FORMATICS

Vol. 21 Suppl. 1 2005, pages i47–i56
doi:10.1093/bioinformatics/bti1007

Protein function prediction via graph kernels

Karsten M. Borgwardt[1,*], Cheng Soon Ong[2], Stefan Schönauer[1], S. V. N. Vishwanathan[2], Alex J. Smola[2] and Hans-Peter Kriegel[1]

[1]Institute for Computer Science, Ludwig-Maximilians-University Munich, Oettingenstraße 67, 80538 Munich, Germany and [2]National ICT Australia, Canberra, 0200 ACT, Australia

Received on January 15, 2005; accepted on March 27, 2005

Computational approaches to protein function
r protein function by finding proteins with sim-
structure, surface clefts, chemical properties,
otifs, interaction partners or phylogenetic pro-
ent a new approach that combines sequential,
chemical information into one graph model of
redict functional class membership of enzymes
mes using graph kernels and support vector
ification on these protein graphs.
graph model, derivable from protein sequence
only, is competitive with vector models that
nal protein information, such as the size of
s. If we include this extra information into our
our classifier yields significantly higher accuracy
vector models. Hyperkernels allow us to select
y combine the most relevant node attributes in
aphs. We have laid the foundation for a protein
tion system that integrates protein information
ources efficiently and effectively.
More information available via www.dbs.ifi.lmu.
borgwardt.html.
wardt@dbs.ifi.lmu.de

UCTION

the molecular mechanisms of life requires the
e functions of proteins in an organism. Tens of
roteins have been sequenced over recent years,
res of thousands of proteins have been resolved
n *et al*., 2000). Still, the experimental determ-
function of a protein with known sequence and

known function is consequently the basis of current function prediction (Whisstock and Lesk, 2003). A newly discovered protein is predicted to exert the same function as the most similar proteins in a database of known proteins. This similarity among proteins can be defined in a multitude of ways: two proteins can be regarded to be similar, if their sequences align well [e.g. PSI-BLAST (Altschul *et al*., 1997)], if their structures match well [e.g. DALI (Holm and Sander, 1996)], if both have common surface clefts or bindings sites [e.g. CASTp (Binkowski *et al*., 2003)], similar chemical features or common interaction partners [e.g. DIP (Xenarios *et al*., 2002)], if both contain certain motifs of amino acids (AAs) [e.g. Evolutionary Trace (Yao *et al*., 2003)] or if both appear in the same range of species (Pellegrini *et al*., 1999). An armada of protein function prediction systems that measure protein similarity by one of the conditions above has been developed. Each of these conditions is based on a biological hypothesis; e.g. structural similarity implies that two proteins could share a common ancestor and that they both could perform the same function as this common ancestor (Bartlett *et al*., 2003).

These assumptions are not universally valid. Hegyi and Gerstein (1999) showed that proteins with similar function may have dissimilar structures and proteins with similar structures may exert distinct functions. Furthermore, a single AA mutation can alter the function of a protein and make a pair of structurally closely related proteins functionally different (Wilks *et al*., 1988). Exceptions are also numerous if similarity is measured by means other than structure (Whisstock and Lesk, 2003). Due to these exceptions, none of the existing function prediction systems can guarantee generally good

methods and support vector machines

ds are a popular method for machine learn-
of and Smola, 2002). This paper uses kernel
cifically support vector machines (SVMs), to
in function prediction. We denote by $\mathcal{X}$ the space
the proteins) and by $\mathcal{Y}$ the space of labels (their
$:= \{x_1, \ldots, x_m\}$ denotes the training data and
$y_m\}$ a set of corresponding labels, jointly drawn
and identically from some probability distribu-
$\mathcal{X} \times \mathcal{Y}$. For a new example $x \in \mathcal{X}$, the problem
e label y using our prior knowledge of the prob-
aining examples. Observe that we do not know
ence the algorithm has to perform predictions
nformation provided by the training data.

ods have been highly successful in solving vari-
in machine learning. The algorithms work by
pping the inputs into a feature space and finding
othesis in this new space. The feature map $\phi(\cdot)$
defined by a kernel function k, which allows us
t products in feature space using only objects in
e, i.e. $k(x_i, x_j) := \langle\phi(x_i), \phi(x_j)\rangle$. The kernel
be positive definite for the SVM. Examples of
te kernels are the Dirac, Gaussian and Brownian
(Schölkopf and Smola, 2002).

ased on finding a good linear hypothesis in this
(Cortes and Vapnik, 1995). More specifically,
the hyperplane which maximizes the margin in
thereby aiming at separating different classes
points in feature space. The margin is the max-
between a training example in feature space and
hyperplane. The C-SVM we use in this paper
e 'soft margin', where instead of disallowing
from being misclassified, we penalize misclas-
g a linear cost. Figure 1 shows a toy example
margin SVM was used for classification. SVMs
e of a convex optimization problem (Boyd and
, 2004). Efficient algorithms exist for solving
ms, which means that large-scale problems can

in Biology

of SVM classification in molecular biology
and the importance of kernel methods for
is steadily growing (Schölkopf *et al.*, 2004).

