IL NUOVO CIMENTO Vol. 35 C, N. 5 Suppl. 1 Settembre-Ottobre 2012
DOI 10.1393/ncc/i2012-11330-6

Colloquia: PR PS BB 2011

Large-scale annotation of proteins with labelling methods

R. Casadio([1])([2])([*]), P. L. Martelli([1])([2]), C. Savojardo([1])([3])
and P. Fariselli([1])([3])

([1]) *Biocomputing Group, University of Bologna - Via San Giacomo 9/2, Bologna, Italy*
([2]) *Department of Biology, University of Bologna - Bologna, Italy*
([3]) *Department of Computer Science, University of Bologna - Bologna, Italy*

ricevuto il 30 Settembre 2011; approvato l' 1 Dicembre 2011

Summary. — We revise a major important problem in bioinformatics: how to annotate protein sequences in the genomic era and all the solutions that have been described by implementing tools based on labelling methods. In this paper we mainly focus on our own work and the theoretical methods that are popular in the field of biosequence analysis in modern molecular biology. We will also review a recent application from our group that largely improves on the topology prediction of disulfide bonds in proteins from Eukaryotic organisms.

PACS 87.15.B- – Structure of biomolecules.
PACS 87.18.Xr – Proteomics.
PACS 07.05.Mh – Neural networks, fuzzy logic, artificial intelligence.

1. – Introduction

As a result of large sequencing projects, data banks of protein sequences and structures are growing rapidly. The number of sequences is however orders of magnitude larger than the number of structures known at atomic level and this is so in spite of the efforts in accelerating processes aiming at the resolution of protein structure. Tools have been developed in order to bridge the gap between sequence and protein 3D structure, based on the notion that information is to be retrieved from the data bases and that knowledge-based methods can help in approaching a solution of the protein folding problem. By this several features can be predicted starting from a protein sequence such as structural and functional motifs and domains, including the topological organisation of a protein inside the membrane bilayer, and the formation of disulfide bonds in a folded protein structure. Our group has been contributing to the field with different computational tools, mainly based on machine learning (neural networks (NNs), hidden markov models

([*]) E-mail: casadio@biocomp.unibo.it

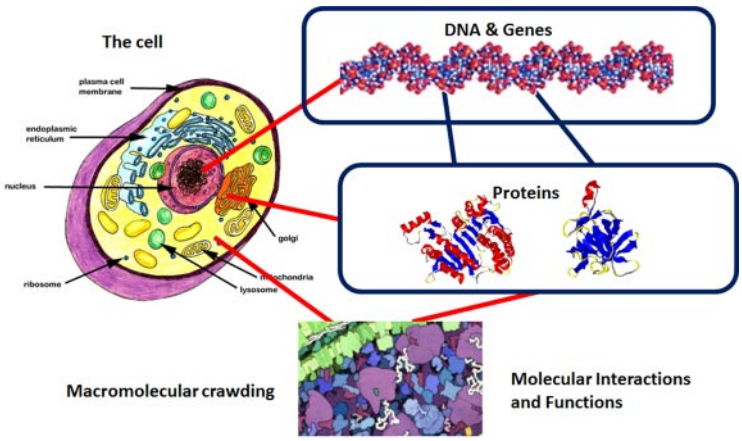

Fig. 1. – The ingredients of biological complexity in the cell. From genes to proteins and their interaction for biological processes.

(HMMs), support vector machines (SVMs), hidden neural networks (HNNs) and extreme learning machines (ELMs)). The methods routinely compute the likelihood of a given feature starting from the protein sequence and/or structure (`www.biocomp.unibo.it`). Our methods can add to the process of large scale genome and proteome annotation (endowing sequences with functional and structural features) [1, 2].

Recently, Conditional Random Fields (CRFs) have been introduced as a new promising framework to solve sequence labelling problems in that they offer several advantages over Hidden Markov Models (HMMs), including the ability of relaxing strong independence assumptions made in HMMs [3, 4]. However, several problems of sequence analysis can be successfully addressed only by designing a grammar in order to provide meaningful results. We therefore introduced Grammatical-Restrained Hidden Conditional Random Fields (GRHCRFs) as an extension of Hidden Conditional Random Fields (HCRFs). GRHCRFs while preserving the discriminative character of HCRFs, can assign labels in agreement with the production rules of a defined grammar [5]. The main GRHCRF novelty is the possibility of including in HCRFs prior knowledge of the problem by means of a defined grammar. Our current implementation allows regular grammar rules. We tested our GRHCRF on two typical biosequence labelling problem: the prediction of the topology of Prokaryotic outer-membrane proteins and the prediction of bonding states of cysteine residues in proteins [6, 7], proving that the separation of state names and labels allows to model a huge number of concurring paths compatible with the grammar and with the experimental labels without increasing the time and space computational complexity.

2. – The genomic era and the problem of protein sequence annotation

A general simplified scheme of the intrinsic complexity of an eukariotic cell is provided in fig. 1. The cell dynamics allows a constant amount of chemical information being transferred from the nucleus where the genetic material is localised to the ribosomes where protein synthesis is completed by mRNA (messenger RNA) translation and protein generation. We know that genes in the chromosomes code for specific proteins and that

eventually proteins by interacting in the different compartments can perform all the different biological processes that are at the basis of the cell life cycle. This general framework can be investigated by reducing the complexity of the cell space into specific subspaces where to address some of the different steps whose integration is sketched in fig. 1. Summing up, to our discussion the following steps are relevant:

1. Genes codes for proteins

2. Proteins are responsible of all the biological functions occurring in the cell

The genomic era is characterised by an increasing effort in sequencing all the genomic content characterising the different species and more recently also the variability of the genetic content in relation to the whole human population, with the aim of deciphering at a molecular level also possible molecular determinants related to maladies. Thanks to recent technological advancement DNA is sequenced at an unprecedented rate. More than 4 Terabytes of data are weekly produced and stored in specific data bases. Therefore Bioinformatics deals first of all with repositories, their link and informational retrieval in a rational way. One problem is at hand: how to cope with the enormous gap among DNA sequences and the correspondent proteins whose structure and function is important for the cell (`www.ncbi.nlm.nih.gov`).

