

# Applying fuzzy technologies to Equivalence Learning in Protein Classification

József Dombi\* and Attila Kertész-Farkas\*<sup>1</sup>

\*Department of Informatics, University of Szeged, Árpád tér 2.  
H-6720, Szeged, Hungary

\*Research Group on Artificial Intelligence of the Hungarian  
Academy of Sciences and University of Szeged, Aradi vértanúk  
tere 1, H-6720, Szeged, Hungary

e-mail: \*dombi@inf.u-szeged.hu, \*kfa@inf.u-szeged.hu

## Abstract

When sequencing a new genome, its function and structure may be one of the first important questions. The inferring methods are based on protein sequence similarity methods. However sequence groups are mostly different their parameters, like the number of group members and the intra- and inter- class variability. A method that performs well on one group may perform worse on another and vice versa. Learning of similarity in a supervised manner could provide a general framework to set a similarity function to a specific sequences classes.

Here we describe a novel method which learns a similarity function between proteins by using a binary classifier and pairs of equivalent sequences (belonging to the same class) as positive samples and non-equivalent sequences (belonging to different classes) as negative training samples. For sequence pair representation we propose to use advanced techniques from fuzzy theory, including a sigmoid-type function for normalization and the class of Dombi operators which provide a more robust method. Using some additional constraints, the learned function turns out to be a valid kernel or metric function and we present a new way of learning it along with a new parameter weighting technique. Using a dataset of archaeal, bacterial and eukaryotic 3-phosphoglycerate-kinase sequences (3PGK) and clusters from COG we will evaluate this equivalence learning method from a protein classification point of view. A ROC analysis shows that we get a much more robust and accurate methodology for protein classification when these techniques are applied together.

## 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

---

<sup>1</sup>To whom correspondence should be addressed

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 (Needleman and Wunsch, 1970; Smith and Waterman, 1981) or faster, heuristic algorithms (Pearson, 1990; Altschul *et al.*, 1990). 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 (Eddy, 1998), but there are many other, simpler types of description in use (for a review, see (Mount, 2004)). Discriminative models (such as artificial neural networks and support vector machines) 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 whether 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.

The kernel functions can be regarded as a similarity function which has the additional property of always being positive semi-definite, which is a simple way to extend the well-tried linear applications to a non-linear model while preserving their computational advantages (Shawe-Taylor and Cristianini, 2004). Over the past decade, many kernels have been developed for sequences such as the String kernel (Lodhi *et al.*, 2002), Mismatch kernel (Leslie *et al.*, 2004), Spectrum kernel (Leslie *et al.*, 2002), Local Alignment Kernel (Vert *et al.*, 2004) and the Fisher kernel (Jaakkola *et al.*, 1999). For a good review of these applications, see (Shawe-Taylor and Cristianini, 2004).

The *Distance Metric Learning* (DML) approach seeks to learning a Mahalanobis distance

$$\|x_i - x_j\|_M^2 = (x_i - x_j)'M(x_i - x_j), \quad (1)$$

for  $n$ -dimensional real valued vectors  $x_i, x_j$ , that is the learning of positive a semidefinite matrix  $M$ , such that it reduces the intra-

class variability and increases the inter-class variability. (Xing *et al.*, 2003) proposed a method for learning a Mahalanobis distance, but the optimization involves diagonalization and eigen decomposition methods, hence they gave an iterative procedure using gradient descent and projection techniques.

The ideal similarity matrix  $Y$  was introduced first in (Cristianini *et al.*, 2001) and can be considered as an "oracle" that tells us which object pairs belong to the same class. Such an ideal similarity matrix would be perfect for classification, but it cannot be computed for the unlabelled part of data. The idea in (Tsang and Kwok, 2003; Kwok and Tsang, 2003) was to optimize  $M$  such that the kernel matrix

$$K_{ij} = x_i' M x_j \quad (2)$$

is geared to the ideal matrix  $Y$  on the training sample relying on it will give a better alignment on the unlabeled data. This alignment of two matrices was measured by the Frobenius product of matrices. (Davis *et al.*, 2007) presents an information-theoretic approach where the alignment to a certain matrix is measured by the Kullback-Leibler divergence. (Weinberger *et al.*, 2006) propose a distance metric learning for kNN classification via optimizing k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. However this learned Mahalanobis distance is still a linear feature weighting method for vectors.

The kernel learning differs from the above mentioned former DML. (Lanckriet *et al.*, 2004) presented a learning of general kernel matrices such that it maximizes the margin of the separation hyperplane on the training data over the cone of a positive semidefinite matrix by semidefinite programming. The *Multiple Kernel Learning* (MKL) approach is dedicated to learn an optimal convex combination  $\sum_i \mu_i K_i$  of some initial kernels  $K_i$  for better classification. (Lanckriet *et al.*, 2004) presented a quadratically constrained quadratic programming task that determines the optimal weights  $\mu$  for this combination. (Zien and Ong, 2007) optimize linear combination of kernels in a direct way for SVM classification. (Amari and Wu, 1999) propose a method to modifying a kernel function using Riemannian geometrical structure induced by the kernel function enlarging the spatial resolution around the separation boundary surface such that the separability between classes is increased.