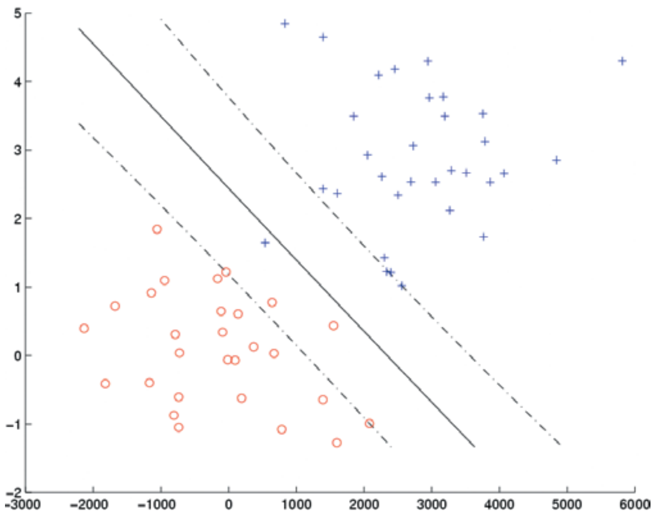


Fig. 1. The C-SVM maximizes the margin between the training examples and the hyperplane. The solid line denotes the separating hyperplane and the dashed line denotes the margin. Plus (+) and circle (o) data points represent two distinct classes of input data.

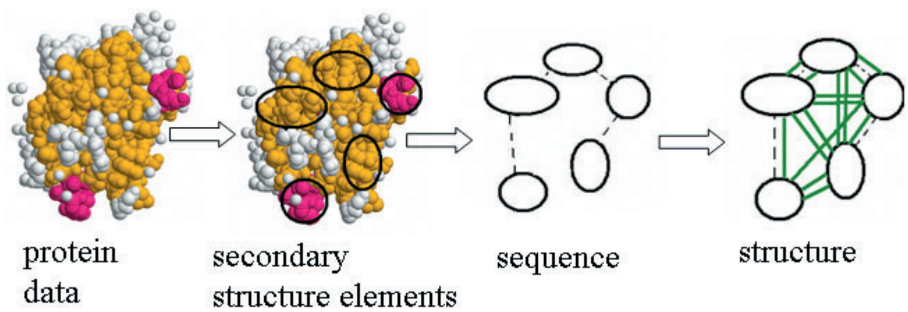
feature vectors. Both then perform SVM classification on these feature vectors to predict protein function.

Despite the success of SVMs in biology, their application is almost always connected with a transformation of structured biological data into a simplified feature vector description. As a result, even a complex protein structure is represented by vector components that summarize detailed information into one simplified total value. To avoid this loss of information, GRATH (Harrison *et al.*, 2002) and SSM (Krissinel and Henrick, 2003) represent protein structures as graphs of secondary structure elements (SSEs) and then perform graph-matching algorithms to measure structural similarity. Our target was therefore to design a kernel function for a graph model of proteins that still allows us to perform SVM classification.

In short, in our project we aimed at the following goals: to model proteins using graphs, which is the most adequate data structure, to include sequence and chemical information into the model, and to classify proteins—based on this model—into their correct functional class.

2 APPROACH

In this section, we design a graph model for proteins, in which



tic illustration of graph generation from PDB protein file (Berman *et al*., 2000) (circles, SSEs; thin dashed lines, sequential
d lines, structural edges).

r to labels as attributes. In our case, attributes
pairs of the form (attribute-name, value).
cy matrix A of G is defined as

$$[A]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases}$$

$_j$ are nodes in G. A walk of length $k-1$ in a
ence of nodes $v_1, v_2, \ldots, v_k$ where

$$(v_{i-1}, v_i) \in E \quad \textit{for } 1 < i \leq k.$$

structure of proteins We design our graph
tain information about structure, sequence and
erties of a protein. For this purpose, we model
ibuted and undirected graphs. Each graph rep-
one protein. Nodes in our graph represent SSEs
tein structure, i.e. helices, sheets and turns.
t nodes if those are neighbors along the AA
they are neighbors in space within the protein
y node is connected to its three nearest spatial

a type label, stating whether they represent a
turn, and physical and chemical information,
drophobicity, the van der Waals volume, the
larizability of the SSE represented by this node.
nalized van der Waals value is determined for
ividually. Additionally, each node is labeled
number of its residues with low, medium or
ed van der Waals volume separately; we will

between their centers, where the center of an SSE is the midpoint of the line between the C_α atom of its first and the C_α atom of its last residue.

