Journal of Bioinformatics and Computational Biology
Vol. 5, No. 3 (2007) 717–738
© Imperial College Press

Imperial College Press
www.icpress.co.uk

INCORPORATING HOMOLOGUES INTO SEQUENCE EMBEDDINGS FOR PROTEIN ANALYSIS

ELEAZAR ESKIN*
Department of Computer Science, Department of Human Genetics,
University of California, Los Angeles,
Los Angeles, CA, 90095
eeskin@cs.ucla.edu

SAGI SNIR
School of Computer Science and Mathematics,
Netanya Academic College,
Netanya, Israel, 42100
ssagi@netanya.ac.il

Received 16 December 2005
Revised 23 October 2006
Accepted 23 February 2007

Statistical and learning techniques are becoming increasingly popular for different tasks in bioinformatics. Many of the most powerful statistical and learning techniques are applicable to points in a Euclidean space but not directly applicable to discrete sequences such as protein sequences. One way to apply these techniques to protein sequences is to embed the sequences into a Euclidean space and then apply these techniques to the embedded points. In this work we introduce a biologically motivated sequence embedding, the *homology kernel*, which takes into account intuitions from local alignment, sequence homology, and predicted secondary structure. This embedding allows us to directly apply learning techniques to protein sequences. We apply the homology kernel in several ways. We demonstrate how the homology kernel can be used for protein family classification and outperforms state-of-the-art methods for remote homology detection. We show that the homology kernel can be used for secondary structure prediction and is competitive with popular secondary structure prediction methods. Finally, we show how the homology kernel can be used to incorporate information from homologous sequences in local sequence alignment.

Keywords: Protein classification; sequence alignment; kernel methods.

1. Introduction

The analysis of protein sequences is one of the most successful areas in bioinformatics. Three major ideas and intuitions are used over and over again in many

[*] Corresponding author.

of the successful techniques. These include the use of local alignments, the use of homologous sequences and the use of secondary structure information.

Local alignments provide insight into the evolutionary relationship between two sequences. This evolutionary information can be used to infer a functional or structural relationship from the strength of the local alignment. The Smith–Waterman[24] and BLAST[2] algorithms for local sequence alignment are some of the most useful methods for protein analysis. Information from homologous protein sequences has also been applied to many different problems. Some of the most popular applications of this technique are the PSI-BLAST[3] and PHD[18] algorithms. PSI-BLAST uses homologues to generate a profile for a protein query sequence to refine database homology search. PHD uses homologues to generate profiles for the prediction of secondary structure. In many cases function and structure are shared among homologous sequences and intuitively, common amino acids among the homologues are likely to be either functionally or structurally important. Using secondary structure has also become popular due to the existence of relatively accurate secondary structure prediction techniques. Secondary structure is often used to improve sequence alignments and to help infer functional information.

A general challenge in analyzing protein sequences is that they are by nature discrete sequences. Although there exist many powerful techniques for the analysis of discrete sequences such as hidden Markov models (HMMs) which have been successfully applied to biological sequences,[6,15] many of the most powerful techniques in statistics and learning are defined over points in a Euclidean space, and it is nontrivial to adapt them to discrete sequences. Recently, several methods have been proposed to analyze the space of proteins as a finite metric space.[20] Other works such as the spectrum or mismatch kernels[11,12] implicitly represent sequences in a Euclidean space or through implicit embeddings using "bio-basis" functions.[27,28]

In this paper, we present the *homology kernel*, a sequence embedding that incorporates the biological intuitions of local alignment, information from homologous sequences, and information from secondary structure. The idea behind our approach is to embed a sequence as a point in a Euclidean space referred to as the *feature space*. The position of the image of the sequence in the embedded space depends on the sequence as well as its homologues and secondary structure. This means that distances between points in the feature space capture intuitions of the local alignment. A pair of sequences with homologues which have close local alignments to each other will be closer in the feature space than a pair of sequences with homologues distant from each other. In general, we introduce the notion of *neighborhood* of sequences. Since the dimension of the Euclidean space is very large and the sequence embeddings are sparse, processing the points in the feature space directly is very computationally expensive. We instead process the embeddings using a kernel which allows us to efficiently compute inner products of images in the feature space directly from the sequences.

The homology kernel builds upon a general framework for constructing string kernels based on substrings. In this paper, we introduce this framework in the construction of the homology kernel. In particular, the homology kernel uses *normalized* kernel mapping that takes into consideration overly represented substrings and corrects this artifact in the mapping. This is a mandatory requirement posed by the homology kernel as it amplifies the number of sub sequences used for determining relationship between sequences. It also generalizes the notion of mismatches by using the *sparsity* idea. Eventually, it allows for amino acids *grouping* that accounts for similar amino acid structures or functions. Many previously presented string kernels such as the spectrum kernel, mismatch kernels, or wildcard kernels can be approximated by specific instances of our general framework in which they are often more efficiently computed. This generalization is much more flexible than the previous kernels and we show that it significantly outperforms both the mismatch and spectrum kernels as well as state of the art methods applied to protein family classification. By applying this framework, we are able to embed sequences, their predicted secondary structure, and their homologues, in a Euclidean space. Finally, computation of the sparse and normalized kernels is efficient through the use of specialized data structures.

The homology kernel has several major advantages. Distances induced by the homology kernel capture the intuitions of local alignment much better than previous kernels such as the spectrum or mismatch kernel. Since the method defines an explicit mapping to a Euclidean space, we can apply statistical and learning techniques directly in the feature space. As compared to the traditional profile based measures of similarity, the homology kernel does not lose information from the sequences which is lost when collapsing the sequences into a profile.

We demonstrate the use of the homology kernel for learning from sequences, their homologues and their predicted secondary structure. We apply the homology kernel to two learning problems: protein family classification and prediction of secondary structure. The homology kernel applied to protein family classification, HK-PFC, outperforms several previous methods on a widely used benchmark dataset. The homology kernel applied to secondary structure prediction, HK-SSP, gives promising results compared with popular secondary structure classification methods.

Our third application, HK-ALIGN, demonstrates how the homology kernel can be used to locally align sequences. Instead of using a substitution matrix for local alignment, HK-ALIGN uses the distance induced by the homology kernel over two aligned columns as the similarity score for local sequence alignment. In this way, HK-ALIGN aligns two protein sequences using dynamic programming taking into account their homologues. This approach has some similarities to the approach of Yona and Levitt,[31] who compare pairs of profiles and approaches that incorporate secondary structure into alignments. Again, HK-ALIGN differs from profile based techniques in that it compares the sequences as opposed to the profiles. We evaluate HK-ALIGN by showing how it can align a pair of remote homologues from the MEROPS[17] database which according to BLASTP, have no significant sequence similarity.