Nowadays about 20 millions of protein sequences derived from DNA sequencing of over 3000 organisms are stored in UniProtKB, the reference data base of protein sequences freely available at `www.uniprot.org`. The enormous amount of stored sequences is simply derived by electronically translating information stored in genes, after they have been recognised in the DNA sequence. However most of the proteins (some 90%) still deserve experimental validation and their existence is deduced on the basis of sequence similarity to other proteins or predicted.

Some 80000 proteins have been solved with atomic resolution and can be found in the Protein Data Bank (PDB, `www.rcsb.org`). All the chemico-physical details relating structure to function can be statistically derived from protein three dimensional structure. These features are used to train algorithms that eventually generalise on a selected training set (non-redundant set of examples) and can then infer with some reliability the same type of property on a never seen before testing set. The set of atomic solved protein structures, together with the biochemical properties describing their molecular function, the biological process they participate into and eventually the cellular localisation or sub-compartments where they are active constitute the basic knowledge required to extract information for the annotation process.

With the information derived by means of different methods from the well characterised proteins we can recognise with computational methods proteins that are likely to exist and function on the basis of what we learn from biology and our understanding of evolution. The inference process endowing with functional and structural properties any new sequence on the basis of previous knowledge is known as the annotation process.

In the following we will focus on which computational methods are the state of art in annotating protein sequences based on labelling techniques and some of our applications in this field:

– Hidden Markov Models (HMMs),

– Grammatical Restrained Hidden Conditional Random Fields (GRHCRFs).

TABLE I. – *HMM vs. CRF/HCRF vs. GRHCRF.*

HMM	CRF/HCRF	GRHCRF
Model joint probability $p(x, y)$	Model conditional probability $p(x\|y)$	Model conditional probability $p(x\|y)$
Parameters are transition and emission probabilities	Parameters are weights associated to feature functions	Parameters are weights associated to feature functions
Per state normalization	Global normalization	Global normalization
Emission probabilities describe a single position in the sequence	Feature states can describe multiple positions in a sequence	Feature states can describe multiple positions in a sequence
Decoupling between states and labels	States and labels coincide	Decoupling between states and labels

HMM = Hidden Markov Models, CRF/HCRF = Conditional Random Fields /Hidden CRF;
GRHCRF = Grammatical Restrained Hidden Random Fields.

These two methods that have been largely adopted for protein sequence analysis are compared in table I, where their characteristics are compared and their differences highlighted in a schematic way.

3. – Hidden Markov Models

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (*hidden*) states. Here we are considering only the discrete version of HMMs. In what follows we report a detailed description of the main HMM algorithms.

HMMs are charaterized by states whose links define a regular (*statistical*) grammar and an alphabet of symbols that can be emitted from each states. These two entities define the HMM parameters that are a set of transition probabilities and a set of emission probabilities

Here we make use of an explicit *BEGIN* state to model the starting probability, while the end state (states) is (are) problem dependent and treated as general.

An observed sequence of length L is indicated as O $(= O_1 \dots O_L)$ or sometimes x $(=x_1 \dots x_L)$, both for a single-symbol-sequence (as in the standard HMMs) or for a vector-sequence as described before [8]. $label(s)$ indicates the label associated to the state s, while $\Lambda(= \Lambda_i, \dots \Lambda_L)$ is the list of the labels associated to each sequence position i obtained after the application of a decoding algorithm. A HMM consisting of N states is therefore defined by the three probability distributions

Starting probabilities:

$$a_{BEGIN,k} = P(k|BEGIN), \tag{1}$$

Transition probabilities:

$$a_{s,k} = P(k|s), \tag{2}$$

Emission probabilities:

$$e_k(O_i) = P(O_i|s). \tag{3}$$

Then we use the following notation for the forward probability

$$f_k(i) = P(O_1, O_2 \dots O_i, \pi_i = k), \tag{4}$$

which is the probability of having emitted the first partial sequence up to i ending at the state k and backward probability

$$b_k(i) = P(O_{i+1}, \dots O_{L-1}, O_L|\pi_i = k), \tag{5}$$

which is the probability of having emitted the sequence starting from the last element back to the $i+1$ element given that we end at the position i into the state k.

There are three main problems that we want to solve for the HMMs:

1. Given a HMM model M and a sequence of observations O, find $P(O|M)$, the probability that the observed sequence O is generated by the model M.
2. Given a HMM model M and an observation sequence O, find an optimal *state sequence* for the underlying Markov process. In other words, we want to uncover the hidden part of the Hidden Markov Model.
3. Given a HMM model M and a set of observation sequences, train the model parameters (transition and emission probabilities).

4. – Problem 1: Computing the probability of a sequence given the model

The probability of emitting the whole sequence can be computed using either forward or backward as

$$P(O|M) = f_{END}(L+1) = b_{BEGIN}(0). \tag{6}$$

Forward and backward are also necessary for the updating of the HMM parameters. Using the Baum-Welch algorithm [9, 10]. Alternative gradient-based training algorithm can be applied [10]. In the following we indicate $I_x(s)$ and $O_x(s)$ the sets of states that are connected to state s ($a_{qs} \neq 0$) and from state s ($a_{sq} \neq 0$), respectively. The lower index x indicates the restriction to null (N), emitting (E) or all (S) states.

