On the Role of Long-Range Dependencies in Learning Protein Secondary Structure

Alessio Ceroni
Dipartimento di Sistemi e Informatica
Università di Firenze
50139 Firenze, Italy
E-mail: aceroni@dsi.unifi.it

Paolo Frasconi
Dipartimento di Sistemi e Informatica
Università di Firenze
50139 Firenze, Italy
E-mail: paolo@dsi.unifi.it

***Abstract*— Accuracy of protein secondary structure predictors has been slowly growing during the last decade. Although it is clear that a relatively large fraction of current errors is due to long-range interactions, current predictors are not able to exploit such information. We present a solution based on a generalized bidirectional neural network that learns from sequences and associated interaction graphs to improve secondary structure prediction.**

I. Introduction

Prediction of protein secondary structure (SS) is a classic problem in computational molecular biology and one of the first successful applications of machine learning to bioinformatics. In this task, we are given as input the sequence of a protein (whose actual three-dimensional structure is unknown) and we are interested in learning the folding regularities that are formed at local level (often maintained by hydrogen bonds) and that are traditionally divided into three main classes: alpha helices, beta sheets, and coils. A supervised learning problem is then formulated as the association between an input sequence (representing the protein primary structure) and an output string that contains the SS at each residue.

Most available prediction methods use feedforward neural networks whose input is the multiple alignment profile in a sliding window of residues centered around the target position [1], [2], [3], [4]. By construction, predictions obtained with these methods are local. Long-range dependencies, on the other hand, clearly play an important role in this problem. For example, a beta sheet is composed of two or more strands that are held together by hydrogen bonds but that can be situated very far apart in the primary structure of the protein. In this case, residues that are close in space occupy distant positions in sequence. A similar effect is observed for cysteines that are linked by disulfide bonds.

In [5] it was proposed the use of bidirectional recurrent neural networks (BRNN) for the prediction of SS. The architecture in this case allows us to process the sequence as a whole and to "translate" the input profile at each position into a corresponding output prediction for that position. In this way, architectural design is simplified since there is no need of choosing a window size. Theoretically, the output at any position in a BRNN depends on the entire input sequence and thus a BRNN might actually exploit long-range information. Unfortunately, well known problems of vanishing gradients [6] do not allow us to *learn* these dependencies. In practice, even BRNNs do not outperform the best predictors based on feedforward networks.

In this paper, we are interested in developing an architecture that can effectively exploit long-range dependencies assuming some additional information is available to the learner. This architecture may help towards improving the present state-of-the-art and elucidate in a quantitative way the importance of long-range information in the prediction of SS.

We start from a rather simple intuitive argument: learning efficiently in the presence of long-range dependencies is not possible because of missing information about which remote sequence positions do interact: the learner is only given a set of inputs and a serial order relation on them and must solve a difficult credit assignment problem to identify the interacting positions. However, if the learner had access to information about which positions pairs are expected to interact, its task would be greatly simplified and it could possibly succeed. When available, this information can be conveniently expressed in the form of an undirected graph whose vertices are sequence positions and edges are interacting pairs.

In the case of SS prediction, a reasonable source of information about long-range interaction can be obtained from contact maps, a graphical representation of the spatial neighborhood relation among amino acids. An edge (t, s) in a contact map indicates that the distance between the C_α atoms of the residues at positions t and s is lower than a predefined threshold (see Figure 1). Of course in order to obtain a contact map the protein structure must be known. In addition, it is well known that backbone atoms' coordinates can be reconstructed with small error starting from contact maps [7]. Thus, in a sense, using contact maps information in order to predict SS might appear foolish since most of the information about the 3D structure of the protein is already contained in the map. However, the following considerations suggest that this setting is worth investigation:

- Algorithms that reconstruct structure from contact maps are based on the definition of a potential energy function whose global optimization is not straightforward and requires stochastic optimization techniques to escape local minima [7]. Thus it is not clear that a supervised learning

algorithm can actually *learn* to recover SS from contact maps.

