BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–10

Protein function prediction via graph kernels

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

[a] *Institute for Computer Science, Ludwig-Maximilians-University Munich, Oettingenstraße 67, 80538 Munich, Germany* [b] *National ICT Australia, Canberra, 0200 ACT, Australia*

ABSTRACT

Motivation: Computational approaches to protein function prediction infer protein function by finding proteins with similar sequence, structure, surface clefts, chemical properties, amino acid motifs, interaction partners or phylogenetic profiles. We present a new approach that combines sequential, structural and chemical information into one graph model of proteins. We predict functional class membership of enzymes and non-enzymes using graph kernels and Support Vector Machine classification on these protein graphs.

Results: Our graph model, derivable from protein sequence and structure only, is competitive with vector models that require additional protein information such as the size of surface pockets. If we include this extra information into our graph model, our classifier yields significantly higher accuracy levels than the vector models. Hyperkernels allow us to select and to optimally combine the most relevant node attributes in our protein graphs. We have laid the foundation for a protein function prediction system that integrates protein information from various sources efficiently and effectively.

Availability: More information available via www.dbs.ifi.lmu.de/Mitarbeiter/borgwardt.html.

Contact: borgwardt@dbs.ifi.lmu.de

1 INTRODUCTION

Understanding the molecular mechanisms of life requires to decode the functions of proteins in an organism. Tens of thousands of proteins have been sequenced over recent years, and the structures of thousands of proteins have been resolved so far (Berman et al., 2000). Still, the experimental determination of the function of a protein with known sequence and structure remains a difficult, time- and cost-intensive task. Computational approaches to correct protein function prediction would allow to determine the function of whole proteomes faster and more cheaply.

Simulating the molecular and atomic mechanisms that define the function of a protein is beyond the current knowledge of biochemistry and the capacity of available computational power. Similarity search among proteins with 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 (e.g. Evolutionary Trace (Yao et al., 2003)), or if both appear in the same range of species (e.g. 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; for example, 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 amino acid 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 accuracy.

The remedy is to integrate different protein data sources, i.e. to combine several similarity measures, based on several different data types. If two proteins are similar on more than one scale, then the prediction of their function will be more reliable. In this article, we present how to reach this data integration via two routes: We design a graph model for proteins that can represent several types of information and we define and employ similarity measures for combining

several sources of protein data, namely graph kernels and hyperkernels.

1.1 Kernel Methods and Support Vector Machines

Kernel methods are a popular method for machine learning (Schölkopf and Smola, 2002). This paper uses kernel methods, specifically Support Vector Machines (SVMs), to perform protein function prediction. We denote by $\mathcal{X}$ the space of input data (the proteins) and by $\mathcal{Y}$ the space of labels (their function). Denote by $X := \{x_1, \ldots, x_m\}$ the training data and by $Y := \{y_1, \ldots, y_m\}$ a set of corresponding labels, jointly drawn independently and identically from some probability distribution $\mathrm{P}(x, y)$ on $\mathcal{X} \times \mathcal{Y}$. For a new example $x \in \mathcal{X}$, the problem is to predict the label y using our prior knowledge of the problem and the training examples. Observe that we do not know $\mathrm{P}(x, y)$, and hence the algorithm has to perform predictions based on the information provided by the training data.

Kernel methods have been highly successful in solving various problems in machine learning. The algorithms work by implicitly mapping the inputs into a feature space and finding a suitable hypothesis in this new space. The feature map $\phi(\cdot)$ in question is defined by a kernel function k, which allows us to compute dot products in feature space using only objects in the input space, that is $k(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle$. The kernel function must be positive definite for the SVM. Examples of positive definite kernels are the Dirac, Gaussian and Brownian bridge kernel (Schölkopf and Smola, 2002).

