New Machine Learning Methods for the Prediction of Protein Topologies

Pierre Baldi[1] Gianluca Pollastri[1] Paolo Frasconi[2]
Alessandro Vullo[2]

[1]*Department of Information and Computer Science*
Institute for Genomics and Bioinformatics
University of California, Irvine
Irvine, CA 92697-3425, USA
pfbaldi,gpollast@ics.uci.edu
+1 (949) 824-5809
+1 (949) 824-4056 (FAX)

[2] Dipartimento di Sistemi e Informatica
Università di Firenze
Via di Santa Marta 3, 50139 Firenze, ITALY
paolo,vullo@dsi.unifi.it
+39 055 479-6362
+39 055 479-6363 (FAX)

Abstract. Protein structures are translation and rotation invariant. In protein structure prediction, it is therefore important to be able to assess and predict intermediary topological representations, such as distance or contact maps, that are translation and rotation invariant. Here we develop several new machine learning methods for the prediction and assessment of fine-grained and coarse topological representations of proteins. In particular, we introduce a general class of graphical model architectures together with the corresponding neural network implementations. These architectures can be viewed as Bayesian network generalizations of input-output hidden Markov models (GIOHMMs), involving an input layer, an output layer, and a hidden layer supported by one or several directed acyclic graphs. The corresponding generalized recursive neural network (GRNN) architectures are derived by preserving the graphical structures of the GIOHMMs, but replacing the conditional probability tables with learnable deterministic functions.

Two methods are proposed for the prediction of protein topological structures. The first method uses a GIOHMM organized into six horizontal layers: one input plane, four hidden planes, and one output plane that directly represents the adjacency matrix of the contact map (or the distance matrix). Each hidden plane is associated with one of the four cardinal corners towards which all the edges of the corresponding lattice are oriented. The corresponding GRNNs are used to construct a fine-grained contact map predictor. The second method uses a GIOHMM approach to learn a graph scoring function which, in turn, is used to efficiently search the space of possible configurations. The corresponding GRNNs are used to construct a coarse-grained contact map predictor. Computer simulations show that the predictors for both tasks achieve state-of-the-art performance.

Keywords: graphical models, Bayesian networks, recurrent neural networks.

1 Introduction

Predicting the 3D structure of chains of amino acids from their primary sequence is a fundamental open problem in computational molecular biology. Any approach to the problem must deal with the fundamental property that protein structures are invariant under translations and rotations. To this effect, we have proposed a machine learning approach to protein structure that decomposes the problem into three steps [10] (Figure 1), one of which computes an intermediate topological representation of the protein, the contact map, which is translation and rotation invariant. More precisely, the first step starts from the primary sequence, possibly in conjunction with multiple alignments to leverage evolutionary information, and predicts a number of structural features such as the classification of the amino acids present in the sequence into secondary classes (alpha helices, beta strands and coils), or into relative exposure classes (e.g. surface/buried). The second step starts from the primary sequence and the structural features and attempts to predict the contact map of the protein. The contact map is a 2D representation of neighborhood relationships consisting of an adjacency matrix at some distance cutoff (typically in the range of 6 to 12 Å), or the Euclidean distance matrix. Fine-grained contact maps are derived at the amino acid (or even atomic) level. Coarse contact maps can be derived by looking at secondary structure elements and, for instance, their centers of gravity. The third step is the prediction of actual 3D coordinates from 2D contact maps. Other topological representations can be obtained in terms of local relative angle coordinates.

Predictors for the first step are described in [36, 37], while methods for the third steps have been developed in the NMR literature and elsewhere [42, 43] using distance geometry and stochastic optimization techniques. The focus here is on the second and most difficult step. Various algorithms for the prediction of contacts [38, 32, 15, 16, 36], distances [2, 30, 22], and contact maps [17] have been developed, in particular using neural networks. The best contact map predictor in the literature and at the last CASP prediction experiment [29] reports an average accuracy of 21% correct prediction [17]. While this result is encouraging and well above chance level by a factor greater than 6, it is still far from providing sufficient accuracy for reliable 3D structure prediction. A key issue in this area is the amount of noise that can be tolerated in a contact map prediction without compromising the 3D-reconstruction step. While to the best of our knowledge systematic tests in this area have not yet been published, preliminary tests appear to indicate that recovery of about 50% of distant contacts at a 8Å distance cutoff ought to suffice for proper reconstruction, at least for proteins up to 150 amino acid long (Rita Casadio and Piero Fariselli, private communication and oral presentation during CASP4 [29]).

In this chapter, we describe several new machine learning methods for the prediction and assessment of protein topologies, and in particular of fine-grained and coarse contact maps. The algorithms are all based on connectionist architectures that can be viewed as noncausal generalizations of IOHMMS (input-output hidden Markov models) to process data structures richer than sequences, including spatial structures and undirected graphs.

In the first class of methods, recently described in [9, 35], we introduce a new general class of graphical model architectures together with their associated implementations in terms of recurrent neural network architectures. The one-dimensional case was originally introduced to address bioinformatics sequence analysis problems, in particular the prediction of protein structural features, such as protein secondary structure [6]. The key contribution here is the generalization from one to two dimensions, which opens the door to further generalizations both to higher dimensions and to other, non-necessarily spatial, data structures. Here we use

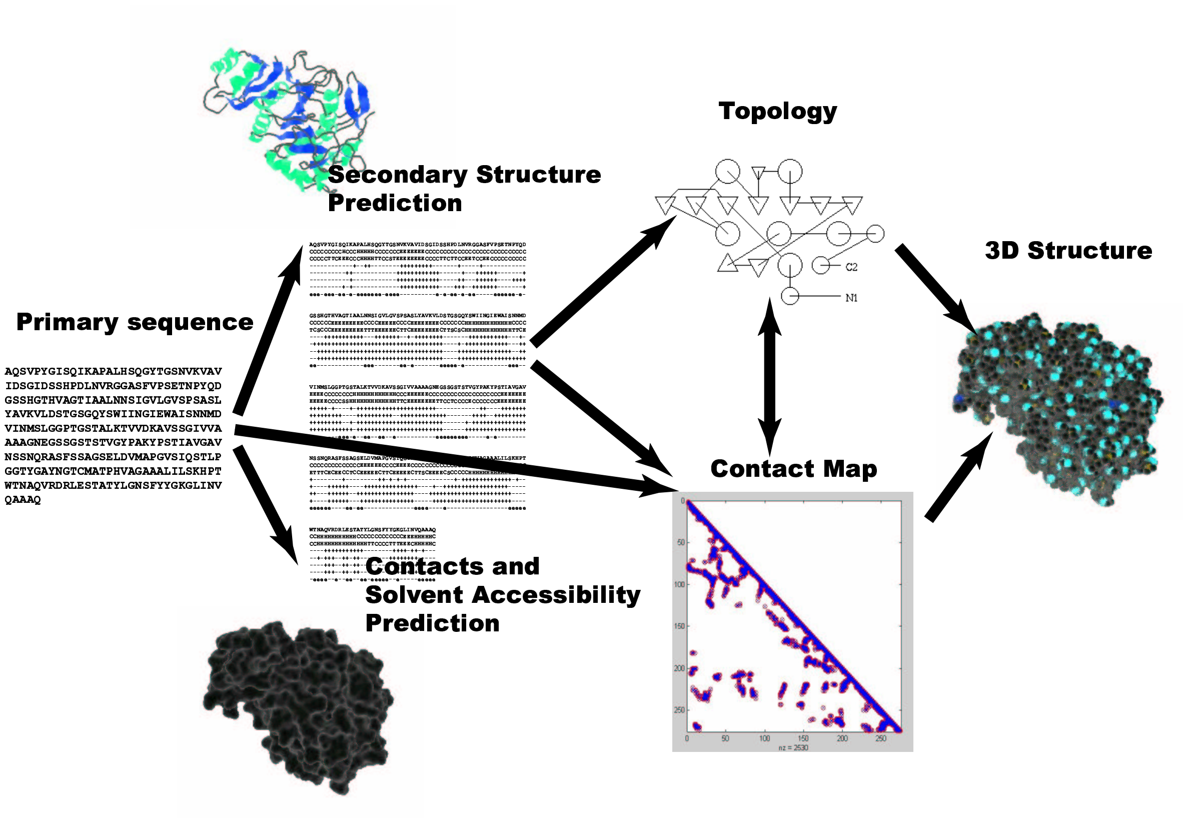
Topology
Secondary Structure Prediction
3D Structure
Primary sequence
AQSVPYGISQIKAPALHSQGYTGSNVKVAV
IDSGIDSSHPDLNVRGGASFVPSETNPYQD
GSSHGTHVAGTIAALNNSIGVLGVSPSASL
YAVKVLDSTGSGQYSWIINGIEWAISNNMD
VINMSLGGPTGSTALKTVVDKAVSSGIVVA
AAAGNEGSSGSTSTVGYPAKYPSTIAVGAV
NSSNQRASFSSAGSELDVMAPGVSIQSTLP
GGTYGAYNGTCMATPHVAGAAALILSKHPT
WTNAQVRDRLESTATYLGNSFYYGKGLINV
QAAAQ
Contact Map
Contacts and Solvent Accessibility Prediction
C2
N1