Since the publication of the spectrum and mismatch kernels[11,12] numerous other methods have been presented that have extended and demonstrated improved performance. In particular the approaches of Weston *et al.*[30] and Kuang *et al.*[10] both extend the string kernels to take into account unlabelled homologous sequences and are motivated by similar intuitions to the homology kernel. Several others including Saigo *et al.*[19] and Rangwala and Karypis[16] have also demonstrated improved performance over newer benchmarks for protein family classification. Due to the multitude of methods and different benchmarks used, a direct comparison of all of these methods are beyond the scope of the paper and we are only able to make claims of performance improvements over the Spectrum and Mismatch kernels.

The rest of this paper is organized as follows: In Sec. 2 we introduce the homology kernel, its properties and the motivations for it. In Sec. 3, we show our experimental results for using the homology kernel in protein family classification and secondary structure prediction and Sec. 4 for using it in sequence alignment. In Sec. 5 we detail about the implementation aspects of the homology kernel and conclude in Sec. 6.

This paper extends a preliminary version of this work.[7]

2. The Homology Kernel

In this section, we introduce the homology kernel and describe its application for learning and alignment of protein sequences. The homology kernel is based on the notion of substring embeddings which we describe later.

2.1. *Embedding sequences in a euclidean space*

The main idea behind the embedding defined by the homology kernel is that sequences are represented as collections of fixed length substrings (of length k) obtained by a sliding window. The homology kernel has its roots in the spectrum kernel[12] which first applied this technique to biological sequences. These sets of fixed length substrings are mapped into a feature space which consists of a dimension for each possible fixed length substring. In the case of protein sequences, the spectrum kernel maps the sequences into a 20^k dimensional space.

We generalize the implicit embedding of the spectrum kernel to take advantage of intuitions of local alignment. The intuitions we wish to incorporate are a tolerance for inexact matching or mismatches and a notion of similarity between certain amino acids such as that defined by substitution matrices. We also show how distances between the images of two embedded sequences are related to their local alignment.

2.1.1. *Local alignments and substrings*

To motivate our embedding, we consider as an example a small portion of a local alignment shown in Fig. 1(a). Intuitively the "quality" of the local alignment is the amount of similarity between the portions of the alignment. One way to quantify

```
LPQSFLLKCLEQVRKVQADGTAHKLCHPEELVLLG
LRLLQALAGISPVRKVQADGTINIWQQMEDLGMAP
            VRKV
             RKVQ
              KVQA
               VQAD
                QADG
                 ADGT
```

(a)

```
LPQSFLLKCLEQVRKVRADGTAHKLCHPEELVLLG
LRLLQALAGISPVRKVQADGTINIWQQMEDLGMAP
            VRKV




                 ADGT
```

(b)

Fig. 1. (a) Two sequences which share a region of length 9. We show the 6 substrings of length 4 which are shared by the sequences from the region. (b) The same sequences with the fifth amino acid of the region changed. Notice that now only 2 substrings of length 4 are shared.

the amount of similarity is to consider the number of overlapping short substrings that the two locally aligned regions share. Due to the overlapping nature of the short substrings, if two sequences share a region of length l, they will share $l-k+1$ substrings of length k from this region. If they share a region of length k, then they will only share a single substring from this region. The number of short substrings that two sequences share can be viewed as an estimate length of regions that they share, but only for regions longer than k.

If we consider an example where two sequences share a region which contains one mismatch such as in Fig. 1(b), we notice that the number of shared substrings decreases significantly. Although the two sequences are not identical, they clearly are very similar to each other with respect to local alignment. In particular, the specific substitution that occurs in the example (from Q to R) is likely due to the similar chemical properties of the amino acids. Therefore, in order for a substring based notion of similarity between sequences to be similar to local alignment, it must take into account some notion of partial matches of substrings, and to take into account the chemical properties of the amino acids as encoded in a substitution matrix.

The main idea behind our embedding is that we embed substrings in a way that allows for partial matchings. In this space a substring is embedded close to other similar substrings. In addition, substrings that differ in amino acids that are similar are closer than substrings that differ in amino acids that are dissimilar. Below we describe a method for constructing such an embedding.

2.1.2. *Constructing embeddings based on substrings*

We introduce a novel flexible framework for constructing embeddings of sequences based on substrings. In this framework, we define several properties of the embedding which we would like to achieve. The first is the *normalization* property, which relates the norm of an embedded sequence to its length and relates the cosine between two embedded sequences to the number of substrings they share. The second is the *sparsity* property, which allows for approximate matching of the substrings. The third is a *grouping* property which allows for the approximate matching to take into account chemical properties of the amino acids. The details of the construction of embeddings that satisfy these groupings are described in Sec. 5. We will show how to construct an embedding that satisfies all of these properties. The embedding we present in this paper, the homology kernel, is constructed from a normalized sparse kernel with groupings. Since these embeddings are explicitly defined, we can very easily apply learning algorithms to images of the sequences in the feature space such as support vector machine (SVM).

2.2. *The homology kernel*

In our embedding, we wish to take advantage of the information contained in homologous sequences. The idea behind the homology kernel is that we not only embed the substrings in a sequence, but also embed all the substrings of the homologous sequences.

The homology kernel measures the similarity between two *sets* of aligned sequences. To apply the homology kernel to measure the similarity between two protein sequences, we first apply BLAST over a large protein database to obtain the homologous sequences for each sequence and align each of the two sets using CLUSTALW.[26] For each set, we embed the complete collection of substrings in the homologues and measure the distance between the two points corresponding to the two sets. This gives a measure of the similarity between the two proteins taking into account their homologues. We define the local homology kernel between two positions in a pair of proteins as the dot product between the embeddings of the set of substrings that are centered around those positions in the respective proteins. The local homology kernel measures the similarity between the positions when considering the homologues. The global homology kernel is defined as the dot product between the embeddings of the complete set of substrings from all of the homologues between two proteins. The global homology kernel measures the similarity between two complete protein sequences taking into account their homologues.

We emphasize that the properties of the sparse normalized kernel with groupings were specifically motivated by the needs of the homology kernel. Since many of the substrings in a column are very close or exact copies of each other, the normalization property avoids the bias of multiple matching substrings of the previous spectrum and mismatch kernels. In addition, some columns in an alignment correspond to a specific type of amino acid such as hydrophobic amino acids. The groupings help capture this kind of information.

An issue in combining local homology kernel scores is how to deal with insertions and deletions in the aligned sequences. We use a special symbol to represent insertions which does not match any other symbol including itself.

2.2.1. *Homology kernel versus profiles*

At a high level, most of the popular methods that use homologues incorporate them in the same way. They first obtain a set of homologous sequences from a database. Then they align these sequences. Finally, they compute a profile for the sequences from the aligned columns by representing each position in the alignment by a probability distribution over amino acids estimated from the amino acid occurrences at the position in the homologous sequences.