2.1.2 Graph generation We generate our protein graphs from protein files of the protein data bank (PDB) (Berman *et al*., 2000) (Fig. 2), except for the chemical and physical node attributes. We assign these to SSEs using AA indices from the Amino Acid Index Database (Kawashima *et al*., 1999), i.e. tables with one value for each AA characterizing a chemical or physical feature of this AA. Normalized AA indices for hydrophobicity (Cid *et al*., 1992), van der Waals volume (Fauchere *et al*., 1988), polarity (Grantham, 1974) and polarizability (Charton and Charton, 1982) are applied to the sequence of each SSE node to derive one total value and one 3-bin distribution each.

2.2 Random walk graph kernel

Using the attributed graphs model of proteins as defined in the previous section, we define a kernel that measures the similarity between two protein graphs. We tested several graph kernels, of which a graph kernel based on random walks turned out to be most successful. For the sake of brevity, we present this kernel and its best parameterization only; a technical report on the accompanying homepage describes two other protein kernels.

Random walk kernels were proposed by Kondor and Lafferty (2002), Cortes *et al*. (2003), Gärtner *et al*. (2003) and Kashima *et al*. (2003). Given two labeled graphs G_1 and G_2, a random walk kernel counts the number of matching roach by Gärtner *et al*. (2003) for calculating all within two graphs uses direct product graphs:

1 (*Direct product graph*). *The direct product graphs $G_1 = (V, E)$ and $G_2 = (W, F)$ shall $G_1 \times G_2$. The node and edge set of the direct are respectively defined as:*

$$\begin{aligned} G_2) = &\{(v_1, w_1) \in V \times W : \\ &(label(v_1) = label(w_1))\}, \\ G_2) = &\{((v_1, w_1), (v_2, w_2)) \in V^2(G_1 \times G_2) : \\ &(v_1, v_2) \in E \wedge (w_1, w_2) \in F \\ &\wedge (label(v_1, v_2) = label(w_1, w_2))\}. \end{aligned}$$

direct product graph, the random walk kernel is

2 (*Random walk kernel*). *Let G_1, G_2 be two $_\times$ denote the adjacency matrix of their direct $= A(G_1 \times G_2)$, and let $V_\times$ denote the node set product. With a weighting factor $\lambda \geq 0$ the graph kernel is defined as*

$$_\times(G_1, G_2) = \sum_{i,j=1}^{V_\times} \left[\sum_{n=0}^{\infty} \lambda^n A_\times^n \right]_{ij}.$$

edges in graph $G_1 \times G_2$ have the same labels onding nodes and edges in G_1 and G_2. Random h n are weighted by λ^n in the sum over all walks. be chosen carefully for the sum to converge. In simplify the approach, we calculate the random r walks up to a predetermined length only.

n graph kernel

rnel defined in the previous section is designed tributes: Attributes of two nodes v_1 and w_1 are milar if they are completely identical, i.e. they via a Dirac kernel. The nodes in our protein n continuous attributes which are almost never entical between two nodes. For that reason, we

$\{1, \ldots, n-1\}$. *The walk kernel will now be defined as*

$$k_{walk}(walk_1, walk_2) = \prod_{i=1}^{n-1} k_{step}((v_i, v_{i+1}), (w_i, w_{i+1})).$$

As above, the modified random walk graph kernel is then the sum over all kernels on pairs of walks in two input graphs. It can be computed as in Definition 2 if we modify the definition of the adjacency matrix of the direct product graph such that

$$[A_\times]_{((v_i,w_i),(v_j,w_j))} = \begin{cases} k_{step}((v_i, v_j), (w_i, w_j)) \\ \quad \textit{if } ((v_i, v_j), (w_i, w_j)) \in E_\times, \\ 0 \quad \textit{otherwise} \end{cases}$$

with $E_\times = E_\times(G_1 \times G_2)$ and $(v_i, v_j) \in E$ and $(w_i, w_j) \in F$.