Figure 1: Overall pipeline strategy for machine learning protein structures. Example of 1SCJ (Subtilisin-Propeptide Complex) protein. The first stage predicts structural features including secondary structure, contacts, and relative solvent accessibility. The second stage predicts the topology of the protein, using the primary sequence and the structural features. The coarse topology is represented as a cartoon providing the relative proximity of secondary structure elements, such as alpha helices and beta-strands. The high-resolution topology is represented by the contact map between the residues of the protein. The final stage is the prediction of the actual 3D coordinates of all residues and atoms in the structure.

the first step in the pipeline to review the one-dimensional version of the architectures and the second step to introduce the key generalization. In the second class of methods [20], the learning task consists of predicting a scoring function associated with a hypothetical contact maps, with the purpose of guiding a graph search algorithm. In this case, recursive neural networks have been extended to handle undirected (and possibly cyclic) graphs. This was achieved by taking advantage of the peculiar property of protein contact maps, where vertices are uniquely ordered (e.g. from protein's N- to C-terminus).

Without getting into the intricacies of protein structure prediction, suffices it to say that the prediction of contact maps is probably the most challenging and essential step in the overall strategy. Because of the example we choose, our discussion will be entirely in terms of processing architectures with inputs and outputs. It should be obvious to the reader, however, that the same concepts can be used to produced similar architectures based on inputs only, outputs only (e.g. HMMs), or even no inputs and no outputs (e.g. Markov chains).

2 The One-Dimensional Case: Bidirectional IOHMMs

Temporal data can be modeled and processed using Markov models, such as Markov chains, HMMs, factorial HMMs, IOHMMs, Kalman filters, and so forth [5]. These models, which can be represented as graphical models (Bayesian networks) [34, 28, 23] with a characteristic left-right architecture, have been successfully used in many domains from speech to bioinformatics, for instance to model protein sequences and protein families. Biological sequences, however, are not temporal objects but rather spatial objects. This observation is crucial in bioinformatics applications, and has led to the introduction in the left-right Markov models of chains that run in the opposite direction from right to left, or from the future towards the past, in the form of bidirectional IOHMMs (Figure 2) [7]. Unlike plain IOHMMs [8, 12], bidirectional IOHMMs contain a directed path connecting any input to any output.

In its most simple form, a bidirectional IOHMMs is a Bayesian network consisting of a set of inputs I_i, outputs O_i, and discrete hidden states H_i^F (forward) and H_i^B (backward) with $i = 1, \ldots, N$ where N is the length of the sequence being processed. Each hidden variable has n states, an integer that controls model complexity. The parameters of the model correspond to three conditional probability distributions: an output distribution, a forward transition distribution (both also present in an IOHMM), and a transition distribution associated with the backward chain:

$$\left\{ \begin{array}{c} P(O_i|I_i, H_i^F, H_i^B) \\ P(H_i^F|I_i, H_{i-1}^F) \\ P(H_i^B|I_i, H_{i+1}^B) \end{array} \right. \tag{1}$$

All these conditional distributions are stationary, i.e. do not depend on the sequence index i (a form of parameters sharing).

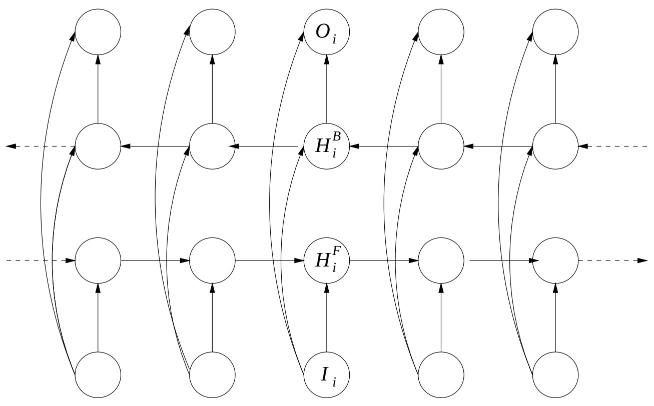

Figure 2: Bayesian network graphical model underlying BRNNs consisting of input units, output units, and both forward and backward Markov chains of hidden states.

Although inference in bidirectional IOHMMs takes time polynomial in n (state size) and N (sequence length), a deterministic version of the model, based on neural networks, is often preferred in real world applications because it is much less computationally demanding [6]. Details of these architecture will be given in Section 4. The neural network version of bidirectional IOHMMs have been extensively used in bioinformatics, and in particular for the first stage of the protein structure prediction pipeline described in the introduction giving rise to some of the best predictors for secondary structure, solvent accessibility, and coordination number. The corresponding predictor servers [36, 37, 5] are accessible at http://promoter.ics.uci.edu/BRNN-PRED/.

3 The General Case: Topological and Generalized IOHMMs

3.1 From 1D to 2D

To predict contact maps, however, the fundamental question is how can the idea of bidirectional IOHMMs be generalized from one dimensional to two dimensional objects? It turns out that there is a "canonical" 2D generalization described in Figures 3 and 4. In its basic version, the generalization consists of a Bayesian network organized into six horizontal layers or planes: one input plane, 4 hidden planes, and one output plane. Each plane contains N^2 nodes arranged on the vertices of a square lattice. Thus in each vertical column there is an input unit $I_{i,j}$, four hidden units $H^{NE}_{i,j}$, $H^{NW}_{i,j}$, $H^{SW}_{i,j}$, and $H^{SE}_{i,j}$ associated with the four cardinal corners, and an output unit $O_{i,j}$ with $i = 1, \ldots, N$ and $j = 1, \ldots, N$. In each hidden planes, the edges are oriented towards the corresponding cardinal corner. In the NE plane, for instance, all edges are oriented towards the North or the East. The parameters of this two-dimensional GIOHMMs are the conditional probability distributions:

$$\left\{ \begin{array}{c} P(O_i | I_{i,j}, H^{NE}_{i,j}, H^{NW}_{i,j}, H^{SW}_{i,j}, H^{SE}_{i,j,}) \\ P(H^{NE}_{i,j} | I_{i,j}, H^{NE}_{i-1,j}, H^{NE}_{i,j-1}) \\ P(H^{NW}_{i,j} | I_{i,j}, H^{NW}_{i+1,j}, H^{NW}_{i,j-1}) \\ P(H^{SW}_{i,j} | I_{i,j}, H^{SW}_{i+1,j}, H^{SW}_{i,j+1}) \\ P(H^{SE}_{i,j} | I_{i,j}, H^{SE}_{i-1,j}, H^{SE}_{i,j+1}) \end{array} \right.$$

again with the obvious adjustments at the boundaries. It is easy to check, and proven below, that this directed graph has no cycles hence it properly defines the support of a Bayesian network. Another variation on this approach is given in the Appendix.

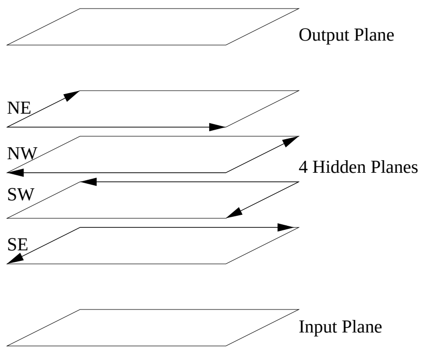

Figure 3: General layout of Bayesian network for processing two-dimensional objects such as contact maps, with nodes regularly arranged in one input plane, one output plane, and four hidden planes. In each plane, nodes are arranged on a square lattice. The hidden planes contain directed edges associated with the square lattices. All the edges of the square lattice in each hidden plane are oriented in the direction of one of the four possible cardinal corners: NE, NW, SW, SE. Additional directed edges run vertically in column from the input plane to each hidden plane, and from each hidden plane to the output plane.