4˙1. *Implementation of forward algorithm.* – The algorithm is divided into three phases:

START. Start: probability of begin (B) $= 1, 0$ for the other states:

$$f_B(0) = 1.0, \quad \forall s \in E \quad f_s(0) = 0.$$

From Begin (B) $\rightarrow$ Null or silent (N) states

$$\forall s \in N \qquad f_s(0) = a_{B,s}.$$

From Null states $(N) \rightarrow$ Null states (N)

$$\begin{aligned} &\forall s \in N, \\ &\forall t \in I_N(s), \\ &f_s(0) = a_{B,s} + \sum_{t \in I_N(s)} f_t(0) a_{t,s}. \end{aligned}$$

To reduce numerical errors (and underflow), it is possible to introduce a numerical rescaling by defining a vector Scale as:

$$\begin{aligned} &\forall s \in S, \\ &Scale(0) = \sum_{s \in S} f_s(0), \\ &f_s(0) = f_s(0)/Scale(0). \end{aligned}$$

Recurrence. For all states S from position 1 to L (sequence length)

All states $(S) \rightarrow$ only emitting states (E)

$$\begin{aligned} &\forall s \in E, \\ &f_s(i) = e_s(x_i) \sum_{t \in I_S(s)} a_{t,s}\, f_t(i-1). \end{aligned}$$

It is worth noticing that if there are labels the previous equations become

$$\begin{aligned} &\forall s \in E, \\ &f_s(i) = \begin{cases} \sum_{t \in I_S(s)} a_{t,s}\, f_t(i-1)\, e_s(x_i) & label(s) = label(x_i), \\ 0, & label(s) \neq label(x_i). \end{cases} \end{aligned}$$

Emitting states $(E) \rightarrow$ Null states (N)

$$\begin{aligned} &\forall s \in N, \\ &f_s(i) = \sum_{t \in I_E(s)} a_{t,s}\, f_t(i). \end{aligned}$$

Null states $(N) \rightarrow$ null states (N) (please remember that the Null states are topologically sorted)

$$\begin{aligned} &\forall s \in N, \\ &f_s(i) = f_s(i) + \sum_{t \in I_N(s)} a_{t,s}\, f_t(i). \end{aligned}$$

Again if Scale is defined:

$$\begin{aligned} &\forall s \in S, \\ &Scale(i) = \sum_{s \in S} f_s(i), \\ &f_s(i) = f_s(i)/Scale(i). \end{aligned}$$

END. Only the states allowed to be end states are to be considered. (at least 1 exists). If $END = \{s \in S | s \text{ is an end state}\}$ we have

$$\sigma = \sum_{s \in END} f_s(L).$$

If Scale is not defined we have that $\sigma = P(x|HMM)$, otherwise the probability of the sequence is

$$P(x|HMM) = \sigma \cdot \prod_{i=0}^{L} Scale(i)$$

4˙2. *Implementation Backward Algorithm.* – As for the case of forward we have three phases:

START. For each end states ($END\_S = END\_N + END\_E$, where S stands for all states N for null and E for emitting states) the end probability is one ($P(s) = 1$ for s in END, 0 for the others)

$$\begin{aligned} &b_t(L) = 0, \quad \forall t \notin ENDS, \\ &b_s(L) = 1.0, \quad \forall s \in END\_N, \\ &b_s(L) = 1.0, \quad \forall s \in END\_E \, AND \, label(s) = label(x_L). \end{aligned}$$

If Scale is defined

$$\begin{aligned} &b_t(L) = 0 \, t \notin ENDS \\ &b_s(L) = 1/\sigma, \quad \forall s \in END\_N \\ &b_s(L) = 1/\sigma, \quad \forall s \in END\_E \, AND \, label(s) = label(x_L). \end{aligned}$$

Remember that σ was computed using the forward.

If we call R_N the subsets of null states sorted in reversed order (reversed topological sort) we have

From Null states ← Null states

$$\begin{aligned} &\forall s \in R\_N \quad AND \, s \notin END\_N, \\ &\forall t \in O_N(s), \\ &b_s(L) = \sum_t b_t(L) a_{s,t}. \end{aligned}$$

From Emitting ← Null

$$\begin{aligned} &\forall s \in E \quad AND \, s \notin END\_E, \\ &\forall t \in O_N(s), \\ &b_s(L) = \sum_t b_t(L) a_{s,t}. \end{aligned}$$

If Scale is defined

$$\forall s \in S,$$
$$b_s(L) = b_s(L)/Scale(L),$$

Recurrence. For every state and for each sequence position i, from $L-1$ (L = sequence length) to 1 we have

From Null position $i \leftarrow$ Emitting position $i+1$

$$\forall s \in E,$$
$$b_s(i) = \sum_{t \in O_N(s)} a_{s,t}\, b_t(i+1)\, e_t(x_{i+1}).$$

In case of labeling we have

$$\forall s \in S,$$
$$b_s(i) = \begin{cases} \sum_{t \in O_E(s)} a_{s,t}\, b_t(i+1)\, e_t(x_{i+1}) & label(s) = label(x_i), \\ 0, & label(s) \neq label(x_i). \end{cases}$$

From Null $\leftarrow$ to Null

$$\forall s \in R\_N,$$
$$b_s(i) = \sum_{t \in O_N(s)} a_{s,t}\, b_t(i) + b_s(i).$$

From Emitting $\leftarrow$ Null

$$\forall s \in E,$$
$$b_s(i) = \sum_{t \in O_N(s)} a_{s,t}\, b_t(i) + b_s(i).$$

If the rescaling procedure is adopted the equations change accordingly as

$$\forall s \in S,$$
$$b_s(i) = b_s(i)/Scale(i)$$

END. From Null states $\leftarrow$ Emitting states

$$\forall s \in N,$$
$$b_s(0) = \sum_{t \in O_E(s)} a_{s,t}\, b_t(1).$$

From Null states $\leftarrow$ Null states

$$\forall s \in R\_N,$$
$$b_s(0) = \sum_{t \in O_N(s)} a_{s,t}\, b_t(0) + b_s(0),$$

and finally for the begin we have

$$b_B(0) = \sum_{t \in O_E(B)} a_{B,t}\, b_t(1)\, e_t(x_1) + \sum_{t \in O_N(B)} a_{B,t}\, b_t(0)$$

$b_B(0)$ = sequence probability if Scale is not used. On the contrary if Scale is used the last term is to be rescaled too

$$b_s(0) = b_s(0)/Scale(0)$$

and there should be $b_B(0) = 1.0$.

