Mining Biological Repetitive Sequences Using Support Vector Machines and Fuzzy SVM

Torabi Dashti, Hesam; Masoudi-Nejad, Ali*[+]

Laboratory of Systems Biology and Bioinformatics (LBB), Institute of Biochemistry & Biophysics (IBB), University of Tehran, Tehran, I.R. IRAN

ABSTRACT: *Structural repetitive subsequences are most important portion of biological sequences, which play crucial roles on corresponding sequence's fold and functionality. Biggest class of the repetitive subsequences is "Transposable Elements" which has its own sub-classes upon contexts' structures. Many researches have been performed to criticality determine the structure and function of repetitive subsequences. The sequencing noises and the sequences' substitutions probability are obstacles of these researches. Some statistical and approximation algorithms have introduced to tackle these obstacles. By introducing conspicuous statistical machine learning methods upon Support Vector Machines, machine learning approaches act as potent methods to solve the pattern-finding problem. Support vector machines methods are time efficient approaches, which based on their parameters can be precise and accurate. In this Review, mathematical definition of structural repetitive subsequences are introduced, thereafter proposed algorithm to tackle simple pattern finding problem, which can be applicable on structural patterns are reviewed. Theoretical aspects of Support Vector Machines on computational biology platform are considered. Finally, novel evolutionary Fuzzy SVM will be introduced, which is applicable on wide range of bioinformatics problems especially the problem of structural repetitive subsequences.*

KEY WORDS: *Fuzzy support vector machines, Pattern discovery problem, Transposable elements.*

INTRODUCTION

Nowadays, by improving sequencing tools, studying and analyzing genomic sequences becomes a major field of study that motivates scientists to propose efficient and precise approaches. Since *Britten et al.* discovered structure of eukaryotes' genome and showed that significant subsequences are dispersed through eukaryotes' gnome, the first step of studying eukaryote genome became extracting significant subsequences [1,2]. On the other hand, in spite of improvement in sequencing methods, the uncertainty which exists on sequenced genomes necessitates the use of IUPAC letters in sequence content.

Experimental results showed that there are some significant subsequences with similar contents.

* *To whom correspondence should be addressed.*
+ *E-mail: amasoudin@ibb.ut.ac.ir*
1021-9986/10/4/1 17/$/3.70

Consequently this kind of significant subsequences is called "repeated subsequences". Repeated subsequences are divided into different classes, where each class's members are similar to each other. Upon the members' similarity, each class represents a specific family of repeated subsequences.

Moreover, discovery of homolog families among different eukaryote species led to establishment of some repeat libraries that include discovered repeated subsequences' context and families. These homolog families imply that repeated subsequences are not only most conserved portions of eukaryotes genome but also play crucial roles in species lives. Hence in this context discovering repeated subsequences and also resolving their family are two essential problems towards solving sequence analyzing problem.

There is a specific style of repeated subsequences that covers huge numbers of repeated subsequences. This style's members which are called "structural repeated subsequences" are following different repetition rules and have different roles on genome structure or cells' living cycle.

A lot of approaches have been introduced for repetitive subsequences discovery and more issues have published on classifying the discovered subsequences. Some library-based approaches try to discover repeated subsequences which their family had been discovered before. On the other hand, lots of *ab-initio* approaches have been introduced that discover a genome's repetitive subsequences without using current repeat-libraries [3-5].

In this Review, we are going to overview the pattern finding problem and thereafter one of potential pattern discovery approaches, called SVM will be introduced. Finally, a novel idea on structural repeat discovery via fuzzy SVM will be represented.

BIOLOGICAL ASPECTS OF PATTERN FINDING PROBLEM

This section is focused on describing pattern finding problems in biological systems and thereafter a brief review of definitions of structural repetitive sequences will be presented. These definitions would be enough to embark computational aspects of the problem. Repetitive patterns which are discussed in this section are usually functionally or structurally important elements in proteins or DNA sequences, in that they cover 50 percents of human genome [6]. These sequences occur more frequently than expected and most biologists, to present a reason, hold an opinion that these sequences are best conserved in evolution because they play important roles on genome's structure and functionality. So pattern discovery is one of the fundamental problems in bioinformatics. Introduced approaches to solve this problem can be used in other bioinformatics problems such as multiple sequence alignment, protein structure and functionality prediction, characterization of protein families, promoter signal detection, and etc [7, 8].

Biological Motivations for Pattern Discovery

Nucleotide and protein sequences contain patterns or motifs that have been preserved through evolution because they are important to the structure or function of the molecule. In case of proteins, these conserved sequences may be involved in the binding of the protein to its substrate or to another protein, may comprise the active site of an enzyme or may determine the three dimensional structure of the protein. Some of these repeated sequences are flanked between two coding regions; on the other hand, there is a biological fact that in general, sequences outside of coding regions tend to be less conserved through organisms, except where they play important function. This rigorous fact proves the importance of repetitive sequences when it is discovered previously that the flanked repeated subsequences are involved in the regulation of gene expression [9].

According to above description on conserved sequences inside of DNA or protein, discovering these sequences can lead to elucidate evolutionary relationships among sequences.

Pattern discovery in proteins

Several applications are available for identifying motifs in proteins that have evolved by divergent evolution. These patterns which appear in families with common ancestors can be used in methods which organize all of the proteins into families based on the presence of common signature sequences [10,11]. In order to increase the certainty that a protein has been assigned to a correct family, members of protein

families are often characterized by more than one motif (on average each family has 3-4 conserved regions) Since hierarchical trees of protein clusters often reveal functional and evolutionary relationships among proteins, proteins can be assigned to a family based on sequence homology as determined primarily by alignment. However, there are two problems with this assumption:

- Significant sequence similarities are not always indicative of close evolutionary relations,
- Despite partial sequence homology, proteins can have structural and mechanistic similarities, and even common ancestry is not apparent through alignment.

Therefore, considering structural information could be convenient when attempting to classify proteins that are highly divergent in homology, yet functionally equivalent. Such this approach has been used to identify motifs in proteins that may be related through convergent evolution. Leucine zipper sequences, involved in protein demyelization, appear in diverse families that lack a common ancestor and thus may be an example of a convergent motif.

Another analysis on identification of patterns that have been conserved through evolution can lead to the discovery of association of these sequences with protein function or structure. A first step towards function prediction is to look for sequence features that are common to groups of proteins with a specified activity but are absent from proteins without the activity. *Savoie et al.* [12] used such an approach to develop a recognition rule for sequences that determine whether a peptide will activate a T-cell response. These conserved sequences are essentially antigenic determinants that elicit an immune response and can be used to develop vaccines. The premise of all attempts to assign a function to unknown proteins by pattern recognition is that highly conserved sequences have been preserved through evolution because they are important to the function or structure of the protein. While intuitively this seems a valid assumption, it is possible that some conserved sequences simply correspond to regions with a lower rate of mutation.

Albeit the above analyzes have been performed, functional motifs may not be apparent in the protein primary sequence when they consist of single conserved amino acid residues separated by long, variable regions, these conserved residues may come together to form a functional group when the protein is folded into its three dimensional structure [13]. On the other hand, patterns of conserved sequences can often highlight elements that are responsible for structural similarity between proteins and can be used to predict the three dimensional structure of a protein.

Because some amino acids share similar characteristics such as size, charge or hydrophobicity, substitutions are often permitted in protein motifs even where residues are important to structure or function. *Nevill-Manning et al.* [14] describe a method for discovering conserved motifs that characterize a protein family but are somewhat flexible in the amino acids allowed in particular positions within the motif. Groups of amino acids occurring at each position in a motif with significant frequency were identified and used to characterize subsets of motifs that are biologically relevant. While a motif should be sensitive enough to allow identification of new family members with a minimum of false negatives, there may be a tradeoff in terms of specificity; lower specificity leads to the identification of false positive sequences.

