Inference of prototypical genetic sequences

Wit Forys*, Barbara Barabasz**
*Jagiellonian University
Institute of Computer Science
*AGH University of Science and Technology
Department of Computer Methods in Metallurgy

Introduction

We present a model of generation genetic sequences which is based on Informed Parsimonious inference model introduced in [1] and, as we hope, improves description or estimation of sets of data under consideration. The IP model is used in biology to infer classes of genetic sequences and to classify them. Informed Parsimonious inference model uses three algorithms:

1. algorithm *CREATOR*
2. algorith *NATURE*
3. algorithm *MASC* (Multiple Aligned Sequence Classification)

The final result of these algorithms is a model which is optimal (at least locally) according to the introduced complexity measure. The main goal of the paper is a modification of the *CREATOR* and *NATURE* algorithms. We modify these algorithms because the original formulation is based on informations about probabilistic properties of genetic sequences. These properties are derived just from a database and taking into account a size of any biological database it is not easy to derive properly these properties. The idea of the modifications is based on theory of grammars and formal languages, namely on problems of inferring grammars.

Model of sequence generation

In the sequel we consider a sample of words as an $m-$tuple $S = (S_1, ..., S_m)$ where words (aligned sequences) $S_1, ..., S_m$ are over a finite alphabet Γ . In the case of DNA sequences the alphabet $\Gamma = \{A, G, C, T\}$ represents the four nucleocides that forms DNA sequences. In the case of proteins the alphabet consists of twenty letters representing amino acids.

A pool of prototypes θ is a $k-$tuple $(\theta_1, ..., \theta_k)$ where each θ_i is a word over the alphabet Γ. A prototype is used to represent a common ancestor having common general properties of the words from the particular class. The properties covers frequencies of letters from Γ that appear in sequences of the

considered class of a database and structures of them, as well. These is achieved by modifying the approach of [1].

Below we present algorithms that produce a pool of prototypes and a sample of words.

In the paper MHJ an algorithm CREATOR was presented. The algorithm produces a pool of prototypes and is based on probabilistic properties of genetic sequences in a considered database. Namely, it uses probability distributions - P^A and P^B_j. The distribution P^A gives an information how many prototypes we should to derive. The probability distributions P^B_j for $j = 1, ..., k$ govern the choice of the letters in prototypes.

Algorithm *CREATOR*

1. pick according to P^A the number of prototypes k from $\{1, ..., m\}$
2. for $j := 1, ..., k$
   for $l := 1, ..., n$
   pick according to $P^B_{j,l}$ the $l-$th letter of the $j-$th prototype $\theta_{j,l}$ from Γ
   $\theta_j := (\theta_{j,1}, ..., \theta_{j,n})$
3. $\theta = (\theta_1, ..., \theta_k)$
4. return (k, θ)

Note that it is assumed by MHJ that P^A is a uniform distribution. It means that the authors, in fact, have no information about a structure of genetic sequences of the database. For them it is equally probable that there exists only one prototypical sequence as there exist m such sequences. We modify the algorithm to involve in it a probable structure of genetic sequences typical for a sample picked up from the database. It is done accordigly to the following lines.

Algorithm *CREATOR - modified*

1. pick randomly from the database a sample $s = (s_1, ..., s_m)$ of genetic sequences, that is words in Γ^*
2. infer a grammar G such that any sequence s_i is in $L(G)$ for $i = 1, ..., m$
3. analize all derivative trees T_G of the grammar G and establish a set of basic subtrees T_S - set $k = \#T_S$
4. form fronts θ_i for $i = 1, ..., k$ of all basic subtrees in T_S fulfilling gaps by $\flat$ - an additional, not in alphabet Γ, letter.
5. return $(k, \theta_1, ..., \theta_k)$

obtained θ_i are prototypes of words produced by the algorithm NATURE in a classical meaning - representing the common ancestor of the sequences from a particular class and are also prototypical in the sense of containing a common structure of these sequences.

Remarks

1. The point 2 is realized using known algorithms which infere, for example, a context-free grammar from a finite sample of words, see [2]. We suggest using here linear grammars for which algorithms are of the low complexity [3].

2. The point 3 is realized as follows. On the set of derivative trees T_G of a grammar obtained in 2. define a relation $\sim_l$ which puts into an equivalence class two derivative trees if they are identical as labeled trees up to the $l-$th level. Denote by $k(l)$ the number of elements of $T_S = T_G/\sim_l$.

3. The point 4 is realized as follows. We take a representant of an equivalence class and all nonterminal leaf labels change into $\flat$ and leaving terminal leaf labels unchanged.

We associate with the algorithm CREATOR a probability $P^G(l) = 1 - (k(l) + 1)^{-1}$ for any fixed a priori number m of genetic sequences picked up from the database.

Presented in the paper MHJ an algorithm NATURE produces, having as an input a pool of prototypes $(k, \theta_1, ..., \theta_k)$, a sample S. Any obtained in this way sample is assumed to represent the database and is a subject of genetic research.

Algorithm *NATURE*

1. pick the index $c(i)$ from the set $\{1, ..., k\}$ and fix the prototype $\theta_{c(i)}$.