SVMs are based on finding a good linear hypothesis in this feature space (Cortes and Vapnik, 1995). More specifically, this solution is the hyperplane which maximizes the margin in feature space, thereby aiming at separating different classes of input data points in feature space. The margin is the maximal distance between a training example in feature space and the separating hyperplane. The C-SVM we use in this paper maximizes the "soft margin", where instead of disallowing training points from being misclassified, we penalize misclassification using a linear cost. Figure 1 shows a toy example where the soft margin SVM was used for classification. SVMs are an example of a convex optimization problem (Boyd and Vandenberghe, 2004). Efficient algorithms exist for solving convex problems, which means that large scale problems can be solved.

1.2 Support Vector Machines in Biology

Applications of SVM classification in molecular biology are numerous and the importance of kernel methods for bioinformatics is steadily growing (Schölkopf et al., 2004). Classifying proteins into their functional class has emerged as one major field of research in this context. Cai et al. (2004, 2003) use SVMs to classify protein sequences into enzyme classes. Dobson and Doig (2003) employ SVMs to distinguish enzymes from non-enzymes among protein structures. Both approaches represent proteins as vectors describing physical,

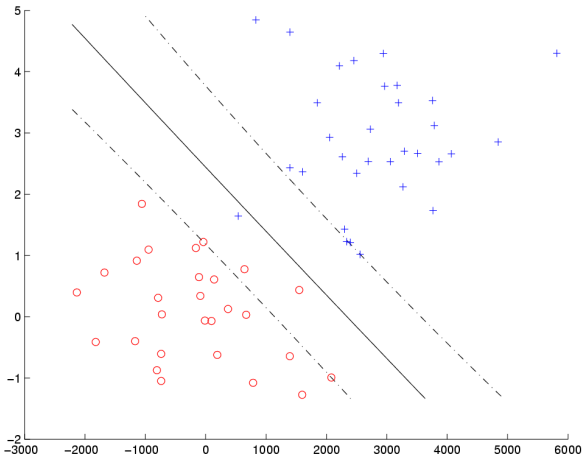

Fig. 1. The C-SVM maximizes the margin between the training examples and the hyperplane. The solid line denotes the separating hyperplane and the dashed line denotes the margin. Plus (+) and circle (o) data points represent two distinct classes of input data.

chemical and structural features of protein sequence and protein structure, respectively; Dobson and Doig (2003) additionally include information about known interaction molecules, surface properties and disulphide bonds into their 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 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 nodes and edges of the graph contain information about the secondary structure of the protein. We modify a graph kernel to define a special random walk graph kernel for proteins. In addition, we review the method of hyperkernels which we

will later apply to select relevant node attributes in our protein graph model.

2.1 Protein Graph Model

A *graph* G consists of a set of *nodes* (or *vertices*) V and *edges* E. An *attributed* graph is a graph with labels on nodes and/or edges; we refer to labels as *attributes*. In our case, attributes will consist of pairs of the form (*attribute-name*, *value*).

The adjacency matrix A of G is defined as

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

where v_i and v_j are nodes in G. A *walk* of length $k-1$ in a graph is a sequence of nodes $v_1, v_2, \cdots, v_k$ where $(v_{i-1}, v_i) \in E$ for $1 < i \leq k$.

Graph Structure of Proteins We design our graph models to contain information about structure, sequence and chemical properties of a protein. For this purpose, we model proteins as attributed and undirected graphs. Each graph represents exactly one protein. Nodes in our graph represent secondary structure elements (SSEs) within the protein structure, i.e. helices, sheets and turns. Edges connect nodes if those are neighbors along the amino acid sequence or if they are neighbors in space within the protein structure. Every node is connected to its three nearest spatial neighbors.

Nodes bear a type label, stating whether they represent a helix, sheet or turn, and physical and chemical information, namely the hydrophobicity, the van der Waals volume, the polarity and polarizability of the SSE represented by this node. One total normalized van der Waals value is determined for each node individually. Additionally, each node is labeled with the total number of its residues with low, medium or high normalized van der Waals volume separately; we will refer to this as the *3 bin distribution*. Analogously, one total value and the 3 bin distribution is added to every node for hydrophobicity, polarity and polarizability. The length of each SSE in amino acids (AAs) and the distance between the C_α atom of its first and last residue in Ångstroms (Å) constitute further node attributes, called *AA length* and *3d length*, respectively.

