Chapter 26
STRIKE: A Protein–Protein Interaction Classification Approach

Nazar Zaki, Wassim El-Hajj, Hesham M. Kamel, and Fadi Sibai

Abstract Protein–protein interaction has proven to be a valuable biological knowledge and an initial point for understanding how the cell internally works. In this chapter, we introduce a novel approach termed STRIKE which uses String Kernel to predict protein–protein interaction. STRIKE classifies protein pairs into "interacting" and "non-interacting" sets based solely on amino acid sequence information. The classification is performed by applying the string kernel approach, which has been shown to achieve good performance on text categorization and protein sequence classification. Two proteins are classified as "interacting" if they contain similar substrings of amino acids. Strings' similarity would allow one to infer homology which could lead to a very similar structural relationship. To evaluate the performance of STRIKE, we apply it to classify into "interacting" and "non-interacting" protein pairs. The dataset of the protein pairs are generated from the yeast protein interaction literature. The dataset is supported by different lines of experimental evidence. STRIKE was able to achieve reasonable improvement over the existing protein–protein interaction prediction methods.

Keywords Pattern classification and recognition · Protein–protein interaction · Protein sequence analysis · Amino acid sequencing · Biological data mining and knowledge discovery

1 Introduction

The prediction of protein–protein interaction (PPI) is one of the fundamental problems in computational biology as it can aid significantly in identifying the function of newly discovered proteins. Understanding PPI is crucial for the investigation

N. Zaki (✉)
Bioinformatics Laboratory, Department of Intelligent Systems,
College of Information Technology, UAE University, 17551 Al-Ain, UAE
e-mail: nzaki@uaeu.ac.ae

H.R. Arabnia and Q.-N. Tran (eds.), *Software Tools and Algorithms for Biological Systems*, Advances in Experimental Medicine and Biology 696,
DOI 10.1007/978-1-4419-7046-6_26,

of intracellular signaling pathways, modeling of protein complex structures and gaining insights into various biochemical processes. To enhance this understanding, many experimental techniques have been developed to predict the proteins' physical interactions which could lead to the identification of the functional relationships between proteins. These experimental techniques are, however, very expensive, significantly time consuming and technically limited which creates a growing need for the development of computational tools that are capable of identifying PPIs. To this end, many impressive computational techniques have been developed. Each of these techniques has its own strengths and weaknesses, especially with regard to the sensitivity and specificity of the method. Some of the techniques such as the Association Method (AM) [1], Maximum Likelihood Estimation (MLE) [2], Maximum Specificity Set Cover (MSSC) [3] and Domain-based Random Forest [4] have used domain knowledge to predict PPI. The motivation of these techniques was that molecular interactions are typically mediated by a great variety of interacting domains. Another method called Protein–Protein Interaction Prediction Engine (PIPE) [5] was developed based on the assumption that some of the interactions between proteins are mediated by a finite number of short polypeptide sequences. These sequences are typically shorter than the classical domains and are used repeatedly in different proteins and contexts within the cell. However, identifying domains or short polypeptide sequences is a long and computationally expensive process. Moreover, these techniques are not universal because their accuracy and reliability are dependent on the domain information of the protein partners.

In this chapter, we introduce a novel approach termed STRIKE which uses string kernel (SK) approach to predict PPI. This has been shown to achieve good performance on text categorization tasks [6] and protein sequence classification [7]. The basic idea of this approach is to compare two protein sequences by looking for common subsequences of a fix-length. The string kernel is built on the kernel method introduced by Haussler et al. [8] and Watkins et al. [9]. The kernel computes similarity scores between protein sequences without ever explicitly extracting the features. The subsequence is any ordered sequence of amino acids occurring in the protein sequence and is not necessarily contiguous. The subsequences are weighted by an exponentially decaying factor of their full length in the sequence, hence emphasizing those occurrences that are close to contiguous.

We understand that the subsequence similarity between two proteins may not necessarily indicate interaction. However, it is an evidence that we cannot ignore. Subsequence similarity would allow one to infer that homology and homologous sequences usually have the same or very similar structural relationships.