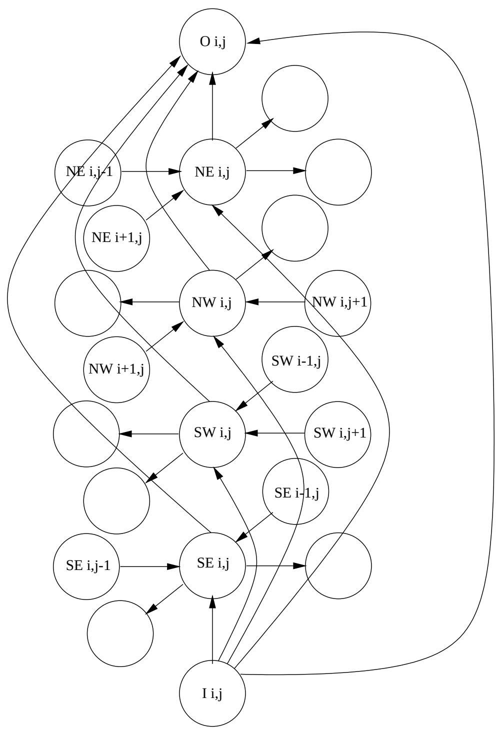

Figure 4: Details of connections within one column of Figure 3. The input unit is connected to the four hidden units, one in each hidden plane. The input unit and the hidden units are connected to the output unit. $I_{i,j}$ is the vector of inputs at position (i, j). $O_{i,j}$ is the corresponding output. Connections of each hidden unit to its lattice neighbors within the same plane are also shown.

3.2 D-Dimensional Case

It should by now be clear how to build canonical GIOHMMs for any dimension d. For instance in 3D, one has one input cube of units $I_{i,j,k}$, with eight cubes of hidden units $H^l_{i,j,k}$ (with $l = 1, \ldots, 8$, and one output cube $O_{i,j,k}$, with i, j and k ranging from 1 to N. In each hidden cubic lattice, edges are oriented towards one of the 8 corners. More generally, in d dimensions one would have 2^d hidden hypercubic lattices. In each hypercubic lattice, all connections would be directed towards one of the corners of the corresponding hypercube. While the 3D version of these architectures could be useful for 3D protein structure prediction problems, in its most simple form the 3D architecture does not address the problems of translation and rotation invariance. This is the main reason, for our current decomposition of the problems into three stages. Additional details can be found in [9].

3.3 Other GIOHMMs

In short, the most general definition of a GIOHMM is a graphical model that can be used to describe a probabilistic input-output mapping between data structures [18, 9] and consisting of a set of N input nodes, M output nodes, and a DAG hidden layer (possibly with multiple connected components), with additional connections running from the input to the output nodes, and from the input nodes to the hidden nodes.

Generalized HMMs can be defined in a similar way, just by removing the input nodes.

GHMMs can be employed for density estimation in structured spaces of trees or graphs. In [14], models specialized for trees have been applied to the classification of documents.

4 Contact Map Prediction As a Graph Search Problem

The second set of methods we have developed is based on two basic ideas: (a) using GIOHMM architectures to learn a graph scoring function; (b) using the graph scoring function to efficiently search the space of possible contact maps. While the methods is very general and can be applied to different classes of graphs, here we focus the analysis and simulations on the coarse contact map of proteins, i.e. the contact map derived at the level of secondary structure elements.

Let $G^\star = (V, E^\star)$ denote the target contact map (an undirected graph). For every candidate map $G = (V, E)$, let $s(E|V)$ be a scoring function taking values in $[0, 1]$ and having the following properties:

1. $s(E|V) = 1$ iff $E = E^\star$;
2. $s(E|V) = 0$ iff (V, E) is the null graph
3. if $E \subset E^\star$ and $s(\{e^\star\} \cup E) = \max_{e \in V^2 \setminus E} \{s(\{e\} \cup E)\}$, then $e^\star \in E^\star$.

If $s(E|V)$ was known, one could easily conceive a heuristic search procedure for finding the correct contact map. The search algorithm takes the vertex set as input and starts by assigning the null graph to G. The main loop essentially consists of three basic operations: generation (all G's successor are generated by applying allowable graph edit operators), evaluation (G' s successors are scored using the function $s(E|V)$), and update (G replaced by a new graph closer to the solution). The algorithm terminates when $s(E|V) = 1$ and $G = G^\star$. Note that because of property 3, only one type of edit operator is required, namely the operator that adds one edge $(u, v)$to E, if (u, v) is not already in E. A hill-climbing version of the algorithm is given in pseudo code (see Figure 5).

The above algorithm maintains the following loop invariant: before the main loop (lines 2–12), E is a subset of $E^\star$. The invariant follows from our assumptions on the scoring function and can be easily proved by induction. At the end of the loop, $s(E|V) = 1$ and therefore, by property 2 of the scoring function, $E = E^\star$. To analyze the algorithm we observe that each iteration requires the evaluation of $s(E|V)$ for each candidate edge, i.e. $O(N^2)$ times, being $N = |V|$. Since the loop 2-12 in SEARCH-CMAP is executed $(|E^\star|$ times, $s(E|V)$ is evaluated $O(|E^\star|N^2)$ times. But each evaluation takes time linear in N, and thus if we assume that $|E^\star| = O(|V|)$ it follows that SEARCH-CMAP is a polynomial algorithm and takes time $O(N^4)$. The apparent simplicity of the overall procedure for finding $G^\star$ is clearly due to the existence of an oracle that can compute $s(E|V)$ for every candidate map. In the following we first suggest a suitable scoring function and then we propose a neural network model capable of learning $s(E|V)$ from examples of successful searches.

4.1 Scoring Function

For each map $G = (V, E)$, let us introduce precision and recall of E (relative to $E^\star$) as follows:

$$P(E) = \frac{|E \cap E^\star|}{|E|} \tag{2}$$

```
SEARCH-CMAP($V$)
 1  $E \leftarrow \emptyset$
 2  repeat
 3        $m \leftarrow 0$
 4        for each $v$ in $V$
 5        do for each $u$ in $V - \{v\}$
 6           do if $\neg(u,v) \in E$
 7                then $a \leftarrow$ SCORE($E \cup \{(u,v)\} | V$)
 8                     if $a > m$
 9                       then $m \leftarrow a$
10                            $e \leftarrow (u,v)$
11        $E \leftarrow E \cup \{e\}$
12    until $m = 1$
13  return $E$
```

Figure 5: Algorithm for finding the correct contact map given a perfect scoring function.

$$R(E) = \frac{|E \cap E^\star|}{|E^\star|} \tag{3}$$

Precision (eq. 2) is the fraction of edges in the predicted set E that are correctly assigned (they are also present in $E^\star$). Recall (eq. 3) is the fraction $E^\star$'s edges that have been correctly discovered, i.e. they are also present in E. It can be easily verified that the function

$$s^\star(G) = \frac{2P(E)R(E)}{P(E) + R(E)} \tag{4}$$

satisfies properties 1-3 and therefore it is a suitable metric for guiding the heuristic search. We note that this function is also known as the F_1 metric in information retrieval.

4.2 Learning the Scoring Function

The scoring function in Eq. 4 depends on $E^\star$, which is unknown. We now introduce a machine learning method for approximating $s^\star(G)$. In this approach, we suggest using a GIOHMM where input nodes are the elements in V (possibly enriched with several attributes), while the hidden layer topology reflects the candidate map E. In the case of fine-grained maps, elements in V would be amino acid symbols. In the case of coarse maps, elements in V are secondary structure elements and they can be described by a set of numerical and categorical attributes, such as length and position within the sequence, secondary structure category, and physic-chemical properties (e.g. related to the average exposure to solvent).

One fundamental issue is the generation of the training set. For each sequence of length N there are $2^{N(N-1)/2}$ possible distinct contact maps (including the null and the complete graph). Clearly, using all these graphs as training examples is not realistic and a subsampling strategy is required. The training set generation can be either *static* (i.e., examples are selected before training begins) or *dynamic* (i.e. examples are inserted and deleted as training proceeds).

In the static case, we note that random selection of graphs for each protein would be extremely unlikely to yield a balanced dataset (the probability of guessing a random graph with high score decreases exponentially with N). A simple strategy that guarantees a reasonable balance between high and low score graphs is to run algorithm SEARCH-CMAP guided by $s^{\star}(G)$ (Eq. 4) and collect as training examples all the $O(|V|^3)$ graphs that are generated during the search. One disadvantage of the static strategy is that after training, the learner is specialized in a relatively narrow region of the search space and repeated errors may drive the search algorithm far away from the goal. In order to mitigate this problem, the hill-climbing procedure in algorithm SEARCH-CMAP can be changed into a beam search. Unlike hill-climbing, that keeps at each stage the best candidate only, beam search maintains a bounded open list of size B. The open list is filled at each stage with the best B candidates, selected from all the possible successors of graphs in the open list at the previous stage.

As an alternative to a static selection of examples, we propose an online strategy where examples are generated on the fly as while network is being trained. Methods that implement state space exploration have been suggested in the reinforcement learning literature [41, 40] as practical ways for dynamically sampling training instances. These methods would be too costly for implementing a dynamic training scheme in the present contest. We propose the following simplification. During the learning phase, for a given sequence V the learner is asked to follow a particular trajectory from the null graph to a final graph according to a given policy. A policy is a mapping from states to actions where a state is the undirected graph of a candidate contact map and an action is an edit operator that adds one edge. Various exploration strategies can be implemented in order to investigate the effects of the exploration-exploitation trade-off, which is well-known in reinforcement learning [41, 40]. In *random exploration*, we choose the successor graph randomly with uniform probability. In *pure exploitation*, we select the successor graph that has maximum score, as predicted by the current network. In *semi-uniform* exploration (also known as ϵ-greedy policy), with probability ϵ we make a random uniform choice from the set of successors, and with probability $1-\epsilon$ we choose the maximum score graph. As we will see later, different strategies may significantly vary the tradeoff between precision and recall (Eqs. 2 and 3).