We define the kernel for each step in the random walk in terms of the original node, the destination node and the edge between them.

DEFINITION 4 (*Step kernel*). *For $i \in \{1, \ldots, n-1\}$, the step kernel is defined as*

$$\begin{aligned} &k_{step}((v_i, v_{i+1}), (w_i, w_{i+1})) \\ &\quad = k_{node}(v_i, w_i) * k_{node}(v_{i+1}, w_{i+1}) \\ &\quad\quad * k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1})), \end{aligned}$$

where k_{edge} is defined as

$$\begin{aligned} &k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1})) \\ &\quad = k_{type}((v_i, v_{i+1}), (w_i, w_{i+1})) \\ &\quad\quad * k_{length}((v_i, v_{i+1}), (w_i, w_{i+1})) \end{aligned}$$

and for $i \in \{1, \ldots, n\}$, k_{node} is defined as

$$\begin{aligned} &k_{node}(v_i, w_i) \\ &\quad = k_{type}(v_i, w_i) * k_{node\ labels}(v_i, w_i) * k_{length}(v_i, w_i). \end{aligned}$$

The matching between nodes and edges is therefore defined via three basic types of kernels: type kernels, length kernels and node labels kernels, which we explain and define in the following.

2.3.1 Type kernel Identical motifs of SSEs both within

5 (*Type kernel*). k_{type} *is defined identically for* *d edges x and* x'*:*

$$(x, x') = \begin{cases} 1 & \textit{if}\ \mathrm{type}(x) = \mathrm{type}(x'), \\ 0 & \textit{otherwise}. \end{cases}$$

kernel Length kernels ensure that we do not edges as being similar if they differ a lot in size. eletion of AA residues might change the length r distance towards each other, while the overall ion of the protein remains unchanged. For this ployed the Brownian bridge kernel, that assigns rnel value to SSEs and edges that are identical ssigns zero to all SSEs and edges that differ in an by a constant c. This maximum difference s set to 2 AA for sequential edges, to 2 Å for es and to 3 Å for SSE nodes.

6 (*Length kernel*). k_{length} *is defined identically and edges x and* x'*, except for the value of c:*

$$') = max(0, c - |length(x) - length(x')|).$$

labels kernel We compare the physico-res of two SSEs via a node labels kernel. We nel to be Gaussian, since these have shown the nce in related studies (Cai *et al.*, 2004); σ was oss-validation.

7 (*Node labels kernel*). *The node labels kernel Gaussian kernel over two vectors representing ll labels of node x and node* x'*:*

$$, x') = \exp\left(-\frac{\|labels(x) - labels(x')\|^2}{2\sigma^2}\right).$$

l to show that this modified graph kernel is still e definite kernel.

The modified random walk graph kernel is e.

e type kernel is a Dirac kernel, the length ker-an bridge kernel and the node labels kernel rnel; these kernels are known to be positive

The positive definiteness of the modified random walk kernel follows directly from its definition as a convolution kernel, proven to be positive definite by Haussler (1999).

Computing a kernel matrix entry for our protein graph kernel may seem expensive, as kernel functions on all nodes and edges have to be evaluated. The high selectiveness of length and type kernel, however, which set many step kernel values to zero, can be exploited to reduce computational costs, thereby enhancing speed and scalability. Computation of the graph kernel matrix scales linearly with the number of its entries. For efficient and scalable SVM training, one can use low rank representations (Fine and Scheinberg, 2001).

2.4 Hyperkernels for choice of best kernel

Our protein random walk graph kernel consists of a combination of a multitude of kernels on a multitude of graph attributes. We are interested in how to optimally combine these kernels on graph attributes as choosing a suitable graph kernel function is imperative to the success of our classifier and function prediction system. Lanckriet *et al*. (2004) showed that kernel learning can be used to combine different data sources for protein function prediction in yeast to yield a joint kernel that performs better than any kernel on a single type of data. One systematic technique which can assist in learning kernels are hyperkernels (Ong *et al*., 2003; Ong and Smola, 2003), which use the idea of defining a kernel on the space of kernels itself. We 'learn' this kernel by defining a quantity analogous to the risk functional, called the quality functional, which measures the 'badness' of the kernel function. The purpose of this functional is to indicate the quality of a given kernel for explaining the training data at hand. Given a set of input data and their associated labels, and a class of kernels $\mathcal{K}$, we would like to select the best kernel $k \in \mathcal{K}$ for the problem. However, if provided with a sufficiently rich class of kernels $\mathcal{K}$, it is in general possible to find a kernel that overfits the data. Therefore, we would like to control the complexity of the kernel function. We achieve this by using the kernel trick again on the space of kernels. This so called hyperkernel $\underline{k}$ defines an associated hyper reproducing kernel hilbert space (hyper-RKHS) $\underline{\mathcal{H}}$. This allows for simple optimization algorithms which consider kernels k in the hyper-RKHS $\underline{\mathcal{H}}$, which are in the convex cone of $\underline{k}$. Analogous to the regularized risk functional, $R_{\mathrm{reg}}(f, X, Y) = (1/m)\sum_{i=1}^{m} l(x_i, y_i, f(x_i)) + (\lambda/2)\|f\|^2$,

