A One-Class Classification Approach for Protein Sequences and Structures

András Bánhalmi[1], Róbert Busa-Fekete[1,2], and Balázs Kégl[2]

[1] Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, Aradi vértanúk tere 1., H-6720 Szeged, Hungary
{banhalmi,busarobi}@inf.u-szeged.hu
[2] LAL, University of Paris-Sud, CNRS, 91898 Orsay, France
kegl@lal.in2p3.fr

Abstract. The One-Class Classification (OCC) approach is based on the assumption that samples are available only from a target class in the training phase. OCC methods have been applied with success to problems where the classes are very different in size. As class-imbalance problems are typical in protein classification tasks, we were interested in testing one-class classification algorithms for the detection of distant similarities in protein sequences and structures. We found that the OCC approach brought about a small improvement in classification performance compared to binary classifiers (SVM, ANN, Random Forest). More importantly, there is a substantial (50 to 100 fold) improvement in the training time. OCCs may provide an especially useful alternative for processing those protein groups where discriminative classifiers cannot be easily trained.

Keywords: One-class classification, Protein classification, ROC analysis.

1 Introduction

The classification of proteins (domain types, structural classes, protein families) is a key issue in genome annotation. The simplest methods of protein classification are based on pairwise comparisons; more advanced approaches use generative models of the positive class like Hidden Markov Models (HMMs), while more recent methods like Support Vector Machines (SVMs) are based on discriminative models in which the positive and negative classes are both used in the training phase. However, the known protein groups have some typical properties that make the application of classification algorithms difficult or impractical. First, protein classes are very heterogeneous in most of their characteristics (such as the number of known members, protein size, within-group similarities, separation from other groups). Second, a large proportion of known protein groups have only one or two known members. Third, the classes are imbalanced, there being many more negative examples than positive ones. The training of support vector machines is difficult on such data, and generative models like the popular HMM need manually curated multiple alignments that require a substantial human overhead. These points also pose problems to the recent machine learning approaches that use new input space representations or similarity measures.

I. Măndoiu, G. Narasimhan, and Y. Zhang (Eds.): ISBRA 2009, LNBI 5542, pp. 310–322, 2009.

One class classification (OCC) approaches have been successfully used in various fields where only positive examples are available for training. Application examples include image retrieval [1], the fault detection of machinery [2], automated currency validation [3], bioacoustic monitoring [4], document classification [5], and spam filtering [6]. While discriminative models try to establish a decision surface between the positive class and the other class(es), OCC methods try to draw a closed decision surface around the positive class that surrounds the majority of the positive training instances (see Figure 1). In this scenario, the negative examples are outliers, and the relevant methods are referred to as Outlier Detection or Novelty Detection in different fields. The area of OCC includes several algorithms like generative probability density estimation methods (Gaussian Mixture Model (GMM) [7,8], Parzen estimator [9]), reconstruction methods (k-means [8], Autoencoder Neural Networks [10]), and boundary estimators (k-centers [11], SVDD [12,13,14], NNDD [15]).

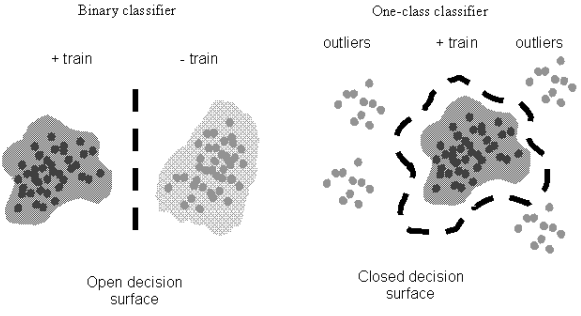

Fig. 1. The difference between the discriminative and OC classification methods

The aim of this paper is to test the performance of OCC algorithms in protein classification tasks, using standardized datasets developed for benchmarking machine learning algorithms. The classification tasks were selected in such a way that the members of a new protein family (test set) had to be detected based on other known members of a protein superfamily (training set), using measures of sequence similarity (BLAST [16], Smith/Waterman [17]) or structure similarity (DALI [18], PRIDE [19]). We carried out a ROC analysis for the characterization of classifier performance and found that OCC methods provide a slight improvement with respect to discriminative methods (SVM [20], ANN [8], Random Forest [21]). On the other hand they require 50 to 100 times less training time, which makes them promising candidates for large scale applications.