2. for $l := 1, ..., n$
   pick for the position l in $\theta_{c(i)}$ the mutation flag p from $\{noisy, not-noisy\}$;
   if $p = noisy$ then
   pick the replacement letter $s_{c(i),l}$ for the $l-$th position in $\theta_{c(i)}$ from $\Gamma \setminus \{\theta_{c(i),l}\}$
   else
   $s_{c(i),l} := \theta_{c(i),l}$;
   $S_i := (s_{i,1}, ..., s_{i,n})$

3. return S_i

It is assumed that the step number 1. in NATURE is done according to a probability distribution P^C. The step 3. is executed according to probability distributions P^D and P^E. In MHJ paper a distribution P^D is described as a coin flipping for each position in the considered prototype. In the presented below algorithm *NATURE - modified* all symbols $\flat$ are replaced by a letter from Γ according to a distribution P^E (similarly as in NATURE) but the fact that a $\flat$ occurs is implied by a structure of the considered database.

The modification which have been done in the algorithm *CREATOR - modified* allow us to resign of a probability distribution P^D.

Algorithm *NATURE - modified*

1. pick the index $c(i)$ from the set $\{1, ..., k\}$ and fix the prototype $\theta_{c(i)}$
2. if $\theta_{c(i),l} = \flat$ then replace $\flat$ by a letter from Γ
   else
   $\theta_{c(i),l}$ remains unchanged
3. $S_i = s_{i,1}, ..., s_{i,n}$
4. return S_i

The algorithm *NATURE - modified* produces, as the result, a sample of genetic sequences which are based on ancestral prototypes taken from the pool of prototypes given by algorithm *CREATOR - modified*. Executing the algorithm *NATURE - modified* we obtain a set of samples $S(m, n, \Gamma)$.

By a model M we understand the the following data:

1. a number of prototypes k and a pool of prototypes $(\theta_1, ..., \theta_k)$
2. set of samples $S(m, n, \Gamma)$ and all probability distributions associated to random steps of *NATURE - modified*.

$M(m, n, \Gamma)$ denotes a family of all such models for a fixed database of genetic sequences.

Models valuation

There are possible two extreme cases. The first when a model has m prototypes and the sequences of the sample are identical to these prototypes. This model of course does eliminate noise (randomnest) from the observation however may be too complex for analysing genetic data. The second extreme possiblity is a model which is based on a single prototype sequence. This model is of course the simplest possible and of the low complexity but probably its genetic sequences may differ greatly from the prototype. Hence a tradeoff occurs between the complexity of the inferred model and its accuracy in describing the genetic data.

A valuation of the complexity of produced models $M(m, n, \Gamma)$ is according to the lines presented in [1]. There is introduced a valuation function $I(S, M)$ defined on a sample and a model. A general idea is that to describe the whole sample we have to describe the model first and then to describe the sample given the model. Hence $I(S, M)$ is the following sum

$$I(S, M) = I(M) + I(S \mid M).$$

Let $P(M)$ denote a probability of the model M computed below

$$P(M) = P^G(l) \prod_{j=1}^{k(l)} \prod_{i=1}^{n} P^B_{j,i}(\theta_{j,i}) P^{[C,E]}$$

where $P^G(l)$ denotes probability connected with the number of prototypes in a fixed grammar G. $P^B_{j,l}(\theta_{j,i})$ is equall 1 for all $\theta_{j,i} \in \Gamma$, that is for all letters derived from derivative trees and depends on a given in advance probability distribution P^E on signs $\flat$ fulfiling all the gaps in these derivative trees. $P^{[C,E]}$ is a probability that the distributions P^C, P^E govern the random steps in *NATURE-modified*.

Now we put

$$I(M) = -logP(T) = -log(P^G(l)\prod_{j=1}^{k(l)}\prod_{i=1}^{n} P^B_{j,i}(\theta_{j,i})P^{[C,E]})$$

In a similar way we compute $P(S \mid M)$ and put $I(S \mid M) = -logP(S \mid M)$. Skipping here all computations we come to the following formula for $P(S \mid M) = \prod_{j=1}^{k(l)} P(S_j \mid M)$ and for $I(S \mid M) = -log\prod_{j=1}^{k(l)} P(S_j \mid M)$. The last formula can be converted to the following one

Finally

$$I(S \mid M) = \sum_{j=1}^{k(l)} -logP^C(t(j)) + \sum_{j=1}^{k(l)}\sum_{\flat} -logP(S_{j,\flat} \mid S_j \; derived \; from \; \theta_{t(j)})$$

So finally we can consider the valuation function as given by

$I(M) + I(S \mid M) = -log(P^G(l)\prod_{j=1}^{k(l)}\prod_{i=1}^{n} P^B_{j,i}(\theta_{j,i})P^{[C,E]}) +$
$\sum_{j=1}^{k(l)} -logP^C(t(j)) + \sum_{j=1}^{k(l)}\sum_{\flat} -logP^E_{t(j),\flat}(S_{j,\flat})$

Optimal model

Optimal model according to the valuation function is obtained using the algorithm *MASC* - Multiple Aligned Sequence Classification exactly in the way described in [1].

References

[1] A.Milosavljevic, D.Haussler, J.Jurka, Informed parsimonious inference of prototypical genetic sequences, Proceedings of 2-nd Workshop on Computational Learning Theory, Santa Cruz, US, 1989, pp.102-117

[2] R.C.Gonzalez, M.G.Thomason, Syntactic Pattern recognition, Addison-Wesley Publishing Company, 1978

[3] M.Forys, Inference of linear grammars, preprint Discrete Mathematics, IIUJ, 2005

[4] A.N.Kolmogorov, Three approaches to the quantitative definition of information, International Journal for Computer Mathematics, vol.2, 1968, pp. 157-168