Evolutionary Optimization of Kernel Weights Improves Protein Complex Comembership Prediction

Marc Hulsman, Marcel J.T. Reinders, and Dick de Ridder

Abstract—In recent years, more and more high-throughput data sources useful for protein complex prediction have become available (e.g., gene sequence, mRNA expression, and interactions). The integration of these different data sources can be challenging. Recently, it has been recognized that kernel-based classifiers are well suited for this task. However, the different kernels (data sources) are often combined using equal weights. Although several methods have been developed to optimize kernel weights, no large-scale example of an improvement in classifier performance has been shown yet. In this work, we employ an evolutionary algorithm to determine weights for a larger set of kernels by optimizing a criterion based on the area under the ROC curve. We show that setting the right kernel weights can indeed improve performance. We compare this to the existing kernel weight optimization methods (i.e., (regularized) optimization of the SVM criterion or aligning the kernel with an ideal kernel) and find that these do not result in a significant performance improvement and can even cause a decrease in performance. Results also show that an expert approach of assigning high weights to features with high individual performance is not necessarily the best strategy.

Index Terms—Classifier design and evaluation, biology and genetics, evolutionary computing and genetic algorithms.

1 Introduction

Combining features for classification is often a recurring problem in bioinformatics. One of the problems is that a common representation is required. Although a vector representation is often used, some features are hard to represent as vectors, such as DNA sequences (which have variable length), textual annotations, or graphs (e.g., molecular structures). A solution is offered by kernel methods [1], [2], [3]. Rather than representing each individual example by a feature vector, it can also be represented by its similarities to all other examples. It is possible to define such similarity measures for a large number of feature types. If the function used to calculate these similarities fulfills a number of mathematical conditions, it is called a kernel function. Different kernel functions, representing different views, can be combined into a single kernel function by simple summation. Several classifier algorithms can be used with kernel functions of which the Support Vector Machine (SVM [4]) is most popular.

Although simple summation of kernel functions already allows us to combine features in one common representation, it may be advantageous to be able to modify the influence of each feature, as features differ in terms of accuracy, noise level, coverage of the data set, scaling, etc. In this work, we discuss how to weight combinations of kernel functions. Determining these weights is not trivial. The standard approach used to determine classifier hyperparameters (i.e., a grid search in combination with cross-validation) does not work for determining kernel weights if the number of kernels to combine becomes large (e.g., >3), yet often there are many more. This has led researchers to develop several alternative methods to determine these weights, based on individual kernel performance, alignment of the kernel with the labels, or optimization of the SVM margin (see Section 1.1 on earlier work). In this paper, we propose to use an evolutionary algorithm (EA) in combination with cross-validation to determine the kernel weights, by optimizing a criterion such as the partial area under the receiver-operator characteristic (ROC) curve (PAUC). We do this both for the standard linear kernel combination (LKC) and a newly proposed nonlinear kernel combination (NKC).

We compare this new method with the previously developed methods by applying them to a large-scale integrative bioinformatics problem, namely, predicting whether two proteins are part of the same protein complex (protein complex comembership). The identification of the components of a complex can give important insights into its function. This identification problem is related to the discovery of normal protein interactions, although for protein complexes, we are only interested in stable interactions (on the other hand, we do include indirect interactions, i.e., using an intermediate protein). Two of the most direct high-throughput measurement methods to discover such interactions are high-throughput mass spectrometric protein complex identification (HMS-PCI) [5] and Tandem-Affinity Purification (TAP) [6]. However, the data obtained by these methods do not completely cover all possible interactions. That is, there are biases toward certain protein types due to the method used, and there is relatively low overlap of discovered interactions

• *The authors are with the Information and Communication Theory Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands. E-mail: {m.hulsman, m.j.t.reinders, d.deridder}@tudelft.nl.*

Manuscript received 4 Apr. 2008; revised 2 Sept. 2008; accepted 11 Dec. 2008; published online 19 Dec. 2008.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2008-04-0063. Digital Object Identifier no. 10.1109/TCBB.2008.137.

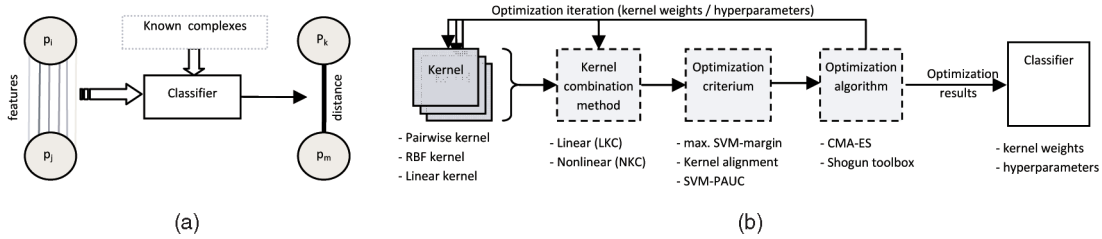

Fig. 1. (a) A classifier integrates features of a protein pair into a posterior probability, i.e., an estimate of the probability that the input protein pair forms part of a complex. This probability can be transformed in a distance measure. (b) A representation of the proposed kernel combination approach. The optimization algorithm can optimize the kernels (kernel hyperparameters), the combination method (kernel weights), and even the scoring method (classifier hyperparameters). The different methods which have been used in this paper are indicated under their corresponding category.

between both methods (about 20-30 percent) [7], [8]. In [8], it has been estimated that about half of the interactions found using high-throughput methods could be false positives. An important reason for this low accuracy is that the number of noninteractions is several orders of magnitude higher than the number of interactions [9]. Since experiments to validate these interactions are expensive and cumbersome, there is a need for a reliable automatic prediction of protein complexes.

The prediction of protein complexes can be viewed as a clustering problem in which proteins are to be clustered into complexes [10]. To make this possible, a distance measure between the proteins is required such that two proteins in the same complex will be assigned a smaller distance than two proteins not in the same complex. In this paper, we create a classifier to integrate protein and protein pair features into such a common distance measure (Fig. 1a). This classifier is trained to predict, for each *pair of proteins*, whether they are members of the same complex. The classifier's posterior probability can then be transformed into a distance measure.

1.1 Earlier Work

A number of computational methods have been developed, which integrate several different high-throughput data sources (features) in order to improve the quality of the protein complex comembership predictions in terms of accuracy and coverage [8], [11], [12], [13], [14]. Features used are, for example, mRNA expression correlation, interaction measurements, and gene sequences. In this study, we use a comprehensive list of features brought together in [13] (for a complete list of features used in our method, see Tables 2 and 3).

Previously proposed kernel combination algorithms include the Support Kernel Machine (SKM) [15], which incorporates training of the kernel weights in the SVM training process. It was applied in a study on protein function prediction [16], combining six different kernel functions. No significant performance improvement over equally weighted kernel combinations was found, although the SKM was more resistant to the addition of noisy kernels. Expanding on this method, Sonnenburg et al. [17] reformulated the problem as a semi-infinite optimization, which can be solved more efficiently. In another study [18], a gradient-based approach was employed to optimize the kernel and SVM hyperparameters using either the cross-validation error or estimates of the leave-one-out error. Because these measures are nonsmooth, they applied a smoothing technique to accurately find the minimum. A performance improvement for several (small-scale) experiments was reported. A similar gradient approach has been used in [19] to optimize the SVM margin criterion (used in the SKM) and an alternative called the kernel alignment criterion [20]. SVM margin optimization was found to outperform kernel alignment for setting the kernel weights, but performance was not compared to kernels combined with equal weight.

