Equivalence Learning in Protein Classification

Attila Kertész-Farkas[1,2], András Kocsor[1,3], and Sándor Pongor[4,5]

[1] Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, Aradi vértanúk tere 1, H-6720, Szeged, Hungary
{kfa, kocsor}@inf.u-szeged.hu
[2] Erasmus Program, Technische Universität Dresden, Germany
[3] Applied Intelligence Laboratory, Petofi S. sgt. 43, H-6725, Szeged, Hungary
[4] Bioinformatics Group, International Centre for Genetic Engineering and Biotechnology, Padriciano 99, I-34012 Trieste, Italy
pongor@icgeb.org
[5] Bioinformatics Group, Biological Research Centre, Hungarian Academy of Sciences, Temesvári krt. 62, H-6701 Szeged, Hungary

Abstract. We present a method, called equivalence learning, which applies a two-class classification approach to object-pairs defined within a multi-class scenario. The underlying idea is that instead of classifying objects into their respective classes, we classify object pairs either as equivalent (belonging to the same class) or non-equivalent (belonging to different classes). The method is based on a vectorisation of the similarity between the objects and the application of a machine learning algorithm (SVM, ANN, LogReg, Random Forests) to learn the differences between equivalent and non-equivalent object pairs, and define a unique kernel function that can be obtained via equivalence learning. Using a small dataset of archaeal, bacterial and eukaryotic 3-phosphoglycerate-kinase sequences we found that the classification performance of equivalence learning slightly exceeds those of several simple machine learning algorithms at the price of a minimal increase in time and space requirements.

1 Introduction

The classification of proteins is a fundamental task in genome research. In a typical application, a protein sequence object (a string of several tens to several hundred characters) has to be classified into one of the several thousand known classes, based on a string similarity measure. Sequence similarity is thus a key concept since it can imply evolutionary, structural or functional similarity between proteins.

Early methods of protein classification relied on the pairwise comparison of sequences, based on the alignment of sequences using exhaustive dynamic programming methods [1] [2] or faster, heuristic algorithms [3][4]. Pairwise comparison yielded a similarity measure that could be used to classify proteins on an empirical basis. The next generation of methods then used generative models for the protein classes and the similarity of a sequence to a class was assessed by a score computed between the model and the sequence. Hidden Markov Models (HMMs) are now routinely used in protein classification [5], but there are many

P. Perner (Ed.): MLDM 2007, LNAI 4571, pp. 824–837, 2007.
© Springer-Verlag Berlin Heidelberg 2007

other, simpler types of description in use (for a review, see: [6]). Discriminative models (such as artificial neural networks and support vector machines etc.) are used in a third generation of protein classification methods where the goal is to learn the distinction between class members and non-members. Roughly speaking, 80-90% of new protein sequence data can be classified by simple pairwise comparison. The other, more sophisticated techniques are used mostly to verify whether a new sequence is a novel example of an existing class or it represents a truly new class in itself. As the latter decisions refer to the biological novelty of the data, there is a considerable interest in new, improved classification methods.

Kernel methods represent a subclass of discriminative models in which a pairwise similarity measure calculated between objects is used to learn the decision surface that separates a class from the rest of the database. The kernel function can be regarded as a similarity function which has the additional property of always being positive semi-definite, which allows many novel applications to non-linear problems [7]. In the context of protein classification, kernel methods have an important practical advantage: the similarity measures developed in protein classification can be used to construct kernel functions, and so decision making can directly capitalize on the considerable empirical knowledge accumulated in various fields of protein classification. Over the past decade, many kernels have been developed for sequences such as the String kernel [8], Mismatch kernel [9], Spectrum kernel [10], Local Alignment Kernel [11] and the Fisher kernel [12]. For a good review of these applications, see [7].

This work aims to use a conceptual approach that is slightly different from the mainstream use of kernel functions. Let us first consider a database of objects and a similarity measure computed between each pair of objects. This setup can be visualized as a weighted graph (network) of similarities where the nodes are the proteins and the weighted edges represent the similarities between them. The network can also be represented as a symmetrical matrix in which the cells represent the pairwise comparison measures between the objects. Figure 1a shows a hypothetical database of 8 objects. We can vaguely recognize two groups in which the members are more similar to each other than to the objects outside the groups. Let us now suppose that an expert looks at the similarity data and decides that the two groups represent two classes, A and B, and there is another object that is not a member of either of these. Figure 1b illustrates this new situation. The members of the groups are now connected by an equivalence relation that exists only between the members of a given group. As a result, the similarity matrix becomes a simpler *equivalence matrix* in which only the elements between the members of the same class are non-zero. The aim of this study here is to use a similarity matrix given between a set of objects, and use it to learn the equivalence matrix defined by the classification task, as shown in the bottom part of Figure 1. We term this approach *equivalence learning*, which is characterized as follows. Protein classification methods seek to classify the objects, i.e. the nodes of the similarity network, which is a multi-class problem. In contrast, here we try to classify the edges of the similarity network into just two classes, one signifying equivalence the other the lack of it.