Once biological dictionaries of protein sequence patterns are constructed (protein motif databases), they can be used to predict the function of newly discovered or unknown proteins, or to screen genomic databases for other proteins with similar function.

Pattern discovery in non-coding regions

Similar to patterns in proteins, motifs in DNA sequences can be used to determine the function of nucleotide sequences on a global level. The first application of motif finding in DNA is in finding all promoters in a genome. Finding all promoters in large genomic sequences necessitates the identification of features that are common to all promoters but are not present in non-promoter sequences. This is a difficult problem, especially in eukaryotic organisms which do not have a single core promoter and are usually associated with multiple regulatory factors. Some of the available approaches to solve the problem include:

- The identification of global signals that interact with RNA polymerase and general transcription factors (e.g., TATA and CAAT boxes, CpG islands),
- The detection of upstream regions with a high

density of transcription factor binding sites (although these are often not clustered),

• The identification of sequence characteristics that influence DNA three dimensional structure (regions downstream of the TATA box tend to be highly bendable while regions upstream have low bend ability) [15].

Another application of motif finding is to determine specific functions such as regions involved in tissue specific regulation of gene expression which involves identifying specific regulatory sequences in promoter regions. Prediction of regulatory protein binding sites can help to infer the function of a gene when homologous genes of known function are not available.

When the transcription factor binding motifs are unknown, they can be found by searching for common elements in the upstream regions of genes that are known to be co-regulated (such genes are known as regulons) [16, 17]. A comparative approach can be used to predict regulatory sequences in different genomes, however, the cognate regulatory factor must be known to be conserved [18].

Finding motifs in RNA sequences is also of researcher's interest; however, RNA molecules such as tRNA, rRNA and catalytic RNA are usually more conserved in structure than in sequence. The properties of an RNA molecule are often determined by its structure which can take various forms as a consequence of intra molecular base pairing within the single stranded molecule, for example, pseudo knots, hairpin loops, bulges, etc. Rather than looking for motifs, an RNA sequence is searched for regions that could potentially contain base pair to form secondary structure. Obviously, in order to do so, distance constraints would have to be applied and a minimum number of base pairs would be required [19].

Structural repetitive sequences

Tandem Repeats

A particularly interesting problem in pattern finding involves the detection of tandem repeats, which are two or more contiguous approximate copies of a pattern of nucleotides. Tandem duplication occurs as a result of mutational events in which an original segment of DNA, the pattern, is converted into a sequence of individual copies. With the progression of time, the individual copies within a tandem repeat may undergo other "uncoordinated" mutations which render the once-identical copies in the original pattern as only approximate variations of each other.

The prevalence of tandem repeats is surprisingly high in genomic sequences. [20] notes that Tandem repeats are presumed to occur frequently in genomic sequences, comprising perhaps 10% or more of the human genome, But, accurate characterization of the properties of tandem repeats has been limited by the inability to easily detect them". As *Benson* explains, the detection of tandem repeats has come to assume an increasing importance in genomic research, for both positive and negative reasons. On the negative side, the appearance of specific kinds of tandem repeats has been linked to a number of different diseases. On the positive side, however, it appears that tandem repeats may play a role in gene regulation (interacting with transcription factors, altering the structure of the chromatin, or acting as protein binding sites [14] and in the development of immune system cells.

Transposable elements

After considering biological aspects of repetitive sequences, in this section is focused on considering two styles of repetitive sequences which cover structural repetitive sequences. As mentioned structural repeated subsequences are subclass of repetitive sequences, therefore computational aspects of this subclass is not illustrated separately. Transposable elements cover big portion of eukaryotes genome and play crucial role on genes functionality. Transposable elements can transpose through genome and in each transition make duplicate of itself. Transposable elements based on their transposition's intermediate are divided to two classes;

• RNA is intermediate (Class I)

• DNA performs, transposable element belong to class II.

Transposable elements include four features; (1) coding regions, (2) repeated/inverted subsequences, (3) non-coding regions, (4) some open reading frames, where each transposable element composed of some of above features. Hence, transposable elements can be divided to more subclasses based on used features and also arrangement of these features.

After a long time that there was not a clear classification for transposable elements subclasses,

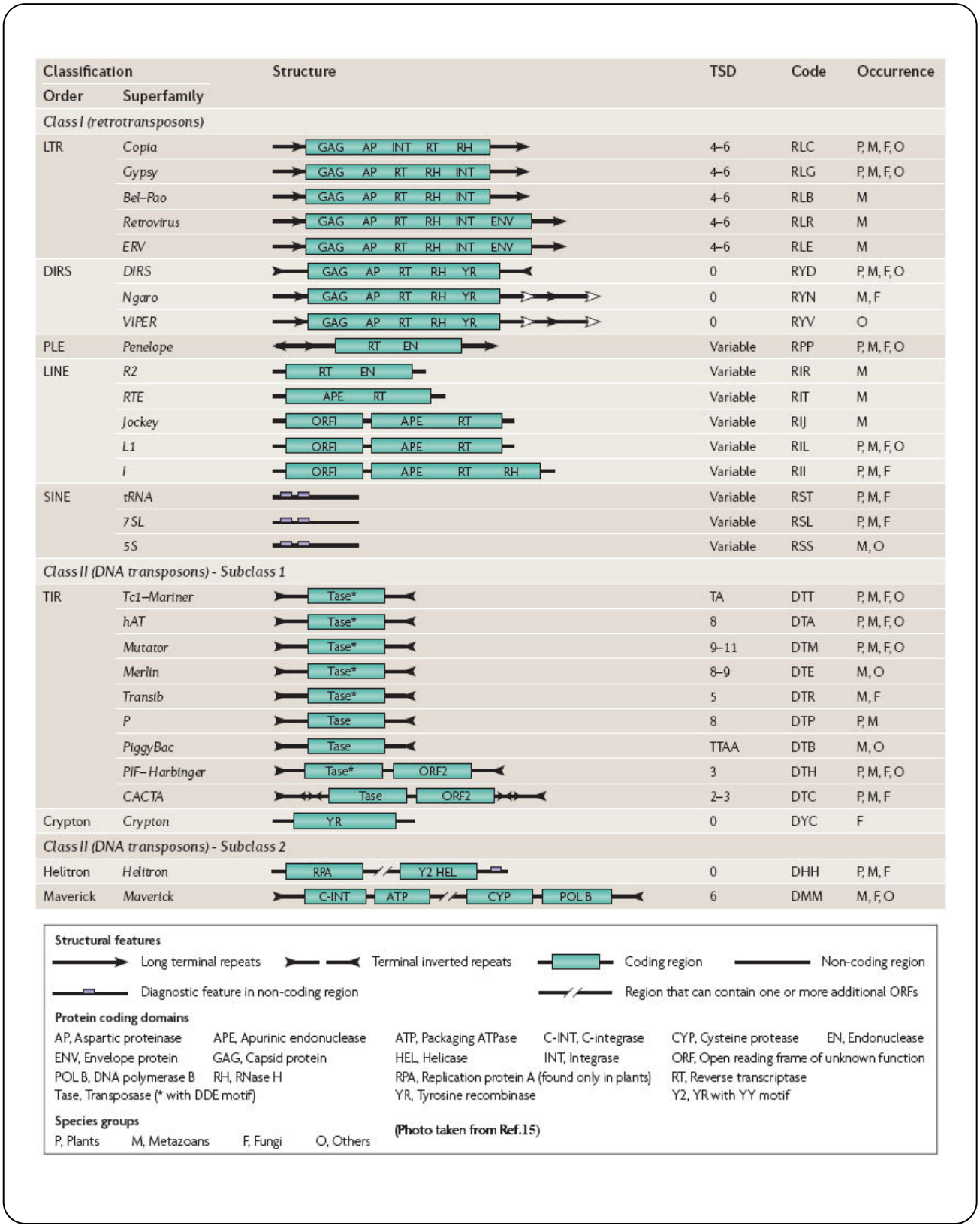

Fig. 1: Hierarchical identification of transposable element (Photo taken from [21]).

recently [21] proposed so clear hierarchical catalogue of transposable elements structures and identified each structure's class, its subclasses, and its major classes (Fig. 1). Here, we just borrow hierarchical classes structure and for comprehensive information refer to [21].