In bioinformatics metric learning is employed for the reconstruction of biological networks e.g. predicting interactions between genes. (Vert *et al.*, 2007) proposed a new kernel function for bio-

logical network inference by tuning the results of (Tsang and Kwok, 2003; Ben-Hur and Noble, 2005). (Vert and Yamanishi, 2004) proposed a method for predicting the absence or presence of an edge in a biological network using the Laplacian matrix induced by the graph. (Yamanishi *et al.*, 2004) employed a variant of kernel canonical correlation analysis to identify correlations between heterogeneous datasets, especially that between a protein network and other genomic attributes. (Tsuruoka *et al.*, 2007) proposed learning a string similarity function for gene names using Logistic Regression via a linear weighted combination of a set of standard string similarity methods. But to the best of our knowledge there has been no study of similarity learning for protein classification.

Here we present a new approach for learning the ideal similarity matrix in a different way from the former methods. In protein classification the goal is to place a protein into its unique class, that is

$$F(s) = y, \tag{3}$$

where  $y$  stands for the class label. This classification of proteins naturally determines the following equivalence relation

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

which is reflexive, symmetric and transitive. A pair of sequences is called equivalent when both of them belong to the same sequence group and it is called non-equivalent when they do not; moreover the set of equivalent (resp. non-equivalent) pairs is called the positive (resp. negative) class. The learning of this function  $\delta$  essentially becomes a two-class classification schemes and hence, two-class classification techniques are applicable such as Support Vector Machines (SVMs). This approach is called by *equivalence learning*.

For example, let us consider a database of objects and a similarity measure computed between each pair of objects. This arrangement 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 by a symmetrical matrix in which the cells store pairwise comparison values between the objects. Figure 1a shows a hypothetical database of 8 objects. We can vaguely recognize two groups where 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 (C) 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.

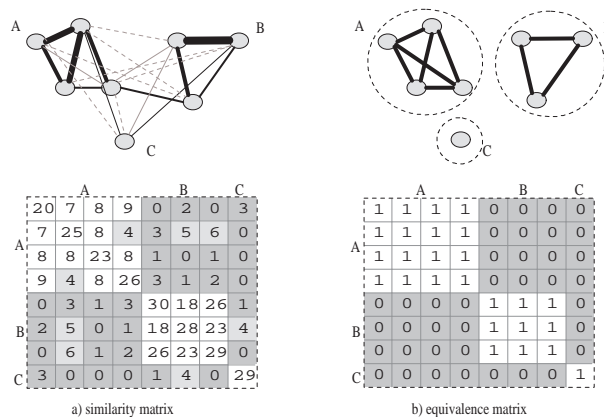


Figure 1: The Principle of Equivalence Learning. A similarity matrix (left) can be determined by an all-vs.-all comparison of a database using algorithms such as BLAST and Smith-Waterman. The equivalence matrix (right) is a representation of groups (A,B) and an outlier (C) is identified by human experts.

A key issue here is to decide how we should represent the similarity between proteins so that we can efficiently predict an equivalence. A simple numerical value is not sufficient so, instead, we will use a vectorial representation of the edges. Hence we will need a projection method

$$P : S \times S \rightarrow \mathbb{R}^n, \quad (4)$$

which captures the similarity between a sequence pair from a different point of view and stores them in vector form. In Section 2.1 we will describe a new vectorization method which realizes this projection mapping  $P$  using an advanced techniques taken from fuzzy theory. Then the equivalence learning task can be reduced to the learning of the two-class classification problem

$$F(P(s, t)) \rightarrow \{0, 1\}, \quad (5)$$

which can be carried out via a typical machine learning algorithm like ANNs. Instead of the predicted class labels using the score obtained from the learned model, it can be viewed as a learned

similarity. In Section 2.2 we will present the kind of properties required to have a valid learned metric or kernel function and we will present a new way of learning them. In Section 3 we will describe the datasets and methods used in our expression then in Section 4 we will present our results along with a brief discussion. Finally, in Section 5 we will provide some conclusions and suggestions for future research.

## 2 Algorithms

### 2.1 Vectorization step

Now we will show how to define projection functions which map any sequence pair into a fixed-length vector of real numbers, that is  $P : S \times S \rightarrow \mathbb{R}^n$ . First, we will define a method to represent sequence in vector form.

Let us consider a set of sequences  $\{f_1, f_2, \dots, f_n\} \subseteq S$  as a feature set in a fixed ordering and let  $sf$  be an arbitrary similarity function. For a sequence  $s \in S$  let the corresponding vector  $w$  be a ranking vector which shows how  $s$  orders the members of the feature set via the given similarity function  $sf$ . Thus  $w_i$  (the  $i$ th feature of  $w$ ) means the order index of the  $i$ th feature sequence in the ranking with respect to  $s$  and  $sf$ .

Then for each component of the vector  $w$  the function  $\kappa$  is used for normalization, which is defined by the following way:

$$\kappa_{\lambda, \mu}(x) = \frac{1}{1 + \left(\frac{\mu}{n-\mu} \cdot \frac{n-x}{x}\right)^{-\lambda}}. \quad (6)$$

$\kappa$  was first introduced in (Dombi, 1988). This function is very useful in fuzzy theory for modelling the membership function, but it can also be used as a probability distribution function. It is useful for normalizing of variables whose domain is bounded.