- Contact maps can be predicted from sequence [8], [9], [10] or can be obtained from structures predicted by ab-initio methods such as Rosetta [11]. Although accuracy of present methods is certainly not sufficient to provide a satisfactory solution to the folding problem, predicted maps may still contain useful information to improve the prediction of lower order properties such as the SS.
- Even if contact maps are given, the design of a learning algorithm that can fully exploit their information content is not straightforward. For example, Meiler and Baker [12] have shown that SS prediction can be improved by using information about inter-residue distances. Their architecture is a feedforward network fed by *average* property profiles associated with amino acids that are near in space to the target position. In this way, relative ordering among neighbors in the contact map is discarded.

The solution proposed in this paper is based on an extended architecture that receives as an additional input a graphical description of the pairwise interactions between sequence positions. We call this architecture interaction enriched BRNN (IEBRNN).

In Section II we briefly review the problem of secondary structure prediction. In Section III we illustrate the learning problem setup and the details of IEBRNN. In Section IV we describe the data used in the experiments and in Section V we report and analyze our results.

II. Background

A. The folding problem

A protein chain is a polymer formed by amino acids linked by peptide bonds. Each amino acid in a chain is also called a *residue*. Since there are 20 amino acids in nature, the sequence can be conveniently represented by a string in a 20 letters alphabet. The biological function of a protein depends on its fold or *tertiary* structure, (i.e. the coordinates of all its atoms). The experimental determination of structures is a costly process that is presently carried out either by X-rays crystallography or by nuclear magnetic resonance methods. Presently there are about 24 million protein sequences derived from the analysis of genomic data, but only less than 21,000 3D structures are known and deposited in the Protein Data Bank (PDB) [13]. According to Anfinsen's thermodynamic hypothesis [14], a protein's native fold is the one having lowest free energy and thus it depends on the sequence and the external environment [15]. Unfortunately, computing and minimizing the potential function from first principles is not feasible for present computers and therefore predictive methods are a very important approach in order to obtain an approximation of 3D structure associated with known sequences. Prediction methods are basically divided into three main classes: homology modeling (that can be applied if the target protein has sufficient sequence similarity with existing structures), fold recognition (that requires similarity at the level of fold category), and ab-initio for new folds. Machine learning is a promising approach in the latter case [16].

Fig. 1. 3D structure (left) and contact map (right) of Glutaredoxin (PDB code 1ABA). Contact resolution in this example is 6 Å. Contacts between residues that are closer than 3 positions in sequences are omitted.

B. SS prediction

Predictors based on neural networks have been pionereed by Qian and Sejinowsky [17] and subsequently refined during the 1990's. Rost and Sander [1] have incorporated evolutionary information in the form of multiple alignment profiles, a technique that significantly boosted prediction accuracy. Subsequently, Riis and Krogh [2] have introduced a number of architectural improvements, including the use of adaptive encoding of amino acids and the use of a structure-to-structure network trained as a postprocessor to filter out local prediction errors. Jones [3] suggested the use of position specific matrices for incorporating evolutionary information, and obtained favorable results in terms of prediction accuracy also thanks to the combination of multiple classifiers. Cuff and Barton [4] have shown how different combinations of neural networks and multiple sequence alignment profiles may significantly affect prediction accuracy.

A widely used measure of performance is Q_3, defined as the fraction of residues whose SS is assigned correctly by the predictor. Currently available methods report a Q_3 level between 75% and 80% in the three-state SS prediction. A complementary measure that quantifies the capability of the classifier to correctly predict entire segments of SS is segment overlap (SOV) – see [18] for details.

III. Interaction-enriched BRNNs

The architecture presented here is an extension of the one proposed in [19] for directed acyclic graphs and the one used in [5] for non-causal processing of protein sequences. Its purpose is to map input sequences, enriched with interaction relations, to corresponding output sequences.