4.3 A Bi-Recursive Topology

The framework for data structures presented in [18] cannot be applied directly since it requires the input graph be directed, ordered, and acyclic, while contact maps are undirected, unordered and possibly cyclic graphs. The extension we propose here is based on forward-backward state space factorization (like in the bidirectional topologies described in earlier sections) in order to avoid message propagation along directed cycles. Factorization is possible thanks to the serial order relation $\prec$ defined on G's vertices, allowing us to interpret G as a pair of directed acyclic graphs: the forward (from N- to C-terminus) graph G_f, having $E_f = \{(u,v) \in E : u \prec v\}$, and its transpose backward graph G_b. Figure 6 shows an example of the architecture. Each node v in the input graph is labeled with a fixed-length tuple $I(v)$. Forward and backward hidden states, $H_f(v)$ and $H_b(v)$, respectively, are linked according to the topologies of G_f and G_b.

The architecture shown in Figure 6b has a single output O, connected to the two extreme hidden variables (at the N terminus for the f hidden layer and at the C terminus for the b hidden layer). This network is supposed to be trained in regression mode, so that the output O approximates $s^{\star}(G)$ as closely as possible. This solution, however, is not necessarily effective because input nodes are connected to the output through long paths, that may introduce

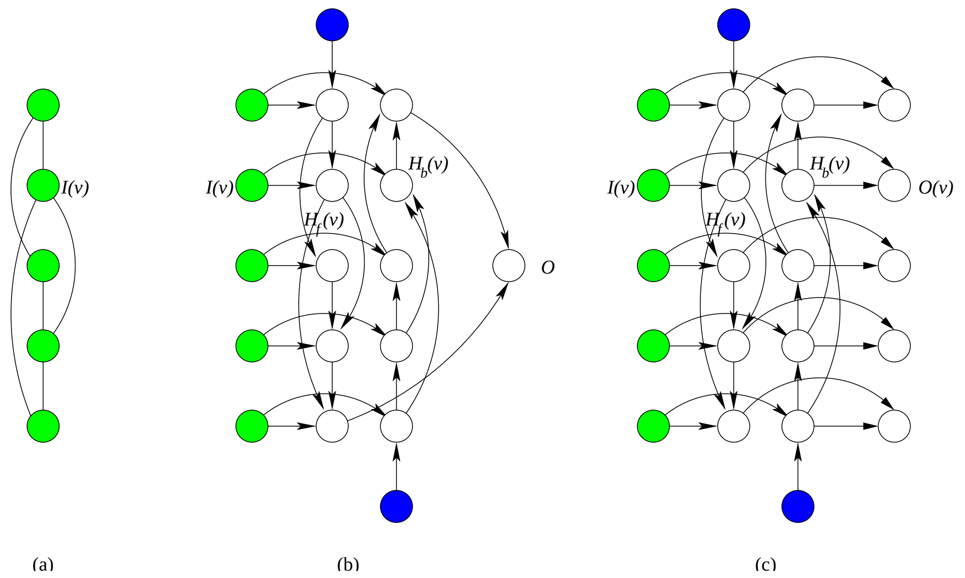

Figure 6: (a): a sample graph G; (b): graphical model for the bi-recursive network, consisting of input nodes, output node, and two sets of hidden states, linked according to forward and backward graphs G_f and G_b; (c): bi-recursive network with one output for each node v.

problems associated with long term dependencies [11]. Therefore, we suggest to introduce an output node for each position v and to decompose the scoring function into a set of local scoring functions (see Figure 6c). More precisely, let $E_v = \{(u, w) \in E : w = v\}$ and $E^\star_v = \{(u, w) \in E^\star : w = v\}$ denote the subsets of edges incident on vertex v respectively in G and $G^\star$. Local precision, recall and F_1 measure of E_v (relative to $E^\star_v$) are defined as follows:

$$P(E_v) = \frac{|E_v \cap E^\star_v|}{|E_v|} \tag{5}$$

$$R(E_v) = \frac{|E_v \cap E^\star_v|}{|E^\star_v|} \tag{6}$$

$$s^\star(E_v) \doteq \frac{2P(E_v)R(E_v)}{P(E_v) + R(E_v)} \tag{7}$$

The following formula computes the score of G as a weighted sum of the local scores computed at node level:

$$\tilde{s}(E|V) = \frac{1}{|V|}\sum_{i=1}^{|V|} s^\star(E_{v_i}) \tag{8}$$

We found that $\tilde{s}(E|V)$ is a good locally based representation of $s^\star(G)$ (their Pearson correlation coefficient is very close to 1). The inference process of our model computes the sequence of output values $\{O(v_1), \ldots, O(v_n)\}$ and predicts G's score as:

$$\tilde{s}_{net}(E|V) = \frac{1}{|V|}\sum_{i=1}^{|V|} O(v_i) \tag{9}$$

In this approach, each output unit is trained in regression mode to approximate $s^\star(E_{v_i})$, for $i = 1, \ldots, n$.

5 Neural Network Architectures

5.1 Replacing Recursive Bayesian Networks with Recursive Neural Networks

As described, GIOHMMs are Bayesian networks and, at least in principle, the general propagation and learning algorithms for Bayesian networks can be applied to them [28, 24]. In practice, however, this approach is not convenient or feasible because of excessive computational demands or computational intractability of the inference step. Learning algorithms for Bayesian networks in the presence of hidden variables require iterative approaches such as EM or gradient descent. In these algorithms, inference is called as a subroutine and thus its computational cost is particularly critical.

For example, let us consider bidirectional IOHMMs. Inference in this case is computationally tractable and the algorithms can be easily derived as special forms of belief propagation. However, the junction tree associated with the network in Figure 2 has cliques with triplets of state variables (this can be easily seen by moralization and triangulation of the network). As a consequence, the complexity of inference scales up as $O(Nn^3)$, being n the number of (forward and backward) discrete states, and N the sequence length. In order to store enough information about the upstream and downstream regions with respect to any position i, a large number of states must be used, leading to an excessive computational burden for training.

The case of multi$D$-dimensional GIOHMMs described in Section 3 is unfortunately much worse, since belief propagation is intractable for $D \geq 2$ (for example, it can be easily seen that triangulation applied to a regular 2D grid yields cliques of exponential size). Approximate inference algorithms (such as those based on variational methods [26]) might help in this case. The totally different approach we adopt in the following is to devise efficient neural network versions, in which the graphical formalism is retained but with very different semantics. While missing arcs in Bayesian networks encode probabilistic conditional independence, existing arcs in the corresponding neural network encode deterministic functional dependencies. The general method for downcasting a Bayesian network $\mathbf{B}$ to a corresponding neural network $\mathbf{N}$ is therefore based on the following steps:

- $\mathbf{N}$ and $\mathbf{B}$ are described by the same graph. However, discrete variables in $\mathbf{B}$ are replaced by real vectors in $\mathbf{N}$.
- For each non-input node (variable) X in $\mathbf{B}$, the conditional probability table $P(X|\text{Pa}[X])$ becomes a deterministic but adaptive function $X = \mathcal{N}(\text{Pa}[X])$, implemented by a feedforward neural network (e.g. a multilayered perceptron).
- Stationarity is maintained: whenever two conditional probability tables are identical in $\mathbf{B}$, the corresponding neural networks in $\mathbf{N}$ have shared weights.

In so doing, computational complexity is dramatically reduced because inference in $\mathbf{N}$ consists of forward propagation of input signals towards the outputs, following any topological sort of the DAG. Learning can then be achieved using gradient descent, amounting to back-propagation through unfolded space or structure [18, 19].