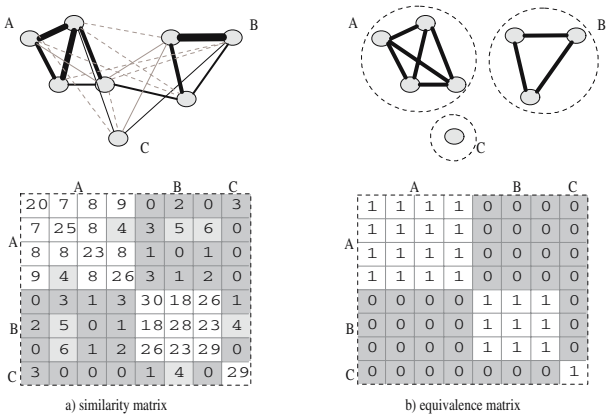

Fig. 1. Principle of Equivalence Learning (EL) I. A similarity matrix (left) can be determined by an all vs. all comparison of a database using algorithms such as BLAST or Smith-Waterman. The equivalence matrix (right) is a representation of groups (A,B) and an outlier identified by human experts. EL tries to learn the equivalence.

In more detail, a sequence pair is called equivalent when both of them belong to the same sequence group and is called non-equivalent or distinct when they do not. We define the *equivalence function* that returns one for equivalent sequences and zero for non-equivalent sequences taken from a different group - where the group could be a protein family, superfamily, fold or class. This function can be formulated as follows

$$\delta(s,t) = \begin{cases} 1 & s \text{ and } t \text{ belong to the same sequence group,} \\ 0 & \text{otherwise.} \end{cases}$$

Since the equivalence function defined over the sequence pairs gives a partition (equivalent/non-equivalent sequence pairs), learning this function essentially becomes a two-class classification problem that can be solved by one of the existing classification schemes. For clarity, the process is shown in Figure 2. A key issue here is to decide how we should represent the edges of the similarity network so that we can efficiently predict equivalence. A simple numerical value is not sufficient since, as shown in the example of Figure 1, a value 9 is sufficient for class membership in class A, but not in class B. As a solution, we will use a vectorial representation for the edges, hence we will need projection methods in order to represent a sequence pair in one vector. Now let $P : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n$ be such a projection function and let the corresponding database be

$$L = \{(x_i, y_i) \mid x_i = P(s_i, t_i),\ \ y_i = \delta(s_i, t_i),\ \ s_i, t_i \in \mathcal{S}\}, \tag{1}$$

where $\mathcal{S}$ denotes the set of sequences.

The construction of an equivalence function has been proposed by Cristianini et al who defined the concept as the ideal kernel, along with a kernel alignment

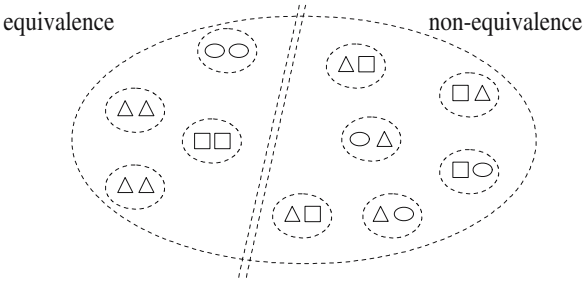

Fig. 2. Principle of equivalence learning II. Equivalence learning is a two-class problem defined on object pairs.

measure that quantifies the difference between a kernel and the ideal one [17]. Semidefinite programming techniques were then used to learn the kernel matrix from data and it was shown that using the labeled part of the data one can learn an embedding too for the unlabeled part [18]. Tsang and Kwok once suggested a feature weighting technique that allows one to approximate the ideal kernel [19].

