Key words – bioinformatics, DNA, classification

Rafał BIEDRZYCKI[*], Jarosław ARABAS[+]

CONSTRUCTIVE INDUCTION IN BIO-SEQUENCES

This paper is devoted to analysis of DNA sequences. The aim of the analysis is to recognize biologically important sites, i.e. the coding and non-coding parts of the DNA sequence. We introduce a novel algorithm which uses constructive induction to perform classification tasks. Each attribute of a DNA sequence is based on the definition of certain pattern. The pattern construction problem is defined as a search task in the space of all possible patterns. We briefly discuss possible application of the algorithm to domains different than DNA analysis, where processing of datastream is required.

1. INTRODUCTION

Bioinformatics is a quite new field of research bringing together biology, mathematics and computer science. Since the inception of the Human Genome Project, bioinformatics has drawn much of attention. Upon completing any genome project one needs to analyze and interpret the vast amount of data (about 3 billion long DNA sequence for human). DNA sequence is a carrier of complete information about organism. It is composed of small building blocks – nucleotides. Each nucleotide consists of three parts: a base molecule (a purine or a pyrimidine), sugar, and one or more phosphorate groups. The purines are: adenine (A) and guanine (G), and the pyrimidines are cytosine (C) and thymine (T). Every DNA strand has its head called 5' end and its tail called 3' end. To be more stable, DNA strand is connected with another, a complementary one. Complementarity means that in each respective site, each strand contains a nucleotide complementary to the one in the other strand. Adenine bonds exclusively with thymine (A-T) and guanine with cytosine (G-C). The connected strands have opposite directions and are composed of complementary parts called base pairs (bp). When DNA is temporally in the form of a single sequence, it could connect with itself (A-T, C-G) and form two- and three-dimensional structures. Such structures, if formed have an influence on various operations, e.g. it could stop transcription of DNA to RNA and, in the consequence, the DNA would not lead to the synthesis of proper proteins as we discuss later in the text.

Identification of all functional elements in the genome, including genes and the regulatory elements, is a fundamental challenge in genomics and computational biology [5]. The first task is genome annotation which consists in finding sequences that encode proteins (genes). The synthesis of a protein is a two-step decoding process. The sketch of that process is depicted in Figure 1. The first stage is the transcription from DNA to RNA; the second stage is the translation of RNA to proteins. In higher organisms (eukaryotes) transcription produces pre-mRNA – the sequence that is

[*] Institute of Electronic Systems, Warsaw University of Technology, rbiedrzy@elka.pw.edu.pl
[+] Institute of Electronic Systems, Warsaw University of Technology, jarabas@elka.pw.edu.pl

composed of regions (subsequences) that are responsible for information coding (exons) and those which are not (introns). During splicing, the mRNA is produced by removing introns from the pre-mRNA by spliceosome; mRNA (messenger RNA) is a sequence that encodes a protein. The next stage of the protein synthesis process is translation. If we need to predict the protein sequence we should only find the ORF (Open Reading Frame) sequence – a part of the sequence that undergoes translation [9].