5.2 Bidirectional RNNs

To be more precise, consider for example the case of one-dimensional bidirectional IOHMMs and the problem of protein secondary structure prediction [37]. Letting i denote position within a sequence, the overall model outputs for each i a probability vector O_i representing the membership probability of the residue at position i in each one of the three classes (alpha/beta/coil). This output is implemented by three normalized exponential output units. The output prediction has the functional form:

$$O_i = \mathcal{N}_O(I_i, H_i^F, H_i^B) \tag{10}$$

where H_i^F denotes the vector of activities associated with the hidden node H_i^F, and similarly for H_i^B. The output depends on the local input I_i at position i, the forward (upstream) hidden context H_i^F, and the backward (downstream) hidden context H_i^B. The vector $I_i \in I\!R^k$ encodes the external input at position i. In the most simple case, where the input is limited to a single amino acid, $k = 20$ by using orthogonal encoding. Larger input windows extending over several amino acids are also possible. The learnable output function is realized by a neural network $\mathcal{N}_O$ (see center and top connections in Figure 7). In a regression task, the performance of the model can be assessed using the usual mean square error. In a multinomial classification task, such as secondary structure prediction, the performance of the model is better assessed using the relative entropy between the estimated and the target distribution. The contextual information contained in the vectors $H_i^F \in I\!R^n$ and $H_i^B \in I\!R^m$ (with usually $m = n$). These satisfy the recurrent bidirectional equations:

$$\begin{array}{rcl} H_i^F & = & \mathcal{N}_F(I_i, F_{i-1}) \\ H_i^B & = & \mathcal{N}_B(I_i, B_{i+1}) \end{array} \tag{11}$$

Here $\mathcal{N}_F(\cdot)$ and $\mathcal{N}_B(\cdot)$ are learnable non-linear state transition functions, implemented by two NNs, $\mathcal{N}_F$ and $\mathcal{N}_B$ (left and right subnetworks in Figure 7). The boundary conditions for H_i^F and H_i^B are set to 0, i.e. $H_0^F = H_{N+1}^B = 0$ where N is the length of the sequence being examined. Intuitively, we can think of H_i^F and H_i^B as "wheels" that can be rolled along the protein. To predict the class at position i, we roll the wheels in opposite directions from the N and C terminus up to position i and then combine what is read on the wheels with I_i to calculate the proper output using $\mathcal{N}_O$. All the weights of the BRNN architecture, including the weights in the recurrent wheels, can be trained in a supervised fashion from examples by a generalized form of gradient descent or backpropagation through time, by unfolding the wheels in time, or rather space. Architectural variations can be obtained by changing the size of the input windows, the size of the window of hidden states considered to determine the output, the number of hidden layers, the number of hidden units in each layer and so forth. In these general architectures for sequence translation, translation or prediction at a given position depends on a combination of local information, provided by a standard feedforward neural network, and more distant context information. Learning is by backpropagation through space/time.

5.3 2D lattice RNNs

It should be clear how to immediately apply the same ideas to the case of 2D lattice GIOHMMs and other GIOHMMs. Here the output and the hidden layer propagations are parameterized by 5 neural networks in the form

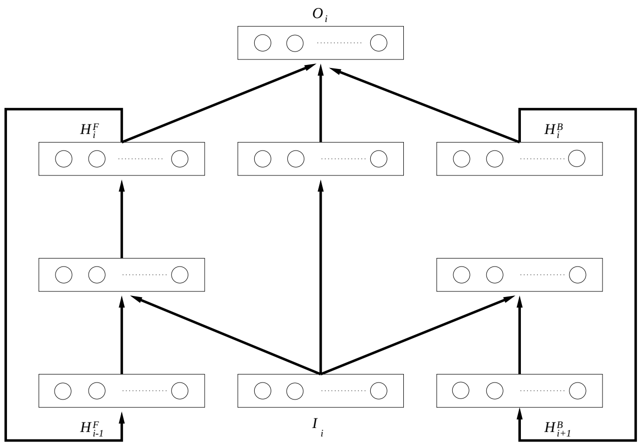

Figure 7: A BRNN architecture with a left (forward) and right (backward) context associated with two recurrent networks (wheels). Connections from input to wheels are not shown.

$$\left\{ \begin{array}{r} O_{ij} = \mathcal{N}_O(I_{ij}, H_{i,j}^{NW}, H_{i,j}^{NE}, H_{i,j}^{SW}, H_{i,j}^{SE}) \\ H_{i,j}^{NE} = \mathcal{N}_{NE}(I_{i,j}, H_{i-1,j}^{NE}, H_{i,j-1}^{NE}) \\ H_{i,j}^{NW} = \mathcal{N}_{NW}(I_{i,j}, H_{i+1,j}^{NW}, H_{i,j-1}^{NW}) \\ H_{i,j}^{SW} = \mathcal{N}_{SW}(I_{i,j}, H_{i+1,j}^{SW}, H_{i,j+1}^{SW}) \\ H_{i,j}^{SE} = \mathcal{N}_{SE}(I_{i,j}, H_{i-1,j}^{SE}, H_{i,j+1}^{SE}) \end{array} \right. \tag{12}$$

In the NE plane, for instance, the boundary conditions are set to $H_{ij}^{NE} = 0$ for $i = 0$ or $j = 0$. The activity vector associated with the hidden unit H_{ij}^{NE} depends on the local input I_{ij}, and the activity vectors of the units $H_{i-1,j}^{NE}$ and $H_{i,j-1}^{NE}$. Activity in NE plane can be propagated row by row, West to East, and from the first row to the last (from South to North), or column by column South to North, and from the first column to the last. These updates schemes are easy to code, however they are not the most continuous since they requires a jump at the end of each row or column (a toroidal architecture may be considered but it would contain directed cycles). A more continuous update scheme is a zig-zag scheme that would run up and down successive diagonal lines oriented SE to NW. Depending on software and hardware implementation or embodiment details, such a scheme could be slightly faster or smoother during online learning.

It is worth noting that if in the homogeneous GIOHMMs described so far activity propagates simultaneously in the hidden DAGs from the source nodes to the sink nodes, then the nodes in the center are the first ones for which propagation in all the hidden DAGs converges. In other words, correct output value stabilize from the center towards the periphery.

5.4 Bi-Recursive Topology

When using a neural network implementation of the dependencies implied by the graphical model shown in Figure 6, the following equations describe the inference dynamics:

$$H_f(v) = \mathcal{N}_f(I(v), H_f(\text{Pa}[v])) \tag{13}$$
$$H_v(v) = \mathcal{N}_b(I(v), H_b(\text{Pa'}[v])) \tag{14}$$

where Pa$[v]$ denotes the parents of v in G_f and Pa'$[v]$ denotes the parents of v in $G_b = G'_f$. Note that, as in the models presented in [18], an upper bound B on the indegree of each hidden node must be assumed. This also means that the data structures that can be processed by these network must have bounded connectivity. If a node has less than B parents (in either subgraph) the corresponding entries in Pa$[v]$ (or Pa'$[v]$) are set to zero. Also, a total order on the parent sets Pa$[v]$ and Pa'$[v]$ is necessary to properly construct the argument list of functions $\mathcal{N}_f$ and $\mathcal{N}_b$. In the present case, this total order is inherited from the serial order on V. Propagation algorithms are straightforward once we recognize that the overall architecture has no cycles.

5.5 Generalization

More generally, consider a connected DAG in the hidden layer where each node has at most K inputs. The nodes that have strictly less than K inputs are called boundary nodes. In particular each DAG has at least one source node, with only outgoing edges, and any source node is a boundary node. For each boundary node i with $l < K$ inputs, we add $K - l$ distinct input nodes, called boundary condition nodes. For source nodes, K boundary condition nodes must be added. After this pre-processing step, the hidden DAG is regular in the sense that all the nodes have exactly K inputs, with the exception of the boundary condition nodes that have become source nodes.

We can now parameterize the corresponding Bayesian network using a neural network that is shared among all nodes. The network has a single output vector corresponding to the activity of a node, and K input vectors. The dimension of each vector can vary. The vectors associated with the boundary nodes are set to the 0 vector, matching the dimensions properly.

Propagation of activity proceeds from the boundary nodes towards the sink nodes. There may be multiple sink and sources, as evidenced by propagation in the hidden trees in the case of tree structures. The fact that the graph is a DAG together with the boundary conditions ensures that there is a consistent order of update of all the nodes. This topological order may not be unique as in the case of 2D lattice, or tree structures.

5.6 Learning

Gradient descent for recurrent networks [4] can be extended for learning in recursive neural networks. In particular, the unfolding procedure and the associated gradient computation described in [19] easily extends to GRNNs, enabling the use of propagation algorithms for gradient computation. In practice however, it is not always trivial to get gradient descent learning procedures to work well in recurrent networks: error gradients can vanish rapidly as a function of time [11] and learning procedures can become stuck in poor local minima [13]. Another important factor in the architectures we are considering is the competition/collaboration tradeoff between the hidden DAGs and their NN equivalents. Especially in homogeneous GRNNs where all the hidden components are equivalent, when the system is initialized with small weights there is an inherent symmetry that needs to be broken. Algorithmic details to address these issues can be found in [9].

6 Simulations

6.1 Data

Fine-grained maps The training and testing data sets used here were extracted from the PDB_select list [25] of February 2001, containing 1520 proteins. The list of structures and additional information can be obtained from the following ftp site:
ftp://ftp.embl-heidelberg.de/pub/databases. To avoid biases, the set is redundancy-reduced, with an identity threshold based on the distance derived in [1], which corresponds to a sequence identity of roughly 22% for long alignments, and higher for shorter ones. This set is further reduced by excluding those chains whose backbone is interrupted. To extract 3D coordinates, together with secondary structure, solvent accessibility and information on beta-sheet partners, we run Kabsch and Sander's DSSP program [27] (CMBI version) on all the PDB files in the PDB_select list, and excluded the ones on which DSSP crashed due, for instance, to missing entries, erroneous entries, or format errors. The final set consists of 1484 proteins. For training and testing purposes, a subset containing only proteins of length up to 100 was also constructed. This set contains 533 proteins and over 2.3 million pairs of amino acids.