Combining kernels, as in these studies, is called *intermediate integration* [21]. In many other studies, in which no special kernels are used, *early integration* is used instead, i.e., concatenating feature vectors and applying a vector kernel to the concatenated vector. As our study involves the use of features that cannot easily be represented by vectors (the pairwise kernel [22], [23]), we will use intermediate integration as well. To our knowledge, there has been no study in which a large set of biologically relevant features has been represented as kernels and combined using optimized kernel weights and hyperparameters.

To optimize the combination of kernels, we employ an EA. A major benefit of the use of an EA is that it allows us to optimize kernel weights and kernel/classifier hyperparameters with respect to a single criterion. EAs have been used before in the context of hyperparameter optimization, i.e., to create a mixture-of-Gaussians kernel consisting of up to five radial basis function (RBF) kernels [24], or to optimize the SVM hyperparameters as well as to scale and rotate an RBF kernel [25]. Both, however, are not in the context of kernel combination and involve the early integration of features, where we use intermediate integration.

1.2 Overview

In the remainder of this paper, we discuss our method of setting kernel weights and hyperparameters by evolutionary optimization of the partial area under the ROC curve (PAUC), and compare it with the existing methods, which set kernel weights by optimizing the SVM margin or by aligning the combined kernel with the labels. Furthermore, we discuss the traditional linear combination of kernels and introduce a novel, nonlinear combination method. In experiments, we find that our method performs significantly better. Investigation of the kernel weights assigned by the different methods shows that optimized kernel weights do not directly correspond to individual feature performance. We investigate computational issues and find that while the nonlinear combination method does not outperform the linear one, it is computationally much

lighter. Finally, we show how the combination methods are robust to the presence of noisy features.

2 Methods

The used pipeline consists of the steps represented in Fig. 1b:

1. a set of kernels which are combined (see Section 3),
2. a method to combine the kernels,
3. an optimization criterion to evaluate the performance of this (weighted) combination, and
4. an optimization algorithm to find the kernel weights/hyperparameters maximizing this performance.

This procedure results after a number of iterations in

5. a final set of optimized kernel weights and hyperparameters, which will be used to build a well-performing classifier.

2.1 Kernel Combination Methods

To enable kernel combination, all nonkernel features are represented by vector kernels (either linear or RBF kernels, depending on the combination method). We tested both linear and nonlinear combination of kernels. LKC, as used by the SKM [15], is given by

$$k_{LKC}(x_i, x_j) = \sum_{p=1}^{P} w_p k_p(x_i, x_j), \tag{1}$$

where P is the number of kernels that are combined. As adding kernel functions k_p creates a kernel whose feature space is the product of the feature spaces of the individual kernels, the influence of each individual kernel space can be changed (scaled) by its corresponding kernel weight w_p. For this combination method, we use the RBF kernel to represent the feature vectors as well as the pairwise kernels.

We also propose to optimize an NKC, an RBF kernel function on the kernel space of a linear combination of kernel functions k_p:

$$k_{NKC}(x_i, x_j) = \exp\left(-\lambda \sum_{p=1}^{P} w_p (k_p(x_i, x_i) + k_p(x_j, x_j) - 2k_p(x_i, x_j))\right). \tag{2}$$

Here, feature vectors are represented by linear kernel functions. Note that by not bounding the sum of the weights w_p, it is not necessary to optimize λ, so it can be removed.

We do not perform explicit normalization of kernels. For the LKC method, the individual RBF kernel functions already map the samples to the unit hypersphere. The combined output of the NKC is normalized similarly by the combining RBF kernel function.

2.2 Criteria

To judge how well a combined kernel classifier using a certain set of kernel weights will perform on a test set, a criterion function is required. Several such functions have been proposed (e.g., [18]). In this work, we compare our proposed method, maximizing the partial area under the ROC curve (1), to maximizing alignment with the optimal kernel (2), and to maximizing the SVM-margin criterion (3).

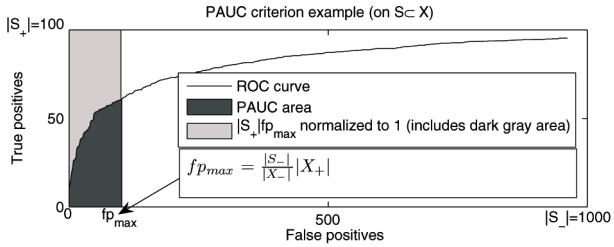

Fig. 2. Illustration of the calculation of the partial area under the ROC curve (PAUC) criterion, for a subset S of data set X. The PAUC criterion is indicated by the dark area; the cutoff $fp_{\max}$ is based on the number of true positives in X, adjusted for the size of S and the possibly different class balance between S and X. The criterion is normalized such that the light gray area + dark area is equal to 1. As in our data set, in this example, there are many more false positives than true positives.

2.2.1 SVM-PAUC Criterion

We propose to evaluate an application-specific criterion. For protein complex comembership prediction, this can be done by calculating the area under the (first part of the) Receiver-Operator Characteristic (ROC) curve (Fig. 2). This curve represents the number of true positives (correctly classified positive examples) and false positives (incorrectly classified negative examples) at different operating points of the classifier. The motivation for using only the first part of the curve is that, due to the imbalance between negative and positive examples, a large part of the curve describes situations in which the number of false positives ($|FP|$) is much larger than the number of positives ($|X_+|$). As a typical application of our prediction algorithm is to guide decisions on what protein pairs to test in a wet-lab setting, we prefer to give predictions for which the probability of a false positive is still reasonably small. Therefore, we focus on the part of the curve for which the number of false positives is smaller than the number of positives: $|FP| \leq |X_+|$ (Partial Area Under Curve, PAUC). In many studies on protein complex prediction, similar criteria (e.g., ROC50) have been used to represent performance (e.g., [13], [22]).

When using a subset S of the data set X with a different balance between positive and negative samples, we focused on the following part of the curve:

$$|FP| \leq fp_{\max} = \left\lfloor \frac{|S_-|}{|X_-|} |X_+| \right\rfloor, \tag{3}$$

where $S_- \subset S$ contains the negative examples in S. The cutoff $fp_{\max}$ is based on the number of true positives in X, adjusted for the size of S and the different class balance between S and X. As a final step, the area of the part of the graph selected by the previous constraint was normalized to one:

$$PAUC_{normalized} = \frac{PAUC}{|S_+| fp_{\max}}, \tag{4}$$

where $S_+ \subset S$ contains the positive examples in S.

TABLE 1
An Overview of the Methods Tested in the Experiments

Method name	Combination method	Kernel weight optimization method	Parameter optimization method (C_+, C_-, λ)
Individual features	-	-	PAUC criterion optimization using CMA-ES
Equal weight/LKC	LKC	-	PAUC criterion optimization using CMA-ES
Equal weight/NKC	NKC	-	PAUC criterion optimization using CMA-ES
Kernel alignment/NKC	NKC	Kernel alignment criterion optimization using CMA-ES	PAUC criterion optimization using CMA-ES
SVM-margin/LKC	LKC	SVM-margin criterion optimization using Shogun toolbox	PAUC criterion optimization using CMA-ES (outer loop)
Reg. SVM-margin (> 0.5)	LKC	SVM-margin criterion optimization using Shogun toolbox (weight constrained to be $> 0.5/P$)	PAUC criterion optimization using CMA-ES (outer loop)
SVM-PAUC/LKC	LKC	PAUC criterion optimization using CMA-ES	
SVM-PAUC/NKC	NKC	PAUC criterion optimization using CMA-ES	

2.2.2 Kernel Alignment

The kernel alignment criterion [20] is based on an ideal kernel function, constructed using the labels y_i, i.e., $k_{ideal}(x_i, x_j) = \langle y_i, y_j \rangle$ (with $y_i \in \{-1, 1\}$). This kernel function can be used to construct an ideal kernel matrix, which includes all pairs of available examples: $K_{ideal} = \mathbf{y}\mathbf{y}^T$, where $\mathbf{y}$ is the label vector for all examples. To score a combined kernel matrix K_c, we can align it with K_{ideal} by employing

$$A(K_c) = \frac{\langle K_c, K_{ideal} \rangle_F}{\sqrt{\langle K_c, K_c \rangle_F \langle K_{ideal}, K_{ideal} \rangle_F}}, \tag{5}$$

where $\langle ., . \rangle_F$ is the Frobenius inner product. The alignment is maximized by changing the kernel weights w_p used to construct K_c. Since inner products can be interpreted as similarities, this can be understood as maximizing the average similarity between protein pairs within the same class, while minimizing the average similarity between protein pairs in different classes.

