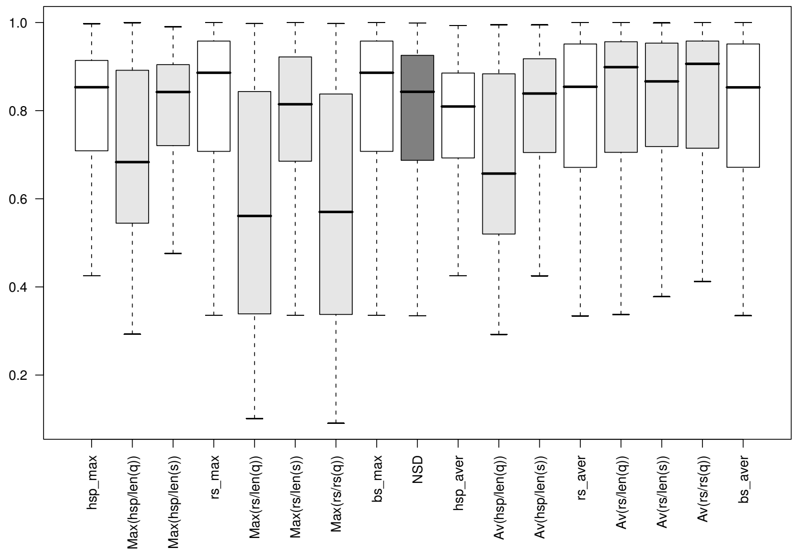

Fig. 3. Boxplot of AAUCs (1NN) for some single and combined BLAST output parameters for the LN dataset. The boxes for the combined output parameters are shown in light gray.

components: a) the alignment i.e. the BLAST running time; b) an aggregation time necessary to format the data to be used by the ML, and c) the learning time taken by the ML algorithm.

BLAST itself has three basic steps: a pre-alignment step, where all high scoring complementary k-words of the query sequence are selected and searched in the database; and the actual alignment phase, when the matched regions are expanded. Supposing that N is the length of the query, M is the length of the database, and k is the word length and, since there are 20^k possible words, the complexity of BLAST in terms of big $\mathcal{O}$ notation is given by $\mathcal{O}(N) + \mathcal{O}(N.20^k) + \mathcal{O}(M) + \mathcal{O}(M.N)$, i.e., $\mathcal{O}(M.N)$. Although, it has the same complexity as DP algorithms, BLAST is much more efficient ($\sim$ 500 times) since it discards non significant local alignments.

Concerning SVMs, standard training algorithms have time complexity of $\mathcal{O}(n^3)$, and space complexity of $\mathcal{O}(n^2)$, where n is the size of input data [8] – n is the amount of sequences in the training set. Although there are improved algorithms that can reduce the time complexity to $\mathcal{O}(n^{2.3})$, and to $\mathcal{O}(n)$ with parallelization, our tests were performed in R language using SMO-type decomposition method [9]. The time complexity for the SVM training phase is $n_{SV}.Dn.n$, and $n_{SV}.Dn$ for test phase, where n_{SV} is the number of support vectors and Dn is the dimension of the input data. In a similar way, the space complexity for the training phase is n^2, and $n_{SV}.Dn$ for the test phase.

The total learning computational complexity of the full pairwise approach is $\mathcal{O}(n^3)$, since it comprehends obtaining the kernel using BLAST, $\mathcal{O}(n^2.M^2)$, and