It is essential to notice that contact maps depend strongly on the selection of a distance cutoff. Furthermore, the composition of the data is in general strongly biased in favor of non-contacts. Typically, a contact map of size N^2, contains a number of contacts that is linear in N. To test the effect of various distance thresholds on contacts, thresholds of 6, 8, 10 and 12 Åwere selected for contact classification, yielding four different classification tasks. The number of pairs of amino acid in each class and for each contact cutoff is given in Table 1.

Table 1: Data set composition, with number of pairs of amino acids that are separated by less (close) or more (far) than the distance thresholds in Ångstroms.)

	6Å	8Å	10Å	12Å
far	2125656	2010198	1825934	1602412
close	202427	317885	502149	725671
total	2328083	2328083	2328083	2328083

Coarse-grained maps Contacts in coarse maps are associated with the intuitive spatial "neighborhood" concept between two elements of secondary structure. However, the "neighborhood" relation is not uniquely defined. In our preliminary studies we considered two alternative definitions. According to the first definition, two secondary structure elements are in contact if the distance between their centers of gravity falls below a given cutoff. In the second definition, two elements are in contact if there are any two C_α atoms not belonging to the same element whose distance is below a given cutoff. In addition, we also tested another possible definition considering an average distance between the projections of the principal axes of secondary structure segments convex hulls. After preliminary attempts we did not find major differences. We also noted that a segment contact threshold fixed at 8Åis related to more than 20% of amino acids not in the same segment being in contact. In all the experiments reported below we adopted for the definition of contact as if the distance between any two C_α atoms not in the same segment is less than 8Å.

In the case of coarse maps, we used a significant fraction of the current representative set of non homologous protein data bank chains (PDB Select, [25]). We extracted the chains in the file 2001_Sep.25 (accessible at `ftp://ftp.embl-heidelberg.de/pub/databases/`) listing 1544 proteins (1641 chains) with percentage of homology identity less than 25%. From this set we retained only high quality proteins on which the DSSP program [27] (CMBI version) does not crash, determined only by X-ray diffraction (not multiple NMR models), without any physical chain breaks and resolution threshold less than 2.5 Å. From the filtered set we retained the proteins with sequence length less than 300 amino acids, resulting in a database of 587 proteins. The DSSP program was also used to assign secondary structure categories. The automatic assignments were projected to the three secondary structure states alpha, beta and gamma, with the following criteria: H maps to alpha, E maps to beta and the rest to gamma. Resulting segments with only one amino acid were discarded. As previously noted, working at the segment level results in significant dimensionality reduction. For example in a subset of 2000 PDB sequences with low similarity, the average segment length is 7.12 residues, thus the size of the coarse map is, on average, roughly 2% of the size of the residue resolution map.

6.2 Inputs

In the contact map prediction, one obvious input at each (i, j) location is the pair of corresponding amino acids. Amino acids can be represented using orthogonal encoding, i.e. vectors of length 20 with a single component set to one and all the others to zero. In this case the input has 20 components. However, other structural inputs can be added. It is reasonable to expect that relative solvent accessibility, a percentage indicator of whether a residue is on the surface or buried in the hydrophobic core of a globular protein could be relevant. Likewise secondary structure categories can also be included. The value of these indicators is close to exact when obtained on PDB training data, but would be noisier when estimated from a secondary structure or accessibility predictor.

A second type of input consideration is the use of profile and correlated mutation ideas [21, 33, 31, 17]. Profiles, essentially in the form of alignment of homologous proteins, implicitly contain evolutionary and structural information about related proteins. This information is relatively easy to collect using well-known alignment algorithms that ignore 3D structure and can be applied to very large data sets of proteins, including many proteins of unknown structure. The use of profiles improves the prediction of secondary structures by several percentage points, probably because secondary structure is more conserved than the amino acid sequence. As in the case of secondary structure, the input could be modified to include the profile vector at position i and a profile vector at position j, yielding two 20-dimensional probability vectors-an input with 40 numerical components.

When a distant pair $i, j)$ of positions in multiple alignment is considered, however, horizontal correlations in the sequences may exist that result entirely from 3D structural constraints. Such horizontal correlations are entirely lost if each profile is entered independently. Thus an expanded input, which retains this information, consists of a 20×20, and generally sparse matrix corresponding to the probability distribution over all pairs of amino acids observed in the two corresponding columns of the alignment. A typical alignment will contain a few dozen sequences and therefore the matrix in general is very sparse. The unobserved entries can be set to 0 or regularized to some small values using a standard Dirichlet prior approach [5].

While this has not been attempted here, larger input could be considered where correlations are extracted not only between positions i and j but also with respect to their neighborhoods including, for instance $i-1$, $i+1$, $j-1$, and $j+1$. While this can compensate for small alignment errors, they also rapidly lead to intractably large inputs that scale like 20^{2k}, where k is the size of the neighborhood considered. Compression techniques using weight sharing and/or higher-order neural networks would have to be used in conjunction with these very expanded inputs.

In the experiments reported, we use inputs of size $|I| = 20$ (just the two amino acids or the two profiles), as well as size $|I| = 440$ (the correlated mutation profile, plus the secondary structure and solvent accessibility of each position).

6.3 Architectures

In the simulations, we consider first the problem of predicting protein contact maps at the amino acid level. We use a 2D GIOHMM approach with four hidden plane lattices with diagonal edges associated with four similar but independent neural networks. In each neural network we use a single hidden layer. Thus, a given architecture is described by three key parameters: (1) the number NHO of hidden units in the output neural network; (2) the number NHH of hidden units in each of the four hidden neural networks associated with lateral propagation in each of the four planes; and (3) the number NOH of units in the output layer of the four hidden neural networks, corresponding to the dimension of the vector encoding a hidden state in each of the four hidden planes. While we have experimented with several architectures, the results we report are for $NHH = NHO = NOH = 8$ corresponding to 17,114 parameters when an input of size 20×20 is used.

6.4 Learning and Initialization

Training is implemented on-line, by adjusting all the weights after the complete presentation of each protein. A stronger version of on-line training that updates weights after the presentation of each (i, j) seems both unnecessary and inefficient. In the experiments reported below, we trained the networks using half the data and tested on the remaining half. With protein sequences of length less than 100, the training set had 266 sequences and the test set 267. We used a piecewise linear learning step [9], with a learning rate η equal to 0.1 divided by the number of protein examples (266). Prior to learning, the weights of each unit in the various neural networks are randomly initialized. The standard deviations, however, must be controlled in a flexible way to avoid any bias and ensure that the expected total input into each unit is roughly in the same range.

6.5 Results on Amino Acid Contact Maps

Results of contact map predictions at four distance cutoffs are provided in Table 2. In this experiment, the system is trained on half the set of proteins of length less than 100, and tested on the other half. These results are obtained with plain sequence inputs (amino acid pairs), i.e. without any information about profiles, correlated mutations, or structural features. For a cutoff of $8\AA$, for instance, the system is able of recovering 62.3% of contacts.

It is of course essential to be able to predict contact maps also for longer proteins. This can be attempted by training the recurrent neural network architectures on larger data sets,

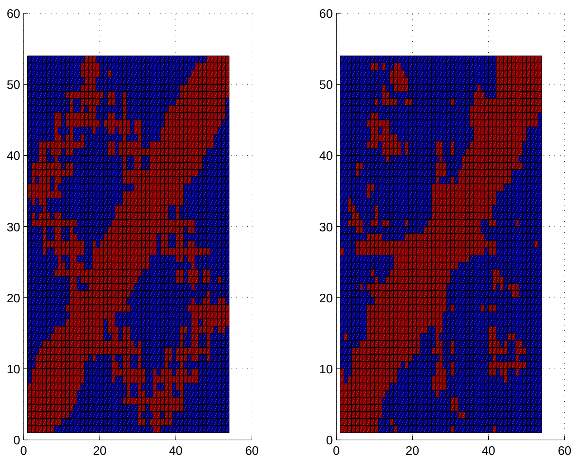

Figure 8: Example of exact (left) and predicted contact map for protein 1DEEH, prior to symmetrization of the prediction. Color code: blue = 0 (non-contact), red =1 (contact).

Table 2: Percentages of correct predictions for different contact cutoffs on the validation set. Model trained and tested on proteins of length less than 100. Inputs correspond to simple pairs of amino acids in the sequence (without profiles, correlated profiles, or structural features).