However, when each column of the alignment is collapsed into a profile, some information about the original sequence and the homologous sequences is lost. For example, consider the sequences in Fig. 2. In this example, columns (a), (b), and (c) all have the same profile. In the first position, the profile is $A = 1.0$ and in the second and third positions the profile is $A = 0.5$ and $C = 0.5$. Clearly columns (b) and (c) are identical and thus more similar to each other while column (a) is different from the other two. A fundamental difference between the homology kernel and profile methods is that the homology kernel is defined over neighboring columns in the alignment instead of a single column which gives some context to the sequences. In the above example, the homology kernel with $k > 1$ would give different scores when comparing (a) and (b) versus (b) and (c).

2.2.2. *Incorporating predicted secondary structure*

Another type of information that we take into account in our embedding is secondary structure. Secondary structure defines the local structure of a protein and has been shown to improve the performance of sequence alignment.

(a)	(b)	(c)
AAA	AAC	AAC
AAA	AAC	AAC
ACC	ACA	ACA
ACC	ACA	ACA

Fig. 2. Example of columns in alignment. Notice that for columns (a), (b), and (c) the profile generated from the column is the same: $A = 1.0 : A = 0.5, C = 0.5 : A = 0.5, C = 0.5$.

Methods that predict the secondary structure of a protein classify each residue as belonging to one of four classes: helix, sheet, coil, or none of the above. For our experiments we use the PHD system[18] to predict secondary structure. For each protein sequence, we obtain the predicted secondary structure. When incorporating secondary structure, we extend our alphabet Σ to contain a symbol for each type of secondary structure element. When embedding a sequence, for each substring of length k, we embed both a substring of the amino acid residues and a substring of the predicted secondary structure of the residues.

3. Learning with the Homology Kernel

One of the advantages of and motivations for a sequence embedding is that we can directly apply learning algorithms to the sequences. In the feature space, we can find a separating hyperplane which divides the images of the sequences into two classes. For example, we can separate the images of the sequences into sequences that are part of a specific protein family and sequences that are not.

For our experiments, we use a support vector machine (SVM) which have previously been used for protein family classification with the mismatch kernel and spectrum kernel[11,12] (see Sec. 5 for details on SVMs).

3.1. *Protein family classification experiments*

We compare the performance of the homology kernel to other methods over the benchmark data set assembled by Jaakkola *et al.*,[9] and show that our homology kernel significantly outperforms previous methods such as the mismatch kernel and the SVM-Fisher method.

We test the homology kernel method using the SCOP[14] (version 1.37) data sets designed by Jaakkola *et al.*[9] for the remote homology detection problem. In this test, remote homology is simulated by holding out all members of a target SCOP family from a given super family. Positive training examples are chosen from the remaining families in the same super family, and negative test and training examples are chosen from outside the target family's fold. The held-out family members serve as positive test examples. There are a total of 33 families in the data. Details of the data sets are available at http://www.soe.ucsc.edu/research/compbio/discriminative.

Because Jaakkola *et al.* needed to train HMMs for the protein families represented in the positive training sets for these experiments, they used the SAM-T98 algorithm to pull in domain homologues from the non-redundant protein database. These additional domain homologues were added to the data set as positive examples in the experiments. We note that this additional training data is more advantageous to the methods that rely on generative models (such as hidden Markov models and SVM-Fisher) than it is to our method.

We use ROC_{50} scores to compare the performance of different homology detection between methods. The ROC_{50} score is the area under the receiver operating characteristic curve — the plot of true positives as a function of false positives — up to the first 50 false positives.[8] A score of 1 indicates perfect separation of positives from negatives, whereas a score of 0 indicates that none of the top 50 sequences selected by the algorithm were positives.

For comparison, we include results from the mismatch kernel which was shown to be competitive over the same dataset with state-of-the-art methods including the SAM-T98 iterative HMM, and the SVM-Fisher method. We compare our results to the $(5, 1)$-mismatch kernel which was the best performing of that class of kernel methods.[11] Each line on the comparison graphs shows the total number of families (the y-axis) for which a given method exceeds an ROC_{50} score threshold (the x-axis).

We perform several tests to measure how effective each aspect of the homology kernel is on its own compared to the other methods. We first apply simply the sparse normalized kernel with four values of $\alpha = \{0.2, 0.4, 0.6, 0.8\}$ versus the other methods. Fig. 3(a) shows the results of this comparison and we see that the sparse normalized kernel significantly outperforms the previous methods. As we discussed previously, the sparse kernel with $\alpha = 0.05$ is an approximation for the mismatch kernel and clearly based on Fig. 3(a) the performance peaks for higher values of α. We also note that $\alpha = 0.4$ is the best performer among the normalized sparse kernels and we use it for the remaining experiments.

We apply sparse kernel with groupings of symbols with $\beta = 0.7$. The groupings of the amino acids[25] and decompose the set of amino acids into the following groups: $\{K, R, H\}$, $\{D, E\}$, $\{I, V, M, L\}$, $\{C\}$, $\{P, S, A, G, T\}$, $\{F, W, Y\}$, $\{N, Q\}$. Figure 3(b) shows the results for this grouping. As we can see, it performs significantly better than without the groupings.

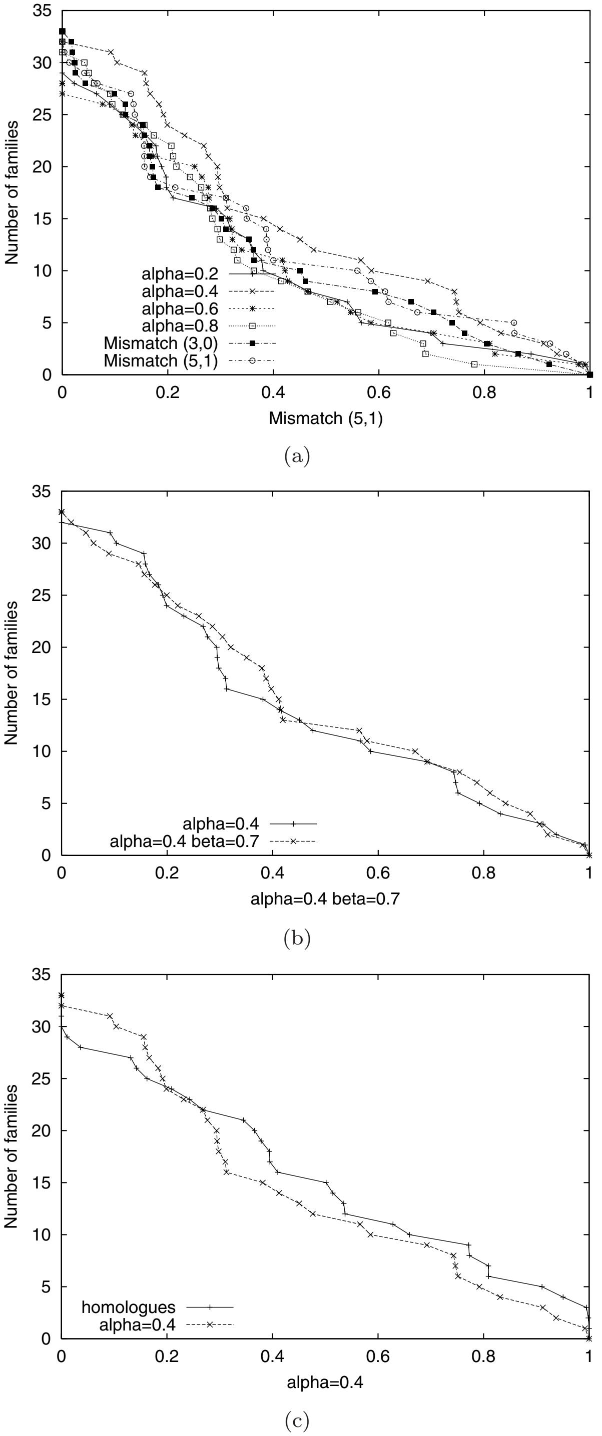
We evaluate the effect of homologues to the sparse normalized kernel. For every sequence in the Jaakkola *et al.* data set, we query the sequence using BLAST against the SWISS-PROT database[4] and retrieve up to 50 homologues. Figure 3(c) shows the results for using the homologues.