5. – Problem 2: Decoding algorithms

For the sake of clarity, for the decoding algorithms we do not report explicitly the case of null states that can be derived from the forward/backward algorithms by analogy.

Viterbi decoding. Viterbi decoding finds the path (π) through the model which has the maximal probability with respect to the others [9, 10] This means that we look for the path π^v which is

$$\pi^v = \text{argmax}_{\{\pi\}} P(\pi|O, M), \tag{7}$$

where $O(= O_1, \ldots O_L)$ is the observed sequence of length L and M is the trained HMM model. Since the $P(O|M)$ is independent of a particular path π, eq. (1) is equivalent to

$$\pi^v = \text{argmax}_{\{\pi\}} P(\pi, O|M) \tag{8}$$

and since $P(\pi, O|M)$ can be easily computed as

$$P(\pi, O|M) = \prod_{i=1}^{L} a_{\pi(i-1),\pi(i)} e_{\pi(i)}(O_i) \cdot a_{\pi(L),END} \tag{9}$$

using the Viterbi algorithm π^v is obtained as

– *Initialization*

$$v_{BEGIN}(0) = 1\, v_k(0) = 0 \quad \text{for} \quad k \neq BEGIN.$$

– *Recursion*

$$v_k(i) = [\max_{\{s\}}(v_s(i-1)a_{s,k})]e_k(O_i),$$
$$p_i(k) = \text{argmax}_{\{s\}} v_s(i-1)a_{s,k}.$$

– *Termination*

$$P(O, \pi^v|M) = \max_{\{s\}}[v_s(L)a_{s,END}],$$
$$\pi^v_L = \text{argmax}_{\{s\}}[v_s(L)a_{s,END}].$$

– *Traceback*

$$\pi^v_{i-1} = p_i(\pi^v_i) \quad \text{for} \quad i = L \dots 1.$$

– *Label assignment*

$$\Lambda_i = label(\pi^v_i) \quad \text{for} \quad i = 1 \dots L,$$

$v_k(i)$ is the probability of the most likely path ending at the state k after having observed the partial sequence $O_1, \dots O_i$. and $p_i(k)$ is the trace-back pointer.

5˙1. *Posterior and posterior sum decoding.* – The *posterior* decoding finds the path which maximizes the product of the *posterior* probability of the states [9,10]. Using the usual notation for forward ($f_k(i)$) and backward ($b_k(i)$) we have

$$P(\pi_i = k|O, M) = f_k(i)b_k(i)/P(O|M) \tag{10}$$

The path π^p which maximizes the posterior probability is then computed as

$$\pi^p_i = \text{argmax}_{\{s\}} P(\pi_i = s|O, M) \quad \text{for} \quad i = 1 \dots L \tag{11}$$

and the corresponding label assignment is

$$\Lambda_i = label(\pi^p_i) \quad \text{for} \quad i = 1 \dots L. \tag{12}$$

If we have more than one state sharing the same label, as it is usual the case, is sometimes more fruitful summing over the states that share the same label (*posterior sum*). In this way we can have a path through the model which maximizes the posterior probability of being in state with $label\lambda$ when emitting the observed sequence element O_i, or more formally

$$\Lambda_i = \text{argmax}_{\{\lambda\}} \sum_{label(s)=\lambda} P(\pi_i = s|O, M) \quad \text{for} \quad i = 1 \dots L. \tag{13}$$

The posterior-decoding drawback is that the state path sequences π^p or Λ may be not allowed paths. However this decoding can perform better than the Viterbi one, when more than one high probable path exits [9,10].

5˙2. *Posterior-Viterbi decoding.* – Posterior-Viterbi decoding is based on the combination of the Viterbi and posterior algorithms. After having computed the *posterior* probabilities we use a Viterbi algorithm to find the best *allowed posterior* path through the model. The basic PV idea is to compute the path π^{PV}

$$\pi^{PV} = \text{argmax}_{\{\pi \in A_p\}} \prod_{i=1}^{L} P(\pi_i|O, M), \tag{14}$$

where A_p is the set of the allowed paths through the model, and $P(\pi_i|O, M)$ is the *posterior* probability of the state assigned by the path π at position i (as computed in eq. (10)).

We then define an posterior probability of a path π as

$$P_a(\pi|O,M) = \prod_{i=1}^{L} \delta^*(\pi_{i-1},\pi_i) P(\pi_i|O,M), \tag{15}$$

where $\delta^*(s,t)$ is set to be 1 if $s \to t$ is an allowed transition of the model M, 0 otherwise. This guarantees that $P_a(\pi|O,M)$ is different from 0 only for allowed paths. Then we can now easily compute the best path π^{PV} using the Viterbi algorithm

– *Initialization*

$$v_{BEGIN}(0) = 1 \quad v_k(0) = 0 \quad \text{for} \quad k \neq BEGIN.$$

– *Recursion*

$$\begin{aligned} v_k(i) &= \max_{\{s\}}[v_s(i-1)\delta^*(s,k)]P(\pi_i = k|O,M), \\ p_i(k) &= \text{argmax}_{\{s\}}[v_s(i-1)\delta^*(s,k)]. \end{aligned}$$

– *Termination*

$$\begin{aligned} P(\pi^{PV}|M,O) &= max_s[v_s(L)\delta^*(s,END)], \\ \pi_L^{PV} &= \text{argmax}_{\{s\}}[v_s(L)\delta^*(s,END)]. \end{aligned}$$

– *Traceback*

$$\pi_{i-1}^{PV} = p_i(\pi_i^{PV}) \quad \text{for} \quad i = L \ldots 1.$$

– *Label assignment*

$$\Lambda_i = label(\pi_i^{PV}) \quad \text{for} \quad i = 1 \ldots L,$$

where $v_k(i)$ is the probability of the most probable *allowed-posterior* path ending to the state k after having observed the partial sequence $O_1, \ldots O_i$ and π_i is the trace-back pointer.