The minimizer of Equation (1) satisfies the
eorem:

(*Representer theorem*). *Denote by $\mathcal{X}$ a set,
arbitrary quality functional. Then each minim-
the regularized quality functional 1, admits a
of the form*

$$x, x') = \sum_{i,j=1}^{m} \beta_{i,j} \underline{k}((x_i, x_j), (x, x')). \quad (2)$$

that even though we are optimizing over a whole
of kernels, we still are able to find the optimal
oosing among a finite number, which is the span
n the data.

nidefinite programming (SDP) formulations of
on problems arising from the minimization of
d quality functional (Ong and Smola, 2003).
timization of a linear objective function subject
which are linear matrix inequalities and affine

tion, we define the following notation. For
$n \in \mathbb{N}$ let $r = p \circ q$ be defined as element
ultiplication, $r_i = p_i \times q_i$. The pseudo-inverse
enrose inverse) of a matrix K is denoted by
he hyperkernel Gram matrix $\underline{K}$ by $\underline{K}_{ijpq} =$
$, x_q))$, the kernel matrix $K = \text{reshape}(\underline{K}\beta)$
m^2 by 1 vector, $\underline{K}\beta$, to an $m \times m$ matrix),
a matrix with y on the diagonal and zero other-
$= YKY$ (the dependence on β is made explicit)
of ones.

of training examples is assumed to be m. Where
is a Lagrange multiplier, while η and ξ are
grange multipliers from the derivation of the
r the SDP, β are the hyperkernel coefficients, t_1
auxiliary variables.

(Linear SVM (C-style)). A commonly used
r classifier, the C-SVM uses an ℓ_1 soft mar-
$x_i, y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$, which
on the training set. The parameter C is given
etting the quality functional $Q_{\text{emp}}(k, X, Y) =$
$) \sum_{i=1}^{m} l(x_i, y_i, f(x_i)) + (1/2C)\|w\|^2_{\mathcal{H}}$, the res-

$$\min \quad \frac{1}{2}t_1 + \frac{C}{m}\xi^\top \mathbf{1} + \frac{C\lambda_Q}{2}t_2$$

We apply hyperkernels in Section 3.2 in two ways: first to combine the various attribute kernels in an optimal fashion and second to investigate the weights of the various attributes. From the representer Theorem 1, the kernels on various attributes are weighted in the final optimal kernel, and hence the weights reflect the importance of that particular attribute for protein function prediction. The higher the weight of the kernel of an attribute in the final linear combination, the more important it is for good prediction accuracy. Similar to Ong and Smola (2003), we use a low rank approximation for our optimization problem, hence resulting in a scalable implementation. The computational cost is a constant factor larger than a standard SVM, where the constant is determined by the precision of the low rank approximation.

3 RESULTS

To assess the protein function prediction quality of our graph kernels, we tested them on two function prediction problems: classifying enzymes versus non-enzymes, and predicting the enzyme class.

Experimental setting. For the following experiments, we implemented our graph model and kernel in MATLAB® R13, and employed the SVM package SVLAB. We ran our tests on Debian Linux workstations with Intel Pentium 4® CPU at 3.00 GHz.

3.1 Enzymes versus non-enzymes

In our first test, we classified enzymes versus non-enzymes. Our dataset comprised proteins from the dataset of enzymes (59%) and non-enzymes (41%) created by Dobson and Doig (2003). Protein function prediction on this set of proteins is particularly difficult, as Dobson and Doig chose proteins such that no chain in any protein aligns to any other chain in the dataset with a Z-score of 3.5 or above outside of its parent structure.

Dobson and Doig model proteins as feature vectors which indicate for each AA its fraction among all residues, its fraction of the surface area, the existence of ligands, the size of the largest surface pocket and the number of disulphide bonds.