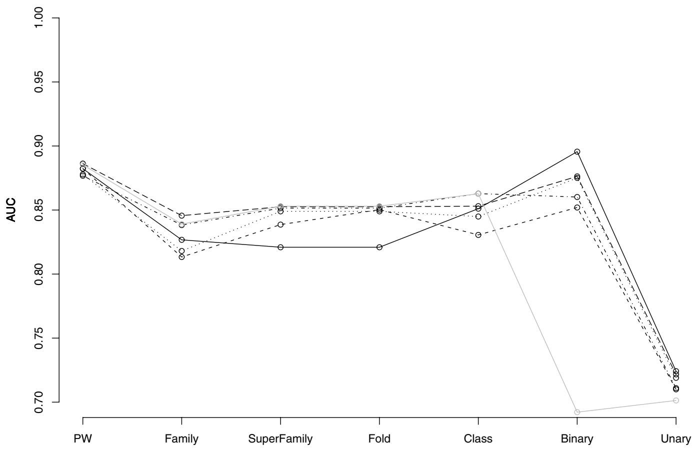

Fig. 4. Performance (AAUCs) of an SVM classifier with a linear kernel for different inputs – rs_{max} (solid), NSD (dashed), $Max(hsp/len(subject))$ (dotted), $Max(hsp/len(query))$ (dot-dashed), $Max(rs/len(subject))$ (long dash), and $Avg(rs/len(subject))$ (gray) – and representation strategies for the SCOP40mini dataset

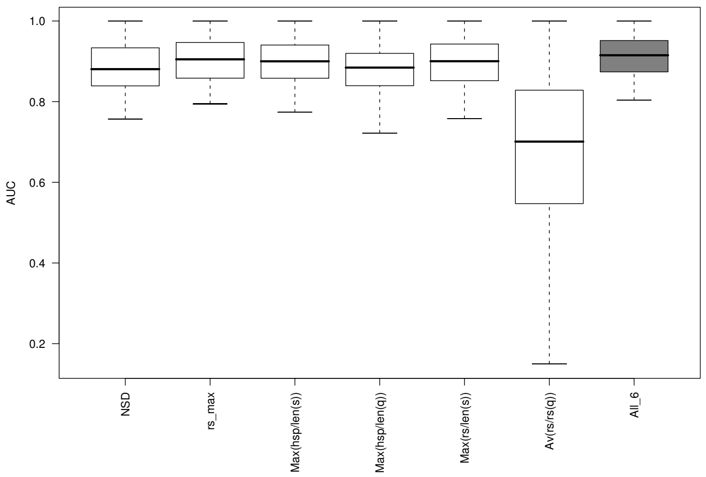

Fig. 5. Boxplot of AUCs for classifiers based on a linear SVM for the SCOP40mini dataset using 6 different similarity measures and a combination of them

the SVM training, $n^2.n_{SV}$. For the groupwise approaches the BLAST complexity remains the same, but an aggregation step with complexity $\mathcal{O}(n^2)$ is added to reduce the full pairwise matrix. Furthermore, the SVM training complexity step diminishes to $n_g.n.n_{SV}$ because of the reduction of the kernel matrix. In conclusion, the computational complexities for groupwise approaches are still polynomial, but one degree lower, $\mathcal{O}(n^2)$. It is noteworthy that using 1NN requires no training at all.

The first step of test phase consists of aligning a query to every sequence in training set using BLAST. This step has time complexity of $n.M^2$. BLAST running time are constant, independent of the algorithm or representation used. The complexity of the aggregation step is $\mathcal{O}(n)$. The SVM test complexity is given by $n^2.n_{SV}$ when using full pairwise strategy, and it is $n.n_{SV}$ for groupwise approaches. Hence, the computational complexity for test phase can be given by $\mathcal{O}(n^2)$ for the full approach, and $\mathcal{O}(n)$ for the aggregate ones. For test phase, the computational complexity using 1NN is equivalent to SVM.

The SVM time complexity used here was based on a linear kernel. When using a radial kernel, for example, the complexity increases proportionally to n_{SV}.

Briefly, the more efficient the aggregate representation, the lower is the learning complexity. Usually, running time for the test phase is smaller than that for the training phase. As expected, the most resource intensive strategy is the full approach.