A. Representation of the data

In a standard supervised sequence learning problem we are given a dataset of input-output pairs $D_m = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$ where $\boldsymbol{x}_i = (\boldsymbol{x}_i(1), \boldsymbol{x}_i(2), \ldots, \boldsymbol{x}_i(N_i))$ is an input sequence of length N_i and $\boldsymbol{y}_i = (y_i(1), y_i(2), \ldots, y_i(N_i))$ is the corresponding desired output sequence. The learning problem

consists of seeking a function $\boldsymbol{f}(\boldsymbol{x})$ for predicting $\boldsymbol{y}$ on new sequences. Since an input and its corresponding output sequences have the same length, we may actually use a function $f(\boldsymbol{x}, t)$ that predicts $y(t)$ for each position t. Each input element will be assumed to be a real vector, i.e. $\boldsymbol{x}(t) \in I\!R^n$. Each output element will be assumed to be integer-valued, i.e. $y_i(t) \in [1, \ldots, K]$ as in multiclass classification.

In our extended setting, inputs are enriched with interaction relations. In particular, each input will be now described as a pair $\boldsymbol{X}_i = (\boldsymbol{x}_i, G_i)$ where $\boldsymbol{x}_i$ is the input sequence as before and $G_i = (V_i, E_i)$ is an undirected graph whose vertex set is simply $V_i = \{1, 2, \ldots, N_i\}$ and whose edges represent interactions, i.e. $(t, s) \in E_i$ if and only if positions t and s interact. Clearly $\boldsymbol{X}_i$ can also be seen as an undirected graph with labels $\boldsymbol{x}_i(t)$ on its vertices (see Figure 2). If we need to further enrich the input with attributes associated with the interactions themselves, then we could add labels to the edges of $\boldsymbol{X}_i$. Note, however, that if we assume bounded connectivity, then edge labels can be moved into both adjacent vertices without loss of information (remember that vertices are ordered and thus edge labels can be stored into vertices following the ordered adjacency list); therefore they will not further considered.

B. Architecture

The architecture used in this paper is closely related to the recursive neural networks (RNN) described in [19] and the bidirectional recurrent neural network (BRNN) used in [5]. In these architectures, memories are realized with (hidden) state variables and the input-output mapping is obtained as the composition of a state transition function and an output function. RNNs can solve structural transduction problems, i.e. mapping a labeled input graph $\boldsymbol{x}$ into a corresponding output graph $\boldsymbol{y}$ isomorph to $\boldsymbol{x}$. Basically in RNNs the label of each input vertex is mapped into a label of the associated output vertex and the mapping depends on the context defined by other vertices in the graph. RNNs interpret edges as *causal* dependencies and therefore impose a restriction of the graph that needs to be directed and acyclic. Causality in this context means that outputs at a given vertex v, $f(\boldsymbol{x}, v)$ only depend on the input labels found on vertices of $\boldsymbol{x}$ that can be reached from v. BRNNs, on the other hand, are based on a factorization of the state space and use two transition functions that process the input sequence in both directions. BRNNs do not make any causality assumption in the data[1] but cannot deal with graphs. The solution suggested here extends the transition functions of BRNNs to incorporate dependencies between interacting positions.

To simplify notation, in the following we omit the integer subscripts that index training examples and we focus on a single input-output pair $(\boldsymbol{X}, \boldsymbol{y})$. For each position t, we introduce two real vectors $\boldsymbol{\varphi}(t)$ and $\boldsymbol{\beta}(t)$ that we call the *forward* state and the *backward* state, respectively. For simplicity

[1]This makes sense, for example, in the case of protein data: it is not true that SS at position t depends only on amino acids situated *before* t or only on those situated *after* t.

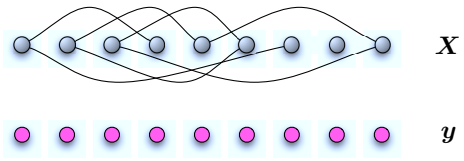

Fig. 2. Input and output portion of the data.