3.2. *Secondary structure prediction*

We perform our secondary structure experiments over the 513 protein secondary structure prediction dataset of Cuff and Barton.[5] The 513 proteins in this data set are nonhomologous and each residue is labeled with one of 8 secondary structures. Using the standard conversion, we reduce this set of 8 secondary structure types to three types: H helix, E sheet, C coil or N no structure. For each protein, there are anywhere from 2 to 50 aligned homologous sequences.

We construct a training example for each position at each of the 513 proteins. The label of the training example is the secondary structure of the position. The sequences for the training example are the k-mers of length 5 centered at the position in the original sequence and all of the alignments. We use the local homology kernel to compute the images of the examples in the feature space.

We set aside 20% of the sequences for testing and used them to evaluate the quality of the predictions of a classifier trained on 80% of the data.



(a)

(b)

(c)

Fig. 3. Protein remote homology results. (a) Different values for α. (b) Incorporating groupings. (c) Incorporating homologues.

We measure the performance of our classifiers by evaluating the predictions over the test data. We use the standard Q_3 evaluation measure for secondary structure. Q_3 measures the percentage of predicted residues and is defined by

$$Q_3 = \sum_{i \in \{H,E,C\}} \frac{\text{correctly predicted}_i}{\text{observed}_i} \times 100 \tag{1}$$

HK-SSP achieved a Q_3 score of 63.4 while PHD achieved a Q_3 score of 71.9. Although HK-SSP does not perform as well as PHD, it does perform very well considering it uses much less information because it considers only a very small window around the residue.

4. Aligning a Sequence and Its Homologues

The homology kernel has applications beyond learning. We can view distances between points in the embedding as characterizing the similarity between sequences taking into account their homologues and secondary structure. We can use this to define an alignment algorithm that aligns a pair of proteins not only based on their sequence similarity, but based on the similarity of their homologues.

For our proof of concept application, we consider two sequences from the MEROPS database.[17] This database contains peptidase proteins grouped into a set of families which are grouped into a set of clans. Families are defined by a minimum amount of sequence similarity. Clans are defined by similar tertiary structure between families or similar ordering of active residues. Proteins within families, due to their sequence similarity, have strong local sequence alignments. Proteins within the same clan but in different families have very weak local sequence alignments. However, the MEROPS database provides alignments based on structure for the clans.

We consider two sequences from different families, but members of the same clan. Specifically, we consider the following two proteins: $P10274$ and $P00790$. The alignment provided in the MEROPS database is shown in Fig. 4. We applied BLASTP to the proteins which detected no statistical similarity between the two sequences.

We apply the local homology kernel to alignment as follows. We first apply BLAST to retrieve a set of 50 homologues for each sequence and align them with CLUSTALW.[26] We then apply the local homology kernel with $k = 3$ to compute the similarity between each column of the alignment and divide the values by 50 in order to get a score between 0 and 1. We then convert this score to a log odds ratio by applying the function $\log(x) - \log(0.05)$. We apply the Smith–Waterman local alignment algorithm, but instead of using the similarity between sequence symbols defined by a substitution, we use the log odds ratios computed by the homology kernel. Note that if we apply the homology kernel with $k = 1$ and no homologous sequences, the method is very similar to applying the traditional Smith–Waterman algorithm. The best local alignment using the homology kernel is shown in Fig. 5 which is consistent with the structural alignment.

MEROPS Alignment	
Family A1	`ACAEGCQAIV`**D**`TGTSLLTGPT`
Family A2A	`IGGQLKEALL`**D**`TGADDTVLEE`

Fig. 4. Alignments of two families $A1$ and $A2$, both part of the AA clan from the MEROPS database. The alignment is anchored at the active site highlighted in the alignment. The sequences shown in the alignment are the consensus sequences from the families.

Homology Kernel Alignment	
P00790 (family A1)	`Q A I V D T - G T S L L T G P T S P I A N = I Q S D I G A S E N`
Q85727 (family A2)	`E A L L D T = G A D M T V L P I A L F S S N T P L K N T S V L G`
Position Scores	`7 8 6 3 9 8 - 9 6 4-6 6 2 2 4 3 6 3 4 5-4 2 3 1 5-2-1 2 5 4 3 2`

Fig. 5. HK-ALIGN predicted alignments of two sequences from MEROPS database, one from family $A1$ and the other from family $A2$, both part of the AA clan. The alignment contains the active site as defined by the database. The position scores show the local homology kernel similarity between positions in the sequence.

5. Implementation Details

In this section, we describe the details of the construction of the HK. We describe this construction in the context of a general framework for constructing subsequence-based embeddings.

5.1. *The spectrum kernel and normalized spectrum kernels*

Each sequence is denoted $x_i \in \Sigma^*$ where Σ is the alphabet of amino acids. A k-mer, $a \in \Sigma^k$ is a sequence of length k. Sequence x_i contains k-mer a if $x_i = uav$. In this case we can write $a = x_i^p$ where a starts at the pth position of x_i. We define $|x_i|$ as the number of k-mers in x_i. Let $N(a, x_i)$ be the number of times k-mer a appears in sequence x_i.