2 Protein Classification Datasets

We used two different classification benchmark datasets taken from the Protein Classification Benchmark Database [22].

2.1 The COG Dataset

This dataset is a subset of the COG database of functionally annotated orthologous sequence clusters [23]. In the COG database, each COG cluster contains functionally related orthologous sequences belonging to unicellular organisms, including archaea, bacteria and unicellular eukaryotes. For a given COG group, the positive test set included the sequences from three unicellular eukaryotic genomes, while the positive training set was compiled from the rest of the sequences in the group. Of the over $5,665$ COGs we selected 117 that contained at least 8 eukaryotic sequences (positive test group) and 16 additional prokaryotic sequences (positive training group). This dataset contained $17,973$ sequences. The negative training/test sets were obtained by randomly assigning sequences from the remaining COGs to the two groups. In this way it reduced to 117 classification tasks, but we used only a subset of them, because we did not use those which had a 1.0 Nearest Neighbor AUC performance. So we chose just 13 tasks from the 117. These COGs were used as positive elements in these learning tasks, that is: COG0406, COG0526, COG0631, COG0695, COG0697, COG0699, COG0814, COG0842, COG0847, COG1310, COG1752, COG2036, COG2801.

2.2 The SCOP40 Dataset

The evaluation of classification performance was tested on a sequence dataset designed to test distant protein similarities [24]. This set consists of $4,352$ protein domain sequences (whose lengths range from 20 to 994 amino acids) selected from the SCOP database [25]. The sequences of this dataset belong to 55 superfamilies, which were divided into training sets and test sets in such a way that the test set consisted of members of a protein family that was not represented in the training set, i.e. there was a low degree of sequence similarity and no guaranteed evolutionary relationship between the two sets.

2.3 Sequence Comparison Algorithms

The protein comparison datasets were taken from the Protein Classification Benchmark website [22]. In experiments a protein sequence comparison was performed with BLAST version 2.2.4 of the BLAST program [16] using a cutoff score of 50, or with the Smith-Waterman algorithm [17], as implemented in MATLAB. The BLOSUM 62 matrix [26] was also used in each case. Afterwards, a protein structure comparison was carried out with DaliLite [18] and with PRIDE2 [19] using default parameters.

2.4 Data Representation

The machine learning algorithms can accept only fixed length, real-valued input vectors such as a kernel representation [24] in which each protein X is represented by a feature vector $F_X = f_{x_1}, f_{x_2}, \ldots f_{x_n}$, where n is the total number of proteins in the training set and f_{x_i} is a similarity/distance score, such as the BLAST score, between X and the ith sequence in the training set. Here we

used a more compact representation, where each sequence was represented by its average similarity score obtained from one of the superfamilies represented in the training set [27,28]. After the aggregation step, the length of the training vectors became be equal to the number of superfamilies. Thus in the case of the SCOP40 dataset we had 24 dimensional vectors instead of 1357. In the case of the COG dataset [23], we used the average similarity score on various COGs as an aggregate similarity measure and this resulted in 117 dimensions instead of the original 17,947. We also found that this aggregation does not affect very much the classification performance.

3 Methods

3.1 Data Description Methods

In the following we will give a brief description of the Data Description (or One-Class Classification) methods used in our experiments. The methods which estimate the probability density give a probability value for a test data sample ($p(x)$), and this value can be used for ranking the test samples. Other methods supply distance values ($d(x)$), and the negative values of the magnitudes of these distances can be used for scoring.