A drawback of this approach emerges when the level of similarity between the protein pairs is too low to pick up interaction. The reasonable explanation is that in the case of low similarity sequence, there are always similar patterns of identical amino acid residues which could be seen in the two sequences. The pattern of sequence similarity reflects the similarity between experimentally determined structures of the respective proteins or at least corresponds to the known key elements of one such structure [10]. Structural evidence indicates that interacting pairs of close homologs usually interact in the same way [11]. Other evidences are derived using a

combination of some genomic features such as structurally known interacting Pfam domains and sequence homology as a means to assign reliability to the PPIs in *Saccharomyces cerevisiae* [12]. The likelihood ratio in this study expresses the reliability of such genomic feature. In our case, there is no doubt that the SK method will be a good technique of reflecting homology between protein pairs. The intensive comparison between subsequences existing in protein pairs may capture structural domain knowledge or typically subsequences that are shorter than the classical domains and could appear repeatedly in the protein pairs of interest. We are also encouraged by the success of a recently published work using pair-wise alignment as a way to extract meaningful features to predict PPI. The PPI based on Pairwise Similarity (PPI-PS) method consists of a representation of each protein sequence by a vector of pair-wise similarities against large subsequences of amino acids created by a shifting window which slides over concatenated protein training sequences. Each coordinate of this vector is typically the E-value of the Smith–Waterman score [13]. One major drawback of the PPI-PS is that each protein is represented by computing the Smith–Waterman score against large subsequences created by concatenating protein training sequences. However, comparing short sequence to a very long one will result in some potentially valuable alignments to be missed out. The SK, however, tackles this weakness by capturing any match or mismatch existing in the protein sequence of interest.

2 Method

In order to classify protein pairs as "interacting" or "non-interacting", STRIKE performs the following steps: (1) data preparation step in which protein pairs in the dataset are concatenated; (2) the training step in which the support vector machine (SVMs) classifiers are constructed and (3) the testing step which uses SVMs to determine whether the protein pair is "interacting" or "non-interacting". Steps (2) and (3) require the computation of kernel similarity scores between protein sequences. The feature space is generated by all subsequences of bounded length. In order to derive the SK, we start from the features and then compute their inner product. SK maps strings to a feature vector indexed by all k-tuples of amino acids. A k-tuple will have a non-zero entry if it occurs as a subsequence anywhere (not necessarily contiguous) in the string.

In order to derive the string kernel (SK), we start from the features and then compute their inner product. Hence, the criterion of satisfying the Mercer's condition (positive semi-definiteness) automatically applies here. It maps the amino acid sequence (string) to a feature vector indexed by all k-tuples of amino acids. A k-tuple has a non-zero entry if it occurs as a subsequence anywhere (not necessarily contiguous) in the protein sequence of interest. The weighting of the feature will be the sum over the occurrences of the k-tuple in the protein sequence.

Following Cristianini et al. [14] and Lodhi et al. [6], the string kernel can be defined as follows.

Let $\sum$ be a finite set of amino acids. A string is a finite sequence of amino acids from $\sum$, including the empty sequence. For protein sequences s, t, we denote by $|s|$ the length of the sequence $s = s_1 \ldots s_{|s|}$ and by st the string obtained by concatenating the sequences s and t. The string $s[i:j]$ is the substring $s_i \ldots s_j$ of s. We say that u is a subsequence of s, if there exist indices $i = (i_1, i_2, \ldots, i_{|u|})$, with $1 \leq i_1, i_2, \ldots, i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$ for $j = 1, 2, \ldots, |u|$ or $u = s[i]$ for short. The full length $l(i)$ of the subsequence in s is $i_{|u|} - i_1 + 1$. We denote by $\sum^n$ the set of all finite amino acid sequences of length n and by $\sum^*$ the set of all protein (strings):