This is an increasingly monotone function and maps order numbers from  $[1, n] \subseteq \mathbb{R}$  onto the interval  $[0, 1]$ . The shape of this function  $\kappa$  is similar to a sigmoid function but the parameter  $\mu \in (1, n)$  regulates the location of the inflection point while the parameter  $\lambda$  regulates its gradient at the point  $\mu$ , moreover  $\kappa_{\lambda, \mu}(\mu) = 0.5$ . The protein sequence vectorization method we get will be denoted by  $\phi_{sf}^{\kappa} : S \rightarrow \mathbb{R}^n$ , where  $sf$  is a similarity function used for ranking and each component of the vector we get is normalized by  $\kappa$ . It should be mentioned here that each vector component lies between 0 and 1 and that each vector has the same (but not unit) length.

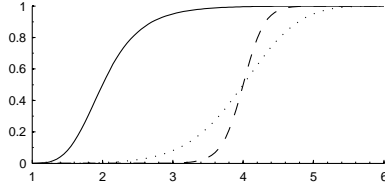


Figure 2:  $\kappa$  functions with different parameter values. Here  $n = 6$ ,  $(\mu_1 = 2, \lambda_2 = -3)$ ,  $(\mu_2 = 4, \lambda_2 = -3)$ ,  $(\mu_3 = 4, \lambda_3 = -8)$ , dotted).

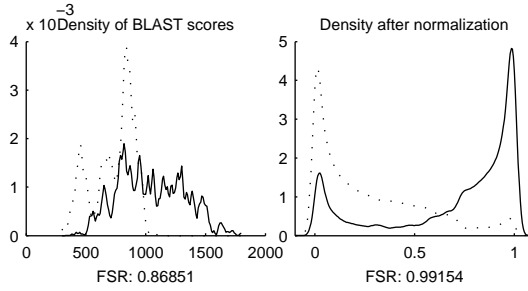


Figure 3: The density of scores for equivalent (solid line) and nonequivalent (dotted line) pairs, which are obtained by BLAST (left) and after normalization by  $\kappa$  (right). Below each figure the Fisher Separation Ratio (FSR) is shown.

The next step is to form a vector from two vectors and we will do this component-wise. The widely-used Dombi operator class (Dombi, 1982) can be used to describe a multivalued logic function like

$$o^\alpha(x, y) = \frac{1}{1 + \left( \left( \frac{1-x}{x} \right)^\alpha + \left( \frac{1-y}{y} \right)^\alpha \right)^{1/\alpha}}, \quad (7)$$

If  $\alpha > 0$ , then  $o$  is a conjunctive operator, and if  $\alpha < 0$ , then  $o$  is a disjunctive operator. Most common case  $\alpha = \pm 1$ . The equivalence relation in logic is defined by

$$x \equiv y = (\bar{x} \vee y) \wedge (x \vee \bar{y}), \quad (8)$$

where  $\bar{x}$  is the negation and  $\bar{x} = 1 - x$ . Using the Dombi operator class we get

$$e(x, y) = xy + (1 - x)(1 - y). \quad (9)$$

Two important requirements are valid in classical logic. These are  $x \equiv x = 1$  and  $x \equiv \bar{x} = 0$ . Most papers concentrate on the

first case and wish to find an equivalence in  $e(x, x) = 1$  and ignore  $e(x, n(x)) = 0$ . In many-valued logic both conditions cannot be simultaneously valid. If we interpretate the  $x$  values as a certainty then if  $x = \frac{1}{2}$  and  $y = \frac{1}{2}$ , the result of equivalence is also  $\frac{1}{2}$ , and the resulting  $e(x, y)$  has the following properties:

$$e(0, 0) = e(1, 1) = 1, \quad e(0, 1) = e(1, 0) = 0 \quad \text{and} \quad e\left(\frac{1}{2}, \frac{1}{2}\right) = \frac{1}{2}. \quad (10)$$

The *mean conjunctive operator* of the Dombi operator class is

$$c(x, y) = \frac{1}{1 + \frac{1}{2} \left( \frac{1-x}{x} + \frac{1-y}{y} \right)} \quad (11)$$

This operator class is used with the sigmoid function, that is

$$x = \frac{1}{1 + e^{-\lambda(u-v)}} \quad y = \frac{1}{1 + e^{-\lambda(v-u)}}$$

$x$  (and  $y$ ) can be interpreted as the degree of the preference of  $u$  over  $v$ . The exponential composition method can be given that the (degree of  $u$  greater then  $v$ ) and (degree of  $v$  greater then  $u$ ), where for and we use 11. For  $\lambda = 1$  we get

$$c_C(u, v) = \frac{1}{1 + \frac{1}{2} (e^{(u-v)} + e^{(v-u)})}. \quad (12)$$

Table 1 summarizes the methods which were used in our experiments. Here, for ease of notation, the operators 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  $(v^n)_i = (v_i)^n$ .