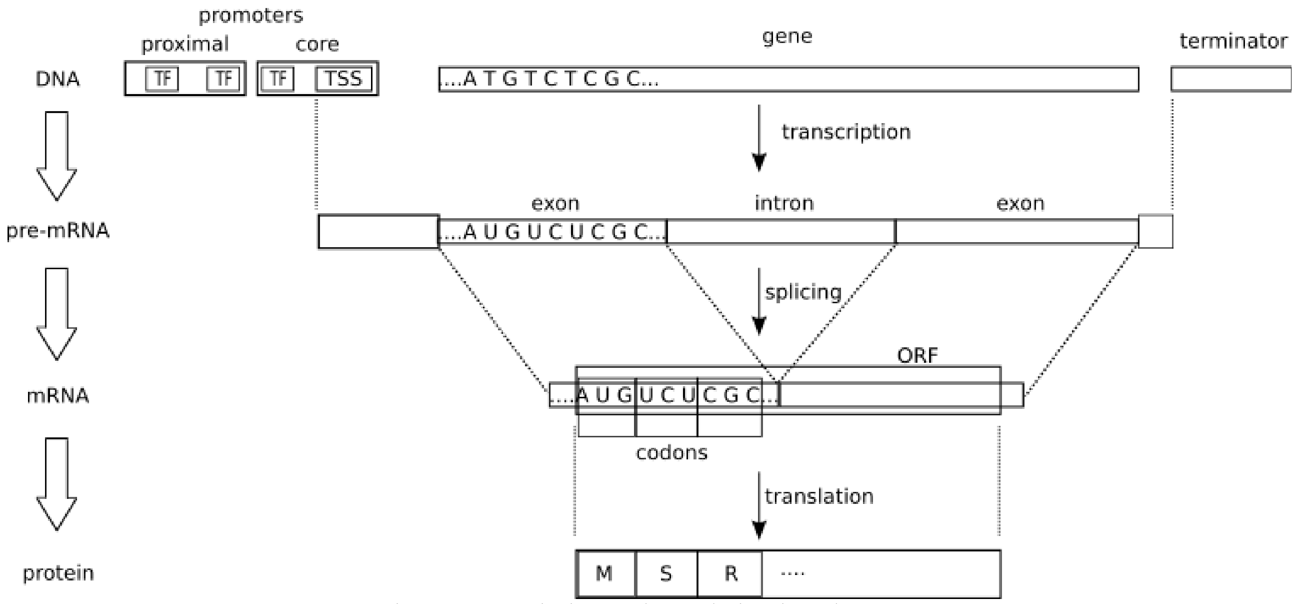

Fig. 1. Transcription and translation in eukaryotes

So far the annotation process has been performed in a collective way – by the community annotation via the Internet [4]. That kind of annotation needs supporting experimental evidences and www servers with dedicated tools. There exist libraries of such evidences that collect DNA resulting from the reverse transcription from mRNA, so they are composed of exons only. Such long sequences are called cDNA; short sequences, caused by the errors in reverse transcription, are called EST. Internauts, after reading tutorial, use www tools to put their annotations based on cDNA, EST, and a general knowledge. Those annotations are later accepted or rejected by experts – curators of the database. This procedure needs great effort of many experts and still there are no supporting sequences for some annotations.

DNA annotation covers the understanding of all functions that are encoded in the DNA sequence. In this paper, we propose a framework for performing annotation tasks in biosequences. For clarification purpose we will concentrate on recognizing coding sequences (exons) in DNA sequences.

1.1. PATTERN REPRESENTATION

Every problem connected with the computational DNA analysis, annotation being an example, is at least related to the issues of pattern representation, building, and matching.

Several notations for describing patterns have been used [8], but most of them are the versions of a standard notation used in regular expressions. The basic pattern notation is just a sequence of characters, each of them denoting a single nucleotide or a group of nucleotides. A more powerful representation introduces ambiguous symbols. An ambiguous symbol is the expression that allows more than a single nucleotide in certain position. In the most commonly used notation [8], alternatives for considered position are enclosed in square brackets, e.g. A-[TC] matches sequences

AT and AC. A special case of those patterns are patterns with so called "don't care" symbols or gaps, e.g. A-x-T-x-x-G. Such symbol represents any letter from the alphabet. A flexible gap is defined by two numbers: the minimum and the maximum number of "don't cares" allowed, e.g. [CG]-x(2,3)-T matches sequences of the length four or five that end with T and start with C or G. That kind of patterns is used, for instance, in the PROSITE database (database of protein families described by common patterns). PROSITE notation introduces also symbols which allow for relating pattern to the matching position. Symbol '<' before pattern *P* means that we require *P* to match at start of a sequence, and '>' after *P*, requires *P* in the sequence end. Such patterns are quite powerful but hard to be effectively acquired from sequences.

Another approach is a matrix pattern representation. The basic idea is to use a matrix of numbers containing scores for each nucleotide in each position of a fixed-length subsequence. There are two types of weight matrices: a position frequency matrix (PFM) and a position weight matrix (PWM). PFM records the position-dependent frequency of each nucleotide, and PWM contains logarithmic weights for computing a match score.

Patterns are also modelled by Markov Models. The *k*-order Markov Model is the probabilistic model that takes into consideration *k* previous elements in the sequence. The higher order of the model, the better ability to describe DNA, but the more parameters to be estimated (so the harder to compute). In practice *k* ranges from 0 to 5. Unfortunately even such a sophisticated model cannot capture distant dependences which often occur in DNA sequences.

1.2. ALGORITHMS FOR GENE FINDING

A main part of the annotation process is gene finding. This task includes detection of protein coding genes and splice sites. There are two types of splice sites: a donor site (start of the intron) and an acceptor site (the intron end).