Every edge is labeled with its type, i.e. structural or sequential. Sequential edges are labeled with their length in AAs and structural edges with their length in Å. The length of a structural edge between two SSEs is calculated to be the distance 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.

Graph Generation We generate our protein graphs from protein files of the Protein Data Bank (PDB) (Berman et al., 2000) (see Figure 2), except for the chemical and physical node attributes. We assign these to SSEs using amino acid indices from the Amino Acid Index Database (Kawashima et al., 1999), i.e. tables with one value for each amino acid characterizing a chemical or physical feature of this AA. Normalized Amino Acid 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.

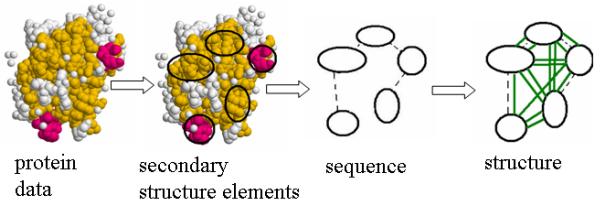

Fig. 2. Schematic illustration of graph generation from PDB protein file (Berman et al., 2000) (circles = secondary structure elements, thin dashed lines = sequential edges, thick solid lines = structural edges).

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* labeled random walks. The match between two nodes or two edges is determined by comparing their attribute values. The measure of similarity between two random walks is then the product of the kernel values corresponding to the nodes and edges encountered along the walk. The kernel value of two graphs is then the sum over the kernel values of all pairs of walks within these two graphs:

$$k_{graph}(G_1, G_2) = \sum_{walk_1 \in G_1} \sum_{walk_2 \in G_2} k_{walk}(walk_1, walk_2).$$

An elegant approach by Gärtner et al. (2003) for calculating all random walks within two graphs uses direct product graphs:

Definition 1 (Direct Product Graph). *The direct product graph of two graphs $G_1 = (V, E)$ and $G_2 = (W, F)$ shall be denoted by $G_1 \times G_2$. The node and edge set of the direct*

product graph are respectively defined as:

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

Based on this direct product graph, the random walk kernel is defined as

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

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

Nodes and edges in graph $G_1 \times G_2$ have the same labels as the corresponding nodes and edges in G_1 and G_2. Random walks of length n are weighted by λ^n in the sum over all walks. Hence λ must be chosen carefully for the sum to converge. In this paper, to simplify the approach, we calculate the random walk kernel for walks up to a predetermined length only.

2.3 Protein Graph Kernel

The graph kernel defined in the previous section is designed for discrete attributes: Attributes of two nodes v_1 and w_1 are considered similar if they are completely identical, i.e. they are compared via a Dirac kernel. The nodes in our protein graphs contain continuous attributes which are almost never completely identical between two nodes. For that reason, we replaced the Dirac kernel by more complex kernels which reflect biological knowledge about protein structure. In the following, we will define a modified random walk kernel that measures similarity between protein graphs.

Definition 3 (Modified Random Walk Kernel). *Let $G_1 = (V, E)$ and $G_2 = (W, F)$ be graphs as above. Consider two walks, $walk_1 = (v_1, v_2, ..., v_{n-1}, v_n)$ in G_1 and $walk_2 = (w_1, w_2, ..., w_{n-1}, w_n)$ in G_2 where $v_i \in V$, $w_i \in W$ for $i \in \{1, \ldots, n\}$ and $(v_i, v_{i+1}) \in E$, $(w_i, w_{i+1}) \in F$ for $i \in \{1, \ldots, n-1\}$. 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 $i \in \{1, .., n-1\}$, the step kernel is defined as*

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

where k_{edge} is defined as

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

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

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

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.