Table 1: Summary of the vector composition method we applied

Name	Formula
Sum	$C_+(u, v) = u + v$
Product	$C_\bullet(u, v) = u \cdot v$
Quadratic	$C_Q(u, v) = (u - v)^2$
Hellinger	$C_H(u, v) = (\sqrt{u} - \sqrt{v})^2$
Dombi	$C_D(u, v) = u \cdot v + (1 - u) \cdot (1 - v)$
Conjunct.	$C_C(u, v) = 1 / (1 + (e^{(u-v)} + e^{(v-u)}) / 2)$

Now we will use the notation  $P_C^V : S \times S \rightarrow \mathbb{R}_+^n$  to stand for a projection function which maps any sequence pair into an  $n$ -dimensional vector space. The superscript  $V$  denotes the vectorization method

for both sequences and the subscript  $C$  defines the vector composition method. For example, if the Smith-Waterman ( $SW$ ) similarity method is used as a ranking function to vectorize a sequence with normalization  $\kappa$ , and composition method  $C_\bullet$  is used. Afterwards the projection function we get will be denoted by

$$P_\bullet^{SW}(x, y) = C_\bullet(\phi_{SW}^\kappa(x), \phi_{SW}^\kappa(y)). \quad (13)$$

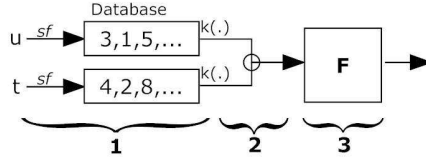


Figure 4: A diagram of the equivalence learning task. The first step is the vectorization of sequences  $u$  and  $t$  carried out by ranking of the database via a sequence similarity measure  $sf$ . Then the non-linear normalization function  $k$  is employed. The second step is the combination of the two vectors with a particular method from Table 1. Finally, the third step is the application of an learned model that yields a similarity score for an input sequence pair.

## 2.2 Learned kernel functions

Here we define a class of kernel functions and we present a new way to learn it. Let  $\theta$  be a positive-valued  $n$ -dimensional vector and let

$$P_C^\phi : S \times S \rightarrow \mathbb{R}_+^n \quad (14)$$

be a symmetric projection method where  $\phi$  is an arbitrary positive-valued vectorization method. The following functions  $\langle P_D^\phi(s, t), \theta \rangle$ ,  $\langle P_\bullet^\phi(s, t), \theta \rangle$ ,  $\langle -P_Q^\phi(s, t), \theta \rangle$ ,  $\langle -P_H^\phi(s, t), \theta \rangle$  and  $\exp(\sigma \langle P_+^\phi(s, t), \theta \rangle)$  are valid kernels, where  $-P$  is the multiplication of  $P$  by the scalar  $-1$ . The proof is straightforward but not obvious because in the inner product the two variables  $s, t$  vary in the first argument, while the second argument is constant. The proof can be found in Lemma 1 of (Kertész-Farkas *et al.*, 2007b). We note that  $\langle P_Q^\phi(s, t), \theta \rangle$  and  $\langle P_H^\phi(s, t), \theta \rangle$  are both negative definite and vanish when  $s = t$ , thus they are valid metric functions and obey the triangle inequality. The SVM approach provides a decision boundary  $f(z) = \langle w, z \rangle + b$  between two classes such that the margin associated with  $w$  is a

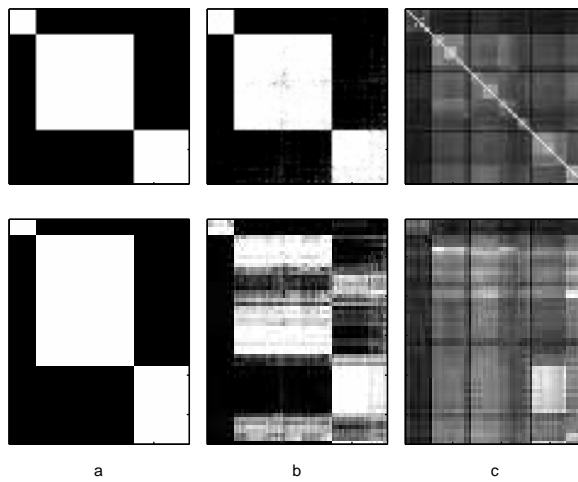


Figure 5: A heat map representation of the equivalence matrix(a) and equivalence matrix learned by RF(b) with the  $C_D$  composition method on one of the classification tasks of 3PGK. The train matrices (above) were calculated in a pairwise manner between the train sequences, and the test matrices (below) were calculated between the train and test sequences in a pairwise manner as well. The Smith-Waterman similarity matrices (c) are shown here for comparison.

maximum (Vapnik, 1999). It is well-known that the norm vector  $w$  can be expressed as a weighted linear combination of support vectors  $x_i$ ; that is  $w = \sum_i \alpha_i x_i$ , where  $\alpha_i$  is the so-called Lagrangian multiplier corresponding to the support vector  $x_i$  and its sign corresponds to the class of support vectors i.e. it is negative (resp. positive) if  $x_i$  corresponds to the negative (resp. positive) class. Thus the decision boundary we get can be written by

$$f(z) = \sum_i \alpha_i \langle z, x_i \rangle + b. \quad (15)$$

The Gaussian RBF kernel

$$rbf(z, x_i) = \exp(-\sigma \|z - x_i\|) \quad (16)$$

can be interpreted as a "hyper ball" around a support vector  $x_i$ , where  $\sigma$  regulates the radius, and it measures how close a sample  $z$  is to the support vector  $x_i$ . Replacing the inner product by the Gaussian RBF kernel in  $f(z)$ , the decision boundary can then be viewed as a weighted sum of how well the sample  $z$  corresponds to each support vector. During the SVM training the parameters of this function are learned in such a way that it separates the two class with the largest, possibly non-linear margin. Using the former inner products in the SVM's decision boundary, we get the following functions:

$$SVK_P(s, t) = \sum_i \alpha_i \exp(\sigma \langle P(s, t), x_i \rangle) \quad (17)$$