In this paper we will examine how an equivalence function can be learnt with a standard machine learning method such as Artificial Neural Networks (ANNs) [13], Support Vector Machines (SVMs) [14] Logistic Regression (LogReg) [15] or Random Forest (RF) [16]. These methods learn a function f and for any sequence pair (s, t) return a score $f(z)$, where $z = P(s, t)$ and P is a projection function. If this score is greater than a certain threshold then the sequence pair classified is an equivalent pair, otherwise it is a non-equivalent pair. Unlike a class label predicted for a single protein, the score $f(z)$ can be considered as a similarity measure for a sequence pair (s, t) and thus the learned machine can be regarded as a fast similarity function for a sequence pair. In Section 2.3 we offer a way of constructing a kernel function from a decision boundary $f(z)$ obtained by SVM. Section 3 describes experiments that illustrate how these method can learn the equivalence function and how we can evaluate them in a protein classification context. Section 4 then gives a brief summary along with some conclusions.

2 Methods

2.1 Vectorization Step

Now we will present a way of defining projection functions which map any sequence pair into a fixed-length vector of real numbers, that is $P : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n$.

First, we define a method to vectorize a single sequence into a vector. The essential idea behind this is that a chosen protein sequence can be effectively captured by asking how similar a protein is to a large collection of other proteins [20][21]. Let us view a fixed set of sequences $\{f_1, f_2, \ldots, f_n\} \subseteq \mathcal{S}$ as a feature set and an arbitrary similarity function D. For a sequence $s \in \mathcal{S}$ let the corresponding vector w whose components are indexed by f_i and its corresponding

value w_{f_i} be the similarity score between f_i and the target sequence s, that is $\phi_D : \mathcal{S} \to \mathbb{R}^n$ such that $\phi_D(s)_{f_i} = D(s, f_i)$, where v_i denotes the component of vector v indexed by i. A mapping of this type is also known as an empirical feature map [22].

The next step is to form a vector from two sequence objects. Table 1 summarizes a few simple methods which were used in our experiments. For ease of notation the operators $a \cdot b$, $\sqrt{a}$ and a^n on $\mathbb{R}$ were extended to vectors and they were defined on vectors in a coordinate-wise manner, i.e. for any vector $u, v \in \mathbb{R}^n$ $(u \cdot v)_i = u_i v_i$, $(\sqrt{v})_i = \sqrt{v_i}$ and $(v^n)_i = (v_i)^n$. We should note that unlike other operators the concatenation operator maps to $\mathbb{R}^{2n}$ instead of $\mathbb{R}^n$.

Table 1. Summary of the used vector composition method

Name	Formula	Description
Concatenation	$C_C(u, v) = (u, v)$	gives the concatenation of two vector
Sum	$C_+(u, v) = u + v$	summarizes the vector components
Product	$C_\bullet(u, v) = u \cdot v$	product of the vector components
Quadratic	$C_Q(u, v) = (u - v)^2$	quadratic distance between the components
Hellinger	$C_H(u, v) = (\sqrt{u} - \sqrt{v})^2$	a normalized form of quadratic distance

We use the notation $P_V^C : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+^n$ for the projection function which maps any sequence pair into an n-dimensional vector space. The subscript V of this function denotes the vectorization method for each sequence and the superscript C defines the vector composition method. For example, if the Smith-Waterman (SW) similarity function is used to vectorize a sequence, and the product function $C_\bullet$ is used to construct one vector from two, then the projection function we get will be denoted by $P_\bullet^{SW}(x, y) = C_\bullet(\phi_{SW}(x), \phi_{SW}(y))$.

2.2 Classifier Algorithms

Now the machine learning methods on the set L defined by Eq. 1 should be able to learn an equivalence function. In the following section we will give a short summary of classification methods used in our experiments.

Artificial Neural Networks (ANN) are good at fitting functions and recognizing patterns. In fact there is a proof that a fairly simple neural network can fit any practical function $f(x) = y$. An ANN is composed of interconnected simple elements called neurons and organized in levels. In our study the network structure consisted of one hidden layer with 40 neurons and the output layer consisted of one neuron. In each neuron the log-sigmoid function was used as the transfer function and the Scaled Conjugate Gradient (SCG) algorithm was used for training. The package we applied was the Neural Network Toolbox 5.0 version part of Matlab.

The Support Vector Machine (SVM) gives a decision boundary $f(z) = \langle z, w \rangle + b$ (also called a hyperplane) with the largest margin between the positive and

negative classes. Replacing the inner product by a kernel function leads to a nonlinear decision boundary in the feature space. In our experiments the Radial Basis Function kernel was used and its width parameter σ was the median Euclidean distance from any positive training example to the nearest negative example. During the training the class labels 0 for the miscellaneous pair were replaced by -1. Here the SVM used was the LibSVM [23]. The One-Class SVM also also evaluated to learn equivalence members.