Type Kernel Identical motifs of SSEs both within protein structure and along the amino acid chain are strong hints at structural and functional relationship; most databases group proteins into structural and functional families by secondary structure content analysis (SCOP (Andreeva et al., 2004), CATH (Orengo et al., 2003)). Hence we introduce a type kernel that makes sure that a step in a random walk in two input graphs can only be performed if both edges are of the same type, i.e. both sequential or both structural, and both origin nodes and both target nodes are of the same type, i.e. helix, sheet or turn.

Definition 5 (Type Kernel). *k_{type} is defined identically for both nodes and edges x and x':*

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

Length Kernel Length kernels ensure that we do not count SSEs or edges as being similar that differ a lot in size. Insertion and deletion of amino acid residues might change the length of SSEs or their distance towards each other, while the overall fold and function of the protein remains unchanged. For this reason, we employed the Brownian bridge kernel, that assigns the highest kernel value to SSEs and edges that are identical in length and assigns zero to all SSEs and edges that differ in length more than by a constant c. This maximum difference constant c was set to 2 AA for sequential edges, to 2 Å for structural edges and to 3 Å for SSE nodes.

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

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

Node Labels Kernel We compare the physico-chemical features of two secondary structure elements via a node labels kernel. We chose this kernel to be Gaussian, since these have shown the best performance in related studies (Cai et al., 2004); σ was set to 13 by cross-validation.

Definition 7 (Node Labels Kernel). *The node labels kernel $k_{node\ labels}$ is a Gaussian kernel over two vectors representing the values of all labels of node x and node x':*

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

It is essential to show that this modified graph kernel is still a valid positive definite kernel.

Lemma 8. *The modified random walk graph kernel is positive definite.*

Proof. The type kernel is a Dirac kernel, the length kernel a Brownian bridge kernel and the node labels kernel a Gaussian kernel; these kernels are known to be positive definite (Schölkopf and Smola, 2002). Since pointwise multiplication preserves positive definiteness, node kernel, edge kernel and step kernel are consequently positive definite.

We can now define a positive definite kernel on walks, $\bar{k}^{(j)}_{walk}$, which is identical to our walk kernel k_{walk} for pairs of walks of length j and zero otherwise (Kashima et al., 2003). $\bar{k}^{(j)}_{walk}$ is a tensor product of step kernels (Schölkopf and Smola, 2002) for walks of length j which is zero-extended to the whole set of pairs of walks; hence it is positive definite (Haussler, 1999). k_{walk} as the sum over all $\bar{k}^{(j)}_{walk}$ is therefore a valid kernel.

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, $R_{\mathrm{reg}}(f, X, Y) = \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, f(x_i)) + \frac{\lambda}{2}\|f\|^2$, we regularize the empirical quality functional $Q_{\mathrm{emp}}(k, X, Y)$.

Definition 9 (Regularized Quality Functional).

$$Q_{\mathrm{reg}}(k, X, Y) := Q_{\mathrm{emp}}(k, X, Y) + \frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}} \qquad (1)$$

where $\lambda_Q > 0$ is a regularization constant and $\|k\|^2_{\underline{\mathcal{H}}}$ denotes the RKHS norm in $\underline{\mathcal{H}}$.

Minimization of Q_{reg} is less prone to overfitting than minimizing Q_{emp}, since the regularizer $\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}$ effectively controls the complexity of the class of kernels under consideration. The minimizer of Equation (1) satisfies the representer theorem:

Theorem 10 (Representer Theorem). *Denote by $\mathcal{X}$ a set, and by Q an arbitrary quality functional. Then each minimizer $k \in \underline{\mathcal{H}}$ of the regularized quality functional 1, admits a representation of the form*

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

This shows that even though we are optimizing over a whole Hilbert space of kernels, we still are able to find the optimal solution by choosing among a finite number, which is the span of the kernel on the data.