COMPUTATIONAL ASPECTS

Computer science questions on repetitive pattern finding

In this section, we will discuss how to translate mentioned biological questions about pattern discovery to formal computer science problems.

Classification problems

One class of problems is classification problems. These problems occur for example in specification of protein families. One of the goals of finding common motifs in protein families is to use these motifs as classifiers: given an unknown protein, we can classify it as a member or nonmember of a family, based on the fact that it contains the motifs characteristic for the family. In this case, we may formulate question as a machine learning problem: given a set of sequences belonging to the family (positive examples) and a set of sequences not belonging to the family (negative examples) one may wish to find a function f which for each protein decides whether it belongs to the family or not. In context of motif discovery we are mostly interested in such classes of functions f that involve matching some discovered patterns against the unknown sequence. Note that negative examples are simply other known proteins taken from protein databases such as SWISS-PROT. Quite often people start only from positive examples and negative examples are used for evaluation of their classifiers.

Finding significant patterns

Motif discovery is not always formulated as a classification problem. For example if we want to find a regulatory element, we might have a set of regions that contain motif subsequences. However it does not mean that this element cannot occur in other places in genome or that all of these sequences must contain common regulatory elements. Also in context of protein family motifs we are interested in finding conserved regions that may indicate structurally or functionally important elements, regardless of whether they have enough specificity to distinguish between this family and other families. Therefore, it is more complicated to formulate the question precisely. Usually people define a class of patterns they want to find and they are interested in discovering the highest scoring pattern from this class that has enough support. Various pattern finding approaches differ in the way of defining a support and a score of a pattern. Support of a pattern usually means the number of sequences in which the pattern occurs. We can require that pattern should occur in all sequences or there are a minimum number of occurrences specified by user. In some cases the number of occurrences is not specified but it is part of a scoring function (longer pattern with fewer occurrences can be sometimes more interesting than shorter pattern with more occurrences). The situation is even more complicated in the case of probabilistic patterns, such as hidden *Markov* models. Deterministic patterns either match sequence or do not (zero or one), whereas probabilistic models give a probability between 0 and 1. Therefore, there are different degrees of "\matching". It is necessary to set some threshold on what should be considered a match or to include these matching probabilities in the score of the pattern. Methods for scoring patterns also differ from paper to paper. Score can describe only the pattern itself (e.g. its length, degree of ambiguity etc.) or it can be based on the occurrences of the pattern (their number, how much these occurrences differ from the pattern). Scoring functions are sometimes based on statistical significance. For example we may ask what the probability that a pattern would have so many occurrences if the sequences were generated by random. If this probability is small, the pattern is statistically significant.

The goal of an algorithm may be to find the best (i.e. usually the highest scoring patterns), or to find several best scoring patterns, or all patterns with some predefined level of support and score.

Pattern discovery vs pattern matching

So far we have discussed the problem of pattern discovery, the algorithm is supposed to discover pattern which is unknown in advance. However in biology many consensus sequences are known and it is important to have tools that allow finding occurrences

of known patterns in new sequences. This problem will be called pattern matching.

Pattern discovery problem

Input sequences

The input of pattern discovery programs usually consists of several sequences, expected to contain the pattern. We will use ∑ as the alphabet of all possible characters occurring in the sequences. Thus, ∑ = {A, C, G, T} for DNA sequences and also ∑ is a set of all 20 amino acids for protein sequences. Some algorithms not only make use of sequences, but also may use information about secondary or tertiary structure, evolutionary relationships between sequences and so on. However, we usually concentrate on the discovery of patterns only from unaligned sequences. Following we will describe Pattern representation language to formally express our problem.

Regular expressions

The simplest kind of a pattern is just a sequence of characters from alphabet ∑, such as TATAAAA, the TATA box consensus sequence. We can also allow more complex patterns, adding some of the following frequently used features.

Ambiguous character

This is an unspecified character in a sequence which can be chosen from a certain subset of ∑. Ambiguous character thus matches any character from this subset. Such sets are usually denoted by a list of its members enclosed in square brackets e.g. [LF] is a set containing L and F. A-[LF]-G is a pattern in a notation used in PROSITE database. This pattern matches 3-character subsequences starting with A, ending with G and having either L or F in the middle. For nucleotide sequence there is a special letter for each set of nucleotides, where R=[AG], Y=[CT], W=[AT], S=[GC], B=[CGT], D=[AGT], H=[ACT], V=[ACG], N=[ACGT].

Wild-card or don't care

is a special kind of ambiguous character that matches any character from ∑. Wild-cards can be denoted by N in nucleotide sequences and X in protein sequences. But they are often denoted by dot '.'. Sequence of one or several consecutive wild-cards is called gap and patterns allowing wild-cards are often called gapped patterns.

Flexible gap

Flexible gap is a gap of variable length. In PROSITE database it is denoted by x(i,j) where i is the lower bound on the gap length and j is an upper bound. Thus x(4,6) matches any gap with length 4, 5, or 6. They also denote a gap of length i as x(i) (e.g. x(3) = ...). Finally * denotes gap of any length (possibly 0).

Following string is an example of a PROSITE pattern containing all mentioned features: F-x(5)-G-x(2,4)-G-*-H. Some programs do not allow all these features, for example they do not allow flexible gaps or they allow any gaps but do not allow ambiguous characters other than a wild-card.

Patterns with mismatches.

One can further extend expressive power of deterministic patterns by allowing certain number of mismatches. Most commonly used type of mismatches is substitutions. By considering substitution probability, subsequence S matches pattern P with at most k mismatches, if there is a sequence S' exactly matching S that differs from S in at most k positions.

Sometimes we may also allow insertions or deletions, i.e. the number of mismatches would be an edit distance between the substring S and a closest string matching the pattern P.

Position weight matrices.

Even the most complicated deterministic patterns cannot capture some subtle information hidden in a pattern. Assume we have a pattern that has C as the first letter in 40% cases and G as the first letter in 60% cases. The ambiguous symbol [CG] gives the same importance to both nucleotides. It is so important in strong patterns, but it may be important in weak patterns, where we need to use every piece of information to distinguish the pattern from a random sequence.