2.2.3 SVM Margin

The SVM classifier is based on maximization of the margin around the decision boundary. In the case of overlapping classes, a regularization term is added, resulting in the soft-margin SVM criterion:

$$\frac{1}{2}\mathbf{v}^T\mathbf{v} + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=-1} \xi_i, \tag{6}$$

where C_+ and C_- are the penalty terms for margin violations in the two different classes.[1] This criterion is optimized w.r.t. weights v_i and bias b under the constraints $y_i(\mathbf{v}^T\mathbf{x} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, $\forall i$. In [15], the authors propose to use this criterion to optimize the kernel weights $\mathbf{w}$ as well, resulting in the SKM.

An SVM is normally optimized using its dual formulation, which is stated in terms of a kernel function k, sample weights α_i, and sample labels $y_i = \{-1, 1\}$:

$$\max_{\boldsymbol{\alpha}} 2\sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j). \tag{7}$$

The weights $\boldsymbol{\alpha}$ are optimized during SVM training. As this is the dual, we have to minimize it to find the optimal kernel weights w_p for kernels k_p:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} 2\sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_{p=1}^{P} w_p k_p(x_i, x_j). \tag{8}$$

When minimizing this function, it is necessary to add a constraint on $\mathbf{w}$, as otherwise, the margin width can simply be increased by using higher weights w_p. In [15], the following constraint was used: $\mathrm{trace}(\sum_{p=1}^{P} w_p k_p(x_i, x_j)) = c$, where c is a constant value. However, as the trace of all our kernel matrices K_p is equal (the used RBF kernel functions give only 1s on the diagonal), we can simplify this to $\sum_{p=1}^{P} w_p = 1$.

The SVM margin criterion cannot be used to optimize other parameters, such as the RBF kernel hyperparameter λ in the NKC. Increasing λ leads to larger distances between examples, and thereby, increased margin width. However, it also increases the nonlinearity of the resulting classifier, which can lead to a reduced generalization performance due to overfitting. Therefore, maximizing the margin w.r.t. λ will not maximize performance of the classifier. For this reason, we employ SVM margin maximization only with LKC kernel combination.

We noticed that the solutions using the SVM margin criterion were often relatively sparse. In [26], a regularized version of this method is proposed. It constrains the minimum value of the kernel weights and gives better results. We tested this regularized SVM margin method as well, using $0.5/P$ as minimum kernel weight.

2.3 Optimization Algorithm

We need to optimize both kernel weights and hyperparameters (i.e., RBF kernel width λ, SVM regularization parameters C_+, C_-) (see Table 1). The optimization of hyperparameters is necessary to be able to do a fair comparison between methods, as the optimal values of the hyperparameters change with the kernel weights used.

In our proposed method, based on the SVM-PAUC criterion, we employ an EA. We use the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [27] since it performs well and a Matlab implementation is readily available. Many EAs use the concept of mutation adaptation, where not only the solution variables are mutated but also the variables governing the mutation process. CMA-ES is a variation on this concept in which adaptation of the mutation variables is derandomized. In each generation, the best solutions are used to update a covariance matrix and a mutation scale parameter. These parameters are subsequently used to generate mutations

1. Normally, a single penalty constant C is used, but due to the large imbalance between the classes, we chose to use class-specific penalty terms C_+ and C_-, and optimize each individually.

TABLE 2
Features on Proteins

Feature	Source	Values	Description
mRNA expression	[33]–[35]	$v_i \in R^+$	Expression correlation can be an indicator of common function and possible interaction. We used the 11 microarray data sets which were also used in [13]. Each of these datasets consists of at least 40 microarrays. Of these datasets, 9 were extracted from the Stanford Microarray Database [33], while the other two were obtained from [34], [35].
mRNA copy number	[36]	$v_i \in R^+$	This feature should not contain information useful for protein complex prediction. It was used to determine the handling of random features by our method.
RAP1 binding sites	[37]	$v_i \in R^+$	Binding sites of protein RAP1 on promoters of certain genes. This feature was used to determine the handling of almost random features by our method.
Motifs	[38]	$v_i \in \{0, 1\}$	Common regulation mechanisms can be an indication of a common complex-membership.
Protein sequence (PFAM)	[32]	$v_i \in \{0, 1\}$	Protein sequence determines protein structure, which determines if proteins bind. These (normalized) linear kernels use every sequence domain occuring in PFAM, eMotif or the complete list of 3-mers. These sequence domains are seen as binary features, indicating if the sequence domains occurs in a particular protein.
Protein sequence (eMotif)			
Protein sequence (spectrum, $k = 3$)			

biased to occur in the directions and range that gave the best improvements in previous generations. The algorithm has proven to have a very competitive local and global optimization performance [28]. We use this algorithm to optimize both the kernel weights and the hyperparameters. The advantage of the PAUC-based criterion is that we can optimize these simultaneously.

To optimize the kernel alignment criterion w.r.t. the kernel weights, we also use the CMA-ES approach. The kernel alignment is influenced by the used RBF kernel width. We found that an approach of setting the RBF kernel width to a value optimized for the equal weight method did not perform well, as it resulted in a very sparse assignment of the kernel weights. A better performance was obtained by optimizing the RBF kernel width simultaneously with the kernel weights. The kernel alignment score cannot be used to optimize the classifier hyperparameters (i.e., C_+ and C_-). For this reason, we optimize these parameters afterward by optimizing the PAUC score using CMA-ES. We also simultaneously optimize the RBF kernel width again, as we found that this improves performance. Such an independent SVM-PAUC-based optimization of hyperparameters is also used for the methods using equal weights and only individual features.

Although we could, in principle, also use CMA-ES to optimize the SVM margin criterion, a number of specialized algorithms are available, incorporating the training of the kernel weights in the SVM training [15], [29], [30]. A relatively fast algorithm was developed in [17] in which the authors reformulated the optimization problem as a semi-infinite linear program, reusing efficient implementations already available for the standard SVM. We use this algorithm to determine the kernel weights. A complicating factor here is that we cannot optimize the hyperparameters using this criterion, as they are not independent (i.e., they influence the obtained set of optimal kernel weights). To make the comparison as fair as possible, we use the computationally very costly approach of setting the hyperparameters in an outer loop (by optimizing the PAUC criterion using CMA-ES) around the kernel weight determination (by optimizing the SVM margin using [17]). In each iteration of CMA-ES, we set a certain set of hyperparameters, find the kernel weights using the SVM margin criterion, and determine the SVM-PAUC criterion.

3 Experiments

3.1 Data

For predicting protein complex comembership, we use features on proteins (genes) as well as features on protein pairs (gene pairs). The first group (Table 2) consists of features such as the protein sequence, the presence of motifs, and mRNA expression, while the second group (Table 3) consists of features such as experimentally derived interactions, gene cooccurrence, and gene coessentiality. These features were brought together in [13], based on previous studies. We normalize these features by dividing them by their maximum value, as dividing them by standard deviation would lead to some very high values for the sparse features. Furthermore, we did not perform normalization of the feature mean because this would remove the sparsity of some features (i.e., all zeros would become nonzeros). This sparsity is used to improve the speed of kernel calculation, as only nonzero values are used during calculation.