The Logistic Regression (LogReg) is one of the generalized linear models which is used when the response variable is a dichotomous variable (i.e. it can take one of two possible values) and the input variables are continuous. Unlike linear regression, logistic regression does not assume a linear relationship between the input and the output, but it does exploit the advantages of the linear methods. To do this, it uses an $ln\left(\frac{p}{1-p}\right) = \langle w, x\rangle + b$ function, called a link function, and thus it leads to a non-linear relationship, where the p is the probability that $y_i = 1$. In our study the LogReg was part of Weka version 3-4.

The Random Forest (RF) technique is a combination of decision trees such that each tree is grown on a bootstrap sample of the training set. For each node the split is chosen from $m \ll M$ variables (M being the number of dimensions) selected from an independent, identically distributed random variable taken from the feature set. In our experiments 50 trees were used and the number of features m was set to $\log l + 1$, where l is the number of input patterns. The RF was part of Weka Version 3-4.

2.3 Learned Kernel Functions

Here we present a way of constructing a kernel function from the decision boundary $f(z) = \langle z, w\rangle - \rho$ obtained by One-Class SVM. After training a One-Class SVM on the set of equivalence members $L = \{x_i \mid x_i = P(s_i, t_i), 1 = \delta(s_i, t_i)\}$ the w parameter of the decision boundary can be expressed as a weighted linear combination of support vectors, that is $w = \sum_i \alpha_i x_i$, where $\alpha_i > 0$ are the corresponding Lagrangian multiplier. Here the support vectors are equivalence sequence pairs which belong to L. Thus the decision function

$$f(P(s,t)) = \sum\nolimits_i \alpha_i \langle P(s,t), x_i\rangle - \rho \tag{2}$$

can be regarded as a similarity function over sequence pairs. Moreover, omitting the ρ additive constant from f does not change the essence of similarity. In the following lemma we shall examine what kind of projection function would make a kernel function.

Lemma 1. *Let $P^{\phi}_{\bullet}, P^{\phi}_{+}, P^{\phi}_{Q}, P^{\phi}_{H} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n_+$ be a symmetric projection function, where $\phi : \mathcal{S} \to \mathbb{R}^n_+$ is an arbitrary positive feature mapping. Using support vectors x_i and their corresponding Lagrangian multiplier $\alpha_i > 0$ the functions*

$$SVK_{\bullet}(s,t) = \sum\nolimits_i \alpha_i \exp(\sigma\langle P^{\phi}_{\bullet}(s,t), x_i\rangle) \tag{3}$$

$$SVK_{+}(s,t) = \sum\nolimits_i \alpha_i \exp(\sigma\langle P^{\phi}_{+}(s,t), x_i\rangle) \tag{4}$$

$$SVK_{Q}(s,t) = \sum\nolimits_i \alpha_i \exp(-\sigma\langle P^{\phi}_{Q}(s,t), x_i\rangle) \tag{5}$$

$$SVK_{H}(s,t) = \sum\nolimits_i \alpha_i \exp(-\sigma\langle P^{\phi}_{H}(s,t), x_i\rangle) \tag{6}$$

are kernel functions over $\mathcal{S} \times \mathcal{S}$, where $\sigma > 0$. The class of such kernel functions is called the Support Vector Kernel (SVK).

For more detail about kernel functions and their properties, the reader should peruse [25].

Proof. The class of kernel function is closed under direct sum and positive scalar multiplication. Here, it is sufficient to prove that the exponential expressions are kernels. First, let θ a positive valued vector. Then $\langle P^{\phi}_{\bullet}(s,t), \theta\rangle$ is a kernel function because it is a weighted inner product. This follows from the fact that

$$\langle P^{\phi}_{\bullet}(s,t), \theta\rangle = \langle \phi(s) \cdot \phi(t), \theta\rangle = \phi(s)\Theta\phi(t)$$

where Θ is a diagonal matrix whose diagonal elements are taken from θ. Thus $\exp(\sigma\langle P^{\phi}_{\bullet}(s,t), \theta\rangle)$ is a kernel too.