The simplest type of probabilistic pattern is position-weight matrix (PWM). PWMs are also sometimes called position-specific score matrix (PSSM), or a profile (however profiles are often more complicated patterns, allowing gaps). PWM is a simple

ungapped pattern specified by a table. For each pair this table contains information about (position; character) and the relative frequency of the character at that position of the pattern (see Fig. 2 for an example). Assume that the pattern (i.e. PWM) has length k (number of columns of the table). The score of a sequence segment $x_1 \ldots x_k$ of length k is:

$$\prod_{i=1}^{k} \frac{A[x_i,i]}{f(x_i)}$$

where A [x_i , i] is an entry of position weight matrix corresponding to position i of the pattern and character c and f(c) is background frequency of character c in all considered sequences. This product represents odd-score that the sequence segment $x_1 \ldots x_k$ belongs to the probability distribution represented by the PWM [22]. In order to simplify computation of the score we can store log-odd scores $\log\frac{A[x_i,i]}{f(x_i)}$ in the table, instead of plain frequencies A[c,i]. Then the following formula gives us log-odd score instead of odd score (A'[c, i] is an entry of the table containing log-odd scores):

$$\sum_{i=1}^{k} A'[x_i,i]$$

Position-weight matrices can be visualized in the form of sequence logos [23] (see Fig. 2). Each column of a sequence logo corresponds to one position of the pattern. Relative heights of the characters in one column are proportional to the frequencies A[c,i] at the corresponding position of the pattern. The characters are displayed in an ordered manner according to their frequency, with the most frequent character on top. Each column is scaled so that its total height is proportional to the information content of the position, computed as

$$\log_2^{|\Sigma|} + \sum_c A[c,i] \log_2^{A[c,i]}$$

The value of $\log_2^{|\Sigma|}$ is added in order to obtain positive values, where this value depends on the size of alphabet $\sum$. Sequence logos were further improved by [24] to take background distribution into account, and display characters that occur less frequently than expected upside down.

PWM with relative frequencies

A	0.26	0.22	0.00	0.00	0.43	1.00	0.11
C	0.17	0.18	0.59	0.00	0.26	0.00	0.35
G	0.09	0.15	0.00	0.00	0.3	0.00	0.00
T	0.48	0.45	0.41	1.00	0.00	0.00	0.54

PWM with log-odd scores (using f(c) =(1/4))

A	-3.94	-4.18	-∞	-∞	-3.22	1.00	-5.18
C	-4.56	-4.47	-2.76	-∞	-3.94	0.00	-3.51
G	-5.47	-4.74	-∞	-∞	-3.74	0.00	-∞
T	-3.06	-3.15	-3.29	-2.00	-∞	0.00	-2.59

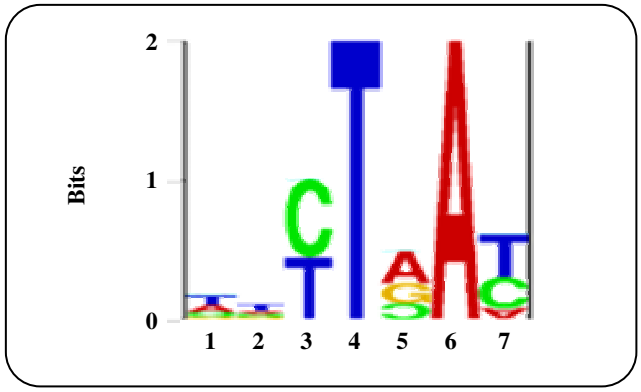

Fig. 2: Position weight matrix of vertebrate branch point in form of a table and corresponding visual representation as a sequence logo. The sequence logo was created using on-line software at http://www.cbs.dtu.dk/gorodkin/appl/slogo.html.

A quick look at a sequence logo reveals most preserved positions, consensus characters at all positions, etc. Notice that size of the characters in different columns cannot be directly compared.

Stochastic models

All types of patterns discussed so far are explicit in a sense that the user can easily see important characteristics of the occurrences of a pattern. Sometimes it is advantageous to represent a pattern in a more implicit form, usually as some discrimination rule, which decides whether a given sequence is an occurrence of the modeled pattern or not. Such a discrimination rule can be based on some stochastic model, such as Hidden *Markov* Model (HMM), or can employ machine learning methods, for example neural nets, and etc. It is possible to argue whether such rules constitute a pattern at all, but obviously

they can be trained (In case of pattern discovery) and then they can be used for discrimination (In case of pattern matching). Therefore they are applicable in pattern-related tasks such as protein family classification, binding sites discovery, etc. In some cases (such as HMMs with simple topology) it is even possible to obtain some information such as relative frequencies of characters at individual conserved positions about a modeled pattern.

Type of Algorithms for Finding Patterns

Many computer science problems related to pattern discovery are provably hard; therefore one cannot hope to find a fast algorithm which would guarantee to find the best possible solution. Therefore many approaches are based on exhaustive search with algorithms that may run in exponential time in the worst case. However these programs often use sophisticated pruning techniques that make the search feasible for typical input data.

Enumerating all patterns.

The simplest approach to pattern discovery is to enumerate all possible patterns satisfying constraints given by the user, for each pattern find its occurrences in input sequences and based on this occurrences assign score or statistical significance to each pattern. Then we can output patterns with highest score or all patterns with scores above some threshold.

Application of enumerative method.

Many protein binding sites in DNA are actually short ungapped motifs, with certain variability. They can be modeled quite well with simple patterns allowing small number of mismatches. Therefore we can apply exhaustive search to find this type of binding sites. Recent examples of this approach can be found in [25, 26]. In both papers authors use straightforward enumeration of all possible patterns and concentrate more on estimating statistical significance of their occurrence. [25] tries to find a pattern that appears in several copies in most sequences (GATA box). [26] allows mismatches and tries to find statistical significance of the given number of occurrences of the pattern with mismatches.

Enumerating gapped patterns.

In some contexts it is more reasonable to search for patterns with gaps. Example of such system is MOTIF [27].

Pruning pattern enumeration.

If we want to find longer or more ambiguous patterns, we cannot use straightforward exhaustive search. Assume that our aim is to find a long ungapped pattern occurring possibly with some mismatches in at least K sequences. We can start searching from short patterns (for example patterns of length 1) that appear in at most K sequences and extend them until the support does not go below K. In each step we need to extend the pattern in all possible ways and check whether the new pattern still occur in at least K sequences. Once we get a pattern that cannot be extended without loss of support, this pattern is maximal and can be written to output. Examples of this strategy include Pratt algorithm and TEIRESIAS algorithm [28]. Both programs find patterns allowing gaps.

Exhaustive search on graphs

Assume we have t sequences of length n, and we want to find a pattern of given length L which occurs in at least q sequences with at most d mismatches. Then if we tae two occurrences of such a pattern, they will differ in at most 2d positions because they both differ from the original pattern in at most d positions. The idea is to make a graph t partite such that each part has n-L nodes (subsequence with L-length) and then there is an edge between two nodes in different parts if two nodes differ at most 2d positions. After making graph, the problem would be transformed to finding a q-clique in the graph [29].

Iterative heuristic methods

So far we have mainly described algorithms that guarantee to find the best pattern. However for more complicated types of patterns we cannot implement such expensive methods. Thus we have to use heuristic approaches that do not necessarily find the best pattern, but may conveniently converge to a local maximum.

Gibbs sampling

Lawrence et al. Presented a heuristic algorithm for pattern discovery based on so called Gibbs sampling method. In the simplest version of this method, we are looking for the best conserved ungapped pattern of fixed length L in the form of position weight matrix assuming that the pattern o

ccurs in all sequences. The algorithm worksin iterations. The algorithm starts by selecting random subsequence of length L from each input sequence. These subsequences will form our initial set of occurrences. Then it would be possible to compute a position weight matrix characterizing the pattern from this weight matrix will be computed based on all occurrences except i'th occurrence. Let's denote this position weight matrix P. Then each subsequence of sequence i of length L is selected, and a score of this subsequence is computed according to matrix P. One of the subsequences that has the best score is chosen afterwards. The algorithm must repeat until for each of the n iterations, information content position weight matrix becomes lower than previous step [30].

Expectation maximization

Lawrence & Reilly used a simple learning algorithm called expectation maximization (EM) algorithm. The purpose of the algorithm is to estimate parameters of the stochastic model of the pattern, which occurs once at an unknown position in each sequence from the given family of sequences. The position weight matrix is used as an underlying stochastic model in [31].