$$\sum^* = \bigcup_{n=0}^{\infty} \sum^n. \tag{1}$$

We now define the feature space $F_n = \Re^{\Sigma^n}$. The feature mapping ϕ for protein sequence s is given by defining the u coordinate $\phi_u(s)$ for each $u \in \sum^n$. We define

$$\phi_u(s) = \sum_{i:u=s[j]} \lambda^{l(i)} \tag{2}$$

for some $\lambda \leq 1$. These features measure the number of subsequence occurrences in the protein sequence s weighting them according to their lengths. Hence, the inner product of the feature vectors for two protein sequences s and t gives a sum over all common subsequences weighted according to their frequency of occurrence and lengths:

$$k_n(s,t) = \sum_{u \in \sum^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle \tag{3}$$

$$= \sum_{u \in \sum^n} \sum_{i:u=s[i]} \lambda^{l(i)} \sum_{j:u=t[j]} \lambda^{l(j)} \tag{4}$$

$$= \sum_{u \in \sum^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{l(i)+l(j)} \tag{5}$$

This technique can be illustrated by a simple example. Consider the following four concatenated protein pairs sequences:

$>$ s1	$>$ s2	$>$ s3	$>$ 4
lql	*lqal*	*hgs*	*gqsl*

In this case, sequences s1 and s2 belong to the interacting set and the sequences s3 and s4 belong to the non-interacting set. For each substring, there is a dimension of feature space, and the value of such coordinate depends on how frequently and compactly such string (such as the ones highlighted in bold) is embedded in the protein sequences of interest. Let us first assume that we are comparing the first two concatenated protein sequences s1 and s2, where there exists one string in each sequence. For simplicity, we set the length of substring to two. In other words, these sequences are implicitly transformed into feature vectors, where each feature

Table 1 Mapping two strings "*lql*" and "*lqal*" to six-dimensional feature spaces

	lq	ll	ql	la	qa	al
$\phi(lql)$	λ^2	λ^3	λ^2	0	0	0
$\phi(lqal)$	λ^2	λ^4	λ^3	λ^3	λ^2	λ^2

Table 2 Mapping all the strings to dimensional feature spaces

	S1(lql)	S2(lqal)	S3(hgs)	S4(gqsl)
S1(lql)	–	$k(lql, lqal) = 0.102$	$k((lql), hgs) = 0$	$k(lql, gqsl) = 0.031$
S2(lqal)		–	$k((lqal), hgs) = 0$	$k(lqal, gqsl) = 0.016$
S3(hgs)			–	$k(hgs, gqsl) = 0.031$
S4(gqsl)				–

vector is indexed by the substrings of length two. The six-dimensional feature space, feature vectors $\phi(lql)$, $\phi(lqal)$ and the corresponding kernel are given in Table 1.

In this case, the un-normalized kernel $k(lql, lqal)$ can clearly be computed as $\lambda^4 + \lambda^7 + \lambda^5$. Assuming that the decay factor λ is equal to 0.5, $k(lql, lqal) = 0.102$. Similarly, the remaining kernels are computed and summarized in Table 2.

From Table 2, it is clearly shown that the proposed method is able to capture the potential interaction between the two sequences "*lql*" and "*lqal*".

3 Experimental Work

STRIKE is implemented in Perl. However, the SK component is a modified version of Lodhi's [6] approach which was implemented in C++. SK is based on a simple gradient-based implementation of SVM [15] known as "Adatron". Many different techniques have been applied to speed up the computation of the kernel matrix. First, the program reads a single file containing training protein pairs and test protein pairs sets. Second, two files containing the indexes of training set and test set are added.