We use Semidefinite Programming (SDP) formulations of the optimization problems arising from the minimization of the regularized quality functional (Ong and Smola, 2003). Semidefinite programming is the optimization of a linear objective function subject to constraints which are linear matrix inequalities and affine equalities.

In this section, we define the following notation. For $p, q, r \in \mathbb{R}^n, n \in \mathbb{N}$ let $r = p \circ q$ be defined as element by element multiplication, $r_i = p_i \times q_i$. The pseudo-inverse (or Moore-Penrose inverse) of a matrix K is denoted by $K^\dagger$. Define the hyperkernel Gram matrix $\underline{K}$ by $\underline{K}_{ijpq} = \underline{k}((x_i, x_j), (x_p, x_q))$, the kernel matrix $K = \text{reshape}(\underline{K}\beta)$ (reshaping an m^2 by 1 vector, $\underline{K}\beta$, to an $m \times m$ matrix), $Y = diag(y)$ (a matrix with y on the diagonal and zero otherwise), $G(\beta) = YKY$ (the dependence on β is made explicit), and $\mathbf{1}$ a vector of ones.

The number of training examples is assumed to be m. Where appropriate, γ is a Lagrange multiplier, while η and ξ are vectors of Lagrange multipliers from the derivation of the Wolfe dual for the SDP, β are the hyperkernel coefficients, t_1 and t_2 are the auxiliary variables.

Example 1 (Linear SVM (C-style)). *A commonly used support vector classifier, the C-SVM uses an ℓ_1 soft margin, where $l(x_i, y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$, which allows errors on the training set. The parameter C is given by the user. Setting the quality functional $Q_{\text{emp}}(k, X, Y) = \min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m l(x_i, y_i, f(x_i)) + \frac{1}{2C}\|w\|^2_{\mathcal{H}}$, the resulting SDP is*

$$\begin{array}{ll} \min\limits_{\beta,\gamma,\eta,\xi} & \frac{1}{2}t_1 + \frac{C}{m}\xi^\top \mathbf{1} + \frac{C\lambda_Q}{2}t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2 \\ & \begin{bmatrix} G(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0, \end{array} \tag{3}$$

where $z = \gamma y + \mathbf{1} + \eta - \xi$.

The value of α which optimizes the corresponding Lagrange function is $G(\beta)^\dagger z$, and the classification function, $f = sign(K(\alpha \circ y) - b_{offset})$, is given by $f = sign(KG(\beta)^\dagger(y \circ z) - \gamma)$.

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 10, 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.

Kernel type	**Accuracy**	**St. dev.**
Vector kernel	76.86	1.23
Optimized vector kernel	80.17	1.24
Graph kernel	77.30	1.20
Graph kernel without structure	72.33	5.32
Graph kernel with global info	84.04	3.33
DALI classifier	75.07	4.58

Table 1. Accuracy of prediction of functional class of enzymes and non-enzymes in 10-fold cross-validation with C-SVM. The first two results are the results obtained by Dobson and Doig (2003). "Graph kernel" is our protein kernel defined as in Section 2.3, "Graph kernel without structure" is the same kernel but on protein models without structural edges, "Graph kernel with global info" is our protein graph kernel plus additional global node labels. "DALI classifier" is a Nearest Neighbor Classifier on DALI Z-scores.

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 vs. 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 amino acid 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 optimized subset of their attributes, they improved their accuracy to 80.17% in 10-fold cross-validation.

We created protein graphs for all proteins from their dataset for which the secondary structure is given in the corresponding PDB file. This meant that we could use 1128 out of 1178 proteins for our tests (the fraction of enzymes remained at 59%). On these protein graphs, we calculated our protein random walk graph kernel. We performed 10-fold cross-validation using C-Support Vector Machine (C-SVM (Cortes and Vapnik, 1995)) classification and report the results in Table 1. As a comparison, we implemented and ran a Nearest Neighbor Classifier based on DALI Z-scores (Holm and Sander, 1996) on the same dataset.