On the complete dataset, Dobson and Doig had reached 76.86% accuracy in 10-fold cross-validation, on an optim-

y of prediction of functional class of enzymes and non-
d cross-validation with C-SVM

	Accuracy	SD
	76.86	1.23
kernel	80.17	1.24
	77.30	1.20
out structure	72.33	5.32
global info	84.04	3.33
	75.07	4.58

are the results obtained by Dobson and Doig (2003). 'Graph kernel'
defined as in Section 2.3, 'Graph kernel without structure' is the
rotein models without structural edges, 'Graph kernel with global
raph kernel plus additional global node labels. 'DALI classifier' is
assifier on DALI Z-scores.

a comparison, we implemented and ran a
or classifier based on DALI Z-scores (Holm
96) on the same dataset.
show that our graph kernel is competitive with
ctor kernel approach, although it relies on less
an the vector approach. Our graph model can
from sequence and structure, while the vector
s additional information about ligands, surface
ds of the proteins in question. Furthermore, our
lso gives better results than the DALI classi-
based on state-of-the-art structure comparison

ts suggest two further experiments: first, to
we can reach similarly good results if we do not
ural edges into our protein model. This kind of
uld be generated without knowing the structure
lying solely on the sequence and on a secondary
ction system. We tested our kernel on graphs
ural edges and found a significant deterioration
diction accuracy (Table 1).
tested whether our protein classifier could be
ncorporating Dobson and Doig's extra inform-
nded our protein graphs to include additional
node labels. These global node attributes are
ll nodes in one graph; they represent the exist-
s, the number of disulphide bonds, the size of
face pocket (Binkowski *et al.*, 2003) and the
h AA type on the protein surface (Tsodikov

3.2 Enzyme class prediction

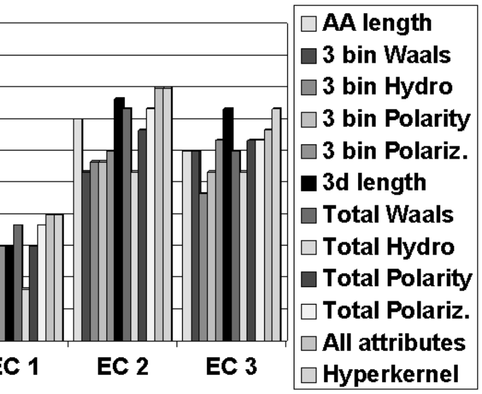
After showing that our graph classifier reaches at least state-of-the-art prediction accuracy, we examined which of our 10 local node attributes contribute most to successful classification. The standard approach to this problem is to define kernels on individual node attributes and to then test the performance of these kernels on a test set. Attributes whose kernels show best classification accuracy in these tests are then deemed to be most important for good prediction accuracy.

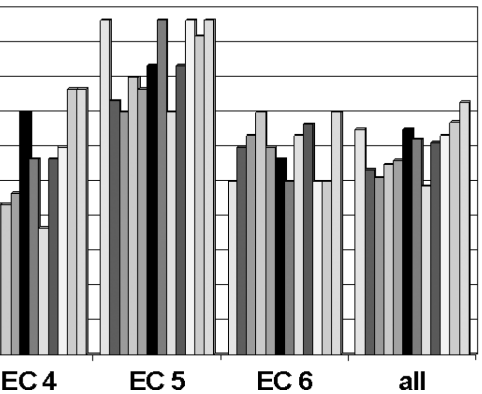
We propose to employ hyperkernels for selecting relevant node attributes. The hyperkernel finds a linear combination of kernels defined on single node attributes that maximizes prediction accuracy. Node attributes receiving highest weight in the hyperkernel optimal combination can then be regarded as most valuable for correct classification.

For that purpose, we created protein graph models with only one of our 10 node attributes, each for a dataset of 600 enzymes from the BRENDA database (Schomburg *et al.*, 2004). This dataset included 100 proteins from each of the 6 Enzyme Commission top level enzyme classes (EC classes) and the goal was to correctly predict enzyme class membership for these proteins. We computed protein graph kernel matrices (defined as in Section 2.3) on these single attribute models, normalized them and employed a hyperkernel to find an optimal linear combination of these 10 normalized kernel matrices. As a comparison, we also ran our default protein graph kernel with all node attributes on the same dataset.

For each EC class, we conducted 1-versus-rest SVM classification for all our kernels and the hyperkernel, in 6-fold cross-validation on all 600 proteins. As the number of non-members of an EC class is five times that of the members in both training and test set, a naive classifier predicting all enzymes to be non-EC-class-members would always yield 83.33% accuracy. We report classification results in Figure 3 and hyperkernel weights in the optimal linear combination in Table 2.

