BMC Bioinformatics

Research

Classification of protein sequences by means of irredundant patterns

Matteo Comin* and Davide Verzotto

Address: Department of Information Engineering, University of Padova, Italy

E-mail: Matteo Comin* - ciompin@dei.unipd.it; Davide Verzotto - verzotto@dei.unipd.it
*Corresponding author

Abstract

Background: The classification of protein sequences using string algorithms provides valuable insights for protein function prediction. Several methods, based on a variety of different patterns, have been previously proposed. Almost all string-based approaches discover patterns that are not "independent, " and therefore the associated scores overcount, a multiple number of times, the contribution of patterns that cover the same region of a sequence.

Results: In this paper we use a class of patterns, called irredundant, that is specifically designed to address this issue. Loosely speaking the set of irredundant patterns is the smallest class of "independent" patterns that can describe all common patterns in two sequences, thus they avoid overcounting. We present a novel discriminative method, called Irredundant Class, based on the statistics of irredundant patterns combined with the power of support vector machines.

Conclusion: Tests on benchmark data show that Irredundant Class outperforms most of the string algorithms previously proposed, and it achieves results as good as current state-of-the-art methods. Moreover the footprints of the most discriminative irredundant patterns can be used to guide the identification of functional regions in protein sequences.

Background

The increasing availability of biological sequences, from proteins to entire genomes, poses the need for the automatic analysis and classification of such a huge collection of data. Alignment methods and pattern discovery techniques have been used, for quite some time, to attach various problems emerging in the field of computational biology. Unfortunately most of these methods do not scale well with the lengths of the biological sequences under examination and therefore they are unpractical for genome-wide applications. To overcome this recent obstacle a number of techniques, which do not relay on alignment, have been conceived; these methods are also called alignment-free methods (see [1] for a comprehensive review). Although several alignment methods have been proposed over the years, the development of tools to classify and digest entire genome sequences is still in its infancy.

In this paper we address the classification and analysis of biological sequences without resorting to alignment. We present a method for the classification of protein sequences by means of *irredundant patterns*. Our method achieves results as good as the current state-of-the-art methods. We show also that the most discriminative irredundant patterns, which are used to distinguish different protein families, provide valuable information on the consensus pattern shared by a particular protein family.

Related works

The protein classification problem can also be treated as a string classification problem. Historically this problem has been studied in the field of text documents classification. Unfortunately most of these approaches, developed for a different kind of strings, fail when applied to biological sequences. The main reasons of this failure are the different nature of biological sequences, particularly rich of regularities known as patterns, and because of their lengths difficult to digest for a general purpose application.

Thus a number of methods have been studied for protein classification based on primary structure. The main distinction is between generative methods against discriminative methods. The former class includes methods such as protein family profiles [2], hidden Markov models (HMMs) [3-5], and PSI-BLAST [6]. These methods try to derive a model for a set of proteins and then check whether a candidate protein fits the model or not. Unlike generative methods, discriminative methods such as [7-14] are supervised approaches focused on finding which sequences (including negative examples) can describe a set of proteins despite of another set. This class makes extensive use of support vector machines (SVMs, [15]) based on features of proteins.

Recent results [8,10] suggest that the best-performing methods are discriminative string algorithms, sometimes called string kernels. The string-based learning algorithm extracts information from the sequences and computes either a feature vector for each sequence or directly a matrix with scores between pairs of sequences. Then sequences are seen as a set of labeled examples (positive if they are in the family and negative otherwise) and an SVM attempts to learn a decision boundary between the different classes. The first string kernel, called Fisher kernel [7], uses an HMM to provide the necessary means of converting proteins into fixed-length vectors. The vector summarizes how different the given sequence is from a typical member of the given protein family. In contrast in the Pairwise method [8] the feature vector, corresponding to a protein, is formed by all the *E*-values

of the Smith-Waterman score (or BLAST score) between the sequence analyzed and each of the training sequences. Other methods, like Spectrum and Mismatch [9,10], use as protein features the set of all possible substrings of amino acids of fixed length k (k-mer). If two sequences contain many of the same k-mers, their inner product under the k-Spectrum kernel will be large. Equivalently, the Mismatch kernel computes a large inner product between two sequences if these sequences contain many k-mers that differ by at most m mismatches. Then, the SVM-I-sites method [12] encodes into feature vectors the protein's three-dimensional structure, instead of using sequence similarity; conversely the eMOTIF database method [16] defines the kernel in terms of the sequence motifs that appear in a pair of sequences. These motifs were previously extracted from the eBLOCKs database using the eMOTIF-maker algorithm that derives patterns from sequence alignments [17,18].

More recently, the Local Alignment method [11] mimics the behavior of the Smith-Waterman score to build a family of valid kernels. Following the work of [19] they defined a kernel to detect all local alignment between strings by convolving simpler kernels, and hence comparing, in a simple way, strings of different lengths which share common parts. The Profile-based Mismatch method [14] uses probabilistic profiles, such as those produced by the PSI-BLAST algorithm, to define a kernel established on the position-dependent mutation neighborhoods for inexact matching of k-mers in the data (like the Mismatch method). Whereas in the Profile-based Direct method [13] kernel functions are constructed by combining sequence profiles with different approaches for determining the similarity between pairs of protein sequences. However, this kernel makes an extensive use of hyperparameters that increases the risk of overfitting when no dedicated validation data set is used. Finally, in the Word Correlation Matrices method [20] the kernel is defined by average pairwise k-mer (or word) similarity between two sequences (similarly to the Spectrum kernel). The consequent analysis of discriminative words allows also the identification of characteristic regions in biological sequences.

In general, all pattern-based methods operate, directly or indirectly, two distinct steps: first extract patterns from sequences, then use this set of patterns as features for an automatic classification tool, e.g. SVMs.

Remarkably almost all string algorithms consider patterns that are not independent, and therefore the associated scores are obtained using a set of redundant features. In this paper we want to stress the idea that if the learning process has to deal with a set of redundant

features, the resulting score is overcounting, a multiple number of times, the contribution of patterns that cover the same region of a sequence. Our conjecture is that the overcounting, due to redundant patterns, might mislead the classification. In the case of pattern-based method we can define if two patterns are independent through the resort of the notion of *irredundancy*. The class of *irredundant patterns* is specifically designed to address this issue and it was previously studied in a number of papers [21-24]. In [25] we applied a similar technique for the detection of transcription factor binding sites and we showed that the use of irredundancy is useful since redundant patterns can be filtered out without leading to overcounts.