Gaussian Data Description: The Gaussian Data Description [29] seeks to directly approximate the class-conditional probability distribution corresponding to a class with a multidimensional Gaussian density function ($p(x)$):

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

Here D denotes the dimension of the vector-space, and μ and Σ represent the mean and covariance, respectively. The parameters can be computed directly from the training data. The main computational effort is the inversion of the covariance matrix. When the covariance matrix is singular the following regularization step is necessary before the inversion

$$\Sigma = \Sigma + rI,$$

where $r \in \mathbb{R}^+$ is a small regularization parameter and I is the D dimensional identity matrix.

Parzen Data Description: This method [15] estimates the class-conditional probability density of the training data via the sum of kernel functions centered to the training examples. The most commonly employed kernel function is the Gaussian function with a zero mean and a variance of one:

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \varphi(x, x_i, hI),$$

where I is the unit covariance matrix, the x_i values represent the training data samples, and N stands for the number of data samples. The parameter h is

the bandwidth (a smoothing parameter), which can be fixed beforehand or an optimal value can be found using a maximum likelihood method.

Nearest Neighbor Data Description: Here the score of a test vector is expressed by the distance between the test vector and its nearest neighbor in the training set:

$$\varepsilon(x) = -\|x - NN(x)\|$$

K-means Data Description: This method belongs to the family of so-called reconstruction methods, because it minimizes a reconstruction error on the training set. The score of the test data samples can also be expressed using this reconstruction error function. For the k-means data description method, the following reconstruction error is applied [15]:

$$\varepsilon(x_i) = \min_k \|x_i - \mu_k\|^2$$
$$\varepsilon = \sum_{i=1}^{N} \varepsilon(x_i),$$

where x_i represent the training data samples, N denotes the number of samples, and μ_k is one of the mean values to be optimized.

After the training phase, the distance score for a test vector z is calculated via the following formula:

$$d(z) = \min_k (z - \mu_k)$$

K-centers Data Description: The only difference between this method and the K-means Data Description is the objective function to be minimized [7]. Here the objective function is the maximal radius of the hyperspheres embedding the clusters.

$$\varepsilon(x_i) = \min_k \|x_i - \mu_k\|^2$$
$$\varepsilon = \max_i \varepsilon(x_i),$$

After the training phase, the distance score for a test vector z is calculated by using the formula:

$$d(z) = \min_k (z - \mu_k)$$

Self-Organizing Map (SOM): This method is similar to the k-means procedure in the sense that reference vectors corresponding to a training set are iteratively updated until convergence. Here denoting the k-th reference vector by w_k (it is similar to μ_k above) for each training data sample x, the closest reference vector w_k is determined. After, this the closest reference vector is altered so as to be closer to the actual x point. The difference here between this method and other similar (so-called Learn Vector Quantisation) methods is that each reference vector corresponds to a grid point in a low $(1-3)$ dimensional space.

In the learning phase not only the closest w_k reference vector to x is updated, but also some of the vectors whose grid point is close to the grid point corresponding to w_k. In this way –after convergence– the reference vectors corresponding to the grid points will be close to each other when they are close on the grid, so the low dimensional grid tries to preserve the topological properties of the input space [30].

The test mechanism is the same as that for previous methods; that is, the distance score for a test vector z is calculated via the formula:

$$d(z) = \min_k (z - w_k)$$

Support Vector Data Description: This novel approach finds the minimum enclosing ball (hypersphere) for the positive training data, and by applying the 'kernel trick' the minimum enclosing ball is found in the kernel-space. Various kernel-functions can be used, but in our experiments the radial basis kernel was applied. Though we do not give a detailed description of the SVDD method here, the interested reader can peruse [13].