6. – Problem 3: HMM learning with Baum-Welch

If $E_k(c)$ is the number of time in which the symbol c is emitted in the state k, and with $A_{i,k}$ the number of time in which we count the transition from state i to state k, the parameter evaluation is then

$$\begin{aligned} a_{i,k} &= \frac{A_{i,k}}{\sum_l A_{i,l}}, \\ e_k(c) &= \frac{E_k(c)}{\sum_l E_k(l)}. \end{aligned}$$

$A_{i,k}$ and $E_k(c)$ are computed using the forward and backward [9,10]:

$$A_{i,k} = \sum_{p=1}^{N_p} \frac{1}{P(x^p)} \sum_{t=0}^{L_p-1} f_i(t)\, a_{i,k} e_k(x^p_{t+1}) b_k(t+1),$$
$$E_k(c) = \sum_{p=1}^{N_p} \frac{1}{P(x^p)} \sum_{x_t=c}^{L_p} f_k(t) b_k(t).$$

If we use the scaling factor we have

$$A_{i,k} = \sum_{p=1}^{N_p} \sum_{t=0}^{L_p-1} f_i(t)\, a_{i,k} e_k(x^p_{t+1}) b_k(t+1),$$
$$E_k(c) = \sum_{p=1}^{N_p} \sum_{x_t=c}^{L_p} f_k(t) b_k(t) Scale(t).$$

In the case we are using a vector emission approach [8] in which the emission is $eV_k(\overrightarrow{x}) = \langle \overrightarrow{e}_k, \overrightarrow{x} \rangle$ (instead of $e_k(c)$) the new updating equations are

$$A_{i,k} = \sum_{p=1}^{N_p} \frac{1}{P(x^p)} \sum_{t=0}^{L_p-1} f_i(t)\, a_{i,k} eV_k(x^p_{t+1}) b_k(t+1),$$
$$A_{i,k} = \sum_{p=1}^{N_p} \sum_{t=0}^{L_p-1} f_i(t)\, a_{i,k} eV_k(x^p_{t+1}) b_k(t+1).$$

And for the emissions

$$E_k(c) = \sum_{p=1}^{N_p} \frac{1}{P(x^p)} \sum_{t=1}^{L_p} f_k(t) b_k(t) x_t(c),$$
$$E_k(c) = \sum_{p=1}^{N_p} \sum_{t=1}^{L_p} f_k(t) b_k(t) Scale(t) x_t(c).$$

where $x_t(c)$ is the component c of the vector $\overrightarrow{x}_t$, representing the t-th sequence position

For the Null states we have

$$A_{i,k} = \sum_{p=1}^{N_p} \frac{1}{P(x^p)} \sum_{t=0}^{L_p} f_i(t)\, a_{i,k} b_k(t).$$

In the case of scaling procedure we have

$$A_{i,k} = \sum_{p=1}^{N_p} \sum_{t=0}^{L_p} f_i(t)\, a_{i,k} b_k(t) Scale(t).$$

7. – GRHCRF: Grammatical Restrained Hidden Conditional Random Fields

Here we define GRHCRF as an extension of a Hidden Conditional Random Fields and we provide the basic inference equations and we introduce a new decoding algorithm for CRF models.

In what follows $\mathbf{x}$ is the random variable over the data sequences to be labeled, $\mathbf{y}$ is the random variable over the corresponding label sequences and $\mathbf{s}$ is the random variable over the hidden states. We use an upper script index when we deal with multiple sequences. The problem that we want to model is then described by the observed sequences $\mathbf{x}^{(i)}$, by the labels $\mathbf{y}^{(i)}$ and by the underlying grammar G that is specified by its production rules with respect to the set of the hidden states. Although it is possible to imagine more complex models, in what follows we restrict each state to have only one possible associated label. Thus we define a function that maps each hidden state to a given label as

$$\Lambda(s) = y.$$

The difference between the CRF and GRHCRF (or HCRF) models is the fact that there is a layer that decouples the states from their lables. GRHCRF and HCRF are indistinguishable from their graphical structure representation since it depicts only the conditional dependence among the random variables. Since the number of the states $|\{\mathbf{s}\}|$ is always greater than the number of possible labels $|\{\mathbf{y}\}|$ the GRHCRFs (HCRFs) have more expressive power than the corresponding CRFs.

We further restrict our model to linear HCRF, so that the computational complexity of the inference algorithms remains linear with respect to the sequence length. This choice implies that the embedded grammar will be *regular*. Our implementation and tests are based on first-order HCRFs with explicit transition functions ($t_k(s_{j-1}, s_j, \mathbf{x})$) and state functions ($g_k(s_j, \mathbf{x})$) unrolled over each sequence position j. However, for the sake of clarity in the following we use the compact notation:

$$\sum_k \lambda_k f_k(s_{j-1}, s_j, \mathbf{x}) = \sum_l \lambda_l t_l(s_{j-1}, s_j, \mathbf{x}) + \sum_n \mu_n g_n(s_j, \mathbf{x}),$$

where $f_k(s_{j-1}, s_j, \mathbf{x})$ can be either a transition feature function $t_l(s_{j-1}, s_j, \mathbf{x})$ or a state feature function $g_n(s_j, \mathbf{x})$. Following the usual notation we extend the local functions to include the hidden states as

$$\psi_j(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \exp\left(\sum_k \lambda_k f_k(s_{j-1}, s_j, \mathbf{x})\right) \cdot \Gamma(s_{j-1}, s_j) \cdot \Omega(s_j, y_j) \tag{16}$$

and we set the two constraints as

$$\Gamma(s, s') = \begin{cases} 1, & \text{if}(s, s') \in G, \\ 0, & \text{otherwise}, \end{cases}$$

$$\Omega(s, y) = \begin{cases} 1, & \text{if}\Lambda(s) = y, \\ 0, & \text{otherwise}. \end{cases}$$