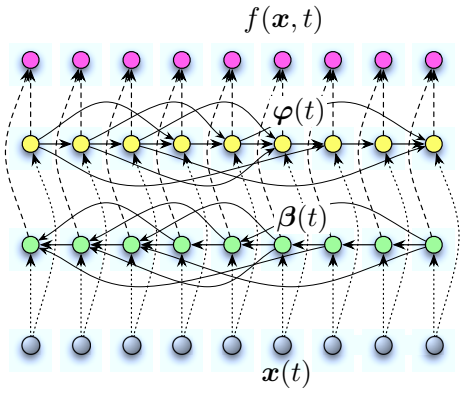

Fig. 3. Graphical representation of the message passing in the IEBRNN architecture.

we assume they have the same dimension d. Intuitively, the forward state at t is a fixed-size representation of the "past" substring $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(t)$ while the backward state is a fixed-size representation of the "future" substring $\boldsymbol{x}(t), \ldots, \boldsymbol{x}(N)$. The concatenation of both vectors is expected to contain all the information about the input sequence that is needed to make a prediction at position t. In a standard BRNN, $\boldsymbol{\varphi}(t)$ is recursively updated from $\boldsymbol{\varphi}(t-1)$ and $\boldsymbol{\beta}(t)$ is recursively updated from $\boldsymbol{\beta}(t+1)$. In the IEBRNN we introduce shortcuts in the signal flow associated with the state variables update. These shortcuts will be placed in correspondence with interacting positions, as specified by G.

We begin by building two directed graphs from G as follows. Let

$$E_F \doteq \{(s,t) : \{s,t\} \in E, s < t\} \quad (1)$$

$$E_B \doteq \{(s,t) : \{s,t\} \in E, s > t\}. \quad (2)$$

We now have a predecessor graph $G_F = (V, E_F)$ with forward oriented edges and a successor graph $G_B = (V, E_B)$ with backward oriented edges. It is immediate to see that G_F and G_B are acyclic. Moreover, for each vertex t, edges (t, s) incident on t can be sorted in increasing order of s. Also, we assume that all graphs have degree smaller than a fixed constant M. Define

$$\ell_{t,j} = \begin{cases} j\text{-th vertex in the sorted adjacency list of } t \text{ in } G_F & \text{if } t \text{ has at least } j \text{ parents in } G_F \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

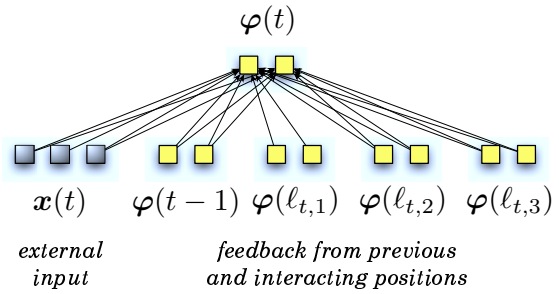

Fig. 4. Realization of state transition function $\mathcal{T}_F$ by a feedforward network. Here $n = 3$, $d = 2$ and $M = 3$ — see (5).

and

$$r_{t,j} = \begin{cases} j\text{-th vertex in the sorted adjacency list of } t \text{ in } G_B & \text{if } t \text{ has at least } j \text{ parents in } G_B \\ N+1 & \text{otherwise} \end{cases} \quad (4)$$

The IEBRNN is then based on the following non-causal dynamics:

$$\boldsymbol{\varphi}(t) = \mathcal{T}_F\left(\boldsymbol{x}(t), \boldsymbol{\varphi}(t-1), \boldsymbol{\varphi}(\ell_{t,1}), \boldsymbol{\varphi}(\ell_{t,2}), \ldots, \boldsymbol{\varphi}(\ell_{t,M}); \boldsymbol{\theta}_F\right) \quad (5)$$