Our results show that our graph kernel is competitive with the existing vector kernel approach, although it relies on less

information than the vector approach. Our graph model can be generated from sequence and structure, while the vector model requires additional information about ligands, surface clefts and bonds of the proteins in question. Furthermore, our graph kernel gives also better results than the DALI classifier, which is based on state-of-the-art structure comparison results.

These results suggest two further experiments: First, to check whether we can reach similarly good results if we do not include structural edges into our protein model. This kind of graph model could be generated without knowing the structure of a protein, relying solely on the sequence and on a secondary structure prediction system. We tested our kernel on graphs without structural edges and found a significant deterioration to 72.33% prediction accuracy (see Table 1).

Second, we tested whether our protein classifier could be improved by incorporating Dobson and Doig's extra information. We extended our protein graphs to include additional information as node labels. These *global* node attributes are the same for all nodes in one graph; they represent the existence of ligands, the number of disulphide bonds, the size of the largest surface pocket (Binkowski et al., 2003) and the fraction of each amino acid type on the protein surface (Tsodikov et al. , 2002), analogous to (Dobson and Doig, 2003). On this protein graph model with global information, we improved our classification accuracy to 84.04% (see Table 1). This is highly significantly better (with Yates' corrected $\chi^2 = 18.56$ and $p = 0.00002$) than the standard vector kernel which has an accuracy of 76.86%. This is also significantly better (with Yates' corrected $\chi^2 = 5.71$ and $p = 0.0169$) than the vector kernel on an optimized subset of attributes which has an accuracy of 80.17%.

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 ten 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 ten 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 ten normalized kernel matrices. As a comparison, we also ran our default protein graph kernel with all node attributes on the same dataset.

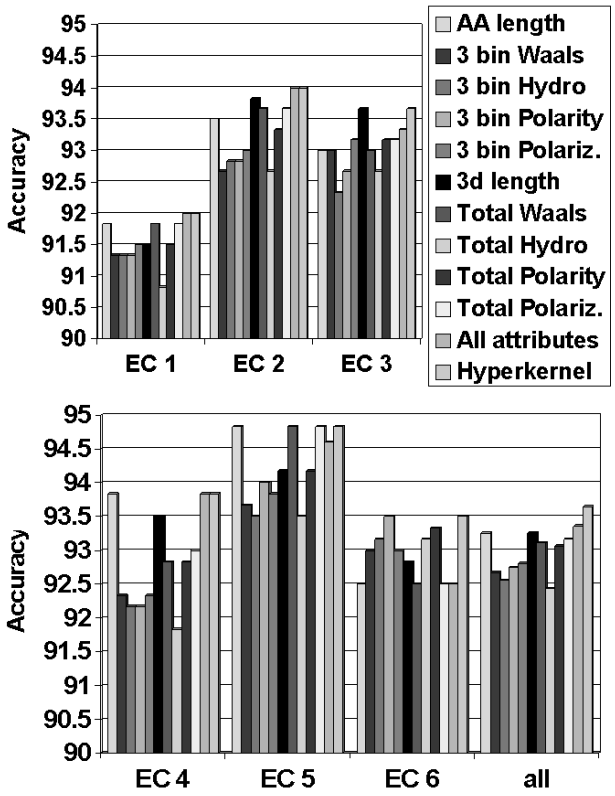

Fig. 3. Prediction accuracy using kernel matrices on individual attributes, one kernel on all attributes and the hyperkernel (see Example 1) in 6-fold cross-validation on 600 enzymes from 6 EC top level classes (AA = amino acid, Waals = van der Waals volume, Hydro. = Hydrophobicity, Polariz. = Polarizability).

For each EC class, we conducted 1-vs.-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. The hyperkernel performs on average best of all kernels. Across all EC classes, the hyperkernel reaches at least the accuracy of the best individual kernel, i.e. the hyperkernel

technique succeeds in finding an optimal linear combination of input kernels. The hyperkernel performs even slightly better than our original kernel with all attributes.