In the next sections we will show that most approaches are using patterns, of various forms, that are redundant. We suppose that a set of irredundant patterns, and consequently a set of independent features, can improve the automatic learning and classification of sequences. Here we present a novel discriminative pairwise method for the classification of protein sequences, called *Irredundant Class*. *Irredundant Class* is based on the statistics of irredundant patterns between two sequences, combined with the discriminative power of SVMs. We selected for comparison some of the string-based learning approaches presented above, including the best-performing methods in the protein classification and remote homology detection [7-11, 20]. Our method outperforms most of the previous approaches, and it achieves results as good as current state-of-the-art methods. Moreover, given a protein family, we study the set of most discriminative irredundant patterns and their distribution. We found that the location of the functional consensus pattern for a family, as reported by PROSITE [26], can be identified by the irredundant patterns footprint.

Methods
Irredundant Class
Our method is based on the extraction of irredundant patterns that are common to two sequences. We use the idea of irredundant patterns, to avoid overcounting of patterns that cover the same region multiple times on a sequence. In this section we present the class of irredundant patterns, along with some properties.

Recently the notion of irredundancy was studied for the case of a single sequence [21,22], in this paper we extend the notion of irredundancy to the case of two sequences. In particular, our approach is substantially inspired by [22], and we refer the reader for a more comprehensive treatment on the original notion. To keep the paper self-contained, we report here the basic concepts, although most of them are only slightly adapted.

Let $s = \sigma_1\sigma_2 \ldots \sigma_n$ be a sequence of length $|s| = n$ over an alphabet Σ. A character from Σ, say σ, is called a *solid* character, while a "don't care" character '.' represents any character. A *pattern* is a string over $\Sigma \cdot (\Sigma \cup \{.\})^* \cdot \Sigma$, thus containing at least two solid characters. For instance, $p = d.dg.g.i\ldots e$ is a pattern that occurs at locations 1 and 15 in the sequence $s = dadgggdistketvdedgsgtidfee$.

To give an idea of the notion of irredundancy applied to two sequences s_1 and s_2, let us consider $s_1 = abababab$ and $s_2 = babababa$, and the two patterns $p_1 = abababa$ and $p_2 = ababa$ that are contained in both the sequences. We can note that the existence of p_1 in both s_1 and s_2 affects the existence of p_2 in all locations in which p_2 appears. By simply deleting the last *ba* from p_1 or right shifting twice p_2 along p_1 we can obtain p_2. Loosely speaking the two patterns p_1 and p_2 are not independent, or equivalently they are not irredundant. Intuitively, a pattern is irredundant if it cannot be deduced, along with its location list, by some other patterns. Consequently any redundant pattern can be derived from the set of all irredundant patterns without knowing the original sequences, thus it is not informative. We want to discard all redundant common patterns as non-informative for the learning process.

Definition 1. *(Common pattern p, Location list $\mathcal{L}_p$) Let s_1 and s_2 be two sequences on Σ. A string p on $\Sigma \cup \{.\}$ is a common pattern with location list $\mathcal{L}_p = (l_1, l_2, \ldots, l_q)$ if all of the following hold. (i) $|p| \geq 2$, (ii) $p[1], p[|p|] \in \Sigma$, (iii) p occurs both on s_1 and s_2, and (iv) there exists no location $l \notin \mathcal{L}_p$ such that p occurs at l either on s_1 or s_2 (the location list is of maximal size).*

Clearly, in the two sequences context, a common pattern occurs at least twice, one per sequence. Extending the notation of location list we can then denote by $(\mathcal{L}_p + i)$, $0 \leq i \leq |p| - 1$, all the locations in $\mathcal{L}_p$ shifted to the right by i positions.

Definition 2. *($p_1 \preccurlyeq p_2$) For characters σ_1 and σ_2 we write that $\sigma_1 \preccurlyeq \sigma_2$ if and only if σ_1 is a don't care or $\sigma_1 = \sigma_2$. Given two patterns p_1 and p_2, with $|p_1| \leq |p_2|$, $p_1 \preccurlyeq p_2$ holds if $p_1[j] \preccurlyeq p_2[j + d]$ for all $1 \leq j \leq |p_1|$, with $0 \leq d \leq |p_2| - |p_1|$.*

We also say in this case that p_1 is a *subpattern* of p_2 (or p_1 occurs in p_2), and that p_2 implies or extends p_1.

Definition 3. *(Coverage) Given two patterns p_1 and p_2 we say that the occurrence at j of p_2 is covered by p_1 (or by a subpattern of p_1) if $p_2 \preccurlyeq p_1$ and $j \in (\mathcal{L}_{p_1} + i)$ for an integer $0 \leq i \leq |p_1| - |p_2|$.*

For instance, the pattern $p_2 = ababa$ with location list $\mathcal{L}_{p_2} = (1_{s_1}, 3_{s_1}, 2_{s_2}, 4_{s_2})$ over $s_1 = abababab$ and $s_2 =$

babababa, where j_{s_h} denotes the occurrence at location j in the sequence s_h, is covered at position 3_{s_1} by $p_1 = abababa$ with $\mathcal{L}_{p_1} = (1_{s_1}, 2_{s_2})$ and $i = 2$. Note that $\mathcal{L}_{p_1} \subseteq \mathcal{L}_{p_2}$ because p_2 is a subpattern of p_1 obtained by deleting the last *ba* from p_1 (i.e., the shift integer i is 0), and that $(\mathcal{L}_{p_1} + 2) \subseteq \mathcal{L}_{p_2}$ because p_2 occurs at location 2 in p_1. Another example with don't cares is the following, $p_3 = a.a.a$ with $\mathcal{L}_{p_3} = (2_{s_3}, 4_{s_3}, 2_{s_4}, 4_{s_4})$ over $s_3 = aababababab$ and $s_4 = babacacac$ is covered at all the positions by $p_4 = aba.a.a$ with $\mathcal{L}_{p_4} = (2_{s_3}, 2_{s_4})$.