over  $S \times S$ , where  $\sigma > 0$  and  $P$  stands for  $P_D^\phi$ ,  $P_\bullet^\phi$ ,  $P_+^\phi$ ,  $-P_Q^\phi$  and  $-P_H^\phi$ , respectively. The  $x_i$ s are called support vectors and they are either a vectorized equivalent pair or non-equivalent pair. The condition for (17) to be a valid kernel function is that the  $\alpha_i$  coefficients have to be positive-valued because the class of kernel function is closed under the direct sum and positive scalar multiplication. The class of such kernel functions is called the *Support Vector Kernel (SVK)*. To learn this sort of function the One-Class SVM can be used as in (Kertész-Farkas *et al.*, 2007b) but only one of the classes can be used otherwise a negative Lagrangian multiplier would be occurred and then the obtained function won't be kernel anymore. Now we will present a different way where both the equivalence and the non-equivalence sequences can be applied. Let  $L = \{(s_i, t_i) \mid i = 1 \dots l\}$  be the training set containing both equivalent and non-equivalent pairs, let a randomly chosen small part of  $L$  be the set of support

vectors denoted by

$$SV = \{x_j = P(s_{i_j}, t_{i_j}) \mid j = 1 \dots k \leq l, (s_{i_j}, t_{i_j}) \in L\} \quad (18)$$

and let

$$A_{ij} = \exp(\sigma \langle P(s_i, t_i), x_j \rangle) \quad (19)$$

where  $P$  and  $\sigma$  represent one of the SVK types. Then the  $\alpha_i$  parameters in the SVK can be learned by solving the following linear equation system with non-negative Least-Square optimization:

$$\begin{aligned} A_{11}\alpha_1 + A_{12}\alpha_2 + \dots + A_{1k}\alpha_k &= \delta(s_1, t_1) \\ A_{21}\alpha_1 + A_{22}\alpha_2 + \dots + A_{2k}\alpha_k &= \delta(s_2, t_2) \\ &\vdots \\ A_{l1}\alpha_1 + A_{l2}\alpha_2 + \dots + A_{lk}\alpha_k &= \delta(s_l, t_l) \end{aligned}$$

We should mention that if the number of equations equals the number of variables  $\alpha_i$ , this system may become numerically unstable and hard to solve. In our experiments we used half of the training set as support vector data.

*On the parameter  $\sigma$ .* In (17) the parameter  $\sigma$  that we used was a feature weighting technique, that is

$$\langle P(s, t), \theta \rangle_\Lambda = P(s, t)\Lambda\theta \quad (20)$$

, where  $\Lambda$  is a positive-valued diagonal matrix whose elements were obtained by Fisher Linear Discriminant (FLD) analysis (Duda *et al.*, 2000). The vector

$$w = S^{-1}(m_1 - m_2) \quad (21)$$

, where  $S$  is the within scatter matrix and  $m_i$  the mean vector of the class  $i$ , got by FLD was normalized to the unit length and the absolute value of each component was used to form the diagonal matrix  $\Lambda$ . Then the  $i$ th component denotes the magnitude the  $i$ th dimension in the separation.

### 3 Materials and Methods

#### 3.1 Dataset and comparison methods

In order to evaluate our method we chose two datasets. The first is a set of 131 sequences representing the essentially ubiquitous glycolytic enzyme, 3-phosphoglycerate kinase (3PGK) protein that consist of 15 archaean, 83 bacterial and 33 eukaryotic species, from 358

to 505 residues in length. This set seems to be small but it is quite difficult to handle because the groups are vastly different in the number of members and the average similarity within and between groups with a particular sequence similarity method.

The subdivision of this dataset into train and test sets was performed by a supervised cross-validation technique which was designed to show how a learning algorithm will generalize to novel, distantly related subtypes of the known protein classes (Kertész-Farkas *et al.*, 2007a). In each classification task the positive examples were taken from a given kingdom and one of the phyla (with at least five members) was the test set while the remaining phyla of the kingdom were used as the training set. In addition, the negative set contained members of the other two kingdoms subdivided in such a way that members of the phylum could be either test or train. In this way 10 classification tasks were obtained and they are freely available at (Sonogo *et al.*, 2007).

For the second dataset we selected some groups from the COG database of functionally annotated orthologous sequence clusters Tatusov *et al.* (2003). Of the over 5665 COGs we selected 42 resulting 3689 sequences altogether. For a given COG group, the positive set is the archaea, while the negative set is the rest of the sequences. The subdivision into train and test set was performed by random selection (50%-50%). This dataset and details about classification tasks are available at our supplementary data.

We chose the alignment-based sequence comparisons methods BLAST algorithms to measure the similarity and to rank the sequences in the vectorization step. We used version 2.2.4 of the BLAST with BLOSUM62 matrix (Henikoff *et al.*, 1999) and default gap opening and extension. In our experiments other sequence similarity methods were carried out for the comparison. We evaluated the Smith-Waterman (SW) and Local Alignment Kernel (LAK) with the BLOSUM62 substitution matrix and with default gap opening and extension parameters. In LAK the  $\lambda$  parameter value was suggested by the authors (Vert *et al.*, 2004). Compression-Based Distance measures (CBDs) (Cilibrasi and Vitanyi, 2005) were calculated via the following formula

$$CBD(s, t) = \frac{(C(st) - \min \{C(s), C(t)\})}{\max \{C(s), C(t)\}}, \quad (22)$$

where  $s$  and  $t$  are sequences to be compared and  $C(\cdot)$  denotes the length of a compressed string, compressed by a particular compressor  $C$ , such as the LZW or the PPMZ algorithm (Kocsor *et al.*,

2006). In this study the LZW algorithm was implemented in  $C$  while the PPMZ algorithm was downloaded from Charles Bloom's homepage (Bloom, 1998).