With this choice, the local function $\psi_j(\mathbf{s}, \mathbf{y}, \mathbf{x})$ becomes zero when the labeling $(\Omega(s_j, y_j))$ or the grammar production rules $(\Gamma(s, s'))$ are not allowed. In turn this sets to zero the corresponding probabilities. As in the case of the HCRF, for the whole sequence we define $\Psi(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \prod_j \psi_j(\mathbf{s}, \mathbf{y}, \mathbf{x})$ and the normalization factors (or partition functions) can be obtained summing over all possible sequences of hidden states (or latent variables):

$$Z(\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{s}} \Psi(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \sum_{\mathbf{s}} \prod_j \psi_j(\mathbf{s}, \mathbf{y}, \mathbf{x}),$$

or summing over all possible sequences of labels and hidden states:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \sum_{\mathbf{s}} \Psi(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \sum_{\mathbf{y}} Z(\mathbf{y}, \mathbf{x}).$$

Using the normalization factors the joint probability of a label sequence $\mathbf{y}$ and an hidden state sequence $\mathbf{s}$ given an observation sequence $\mathbf{x}$ is

$$p(\mathbf{y}, \mathbf{s}|\mathbf{x}) = \frac{\Psi(\mathbf{s}, \mathbf{y}, \mathbf{x})}{Z(\mathbf{x})}.$$

The probability of an hidden state sequence given a label sequence and an observation sequence is

$$p(\mathbf{s}|\mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{s}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})} = \frac{\Psi(\mathbf{s}, \mathbf{y}, \mathbf{x})}{Z(\mathbf{y}, \mathbf{x})}.$$

Finally, the probability of a label sequence given an observation sequence can be computed as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{Z(\mathbf{y}, \mathbf{x})}{Z(\mathbf{x})}.$$

7˙1. *Parameter estimation.* – As for the HMM one relevant problem is the parameter training. The model parameters (Θ) can be obtained by maximizing the log-likelihood

of the data:

$$\begin{aligned}\mathcal{L}(\Theta) &= \log \prod_{i=1}^{N} p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)};\Theta) \\ &= \log \prod_{i=1}^{N} \frac{Z(\mathbf{y}^{(i)},\mathbf{x}^{(i)})}{Z(\mathbf{x}^{(i)})} \\ &= \sum_{i=1}^{N} \log Z(\mathbf{y}^{(i)},\mathbf{x}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}),\end{aligned}$$

where the different sequences are supposed to be independent and identically distributed random variables. Taking the first derivative with respect to parameter λ_k of the objective function we obtain:

$$\begin{aligned}\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} = &\underbrace{\sum_{i=1}^{N} \frac{\partial}{\partial \lambda_k} \log Z(\mathbf{y}^{(i)},\mathbf{x}^{(i)})}_{\mathcal{C}} \\ &\underbrace{- \sum_{i=1}^{N} \frac{\partial}{\partial \lambda_k} \log Z(\mathbf{x}^{(i)})}_{\mathcal{F}},\end{aligned}$$

where, in analogy with the Boltzmann machines and HMMs for labelled sequences, $\mathcal{C}$ and $\mathcal{F}$ can be seen as *clamped* and *free* phases. After simple computations we can rewrite the derivative as

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} = E_{p(\mathbf{s}|\mathbf{y},\mathbf{x})}[f_k] - E_{p(\mathbf{s},\mathbf{y}|\mathbf{x})}[f_k],$$

where the $E_{p(\mathbf{s}|\mathbf{y},\mathbf{x})}[f_k]$ and $E_{p(\mathbf{s},\mathbf{y}|\mathbf{x})}[f_k]$ are the expected values of the feature function f_k computed in the clamped and free phases, respectively. Differently from the standard CRF, both expectations have to be computed using the Forward and Backward algorithms. These algorithms must take into consideration the grammar restraints.

To avoid overfitting, we regularize the objective function using a Gaussian prior, so that the function to maximize has the form of

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \log Z(\mathbf{y}^{(i)},\mathbf{x}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) - \sum_{k} \frac{{\lambda_k}^2}{2\sigma^2}$$

and the corresponding gradient is

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} = E_{p(\mathbf{s}|\mathbf{y},\mathbf{x})}[f_k] - E_{p(\mathbf{s},\mathbf{y}|\mathbf{x})}[f_k] - \frac{\lambda_k}{\sigma^2}.$$

7˙2. *Computing the expectations*. – The partition functions and the expectations can be computed using the dynamic programming by defining the so called forward and backward algorithms [9, 10]. For the clamped phase the forward algorithm is

$$\begin{aligned}\alpha_0(\mathbf{BEGIN}|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= 1,\\ \alpha_0(s|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= 0, \forall s \in S \setminus \{\mathbf{BEGIN}\},\\ \alpha_j(s|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= \sum_{s'\in S} \alpha_{j-1}(s'|\mathbf{y}^{(i)},\mathbf{x}^{(i)})\\ &\quad \cdot M_C(s',s,j),\end{aligned}$$

where the clamped phase matrix M_C takes into account both the grammar constraint ($\Gamma(s',s)$) and the current given labeling $\mathbf{y}$ ($\Omega(s,y_j^{(i)})$).