Remark 1. *Let s_1 and s_2 be two sequences, and p_1 and p_2 be two common patterns with $p_1 \preccurlyeq p_2$. Then, by definition p_2 must cover at least an occurrence of p_1 per sequence.*

Definition 4. *(Maximal common pattern) Let $\mathcal{P}_{s_1,s_2} = \{p_1, p_2, \ldots, p_k\}$ be the set of common patterns of the sequences s_1 and s_2. A pattern $p_i \in \mathcal{P}_{s_1,s_2}$ is maximal in composition if and only if there exists no $p_l \in \mathcal{P}_{s_1,s_2}$, $l \neq i$, with $p_i \preccurlyeq p_l$ and $\mathcal{L}_{p_i} = \mathcal{L}_{p_l}$. A pattern p_i maximal in composition is also maximal in length if and only if there exists no pattern $p_j \in \mathcal{P}_{s_1,s_2}$, $j \neq i$, such that $p_i \preccurlyeq p_j$ and $\left| \mathcal{L}_{p_i} \right| = \left| \mathcal{L}_{p_j} \right|$. A maximal common pattern is a pattern that is maximal both in composition and length.*

Requiring maximality in composition and length limits the number of common patterns that may be usefully extracted and accounted for in two sequences. However, the notion of maximality alone does not suffice to bound the number of such patterns. It can be shown that there are sequences that have an unusually large number of maximal common patterns without conveying extra information about the input. A maximal common pattern p is *irredundant* if p and the list $\mathcal{L}_p$ of its occurrences cannot be deduced by the union of a number of lists of subpatterns of other maximal common patterns. Conversely, we call a common pattern p *redundant* if p (and its location list $\mathcal{L}_p$) can be deduced from the other common patterns without knowing the input sequences. More formally:

Definition 5. *(Redundant/Irredundant common pattern) A maximal common pattern p, with location list $\mathcal{L}_p$, is redundant if there exist maximal common subpatterns p_i, $1 \leq i \leq k$, such that $\mathcal{L}_p = \mathcal{L}_{p_1} \cup \mathcal{L}_{p_2} \cup \ldots \cup \mathcal{L}_{p_k}$ (i.e., every occurrence of p on s_1 and s_2 is already covered by some maximal common patterns). A maximal common pattern that is not redundant is called irredundant common pattern.*

Since the redundant common patterns bring no extra information about the two sequences, the set of independent patterns is precisely the class of irredundant common patterns. For instance, we consider the sequences $s_1 = abababab$ and $s_2 = babababa$ of length 8. Then the list of all irredundant common patterns is the following. $p_1 = abababa$

with $\mathcal{L}_{p_1} = (1_{s_1}, 2_{s_2})$, $p_2 = bababab$ with $\mathcal{L}_{p_2} = (2_{s_1}, 1_{s_2})$. The other redundant maximal common patterns are. $p_3 = ababab$ with $\mathcal{L}_{p_3} = (1_{s_1}, 3_{s_1}, 2_{s_2})$, $p_4 = bababa$ with $\mathcal{L}_{p_4} = (2_{s_1}, 1_{s_2}, 3_{s_2})$, $p_5 = ababa$ with $\mathcal{L}_{p_5} = (1_{s_1}, 3_{s_1}, 2_{s_2}, 4_{s_2})$, $p_6 = babab$ with $\mathcal{L}_{p_6} = (2_{s_1}, 4_{s_1}, 1_{s_2}, 3_{s_2})$, $p_7 = abab$ with $\mathcal{L}_{p_7} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 2_{s_2}, 4_{s_2})$, $p_8 = baba$ with $\mathcal{L}_{p_8} = (2_{s_1}, 4_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2})$, $p_9 = aba$ with $\mathcal{L}_{p_9} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 2_{s_2}, 4_{s_2}, 6_{s_2})$, $p_{10} = bab$ with $\mathcal{L}_{p_{10}} = (2_{s_1}, 4_{s_1}, 6_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2})$, $p_{11} = ab$ with $\mathcal{L}_{p_{11}} = (1_{s_1}, 3_{s_1}, 5_{s_1}, 7_{s_1}, 2_{s_2}, 4_{s_2}, 6_{s_2})$, $p_{12} = ba$ with $\mathcal{L}_{p_{12}} = (2_{s_1}, 4_{s_1}, 6_{s_1}, 1_{s_2}, 3_{s_2}, 5_{s_2}, 7_{s_2})$.

It is easy to check that p_1 and p_2 cannot be deduced by other common patterns, whereas p_5 along with its location list can be simply deduced by p_1, and p_7 can be derived from the union of the occurrence lists of some subpatterns of p_1 and p_2. We want to emphasize that if the two sequences are identical there is only one irredundant common pattern, the sequence itself. This difference with the single sequence approach is due to a peculiarity of the original notion, in which the sequence itself is not considered as a valid pattern.

Definition 6. *(Consensus, Meet) The consensus of two sequences s_1 and s_2 is obtained by matching the characters corresponding to the same positions of the sequences, and inserting a don't care in case of mismatch. Deleting all leading and trailing don't cares from the consensus yields the meet of s_1 and s_2.*

For instance, the consensus of the sequences *aaaaab* and *baaaaa* is *.aaaa.*, while their meet is *aaaa*. Note that a meet is a common pattern between two sequences.

Let now $s = \sigma_1 \sigma_2 \ldots \sigma_n$ be a sequence of n characters over an alphabet Σ. We use suf_j^s to denote the suffix $\sigma_j \sigma_{j+1} \ldots \sigma_n$ of s; $s^{(j)}$ the sequence where the location j appears; and $\mathcal{L}_p^s$ the location list of p on s. Clearly $suf_1^s = s$; $s^{(1_{s_1})} = s_1$; $\mathcal{L}_p = \mathcal{L}_p^{s_1} \cup \mathcal{L}_p^{s_2}$; and $j \in \mathcal{L}_p$ if and only if $j \in \mathcal{L}_p^{s^{(j)}}$.

In the following we will briefly present the most important properties of the irredundant common patterns. Those properties are specular with respect to the single sequence approach, as in [22].

Lemma 1. *Let p be a common pattern on s_1 and s_2. Then, p is irredundant if and only if there exists $j \in \mathcal{L}_p^{s^{(j)}}$ such that the meet of $s^{(\min\{j, \ k\})}$ and $suf_{\max\{j,k\}-\min\{j,k\}+1}^{s^{(\max\{j,k\})}}$ is p for all $k \in \mathcal{L}_p^{s_h}$, where s_h is the other sequence with respect to $s^{(j)}$.*

Proof. We can prove this Lemma using the Lemmas 4 and 5 described in [22]. They prove that a pattern p' is irredundant, relatively to the approach with a single

sequence s, if and only if there exists an occurrence j' of p' in s such that the meet of s and $suf^2_{\max\{j',k'\}-\min\{j',k'\}+1}$ is p' for all k' in $\mathcal{L}^s_{p'}$. Following this work, we have that a pattern p' is irredundant if and only if at least one of its occurrences in s is not covered by other patterns. Similarly a common pattern p is irredundant, in the two sequences approach, if and only if it has at least an occurrence j in s_1, or in s_2, that is not covered by other maximal common patterns. Now assume, w.l.o.g., that $j \in s_1$, and therefore s_h is s_2. To make the statement of the Lemma valid we have to examine the case in which a common pattern p'' covers together the occurrence j of p in s_1 and another occurrence l of p in s_1 (i.e., the same sequence in which j appears). Note that p'' covers some occurrences of p if $p \preccurlyeq p''$; hence, by definition, p'' must cover also an occurrence of p in s_2 (as seen in Remark 1). This contradicts the assumptions in which we have to obtain p whenever we intersect the sequences in correspondence of two particular occurrences of p: j in s_1 and a $k \in \mathcal{L}^{s_2}_p$ in s_2.