$$\boldsymbol{\beta}(t) = \mathcal{T}_B\left(\boldsymbol{x}(t), \boldsymbol{\beta}(t+1), \boldsymbol{\beta}(r_{t,1}), \boldsymbol{\beta}(r_{t,2}), \ldots, \boldsymbol{\beta}(r_{t,M}); \boldsymbol{\theta}_B\right) \quad (6)$$

with boundary conditions

$$\boldsymbol{\varphi}(0) = \boldsymbol{\beta}(N+1) = 0. \quad (7)$$

In the above recursions, $\mathcal{T}_F$ and $\mathcal{T}_B$ are the forward and backward state transition function, respectively. They are parametric functions with adjustable parameters $\boldsymbol{\theta}_F$ and $\boldsymbol{\theta}_B$ that will be determined by learning. We assume here that they are realized by feedforward neural networks with $n + (M+1)d$ inputs and d outputs. An example is shown in Figure 4.

Outputs are then computed as follows:

$$f(\boldsymbol{x}, t) = \eta(\boldsymbol{\varphi}(t), \boldsymbol{\beta}(t); \boldsymbol{\theta}_Y); \quad (8)$$

where η is also a parametric function with adjustable parameters $\boldsymbol{\theta}_Y$. Sequential translation with discrete outputs assigned to each position is similar to multiclass classification. Assuming K classes, we define

$$a_k(\boldsymbol{x}, t) = < [\boldsymbol{\varphi}(t), \boldsymbol{\beta}(t)], \boldsymbol{\theta}_{k,Y} > \quad k = 1, \ldots, K \quad (9)$$

where $[\boldsymbol{\varphi}(t), \boldsymbol{\beta}(t)]$ is the concatenation of $\boldsymbol{\varphi}(t)$ and $\boldsymbol{\beta}(t)$ and $< \cdot, \cdot >$ denotes dot product. We then compute

$$f_k(\boldsymbol{x}, t) = \frac{e^{a_k(\boldsymbol{x},t)}}{\sum_{j=1}^{K} e^{a_j(\boldsymbol{x},t)}}. \quad (10)$$

The use of normalized exponentials allows us to interpret $f_k(\boldsymbol{x}, t)$ as the conditional probability $p(y(t) = k|\boldsymbol{x})$ that the class at position t is k, given the input $\boldsymbol{x}$.

The computations described by Equations (5), (6), and (8) can be graphically described as shown in Figure 3. Nodes in the diagram represent input, state, and output vectors at different sequence positions. Arcs represent arguments to transition and output functions. Dotted arcs represent the first argument of functions $\mathcal{T}_F$, $\mathcal{T}_B$. Solid arcs going left-to-right and right-to-left represent the second argument of functions $\mathcal{T}_F$, $\mathcal{T}_B$ and are also found in the computation of standard BRNNs. Thin arcs are shortcut inherited from the interaction graphs associated with the input sequence. Finally, dashed arcs represent the arguments of the output function η.

C. Learning

Training is carried out following a rather standard neural network approach: a likelihood function of the parameters and the training set is obtained thanks to the probabilistic interpretation of the outputs. Assuming iid sequences and denoting by $\boldsymbol{\theta}$ the whole set of parameters we have

$$\ell(D_m|\boldsymbol{\theta}) \doteq \log p(D_m|\boldsymbol{\theta}) = \sum_{i=1}^{m} \sum_{t=1}^{N_i} p(y_i(t)|\boldsymbol{x}_i, \boldsymbol{\theta}) \quad (11)$$

that can be rewritten as

$$\ell(D_m|\boldsymbol{\theta}) = \sum_{i=1}^{m} \sum_{t=1}^{N_i} \sum_{k=1}^{K} z_{i,k}(t) \log f(\boldsymbol{x}_i, t; \boldsymbol{\theta}). \quad (12)$$

where $z_{i,k}(t) = 1$ if $y_i(t) = k$ and 0 otherwise.

Gradient computation for likelihood optimization can be carried out analitically using backpropagation on the unfolded architecture (see Figure 3) and taking into account the fact that parameters of the transition functions and the output function are shared across different sequence positions. Details on error propagation in recursive networks for graphs can be found in [19].