	6Å	8Å	10Å	12Å
Far	99.1%	98.9%	97.8%	96.0%
Close	66.5%	62.3%	54.2%	48.1%
All	96.2%	93.9%	88.6%	81.0%

containing long proteins. While such experiments are in progress, it should be noted that because the systems we have developed can accommodate inputs of arbitrary lengths, we can still use a system trained on short proteins $(l \leq 100)$ to produce predictions for longer proteins. In fact, because the overwhelming majority of contacts in proteins are found at linear distances shorter than 100 amino acids, it is reasonable to expect a decent performance from such a system. Indeed, this is what we observe in Table 3. At a cutoff of $8\AA$, the percentage of correctly predicted contacts for all proteins of length up to 300 is still 54.5%.

Table 3: Percentages of correct predictions for different contact cutoffs on the validation set. Model trained on proteins of length less than 100, but tested on all proteins with length up to 300. Inputs correspond to simple pairs of amino acids in the sequence.

	6Å	8Å	10Å	12Å
Far	99.6%	99.6%	99.2%	97.8%
Close	64.5%	54.5%	45.7%	39.9%
All	98.3%	96.8%	93.7%	88.6%

A typical example of prediction is reported in Figure 8. In this example we display the raw output of the network which is *not* symmetric since symmetry constraints are not enforced during learning. A symmetric output is easy to derive from a non-symmetric output by averaging the output values at positions (i, j) and (j, i). Application of this averaging procedure yields a small improvement in the overall prediction performance, as seen in Table 4. A

Table 4: Same as Table 3 but with symmetric prediction constraints.

	6Å	8Å	10Å	12Å
Far	99.1%	98.9%	97.8%	96.0%
Close	67.3 (+0.8)%	63.1 (+0.8)%	54.9 (+0.7)%	49.0 (+0.9)%
All	96.3 (+0.1)%	94.0 (+0.1)%	88.7 (+0.1)%	81.3 (+0.3)%

possible alternative is to enforce symmetry during the training phase.

Table 5: Percentages of correct predictions for different contact cutoffs on the validation set. Model trained and tested on proteins of length less than 100. Inputs of size 20×20 correspond to correlated profiles in the multiple alignments derived using the PSI-BLAST program.

	6Å	8Å	10Å	12Å
Far	99.0%	98.9%	97.6%	96.1%
Near	67.9%	63.0%	55.3%	49.4%
All	96.3%	94.0%	88.5%	81.5%

Table 6: Percentages of correct predictions for different contact cutoffs on the validation set. Model trained and tested on proteins of length less than 100. Same as Table 5 but inputs include also secondary structure and relative solvent accessibility at a threshold of 25% derived from the DSSP program. Last row represents standard deviations on a per protein basis.

	6Å	8Å	10Å	12Å
Far	99.6%	99.5%	98.5%	95.3%
Near	73.8%	67.9%	58.1%	55.5%
All	97.3%	95.2%	89.8%	82.9%
Std	2.3%	3.7%	5.9%	8.5%

The results of additional experiments conducted with larger inputs are displayed in Tables 5 and 6. When inputs of size 20×20 corresponding to correlated profiles are used, the performance increases marginally by roughly 1% for contacts (for instance, 1.4% at 6Å and 1.3% at 12Å) (Table 5). When both secondary structure and relative solvent accessibility (at 25% threshold) are added to the input, however, the performance shows a remarkable further improvement in the 3-7% range for contacts. For example at 6Å contacts are predicted with 73.8% accuracy. The last row of Table 6 provides the standard deviations of the accuracy on a per protein basis. These standard deviations are reasonably small so that most proteins are predicted at levels close to the average. These results support the view that secondary structure and relative solvent accessibility are very important for the prediction of contact maps and more useful than profiles or correlated profiles. This is also confirmed by the results obtained by this model (trained on short proteins with correlated profile inputs augmented by structural features) when tested on proteins of length up to 300 (Table 7). At an 8Å cutoff, the model still predicts over 60% of the contacts correctly, achieving state-of-the-art performance above any previously reported results. In terms of off-diagonal prediction, the sensitivity for amino acids satisfying $|i - j| \geq 7$ is 0.27 at $8\AA$ and 0.45 at $10\AA$, to be contrasted with 0.21 at $8.5\AA$ reported in [17]. Finally, a further small improvement can be derived by combining the output of the four predictors using another similar architecture with $NHH = NOH = NHO = 5$ (Table 8) trained on the same data since no overfitting is detected. The global improvement is most visible at $12\AA$ with a 0.5% improvement over Table 6. But even at $6\AA$, there is a non-trivial 0.4% improvement on the prediction of contacts.

Table 7: Percentages of correct predictions for different contact cutoffs on the validation set. Model trained on proteins of length less than 100, but tested on all proteins with length up to 300. Inputs include correlated profiles, secondary structure, and relative solvent accessibility (at 25%).

	6Å	8Å	10Å	12Å
Far	99.9%	99.9%	99.3%	95.1%
Near	70.7%	60.5%	51.3%	49.2%
All	98.8%	97.5%	94.4%	87.8%

Table 8: Percentages of correct predictions for different contact cutoffs on the validation set obtained by a network combining four predictors trained on each distance cutoff. Model trained on proteins of length less than 100 and tested on proteins with length up to 100. Inputs include correlated profiles, secondary structure, and relative solvent accessibility (at 25%).

	6Å	8Å	10Å	12Å
Far	99.6%	99.0%	97.4%	96.5%
Near	74.1%	70.8%	62.1%	54.8%
All	97.3%	95.2%	89.8%	83.4%

6.6 Results on Coarse Contact Maps

The method detailed in Section 4 have been tested for the prediction of protein contact maps at the coarse level. Eight numerical features encode the input label of each node and comprise one-hot encoding of secondary structure type; normalized linear distances from the N to C terminus; average, maximum and minimum hydrophobic character of the segment (based on the Kyte-Doolittle scale on moving 7-length window centered at all residues positions in the segment). Note that the network model in its present implementation uses as input a minimal amount of biologically significant information.

Table 9: Graph search with dynamic sampling: summary of experimental results. We report micro-averaged precision, recall, and F_1, denoted mP, mR, and mF_1, respectively. The corresponding macro averages are denoted MP, MR, and MF_1; $mP(nc)$ and $MP(nc)$ are the micro- and macro-averaged precisions in predicting non contacts.

Sampling strategy	mP	$mP(nc)$	mR	mF_1	MP	$MP(nc)$	MR	MF_1
Random exploration	.715	.769	.418	.518	.767	.709	.469	.574
Semi-uniform exploration	.454	.787	.631	.526	.507	.767	.702	.588
Pure exploitation	.431	.806	.726	.539	.481	.793	.787	.596
Hybrid	.417	.834	.79	.546	.474	.821	.843	.607

Several preliminary experiments were carried out to tune up the prediction system and choose the best architectural parameters. Splitting the proteins into training, test and validation sets we selected a Bi-Recursive NN architecture with state vectors of dimension five (both for forward and backward dynamics) and without hidden layers. By using hidden layers or a greater dimension for the state vectors we found the model particularly sensitive to the overfitting phenomena. We then performed a set of experiments trying to figure out the effect of the static training set generation as explained in Section 4. For each protein we generated a sample of graphs by means of a search procedure towards the target graph guided by a perfect evaluation function (but collecting all the valid successors during the search) ending up in a total of 40,275 graphs. The above dataset was split in a training set of 300 proteins (19,574 graphs), a test set of 150 proteins (11,117 graphs) which was used to estimate the prediction accuracy, and a validation set of 137 (9,584) used for early stopping. The adapted version of BPTS for Bi-Recursive model was used as training procedure. The trained network replaced $s^\star(G)$ as a heuristic evaluation function in the subsequent topology search algorithm, which was based on a beam search procedure with beam size $B = 10$. The search algorithm scales as $O(BW|V|^2)$, being W the number of weights in the network. In the following experiments, we compare the graph search algorithms to the GIOHMM architecture of Section 3.1.

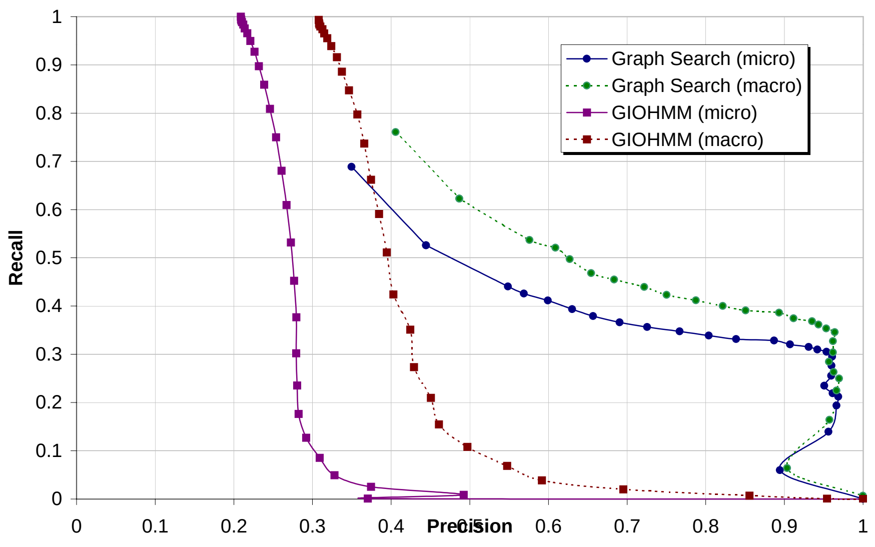

Figure 9: Precision-recall plots comparing the search algorithm and GIOHMMS.