By the previous facts and the proofs in [22], we can conclude that an irredundant common pattern must have at least an occurrence j in one of the two sequences such that the second part of the Lemma holds, and vice versa. □

To clarify the meaning of this lemma, we refer the reader to the general example that follows the algorithm.

Theorem 1. *Every irredundant common pattern over two sequences s_1 and s_2 is the meet of a sequence and a suffix of another one.*

Proof. From Lemma 1, an irredundant common pattern p (that occurs certainly in both the sequences) must appear at least in the meet of a sequence and a suffix of the other one. □

An immediate consequence of Theorem 1 is a linear bound for the cardinality of the set of irredundant common patterns. Thus

Theorem 2. *The number of irredundant common patterns over two sequences s_1 and s_2 of length, respectively, m and n is $O(m + n)$.*

Proof. From Theorem 1 we have at most $m + n - 1$ irredundant common patterns, and so $O(m + n)$. □

With its underlying convolutory structure, Lemma 1 suggests an immediate way for the extraction of irredundant common patterns from sequences and arrays, using available pattern matching with or without FFT [22].

The algorithm

The discovery of all irredundant common patterns over two sequences s_1 and s_2 is derived from Lemma 1. In this context we are interested in a proof of concept, because the aim is to verify the effectiveness of our method in the classification of protein sequences. The algorithm complexity is dominated by the most computationally intensive operation, step (b), which is the global searching of all occurrences of patterns in the sequences. We implemented this step by means of a naive string searching algorithm with don't cares that accounts for $O((m + n)^3)$ time, where m and n are the sizes of s_1 and s_2. The complete description of the algorithm follows.

Input: two sequences s_1 and s_2, where $|s_1| = m$ and $|s_2| = n$.

Output: the set of all irredundant common patterns over s_1 and s_2.

1. Compute the $m + n - 1$ meets between s_1 or s_2 and a suffix of the other sequence; then discard patterns of length < 2.
2. For each meet p:

(a) for each occurrence j found in the previous step, called exposed occurrence, increment a counter, $I_1[j]$ or $I_2[j]$, depending on the sequence in which j appears;
(b) perform a string search over s_1 and s_2 to find the number of occurrences of p in s_1 and s_2, called respectively q_1 and q_2;
(c) check if the meet p is irredundant (see Lemma 1) by finding at least an exposed occurrence j of p in $s^{(j)}$ that has a counter value equal to the number of occurrences of p in the other sequence (with respect to $s^{(j)}$). Equivalently, find if there exists an occurrence j in s_1 such that $I_1[j] = q_2$ or an occurrence j in s_2 such that $I_2[j] = q_1$.

Example. Consider the sequences $s_1 = aababbabab$ and $s_2 = babacacac$ of length 9. One of the meets computed by the algorithm is the meet of $s^{(\min\{1_{s_1}, 3_{s_2}\})}$ and $suf^{s^{(\max\{1_{s_1}, 3_{s_2}\})}}_{\max\{1_{s_1}, 3_{s_2}\}-\min\{1_{s_1}, 3_{s_2}\}+1}$ that is equivalent to compute the meet of $s^{(1_{s_1})}$ and $suf^{s^{(3_{s_2})}}_3$. Finally, it can be expressed as the meet of s_1 and $suf^{s_2}_3$, which is actually $p = a.a.a$ (see Table 1).

The only exposed occurrences of the pattern p are 2_{s_1} and 4_{s_2} (given by the meet between positions 1_{s_1} and 3_{s_2}) thus $I_1[2] = 1$ and $I_2[4] = 1$. Accordingly Table 2 shows the counters, I_1 and I_2, of p for each position of s_1 and s_2.

We note that $z_1 = \max_{1 \le j \le |s_1| = 9}\{I_1[j]\} = 1$ and that $z_2 = \max_{1 \le j \le |s_2| = 9}\{I_2[j]\} = 1$. Then step (b) performs a string search of p over s_1 and s_2. We obtain that $\mathcal{L}^{s_1}_p$ is $(2_{s_1}, 4_{s_1})$ with cardinality $q_1 = 2$, and that $\mathcal{L}^{s_2}_p$ is

Table 1: Example of meet between a sequence and a suffix of the other sequence.

position j	1	2	3	4	5	6	7	8	9
s_2	b	a	b	a	c	a	c	a	c
s_1			a	a	b	a	b	a	b
position j			1	2	3	4	5	6	7
p			a	.	a	.	a		

Meet between s_1 and with sequences s_1 = aababab and s_2 = babacacac of length 9.

Table 2: Example of counters l_1 and l_2 of a meet.

position j	1	2	3	4	5	6	7	8	9
$l_1[j]$	0	1	0	0	0	0	0	0	0

position j	1	2	3	4	5	6	7	8	9
$l_2[j]$	0	0	0	1	0	0	0	0	0

Counters l_1 and l_2 of the pattern p = a.a.a (that is the meet between s_1 and) for each position of s_1 and s_2 with sequences s_1 = aababab and s_2 = babacacac of length 9.

$(2_{s_2}, 4_{s_2})$ with cardinality $q_2 = 2$. Since $z_1 < q_2$ and $z_2 < q_1$ we can conclude by Lemma 1 that p is redundant.

Scoring irredundant patterns

First of all we have to empirically motivate our choice for irredundant patterns rather than considering all maximal patterns. Table 3 shows the number of irredundant against maximal common patterns for 10 pairs of protein sequences used in the final experiments. Results indicate that the number of irredundant patterns tends to be an order of magnitude lower than maximal patterns for not-so-short sequences, and that the latter can be exponential in the size of sequences; thus we avoid the overcount of many sequence regions. More-over maximal patterns can be prohibitive to extract for some long sequences. Whereas we have already proved that the number of irredundant patterns is at most linear in the size of sequences. Once a set $\mathcal{I}_{s_1, s_2}$ of irredundant

common patterns over two protein sequences s_1 and s_2 is acquired, we compute a scoring function based on the frequency of patterns and on the properties of amino acid compositions. Here we report the general form of the scoring function:

$$Score(s_1, s_2) = \ln \left(\sum_{p \in \mathcal{I}_{s_1, s_2}} \frac{F_p}{E[F_p]} \right),$$

where F_p is defined as the number of occurrences of the pattern p in s_1 and s_2, and $E[F_p]$ is the expected value of F_p. To compute the expected value of F_p we assume that the sequences are drawn from an i.i.d. process. The probability of a pattern p is simply the product of the probabilities of its symbols $a_i \in p$. If a_i is a solid character we compute its probability using the BLOSUM62 substitution matrix [27]; whereas the probability of a don't care is fixed to 1. Since we have assumed that the sequences come from an i.i.d. process, the expected number of occurrences of the pattern p in s_1 and s_2 is:

$$E[F_p] = (m + n - 2(|p| - 1)) \times \prod_{a_i \in p} P(a_i),$$

where m, n, and $|p|$ are, respectively, the lengths of the two sequences and the pattern p. Given a set of N sequences the input for the SVM learning process is the matrix of scores, i.e. $Score(s_i, s_j)$ with $1 \le i, j \le N$. The result of this process is a learning distance function that can be treated as an indefinite kernel. When applying SVMs with this kernel, we therefore have to resort to one of the workarounds discussed in [28]. In particular, in case of weak non-positivity of the learning function, we force the optimizer to stop after a maximum number of iterations. Despite these manageable problems, we successfully applied the matrix of scores as a kernel matrix in SVMs, and we retain for future work the task of bridging the gap in the non-positivity of the learning function.