Hidden *Markov* models

Hidden *Markov* Models (HMMs) can be used as a model of a family of sequences. For detailed description of HMMs and related algorithms see [32]. There are three issues, which need to be addressed, when using HMM as a representationof a sequence family:

• Topology of HMM.

Topology specifies general layout of the model, which we use to represent a sequence family.

• Training process.

The learning process is needed to estimate the parameters of the model so that the sum of scores of sequences in the family is optimized.

• Search for sequences.

The searching process should allow us to distinguish between the sequences, that belong to the family and sequences, which do not.

Support Vector Machines

Conventional Support Vector Machines

Support vector machines are supervised learning methods based on structural risk minimization which are proven to be a powerful means of classification with applications in bioinformatics. The introduction of SVM by *Vladimir Vapnik* [33], *Cortes & Vapnik* started a new branch of pattern recognition algorithms based on statistical learning theory [34]. Similar to variety of other supervised learning methods, SVM is used to solve complex problems for which there is no model to map input data to output data. Indeed, establishment of a new paradigm for estimating the functional dependencies from data paved the way for modern statistical learning approaches. According to this paradigm, in order to estimate dependency from data, it is sufficient to know some general properties of the set of functions to which the unknown dependency belongs. In this sense, SVMs learn the dependency solely based on the training data and without having predefined information such as probability distribution of input data or information about the structure of the dependency [35].

Although SVM can be used to classify linear data sets, they are essentially used to classify two sets of vectors which are nonlinearly separable in n-dimensional space. The basic idea behind the classification is that input vectors are mapped to a high dimensional space called feature space through a nonlinear function so that the data becomes linearly separable in this space. This way we transform the problem to a linear problem of finding a n-1 dimensional hyperplane which can separate our n-dimensional vectors in feature space. This plane is called a linear classifier. Although many hyperplanes are available with mentioned property, the desired plane is one with maximum distance from the both set of vectors. This way we can achieve the maximum separation (margin) between the two classes. In this sense, our classification problem is reduced to an optimization problem to find the optimal hyperplane with maximum margin. SVM solutions are obtained from quadratic programming problems which compute a global solution and are not discussed here.

An important and unique feature of this approach is that the solution is based only on those data points, which are at the margin. These points are called support vectors. Another important advantage of the SVM

Is that it is not necessary to implement the nonlinear transformation and to determine the separating hyperplane in the possibly very-high dimensional feature space, instead a kernel representation can be used, where the solution is written as a weighted sum of the values of certain Kernel function evaluated at the support vectors [36].

In the following we're going to explain different classification problems and explain how a SVM can be solved as a quadratic programming problem.

Linear maximal margin classifier

The simplest SVM which only classifies linearly separable data is called maximal margin classifier. Consider the input, $x=\{x_1,x_2,\ldots,x_i\}$a binary classification function f will return a positive value for the inputs which belong to the positive class and return a negative value for the ones belong to the other class. In case f is a linear function of $x \in X$, this separation is actually done by a hyperplane that splits the training samples into two corresponding parts. The following is a formal definition of a training set:

$$S=\{(x_1,y_1), (x_2,y_2),\ldots, (x_i,y_i)\} \subseteq (x\times y)^i \quad x \subseteq R^n \quad y=\{1,-1\}$$

X is a representation of input space and Y represents the domain of the output. And is the number of examples.

We can write function f(x) according to training set as follows:

$$f(x) = (w \times x) + b = \sum_{i=1}^{n} w_i x_i + b$$

where $(w,b) \in R^n \times R$ are parameters that control the function and must be learned from the training data. In the following figure, a hyper plane is demonstrated. As it can be seen, vector indicates the normal vector of the plane, is the vector which decides displacement of the plane from the origin.

Although many hyperplanes can be found that split two data sets, our aim is to find the hyper plane that provides the largest margin. In the following figure, two different hyperplanes and their margins are demonstrated. As it is demonstrated the hyperplane with the largest margin provides a more acceptable classification of data sets with minimum risk of classifying wrong data. The points which lie on a margin are called support vectors.

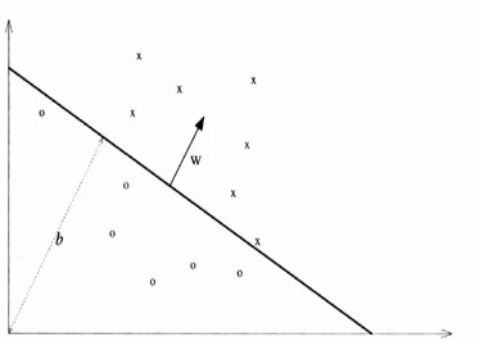

Fig. 3: A linear classifier which separates twpo classes of training data. Vectors b and w are visualized in this picture. Photo taken from [37].

As we can see in the figure the dashed line is the separation line for which the f(x)=0 After the machine learned the parameters w which can be interpreted as weights and b which is the bias, it can generalize the unseen pattern x_p using the decision function as follows:

$$h(x) = 0 = \text{sign}(f(x))$$

Where "o" is the notation for the output. Note that both i and f(x) belong n+1 to dimensional space. We define the hyperplane f(x)=0 as separation boundary which exists in the n-dimensional space as input vectors does. In the following figure we can see that many separating planes exist for a given set of inputs. Indeed, given function f(x) with parameters (w,b), all the hyperplanes with parameters (kw,kb) where is a positive value, are acceptable separation hyperplanes. Since (w,b)and (kw,kb) define the same separation boundaries, we have to define notion of canonical hyperplane as follows:

$$\min \left| (w.x_i) + b \right| = 1$$

This implies that for every support vector the value of f(x) is either 1 or -1 which means that the margins for our hyperplane are 1 and -1. Another interpretation of this situation is that for canonical hyperplanes the values of and are equal for support vectors as it is shown in the following figure.

The numbers of canonical hyperplanes which can separate the data correctly are far smaller than numbers of hyperplanes. However, it is necessary to find the optimal canonical hyperplane which provides the maximal margin. In fact, in a search for a classification

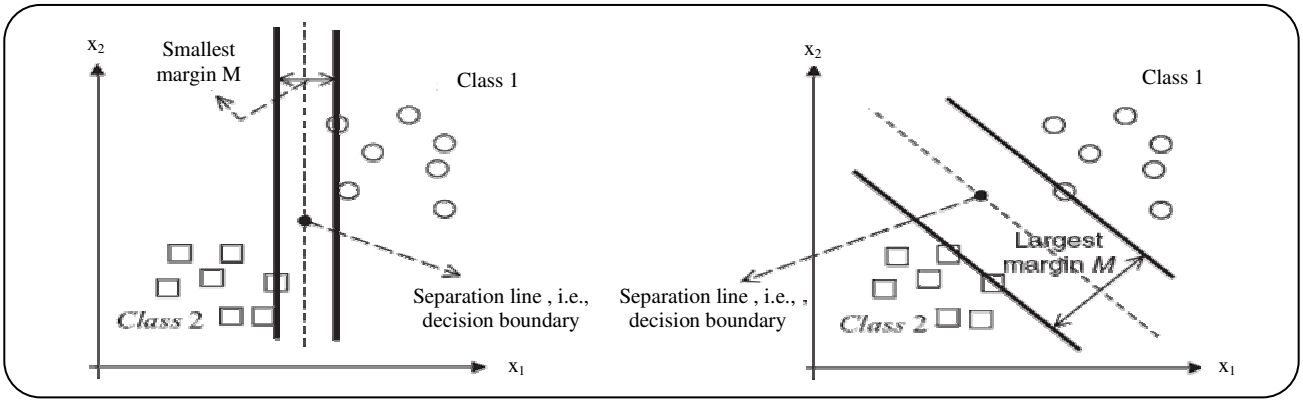

Fig. 4: This figure demonstrates two different separating lines with different margins for a set of training data. It is obvious that the line in the right figure provides a wider margin than the left one and thus is more desirable. (Photo taken from [38]).