The statement for Eq. 4 follows immediately from

$$\begin{aligned}\exp(\langle P^{\phi}_{+}(s,t), \theta\rangle) &= \exp(\langle \phi(s) + \phi(t), \theta\rangle) = \exp(\langle \phi(s), \theta\rangle + \langle \phi(t), \theta\rangle) = \\ &= \exp(\langle \phi(s), \theta\rangle) \exp(\langle \phi(t), \theta\rangle)\end{aligned}$$

because a function of the form $k(x,y) = f(x)f(y)$ is always a kernel function. And $\langle P^{\phi}_{Q}(s,t), \theta\rangle$ is a quadratic Euclidian metric weighted by θ, that is

$$\begin{aligned}\langle P^{\phi}_{Q}(s,t), \theta\rangle &= \langle \phi^2(s) + \phi^2(t) - 2\phi(s)\phi(t), \theta\rangle \\ &= \langle \phi^2(s), \theta\rangle + \langle \phi^2(t), \theta\rangle - 2\langle \phi^2(s)\phi^2(t), \theta\rangle \\ &= \phi(s)\Theta\phi(s) + \phi(t)\Theta\phi(t) - 2\phi(s)\Theta\phi(t).\end{aligned}$$

Thus $\exp(-\sigma\langle P^{\phi}_{Q}(s,t), \theta\rangle)$ is a kernel. The assertion for $\exp(-\sigma\langle P^{\phi}_{H}(s,t), \theta\rangle)$ can be proved in a similar way.

The training points in L are also vectorized sequence pairs represented by a projection function. Hence the training points are a positive-valued vector, and the support vectors obtained are also positive-valued vectors. This proves our statement. □

3 Experiments

Dataset and performance evaluation. 3PGK is a set of 131 sequences representing the essentially ubiquitous glycolytic enzyme, 3-phosphoglycerate kinase (3PGK,

358 to 505 residues in length) - obtained from 15 archaean, 83 bacterial and 33 eukaryotic species. This dataset was designed to show how an algorithm will generalize to novel, distantly related subtypes of the known protein classes [26]. The dataset is freely available at [27].

The sequences were represented by the so-called pairwise method, where the feature set was the whole train set sequences containing both positive and negative sequences. For the underlying similarity measure we chose two alignment-based sequence comparisons methods, namely the BLAST and Smith-Waterman algorithms. We used version 2.2.4 of the BLAST program with a cutoff score of 25, the Smith-Waterman algorithm was used as implemented in MATLAB. The BLOSUM 62 matrix [28] was used in both cases.

The performance evaluation was carried out by standard receiver operator characteristic (ROC) analysis, which is based on the ranking of the objects to be classified [29]. The analysis was performed by plotting sensitivity vs. 1-specificity at various threshold values, and the resulting curve was integrated to give an "area under the curve" (AUC) value. We should remark here that for a perfect ranking AUC = 1.0 while for a random ranking AUC = 0.5. In our experiments the ROC score for the 3PGK dataset was AUC averaged over 8 tasks.

Equivalence learning. For each classification task, the set of training pairs L defined in Eq. 1 consists of pairs made up from training sequences. The equivalence function δ was calculated via class labels. Here a sequence pair (s, t) is treated as equivalent if s and t belong to the same species and their equivalence score is 1. If they belong to two different species, then their equivalence score is 0. This δ may be regarded as a similarity function and be denoted by D_M where the subscript M stands for a given machine. Thus for an ANN the similarity function obtained is denoted by D_{ANN}. δ was learned on L by several classification methods.

During the evaluation, a test sequence u was paired with each of the train sequences s to check whether they belong to the same group. For the ROC analysis these sequence pairs were ranked by their score obtained via $D_M(u, s)$. As a comparison the sequence pairs were also ranked by their Smith-Waterman, and their BLAST score. The corresponding ROC score for these functions is given in Table 2.

The train pairs and test pairs can be arranged in a matrix M^D whose columns are indexed by train sequences and whose rows are indexed by test sequences. Then an element of $(M^D)_{s,t}$ is a similarity score of (s, t) obtained by a measure D. A heat map representation of a train and test matrix of one of 8 classification tasks is shown in Figure 3 below.

Train set construction. To learn an equivalence function, we randomly selected a small part of the positive and negative pairs from the train sequences. This step is necessary in order to avoid overlearning, to speed up the training and to reduce the training set to a computationally manageable size. In order to select the best number of training pairs we calculated the learned similarity function by varying the number of datasets and repeated the procedure 10 times (Figure 5). We may