$$\begin{aligned}M_C(s',s,j) = \exp\left(\sum_k \lambda_k f_k(s_{j-1}=s', s_j = s, \mathbf{x}^{(i)})\right)\\ \cdot\Gamma(s',s)\cdot\Omega(s,y_j^{(i)}).\end{aligned}$$

The forward algorithm for the free phase is computed as

$$\begin{aligned}\alpha_0(\mathbf{BEGIN}|\mathbf{x}^{(i)}) &= 1,\\ \alpha_0(s|\mathbf{x}^{(i)}) &= 0, \forall s \in S \setminus \{\mathbf{BEGIN}\},\\ \alpha_j(s|\mathbf{x}^{(i)}) &= \sum_{s'\in S} \alpha_{j-1}(s'|\mathbf{x}^{(i)}) M_F(s',s,j),\end{aligned}$$

where the free phase matrix M_F is defined as

$$\begin{aligned}M_F(s',s,j) = \exp\left(\sum_k \lambda_k f_k(s_{j-1}=s', s_j = s, \mathbf{x}^{(i)})\right)\\ \cdot\Gamma(s',s).\end{aligned}$$

It should be noted that also in the free phase the algorithm has to take into account the grammar production rules $\Gamma(s',s)$ and only the paths that are in agreement with the grammar are counted. Analogously, the backward algorithms can be computed for the clamped phase as

$$\begin{aligned}\beta_{L^{(i)}+1}(\mathbf{END}|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= 1,\\ \beta_{L^{(i)}+1}(s|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= 0, \forall s \in S \setminus \{\mathbf{END}\},\\ \beta_j(s|\mathbf{y}^{(i)},\mathbf{x}^{(i)}) &= \sum_{s'\in S} \beta_{j+1}(s'|\mathbf{y}^{(i)},\mathbf{x}^{(i)})\\ &\quad \cdot M_C(s,s',j),\end{aligned}$$

where $L^{(i)}$ is the length of the i^{th} protein. For the free phase we have

$$\begin{aligned}\beta_{L^{(i)}+1}(\mathbf{END}|\mathbf{x}^{(i)}) &= 1\\ \beta_{L^{(i)}+1}(s|\mathbf{x}^{(i)}) &= 0, \forall s \in S \setminus \{\mathbf{END}\}\\ \beta_j(s|\mathbf{x}^{(i)}) &= \sum_{s' \in S} \beta_{j+1}(s'|\mathbf{x}^{(i)}) M_F(s, s', j).\end{aligned}$$

The expectations of the feature functions ($E_{p(\mathbf{s}|\mathbf{y},\mathbf{x})}[f_k]$, $E_{p(\mathbf{s},\mathbf{y}|\mathbf{x})}[f_k]$) are computed as

$$\begin{aligned}E_{p(\mathbf{s}|\mathbf{y},\mathbf{x})}[f_k] = &\sum_{i=1}^{N} \sum_{j=1}^{L^{(i)}+1} \sum_{s',s \in S} f_k(s_{j-1} = s', s_j = s, \mathbf{x}^{(i)})\\ &\cdot \frac{\alpha_{j-1}(s'|\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) M_C(s', s, j) \beta_j(s|\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}{Z(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}.\\ E_{p(\mathbf{s},\mathbf{y}|\mathbf{x})}[f_k] = &\sum_{i=1}^{N} \sum_{j=1}^{L^{(i)}+1} \sum_{s',s \in S} f_k(s_{j-1} = s', s_j = s, \mathbf{x}^{(i)})\\ &\cdot \frac{\alpha_{j-1}(s'|\mathbf{x}^{(i)}) M_F(s', s, j) \beta_j(s|\mathbf{x}^{(i)})}{Z(\mathbf{x}^{(i)})}.\end{aligned}$$

The partition functions can be computed using both forward or backward algorithms as

$$\begin{aligned}Z(\mathbf{y}, \mathbf{x}) &= \alpha_{L+1}(\mathbf{END}|\mathbf{y}, \mathbf{x}) = \beta_0(\mathbf{BEGIN}|\mathbf{y}, \mathbf{x}),\\ Z(\mathbf{x}) &= \alpha_{L+1}(\mathbf{END}|\mathbf{x}) = \beta_0(\mathbf{BEGIN}|\mathbf{x}),\end{aligned}$$

where for simplicity we dropped out the sequence upper-script ((i)).

7˙3. *Decoding.* – Decoding is the task of assigning labels ($\mathbf{y}$) to an unknown observation sequence $\mathbf{x}$. Viterbi algorithm is routinely applied as decoding for the CRFs, since it finds the most probable path of an observation sequence given a CRF model. Viterbi algorithm is particular effective when there is a single strong highly probable path, while when several paths compete (have similar probabilities), *posterior decoding* may perform significantly better. However, the selected state path of the posterior decoding may not be allowed by the grammar. A simple solution of this problem is provided by the posterior-Viterbi decoding, that was previously introduced for HMMs. Posterior-Viterbi, exploits the posterior probabilities and at the same time preserves the grammatical constraint. This algorithm consists of three steps:

- for each position j and state $s \in \mathcal{S}$, compute posterior probability $p(s_j = s|\mathbf{x})$,
- find the allowed state path $\mathbf{s}^* = \text{argmax}_{\mathbf{s}} \prod_j p(s_j = \mathbf{s}|\mathbf{x})$,
- assign to $\mathbf{x}$ a label sequence $\mathbf{y}$ so that $y_j = \Lambda(s_j)$ for each position j.