As the objects classified are protein pairs, we need to convert features on individual proteins into features on pairs of proteins. We use Pearson correlation for the mRNA expression vectors. Protein sequence kernels are converted in two different ways: using a protein similarity kernel (a linear kernel on sequence kernels) and using a pairwise kernel on sequence kernels [22] (for more detailed description of these kernels, see Fig. 3). In total, we have $P = 49$ kernels.

The class labels needed to create a train- and test set were extracted from the MIPS yeast complex database [31]. Category 550, which covers complexes determined by high-throughput experiments, was excluded because these high-throughput experiments are used as features. For each known MIPS complex, we use all protein pairs within the complex as positive examples. This results in 10,480 positive examples of protein complex comemberships, derived from 216 complexes covering 1,168 genes. It is not that obvious which examples should be used as negative examples [32]. Following [13], we decided to pair proteins that take part in different complexes and use these as negative examples, which results in a total of 722,175 negative examples. These negative examples are relatively trustworthy since they contain well-studied proteins. In our experiments, we use a more equally balanced subset rather than the whole data set (with a positive:negative ratio of 1:4, see Sup. Fig. 6, which can be

TABLE 3
Features on Protein Pairs

Feature	Source	Values	Description
MIPS (category 550)	[5], [6], [31], [39]	$\{0,1\}$	The MIPS complex database has a special category (550) for complexes derived using high-throughput experiments. Each pair of proteins within these experimentally derived complexes are given value 1, while all other protein pairs are given value 0 (matrix model).
Conserved gene neighborhood	[8]	N^+	It has been noted that genes that are located near each other on the genome are more likely to interact. This is especially the case in prokaryotic operons. In [8] it was determined if orthologs of genes occur near each other on at least 2 of 42 sequenced genomes.
Gene fusion		N^+	Some interacting proteins in one organism are fused together in another organism. By searching whether a certain pair of proteins is fused together in other organisms an indication is obtained of possible interactions. Gene fusions were detected by searching for genes which are present in more than one Cluster of Orthologous Genes (COG) [8].
Gene coocurrence		$\{0,1\}$	It is expected that the genes for proteins that are part of a complex occur together in genomes. Using a mutual information measure and an ortholog database it was determined which genes have a tendency to occur together in 42 sequenced genomes, as collected by [8].
Synexpression		$\{0,1\}$	Expression correlation can be an indication of interaction. In [8] the expression correlation was calculated, and a cut-off parameter was chosen to convert this correlation value to a binary feature.
Gene co-essentiality		$\{0,1\}$	Sometimes knocking out two genes at the same time causes lethality, while knocking out the individual genes does not. This indicates a relation between the two genes, for example being part of different pathways with the same functionality.
Disruption	[31]	$\{0,1,2\}$	If a protein complex contains an essential protein (removal is lethal), then the other parts of the complex are probably also essential. The feature value indicates how many proteins of a pair are essential.
Tandem affinity purification (TAP)	[8], [40]	N^+	These experimentally derived interactions were obtained from the DIP database as well as from [8]. The features were handled separately. The feature value indicates how many times an interaction was found in the dataset.
High throughput mass-spectometric protein complex identification (HMS-PCI)		N^+	
Two-hybrid test		N^+	
Affinity chromatography	[40]	N^+	These experimentally derived interactions were obtained from the DIP database. The small scale feature 'DIP various' contains all experimental methods with less than 100 interactions [13] (see Table S1 for details). The feature value indicates how many times an interaction was found in the dataset.
Split ubiquitin system		N^+	
In vitro binding		N^+	
Immunopreciptation		N^+	
In vivo kinase activity assay		N^+	
DIP various		N^+	
Interolog	[41], [42]	R^+	Using sequence similarity between genes, interactions can be mapped from one organism to another. The feature value is based on the joint sequence similarity.

found on the Computer Society Digital Library at http://doi.ieeecomputersociety. org/10.1109/TCBB.2008.137).

To prevent any biases during validation and testing, the data set was split into seven parts in such a way that no single protein is used in multiple parts. Of these seven parts, four parts were used for training and validation, while the remaining three parts were used for testing. When cross-validation was needed for optimizing the PAUC score, we used a fourfold cross-validation scheme in which we use only one part for training and the other three parts for validation. The training parts consist of 656 positive pairs and 2,624 negative pairs each. The final test is done on the test subset, which consists of 2,021 positive pairs and 39,759 negative pairs.

3.2 Implementation

The tested methods are described in Table 1. To optimize the SVM, we used a modified version of LibSVM [43] as well as PRTools [44]. For the SVM margin method, we used the Shogun toolbox [17]. Kernel weights and hyperparameters as well as the kernel alignment criterion were optimized using the CMA-ES software [27]. We stop each kernel weight optimization using CMA-ES after 2,500 function evaluations (i.e., cross-validations). Optimizations of hyperparameters are stopped after 250 function evaluations. Since both subselecting a data set and evolutionary optimization are random, we repeat each experiment five times and report the average test score and its standard deviation.

As suggested in [18] and [45], we formulate the kernel weights as $w_p = 2^{\omega_p}$ and optimize ω_p. The advantage is that the parameters that define the weights (ω_p) no longer have to be restricted to be positive, while optimization performance is also reported to improve. The same thing is also done for the hyperparameters. Furthermore, we put a bound on C_+ and C_- of 2^{15}, as too high values cause numerical instabilities and slow convergence.

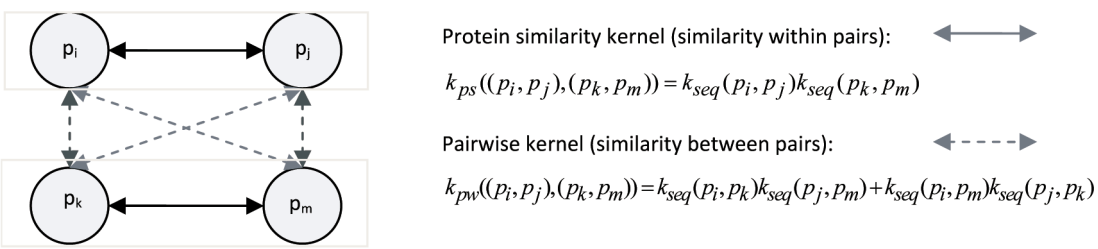

Fig. 3. Kernels for protein pairs, based on protein sequence kernels (PFAM, eMotif, and spectrum). The protein sequence kernels are linear kernels, comparing sequence similarity within pairs. The pairwise kernels [22] compare the (crosswise) similarity of the sequences of two protein pairs.

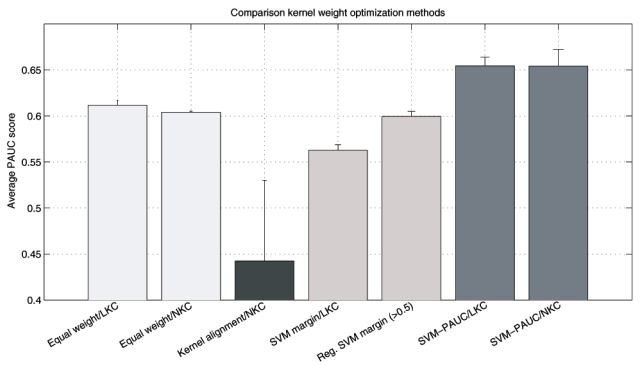

Fig. 4 Performances of the different methods for kernel combination. The average score and standard deviations are shown, based on five repetitions of each experiment (three for the SVM margin methods). Regularized margin maximization is indicated by (> 0.5), indicating that the weight for each individual kernel is larger than $0.5/P$.