Counter-Example Generation-Based Data Description: Here the problem of one-class classification (or ranking) is transformed into a two-class problem: a simple algorithm is applied that automatically generates artificial counter-examples using just the positive data, in such a way that the generated counter-examples lie outside the region of positives at a predefined distance from them. Afterwards, traditional discriminative classification methods can be utilized to separate the positive and the artificial negative examples [31]. For the negative example generation, and for the two-class classification problem some new methods were introduced in [32]. Here we will use the original counter-example generation method, but for the purpose of separating the positive examples from the generated negative ones we will apply both the ν-SVM (Support Vector Machine) and the RBN (Radial Basis Network) classifiers.

3.2 Reference Binary Classifiers

The implementation of Artificial Neural Network (ANN) and Random Forest (RF) was done using WEKA, which is an open-source JAVA package for machine learning [33]. The class-conditional probability estimation for the test elements was performed in the same way as that implemented in WEKA. The data representation employed by these models was the same as the one we used for the OCCs. In addition, we used the SVMLight program [34], which is a Support Vector Machine implementation. The main advantage of the latter implementation is that it can handle sparse data representations as well.

4 Performance Evaluation

The evaluation was carried out via standard receiver operating characteristic (ROC) analysis [35,36], which provides a measure of both the sensitivity and the

Table 1. Comparison of the AUC performance of different Classifiers on SCOP40 dataset. The comparison was carried out using the aggregate feature representation described in the 3.1 section.

	AUC AUC_{50}	BLAST	SW	DALI	PRIDE
One-class classifiers	NNDD	0.8980 0.7315	0.8831 0.6640	0.8845 0.6622	0.8109 0.4104
	CE-OC(SVM)	**0.9213** **0.8043**	**0.9558** **0.8227**	0.9527 0.8457	0.8831 0.6071
	CE-OC(RBN)	0.9147 0.7913	0.9185 0.7267	0.9398 0.8317	0.8965 0.6098
	K-center	0.9184 0.7945	0.9197 0.7327	0.9834 0.8686	0.8704 0.5766
	K-means	0.9206 0.7914	0.9301 0.7755	0.9857 0.9058	0.8849 0.6655
	SVDD	0.9200 0.7709	0.9421 0.7840	0.9895 0.9402	0.9030 0.6581
	Parzen	0.9048 0.7574	0.9384 0.7589	0.9825 0.8747	0.8947 0.5863
	SOM	0.9184 0.7630	0.9362 0.7190	0.9827 0.8685	0.8963 0.6291
	Gauss	0.9185 0.7962	0.9387 0.8036	0.9912 0.9526	0.8581 0.6585
Binary classifiers	ANN	0.8907 0.7912	0.9383 0.8231	0.9932 0.9795	**0.9222** **0.7686**
	SVM	0.8857 0.7878	0.9441 0.8195	0.9108 0.8434	0.9171 0.7326
	RForest	0.8082 0.6332	0.8884 0.6984	**0.9948** **0.9678**	0.8988 0.6472

specificity of the classification based on a ranking of the objects to be classified [37]. The ROC curve is a curve which plots the sensitivity against the specificity. The integral of this curve is the "area under curve" or AUC value which is is equal to the probability that the score of a randomly chosen positive example is higher than that of a randomly chosen negative one [38,39]. $AUC = 1.0$ corresponds to perfect ranking, while a random ranking has an $AUC = 0.5$ value on average [35]. In order to limit the effect of class imbalance and to have datasets of a manageable size, it is customary to truncate the toplist so as to include only a limited number of negative samples [37]. The resulting measures are the so-called (e.g. ROC_{10}, ROC_{50}) values. In our experiments we used both the full AUC and AUC_{50}.

An other important question in classification tasks is how to choose the parameters of the different machine learning models. For the reference multi-class methods (like SVM or ANN) their parameters were chosen according to the references in the protein classification benchmark [22]. For One-Class models which have one or more parameters influencing the performance of the ranking

Table 2. Comparison of the AUC performance of different Classifiers on SCOP40 dataset. The comparison was carried out without using any aggregated features. This table is just for a comparison between the results of the applications of aggregated and non-aggregated features.