### 3.2 Classifier algorithms

Here we will give a short summary of the classification methods used in our experiments.

Artificial Neural Networks (ANN) are good at fitting functions and recognizing patterns (Bishop, 1996). 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 (Vapnik, 1999). In our experiments the Radial Basis Function kernel was used and its width parameter  $\sigma$  was used in the same way as in SVK. The SVM we used here was the LibSVM (Chang and Lin, 2001).

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 (Rice, 1994). 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 (Breiman, 2001). 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.

The Nearest Neighbour (1NN) (Duda *et al.*, 2000) approach also were applied such that test samples were ranked by the largest similarity score (that is, obtained by similarity measures like BLAST or learned equivalence) to the positive training samples.

### 3.3 Performance evaluation

A performance evaluation was carried out by standard receiver operator characteristic (ROC) analysis, which is based on the ranking of the test samples via its score, got by a particular learned model (Gribnikov and Robinson, 1996). The analysis was performed by plotting sensitivity vs. 1-specificity at various decision threshold values, and the resulting curve was integrated to give an "area under the curve" (AUC) value. In general we expect that a score obtained from a learned model should be higher for an equivalent sequence pair and lower for a non-equivalent pair. Furthermore, an ROC analysis is equal to the Wilcoxon non-parametric statistics and thus, in equivalence learning, the probabilistic meaning of the AUC value is the probability of a (similarity) score we get for an equivalence pair is greater than or equal to the score for an non-equivalence (Hanley and Mcneil, 1982). We should remark here that for a perfect ranking  $AUC = 1.0$ , while for a random ranking  $AUC = 0.5$ . The AUC score presented in the tables is the average of AUCs over the classification tasks.

The Fisher Separation Ratio (FSR) (Duda *et al.*, 2000) was also evaluated in order to see how well the classes are separated from each other. It is defined by  $FSR = ((\mu_1 - \mu_2)^2) / (\sigma_1^2 + \sigma_2^2)$ , where  $\mu_i$  is the centre and  $\sigma_i$  is the standard deviation of class  $i$  ( $i = 1, 2$ ).

## 4 Results and discussions

In each classification task the train set of the equivalence and non-equivalence pairs was a small, randomly selected part of the full 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 because the training pairs grows quadratically w.r.t. the number of train sequences. In our experiments 500 equivalence and non-equivalence training pairs were used respectively, each test was repeated 10 times and the results given in tables are the average values on them. We may conclude here that the standard deviation is generally small, and increasing the training points only makes it smaller (Kertész-Farkas *et al.*, 2007b). We should also mention here that a reasonable choice of number of training points depends on the variability of the training set; that is the number of protein groups, number of group members, average protein size, similarity within and between groups and so on.

In the vectorization step the ranking of the feature set was carried

out by using the BLAST method. For the parameter setting in the normalization function  $\kappa_{\lambda,\mu}$  in our experiments we found that the best results were obtained when  $\mu$  was set to the ratio of the non-equivalence pairs and  $\lambda$  was set to the tangent at the point  $\mu$ , that is  $\lambda = 1/(\mu - 1)$  (data not shown). Finally, a vector composition method was applied (Table 1). In order to see how well the vectors of the equivalence and the non-equivalence sequence pairs were separated from each other the FSR method was applied. Also to see how normalization helps the separation, the FSR was also evaluated on the original data that is, in the vectorization step just the similarity score was used instead of the normalized order number. The results are shown in Table 2.

Table 2: The Fisher Separation Ratio (FSR) with different composition methods applied on the original and on the normalized training data of 3PGK.

FSR	$C_+$	$C_\bullet$	$C_Q$	$C_H$	$C_D$	$C_C$
original	0.055	0.073	0.087	0.136	n.a. <sup>1</sup>	0.034
normalized	0.067	0.136	0.391	0.35	<u>0.392</u>	0.146

<sup>1</sup>The Dombi operator ( $C_D$ ) is only defined on in the  $[0,1]$  interval. The largest value is underlined.

Overall we may conclude that the normalization with  $\kappa$  makes the classes more separated and it makes equivalence learning more robust and less sensitive to different composition and learning methods, as well as yielding a good performance in general (data not shown).

Table 3: The AUC results of equivalence learning using different composition and learning methods.

	$C_+$	$C_\bullet$	$C_Q$	$C_H$	$C_D$	$C_E$	<i>avg</i>
SVM	0.832	0.781	0.873	0.848	0.857	0.861	<i>0.842</i>
ANN	0.797	0.804	0.873	0.860	0.873	0.858	<i>0.844</i>
LogReg	0.721	0.781	0.860	0.828	0.875	0.845	<i>0.818</i>
RF	0.847	0.841	0.880	0.874	0.878	<u>0.905</u>	<u><i>0.871</i></u>
nnLS <sup>1</sup>	0.667	0.629	0.880	0.857	0.864	n.a.	<u><i>0.779</i></u>
1SVM <sup>1</sup>	0.680	0.692	0.790	0.766	0.863	n.a.	<i>0.737</i>
<i>avg</i>	<u><i>0.760</i></u>	<u><i>0.765</i></u>	<u><i>0.844</i></u>	<u><i>0.824</i></u>	<u><i>0.868</i></u>	<u><i>0.867</i></u>	<u><i>0.815</i></u>

The AUC value of the BLAST similarity method is 0.737. The largest values is underlined here.<sup>1</sup>Learning of SVK.