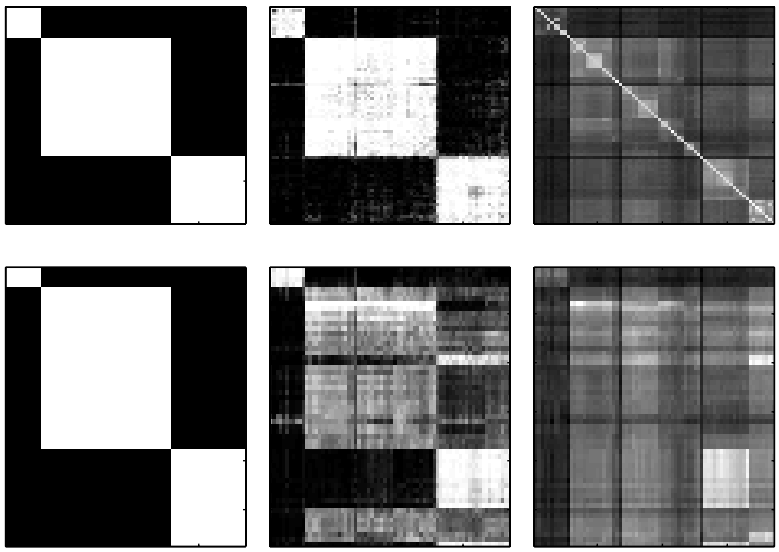

Fig. 3. A heat map representation of train and test matrices from the 3PGK database. The train matrix (above) is an equivalence matrix over train sequences and the test matrix (below) is an equivalence matrix between train and test sequences. The three sequence groups are easily recognizable in the equivalence matrix (a). The equivalence was learnt by RF where the sequence pair was projected by P_+^{SW}. The obtained train and the test matrices are displayed in (b). The Smith-Waterman similarity matrices (c) are also shown for comparison.

Table 2. Evaluation of the equivalence function learning using different vectorization methods.

		D_{SVM}	D_{SVK}	D_{ANN}	D_{LogReg}	D_{RF}
	C_C	0.9146	n.a	0.8600	0.6856	0.8311
SW	C_+	0.9107	0.6308	0.8948	0.7110	0.8446
0.7302[1]	$C_\bullet$	0.9022	0.6508	0.8870	0.6452	0.8516
	C_Q	0.8473	0.7901	0.8135	0.7448	0.8383
	C_H	0.8586	0.7919	0.8429	0.7571	0.8250
	C_C	0.9193	n.a	0.8800	0.6879	0.8605
BLAST	C_+	0.9184	0.6339	0.8906	0.7189	0.8649
0.7372[1]	$C_\bullet$	0.9085	0.6565	0.8839	0.6517	0.8703
	C_Q	0.8561	0.7966	0.8068	0.7530	0.8209
	C_H	0.8587	0.8037	0.8486	0.7617	0.8548

[1]The corresponding ROC score for the similarity method to measure how it can express the equivalence.

conclude here that the standard deviation is generally small and increasing the training points only makes it smaller. We should mention here that a reasonable choice of number of training points depends on the variability of the training

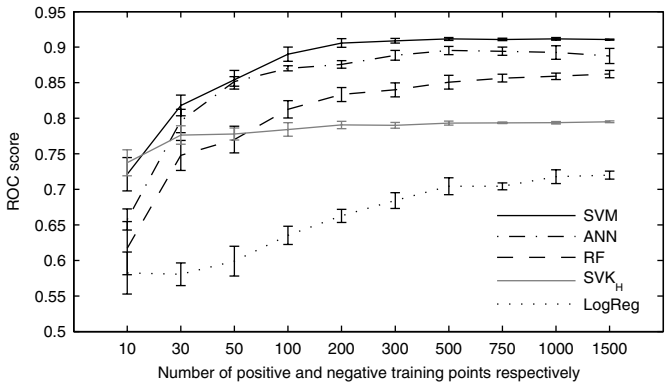

Fig. 4. Dependence of the equivalence learning results on the train set size. Except for the SVK_H case the P_+^{SW} projection method was applied.

set; protein groups in real-life databases are known to be vastly different in the number of members, in average protein size, similarity within group and so on. In our experiments 500 positive and negative training pairs were used for learning, respectively.

In the experiments we dealt only with the situation where the class labels are known for the negative sequences. If the class labels are unavailable, (i.e. it cannot be decided if two negative sequences belong to the same group), there are two potential solutions. First, any (x, y) pair where either or both of them are negative sequences, can be treated as a non-equivalent pair. In this case, the learner learns that a sequence pair belong to the particular species. The second possibility is that only the positive-negative sequence pairs are considered non-equivalent, and negative-negative pairs are removed from the training set. In our study, only the first method gave better results than the second (data not shown).