	AUC AUC_{50}	BLAST	SW	DALI	PRIDE
One-class classifiers	NNDD	0.7403 0.5076	0.7102 0.5076	0.7521 0.5494	0.8027 0.4330
	CE-OC(SVM)	0.8393 0.5686	0.96 0.7911	0.9527 0.8457	0.9081 0.6826
	CE-OC(RBN)	0.8283 0.4942	**0.9673** **0.8327**	0.9398 0.8317	0.8988 0.6729
	K-center	0.7943 0.5762	0.9299 0.6862	0.9469 0.7835	0.8710 0.6198
	K-means	0.7758 0.4575	0.9547 0.7595	0.9443 0.8171	0.8877 0.6067
	SVDD	0.8405 0.6015	0.9583 0.7828	0.9695 0.9028	0.8993 0.6571
	Parzen	**0.8949** **0.6846**	0.9405 0.7103	0.8386 0.6208	0.8361 0.6660
	SOM	0.8014 0.5111	0.9511 0.7589	0.9440 0.7912	0.9019 0.6738
	Gauss	0.8266 0.4789	0.9628 0.8134	0.9499 0.7200	0.8790 0.6524
Binary classifiers	ANN	0.7054 0.5278	0.7896 0.5336	0.9543 0.7319	0.9253 0.7098
	SVM	0.8854 0.7593	0.9437 0.8237	**0.9886** **0.9822**	**0.9389** **0.8344**
	RForest	0.6521 0.6132	0.8599 0.7122	0.9944 0.98	0.8945 0.7561

(or classification) problem, in the most cases the default value of the parameters proved to be the best ones, cross validation methods were not applied because of the very few positive training data. Only two exceptions should be mentioned. When Gauss DD was tested, the regularization matrix had a 0.05 multiplier, and the with of the Parzen window was set to 0.5.

All the training datasets were normalized to the $[-1, 1]$ interval.

5 Results and Discussion

Table 1 shows the performance of different classifiers on the SCOP40 database using BLAST, Smith-Waterman, DALI and PRIDE similarities taken from the Protein Classification Benchmark dataset (PCB) [22]. Here the AUC values were determined for the 55 classification tasks specified in PCB, and the average of the 55 AUC values is given in Table 1. The results show that OCCs can achieve

Table 3. Comparison of the AUC performance of different classifiers on selected classification tasks taken from COG dataset. The comparison was carried out using the aggregate feature representation described in the 3.1 section.

AUC AUC_{50}	BLAST	SW
NNDD	0.9669 0.9423	0.8027 0.4330
CE-OC(SVM)	0.9801 0.7219	0.9739 0.8073
CE-OC(RBN)	0.9810 0.7326	0.9744 0.7476
K-center	0.9975 0.9512	0.9824 0.8136
K-means	0.9977 0.9493	0.9790 0.8195
SVDD	0.9817 0.9520	0.9715 0.7833
Parzen	0.9044 0.7637	0.8696 0.6877
SOM	0.9975 0.9483	**0.9829** **0.7778**
Gauss	**0.9978** **0.9471**	0.9742 0.9382
ANN	0.9763 0.7838	0.9627 0.8178
SVM	0.9885 0.9802	0.9752 0.9394
RForest	0.8558 0.7849	0.8173 0.9437

a modest but consistent improvement compared to the binary classifiers, both in AUC and in AUC_{50}.

The COG dataset consists of groups of orthologues which were generated by a clustering method based on pairwise BLAST sequence comparison. The resulting orthologue clusters (COGs) are compact and even a simple classifier like 1NN can achieve an AUC of 1.00 on the majority of the groups. In order to get a "difficult" subset, we picked 13 COG groups ($14,939$ sequences) with a 1NN AUC score below 0.95. The results we got on this subset are shown in Table 3. These results are consistent with the SCOP results, where we found that OCCs outperform the binary classifiers. After, typical training times are shown in Table 4. Here it is apparent that some OCCs have a 50 to 100 times smaller training time than binary classifiers.