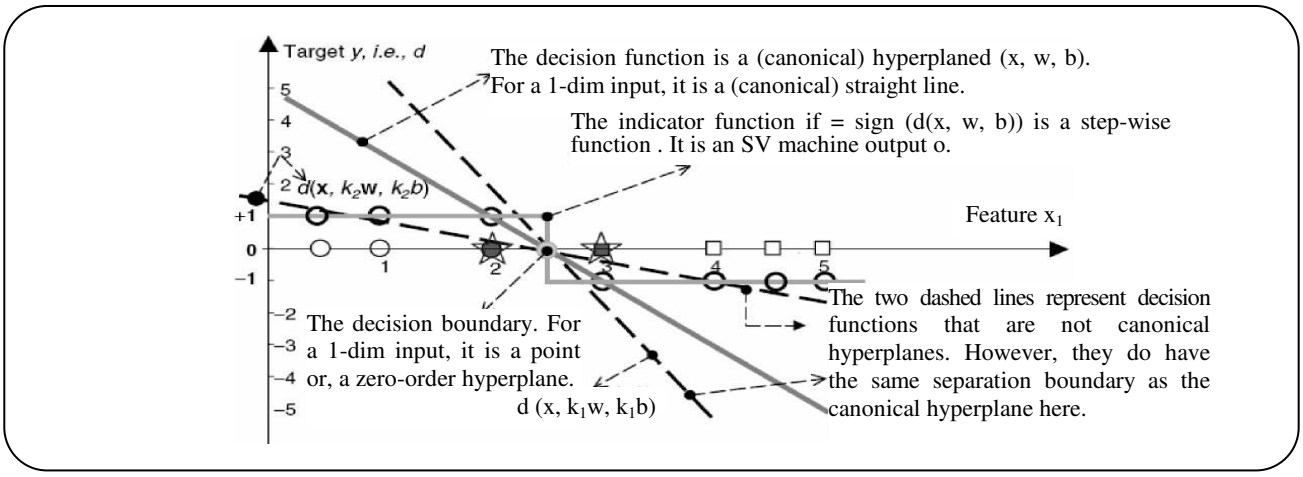

Fig. 5: This photo demonstrates the difference between canonical separating line and other separating files. (Photo taken from [38]).

function, a hyperplane with maximal margin is desirable for us since it has lower risk of misclassification and a canonical hyperplane is desirable because it will make the search for support vectors easier.

The length of margin M of a canonical hyper plane can be easily calculated and is

$$M = \frac{2}{\|w\|}$$

From the above equation it can be inferred that by minimizing the norm of we can maximize the margin M. Since $\|w\| = \sqrt{w_1^2 + w_2^2 + ... + w_n^2}$ so maximizing $\|w\|^2$ would have the same effect as maximizing $\|w\|$ subject to inequality constraints introduced later which lead us to solving the classical quadratic programming problem.

The following is the definition of our optimization problem: Given a set of training data

$$S=\{(x_1,y_1), (x_2,y_2),\ldots, (x_i,y_i)\} \subseteq (x\times y)^i \quad x \subseteq R^n \quad y=\{1,-1\}$$

In order to find a maximal margin hyperplane we have to solve the following problem:

minimize (w,w)

subject to $y_i((w,x)+b \geq 1)$ i= 1…t

In optimization problems, Lagrangian Multipliers is a method to find the exterma of function subject to one or more constraints. The following is the primal lagrangian of our problem:

$$L(w,b,a) = \frac{1}{2}(w.w) - \sum_{i=1}^{t} a_i[y_i((w.x)+b)-1]$$

where a_i are lagrangian multipliers. We now show

how to transform a lagrangian to its dual form. A dual representation of a Lagrangian is another form of the problem which is easier to solve than the primal form and is more convenient when using kernel functions. The dual of a lagrangian is calculated by differentiating the primal form with respect to w and b

$$\frac{\delta L(w,b,a)}{\delta w} = w - \sum_{i=1}^{t} y_i a_i x_i = 0$$

$$\frac{\delta L(w,b,a)}{\delta b} = \sum_{i=1}^{t} y_i a_i = 0$$

which results in

$$w = \sum_{i=1}^{t} y_i a_i x_i \quad , \quad \sum_{i=1}^{t} y_i a_i = 0$$

The first result implies that we can write vector w as a linear combination of training points. By substituting above equations in the primal form we will have the dual form as follows

$$L(w,b,a) = \sum_{i=1}^{t} a_i - \sum_{i=1}^{t} y_i y_2 a_i a_2 (x_i . x_2)$$

Now our problem transformed to solving the following quadratic problem:

Maximize $$w(a) = \sum_{i=1}^{t} a_i - \sum_{i,j=1}^{t} y_i y_j a_i a_j (x_i . x_j)$$

Subject to $$\sum_{i=1}^{t} y_i a_i = 0 \qquad a_i \geq 0 \ , \ t=1,\ldots,t$$

The solution to the stated problems is a set of parameters indicating vector w which constitutes the maximum margin hyperplane. To calculate vector b of the hyperplane we must refer to the primal form of the lagrangian.

Nonlinear Classification

As it was mentioned before, the essential application of SVMs are in nonlinear classification. Yet, the significance of the method explained above is in that it can be easily extended to nonlinear classification. In order to do so, we can make use of kernel representations which map the input vectors x to vectors $\phi(x)$ of a high dimensional space called feature space to increase the computational power of linear machines. As a result of this mapping, it is possible to solve nonlinear problem of classifying vectors linearly by applying linear SVM formulation on vectors $\phi(x)$ in the feature space. This will lead to solving a quadratic programming problem with similar constraints on $\phi(x)$.

An advantage of this method is that the dual representation of the problem makes it possible to perform the mapping implicitly.

If $\phi 1 x \longrightarrow F$ is a nonlinear mapping from input space to feature space then the separator hyperplane we're looking for would be of the form

$$f(x) = \sum_{t=1}^{n} w_t \phi_t (x)$$

As we saw in previous section vector w can be written as the linear combination of training points which is $\sum_{i=1}^{t} y_i a_i x_i$ which allows us to rewrite the formula in its dual

form:

$$\sum_{i=1}^{n} y_i a_i (\phi(x_i).\phi(x))$$

This implies that decision function h(x) can be calculated by having inner products of training points and unseen points. One of the problem we face in this kind of machines is the problem of choosing a mapping function which result in a good separating hypersurface. Another problem is calculating the product of $\phi(x_i).\phi(x)$ which could be a formidable task considering probable large dimensions of the feature space.

Fortunately, a set of function called kernel functions can directly calculate the inner product $\phi(x_i).\phi(x_j)$ by only having x_i and x_j as their inputs. A kernel K, is a function on space X , such that for every x_i , $x_j \in$ X.