Attribute	EC1	EC2	EC3	EC4	EC5	EC 6
AA length	**1.00**	**0.31**	**1.00**	**1.00**	**0.73**	0.00
3 bin Waals	0.00	0.00	0.00	0.00	0.00	0.00
3 bin Hydro.	0.00	0.00	0.00	0.00	0.00	0.00
3 bin Polarity	0.00	0.01	0.00	0.00	0.00	**1.00**
3 bin Polariz.	0.00	0.00	0.00	0.00	**0.12**	0.00
3d length	0.00	**0.40**	0.00	0.00	0.00	0.00
Total Waals	0.00	0.00	0.00	0.00	0.00	0.00
Total Hydro.	0.00	**0.13**	0.00	0.00	0.01	0.00
Total Polarity	0.00	**0.14**	0.00	0.00	0.01	0.00
Total Polariz.	0.00	0.01	0.00	0.00	**0.13**	0.00

Table 2. Hyperkernel weights for individual node attributes (AA = amino acid, Waals = van der Waals volume, Hydro. = Hydrophobicity, Polariz. = Polarizability).

The hyperkernel assigns on average the highest weight to the node attribute amino acid length. Results differ significantly between EC classes. While in EC classes 1, 3 and 4 attribute AA length receives the maximum weight of 1.00, 3 bin polarity receives maximum weight for EC class 6. This is consistent with the observation that the 3 bin polarity kernel reaches the maximum prediction accuracy for EC class 6 among all attribute kernels. In EC class 2 and EC class 5, the hyperkernel detects a combination of several attribute kernels yielding the maximum accuracy.

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 Support Vector Machine 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 secondary structure elements and edges that represent sequential or spatial neighborship between these elements. Graph models based on smaller subunits of proteins, amino acid 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 ten times or one hundred 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 secondary structure elements.

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 physicochemical properties of their secondary structure elements. 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 weight to the node attribute amino acid length. Functional similarity between proteins seems to be closely linked to the question whether the secondary structure elements of these proteins are equally long. This finding is consistent with the observation that the structure of a protein - which includes the length of its secondary structure elements - is the biggest hint at its function (Bartlett et al., 2003). The fact that the node attribute ”polarity” is most important for classifying enzymes from EC class 6 illustrates that approximate chemical properties of proteins can also help to identify protein function. In addition, hyperkernels will allow us to combine our protein graph information with other proteomic information to improve our classifier.

Future work will aim at refining our protein graph model by adding more node and edge labels and at integrating more protein information into our classifier to make function predictions more accurate. Attributed graphs, our protein graph kernels and hyperkernels will be essential for this process of data fusion.

ACKNOWLEDGEMENTS

This work was supported in part by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 031U112F within the BFAM (Bioinformatics for the Functional Analysis of Mammalian Genomes) project which is part of the German Genome Analysis Network (NGFN). National ICT Australia is funded through the Australian Government’s *Backing Australia’s Ability* initiative, in part through the Australian Research Council.

REFERENCES

S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *NAR*, 25: 3389–3402, 1997.

A. Andreeva, D. Howorth, S. E. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin. Scop database in 2004: refinements integrate structure and sequence family data. *NAR*, 32 Database issue:D226–D229, Jan 2004.

G. J. Bartlett, A. E. Todd, and J. M. Thornton. Inferring protein function from structure. In P. E. Bourne and H. Welssig, editors, *Structural Bioinformatics*, pages 387–407. Wiley-Liss, Inc., New York, 2003.

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *NAR*, 28:235–242, 2000.

T. A. Binkowski, S. Naghibzadeh, and J. Liang. Castp: Computed atlas of surface topography of proteins. *NAR*, 31(13):3352–3355, Jul 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

C. Z. Cai, L. Y. Han, Z. L. Ji, and Y. Z. Chen. Enzyme family classification by support vector machines. *Proteins*, 55(1):66–76, Apr 2004.