Iteration. The functions obtained by the machine learning algorithms can be used as an underlying similarity function in the pairwise vectorization approach. This step can be repeated in an iterative fashion. Here, as shown in Figure 5, the results become stable after 3-4 steps. Our empirical test told us that the trained similarity matrices (Figure 5, top) really converge to the ideal equivalence matrix but the test similarity matrix (bottom) kept some of its original mistakes, and during the iteration process these errors became more pronounced.

3.1 Classification Results Via Learned Similarity Functions

These learned similarity functions were evaluated in a protein classification context. A sequence was vectorized by the pairwise vectorization method and the feature set was the whole train sequence set. For the underlying similarity functions D_{ANN}, D_{SVM}, D_{LogReg}, D_{RF} and the original Smith-Waterman were used, while the classification method employed was SVM. The results obtained are listed in the table below.

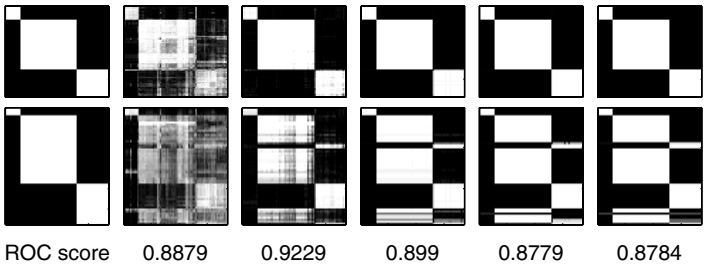

Fig. 5. An iteration of ideal similarity learning by RF with P_{+}^{SW}. The leftmost heat map pair is the target equivalence matrix which was calculated by using class labels.

Table 3. Evaluation of the sequence classification via SVM

		D_{SVM}	D_{ANN}	D_{LogReg}	D_{RF}
	C_C	0.9694	0.9778	0.7709	0.9545
	C_+	0.9749	0.9866	0.7802	0.9700
SVM	$C_\bullet$	0.9759	0.9691	0.8730	0.9712
0.9651[1]	C_Q	0.9641	0.8823	0.9360	0.9434
	C_H	0.9614	0.9242	0.9499	0.9460

[1]The ROC score obtained by SVM classification with SW feature extraction.

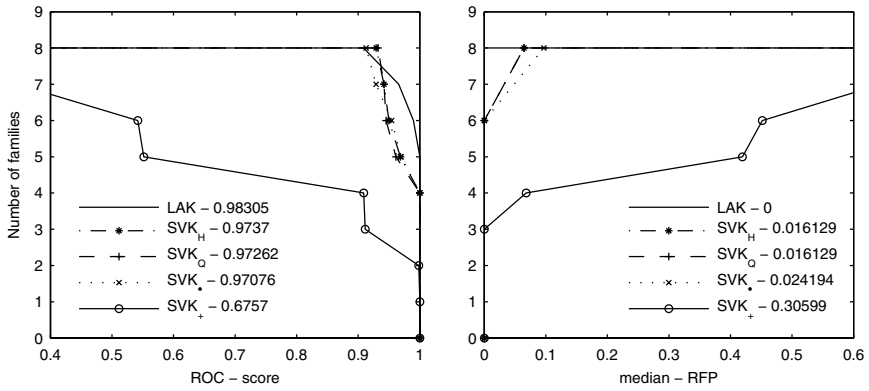

Fig. 6. SVM classification results with kernel functions. Here the BLAST algorithm was used for feature extraction.

Learned kernel results. Experiments for the support vector kernel. The results we got are summarized in Table 3. For a comparison, the Local Alignment Kernel was also evaluated. The program was used with the parameter values suggested by the authors [11].

4 Conclusions

We described a method termed equivalence learning, which applies a two-class classification approach to object-pairs defined within a multi-class scenario. The underlying idea is that instead of classifying objects into their respective classes, we classify object pairs either as equivalent (belonging to the same class) or non-equivalent (belonging to different classes). The method is based on a vectorisation of similarity between the objects. We should note that this is one of the most important steps and the results are sensitive to small changes. We think further methods should be developed to characterize similarity by several values, represented in vector form rather than by just a single value. The application of kernel methods to routine problems in protein classification is apparently hampered by the high dimensionality of vector representations. Equivalence learning is plagued by the same problem, so the reduction of dimensionality may be an important step if equivalence learning is to be applied to real-life protein databases.

The method is more complex than simple machine learning algorithms but its time and storage space requirements are not exceedingly high as compared to these methods, and in some cases it provides a better performance.