Using this notation, the spectrum kernel[12] can be defined as

$$SK_k(x_i, x_j) = \sum_{m \in \Sigma^k} N(m, x_i) \times N(m, x_j) \tag{2}$$

Another equivalent way to define this kernel is through comparing all pairwise subsequences in the two sequences. If we denote x_i^p and x_j^q to be the pth subsequence of sequence x_i and the qth subsequence of sequence x_j respectively and $M(a, b)$ is a match function defined on k-mers which returns 1 if the k-mers match and 0 otherwise, we can define the spectrum Kernel as:

$$SK_k(x_i, x_j) = \sum_{x_i^p \in x_i, x_j^q \in x_j} M(x_i^p, x_j^q) \tag{3}$$

.

The spectrum kernel is biased toward sequences that contain multiple instances of k-mers. Suppose two strings s_1 and s_2 contain the k-mer a twice each. Then the term corresponding to a in Eq. (2) will be $N(a,s_1)*N(a,s_2)=4$. The contributions of two different shared k-mers is only 2.

We can define the normalized spectrum kernel as follows

$$NSK_k(x_i,x_j)=\sum_{m\in\Sigma^k}\min(N(m,x_i),N(m,x_j)) \tag{4}$$

The normalized kernel is an explicitly defined mapping. For all of our mappings we use the notation $\phi(x)$ to define the function that does the mapping and $\phi_j(x)$ to denote the jth dimension of the feature space. For the normalized spectrum kernel, let $\phi(x)$ map a sequence of maximum length n, $x\in\Sigma^n$, to a feature space of dimension $\Sigma^k n$ where each dimension is indexed by a k-mer a and an integer $1\le i\le n$. The mapping is as follows:

$$\phi_{w,i}(x)=\begin{cases}1 & \text{if k-mer } w \text{ appears at least } i \text{ times in } x\\ 0 & \text{otherwise.}\end{cases}$$

It is easy to see that $\langle\phi(x),\phi(y)\rangle=\sum\phi_{w,i}(x)\cdot\phi_{w,i}(y)$ implements Eq. (4). The contribution to $\langle\phi(x),\phi(y)\rangle$ for all terms in the sum indexed by w is the number of times w occurs in *both* sequences.

The normalized spectrum kernel has some very intuitive properties. For sequences x_i and x_j, the maximum of $NSK_k(x_i,x_j)$ is $\min(|x_i|,|x_j|)$. Similarly the norm of an image of a sequence x_i in the feature space, $||\phi(x_i)||^2=NSK_k(x_i,x_i)=|x_i|$. The normalized spectrum kernel is equivalent to considering the number of common k-mers between two sequences. Consider two sequences x_i and x_j. Let $||x_i\cap x_j||$ denote the number of shared substrings. The cosine of the angle θ between the two sequences $\cos\theta=\frac{||x_i\cap x_j||}{||\phi(x_i)||||\phi(x_j)||}$ which is very intuitive. We refer to this property as *normalization*.

The normalized kernel can be implemented in linear time. The complexity of performing a kernel computation between two sequences x_i and x_j is $O(k(|x_i|+|x_j|))$ if implemented using a trie data structure.[12]

5.2. *The sparse kernel*

As motivated in Fig. 1, exactly matching substrings fail to capture the intuitions behind local alignment. We introduce the sparse kernel, a new method for incorporating partial matches between substrings. The contribution of two substrings from different sequences is defined to be

$$SpK(a,b)=\alpha^{d_H(a,b)} \tag{5}$$

where $d_H(a,b)$ is the Hamming distance or number of mismatches between k-mers a and b, and α is a parameter. Note that since we can adjust α to be any value $0<\alpha<1$, the sparse kernel is a very flexible family of embeddings. For a low value of α, substrings with a few differences from each other are very far apart, while for a higher value of α, they are closer together.

In order to implement the sparse kernel, we augmented the alphabet Σ with a new character — the "wildcard" character denoted by "$*$". Let $\Sigma' = \Sigma \cup \{*\}$. "$*$" matches with every character $c \in \Sigma'$. A string with a wildcard at position i will match to all strings that differ from it only at position i.[a]

For $a \in \Sigma^k$, the sparse kernel is defined by the following explicit map

$$\phi_w(a) = \begin{cases} \sqrt{\alpha^\ell (1-\alpha)^{(k-\ell)}} & \text{if } w \in \Sigma'^k \text{ matches } a \in \Sigma^k \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

where ℓ is the number of wildcards in the w. Note that if $k = 1$, then a substring is mapped to itself with weight $\sqrt{(1-\alpha)}$ and to $*$ with weight $\sqrt{\alpha}$. In this case, $\langle \phi(a), \phi(a) \rangle = \sqrt{(1-\alpha)}\sqrt{(1-\alpha)} + \sqrt{\alpha}\sqrt{\alpha} = 1$ and $\langle \phi(a), \phi(b) \rangle = \sqrt{\alpha}\sqrt{\alpha} = \alpha$.

Claim 1. *The mapping in Eq.* (6) satisfies the property of Eq. (5).

Proof. Let a and b be two k-mers with Hamming distance $d_H(a, b) = d$. Then the dot product $\langle \phi(a), \phi(b) \rangle$ is the sum of $(\phi_w(a))^2$ over all w such that w matches a and b. Such k-mers w must contain wildcards on the mismatches (there are d such positions) and either the consensus letter or a wildcard at all other locations (the remaining $k - d$ positions). Therefore the dot product $\langle \phi(a), \phi(b) \rangle$,

$$\begin{aligned} \langle \phi(a), \phi(b) \rangle &= \sum_{i=0}^{k-d} \binom{k-d}{i} \left(\sqrt{\alpha^{d+i}(1-\alpha)^{k-d-i}} \right)^2 \\ &= \alpha^d (1-\alpha)^{k-d} \sum_{i=0}^{k-d} \binom{k-d}{i} \left(\frac{\alpha}{(1-\alpha)} \right)^i \cdot 1^{k-d-i} \\ &= \alpha^d (1-\alpha)^{k-d} \left(1 - \frac{\alpha}{1-\alpha} \right)^{k-d} = \alpha^d \end{aligned}$$

□

As above, we described the sparse kernel for a single string of length k. For a longer sequence, we can simply add the images of its substrings together. We refer to this embedding as the *unnormalized sparse kernel*, and it can be computed by

$$UnSpK_k(x_i, x_j) = \sum_{x_i^p \in x_i, x_j^q \in x_j} \alpha^{d_H(x_i^p, x_j^q)} \tag{7}$$

which is similar to Eq. (3). Alternatively, we can use the same trick as in the normalized spectrum kernel to construct the normalized sparse kernel and consider a feature space of dimension Σ'^{kn} and $\phi_{w,i}$ indexed both by a k-mer, $w \in \Sigma'^k$, and an integer i with the mapping

$$\phi_{w,i}(x) = \begin{cases} \sqrt{\alpha^\ell (1-\alpha)^{(k-\ell)}} & \text{if } w \in \Sigma'^k \text{ matches at} \\ & \quad \text{least } i \text{ substrings in } x \\ 0 & \text{otherwise.} \end{cases}$$

[a] Σ' is an auxiliary alphabet and serves only for the computation of the dot product.