The first step can be accomplished using the Forward-Backward algorithm as described for the free phase of parameter estimation. In order to find the best allowed state path, a Viterbi search is performed over posterior probabilities. In what follows $\rho_j(s|\mathbf{x})$ is the most probable allowed path of length j ending in state s and $\pi_j(s)$ is a traceback pointer. The algorithm can be described as follows:

1. Initialization:

$$\rho_0(\mathbf{BEGIN}|\mathbf{x}) = 1,$$
$$\rho_0(s|\mathbf{x}^{(i)}) = 0, \forall s \in S \setminus \{\mathbf{BEGIN}\}.$$

2. Recursion

$$\rho_j(\mathbf{s}|\mathbf{x}) = \max_{s'} \rho_{j-1}(\mathbf{s}|\mathbf{x}) \cdot \Gamma(s', s) \cdot p(s_j = s|\mathbf{x}),$$
$$\pi_j(s) = \text{argmax}_{s'} \rho_{j-1}(\mathbf{s}|\mathbf{x}) \cdot \Gamma(s', s).$$

3. Termination and Traceback

$$s^*_{n+1} = \mathbf{END},$$
$$s^*_j = \pi_{j+1}(s^*_{j+1}) \quad \text{for} \quad j = n, \ldots, 1,$$
$$s^*_0 = \mathbf{BEGIN}.$$

The labels are assigned to the observed sequence according to the state path $\mathbf{s}^*$. It is also possible to consider a slightly modified version of the algorithm where, for each position, the posterior probability of the *labels* is considered, and the states with the same label have associated the same posterior probability. The rationale behind this is to consider the aggregate probability of all state paths corresponding to the same sequence of labels to improve the overall per label accuracy. In many applications this variant of the algorithm might perform better.

8. – A test case: How to predict disulfide bridges in proteins

Protein folding in vivo can be constrained by post-translational modifications, chemical modifications that occur after protein translation at the ribosome level. These changes have routinely a physiological meaning and depending on the protein final function and they include cuts of the peptide chain or other modifications such as glycosylation (the enzymatic process that attaches glycans to specific lateral side chains), phosphorylation (the addition of a phosphate group that activates and deactivates many important biological processes) and formation of disulphide bridges. This latter modification generally occurs when proteins are secreted before becoming functionally active. A disulfide bridge is the only covalent bond among two cysteines lateral side chains that during the folding process become optimally oriented in the protein space. The bond is reversible and its presence/absence is regulated by the ambient redox potential. When a protein chain contains different cysteines the question then arises as to which cysteines pairs can be bonded and to which extent the disulfide bond formation depends on the flanking residues of the cysteines. We developed different approaches through the years. From the original method based on neural networks [11-13], to a recent one based on hidden neural newtworks, that combines neural networks and hidden Markov models [14, 15]. More recently we developed DISLOCATE [6], a two-step method based on GRHCRF for predicting both the bonding state and the connectivity patterns of cysteine residues in

a protein chain of Eukaryotes. We found that the inclusion of protein subcellular localization [16] improves the performance of these predictive steps by 3 and 2 percentage points, respectively. When compared with previously developed methods for predicting disulfide bonds from sequence, DISLOCATE improves the overall performance by more than 10 percentage points and it is the state of art predictor for the problem at hand. The method is freely available at `www.biocomp.unibo.it/savojard/Dislocate.html`, and the GRHCRF code is available at `www.biocomp.unibo.it/savojard/biocrf.html`.

* * *

RC thanks the following grants: PRIN 2009 project 009WXT45Y (Italian Ministry for University and Research: MIUR), COST BMBS Action TD1101(European Union RTD Framework Programme), and PON project PON01_02249 (Italian Ministry for University and Research: MIUR). CS is a recipient of a PHD fellowship from the Ministry of the Italian University and Research.

REFERENCES

[1] Piovesan D., Martelli P. L., Fariselli P., Zauli A., Rossi I. and Casadio R., *Nucleic Acids Res.*, **39** (2011) 197.
[2] Pierleoni A., Indio V., Savojardor C., Fariselli P., Martelli P. L. and Casadio R., *Nucleic Acids Res.*, **39** (2011) 375.
[3] Lafferty J., McCallum A. and Pereira F., *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, in *Proc. 18th International Conf. on Machine Learning*, edited by Kaufmann Morgan (2001) pp. 282-289.
[4] Wang S., Quattoni A., Morency L. P., Demirdjian D. and Darrell T., *Computer Vision and Pattern Recognition*, **39** (2006) 1521.
[5] Fariselli P., Savojardo C., Martelli P. L. and Casadio R., *Algorithms for Molecular Biology*, **4** (2009) 13.
[6] Savojardo C., Fariselli P., Alhamdoosh M., Martelli P. L. and Casadio R., *Bioinformatics*, **27** (2011) 2224.
[7] Savojardo C., Fariselli P., Martelli P. L., Shukla P. and Casadio R., *Prediction of the bonding state of cysteine residues in proteins with machine-learning methods*, in *CIBB 2010, Lecture Notes in Bioinformatics*, edited by Rizzo R. and Lisboa P. J. G. (2010) pp. 98-111.
[8] Martelli P. L., Fariselli P., Krogh A. and Casadio R., *Bioinformatics*, **18** (2002) 46.
[9] Durbin R., Eddy S., Krogh A. and Mitchinson G., *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge University Press) 1998.
[10] Baldi P. and Brunak S., *Bioinformatics: the Machine Learning Approach* (MIT Press) 2001.
[11] Fariselli P., Riccobelli P. and Casadio R., *Proteins*, **36** (1999) 340.
[12] Fariselli P. and Casadio R., *Bioinformatics*, **17** (2001) 957.
[13] Fariselli P., Martelli P. and Casadio R., *A neural network based method for predicting the disulfide connectivity in proteins*, in *KES 2002*, edited by Damiani E. *et al.* (2002) pp. 464-468.
[14] Martelli P. L., Fariselli P., Malaguti L. and Casadio R., *Protein Eng.*, **15** (2002) 951.
[15] Martelli P. L., Fariselli P. and Casadio R., *Proteomics*, **4** (2004) 1665.
[16] Pierleoni A., Martelli P. L., Fariselli P. and Casadio R., *Bioinformatics*, **22** (2006) 408.