4 Results

4.1 Weighted Kernels

We first compared the different kernel weighting methods to determine whether they increase performance over the equal weighted kernel method (Fig. 4). That optimizing kernel weights can be advantageous is shown by the SVM-PAUC optimization, which performs better than the equal weighted methods for both NKC ($+0.05$ PAUC or predicting 76 percent instead of 70 percent of the positive pairs at less than $fp_{\max}$ errors) and LKC. As the equal weighted combination already uses a state-of-the-art classifier with optimized hyperparameters, this can be considered a significant improvement.

The score for kernel alignment is surprisingly low. We attribute this low score to its focus on a too ideal kernel, optimizing all kernel values with equal weight, while for classification performance, we are mainly interested in the kernel values between objects around the classification border. For SVM margin maximization, we also see a relatively low performance. Inspection of the weights shows that this is probably due to the relatively high weight assigned to nonsparse features, specifically the pairwise kernels (Sup. Fig. SF3, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.137). We hypothesize that the latter is due to the higher dimensionality of the input feature space of these kernels, making it easier to find a well-separating classifier. For this reason, we repeated the experiment without including the pairwise kernels. Results (Sup. Fig. SF5, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.137) indeed improve: performance is closer to that of the equal weight method, although still not close to the SVM-PAUC methods. This is in line with results from earlier studies [16], [26], [46], which also reported no significant increase or a decrease in performance.

Also, in line with results in [26], we found that the regularized SVM margin maximization method leads to an improvement in performance over the original version. However, it still suffers from the relatively high weights assigned to the pairwise kernels, and we find again that better results are found when we leave these kernels out (Sup. Fig. SF5, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.137).

4.2 Individual Kernels

To determine whether combining kernels is worthwhile, we then tested performance using each kernel individually and compared it to the performance obtained using the best-performing combination method (SVM-PAUC/NKC, NKC with the partial area under ROC-curve evaluation function (PAUC)). This also enables us to see if there is any connection between individual kernel performance and optimized kernel weights for NKC. What stands out is the large performance gain that we obtain by combining using SVM-PAUC/NKC, predicting 76 percent of the true positives with less than $fp_{\max}$ false positives, instead of the 42 percent obtained using the best individual kernel. Individually, the mRNA expression features give the best performance, while kernels based on features such as TAP, HMS-PCI, and immunoprecipitation also stand out. However, these individual scores do not directly correspond to the optimized kernel weights (Fig. 5c). The highest weights are instead assigned to two kernels based on very sparse features (gene cooccurrence and various DIPs) with a low number of false positives in their nonzero values (high accuracy). Kernels based on low-accuracy features such as MIPS (high throughput) get a low weight.

We evaluated the importance of each kernel by removing it during testing and calculating the performance penalty incurred (Fig. 5b). The results show that the highest weighted kernels are not the most important for performance of the current classifier, for example, due to their sparseness or overlap with other features. More important are, as expected, kernels based on direct evidence such as MIPS (high throughput), TAP, and immunoprecipitation features. In general, overlap in information content between features complicates the interpretation of the kernel weights. Often, a low weight for a specific kernel is accompanied by a high weight for a relatively similar kernel (see Sup. Fig. SF4, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.137). Another difficulty lies in the relatively large differences between weights of different runs, indicating that there are multiple local optima. Due to these local optima, it is advisable to rerun the optimization procedure several times to get a good indication of performance and obtain the best set of kernel weights.

Easier to interpret are consistently low kernel weights as they signal that the corresponding features do not contribute to the current classifier at all. We have two features in our data set for which we do not expect a contribution to classification performance: DNA copy number [36] and Rap1-binding sites [37]. The kernels based on these features indeed always receive a very low kernel weight. More surprising is that the kernel on the disruption feature, which has a very low individual score, is not assigned a low weight. Removing this kernel actually slightly improves the performance ($p = 0.07$, two-tailed paired t-test). This overfitting, where the kernel seems to be useful on the cross-validation set, but proves not to be so on the test set, also occurs for the synthetic lethality feature. A larger cross-validation set

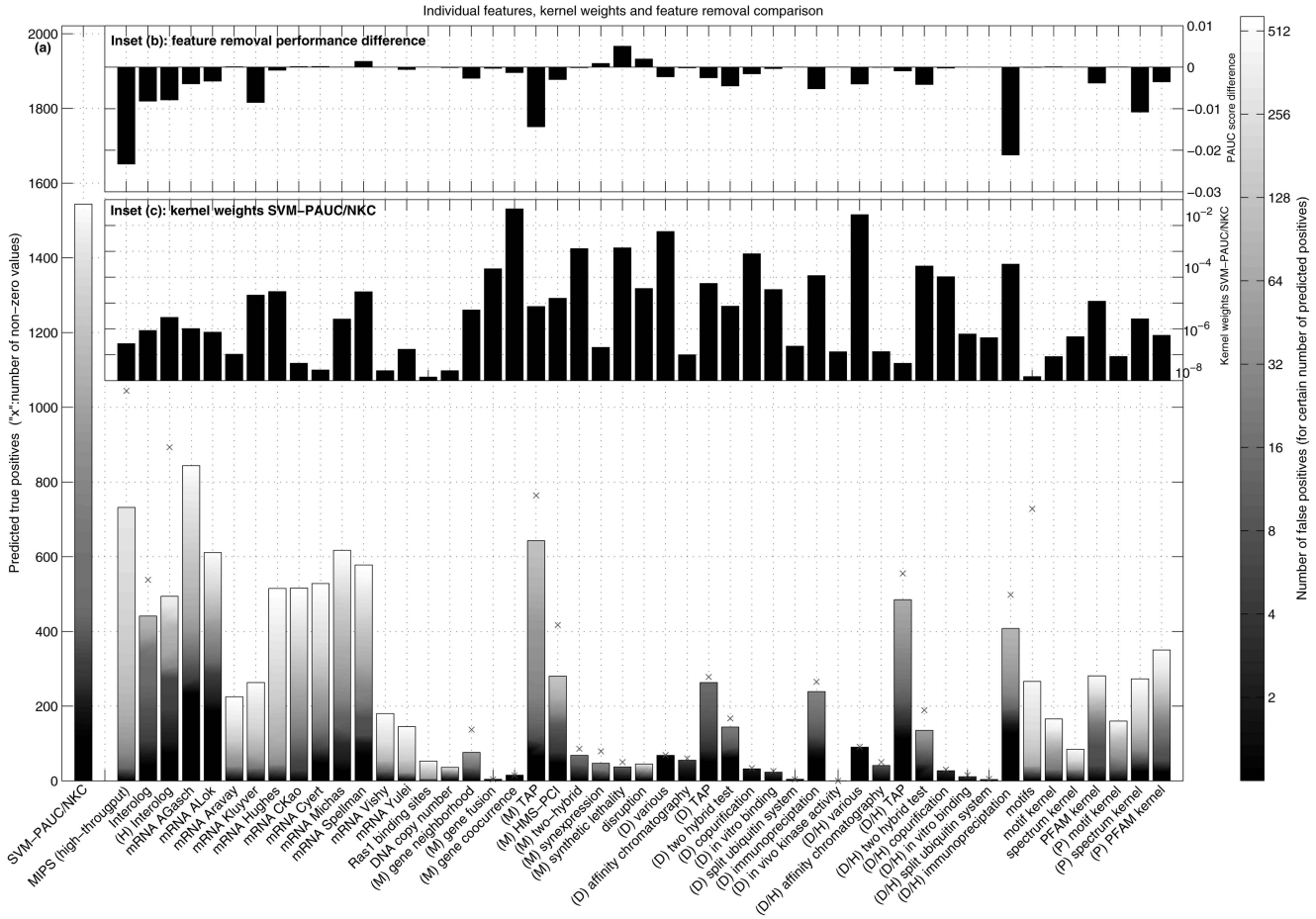

Fig. 5. (a) Performance of SVM-PAUC/NKC and individual features. The height of the bars corresponds to the number of predicted true positives for $fp_{\max} = 577$ false positives, and shading of the bars indicates the number of false positives as a function of the number of predicted true positives. These experiments have been performed on a test set consisting of 39,759 negative pairs and 2,021 positive pairs. For sparse features, the height of the bar is limited to the point where the number of true positives (height) + false positives (shading) is equal to the number of nonzero feature values (indicated with an "x" for these features), as the remainder of the bar would only show random classifier effects. Features belonging to the [8] data set are indicated with (M), features from the DIP database are indicated with (D), and two-hop features [47], [13] are indicated with (H). Inset (b) PAUC performance difference when one kernel is left out during testing of an optimized SVM-PAUC/NKC kernel combination. Inset (c) $\log_{10}$-averaged kernel weights over five experiments, optimized using SVM-PAUC/NKC. We chose to represent the kernel weights using a log scale, as we found that even low weights (to 10^{-8}) can still contribute significantly to classification performance. Weights lower than 10^{-8} have been cutoff (before calculating the average), as kernels with weights lower than that did not affect performance.