The normalized sparse kernel has similar properties to the normalized spectrum kernel such as that the norm is square root of the number of substrings $||\phi(x_i)|| = \sqrt{|x_i|}$.

The intuition for partial matching is similar to that behind the mismatch kernel[11] which allows for partial matching of substrings. A problem with the mismatch kernel is that, due to the way it is defined, the ratio between the score contributed by a close match between two subsequences of length k and an exact match is very low. This means that the impact of near matches is insignificant relative to the impact of exact matches. In fact, the unnormalized sparse kernel with $\alpha = 1/k$ is an approximation for the mismatch kernel for one mismatch, a relation that can be derived by examining the values of the mismatch kernel for exact matching substrings and substrings with one or two mismatches. For the best performing $(5,1)$-mismatch kernel, this corresponds to an $\alpha = 0.2$. In our experimental results above, we show that values of α for protein classification which perform significantly better are higher values such as $\alpha = 0.4$. This is a significant advantage of the sparse kernel over the mismatch kernel.

5.3. *The groupings kernel*

In addition, we want to take into account information about similarities and differences between amino acids. Differences within a grouping of amino acids should correspond to closer images in the feature space than differences outside of groupings. The contribution between two k-mers a and b from different sequences is defined to be

$$SpKG(a,b) = \alpha^{d_M(a,b)}\beta^{d_G(a,b)} \tag{8}$$

where $d_G(a,b)$ is the number of mismatches between a and b within a group and $d_M(a,b)$ is the number of mismatches between groups and $0 < \alpha < \beta < 1$. α and β are parameters to the sparse kernel with groupings to allow flexibility in the differences between the two types of mismatches.

The sparse kernel with groupings is implemented by augmenting the alphabet with a special symbol for each group. For a set of groups G, the augmented alphabet Σ'' is defined: $\Sigma'' = \Sigma \cup \{*\} \cup G$. A symbol c matches itself, the symbol corresponding to its group and the $*$ symbol. A string $w \in \Sigma''^k$ matches any string and $a \in \Sigma^k$ if each of their positions match. Note that $a \in \Sigma^k$ matches 3^k strings in Σ''^k.

The sparse kernel with groupings can be implemented using ϕ defined as follows:

$$\phi_w(a) = \begin{cases} \sqrt{(\alpha)^M(\beta-\alpha)^G(1-\beta)^{k-M-G}} & \text{if k-mer } a \text{ matches } w \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where M is the number of wildcards in w and G is the number of groups characters in w.

Claim 2. *The mapping in Eq.* (9) *satisfies the property of Eq.* (8).

Proof. The dot product $\langle \phi(a), \phi(b) \rangle$ is the sum of $\phi_w(a)$ over all k-mers $w \in \Sigma''^k$ such that w matches both a and b. We look at a specific k-mer $w \in \Sigma''^k$ that matches both a and b. The contribution of w to the sum $\langle \phi(a), \phi(b) \rangle$ is the weighted product $\phi_w(a)$ over all w's positions' weights. w must contain the wildcard at all positions where a and b disagree (i.e. contain characters from different groups). That contributes α^M to $\phi_w(a)$. Next, at all positions where the mismatches are between groups, we must put either a wildcard or the group character. Since there are g such positions, we have $\sum_{i=0}^{g} \binom{g}{i}$ possibilities to locate the group characters, each contributing $(\beta - \alpha)^i \alpha^{g-i}$ to $\phi_w(a)$. Finally we have all the matches where we can locate either a wildcard, the appropriate group character or the character itself. Let $e = k - m - g$. Then we have $\sum_{j=0}^{e} \binom{e}{j}$ possibilities to locate the group characters, and $\sum_{\ell=0}^{e-j} \binom{e-j}{\ell}$ possibilities to locate the wildcard. By the definition of $\phi_w(a)$ we obtain the below equation:

$$\begin{aligned}
&\langle \phi(a), \phi(b) \rangle \\
&\quad = \alpha^m \sum_{i=0}^{g} \binom{g}{i} (\beta - \alpha)^i \alpha^{g-i} \sum_{j=0}^{e} \binom{e}{j} (\beta - \alpha)^j \sum_{\ell=0}^{e-j} \binom{e-j}{\ell} \alpha^{\ell} (1 - \beta)^{e-j-\ell} \\
&\quad = \alpha^m \sum_{i=0}^{g} \binom{g}{i} (\beta - \alpha)^i \alpha^{g-i} \sum_{j=0}^{e} \binom{e}{j} (\beta - \alpha)^j (\alpha + 1 - \beta)^{e-j} \\
&\quad = \alpha^m \sum_{i=0}^{g} \binom{g}{i} (\beta - \alpha)^i \alpha^{g-i} 1^e \\
&\quad = \alpha^m \beta^g
\end{aligned} \tag{10}$$

where the latter is obtained by applying the binomial identity. □

We extend the sparse kernel with groupings from its definition for a single substring of length k to a sequence in the same manner as the sparse kernel.

5.4. *Efficient implementation of the sparse and groupings kernels*

Now, we describe an efficient data structure to compute each entry in the kernel matrix. In this data structure we generalize the *mismatch tree* data structure used in[11] what yields an improved complexity. Since we use an extended trie,[1] we first briefly describe such a tree. A trie over an alphabet Σ is a Σ-regular directed tree. Each edge emanating from an internal node in the tree is labeled with a symbol from Σ. An internal node at depth i correspond to a prefix of length i obtained by concatenating the labels along the path from the root to the node.

In practice, we do not maintain such a tree rather compute the paths defined by each k-mer at a time. This prevents constructing unused subtrees. We process each k-mer of each of the sequences at a time and sum its contribution to the total sum. We start with the normalized kernel. This kernel was implemented by using a standard trie as was used by Leslie *et al.*[11] In order to determine the contribution of a word A to the dot product, we start at the root with a list of all k-mers for each sequence. We follow the path from the root as defined by A while at every internal node at depth i we remove from the list the words that do not fit the path (have a different character at position i). When we arrive at a node at depth k we have for each of the sequences, how many such k-mers it contains.

The implementation of the sparse and grouping kernels required a more sophisticated technique. Our improvement over[11] is by introducing the wildcard into the data structure and hence avoiding the explicit computation of the d-distance neighborhood of each k-mer. Therefore, in addition to the Σ symbols we have also the wildcard symbol so the out-degree of each internal node is $|\Sigma'|$. Every edge in the tree has a weight α if it leads to the wild card and $1 - \alpha$ otherwise. Therefore, the path of length k has weight of the contribution of the word defined by it. While traversing a path in the tree defined by a k-mer, at every internal node we bifurcate to either the descendant defined by the next character in the k-mer or to the descendant representing the wildcard. Therefore, the number of nodes visited per a single k-mer is 2^k. In addition, each move requires eliminating from the list of the ancestor node the k-mers which do not match. Therefore, we obtain a total time complexity of $O(2^k(|x_i| + |x_j|))$ and $O(nk)$ memory.