3.4 A Protein Classifier Committee Machine

Based on the insights from analysis and tests performed here, we have designed a simple heuristic predictor that is based on a few BLAST output parameters. The predictor works like a voting system of 1NN predictors predicting protein function based on sequence similarity. The algorithm description is the following:

Given a query, one runs BLAST against the training set in order to obtain the categories of:

a) the protein (in the training set) which has higher raw-score (rs);
b) the protein which has higher hsp;
c) the proteins which have the five highest bit-score (bs).

The predictor uses the categories, i.e. functions of (a) and (b) as votes. A third vote comes from an aggregate feature that is given by the majority function from the five proteins in (c). The third vote is a variation of the NSD approach, adapted in such a way that categories with few instances are not underweight.

The basic idea is that the majority vote suggests the query function. If no majority is obtained, the Committee Machine follows the NSD vote, because it has yielded the best results during tests in SBASE [3].

The computational efficiency of the predictor is excellent. First, it does not require a training time and, secondly, it only needs running BLAST and ordering its output during tests. For example, if SVMs were used instead, it would require further training/tests for a set of SVMs equivalent to the amount of categories in the dataset.

In spite of its simplicity this 1NN-based Committee Machine has proved to be as accurate as the SVM classifier, providing the best performance on the SCOP40mini dataset. It consists of a set of linear SVMs that uses 6 different BLAST output parameters as inputs. Since the Committee Machine can not be assessed by ROC Curve, the classifiers performances are compared in terms of True Positive Rate (TPR) and True Negative Rate (TNR), in Table 4.

Table 4. Average Classification in terms of True Positive and Negative Rate for a set of 54 SVMs using a linear kernel and 6 BLAST output parameters as input vs. the simple Committee Machine for the SCOP40mini dataset

Classifier	**TPR**	**TNR**
54 SVM(All6)	0.5532	0.9446
1NN-Committee Machine	0.5067	0.9777

4 Conclusions

We have assessed the performance and computational complexity of protein classifiers based on kernel spaces obtained from pairwise sequence alignment. The binary representation strategy has presented one of the best classification performances, and also one of the minimum computational time cost.

We compared different BLAST output parameters in a feature selection analysis. The analysis showed that the feature performances are dataset dependent and there is no unanimously best similarity measures applicable to all datasets. However, some combined features have shown very good performance and could be taken into account when using a reduced approach.

Finally, we have shown that it is possible to design a simple and efficient protein classifier using a combination of separate classifiers into a voting system.

References

1. Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.: Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. Nucleic Acid Research 25, 3389–3402 (1997)
2. Liao, L., Noble, W.S.: Combining Pairwise Sequence Similarity and support Vector machines for Detecting Remote Protein Evolutionary and Structural Relationship. Journal of Computational Biology 10, 857–868 (2003)
3. Murvai, J., Vlahovicek, K., Barta, E., Pongor, S.: The SBASE Protein Domain Library, Release 8. 0: A collection of Annotated Protein Sequence Segments. Nucleic Acid Research 29, 58–60 (2001)
4. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
5. Kaján, L., Kertesz-Farkas, A., Franklin, D., Ivanova, N., Kocsor, A., Pongor, S.: Application of a Simple Log-likelihood Ratio Approximant to Protein Sequence Classification. Bioinformatics 22, 2865–2869 (2006)
6. Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T.: ROCR: Visualizing Classifier Performance in R. Bioinformatics 21(20), 3940–3941 (2005)
7. Sonego, P., Pacurar, M., Dhir, S., Kertézs-Farkas, A., Kocsor, A., Gaspari, Z., Leunissen, J., Pongor, S.: A Protein Classification Benchmark Collection for Machine Learning. Nucleic Acid Research 236, D232–D236 (2006)
8. Tsang, I., Kwork, J., Cheung, P.-M.: Core Vector Machines: FAST SVM Training on Very Large Data Sets. Journal of Machine Learning Research 6, 363–392 (2005)
9. Fan, R.-E., Chen, P.-H., Lin, C.-J.: Working Set Selection Using the Second Order Information for Training SVM. Journal of Machine Learning Research 6, 1889–1918 (2005)