could remedy this. We further test the susceptibility of our method to noisy features in Section 4.4.

The lack of correlation between individual kernel performance and kernel weight suggests that an expert approach of setting kernel weights based on individual kernel performance would not be optimal. We tested this by setting the weights according to the kernel performance, using different lower limits on the kernel weights. The results indicate that setting the weights in this way does not lead to a significant performance improvement (Fig. 6).

We conclude that combining features for predicting protein complex comembership is worthwhile, as performance on the combined kernels is much better than on any individual kernel. Furthermore, an expert approach of setting kernel weights based on individual kernel performance seems not to be optimal; kernel weights are instead influenced more by accuracy and overlap of features.

4.3 Computational Cost

NKC performed nearly the same as LKC (Fig. 4), indicating that the more linear approach of LKC suffices for our problem. However, the computational cost of LKC is significantly higher than that of NKC. On our data set, LKC trained almost 50 times slower than NKC (266 ± 133 hours versus 5.4 ± 0.3 hours). Although the computational cost of calculating the individual kernels is higher for LKC, this does not fully explain the increased cost. We found that the main problem was the slower convergence of the SVM classifier for LKC. This was confirmed by training a linear combination of linear kernels, which was also slower than NKC, although the calculation of the combined kernel itself was less computationally costly. In [48], the authors note a similar effect when comparing the convergence of the SVM classifier for linear and RBF kernels, finding that for high values of the regularization parameter C, the linear kernel becomes much slower in convergence.

A possible remedy for the large computational cost is to choose a smaller upper bound for the regularization hyperparameters C_+ and C_- of the SVM classifier, as this speeds up SVM convergence (for both LKC as well as NKC). Alternatively, we can use less iterations of the

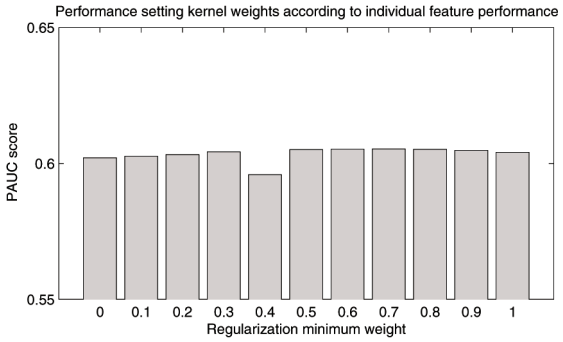

Fig. 6. Performance of NKC when kernel weights are set according to the individual kernel (PAUC) performance. Regularization was performed by setting a minimum weight. Afterward, the weights were normalized to sum to 1; hence, a minimum weight of 1 corresponds to using equal weights. Subsequently, the hyperparameters were optimized using CMA-ES. It is apparent that setting weights according to this method does not significantly increase performance. The lower performance at a regularization minimum weight of 0.4 is probably due to a local minimum in the (nonconvex) optimization problem.

evolutionary algorithm. In our experiment, after less than 250 function evaluations, the score is already higher than for the equal-weighted/NKC method, and after approximately 1,250 function evaluations, the score improvements level off (Fig. 7). This shows that the largest performance improvements are made in the first iterations, so if computation time is limited, even running the optimization for a few hundred iterations can still be advantageous.

Another way to reduce computational cost is to determine kernel weights on a small data set, before training the final classifier on a larger data set. We tested this and the results (see Fig. 8) show that kernel weights determined on a smaller data set still improve performance when used for a larger data set. Note that the kernel weights optimized with and tested on the large data set perform slightly *worse* than those found using the default data set and tested on the large data set. This is caused by the construction of the larger data set (see Fig. 8), which reduces the amount of data available for validation and could have caused overfitting. An important insight is therefore that increasing the validation

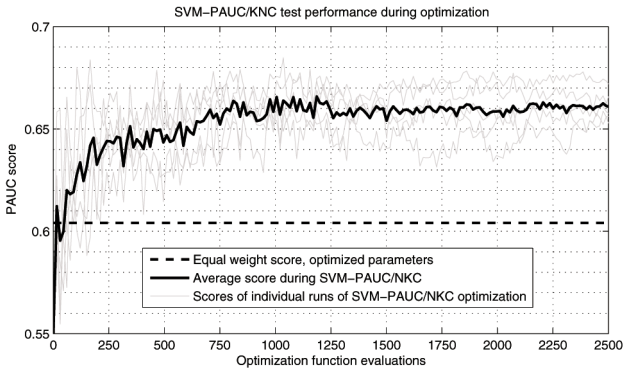

Fig. 7. PAUC test performance during optimization. The scores of the individual runs of the SVM-PAUC/NKC optimization are shown in gray, while the average score is shown in black. Note that after approximately 250 function evaluations, the score is already noteworthy higher than for the equal weighted scenario. After approximately 1,250 function evaluations, a plateau is reached, and further score improvements are marginal.

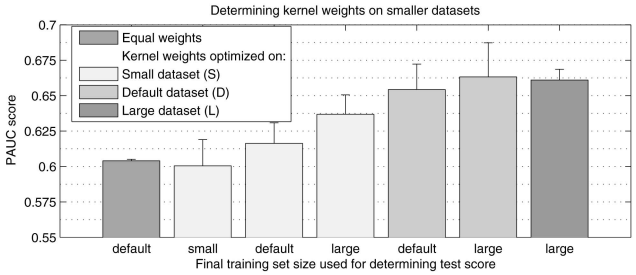

Fig. 8. Test performance when using kernel weights optimized on smaller data sets. The size of the test data set is kept constant for all tests. The data set sizes are varied for: 1) the optimization of kernel weights and hyperparameters (C_-,C_+,λ) by iterative cross-validation and 2) training the final classifiers. These data set sizes are indicated by the gray-value and the labels on the x-axis, respectively. The small data set consists of 656 positive and 656 negative examples per part and the default data set consists of 656 positive and 2,624 negative examples per part. For both, we use four parts to perform fourfold cross-validation, using one part for training and three for validation. To create a large data set, there is not enough data for four large parts; so we use the default data set and change the cross-validation procedure to use three parts for training and one part for validation. This makes the graph less suitable for determining the performance effects of an increased number of examples. However, it serves to show the effects of using kernel weights optimized on smaller data sets to train a classifier on a larger data set.

set size and decreasing the training set size during cross-validation can both increase the performance (by avoiding overfitting) and reduce the computational cost. The decrease in computational cost can be significant, as is shown by the runtimes of the optimizations: respectively, 1.4, 5.4, and 47.5 hours on the small, default, and large data set.