Finally we should mention that even though the expression *equivalence learning* may be novel in machine learning, it has already been used in cognitive studies [30]. We hope that this concept will become more popular in machine learning.

Acknowledgements

The work at ICGEB was supported in part by grants from the Ministero dell'Universita e della Ricerca (D.D. 2187, FIRB 2003 (art. 8), "Laboratorio Internazionale di Bioinformatica"). A. Kocsor was supported by the Janos Bolyai fellowship of the Hungarian Academy of Sciences.

References

1. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453 (1970)
2. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197 (1981)
3. Pearson, W.R.: Rapid and sensitive sequence comparison with FASTP and FASTA. Methods Enzymol. 183, 63–98 (1985)
4. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215, 403–410 (1990)
5. Eddy, S.: HMMER Biological sequence analysis using profile hidden Markov models, Version 2.3.2 (2003) `http://hmmer.janelia.org/`
6. Mount, D.W.: Bioinformatics: Sequence and Genome Analysis. 2nd edn. Cold Spring Harbor Laboratory Press (2004)

7. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
8. Lodhi, H., Saunders, C., Cristianini, N., Watkins, C., Shawe-Taylor, J.: String Matching Kernels for Text Classification. Journal of Machine Learning Research 2, 419–444 (2002)
9. Leslie, C., Eskin, E., Weston, J., Noble, W.S.: Mismatch string kernels for SVM protein classification. In: Advances in Neural Information Processing Systems, vol. 15, MIT Press, Cambridge (2003)
10. Leslie, C., Eskin, E., Noble, W.S.: The spectrum kernel: A string kernel for SVM protein classification. In: Proceedings of the Pacific Symposium on Biocomputing (PSB-2002), World Scientific Publishing, Singapore (2002)
11. Vert, J-P., Saigo, H., Akatsu, T.: Local alignment kernels for biological sequences. In: Schölkopf, B., Tsuda, K., Vert, J.-P. (eds.) Kernel methods in Computational Biology, pp. 131–154. MIT Press, Cambridge (2004)
12. Jaakkola, T., Diekhans, M., Haussler, D.: Using the Fisher kernel method to detect remote protein homologies. In: Proc Int. Conf. Intell. Syst. Mol. Biol. pp. 149–158 (1999)
13. Bishop, D.M.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
14. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
15. Rice, J.C.: Logistic regression: An introduction. In: Thompson, B. (ed.) Advances in social science methodology, vol. 3, pp. 191–245. JAI Press, Greenwich, CT (1994)
16. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
17. Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J.: On Kernel Target Alignment. In: Advances in Neural Information Processing Systems, vol. 14, pp. 367–373 (2001)
18. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the Kernel Matrix with Semidefinite Programming. Journal of Machine Learning Research 5, 27–72 (2004)
19. Kwok, T.J., Tsang, I.W.: Learning with idealized Kernels. In: Proc. of the 28. International Confernece on Machine Learning, Washington DC (2003)
20. Liao, L., Noble, W.S.: Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. J. Comput. Biol. 10, 857–868 (2003)
21. 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 Res. 33, 223–225 (2005)
22. Tsuda, K.: Support vector classification with asymmetric kernel function. Pros. ESANN, 183–188 (1999)
23. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
24. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning tools and Techniques with JAVA implementations. Morgan Kaufman, Seattle, Washington (1999)
25. Berg, C., Christensen, J.P.R., Ressel, P.: Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions. Springer, Heidelberg (1984)
26. Kertesz-Farkas, A., Dhir, S., Sonego, P., Pacurar, M., Netoteia, S., Nijveen, H., Leunissen, J., Kocsor, A., Pongor, S.: A comparison of random and supervised cross-validation strategies and benchmark datasets for protein classification (submitted for publication 2007)

27. Sonego, P., Pacurar, M., Dhir, D., Kertész-Farkas, A., Kocsor, A., Gáspári, Z., Leunissen, J.A.M., Pongor, S.: A Protein Classification Benchmark collection for Machine Learning. Nucleid Acids Research
28. Henikoff, S., Henikoff, J.G., Pietrokovski, S.: Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. Bioin-formatics 15, 471–479 (1999)
29. Gribskov, M., Robinson, N.L.: Use of Receiver Operating Characteristic (ROC) analysis to evaluate sequence matching. Comput. Chem. 20, 25–33 (1996)
30. Johns, K.W., Williams, D.A.: Acquired equivalence learning with antecedent and consequent unconditioned stimuli. J. Exp. Psychol. Anim. Behav. Process 24m, 3–14 (1998)