C. Z. Cai, W. L. Wang, L. Z. Sun, and Y. Z. Chen. Protein function classification via support vector machine approach. *Math Biosci*, 185(2):111–122, Oct 2003.

M. Charton and B. I. Charton. The structural dependence of amino acid hydrophobicity parameters. *J Theor Biol*, 99(4):629–644, Dec 1982.

H. Cid, M. Bunster, M. Canales, and F. Gazitua. Hydrophobicity and structural classes in proteins. *Protein Eng*, 5(5):373–375, Jul 1992.

C. Cortes, P. Haffner and M. Mohri. Positive Definite Rational Kernels. In B. Schölkopf and M. K. Warmuth, editors, *Proceedings of the* 16^{th} *Annual Conference on Learning Theory*, 41–56, 2003.

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

P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J Mol Biol*, 330(4):771–783, 2003.

J. L. Fauchere, M. Charton, L. B. Kier, A. Verloop, and V. Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *Int J Pept Protein Res*, 32(4):269–278, 1988.

S. Fine and K. Scheinberg. Efficient SVM Training Using Low-Rank Kernel Representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

T. Gärtner, P. Flach and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT/Kernel 2003*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 129–143. Springer-Verlag, 2003.

R. Grantham. Amino acid difference formula to help explain protein evolution. *Science*, 185(4154):862–864, 1974.

A. Harrison, F. Pearl, R. Mott, J. Thornton, and C. Orengo. Quantifying the similarities within fold space. *J Mol Biol*, 323 (5):909–926, 2002.

D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99 - 10, Computer Science Department, UC Santa Cruz, 1999.

H. Hegyi and M. Gerstein. The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *J Mol Biol*, 288(1):147–164, 1999.

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

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. *Proceedings of ICML*, 2003.

S. Kawashima, H. Ogata, and M. Kanehisa. Aaindex: Amino acid index database. *NAR*, 27(1):368–369, 1999.

R. S. Kondor and J. Lafferty. Diffusion Kernels on Graphs and Other Discrete Structures. *Proceedings of ICML*, 2002.

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

G. R.G.Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

Cheng Soon Ong, A. J. Smola. Machine Learning with Hyperkernels. *Proceedings of ICML*, 568–575, 2003.

Cheng Soon Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In *Advances in Neural Information Processing Systems 15*, pages 495–502, 2003.

C. A. Orengo, F. M. Pearl, and J. M. Thornton. The cath domain structure database. *Methods Biochem Anal*, 44:249–271, 2003.

M. Pellegrini, E.M. Marcotte, M.J. Thornton, D. Eisenberg, and T.O.E. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of the National Academy of Sciences USA*, 96:4258–4288, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, K. Tsuda, and J.P. Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, Massachusetts, 2004.

I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg. BRENDA, the enzyme database: updates and major new developments. *NAR*, 32 Database issue:D431–D433, 2004

O. V. Tsodikov, M. T. Jr. Record, and Y. V. Sergeev, A novel computer program for fast exact calculation of accessible and molecular surface areas and average surface curvature. *J. Comput. Chem.*, 23,600–609, 2002.

J. C. Whisstock and A. M. Lesk. Prediction of protein function from protein sequence and structure. *Q Rev Biophys*, 36(3):307–340, 2003.

H. M. Wilks, K. W. Hart, R. Feeney, C. R. Dunn, H. Muirhead, W. N. Chia, D. A. Barstow, T. Atkinson, A. R. Clarke, and J. J. Holbrook. A specific, highly active malate dehydrogenase by redesign of a lactate dehydrogenase framework. *Science*, 242(4885):1541–1544, 1988.

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

H. Yao, D. M. Kristensen, I. Mihalek, M. E. Sowa, C. Shaw, M. Kimmel, L. Kavraki, and O. Lichtarge. An accurate,

sensitive, and scalable method to identify functional sites in protein structures. *J Mol Biol*, 326(1):255–261, 2003.