Looking at our results in more detail, our first observation could be what is referred to as 'no free lunch theorem' in machine learning [7]: there is no classifier which is the best for all the problems. The OC methods as well as

Table 4. Typical training times (in seconds) for the classifiers used in this study

Task	a.118.1.	b.40.4.	c.2.1.
+train	21	47	103
-train	664	642	604
OCC methods			
CE-OC (SVM)	6.95	14.32	33.67
CE-OC (RBN)	7.17	16.45	71.15
K-center	0.26	0.29	0.35
K-means	0.12	0.18	0.23
SVDD	0.14	0.51	0.76
Parzen	0.14	0.21	0.31
SOM	16.12	34.64	79.87
Gauss	0.48	0.51	0.56
Binary methods			
ANN	$> 15m$	$> 15m$	$> 15m$
SVM	14.71	14.66	15.75
Random Forest	61.89	54.16	64.15

the discriminative methods have a high diversity in their performance, when using different feature sets on different tasks. It is interesting to note that on the DALI features, discriminative methods (ANN, RF) perform better than OCC methods (CE, Gauss, SVDD, SOM), while on SW features, the situation is just the reverse. It is also seen that the relative ranking of methods is different on the SCOP40 and COG datasets, nevertheless the OCC methods perform better in both cases. Among the binary classifiers tested, ANN apparently gave the most robust results.

Next we should mention that the findings reported here were obtained using a compact, aggregated feature representation. An analysis was also carried out on the complete, non-aggregated feature set. The results given in Table 2 show the same tendencies as those reported above (only the results on scop40 is reported here, in the case of COG similar results were obtained). The results show a very high improvement in the classification performance, when – a very effective dimension-reduction ie. – aggregated features were used for our class-imbalanced and high dimensional problems.

6 Conclusions

Based on the above comparisons, we may conclude that one-class classifiers can provide a viable alternative to binary classifiers in protein classification tasks. They do not require multiple alignment and can be easily incorporated into multiple classifier systems. As they require short training times, they can be especially useful for large-scale applications, and may provide a solution for the protein groups that binary classifiers cannot handle.

Acknowledgements

The authors thank Sándor Pongor for helpful suggestions. This research was supported by the French National Research Agency. This work was partly supported in part by the NKTH grant of the National Technology Programme 2008 (project codename AALAMSRK NTP OM-00192/2008) of the Hungarian government.

References

1. Chen, Y., Zhou, X.S., Huang, T.S.: One-class SVM for learning in image retrieval. In: 2001 International Conference on Image Processing proc., vol. 1, pp. 34–37 (2001)
2. Shin, H.J., Eom, D.-H., Kim, S.-S.: One-class support vector machines: an application in machine fault detection and classification. Comput. Ind. Eng. 48(2), 395–408 (2005)
3. He, C., Girolami, M., Ross, G.: Employing optimised combinations of one-class classifiers for automated currency validation. Pattern Recognition 37, 1085–1096 (2004)
4. Sachs, A., Thiel, C., Schwenker, F.: One-class support-vector machines for the classification of bioacoustic time series. ICGST International Journal on Artificial Intelligence and Machine Learning (AIML) 6(4), 29–34 (2006)
5. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. Journal of Machine Learning Research 2, 139–154 (2001)
6. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley and Son, New York (2001)
8. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
9. Parzen, E.: On the estimation of a probability density function and mode. Annals of Mathematical Statistics 33, 1065–1076 (1962)
10. Japkowicz, N., Myers, C., Gluck, M.A.: A novelty detection approach to classification. In: IJCAI, pp. 518–523 (1995)
11. Ypma, A., Duin, R.: Support objects for domain approximation (1998)
12. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471 (2001)
13. Tax, D.M.J., Duin, R.P.W.: Support vector domain description. Pattern Recogn. Lett. 20(11-13), 1191–1199 (1999)
14. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Mach. Learn. 54(1), 45–66 (2004)
15. Tax, D.M.J.: One-class classification; Concept-learning in the absence of counter-examples. Ph.D thesis, Delft University of Technology (2001)
16. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215(3), 403–410 (1990)

17. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology 147(1), 195–197 (1981)
18. Holm, L., Park, J.: Dalilite workbench for protein structure comparison. Bioinformatics (16), 566–567 (2000)
19. Vlahovicek, K., Gaspari, Z., Pongor, S.: Efficient recognition of folds in protein 3d structures by the improved pride algorithm. Bioinformatics (21), 3322–3323 (2005)
20. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Son, Chichester (1998)
21. Breiman, L.: Random forests. Machine Learning V45(1), 5–32 (2001)
22. Sonego, P., Pacurar, M., Dhir, S., Kertész-Farkas, A., Kocsor, A., Gáspari, Z., Leunissen, A.M., Pongor, S.: A protein classification benchmark collection for machine learning. Nucleic Acids Research 35(suppl. 1), D232–D236 (2007)
23. Tatusov, R.L., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Kiryutin, B., Koonin, E.V., Krylov, D.M., Mazumder, R., Mekhedov, S.L., Nikolskaya, A.N., Rao, B.S., Smirnov, S., Sverdlov, A.V., Vasudevan, S., Wolf, Y.I., Yin, J.J., Natale, D.A.: The cog database: an updated version includes eukaryotes. BMC Bioinformatics 4 (September 2003)
24. Liao, L., Noble, W.S.: Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In: RECOMB 2002: Proceedings of the sixth annual international conference on Computational biology, pp. 225–232. ACM Press, New York (2002)
25. Andreeva, A., Howorth, D., Brenner, S.E., Hubbard, T.J., Chothia, C., Murzin, A.G.: Scop database in 2004: refinements integrate structure and sequence family data. Nucleic Acids Res. 32(Database issue) (January 2004)
26. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. U S A 89(22), 10915–10919 (1992)
27. Vlahovicek, K., Kajan, L., Agoston, V., Pongor, S.: The sbase domain sequence resource, release 12: prediction of protein domain-architecture using support vector machines. Nucleic Acids Research 33(suppl. 1), 223 (2005)
28. Murvai, J., Vlahovicek, K., Szepesvári, C., Pongor, S.: Prediction of protein functional domains from sequences using artificial neural networks. Genome Res. 11, 1410–1417 (2001)
29. Paalanen, P.: Bayesian classification using Gaussian mixture model and EM estimation: Implementations and comparisons. Technical report, Department of Information Technology, Lappeenranta University of Technology, Lappeenranta (2004)
30. Allinson, N.M., Yin, H.: Self-organising maps for pattern recognition. In: Oja, E., Kaski, S. (eds.) Kohonen Maps, pp. 111–120. Elsevier, Amsterdam (1999)
31. Bánhalmi, A., Kocsor, A., Busa-Fekete, R.: Counter-example generation-based one-class classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 543–550. Springer, Heidelberg (2007)
32. Bánhalmi, A.: One-class classification methods via automatic counter-example generation. In: AIAP 2008: Proceedings of the 26th IASTED International Multi-Conference, Anaheim, CA, USA. ACTA Press (2008)
33. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco (2005)
34. Joachims, T.: Making large-scale support vector machine learning practical. MIT Press, Cambridge (1998)

35. Egan, J.P.: Signal Detection theory and ROC Analysis. Academic Press, New York (1975)
36. Sonego, P., Kocsor, A., Pongor, S.: Roc analysis: applications to the classification of biological sequences and 3d structures. Brief Bioinform. (January 2008)
37. Gribskov, M., Robinson, N.: Use of receiver operating characteristic (roc) analysis to evaluate sequence matching (1996)
38. Cortes, C., Mohri, M.: Auc optimization vs. error rate minimization (2004)
39. Ingleby, J.D.: Signal detection theory and psychophysics. Journal of Sound Vibration 5, 519–521 (1967)