On the 3PGK dataset the AUC values for the results of equivalence learning with different learning and composition methods are shown in Table 3. The AUC value based on the original BLAST

similarity matrix is 0.737 and it rose to 0.905 with RF and the  $C_C$  composition method. On average it rose to 0.815. In general the composition methods  $C_\bullet$  and  $C_+$  do not perform as well as the others, and for equivalence learning the LogReg method performed worse than the others. On average we obtained the best AUC results with the RF learner and  $C_D$  composition method. On the COG database we got similar results and trends (data not shown).

#### 4.1 Classification results with learned equivalence

Here we evaluated equivalence learning in the context of protein classification. In the case of 1NN the classification was performed by simply the score obtained by EL. With the other vector-based learning methods (SVM, RF, ANN and LogReg), the  $i$ th vector component for a sequence  $s$  was the learned equivalence scores between  $s$  and the  $i$ th training sequence. To evaluate the classification performance we used ROC analysis and the ranking variable was the score obtained by the given learned model.

The results of the ROC analysis are shown in Table 4. These results suggest that the equivalence learning makes the classification easier, that is, better classification results can be achieved with simpler method. For example, the AUC of the 1NN improved from 0.863 to 0.964 with the equivalence matrix learned by RF and Dombi operator on the dataset 3PGK. The Table 5 presents an overall comparison of several similarity method with several classification methods. The best results were obtained with EL in almost all cases.

Table 4: The AUC results for protein classification on 3PGK.

CM <sup>1</sup>	EL <sup>2</sup>	$C_+$	$C_\bullet$	$C_Q$	$C_H$	$C_D$	$C_C$
1NN	SVM	0.873	0.910	0.946	0.940	<u>0.956</u>	0.899
(0.863)	RF	0.955	0.949	0.960	0.953	<u>0.964</u>	0.937
RF	SVM	0.918	0.910	0.929	<u>0.944</u>	0.923	0.934
(0.885)	RF	0.944	0.939	0.948	0.941	0.948	<u>0.952</u>
SVM	SVM	0.946	0.957	0.956	<u>0.959</u>	0.944	0.803
(0.953)	RF	0.957	0.953	0.964	0.962	<u>0.966</u>	0.962

<sup>1</sup>The classification method applied. <sup>2</sup>The learning method was used to learn the equivalence. In each row the largest value is underlined.

The classification with 1NN on SVKs gives the best AUC results. The SVKs with SVM perform less well than the LAK but we should remark the time complexity of LAK is quadratic and for a sequence pair it computes more operations than the SW method.

Table 5: The overall protein classification results with various similarity and classification methods on 3PGK and COG.

Methods	1NN <sub>p</sub>	SVM	RF	ANN	LogReg	avg
3PGK						
BLAST	0.863	0.953	0.885	0.958	0.953	<i>0.922</i>
SW	0.861	0.953	0.866	0.955	0.948	<i>0.916</i>
LA	0.859	0.966	0.879	0.956	<u>0.959</u>	<i>0.923</i>
LZW	0.783	0.924	0.846	0.928	0.915	<i>0.879</i>
PPMZ	0.812	0.948	0.915	0.959	0.940	<i>0.914</i>
EL <sup>1</sup>	<u>0.964</u>	<u>0.966</u>	<u>0.948</u>	<u>0.963</u>	0.927	<u>0.953</u>
COG						
BLAST	0.892	0.968	0.937	0.968	0.960	<i>0.945</i>
SW	0.762	0.915	0.892	0.961	0.874	<i>0.881</i>
LAK	0.889	0.972	0.931	0.973	<u>0.966</u>	<i>0.946</i>
LZW	0.843	0.941	0.883	0.924	0.920	<i>0.902</i>
PPMZ	0.873	0.955	0.910	0.937	0.942	<i>0.923</i>
EL <sup>1</sup>	<u>0.956</u>	<u>0.978</u>	<u>0.956</u>	<u>0.974</u>	0.956	<u>0.964</u>

<sup>1</sup>Similarity we got by equivalence learning with RF and composition method  $C_D$ . In each column the largest value is underlined.

Table 6: The AUC results for protein classification with LAK and SVK learned by nnLS and 1SVM.

	3PGK			COG		
	nnLS	1SVM	LAK	nnLS	1SVM	LAK
1NN	0.975	0.972	0.860	0.891	0.901	0.888
SVM	0.963	0.930	0.966	0.957	0.953	0.972

## 5 Conclusions

Equivalence learning provides a two-class classification approach for object-pairs defined within a multi-class scenario where the underlying idea is not to classify objects into their respective classes, but rather classify them as equivalent (belonging to the same class) or non-equivalent (belonging to different classes). The method is based on a spectrum of the similarity between the objects represented in vector form. Here we used techniques from fuzzy theory like the normalization function  $\kappa$  and Dombi operator class that make the equivalence learning more robust. We hope this technique will also prove more popular in sequence vectorization in other fields of bioinformatics. The similarity method used during ranking represents the biological knowledge and any special method can be used for a specific sequence group. What is more SVK provides a way of constructing a valid kernel function from any similarity function.