4.4 Influence of Noise

We checked the influence of noisy kernels on the performance by adding 5, 10, 20, or 40 one-dimensional noisy kernels. The noise features were generated using a uniform distribution in the range $[0, 1]$, the kernels were constructed as described in Section 2.1. As the added kernels increase the number of kernel weights to be estimated, we ran the optimization for 10,000 iterations instead of 2,500. Results (Table 4) show that the methods are quite resilient to noise, especially the equal-weighted classifier. A larger decrease in performance of the SVM-PAUC/NKC method is to be expected as optimizing the kernel weight parameters can increase the overfitting effect. However, we see that only after adding 40 noisy kernels to the 49 informative ones, a noteworthy decrease in performance occurs. This shows that our method is able to handle a large amount of kernels which do not contain useful information for the current problem without much overfitting, which makes kernel preselection an easier task.

TABLE 4
Influence of Noise

PAUC ($\pm st.dev$)	SVM-PAUC/NKC (10,000 iterations)	Equal-weight/NKC
All features	0.6543 ($\pm$0.0083)	0.6041 ($\pm$0.0011)
All features + 5 noisy	0.6424 ($\pm$0.0141)	0.6031 ($\pm$0.0012)
All features + 10 noisy	0.6459 ($\pm$0.0139)	0.6018 ($\pm$0.0015)
All features + 20 noisy	0.6539 ($\pm$0.0080)	0.5985 ($\pm$0.0011)
All features + 40 noisy	0.6296 ($\pm$0.0246)	0.5941 ($\pm$0.0026)

5 Conclusion

The integration of features using weighted kernel representations is a powerful method, which, in combination with the SVM, delivers very good performance on the problem of protein complex comembership prediction. In this paper, we proposed and compared a number of methods to optimize kernel weights and hyperparameters. The main conclusion of this paper is that kernel weight optimization can improve performance significantly.

However, not all methods lead to an optimal set of kernel weights. We found that criteria not involving the classifier performance to optimize the kernel weights perform poorly, while also requiring a separate optimization of hyperparameters such as the RBF kernel width. This is in line with [49], where it was also found that better performance is reached by directly optimizing the criterion one is interested in (AUC). Our conclusion is that the wrapper approach of optimizing classifier performance, although computationally expensive, is still the method of choice. We showed that optimization using cross-validation remains feasible by using the CMA-ES evolutionary algorithm. Furthermore, the use of CMA-ES allows us to freely choose the optimization criterion, in order to better match the practical use of the classifier. It has to be noted that this leads to nonconvex optimization problems for which no optimization method (including evolutionary algorithms) can be guaranteed to find the global optimum. However, EAs are specifically developed to tackle this kind of problem and we have no reason to believe that the solutions found are poor. A further advantage of CMA-ES optimization is that it is suitable for parallel execution. With the advent of multicore systems, this could significantly reduce optimization times [27].

We tested both an LKC approach and an NKC approach and found both to improve classifier performance on our problem, with NKC being far less computationally costly due to a faster convergence of the classifier. In case computational cost still is an issue, we showed that high performance can still be reached by training the kernel weights on a smaller data set. Decreasing the training set size and increasing the validation set size during kernel weight optimization can even have a positive influence on the achieved test performance.

On inspection of the relation between kernel weights and performance, we found that an expert approach of setting the weights of the kernels according to their (expected) performance is not optimal. Factors such as accuracy, sparseness, and overlap between features also have to be taken into account.

In conclusion, weighted kernel combination leads to a significant gain in performance. This makes the computational cost worthwhile, especially in cases where quality of results is very important due to the high cost of checking results by experimental methods.

References

[1] B. Schölkopf, K. Tsuda, and J.-P. Vert, *Kernel Methods in Computational Biology.* MIT Press, 2004.

[2] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.

[3] A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, "Support Vector Machines and Kernels for Computational Biology," *PLoS Computational Biology,* vol. 4, no. 10, Oct. 2008.

[4] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.

[5] Y. Ho et al., "Systematic Identification of Protein Complexes in Saccharomyces Cerevisiae by Mass Spectrometry," *Nature,* vol. 415, pp. 180-183, Jan. 2002.

[6] A.-C. Gavin et al., "Functional Organization of the Yeast Proteome by Systematic Analysis of Protein Complexes," *Nature,* vol. 415, pp. 141-147, 2002.

[7] G.D. Bader and C.W.V. Hogue, "Analyzing Yeast Protein-Protein Interaction Data Obtained from Different Sources," *Nature Biotechnology,* vol. 20, pp. 991-997, 2002.

[8] C. von Mering, R. Krause, B. Snel, M. Cornell, S.G. Oliver, S. Fields, and P. Bork, "Comparative Assessment of Large-Scale Data Sets of Protein-Protein Interactions," *Nature,* vol. 417, pp. 399-403, 2002.

[9] R. Jansen and M. Gerstein, "Analyzing Protein Function on a Genomic Scale: The Importance of Gold-Standard Positives and Negatives for Network Prediction," *Current Opinion in Microbiology,* vol. 7, pp. 535-545, 2004.

[10] G.D. Bader and C.W.V. Hogue, "An Automated Method for Finding Molecular Complexes in Large Protein Interaction Networks," *BMC Bioinformatics,* vol. 4, 2003.

[11] R. Jansen, Y. Haiyuan, D. Greenbaum, Y. Kluger, N.J. Krogan, S. Chung, A. Emili, M. Snyder, J.F. Greenblatt, and M. Gerstein, "A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data," *Science,* vol. 302, pp. 449-453, 2003.

[12] L.V. Zhang, S.L. Wong, O.D. King, and F.P. Roth, "Predicting Co-Complexed Protein Pairs Using Genomic and Proteomic Data Integration," *BMC Bioinformatics,* vol. 5, i38-i46, 2004.

[13] R.J.P. van Berlo, L.F.A. Wessels, D. de Ridder, and M.J.T. Reinders, "Protein Complex Prediction Using an Integrative Bioinformatics Approach," *J. Bioinformatics and Computational Biology,* vol. 4, pp. 839-861, 2007.

[14] Y. Qi, Z. Bar-Joseph, and J. Klein-Seetharaman, "Evaluation of Different Biological Data and Computational Classification Methods for Use in Protein Interaction Prediction," *Proteins: Structure, Function, and Bioinformatics,* vol. 63, pp. 490-500, 2006.

[15] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan, "Learning the Kernel Matrix with Semidefinite Programming," *Proc. 19th Int'l Conf. Machine Learning (ICML '02),* pp. 323-330, 2002.

[16] G.R.G. Lanckriet, T. de Bie, N. Cristianini, M.I. Jordan, and W.S. Noble, "A Statistical Framework for Genomic Data Fusion," *Bioinformatics,* vol. 20, pp. 2626-2635, 2004.

[17] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large Scale Multiple Kernel Learning," *J. Machine Learning Research,* vol. 7, pp. 1531-1565, 2006.

[18] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing Multiple Parameters for Support Vector Machines," *Machine Learning,* vol. 46, pp. 131-159, 2002.

[19] O. Bousquet and D. Herrmann, "On the Complexity of Learning the Kernel Matrix," *Advances in Neural Information Processing Systems,* S. Becker, S. Thrun, and K. Obermayer, eds., vol. 15. pp. 415-422, MIT Press, 2003.

[20] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola, "On Kernel Target Alignment," *Advances in Neural Information Processing Systems,* vol. 14, pp. 367-374, MIT Press, 2002.

[21] P. Pavlidis, J. Weston, C. Jinsong, and W.S. Noble, "Learning Gene Functional Classifications from Multiple Data Types," *J. Computational Biology,* vol. 9, pp. 401-412, 2002.

[22] A. Ben-Hur and W.S. Noble, "Kernel Methods for Predicting Protein-Protein Interactions," *Bioinformatics,* vol. 21, no. 1, pp. i38-i46, 2005.