STRIKE uses as input a file containing the training and testing protein pairs. A second file containing all the protein sequences from the yeast *S. cerevisiae* is included to retrieve the corresponding protein sequences. As STRIKE runs from the command line, the user enters the name of the file containing the protein pairs and the number of positive training examples, the negative training examples, the positive testing examples and the negative testing examples. The user has the option to change the weighted decay factor (λ) and the size of the substring (n). Both parameters have effects on the generalization performance of an SVM learner that manipulates the information encoded in a string kernel. SK weighs the substrings of the amino acids according to their proximity in the protein sequence. This is the parameter that controls the penalization of the interior gaps in the substrings. STRIKE outputs the classification results which contain the accuracy, precision, recall and the F1 value. The performance of STRIKE is measured by how well it can distinguish between "interacting" and "non-interacting" protein pairs. STRIKE

was used to classify between 100 "interacting" protein pairs (157 proteins) and 100 "non-interacting" protein pairs (77 proteins). The datasets were randomly selected by Sylvain et al. [5] and used to evaluate the PIPE's performance. It was generated from the yeast protein interaction literature for which at least three different lines of experimental evidence supported the interaction. The dataset can be downloaded from http://faculty.uaeu.ac.ae/nzaki/STRIKE.htm.

4 Results and Discussion

The performance of STRIKE was measured by how well the system can recognize interacting protein pairs. In order to analyze the evaluation measures in PPI prediction, we first explain the contingency table (Table 3). The entries of the four cells of the contingency table are described as follows:

tp = number of interacting sequences classified as "interacting".
fn = number of non-interacting sequences classified as "interacting".
fp = number of interacting sequences classified as "non-interacting".
tn = number of non-interacting sequences classified as "non-interacting".
$n = tp + fn + fp + tn$ (total number of sequences).

The information encoded in the contingency table is used to calculate the following PPI evaluation measures:

$$\text{Precision}(\text{Pr}) = tp/(tp + fp) \tag{6}$$

$$\text{Recall}(\text{Re}) = tp/(tp + fn) \tag{7}$$

$$\text{F1} = 2[(\text{Pr} \times \text{Re})/(\text{Pr} + \text{Re})] \tag{8}$$

$$\text{Accuracy} = (tp + tn)/n. \tag{9}$$

To evaluate the performance of STRIKE, we first grouped the 100 "interacted" protein pairs into two sets: A (50 pairs) and B (50 pairs). We also grouped the 100 "non-interacted" protein pairs into two sets: C (50 pairs) and D (50 pairs). We then combine A with C to create a training dataset and B with D to create a testing dataset.

To study the effectiveness of varying weighted decay factor (λ) on the generalization performance, a series of experiments were conducted. The value of the subsequence length (n) was kept fixed to 2. The analysis results are summarized in Fig. 1. Figure 1a shows the relation between different values of λ and the

Table 3 The contingency table

	Interacting sequence	Non-interacting sequence
Classified interacting	True positives (tp)	False negatives (fn)
Classified non-interacting	False positives (fp)	True negatives (tn)

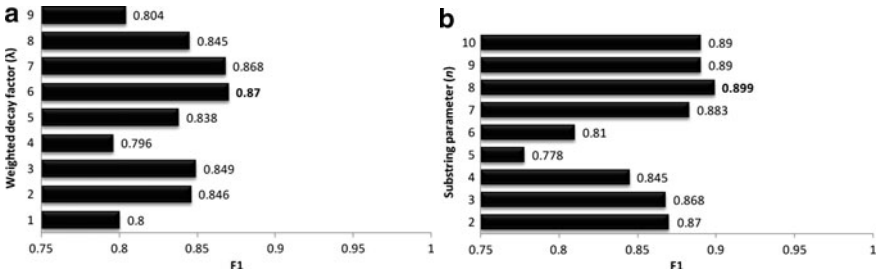

Fig. 1 Performance of STRIKE with a varying weight decay factor (λ) and subsequence length (n) values

corresponding influence of F1. It is interesting to point out that F1 peaks at a value of 0.6.

Further experiments were conducted to study the effectiveness of varying the subsequence length (n). The value of λ was set to 0.6. The analysis results are shown in Fig. 1b. It is apparent in this results that F1 peaks at a subsequence length of 8.

In this case, STRIKE was able to achieve 89% accuracy, 0.831 precision, 0.98 recall and 0.899 F1. The substring parameter (n) and the weighted decay factor (λ) were set to 8 and 0.6, respectively.