Equivalence learning could be applied as a postprocessing method after the similarity between the query and the database have been calculated in a pairwise manner. The time and space requirements are dwarfed by the cost of the similarity method, but also significantly better results can be obtained. We think that this is because not just the similarity between the query and database used, but the relation to the equivalent and the non-equivalent sequences is used, hence additional information could be exploited in the classification.

## References

- Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J Mol Biol*, **215**(3), 403–410.
- Amari, S. and Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, **12**(6), 783–789.
- Ben-Hur, A. and Noble, W. S. (2005). Kernel methods for predicting protein–protein interactions. *Bioinformatics*, **21**(1), 38–46.
- Bishop, C. M. (1996). *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK.
- Bloom, C. (1998). Solving the problem of context modelling.
- Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.
- Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*.
- Cilibrasi, R. and Vitanyi, P. M. B. (2005). Clustering by compression. *IEEE Transactions on Information Theory*, **51**(4), 1523–1545.

- Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. (2001). On kernel-target alignment. In *Advances in Neural Information Processing Systems 14 - Proceedings of NIPS 2001, December 3-8, 2001, Vancouver, Canada*, pages 367–373.
- Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 209–216, New York, NY, USA. ACM.
- Dombi, J. (1982). A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, **8**, 149–163.
- Dombi, J. (1988). Membership function as an evaluation. *Fuzzy Sets and Systems*, **35**.
- Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley Interscience, 2 edition.
- Eddy, S. (1998). Hmmer user’s guide: biological sequence analysis using prole hidden markov models.
- Gribnikov, M. and Robinson, N. (1996). Use of receiver operating characteristic (roc) analysis to evaluate sequence matching.
- Hanley, J. A. and Mcneil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, **143**(1), 29–36.
- Henikoff, S., Henikoff, J. G., and Pietrokovski, S. (1999). Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, **15**(6), 471–479.
- Jaakkola, T., Diekhaus, M., and Haussler, D. (1999). Using the fisher kernel method to detect remote protein homologies. *7th Intell. Sys. Mol. Biol.*, pages 149–158.
- Kertész-Farkas, A., Dhir, S., Sonogo, P., Pacurar, M., Netoteia, S., Nijveen, H., Kuzinar, A., Leunissen, J., Kocsor, A., and Pongor, S. (2007a). Benchmarking protein classification algorithms via supervised cross-validation. *J Biochem Biophys Methods*, **35**, 1215–1223.
- Kertész-Farkas, A., Kocsor, A., and Pongor, S. (2007b). Equivalence learning in protein classification. In P. Perner, editor, *MLDM*, volume 4571 of *Lecture Notes in Computer Science*, pages 824–837. Springer.
- Kocsor, A., Kertész-Farkas, A., Kaján, L., and Pongor, S. (2006). Application of compression-based distance measures to protein sequence classification: a methodological study. *Bioinformatics*, **22**(4), 407–412.
- Kwok, J. T. and Tsang, I. W. (2003). Learning with idealized kernels. In T. Fawcett and N. Mishra, editors, *ICML*, pages 400–407. AAAI Press.
- Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, **5**, 27–72.
- Leslie, C. S., Eskin, E., and Noble, W. S. (2002). The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575.
- Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, **20**(4), 467–476.
- Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *J. Mach. Learn. Res.*, **2**, 419–444.
- Mount, D. W. (2004). *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press.
- Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, **48**(3), 443–53.

- Pearson, W. R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol*, **183**, 63–98.
- Rice, J. C. (1994). Logistic regression: An introduction. In B. Rhompson, editor, *Advances in social science methodology*, volume 3, pages 191–245. JAI Press, Greenwich.
- Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.
- Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.
- Sonego, P., Pacurar, M., Dhir, S., Kertész-Farkas, A., Kocsor, A., Gáspári, Z., Leunissen, J. A. M., and Pongor, S. (2007). A protein classification benchmark collection for machine learning. *Nucleic Acids Research*, **35**(Database-Issue), 232–236.
- 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., and Natale, D. A. (2003). The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, **4**.
- Tsang, I. and Kwok, J. (2003). Distance metric learning with kernels.
- Tsuruoka, Y., McNaught, J., Tsujii, J., and Ananiadou, S. (2007). Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*.
- Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*. Springer, 2 edition.
- Vert, J.-P. and Yamanishi, Y. (2004). Supervised graph inference. In *NIPS*.
- Vert, J.-P., Saigo, H., and Akutsu, T. (2004). Local alignment kernels for biological sequences. In B. Schoelkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, Cambridge, MA. MIT Press.
- Vert, J.-P., Qiu, J., and Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, **8(Suppl 10):S8**.
- Weinberger, K., Blitzer, J., and Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA.
- Xing, E., Ng, A., Jordan, M., and Russell, S. (2003). Distance metric learning, with application to clustering with side-information.
- Yamanishi, Y., Vert, J.-P., and Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, **20**(1), 363–370.
- Zien, A. and Ong, C. S. (2007). Multiclass multiple kernel learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1191–1198, New York, NY, USA. ACM.