An immediate advantage of using kernel functions is that there's no need to choose the mapping function ϕ and calculate the mapped points $\phi(x_i)$ explicitly. Since decision

Function h(x) does only need the inner products of $\phi(x_i).\phi(x)$ we could directly calculate them by means of $K(x_i,x_j)$ Thus kernel functions make it possible to train SVMs which operate in high dimensional spaces and avoid computational problems of high

Kernel Functions	Type of Classifier
$K(x, x_i) = (x^T x_i)$	Linear, dot product, kernel, CPD
$K(x, x_i) = [(x^T x_i) + 1]^d$	Complete polynomial of degree d, PD
$K(x, x_i) = e^{\frac{1}{2}[(x-x_i)^T \Sigma^{-1}(x-x_i)]}$	Gaussian RBF, PD
$K(x, x_i) = \tanh[(x^T x_i) + b]^*$	Multilayer perceptron, CPD
$K(x, x_i) = \frac{1}{\sqrt{\|x-x_i\|^2+\beta}}$	Inverse multiquadric function, PD

* only for certain values of b, (C)PD = (conditionally) positive definite

Fig. 6: This table contains some of the popular Kernel functions [38].

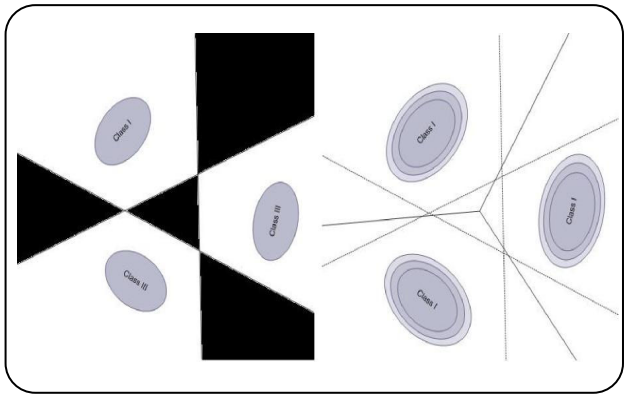

Fig. 7: The black colored area in left picture represents unclassified areas, where these areas are covered by fuzziness model of SVM.

dimensionality. Besides we don't even have to know the underlying mapping $\phi(x)$.

We can rewrite our classifier as follows:

$$f(x) = \sum_{i=1}^{t} y_i a_i k(x_i . x)$$

The learning process in a nonlinear machine is similar to the method used in linear machines. The dual Lagrangian for a given mapping ϕ would be:

$$L(w, b, a) = \sum_{i=1}^{t} a_i - \sum_{i,j=1}^{t} y_i y_j a_i a_j \phi(x_i)\phi(x_j)$$

which can be written in terms of $K(x_i, x_j)$:

$$L(w, b, a) = \sum_{i=1}^{t} a_i - \sum_{i,j=1}^{t} y_i y_j a_i a_j K(x_i, x_j)$$

Subject to $\sum_{i=1}^{t} y_i a_i = 0 \quad a_i \geq 0$, t=1,…,t

This is the problem of finding the optimal separating hyperplane in the features space. The following table contains some of the popular kernel functions:

Fuzzy support vector machines

Concept of fuzzy states from *Zadeh*'s definition is commonly used definition for uncertain conditions. Crawling through absolute true to absolute false is what is used here to adapt Support Vector Machines (SVM). As mentioned, SVM primarily introduced for two class classification; moreover, a multi-class classification method have been proposed that via pair wise two-class classification tackled multi-class classification [39-41]. Thereafter, to solve multi-class classification problem Fuzzy SVM has been introduced. Assuming fuzzy class ID for members of a SVM's training set is one of the important propensities to use fuzziness concept on multi-class classification where as epicted in Fig. 7, using fuzziness concepts yields to covering whole of space but conventional SVM process remains some unclassified portions [41, 42].

In summary, applying fuzziness to support vector machines leads to making a flexible (as fuzzy functions), practical, and precise (as SVM methods [43]) tool. This combined approach can be useful in discovering and annotating genome sequence in structural repeated subsequences.

ANNOTATING STRUCTURAL REPETITIVE SEQUENCES VIA FUZZY SUPPORT VECTOR MACHINES

So far, different approaches have been introduced to tackle the problem of pattern annotating automatically. But as issued in [44] this problem is exhaustively unclear and there is not a efficient enough algorithm for it. This assumption comes from two biological facts; during evolution process nucleotides are mutated and all kind of structural repetitive sequences are not discovered yet. Some works have been performed to present approximate algorithms to reduce effects of mutated nucleotides [45] or by presenting self trained algorithm discover non-discovered structural repeated subsequences [46]. Another reason that

should be considered is subsequences similarity upon their functionality, where considering this reason needs some pre-information about each significant subsequence. To gather these information significant subsequences should be determined first. Indeed, this is our fundamental problem and by solving this problem the problem of finding structural repeated subsequences is resolved. Yet, this loop makes problems unsolvable. In the following, the last reason is neglected and context based repeated subsequences are considered.

In our previous job we had determined structural repeated subsequences [41] via support vector machines method, but as it was mentioned combining this method with fuzziness concept can increase the accuracy and yield to much better results.

CONCLUSION

As mentioned in section of biological aspect, transposable elements are divided to more than two classes-considering subclasses-, therefore some kind of multi-class classification SVM is needed. Proposed n(n-1)/2 pair wise multi-class classification problem which was the first step of combining multi-class classification SVM and fuzziness concepts [15]. Also some other approaches were introduced by combining fuzziness and SVM concepts to solve the problem of unclassified regions which is problem of conventional SVM [36]. By combining the proposed methods in [37] -for converting the problem of structural repeated subsequences from biological systems to a problem in support vector machines - and method proposed by [15] -to solve multi-class classification via fuzzy support vector machines-, the problem of identifying structural repetitive subsequences in biological systems can clearly be solved.

The higher-order structure of DNA, including hairpin turns, bending and curvature, and precise chromatin topology, could provide novel metadata needed to explain genome complexity [41]. Recent data also shows that promoter regions are significantly more curved than coding regions or randomly permuted sequences [43]. Tandem repeats are ubiquitous sequence features in both prokaryotic and eukaryotic genomes. A direct or tandem repeat is the same pattern recurring on the same strand in the same nucleotide order. Tandem repeats play significant structural and functional roles in DNA [41]. These repeats also play a regulatory role when found near genes and perhaps even within genes. Short tandem repeats are used as a convenient tool for the genetic profiling of individuals or for genetic market analysis in mapping studies [41, 42]. Today it has been proved that the function of a genes in genome is not a gene-only role but they act together and form a network of genes playing in a biological process. Many attempts have been done to bring these information together and build different biological networks [44,45]. Gene regulatory networks are examples that are modeled and analyzed in order to gain insight of their exact functions [42].

Acknowledgement

We would appreciate great help from *Fatemeh Zareh* during this project. Part of this work has been supported by Iran National Science Foundation (http://www.insf.org).

Received : March 19, 2010 ; Accepted : July 19, 2010

REFERENCES

[1] Britten R.J., Graham D.E., Neufeld B.R., Analysis of Repeating DNA Sequences by Reassociation Kinetics, *Methods Enzymol*, **29**, p. 363 (1974).

[2] Consortium I.H.G., Initial Sequencing and Analysis of the Human Genome. International Human Genome Consortium, *Nature*, **409**, p. 860 (2001).

[3] Brejova B. et al., Finding Patterns in Biological Sequences, in Technical Report, University of Waterloo. (2000).

[4] Linial M. et al., Global Self-Organization of All Known Protein Sequences Reveals Inherent Biological Signatures, *J. Mol Bio*., **268**, p. 539 (1997).

[5] Rigoutsos I. et al., Dictionary Building via Unsupervised Hierarchical Motif Discovery in Sequence Space of Natural Proteins, *Proteins*, **37**, p. 264 (1999).

[6] Nevill-Manning et al., Highly Specific Protein Sequence Motifs for Genome Analysis, *Proceedings of the National Academy of Sciences of the United States of America*, **95**, p. 5865 (1998.).