Table 3: Irredundant vs. maximal patterns

No.	s_1	s_2	$|s_1| = m$	$|s_2| = m$	$m + n$	Maximals	Irredundants	% Of irredundants
1.	1alo	1bjt	597	760	1357	≫16697	1256	≪7.5
2.	1qax	1cxp	316	466	782	8397	682	8.1
3.	1gai	1nmt	472	227	699	7037	612	8.7
4.	1cvu	1lgr	511	368	879	9014	787	8.7
5.	1gpe	1yrg	392	343	735	6853	653	9.5
6.	1qqj	3pcc	415	236	651	5090	566	11.1
7.	1bxk	1ofg	352	220	572	3549	489	13.8
8.	1ebf	2nac	169	188	357	1126	277	24.6
9.	1a03	1mho	90	88	178	257	108	42.0
10.	1gpt	1ayj	47	50	97	64	45	70.3

Number of irredundant and maximal common patterns over 10 pairs of protein sequences taken from experiments in Table 4. Rows are sorted according to the percentage of irredundants over the total number of maximal patterns.

Results and discussion

We assessed the effectiveness of our method on the classification of protein families into superfamilies. Recent works were compared against the dataset presented in [8]. This dataset is based on the Structural Classification Of Proteins (SCOP, [29]) version 1.53, removing similar sequences with an *E*-value threshold of 10^{-25}. The data consist of 4352 sequences grouped into 54 families and 23 superfamilies selected by Liao and Noble. For each family, proteins within the family are considered positive test examples, and proteins within the superfamily but outside the family are considered positive training examples; negative examples are chosen outside the fold, and were randomly split into training and test sets in the same ratio as the positive examples. Therefore this assessment consists of 54 experiments, each corresponding to a target family having at least 10 positive training examples and 5 positive test examples, and no sequence homologies known a priori. In these experiments there is usually an unbalanced number of negative sequences with respect to the number of positive sequences, as illustrated in Table 4. In short, the task consists on classifying target families of sequences into superfamilies of SCOP using an SVM.

For the comparison against state-of-the-art methods we use as metric the receiver operating characteristic (ROC) score. Given a ranking of the test sequences scores in output from the SVM, ROC score is the normalized area under the curve that plots the number of positive examples correctly predicted (true positives) as a function of the number of negative examples found to be positive (false positives) for each different possible classification threshold of a specific experiment. Our results were obtained using the Gist SVM version 2.3; realized by Noble and Pavlidis [30], it implements the soft margin optimization algorithm described in [31].

For every target family we have to extract the irredundant common patterns, compute the score matrix, train the SVM, and then classify the test sequences. The requirements in terms of time can vary dramatically. In general, they are related to the number of sequences and to the protein family. The most time consuming step is the training of the SVM with times that range between few minutes up to 50 minutes, in some difficult cases. Experimental data and results of other methods were taken from [11,20]. Table 5 shows the scores of ROC, ROC50, and Median rate of false positives (mRFP)

Table 4: Experiments of Liao and Noble

No.	Target family	Training Pos.	Training Neg.	Test Pos.	Test Neg.	No.	Target family	Training Pos.	Training Neg.	Test Pos.	Test Neg.
0	7.3.5.2	12	2330	9	1746	27	7.3.10.1	11	423	95	3653
1	2.56.1.2	11	2509	8	1824	28	3.32.1.11	46	3880	5	421
2	3.1.8.1	19	3002	8	1263	29	3.32.1.13	43	3627	8	674
3	3.1.8.3	17	2686	10	1579	30	7.3.6.1	33	3203	9	873
4	1.27.1.1	12	2890	6	1444	31	7.3.6.2	16	1553	26	2523
5	1.27.1.2	10	2408	8	1926	32	7.3.6.4	37	3591	5	485
6	3.42.1.1	29	3208	10	1105	33	2.38.4.1	30	3682	5	613
7	1.45.1.2	33	3650	6	663	34	2.1.1.1	90	3102	31	1068
8	1.4.1.1	26	2256	23	1994	35	2.1.1.2	99	3412	22	758
9	2.9.1.2	17	2370	14	1951	36	3.32.1.1	42	3542	9	759
10	1.4.1.2	41	3557	8	693	37	2.38.4.3	24	2946	11	1349
11	2.9.1.3	26	3625	5	696	38	2.1.1.3	113	3895	8	275
12	1.4.1.3	40	3470	9	780	39	2.1.1.4	88	3033	33	1137
13	2.44.1.2	11	307	140	3894	40	2.38.4.5	26	3191	9	1104
14	2.9.1.4	21	2928	10	1393	41	2.1.1.5	94	3240	27	930
15	3.42.1.5	26	2876	13	1437	42	7.39.1.2	20	3204	7	1121
16	3.2.1.2	37	3002	16	1297	43	2.52.1.2	12	3060	5	1275
17	3.42.1.8	34	3761	5	552	44	7.39.1.3	13	2083	14	2242
18	3.2.1.3	44	3569	9	730	45	1.36.1.2	29	3477	7	839
19	3.2.1.4	46	3732	7	567	46	3.32.1.8	40	3374	11	927
20	3.2.1.5	46	3732	7	567	47	1.36.1.5	10	1199	26	3117
21	3.2.1.6	48	3894	5	405	48	7.41.5.1	10	2241	9	2016
22	2.28.1.1	18	1246	44	3044	49	7.41.5.2	10	2241	9	2016
23	3.3.1.2	22	3280	7	1043	50	1.41.1.2	36	3692	6	615
24	3.2.1.7	48	3894	5	405	51	2.5.1.1	13	2345	11	1983
25	2.28.1.3	56	3875	6	415	52	2.5.1.3	14	2525	10	1803
26	3.3.1.5	13	1938	16	2385	53	1.41.1.5	17	1744	25	2563

Experiments presented in [8] and associated to 54 protein families of SCOP version 1.53, here ordered by progressive number. For each target family ID is detailed the number of positive and negative sequences of training and test.