Most of the algorithms, like those described in [9], decompose the splice site detection problem into two tasks: finding a donor site and finding an acceptor site. Saeys [9] proposed an approach based on a classifier with the attributes induction. He aligned all training sequences to start in a certain site, e.g., in a donor site. After that, he generated position dependent features using the schema: if a nucleotide T is in the right first position of the donor site, then the attribute T_1 is set to 1 and attributes A_1, C_1, G_1 are set to 0. In a similar way he also defined a large set of position independent attributes. He also performed transformation of DNA sequence into the Fourier spectrum trying to capture periodicity in DNA, and, as the result of that, he generated additional attributes. He used counters of AT/TG dinucleotide percentage upstream and other features. Having huge amount of features he used few of the feature selection techniques to reduce the number of features. After that, he applied a few classification algorithms.

Although he defined such large set of attributes, he still did not cover all the important DNA features. DNA and RNA could fold into the secondary structure. That kind of folding depends mainly on complementary matches of distant parts of a sequence which could not be captured by his approach. Defining large set of simple attributes needs great computational effort. Moreover, classification algorithm cannot be successfully applied to such large set. There is a need to use feature selection techniques that unfortunately do not guarantee that important features will be conserved in the final feature set.

Another popular approach is represented by Genscan algorithm [3]. Genscan models gene using finite state automata. Each state that corresponds to important functional sequence fragment (intron, exon, promoter, intergenic region and others) is modelled by the Hidden Markov Model (HMM).

This approach has also its drawbacks. Predicted gene number may not be correct; the algorithm was developed for human/vertebrate sequences, which results in lower accuracy for other types of sequences. Internal exons are predicted more accurately than initial and terminal exons. Another constraints comes form using HMMs, e.g. algorithm cannot capture long-distant interactions in DNA sequence.

2. THE KIS ALGORITHM

Because existing approaches are still far away from being perfect and they do not consider some important properties of DNA, we proposed a new algorithm for gene finding called KIS (KIS is an abbreviation from Polish translation of: constructive induction in sequences).

2.1. PROBLEM DEFINITION

The term “gene finding” is narrower than annotation but it is still connected with many tasks. We will concentrate only on one of them – recognizing exons. The presented problem we perceive as a classification task – problem of classifying sequences into the two classes: exons and not exons. We use the decision tree based classifier and ID3 algorithm [7]. To be able to use the classifier we need attributes. The process of generating new attributes from the data is called constructive induction [6]. The main goal of KIS is to generate good attributes from the DNA sequence, a good attribute should meaningly contribute to the classification accuracy; in the same time, the attribute value should be easy to compute.

2.2. ATTRIBUTES

We focus on attributes that can be derived from similarity patterns only (i.e., we do not use Fourier transforms). Therefore to define an attribute, a similarity pattern should be defined first. As a base for our pattern notation we choose PROSITE notation. That notation was extended to capture important DNA features, and in particular, it allows to express distant relationships, e.g. A-x(100,200)-T, and it is easy to extend and readable for humans.

To express some DNA features (2D conformations, periodicity in exons), we use additional special symbols added to the standard PROSITE notation:

- **C(*l*, f_L, f_H)** – complementary sequence,
- **P(*l*, f_L, f_H)** – palindromic sequence,
- **L(*l*, f_L, f_H)** – palindrome of complementary sequence,
- **R(*l*, f_L, f_H)** – repeated sequence,
- **Q(*l*, *t*, *n*)** – sequence repeated with a period *t*.

Parameter *l* is the sequence length; parameter f_L is minimum and f_H maximum gap length. Parameter *n* is the percentage of hits, e.g. 50% means that sequence matches in half of possible positions according to period *t*. Example: assume that we have pattern **P(3,8,9)**-T-C-A-G. The parameter *l* here is 3, f_L is 8 and f_H is 9. The number *l*=3 is positive that means that palindrom of 3 consecutive nucleotides from the right hand site of P (underlined sequence) will be placed on the left of P after flexible gap x(8,9). When *l* is a negative number, then nucleotides from the left of P will be placed on the right. In this example after replacing **P(3,8,9)** with appropriate nucleotides and flexible gap we will have pattern A-C-T-**x(8,9)**-T-C-A-G. Second example: pattern A-T-T-C-A-**P(-3,2,2)** will become A-T-T-C-A-**x(2,2)**-A-C-T. Proposed extensions to a pattern language allow for building improper patterns, i.e. patterns with conflicts on some positions. This problem is easy to eliminate during pattern construction process. Every extended pattern will be converted to the standard PROSITE notation in the way mentioned above.