Our results show that with each of the kernels employed, we are able to correctly predict enzyme class membership and non-membership with a high accuracy level of at least 90.83% on average. On average the hyperkernel performs best of all kernels. Across all EC classes, the hyperkernel reaches at least the accuracy of the best individual kernel.





ion accuracy using kernel matrices on individual kernel on all attributes and the hyperkernel 6-fold cross-validation on 600 enzymes from 6 EC s (AA, amino acid; Waals, van der Waals volume; hobicity; Polariz, Polarizability).

ernel weights for individual node attributes

	EC1	EC2	EC3	EC4	EC5	EC 6
	1.00	**0.31**	**1.00**	**1.00**	**0.73**	0.00
	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.01	0.00	0.00	0.00	**1.00**
	0.00	0.00	0.00	0.00	**0.12**	0.00
	0.00	**0.40**	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	**0.13**	0.00	0.00	0.01	0.00
	0.00	**0.14**	0.00	0.00	0.01	0.00

4 DISCUSSION

In this paper, we presented a graph model for proteins and defined a protein graph kernel that measures similarity between these graphs. Based on this protein graph model and kernel, we implemented a SVM classifier for protein function prediction. We successfully tested the performance of this classifier on two function prediction tasks.

Our graph model includes information about sequence, structure and chemical properties, with nodes that represent SSEs and edges that represent sequential or spatial neighborship between these elements. Graph models based on smaller subunits of proteins, AA residues or atoms, might give a more detailed description of the chemical properties of a protein, yet they would lead to graphs with at least 10 times or 100 times more nodes, respectively. As the number of node comparisons for a pair of proteins grows quadratically with the number of nodes, enormous computational costs would be the results of more detailed models. For this reason, we developed a protein model based on SSEs.

Our graph kernel measures structural, sequential and chemical similarities between two proteins. We designed the graph kernel to first detect structural and sequential similarities between proteins and if these are found, to then measure the degree of similarity by comparing physico-chemical properties of their SSEs. Combining these three types of similarity measures into one graph kernel allows us to distinguish enzymes and non-enzymes on the same accuracy level as a vector kernel method requiring additional information and a DALI classifier based on Z-scores; our kernel outperforms both if we use a protein graph model including all extra information used by the vector kernel approach. We conclude that our model is able to capture essential characteristics of proteins that define their function. Furthermore, we showed that structure information is beneficial for our classifier, as removing structural edges from our graphs decreases prediction accuracy significantly.

We successfully applied the hyperkernel technique to the question of how to choose relevant node attributes in our protein graphs and of how to combine these optimally. Consequently, hyperkernels are a useful tool to further optimize our graph model by weighing the importance of individual node attributes for correct classification.

The hyperkernel assigns on average highest weightage to the node attribute AA length. Functional similarity between ith other proteomic information to improve our

will aim at refining our protein graph model
e node and edge labels and at integrating more
ation into our classifier to make function pre-
accurate. Attributed graphs, our protein graph
perkernels will be essential for this process of

EDGEMENTS

supported in part by the German Ministry for
ence, Research and Technology (BMBF) under
112F within the BFAM (Bioinformatics for the
alysis of Mammalian Genomes) project which
German Genome Analysis Network (NGFN).
Australia is funded through the Australian
Backing Australia's Ability initiative, in part
ustralian Research Council.

CES

Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z.,
d Lipman,D.J. (1997) Gapped BLAST and PSI-
w generation of protein database search programs.
s Res., **25**, 3389–3402.
oworth,D., Brenner,S.E., Hubbard,T.J., Chothia,C.
.G. (2004) Scop database in 2004: refinements integ-
and sequence family data. *Nucleic Acids Res.*, **32**,

odd,A.E. and Thornton,J.M. (2003) Inferring pro-
from structure. In Bourne,P.E. and Welssig,H.
ural Bioinformatics. Wiley-Liss, Inc., New York,

Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N.,
hindyalov,I.N. and Bourne,P.E. (2000) The protein
ucleic Acids Res., **28**, 235–242.
Naghibzadeh,S. and Liang,J. (2003) Castp: com-
surface topography of proteins. *Nucleic Acids Res.*,
5.
Vandenberghe,L. (2004) *Convex Optimization*.
niversity Press.
.Y., Ji,Z.L. and Chen,Y.Z. (2004) Enzyme family
by support vector machines. *Proteins*, **55**, 66–76.
W.L., Sun,L.Z. and Chen,Y.Z. (2003) Protein func-
ation via support vector machine approach. *Math.*

Cortes,C. and Vapnik,V. (1995) Support vector networks.*Machine Learning*, **20**, 273–297.