Table 5: Comparison of results against state-of-the-art methods

Learning algorithm	Mean ROC	Mean ROC50	Mean mRFP
Irredundant Class	**0.929**	0.524	0.0554
Local Alignment ("ekm," β = 0.5)	**0.929**	0.600	**0.0515**
Local Alignment ("eig," β = 0.5)	0.925	**0.649**	0.0541
Word Correlation Matrices (k = 6)	0.904	0.447	0.0778
Pairwise	0.896	0.464	0.0837
Mismatch (k = 5, m = 1)	0.872	0.400	0.0837
Spectrum (k = 3)	0.824	0.294	0.1535
Fisher	0.773	0.250	0.2040

The comparison is based on mean scores of ROC, ROC50 (ROC curve up to the first 50 false positives), and Median rate of false positives (mRFP) for the Irredundant Class and state-of-the-art methods. In bold are reported the best results for each score.

averaged out over all the experiments for Irredundant Class and state-of-the-art methods. We report in bold the best results for each score. These scores indicate that our method outperforms most learning algorithms in literature, and it is comparable to state-of-the-art Local Alignment kernels.

For a more detailed view, the ROC scores distribution is illustrated in Figure 1(a). The green line (triangles), corresponding to the Local Alignment version "eig" (that was suggested by the authors for a lower computational expensive with respect to the version "ekm"), seems to perform slightly better than the Irredundant Class in blue (squares), but the minimum ROC score of the Local Alignment is much smaller. Figure 1(b) reports the distribution of ROC scores per family. The Irredundant Class shows a more robust behavior for all experiments

with respect to the other learning algorithms. In particular, the smallest ROC score of our method was obtained for the 15-th experiment with the value 0.614, while all other methods get their lowest peaks for the 13-th experiment with values much smaller, e.g. 0.22 for Local Alignment. Family-by-family details can be caught from Figure 2 for the comparison of the Irredundant Class against, respectively, (a) Mismatch and (b) Local Alignment ",eig"). The former is one of the most efficient kernels in literature, while the latter is known to achieve high performance. Again, we can observe that Irredundant Class achieves better performance than Mismatch and similar results when compared with Local Alignment.

Analysis of irredundant patterns footprint
Although the classification of protein sequences, through an SVM, does not provide an alignment per se, one can use the footprint of irredundant patterns to detect candidate functional sites in protein sequences. Here we are not interested in aligning a set of sequences, but we want to analyze the distribution of the most discriminative irredundant common patterns.

We recall that the result of the SVM learning process, for a target protein family, is a set of weights $\alpha = (\alpha_1, ..., \alpha_N)$ associated to the N training sequences of its superfamily. We want to study the distribution of irredundant common patterns in the test sequences using for each pattern p a weight that is proportional to its score $\frac{F_p}{E[F_p]}$ and to the weight α_i of the corresponding training
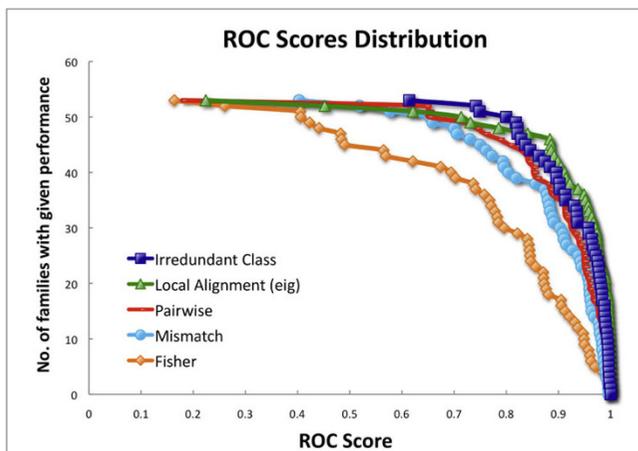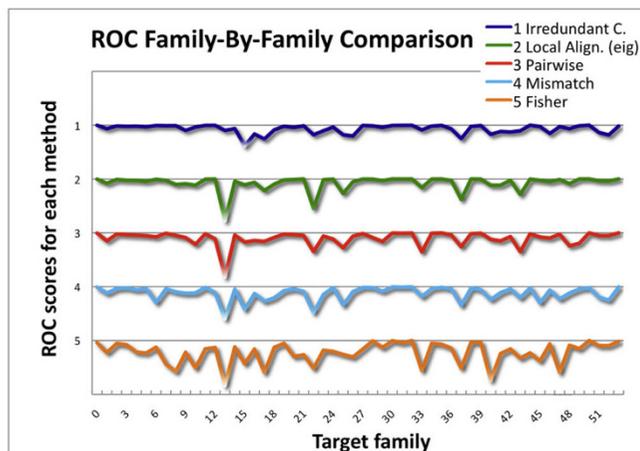
Figure 1
ROC scores distributions. (a) ROC scores distribution for the Irredundant Class and state-of-the-art methods. (b) ROC scores across families.
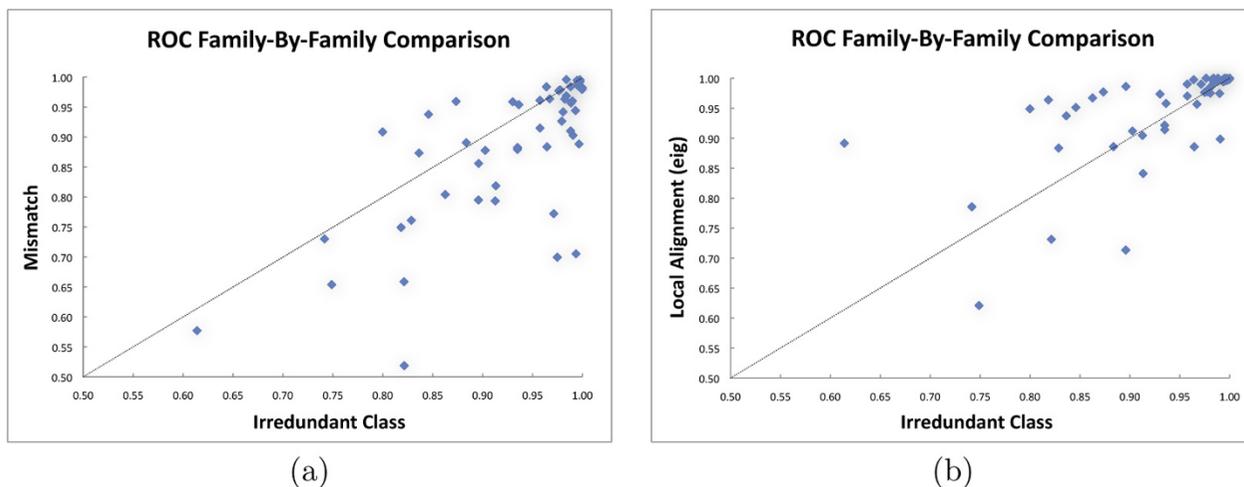
Figure 2
ROC scores family-by-family comparisons. (a) Family-by-family ROC scores comparison of the Irredundant Class against Mismatch. (b) Family-by-family ROC scores comparison of the Irredundant Class against Local Alignment version "eig."