The same data structure is used also in the grouping kernel with the addition that at every node, also the descendant representing the appropriate group of the next character. Therefore, a complexity of $O(3^k(|x_i| + |x_j|))$ is obtained for the grouping kernel.

This is much faster than the mismatch kernel complexity between a pair of sequences which is $O(k^{(d+1)}|\Sigma|^d(|x_i|+|x_j|))$[11] which contains a factor of $|\Sigma|$ because each k-mer is mapped to a large neighborhood. Several recent variants to the mismatch kernel have recently been presented which have similar complexity to the sparse kernel.[13] A complete discussion on efficient implementations of kernels is available in Shawe-Taylor and Christianini.[23]

5.5. *Constructing the homology kernel*

For simplicity, we first consider the similarity between two short regions of length k where each region contains k columns of the alignment of m sequences. The homology kernel is exactly the normalized sparse kernel with groupings applied to these sets of substrings.

Consider the aligned columns of position p in one sequence x_i to position q in the another sequence x_j. Let x_{ir} denote the rth homologue of sequence x_i. Let x_i^p denote the substring of length k centered at position p in sequence x_i and x_{ir}^p denote the substring of length k of the rth aligned sequence. We consider the similarity between the set of sequences x_{ir}^p and x_{ir}^q for all $1 \leq r \leq m$ by applying the normalized sparse kernel with groupings to these sets of substrings. We refer to this embedding as the *local homology kernel*.

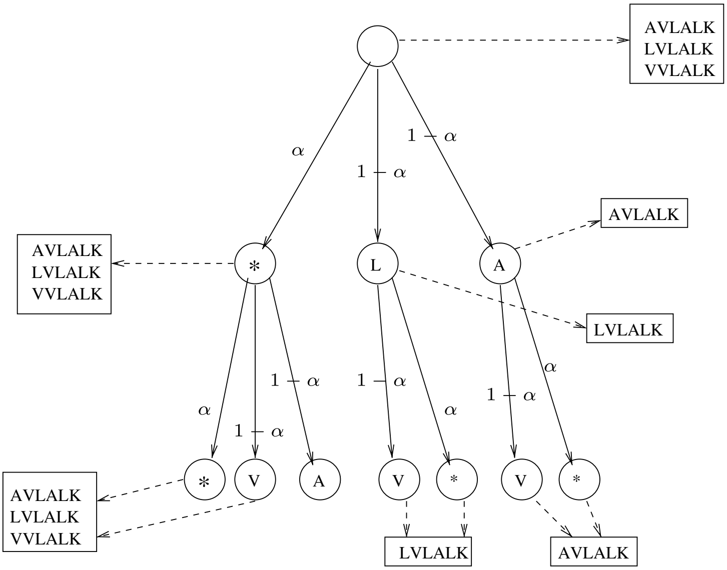


Fig. 6. The weighted trie used for computing the dot product of two embedded strings. At the root we have all k-mers in the sequence. While moving to a descendant only k-mers with the appropriate prefix remain. When moving to a descendant representing the wildcard, all k-mers from the previous ancestor remain.

Between these sets of aligned short regions the local homology kernel defines a similarity score. We can think of the score between the two sets of k columns as similar to measuring the number of subsequences of length k that the two alignments share. This score has the property, the maximum score between the two sequences is m when the two columns are identical and takes into account partial matching.

We can define the homology kernel over two sets of aligned sequences by considering the set of substrings for each set as the complete set of substrings in the sequence and its homologues. Due to the normalization property that the maximum score between two sets of aligned sequences is lm where l is the number of k length substrings in the shorter sequence. Similarly, the norm of m aligned set of sequences with l substrings is $\sqrt{lm}$.

5.6. *Learning with SVMs*

The SVM algorithm, is a class of supervised learning algorithms first introduced by Vapnik.[29] As a first stage, the algorithm is trained by training examples. Each training example consists of a sequence x_i and a label $y^i \in \{+1, -1\}$. Let $\phi(x)$ denote the mapping of the sequence x to its image in the feature space defined by the homology kernel. The goal of the learning algorithm is to obtain a *classifier* which in our representation is a vector w in the feature space representing the normal of the hyperplane defined by the classifier. After the training stage The classification of an unknown sequence x is simply the sign of the dot product between its image and the weight vector $\mathrm{sign}(w \cdot \phi(x))$.

We can use the SVM algorithm to obtain such a hyperplane which maximizes the margin. The SVM algorithm minimizes the following objective function

$$\frac{||w||^2}{2} + C \sum_i \zeta_i \tag{11}$$

with constraints

$$\phi(x_i) \cdot w + b \geq +1 - \zeta_i \quad \text{for } y_i = +1 \tag{12}$$

$$\phi(x_i) \cdot w + b \leq -1 + \zeta_i \quad \text{for } y_i = -1 \tag{13}$$

$$\zeta_i \geq 0 \tag{14}$$

The two terms in the optimization represent a trade off between a hyperplane that will maximized the margin versus minimizing the errors over the training set where C is the scaling factor for this trade off. The first term represent the flatness of the hyperplane while the second term represents the soft margin, where the ζ_i are non-zero for any training examples that are on the wrong side of the hyperplane (training errors). There exist many well known efficient techniques for solving the SVM optimization problem.[22]

However, in many cases the feature space is exponentially large and turns the problem of computing a dot product in that space into computationally expensive. The *kernel trick* enables us to bypass this obstacle. The key point is that since we are only interested in the dot product $\langle x_i, x_j \rangle$ and not in the point themselves, we can compute this product directly and efficiently without explicitly mapping into the feature space and subsequently multiplying.

6. Discussion and Further Research

We have presented the homology kernel, a biologically motivated sequence embedding which takes into account intuitions from local alignments and incorporates information from homologues and secondary structure. We have compared our method to the previously presented mismatch and spectrum kernels over a single benchmark. However, we have not compared our method to the multitude of methods presented since the Mismatch Kernel and over the several different benchmarks used in other recent works. Only a direct comparison of all of these methods over many benchmarks would conclusively answer questions about the true performance of each method and is beyond the scope of the paper.

Within the framework of the homology kernel, there are many possible settings for the sparse kernel parameters k, α, and β and many possible ways to generate homologues for the protein sequences. A direction for future research is to more thoroughly determine which parameter settings are optimal for modeling protein sequences.

We also plan a more thorough benchmark of HK-ALIGN to better set the parameters to be able to align distant homologues.