[23] S. Martin, D. Roe, and J.-L. Faulon, "Predicting Protein-Protein Interactions Using Signature Products," *Bioinformatics,* vol. 21, pp. 218-226, 2005.

[24] T. Phienthrakul and B. Kijsirikul, "Evolutionary Strategies for Multi-Scale Radial Basis Function Kernels in Support Vector Machines," *Proc. 2005 Conf. Genetic and Evolutionary Computation,* pp. 905-911, 2005.

[25] F. Friedrichs and C. Igel, "Evolutionary Tuning of Multiple SVM Parameters," *Proc. European Symp. Artificial Neural Networks (ESANN),* pp. 519-524, 2004.

[26] T. De Bie, C.-L. Tranchevent, L.M.M. van Oeffelen, and Y. Moreau, "Kernel-Based Data Fusion for Gene Prioritization," *Bioinformatics,* pp. i125-i132, 2007.

[27] N. Hansen, S.D. Müller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation,* vol. 11, pp. 1-18, 2003.

[28] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy with Increasing Population Size," *Proc. IEEE Congress Evolutionary Computation (CEC '05),* vol. 2, pp. 1769-1776, 2005.

[29] F. Bach, G.R.G. Lanckriet, and M.I. Jordan, "Multiple Kernel Learning, Conic Duality, and the SMO Algorithm," *Proc. 21st Int'l Conf. Machine Learning,* 2004.

[30] F. Bach, R. Thibaux, and M.I. Jordan, "Computing Regularization Paths for Learning Multiple Kernels," *J. Machine Learning Research,* vol. 5, pp. 1391-1415, 2004.

[31] H.W. Mewes, K. Heumann, A. Kaps, K. Mayer, F. Pfeiffer, S. Stocker, and D. Frishman, "MIPS: A Database for Genomes and Protein Sequences," *Nucleic Acids Research,* vol. 28, pp. 37-40, 2000.

[32] A. Ben-Hur and W.S. Noble, "Choosing Negative Examples for the Prediction of Protein-Protein Interactions," *BMC Bioinformatics,* vol. 7, no. 1, 2006.

[33] J. Gollub et al., "The Stanford Microarray Database: Data Access and Quality Assessment Tools," *Nucleic Acids Research,* vol. 31, no. 1, pp. 94-96, 2003.

[34] T.R. Hughes et al., "Functional Discovery via a Compendium of Expression Profiles," *Cell,* vol. 102, pp. 109-126, 2000.

[35] S.L. Tai, V.M. Boer, P. Daran-Lapujade, M.C. Walsh, J.H. de Winde, J.-M. Daran, and J.T. Pronk, "Two-Dimensional Transcriptome Analysis in Chemostat Cultures: Combinatorial Effects of Oxygen Availability and Macronutrient Limitation in Saccharomyces Cerevisiae," *J. Biological Chemistry,* vol. 280, no. 1, pp. 437-447, 2005.

[36] M.J. Dunham, H. Badrane, T. Ferea, J. Adams, P.O. Brown, F. Rosenzweig, and D. Botstein, "Characteristic Genome Rearrangements in Experimental Evolution of Saccharomyces Cerevisiae," *Proc. Nat'l Academy of Sciences USA,* vol. 99, pp. 16144-16149, 2002.

[37] J.D. Lieb, L. Xiaole, D. Botstein, and P.O. Brown, "Promoter-Specific Binding of Rap1 Revealed by Genome-Wide Maps of Protein-DNA Association," *Nature Genetics,* vol. 28, pp. 327-334, 2001.

[38] C.T. Harbison et al., "Transcriptional Regulatory Code of a Eukaryotic Genome," *Nature,* vol. 431, pp. 99-104, Sept. 2004.

[39] N.J. Krogan et al., "High-Definition Macromolecular Composition of Yeast RNA-Processing Complexes," *Molecular Cell,* vol. 13, pp. 225-239, 2004.

[40] I. Xenarios, L. Salwinski, X.J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, "DIP, the Database of Interacting Proteins: A Research Tool for Studying Cellular Networks of Protein Interactions," *Nucleic Acids Research,* vol. 30, no. 1, pp. 303-305, 2002.

[41] L.J. Lu, Y. Xia, A. Paccanaro, H. Yu, and M. Gerstein, "Assessing the Limits of Genomic Data Integration for Predicting Protein Networks," *Genome Research,* vol. 15, no. 7, pp. 945-953, 2005.

[42] H. Yu, N.M. Luscombe, H.X. Lu, X. Zhu, Y. Xia, J.-D.J. Han, N. Bertin, S. Chung, M. Vidal, and M. Gerstein, "Annotation Transfer between Genomes: Protein-Protein Interologs and Protein-dna Regulogs," *Genome Research,* vol. 14, no. 6, pp. 1107-1118, 2004.

[43] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machine," http://www.csie.ntu.edu.tw/cjlin/libsvm, 2008.

[44] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, and D.M.J. Tax, "PRTools, A Matlab Toolbox for Pattern Recognition," http://prtools.org, 2009.

[45] C.-W. Hsu, C.-C. Chang, and L. Chih-Jen, "A Practical Guide to Support Vector Classification," technical report, Dept. of Computer Science and Information Eng., Nat'l Taiwan Univ., 2003.

[46] D.P. Lewis, T. Jebara, and W.S. Noble, "Support Vector Machine Learning from Heterogeneous Data: An Empirical Analysis Using Protein Sequence and Structure," *Bioinformatics,* vol. 22, pp. 2753-2760, 2006.

[47] S.L. Wong et al., "Combining Biological Networks to Predict Genetic Interactions," *Proc. Nat'l Academy of Sciences USA,* vol. 101, pp. 15682-15687, 2004.

[48] W.-C. Kao, K.-M. Chung, C.-L. Sun, and C.-J. Lin, "Decomposition Methods for Linear Support Vector Machines," *Neural Computation,* vol. 16, no. 8, pp. 1689-1704, 2004.

[49] T. Joachims, "A Support Vector Method for Multivariate Performance Measures," *Proc. 22nd Int'l Conf. Machine Learning (ICML '05),* pp. 377-384, 2005.

Marc Hulsman received the MSc degree in bioinformatics in 2007 from Delft University of Technology, with a thesis on the use of kernel combination for protein comembership prediction. He is currently pursuing the PhD degree in the Bioinformatics Group at the Faculty of Electrical Engineering, Mathematics, and Computer Science of Delft University of Technology, working for the COBIOS project (a European project to construct synthetic biological oscillators). His current research focuses on the normalization and integration of large-scale biological data sets, in order to construct recommendation systems supporting the building and analysis of synthetic biological networks.

Marcel J.T. Reinders received the MSc degree in applied physics and the PhD degree in electrical engineering from Delft University of Technology, The Netherlands, in 1990 and 1995, respectively. In 2004, he became a professor of bioinformatics within the Mediamatics Department of the Faculty of Electrical Engineering, Mathematics, and Computer Science at Delft University of Technology. His background is within pattern recognition. Besides studying fundamental issues, he applies pattern recognition techniques to the areas of bioinformatics, computer vision, and context-aware recommender systems. His special interest goes toward understanding complex systems (such as biological systems) that are severely undersampled.

Dick de Ridder received the MSc degree in computer science and the PhD degree in applied physics from Delft University of Technology, The Netherlands, in 1996 and 2001, respectively. In 2005, he became an assistant professor in bioinformatics at the Faculty of Electrical Engineering, Mathematics, and Computer Science of Delft University of Technology. In the past, he has worked on pattern recognition, neural networks, and image processing, specifically on nonlinear feature extraction algorithms. Currently, his interest is in the development of algorithms to integrate various high-throughput measurements and prior knowledge to model the living cell. These algorithms find applications in medical research, biotechnology, and systems biology.