sequence that generated p, with $1 \leq i \leq N$. Consider a test sequence s_{test} and the set of training sequences $s_1, ..., s_N$; each pair (s_{test}, s_i) generates a set of irredundant patterns $\mathcal{I}_{test, i}$. For each pattern p in $\mathcal{I}_{test, i}$ we compute its score as the product $\ln \left(\frac{F_p}{E[F_p]} \right) \times \alpha_i$ and we associate this score to the positions of s_{test} covered by solid characters of p. We repeat the same process for all patterns in $\mathcal{I}_{test, i}$, for each $1 \leq i \leq N$; and for each location we sum the contributions of all patterns that cover that location. We obtain a histogram of the footprint of the irredundant patterns for the test sequence s_{test}. The interpretation of this histogram is that picks should correspond to conserved regions, thus to candidate functional sites.

We picked three families at random from the dataset used. For every protein family we use as target functional sites the PROSITE [26] manually found consensus patterns. To better highlight the distribution of footprints we build, for each family, a multiple alignment of the test sequences and plot all histograms over this alignment. Figure 3 shows the resulting histogram for the protein family S100. A red bar shows the location of the functional pattern reported by PROSITE, also shown in the picture. For this family we can see that a clear signal is present and that picks correspond quite well with conserved amino acids in the functional site. Similar considerations apply also for the families in Figure 4. In Figure 4(a) we observe picks mostly in correspondence of Cysteines, whereas in Figure 4(b) the pattern reported by PROSITE results in two functional sites that share comparable high scores.

Note that these results are obtained comparing sequences from a protein family and its superfamily, thus the chances to find the actual signal are weak as opposed to standard alignment methods, which consider only the protein family. Nevertheless the profile of the family functional site can be computed as a post process of our analysis by a multiple alignment of candidate regions.

This analysis does not yield to an alignment of sequences, but it is a way to interpret the distribution of irredundant patterns. In summary the most discriminative patterns contain information about the functional site of a protein family, but this information is not explicitly available by simple inspection.

Conclusion

In this paper we studied how the notion of irredundant patterns can solve an issue that is rising in the field of string-based learning algorithms. Almost all string-based approaches consider patterns that are not independent, and therefore the associated scores overcount, a multiple number of times, the contribution of patterns that cover the same region of a sequence, called redundant patterns. Thus we use the class of irredundant common patterns over two sequences, which is specifically designed to address this issue. We design a novel method, called Irredundant Class, that is a discriminative pairwise learning algorithm based on support vector machines. Results on benchmark data show that the Irredundant Class outperforms most of the approaches
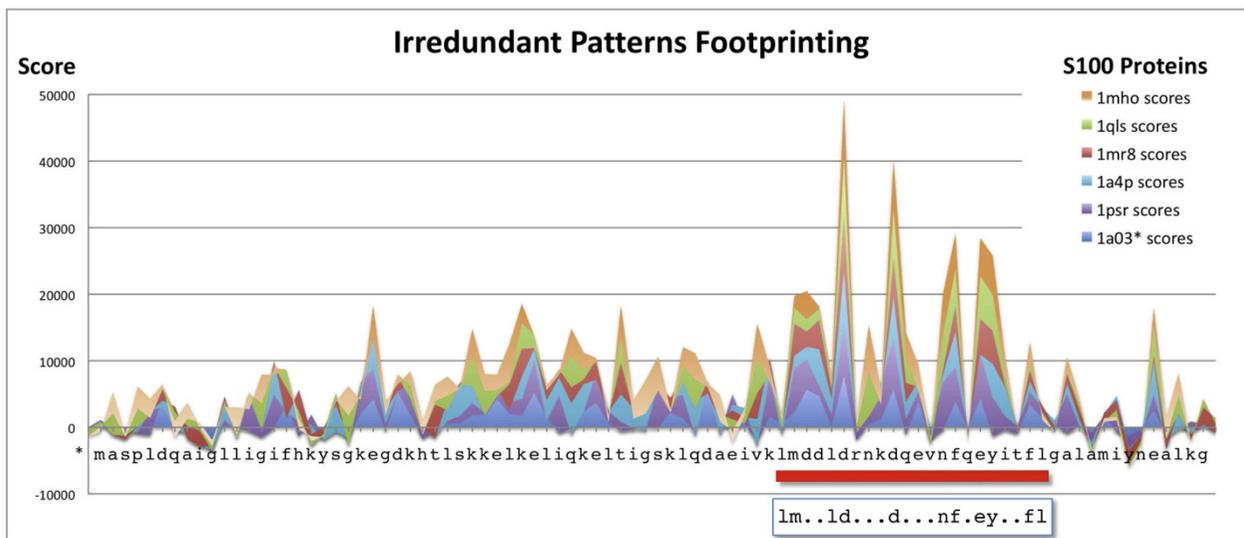
Figure 3
Irredundant patterns footprint for protein family 50. Histogram of the irredundant patterns footprint for S100 proteins (family no. 50 of Table 4).
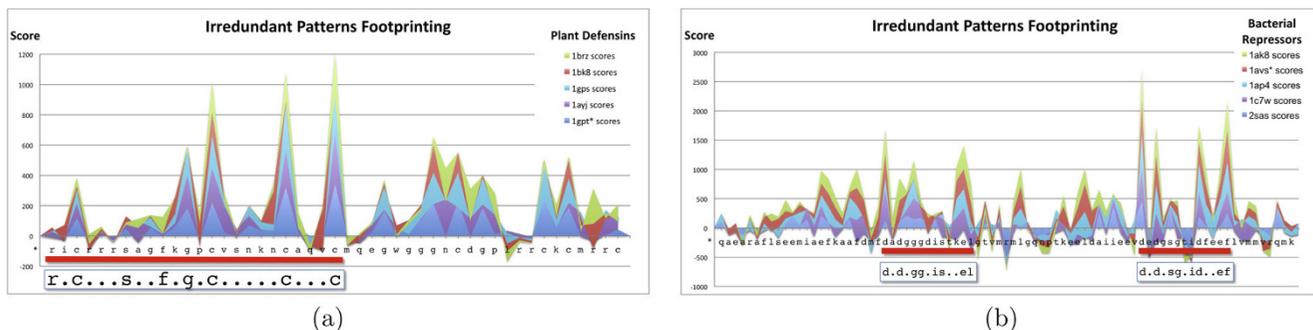

Figure 4
Irredundant patterns footprint for protein families 32 and 53. Histogram of the irredundant patterns footprint for: (a) plant defensis (family no. 32 of Table 4) and (b) bacterial repressors (family no. 53).

previously proposed, and it achieves results as good as current state-of-the-art methods. Moreover the footprint of the most discriminative irredundant patterns can be interpreted, in a biological fashion, as a guide for the identification of characteristic regions in protein sequences. Finally, the Irredundant Class approach is not limited to protein sequences, but it can also be applied to DNA or RNA sequences. The investigation of these areas will be part of future work.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors have equally contributed to, seen and approved the manuscript.