A planned application of the homology kernel is HK-SEARCH which is given a query protein sequence and obtains a set of homologous sequences similarly to PSI-BLAST. A major difference between HK-SEARCH and PSI-BLAST is that since HK-SEARCH is based on the homology kernel, it compares the query sequence and its homologues to other sequences in the database *and their homologues* which are precomputed. This is similar to the approach of the IMPALA program[21] but they are different in the fact that the actual sequences composing the alignments are used to compute the score instead of profiles.

Acknowledgments

E. Eskin is partially supported by National Science Foundation Grant No. 0513612 and National Institutes of Health Grant No. 1K25HL080079.

References

1. Aho AV, Ullman JD, Hopcroft JE, *Data Structures and Algorithms*, Addison Wesley, 1983.
2. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ, A basic local alignment search tool, *J Mol Biol* **215**:403–410, 1990.
3. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman, Gapped DJ, BLAST and PSI-BLAST: A new generation of protein database search programs, *Nucleic Acids Res* **25**:3389–3402, 1997.
4. Bairoch A, The SWISS-PROT protein sequence data bank: Current status, *Nucleic Acids Res* **22**(17):3578–3580, 1994.
5. Cuff J, Barton J, Evaluation and improvement of multiple sequence methods for protein secondary structure prediction, *PROTEINS: Structure, Function, and Genetics* **34**:508–519, 1999.
6. Eddy SR, Multiple alignment using hidden Markov models, in Rawlings C (ed.), *Proc Third Int Conf Intelligent Sys Mol Biol*, AAAI Press, pp. 114–120, 1995.
7. Eskin E, Snir S, The homology kernel: A biologically motivated sequence embedding into euclidean space, *Proc. 2005 IEEE Conf. Comput Intelligence in Bioinform Comput Biol* 2005.
8. Gribskov M, Robinson NL, Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching, *Computers and Chemistry* **20**(1):25–33, 1996.
9. Jaakkola T, Diekhans M, Haussler D, A discriminative framework for detecting remote protein homolo gies, *J Comput Biol*, 2000.
10. Kuang R, Ie E, Wang K, Wang K, Siddiqi M, Freund Y, Leslie C, Profile-based string kernels for remote homology detection and motif extraction. *Proc/IEEE Comput Syst Bioinform Conf, CSB. IEEE Comput Sys Bioinform Conf*, pp. 152–160, No xx, 2004.
11. Leslie C, Eskin E, Noble W S, Mismatch string kernels for SVM protein classification. In *Proceedings of Advances in Neural Information Processing Systems 15 (NIPS)*, Vancouver, Canada, 2002.
12. Leslie C, Eskin E, Noble WS, The spectrum kernel: A string kernel for SVM protein classification, *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, Kaua'i, Hawaii, 2002.

13. Leslie CS, Kuang R, Fast kernels for inexact string matching, *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop (COLT-2003)*, pp. 114–128, 2003.
14. Murzin AG, Brenner SE, Hubbard T, Chothia C, SCOP: A structural classification of proteins database for the investigation of sequences and structures, *J Mol Biol* **247**:536–540, 1995.
15. Park J, Karplus K, Barrett C, Hughey R, Haussler D, Hubbard T, Chothia C, Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods, *J Mol Biol* **284**(4):1201–1210, 1998.
16. Rangwala H, Karypis G, Profile-based direct kernels for remote homology detection and fold recognition, *Bioinformatics* **21**:4239–4247, 2005; 10.1093/bioinformatics/bti687.
17. Rawlings ND, O'Brien EA, Barrett AJ, Merops: the protease database, *Nar* **30**:343–346, 2002.
18. Rost B, Sander C, Prediction of protein secondary structure at better than 70% accuracy, *J Mol Bio* **232**:584–99, 1993.
19. Saigo H, Vert J-P, Ueda N, Akutsu T, Protein homology detection using string alignment kernels, *Bioinform* **20**:1682–1689, 2004; 10.1093/bioinformatics/bth141.
20. Sasson O, Vaaknin A, Fleischer H, Portugaly E, Bilu Y, Linial N, Linial M, Protonet: hierarchical classification of the protein space, *Nucleic Acids Res* **31**(1):348–352, 2003.
21. Schaffer AA, Wolf YI, Ponting CP, Koonin EV, Aravind L, Altschul SF, Impala: matching a protein sequence against a collection of psi-blast-constructed position-specific score matrices, *Bioinform* **15**:1000–1011, 1999.
22. Schölkopf B, Smola AJ, *Learning with Kernels*, MIT Press, 2002.
23. Shawe-Taylor J, Cristianini N, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
24. Smith T, Waterman M, Identification of common molecular subsequences, *J Mol Biol* **147**:195–197, 1981.
25. Taylor WR, The classification of amino acid conservation, *J Theor Biol* **119**:205–218, 1986.
26. Thompson JD, Higgins DG, Gibson TJ, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice, *Nucleic Acids Res* **22**:4673–4680, 1994.
27. Thomson R, Hodgman C, Yang ZR, Doyl AK, Characterising proteolytic cleavage site activity using bio-basis function neural network, *Bioinform* **21**:1741–1747, 2003.
28. Thomson R, Yang ZR, A novel basis function neural network, *Proceedings of the 9th International Conference on Neural Information Processing, ICONIP '02*, Vol. 1, pp. 441–446, 2002.
29. Vapnik V, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
30. Weston J, Leslie C, Zhou D, Elisseeff A, Noble W, Semi-supervised protein classification using cluster kernels. *Proceedings of Neural Information Processing Systems 2003 (NIPS 2003)*, 2003.
31. Yona G, Levitt M, Within the twilight zone: A sensitive profile-profile comparison tool based on information theory, *J Mol Bio* **315**:1257–1275, 2001.

Eleazar Eskin's research interests are in the relationship between genetic variation and disease in humans at the intersection of genetics, genomics and bioinformatics. Through computational analyses of human variation data, he is working to understand the genetic basis of human disease. Eskin received his PhD in 2002 from

Columbia University and spent one year as a post-doctoral fellow at the Hebrew University in Jerusalem, Israel. Prior to joining UCLA, Eskin was an Assistant Professor in Residence in the Computer Science and Engineering department at the University of California, San Diego. He is also affiliated with California Institute for Telecommunications and Information Technology (Calit2).

Sagi Snir received his BA degree from Bar Ilan University at Israel majoring in computer science and economics. He received the MSc and PhD degrees in computer science from the Technion, Israel. After PhD he was a postdoctoral researcher at the math and CS departments in UC Berkeley and is now a senior lecturer at Netanya college and a research fellow at Inst. of Evolution in Haifa U. Before working on the PhD, he worked in various information technologies companies, including IBM Haifa Research Lab. His main research interest is combinatorial and statistical problems in computational biology and, in particular, genomics and phylogenetics.