Dobson,P.D. and Doig,A.J. (2003) Distinguishing enzyme structures from non-enzymes without alignments. *J. Mol. Biol.*, **330**, 771–783.

Fauchere,J.L., Charton,M., Kier,L.B., Verloop,A. and Pliska,V. (1988) Amino acid side chain parameters for correlation studies in biology and pharmacology. *Int. J. Pept. Protein Res.*, **32**, 269–278.

Fine,S. and Scheinberg,K. (2001) Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.*, **2**, 243–264.

Gärtner,T., Flach,P. and Wrobel,S. (2003) On graph kernels: hardness results and efficient alternatives. In Schölkopf,B. and Warmuth,M.K. (eds), *Proceedings of the 16th Annual Conference on Learning Theory*, pp. 129–143.

Grantham,R. (1974) Amino acid difference formula to help explain protein evolution. *Science*, **185**, 862–864.

Harrison,A., Pearl,F., Mott,R., Thornton,J. and Orengo,C. (2002) Quantifying the similarities within fold space. *J. Mol. Biol.*, **323**, 909–926.

Haussler,D. (1999) Convolutional kernels on discrete structures. *Technical Report UCSC-CRL-99-10*, Computer Science Department, University of California, Santa Cruz CA.

Hegyi,H. and Gerstein,M. (1999) The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *J. Mol. Biol.*, **288**, 147–164.

Holm,L. and Sander,C. (1996) Mapping the protein universe. *Science*, **273**, 595–602.

Kashima,H., Tsuda,K. and Inokuchi,A. (2003) Marginalized kernels between labeled graphs. *Proceedings of ICML*, Washington, DC, pp. 321–328.

Kawashima,S., Ogata,H. and Kanehisa,M. (1999) Aaindex: amino acid index database. *Nucleic Acids Res.*, **27**, 368–369.

Kondor,R.S. and Lafferty,J. (2002) Diffusion kernels on graphs and other discrete structures. *Proceedings of ICML*, Sydney, Australia, pp. 325–322.

Krissinel,E. and Henrick,K. (2003) Protein structure comparison in 3D based on secondary structure matching (SSM) followed by C_α alignment, scored by a new structural similarity function. In Kungl,A.J. and Kung,P.J. (eds), *Proceedings of the 5th International Conference on Molecular Structural Biology*, Vienna, pp. 88.

Lanckriet,G.R.G, De Bie,T., Cristianini,N., Jordan,M.I. and Noble,W.S. (2004) A statistical framework for genomic data fusion.*Bioinformatics*, **20**, 2626–2635.

Ong,C.S. and Smola,A.J. (2003) Machine learning with hyperkernels. *Proceedings of ICML*, Washington, DC, pp. 568–575.

Ong,C.S., Smola,A.J. and Williamson,R.C. (2003) Hyper-

Tsuda,K. and Vert,J.P. (2004) *Kernel Methods in* *al Biology*. MIT Press, Cambridge, MA.

Chang,A., Ebeling,C., Gremse,M., Heldt,C., Huhn,G. ourg,D. (2004) BRENDA, the enzyme database: major new developments. *Nucleic Acids Res.*, **32**,

Record,M.T.Jr. and Sergeev,Y.V. (2002) A novel ogram for fast exact calculation of accessible and rface areas and average surface curvature. *J. Comput.* 00–609.

and Lesk,A.M. (2003) Prediction of protein funcotein sequence and structure. *Q. Rev. Biophys.*, **36**,

Wilks,H.M., Hart,K.W., Feeney,R., Dunn,C.R., Muirhead,H., Chia,W.N., Barstow,D.A., Atkinson,T., Clarke,A.R. and Holbrook,J.J. (1988) A specific, highly active malate dehydrogenase by redesign of a lactate dehydrogenase framework. *Science*, **242**, 1541–1544.

Xenarios,I., Salwinski,L., Duan,X., Higney,P., Kim,S.M. and Eisenberg,D. (2002) Dip, the database of interacting proteins: a research tool for studying cellualr networks of protein interactions. *Nucleic Acids Res.*, **30**, 303–305.

Yao,H., Kristensen,D.M., Mihalek,I., Sowa,M.E., Shaw,C., Kimmel,M., Kavraki,L. and Lichtarge,O. (2003) An accurate, sensitive, and scalable method to identify functional sites in protein structures. *J. Mol. Biol.*, **326**, 255–261.