We will say that pattern matches sequence when sequence contains subsequence with nucleotides conformant with the pattern. Some patterns are also connected with their correct relative matching point, e.g. some pattern may match at second position from the beginning of exon. If patterns without information about relative matching point are only allowed, than short but very distinctive subsequences will not be discovered.

Now on we are able to define attribute. Every attribute is connected with exactly one pattern and it returns value one when pattern matches a sequence, and zero otherwise.

As we already mentioned earlier a good attribute should be “orthogonal” to the others, that is, it should meaningly contribute to the classification accuracy. The word orthogonal will be used in that context, not in its pure mathematical meaning.

Because we have large data sets we could allow simple orthogonalization approach. We will eliminate all correctly classified sequences from training set. In that case a good new attribute would allow to classify cases which were improperly classified with the other attributes, so every good attribute would be orthogonal to others.

2.3. METRIC SPACE OF PATTERNS

Consider the set of patterns. Each pattern matches a set of sequences. Pattern P_1 is more general than P_2, if the set of sequences matched by P_2 is completely contained in the set of sequences for P_1; in addition, the set of sequences for P_1 may contain sequences that are not matched by P_2. We will refer to P_2 as to a more specific pattern. Specifity and generality relation can be used to define a metric in the space of sequences. As a metric choose a function that equals a minimum number of elementary operations that have to be performed to get one pattern from another one. An elementary operation is either concretization (making the pattern more specific) or generalization of a pattern. The size of such defined space is huge but only current working point will be stored in a memory. In each iteration the set of all points is generated that are neighbours of a current point. Then patterns that do not fit the training data are removed from the set of neighbours. That is why searching over that space is feasible.

Note that it is possible to get any pattern from any other one with the finite set of elementary changes, so the search space is consistent and the metric values have a finite upper bound. Moreover, for a pair of patterns it may be possible to find many distinct paths between them.

2.4. SCORING FUNCTION

The score result of a pattern should be related to the number of the pattern matches in positive class (exons), in negative class, and to the number of cases when the pattern is met exactly in the same position in different sequences. That position could be related to the left or the right of subsequences form certain class (e.g. exons). To express afore mentioned features as one number we define the following scoring function to be maximized:

$$Q = | S_{LP} + S_{RP} - S_{LN} - S_{RN} | \qquad (1)$$

where *L* and *R* indexes for scores *S* stands for left and right sequence alignment; *P* and *N* stands for positive and negative class, and a score for appropriate alignment and class is computed from equation:

$$S = \sum_{i \in M} n_i^2 \qquad (2)$$

where M is the set of positions of pattern match and n_i is the number of matches at position i.
The $Q = 0$ is minimum score and it means that pattern do not match to the sequence or the pattern has weak distinguish power.

2.5. SEARCH METHODS

We perceive the problem of pattern induction as a search task in the space of patterns. The search starts from some starting point and it is performed until a stop criterion is met. Having the search space defined, it is possible to use general purpose search methods. We will use three different search techniques: evolutionary algorithm, Monte Carlo search and greedy search.

2.5.1. EVOLUTIONARY ALGORITHM

Evolutionary algorithm (EA) is a method inspired by principles of the natural evolution. EA is powerful global search method that many times was successfully applied to various instances if NPC problems (e.g. travelling salesman problem [10]). EA maintains a population of encoded solutions (points from the search space). In each iteration, the population is processed by the selection and genetic operators (see [1] for detailed description).

In our approach, the mutation will be defined as the transition to a random neighbouring node. We plan to use the nonelitist, tournament selection of size two. The population size will initially be set to 20 and the number of generations to 200. The best individual from the last generation will be returned as the result.

2.5.2. MONTE CARLO SEARCH

In the Monte Carlo search a random walk towards more specific patterns will be performed. The probability of pattern concretisation will decrease in proportion to the distance of a current point to the point that represents an empty pattern. The number of walks will be equal to the product of the number of generations and the population size from evolutionary algorithm. A pattern with the best score according to the training set will be returned as a result. Such simple algorithm is one of the global search methods and is able to produce good results in certain conditions. It is also good baseline to compare with the evolutionary algorithm.

2.5.3. GREEDY SEARCH

Greedy search is a fast local search method. The algorithm maintains a single working point from the space of patterns. In each iteration, a new working point is selected which maximizes the scoring function in the neighbourhood of the current working point. The algorithm stops if there is no better point in the neighbourhood of the current working point.