[7] Savoie C.J. et al., Use of BONSAI Decision Trees for the Identication of Potential MHC Class I Peptide Epitope Motifs. In "Pacific Symposium on Biocomputing", pp. 182-189 (1999).

[8] Califano A., SPLASH: Structural Pattern Localization Analysis by Sequential Histograms. *Bioinformatics*, **16**, p. 341 (2000).

[9] Pedersen A.G. et al., The Biology of Eukaryotic Promoter Prediction, *Computers and Chemistry*, **23**, p. 191 (1999).

[10] Mironov A.A. et al., Computer Analysis of Transcription Regulatory Patterns in Completely Sequenced Bacterial Genomes, *Nucleic Acids Research*, **27**, p. 2981 (1999).

[11] Hughes J.D. et al., Computational Identication of Cis-Regulatory Elements Associated with Groups of Functionally Related Genes in Saccharomyces Cerevisiae, *Journal of Molecular Biology*, **296**, p. 1205 (2000).

[12] Gelfand M.S., Koonin E.V., Mironov A.A., Prediction of Transcription Regulatory Sites in Archaea by a Comparative Genomic Approach. *Nucleic Acids Research*, **28**, p. 695 (2000).

[13] Gorodkin J., Heyer L.J., Stormo G.D., Finding the Most Significant Common Sequence and Structure Motifs in a Set of RNA Sequences, *Nucleic Acids Research*, **25**, p. 3724 (1997).

[14] Benson G., Tandem Repeats Finder: aA Program to Analyze DNA Sequences, *Nucleic Acids Research*, **27**, p. 573 (1999).

[15] Wicker T., François Sabot, Hua-Van A., A Unified Classification System for Eukaryotic Transposable Elements, *Nature Reviews Genetics*, **8**, p. 973 (2007).

[16] Dorohonceanu B., Nevill-Manning C.G., Accelerating Protein Classification Using Suffix Trees. In "Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, pp. 128-133 (2000).

[17] Schneider T.D., Stephens R.M., Sequence Logos: A New Way to Display Consensus Sequences, *Nucleic Acids Research*, **18**, p. 6097 (1990).

[18] Gorodkin(a) J., Heyer L.J., Stormo G.D., Displaying the Information Contents of Structural RNA Alignments: the Structure Logos, *Computer Applications in the Biosciences*, **13**, p. 583 (1997).

[19] van Helden J., Andre B., Collado-Vides J., Extracting Regulatory Sites from the Upstream Region of Yeast Genes by Computational Analysis of Oligonucleotide Frequencies, *Journal of Molecular Biology*, **281**, p. 827 (1998).

[20] Tompa M., An Exact Method for Finding Short Motifs in Sequences, with Application to the Ribosome Binding Site Problem. In "Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology", pp. 262-271 (1999).

[21] Smith H.O., Annau T.M., Chandrasegaran S., Finding Sequence Motifs in Groups of Functionally Related Proteins, *Proceedings of the National Academy of Sciences of the United States of America*, **87**, p. 826 (1990).

[22] Rigoutsos I., Floratos A., Motif Discovery without Alignment or Enumeration (Extended Abstract). In "Proceedings of the second annual international conference on Computational molecular biology", pp. 221-227 (1998).

[23] Pevzner P.A., Sze S.H., Combinatorial Approaches to Finding Subtle Signals in DNA Sequences. In "Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology", pp. 269-278 (2000).

[24] Lawrence C.E. et al., Detecting Subtle Sequence Signals: a Gibbs Sampling Strategy for Multiple Alignment, *Science*, **262**, pp. 208-214 (1993).

[25] Lawrence C.E., Reilly A.A., An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences, *Proteins*, **7**, p. 41 (1990).

[26] Durbin R. et al., "Biological Sequence Analysis", Cambridge University Press (1998).

[27] Boser B., Guyon I., Vapnik V., A Training Algorithm for Optimal Margin Classifiers, in "Annual Workshop on Computational Learning Theory", Pittsburgh: ACM Press (1992).

[28] Cortes C., Vapnik V., Support-Network. "Machine Learning", 20 (1995).

[29] Vapnik V., "The Nature of Statistical Learning Theory". New York.: Springer-Verlag (1995).

[30] Suykens J.A.K., Advances in Learning Theory: Methods, Models and Applications Nato Science

Series. Series III, Computer and Systems Sciences, **190**, Amsterdam (2003): I O S Press. Cristianini, J.S.-T., "Supportector Machines and other kernel-based learning methods". Cambridge University Press (2000).

[31] Wang L., "Support Vector Machines: Theory and Applications". Springer (2005).

[32] Kreßel U.H.-G., Pairwise Classification and Support Vvector Machines, "Advances in Kernel Methods: Support Vector Learning", ed. B. Sch¨ olkopf C.J.C.B., Smola A.J., Cambridge, MA: The MIT Press (1999).

[34] Platt J.C., Cristianini N., Shawe-Taylor J., Large Margin DAGs for Multiclass Classification, in "Advances in Neural Information Processing Systems", Solla T.K.L.S.A., Muller K.-R., Editor, The MIT Press (2000).

[35] Inoue T., Abe S., Fuzzy Support Vector Machines for Pattern Classification. in IJCNN '01, "International Joint Conference Neural Networks", Washington, DC, USA: IEEE (2001).

[36] Abe S., Inoue T., Fuzzy Support Vector Machines for Multiclass Problems. in ESANN'2002 Proceedings, Bruges (Belgium): "European Symposium on Artificial Neural Networks" (2002).

[37] Dashti H, Masoudi-Nejad A., De novo LTR Discovery and Clustering, a Support Vector Machine Approach in Center of Excellence in Biomathematics, Institute of Biochemistry and Biophysics: Tehran (2007).

[38] Bao Z., Eddy S., Automated De Novo Identification of Repeat Sequence Families in Sequenced Genomes, *Genome Res*, **8**, p. 1269 (2002).

[39] Pevzner P.A., Tang H., Tesler G., De Novo Repeat Classification and Fragment Assembly. *Genome Res*, **14**, p. 1786 (2004).

[40] Price A.L., Jones N.C., Pevzner P.A., De Novo Identification of Repeat Families in Large Genomes, *Bioinformatics*, **21**, p. i351 (2005).

[41] Masoudi-Nejad A., Movahedi S., Jáuregui R., Genome-Scale Computational Analysis of DNA Curvature and Repeats in Abidopsis and Rice Uncovers Plant-Specific Genomic Prop-Erties, *BMC Genomics,* **12**, p. 214 (2011).

[42] Zamani Z., Hajihosseini A., Masoudi-Nejad A., Computational Methodologies for Analyzing, Modeling and Controlling Gene Regulatory Net-Works, *Biomedical Engineering and Computational Biology*, **2**, p. 47 (2010).

[43] Askary A., Masoudi-Nejad A., Mizbani A., Naderi-Parizi S., Purmasjedi M., N4: A Precise and Highly Sensitive Promoter Predictor Using Neural Network Fed by Nearest Neigh-Bors, Genes and Genetic Systems, **84**, 425-430 (2009).

[44] Masoudi-Nejad A., Tonomura K., Kawashima S., Itoh M., Kanehisa M., Endo T., Goto S., EGassembler: Online Bioinformatics Service for Large-Scale Processing, Clustering and Assembling ESTs and Genomic DNA Fragmets, Nucleic Acids Research, **34**, W459-W462 (2006).

[45] Masoudi-Nejad A., Goto S., Jauregui R., Itoh M., Kawashima S., Moriya Y., Endo T., Kanehisa M., EGENES: Transcriptome-Based Plant Database of Genes with metabolic Pathway Information and EST in-Dices in KEGG, *Plant Physiology*, **144**, p. 857 (2007).