References

1. Vinga S and Almeida J: **Alignment-free sequence comparison - a review.** *Bioinformatics* 2003, **19(4):**513–523.
2. Gribskov M, McLachlan AD and Eisenberg D: **Profile analysis: detection of distantly related proteins.** *Proc Natl Acad Sci U S A* 1987, **84(13):**4355–4358.
3. Krogh A, Brown M, Mian IS, Sjöolander K and Haussler D: **Hidden Markov models in computational biology: applications to protein modeling.** *Journal of Molecular Biology* 1994, **235:** 1501–1531.
4. Baldi P, Chauvin Y, Hunkapiller T and McClure M: **Hidden Markov models of biological primary sequence information.** *Proc Natl Acad Sci U S A* 1994, **91:**1059–1063.
5. Karplus K, Barrett C and Hughey R: **Hidden Markov models for detecting remote protein homologies.** *Bioinformatics* 1998, **14:**846–856.
6. Altschul SF, Madden TL, Schffer AA, Schaffer RA, Zhang J, Zhang Z, Miller W and Lipman DJ: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.** *Nucleic Acids Research* 1997, **25:**3389–3402.
7. Jaakkola T, Diekhans M and Haussler D: **Using the Fisher kernel method to detect remote protein homologies.** *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB 1999)* AAAI: Thomas Lengauer, Richard Schneider, Peer Bork, Douglas L Brutlag, Janice I Glasgow, Hans-Werner Mewes and Ralf Zimmer 1999, 149–158.
8. Liao L and Noble WS: **Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships.** *Journal of Computational Biology* 2003, **10:**857–868.
9. Leslie CS, Eskin E and Noble WS: **The spectrum kernel: a string kernel for SVM protein classification.** *Proceedings of the 7th Pacific Symposium on Biocomputing (PSB 2002)* Russ B Altman, A Keith Dunker, Lawrence Hunter and Teri E Klein 2002, 564–575.
10. Leslie CS, Eskin E, Cohen A, Weston J and Noble WS: **Mismatch string kernels for discriminative protein classification.** *Bioinformatics* 2004, **20(4):**467–476.
11. Saigo H, Vert JP, Ueda N and Akutsu T: **Protein homology detection using string alignment kernels.** *Bioinformatics* 2004, **20(11):**1682–1689.
12. Hou Y, Hsu W, Lee ML and Bystroff C: **Efficient remote homology detection using local structure.** *Bioinformatics* 2003, **19:**2294–2301.
13. Rangwala H and Karypis G: **Profile-based direct kernels for remote homology detection and fold recognition.** *Bioinformatics* 2005, **21(23):**4239–4247.
14. Kuang R, Ie E, Wang K, Wang K, Siddiqi M, Freund Y and Leslie C: **Profile-based string kernels for remote homology detection and motif extraction.** *Journal of Bioinformatics and Computational Biology* 2005, **3(3):**527–550.
15. Vapnik VN: *Statistical Learning Theory* Wiley-Interscience; 1998.
16. Ben-Hur A and Brutlag DL: **Remote homology detection: a motif based approach.** *Bioinformatics* 2003, **19(Suppl 1):**i26–i33.
17. Nevill-Manning CG, Wu TD and Brutlag DL: **Highly specific protein sequence motifs for genome analysis.** *Proc Natl Acad Sci U S A* 1998, **95:**5865–5871.
18. Huang JY and Brutlag DL: **The eMOTIF database.** *Nucleic Acids Research* 2001, **29:**202–204.
19. Haussler D: **Convolution kernels on discrete structures.** *Technical Report* University of California at Santa Cruz, Santa Cruz, CA, USA; 1999UCS-CRL-99-10: .
20. Lingner T and Meinicke P: **Word correlation matrices for protein sequence analysis and remote homology detection.** *BMC Bioinformatics* 2008, **9:**259.
21. Apostolico A and Tagliacollo C: **Incremental discovery of the irredundant motif bases for all suffixes of a string in $O(n^2 \log n)$ time.** *Theoretical Computer Science* 2008, **408(2-3):**106–115.
22. Apostolico A and Tagliacollo C: **Optimal offline extraction of irredundant motif bases.** *Proceedings of the 13th Annual International Computing and Cobinatorics Conference (COCOON 2007)* Springer: Lin G 2007, **4598:**360–371.
23. Apostolico A and Parida L: **Incremental paradigms of motif discovery.** *Journal of Computational Biology* 2004, **11:**15–25.
24. Pisanti N, Crochemore M, Grossi R and Sagot MF: **Bases of motifs for generating repeated patterns with wild cards.** *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2005, **2:**40–50.
25. Comin M and Parida L: **Detection of subtle variations as consensus motifs.** *Theoretical Computer Science* 2008, **395(2-3):** 158–170.
26. Hulo N, Bairoch A, Bulliard V, Cerutti L, Cuche B, de Castro E, Lachaize C, Langendijk-Genevaux P and Sigrist C: **The 20 years of PROSITE.** *Nucleic Acids Research* 2008, **36 Database:** D245–9.
27. Henikoff S and Henikoff JG: **Amino acid substitution matrices from protein blocks.** *Proc Natl Acad Sci U S A* 1992, **89(22):** 10915–10919.
28. Eichhorn J: **Applications of kernel machines to structured data.** *PhD thesis* Technische Universität Berlin; 2007.
29. Murzin AG, Brenner SE, Hubbard T and Chothia C: **SCOP: a structural classification of proteins database for the investigation of sequences and structures.** *Journal of Molecular Biology* 1995, **247:**536–540.
30. Noble WS and Pavlidis P: **Gist support vector machine and kernel principal components analysis software toolkit.** Columbia University; 2002.
31. Jaakkola T, Diekhans M and Haussler D: **A discriminative framework for detecting remote protein homologies.** *Journal of Computational Biology* 2000, **7:**95–114.