IV. Data used in the experiments

The experiments have been performed using a representative set of non homologous chains from the Protein Data Bank (PDB Select [20]). We extracted the sequences from the December 2002 release, listing 1,950 chains with a percentage of homology lower than 25%. From this set we retained only high quality proteins on which the DSSP program does not crash, determined only by X-ray diffraction, without any physical chain breaks and resolution threshold lower than 2.5 Å. The final dataset contained 811 chains (137,926 residues) split in a training set of 398 chains (69,054 residues), a validation set of 151 chains (22,959 residues) and a test set of 262 chains (45,913 residues).

Multiple alignments were generated using PSI-BLAST [21] applied to the Swiss-Prot+TrEMBL non-redundant database [22]. SS labels were assigned using the DSSP program [23]. In case of ambiguities in the PDB files, the coordinates of the C_α atoms used to calculate the contact map were selected according to the same strategy used by DSSP. We reduced DSSP's eight classes into the three main classes by mapping H into α (helices), E into β (strands), and the rest (B, C, G, I,

S, T) into γ (coils). Contacts were defined using a threshold of 6 Å, a value which comprises all the hydrogen bonded pairs and few side-chain contacts. Amino-acids spaced less than 3 positions in the sequence were not considered, because their C_α atoms are always at a distance lower than 6 Å.

V. Prediction of secondary structure from sequence and contact maps

A. Prediction from sequence alone

A first set of experiments was performed to obtain a baseline prediction accuracy for the SS problem on this dataset. In this paper we are interested in highlighting the contributions of the long-range information to the prediction of SS rather than in obtaining state-of-the-art performances from the classifier. For this reason a "bare bones" BRNN classifier with multiple alignment profiles as input was used for this purpose. All common tricks that can be used to boost accuracy (in particular shortcuts connections between non-consecutive states, computation of outputs at each position in the sequence using a window of states, and ensembles of independent classifiers, as in [5], [24]) were omitted.

The network size, together with the parameters of the learning algorithm, were chosen using the validation set: an architecture with $d = 20$ neurons for each recursive state was then selected, the learning rate was set fixed to $1 \cdot 10^{-4}$ and an early stopping procedure has been used to avoid overfitting. The same parameters and sizes were used in all the experiments reported here and in the following. The validation set was also used for the early-stopping procedure.

Results of the baseline experiment are reported in the upper-left corner of Table I. The corresponding confusion matrix is shown in the upper-left corner of Table II. In this matrix, entry at row i and column j is the fraction of residues in class j that are predicted as belonging to class i. For each class, percentages marked in boldface and in italics are precision and recall, respectively.

B. Prediction from contact maps alone

Here we want to estimate the amount of information about SS the IEBRNN architecture is capable to learn from contacts alone. Therefore, in this experiment null inputs were used, i.e. $\boldsymbol{x}(t) = 0$ for each position t. Information on contacts was inserted in the form of an interaction graph, as explained in section III.

As reported in Table I, we obtained $Q_3 \approx 80\%$ and SOV $\approx 73\%$. These results are quite impressive considered that the classifier has no knowledge about the tridimensional conformation of the hydrogen bonded atoms and the physics behind the formation of SS.

Comparing the predictions of this classifier and those of the classifier described above in Section V-A we found that they overlap only for the 69% of the residues in the test set. Moreover, 92% of the times at least one of the two classifiers makes the correct prediction. It is then arguable that BRNNs trained on profile sequences and IEBRNNs trained on contact maps alone capture rather different regularities in the data. This suggests that training the IEBRNN with both inputs should allow us to obtain improved accuracy.

TABLE I

Performances of the various methods presented in this paper.

BRNN	Q_3	SOV	IEBRNN	Q_3	SOV
Profiles only	74.6%	66.7%	Interactions only	79.9%	73.3%
Profiles + context	82.5%	77.4%	Profiles + interactions	84.6%	79.3%
Profiles + context (no γ)	95.9%	94.6%	Profiles + interactions (no γ)	97.9%	95.5%

TABLE II

Confusion matrices for the different methods.