The obtained results are superior to PIPE. PIPE performance resulted in 0.61 sensitivity, 0.89 specificity and 0.75 overall accuracy when tested to detect yeast protein interaction pairs. In addition to its superior accuracy, STRIKE has two further advantages when compared to PIPE. First, the PIPE method is computationally intensive and the evaluation of PIPE's performance over the same dataset took around 1,000 h of computation time, compared to only 54.9 min using STRIKE. Second, as indicated by the PIPE authors, their method is expected to be weak if it is used for the detection of novel interactions among genome-wide large-scale datasets which is not the case using STRIKE.

5 Conclusion

In this chapter, we introduced a novel approach termed STRIKE (String Kernel) which uses String Kernel method to predict protein–protein interaction. STRIKE classifies protein pairs into "interacting" and "non-interacting" sets that is based solely on amino acid sequence information. STRIKE makes no use of prior knowledge, yet it has been used with considerable success.

One of STRIKE's strength is that the subsequences are weighed by an exponentially decaying factor of their full length in the sequence, hence emphasizing those occurrences that are close to being contiguous. However, replacing the weighted decaying factor using a protein weight matrix (scoring matrix) will add sensitivity to the comparison as it implicitly represents a particular theory of protein sequence

evolution. The parallel processing framework to compute the string kernel presented in this chapter will be implemented and assessed in a large dataset such as the one proposed by Xue-Wen et al. [4].

References

1. Sprinzak E, Margalit H (2001) Correlated sequence-signatures as markers of protein-protein interaction. J Mol Biol 311: 681–692
2. Deng M, Mehta S, Sun F, Cheng T (2002) Inferring domain-domain interactions from protein-protein interactions. Genome Res 12: 1540–1548
3. Huang TW, Tien AC, Huang WS, Lee YC, Peng CL, Tseng HH, Kao CY, Huang CY (2004) POINT: a database for the prediction of protein-protein interactions based on the orthologous interactome. Bioinformatics 20: 3273–3276
4. Xue-Wen C, Mei L (2005) Prediction of protein–protein interactions using random decision forest framework. Bioinformatics 21: 4394–4400
5. Sylvain P, Frank D, Albert C, Jim C, Alex D, Andrew E, Marinella G, Jack G, Mathew J, Nevan K, Xuemei L, Ashkan G (2006) PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. BMC Bioinformatics 7: 365
6. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C (2002) Text classification using string kernels. J Mach Learn Res 2: 419–444
7. Zaki NM, Deris S, Illias RM (2005) Application of string kernels in protein sequence classification. Appl Bioinformatics 4: 45–52
8. Haussler D (1999) Convolution kernels on discrete structures. Technical Report UCSC-CRL-99–10, University of California, Santa Cruz
9. Watkins C (2000) Dynamic Alignment Kernels. Advances in Large Margin Classifiers. Cambridge, MA, MIT Press, 39–50
10. Koonin EV, Galperin MY (2002) Sequence-Evolution-Function: Computational Approaches in Comparative Genomics. Dordrecht, Kluwer Academic Publishers
11. Zaki NM (2009) Protein-protein interaction prediction using homology and inter-domain linker region information. In: Lecture Notes in Electrical Engineering, Vol. 39. New York, Springer, 635–645
12. Patil A, Nakamura H (2005) Filtering high-throughput protein-protein interaction data using a combination of genomic features. BMC Bioinformatics 6: 100
13. Zaki NM, Lazarova-Molnar S, El-Hajj W, Campbell P (2009) Protein-protein interaction based on pairwise similarity. BMC Bioinformatics 10: 150
14. Cristianini N, Shawe-Taylor J (2000) An Introduction to Support Vector Machines. Cambridge, UK: Cambridge, University Press
15. Friess T, Cristianini N, Campbell C (1998) The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. 15th International Conference on Machine Learning. Morgan Kaufmann, CA, 188–196