2.6. STARTING POINT

The computation time and the final result of the algorithm depends on selection a starting point for search methods. Conceptually the easiest way is to start from the most general pattern, i.e. ‘x’. This approach guarantees that finding every important pattern is at least theoretically possible. Unfortunately this approach will dramatically slow down the computations. Because DNA alphabet size is only four, small patterns (e.g. “A”) will occur many times in exons and other functional elements. The same about pattern T, [AT], etc., even pairs or triplets will have no classification power. The more sophisticated methodology will be used. First, all training sequences will be aligned to the starting positions of exons. Then we will be able to compute some kind of histogram for exons. Having histogram allows us to detect anomalies like high peaks for specific nucleotides at specific positions. These anomalies may be biologically important places and so, good starting points for a search algorithm could be generated. To get more interesting starting points beside exons left alignment, we will also perform exons right alignment and introns left and right alignments.

3. GENERAL FRAMEWORK

It is not so easy to provide any algorithm that solves complicated problem, e.g. recognizing exons. In that case a good idea is to split the problem into smaller ones that will be easier to solve. Sometimes big problem could also be split to a set of already solved problems. The proposed approach does so – it splits the problem of exon recognition into a set of problems that have been already solved, and leaves two smaller problems how to define the neighbourhood in the search space and how to score elements from such defined space.

Various kinds of problems (signal detection, credit card assignment, robot movement, etc.) could be perceived as classification tasks. Having only a set of labelled examples (training set) allows us to use one from the set of off-the-shelf algorithms to build a classifier. A typical situation is that the features used to build a classifier have the form of attributes, which can be grouped into a vector. Unfortunately, if we analyse sequential data we do not have well-defined attributes so we should construct them.

The attribute definition is another big problem, but if we think of that task as a space search problem, all we need is to define the search space (neighbourhood relation), the scoring function, and to choose one of the search algorithms. It looks like building solutions from small building blocks. Most of those blocks could be exchanged by other with similar function. So we could try different combinations of space, different algorithms and scoring functions. That framework was already used by the authors to search for the best classifier [2].

Every algorithm searches some space but in most cases it is hidden somewhere inside algorithm and we do not have ability "to play" with different search methods, with the neighbourhood relation etc. When something is hidden we are not able to optimize it.

4. CONCLUSIONS

The assumption of a new algorithm (KIS) for recognizing exons in eukaryotes was presented. It brings a mixture of machine learning and optimization into the bioinformatics. Using very successful concepts of a classifier and global search methods gives a possibility to overtake existing approaches based only on elementary statistics. Deriving general framework from KIS allows us to simply switch to other problems in bioinformatics and far beyond.

REFERENCES

[1] ARABAS J., *Lecture notes on evolutionary computation (in Polish),* WNT, Warszawa, 2001.

[2] BIEDRZYCKI R., ARABAS J., *A search based view of decision trees induction,* in proc. of Artificial Intelligence Studies, vol. 3, (26)2006, pp 119-127.

[3] BURGE C.B., KARLIN S., *Finding the genes in genomic DNA,* Curr. Opin. Struct. Biol., Vol. 8 (1998), No. 3, pp. 346-354.

[4] ELSIK C.G., WORLEY K.C., ZHANG L., MILSHINA N.V., JIANG H., REESE J.T., CHILDS K.L., VENKATRAMAN A., DICKENS M.C., WEINSTOCK G.M., GIBBS R.A., *Community annotation: procedures, protocols, and supporting tools*, Genome Res, 2006.

[5] GUHATHAKURTA D., *Computational identification of transcriptional regulatory elements in DNA sequence*, Nucleic Acids Research, Vol. 34 (2006).

[6] KRAWIEC K., *Constructive induction in learning of image representation*, Research Report RA-006/2000, Poznań, Poland 2000.

[7] QUNLAN J.R., *Induction of decision trees*, Machine Learning, 1:81–106.

[8] ROSS B.J., *The evolution of stochastic regular motifs for protein sequences,* New Generation Computing, Vol. 20 (2002), No. 2, pp. 187-213.

[9] SAEYS Y., *Feature selection for classification of nucleic acid sequences*, PhD thesis (2004), Ghent University, Belgium.

[10] TAO G., MICHALEWICZ Z., *Inver-over operator for the TSP*, PPSN, 1998: 803-812