BRNN — Profiles only

	α	β	γ
α	**77.8%**	5.8%	16.4%
β	8.2%	**70.9%**	20.9%
γ	11.6%	14.5%	**74.0%**
Recall	*77.4%*	*58.9%*	*80.2%*

BRNN — Profiles + context

	α	β	γ
α	**85.1%**	2.6%	12.2%
β	4.5%	**76.0%**	19.5%
γ	6.7%	9.6%	**83.7%**
Recall	*87.6%*	*76.5%*	*81.8%*

BRNN — Profiles + context (no γ)

	α	β	γ
α	**93.4%**	6.5%	0.1%
β	6.4%	**93.4%**	0.1%
γ	0.2%	0.9%	**98.9%**
Recall	*95.5%*	*88.3%*	*99.9%*

IEBRNN — Interactions only

	α	β	γ
α	**83.8%**	0.8%	15.5%
β	1.1%	**76.5%**	22.4%
γ	9.8%	11.4%	**78.8%**
Recall	*85.2%*	*75.3%*	*78.4%*

IEBRNN — Profiles + interactions

	α	β	γ
α	**88.6%**	0.7%	10.7%
β	1.2%	**81.9%**	16.9%
γ	8.0%	8.9%	**83.1%**
Recall	*87.6%*	*80.1%*	*84.6%*

IEBRNN — Profiles + interactions (no γ)

	α	β	γ
α	**98.3%**	1.6%	0.1%
β	2.4%	**97.5%**	0.1%
γ	1.2%	0.9%	**97.9%**
Recall	*96.6%*	*95.8%*	*99.9%*

C. Prediction from profiles and contacts together

In this experiment, we trained the IEBRNN with both profiles and contacts as input.

For sake of comparison with the results of Meiler and Baker [12] we also trained a standard BRNN with the same kind of inputs they used. In particular, the spatial *context* of each residue was computed by averaging the profile of the amino-acids in a sphere of 6 Å centered on the residue itself. This additional input was then given to the standard BRNN together with the usual profile. This method can be seen as a simplified version of the IEBRNN in which all contacts give the same contribution to a given position. In particular, order among contacts cannot be distinguished.

As can be seen from Table I, the information about contact ordering is efficiently exploited by the IEBRNN, which

appreciably outperforms the solution based on the average context. Interestingly, prediction accuracy improves for helices and strands but not for coils (see confusion matrices in Table II).

D. Effects of interaction robustness

The results of the above experiments show us that IEBRNNs can effectively exploit the information contained in the contact map to improve prediction accuracy. However there is still about 15% residual error rate that would be interesting to explain. We conjecture that the reliability of the interactions that were injected as an additional input may play a significant role. In facts, edges in a contact map express spatial proximity but do not necessarily imply dependencies between the two close residues. This may be particularly true in the case of contacts that involve coil residues. Instead, contacts between residues that both belong to helices or strands can be expected to encode interactions in a more robust way since they are often maintained by hydrogen bonds.

In order to evaluate the effect of interaction robustness we repeated our experiments using more sparse contact maps where edges only connect residues that belong to helices or strands. In so doing, we removed about 60% of the edges from interaction graphs. Results are reported in the last row of Table I both for the standard BRNN (fed by profiles and context) and for the IEBRNN. The error reduction obtained in this way is dramatic. The residual error is comparable to the disagreement between different SS assignment programs (i.e. DSSP, STRIDE, and DEFINE). These experiments could indicate that part of the information contained in the contact map can be misleading, especially for the shorter segments.

VI. Conclusions

We have introduced IEBRNN as a method for simplifying sequence learning tasks with neural networks by incorporating explicit knowledge about interactions between sequence elements. Empirical results show that this approach can greatly improve prediction of protein secondary structure when interaction knowledge is available. In addition, in this problem IEBRNNs are able to capture the serial order relation within interactions, a property that allows us to obtain better prediction than simply averaging the input at interacting positions in a sequence, as was proposed before by other researchers.

The method described here does not yet advance the state-of-the-art in SS prediction but enlightens possible future directions of investigation. Our findings provides further evidence that the knowledge of a relatively small number of highly reliable interactions (such as contacts between residues in helices and strands) is sufficient with the currently available data to obtain high quality predictions. In order to advance this study we are investigating a tight integration between IEBRNNs and previous methods for prediction of contact maps [10].

Acknowledgment

We thank Andrea Passerini and Alessandro Vullo for useful discussions. This work was supported in part by the Italian Department of Education, University, and Research (MIUR) under grant no. 2002093941.

References

[1] B. Rost and C. Sander, "Improved prediction of protein secondary structure by use of sequence profiles and neural networks," *Proc. Natl. Acad. Sci. USA*, vol. 90, no. 16, pp. 7558–7562, 1993.

[2] S. K. Riis and A. Krogh, "Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments," *J. Comput. Biol.*, vol. 3, pp. 163–183, 1996.

[3] D. Jones, "Protein secondary structure prediction based on position-specific scoring matrices," *J. Mol. Biol.*, no. 292, pp. 195–202, 1999.

[4] J. A. Cuff and G. J. Barton, "Application of multiple sequence alignment profiles to improve protein secondary structure prediction," *Proteins*, vol. 40, pp. 502–511, 2000.

[5] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri, "Exploiting the past and the future in protein secondary structure predicton," *Bioinformatics*, vol. 15, pp. 937–946, 1999.

[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] M. Vendruscolo, E. Kussel, and E. Domany, "Recovery of protein structure from contact maps," *Fold. Des.*, vol. 2, pp. 295–306, 1997.

[8] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio, "Prediction of contact maps with neural networks and correlated mutations," *Prot. eng.*, vol. 14, pp. 835–843, 2001.

[9] G. Pollastri and P. Baldi, "Prediction of contact maps by giohmms and recurrent neural networks using lateral propagation from all four cardinal corners," *Bioinformatics*, vol. 18, Supplement 1, pp. S62–S70, 2002.

[10] G. Pollastri, P. Baldi, A. Vullo, and P. Frasconi, "Prediction of protein topologies using GIOHMMs and GRNNs," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2003.

[11] D. Baker and A. Sali, "Protein structure prediction and structural genomics," *Science*, vol. 294, pp. 93–6, 2001.

[12] J. Meiler and D. Baker, "Coupled prediction of protein secondary and tertiary structure," *Proc Natl Acad Sci U.S.A.*, 2003.

[13] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, pp. 235–242, 2000.

[14] C. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, pp. 223–230, 1973.

[15] J. Berg, J. Tymoczko, L. Stryer, and N. Clarke, *Biochemistry*, 5th ed. W.H. Freeman & Co., 2002.

[16] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[17] N. Qian and T. J. Sejnowski, "Predicting the secondary structure of glubular proteins using neural network models," *J. Mol. Biol.*, vol. 202, pp. 865–884, 1988.

[18] A. Zemla, C. Venclovas, K. Fidelis, and B. Rost, "A modified definition of SOV, a segment-based measure for protein secondary structure prediction assessment," *Proteins*, vol. 34, no. 2, pp. 220–223, 1999.

[19] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. on Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.

[20] U. Hobohm and C. Sander, "Enlarged representative set of protein structures," *Protein Science*, vol. 3, pp. 522–524, 1994.

[21] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI–BLAST: a new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.

[22] A. Bairoch and R. Apweiler, "The Swiss-Prot protein sequence data bank and its new supplement TrEMBL," *Nucleic Acids Research*, vol. 24, pp. 21–25, 1996.

[23] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, pp. 2577–2637, 1983.

[24] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, "Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles," *Proteins*, vol. 47, no. 2, pp. 228–235, 2002.