Results are summarized in Figure 9, where we plot macro- and micro-averaged precision vs. recall. The graph is similar to a ROC curve (that plots true positive rate against false positive rate for different cutoffs of a diagnostic test). Macro-averages are computed by averaging precision and recall over the set of proteins. This measure tends to weight more the performance on short sequences. Micro-averages are obtained by computing precision and recall over the flattened set of segment pairs. The precision of the search algorithm is consistently better than that of GIOHMM, although recall never reaches 100%. This is obtained at significant expense of CPU time, not only in the training phase, but also during prediction.

In a second experiment we investigated the effects of dynamic sampling (i.e. exploration) during training. We applied the three exploration strategies described in Section 4: random exploration, pure exploitation and semi-uniform exploration. The last one was applied trying to find the optimal balance between exploration and exploitation. The probability of uniform random exploration was set to $\epsilon = 0.4$. In addition to these strategies, we also tried a rather different approach in which the network agent always proceed greedily (i.e. as in pure exploitation, at each step following always the best possible successor), but this time the network being updated on a representative sample of current state set of successors. The main purpose of this exploration scheme is to guide gradient descent towards a region where parameters are optimized using the wide possible spectrum of values for candidate alternatives. In these experiments, we used a 5-fold cross validation procedure splitting a representative set of 370 proteins into 5 subsets and routinely testing on one subset and training on the remaining four. After cross validation for each exploration scheme we obtained the results indicated in Table 9. The first column is labeled with the different updating schemes we applied. For each strategy we report performances measured with several indices: micro and macro-averaged precision (mP, MP), recall (mR, MR) and F_1 measure (mF_1, MF_1). Moreover, we report the percentage of correct prediction for non-contacts averaged over the set of proteins ($MP(nc)$) and over the whole segments pairs ($mP(nc)$). Last row (Hybrid) provides the indices obtained with the example selection strategy described above.

7 Conclusion

We have presented several new machine learning methods for predicting protein topologies in the form of contacts between amino acids or between secondary structure elements. The methods are based on a general class of Bayesian networks we call GIOHMMs that can be used to process data structures of variable size associated with particular graphical supports, whether sequences, lattices, trees, or more general graphs (as in the case of coarse contact maps). For efficiency, these architectures can be replaced by their recursive neural network versions we call GRNNs, which can be trained from examples by generalized gradient descent methods. In the case of coarse contact maps, we used GIOHMMs ideas to learn a scoring function that is used in turn to efficiently search the space of possible topological configurations.

In simulations, we have shown that the 2D lattice GIOHMMs perform significantly better than any other method on the prediction of fine-grained contact maps. For coarse contact maps, the combination of GIOHMM/GRNNs with graph-search methods so far has yielded the most promising results. Direction for future work include the integration of predictions at different level of granularity and the computationally efficient extension of graph search methods to fine-grained contact maps.

Acknowledgements

The work of PB and GP is in part supported by a Laurel Wilkening Faculty Innovation award, an NIH grant, and a Sun Microsystems award to PB at UCI. The work of PF and AV is partially supported by a MURST grant.

References

[1] R. A. Abagyan and S. Batalov. Do aligned sequences share the same fold? *J. Mol. Biol.*, 273:355–368, 1997.

[2] A. Aszodi, M. J. Gradwell, and W. R. Taylor. Global fold determination from a small number of distance restraints. *J. Mol. Biol.*, 251:308–326, 1995.

[3] D. Baker and A. Sali. Protein structure prediction and structural genomics. *Science*, 294:93–96, 2001.

[4] P. Baldi. Gradient descent learning algorithms overview: A general dynamical systems perspective. *IEEE Transactions on Neural Networks*, 6(1):182–195, 1995.

[5] P. Baldi and S. Brunak. *Bioinformatics: the machine learning approach*. MIT Press, Cambridge, MA, 2001. Second edition.

[6] P. Baldi and S. Brunak and P. Frasconi and G. Soda and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[7] P. Baldi and S. Brunak and P. Frasconi and G. Pollastri and G. Soda. Bidirectional Dynamics for Protein Secondary Structure Prediction In R. Sun and C.L. Giles (Eds.): *Sequence Learning*, LNAI 1828, pp. 80–104, 2000.

[8] P. Baldi and Y. Chauvin. Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation*, 8(7):1541–1565, 1996.

[9] P. Baldi and G. Pollastri. Generalized IOHMMs and recurrent neural network architectures. 2002. Submitted.

[10] P. Baldi and G. Pollastri. Machine learning structural and functional proteomics. *IEEE Intelligent Systems. Special Issue on Intelligent Systems in Biology*, 17(2), 2002.

[11] Y. Bengio, P. Simard and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166, 1994.

[12] Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Trans. on Neural Networks*, 7:1231–1249, 1996.

[13] M. Bianchini, M. Gori, and M. Maggini. On the problem of local minima in recurrent neural networks. *IEEE Transactions on Neural Networks* 5(2):167–177, 1994.

[14] M. Diligenti, P. Frasconi, and M. Gori Document Categorization using Hidden Tree-Markov Models and Structured Representations. In S. Singh, N. Murshed, and W. G. Kropatsch (Eds.) *Advances in Pattern Recognition, Proc. ICAPR'01*, pages 147–156. LNCS 2013, Springer, 2001.

[15] P. Fariselli and R. Casadio. Neural network based predictor of residue contacts in proteins. *Protein Engineering*, 12:15–21, 1999.

[16] P. Fariselli and R. Casadio. Prediction of the number of residue contacts in proteins. In *Proceedings of the 2000 Conference on Intelligent Systems for Molecular Biology (ISMB00), La Jolla, CA*, pages 146–151. AAAI Press, Menlo Park, CA, 2000.

[17] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio. Prediction of contact maps with neural networks and correlated mutations. *Protein Engineering*, 14:835–843, 2001.

[18] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, 9:768–786, 1998.

[19] P. Frasconi, M. Gori, A. Kuechler, and A. Sperduti From Sequences to Data Structures: Theory and Applications. In J. Kolen and S. Kremer (Eds.) *A Field Guide to Dynamic Recurrent Networks*, IEEE Press, 2001.

[20] P. Frasconi and A. Vullo. Prediction of Protein Coarse Contact Maps using Recursive Neural Networks. *Proc. IEEE-EMBS Conference on Molecular, Cellular, and Tissue Engineering*, (in press) 2002.

[21] U. Gobel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Genetics*, 18:309–317, 1994.

[22] J. Gorodkin, O. Lund, C. A. Andersen, and S. Brunak. Using sequence motifs for enhanced neural network prediction of protein distance constraints. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB99), La Jolla, CA*, pages 95–105. AAAI Press, Menlo Park, CA, 1999.

[23] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowl. Discov.*, 1:79–119, 1997.

[24] D. Heckerman. A tutorial on learning with Bayesian networks. In M.I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, 1998.

[25] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of representative data sets. *Prot. Sci.*, 1:409–417, 1992.

[26] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models *Machine Learning*, 29:245–273, 1997.

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

[28] S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, UK, 1996.

[29] A. M. Lesk, L. Lo Conte, and T. J. P. Hubbard. Assessment of novel fold targets in CASP4: predictions of three-dimensional structures, secondary structures, function and genetics. *Proteins*, 2001. Submitted.

[30] O. Lund, K. Frimand, J. Gorodkin, H. Bohr, J. Bohr, J. Hansen, and S. Brunak. Protein distance constraints predicted by neural networks and probability density functions. *Prot. Eng.*, 10:1241–1248, 1997.

[31] O. Olmea, B. Rost, and A. Valencia. Effective use of sequence correlation and conservation in fold recognition. *J. Mol. Biol.*, 295:1221–1239, 1999.

[32] O. Olmea and A. Valencia. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Fold. Des.*, 2:S25–32, 1997.

[33] F. Pazos, M. Helmer-Citterich, G. Ausiello, and A. Valencia. Correlated mutations contain information about protein-protein interactions. *J. Mol. Biol.*, 271:511–523, 1997.

[34] J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[35] G. Pollastri and P. Baldi. Predition of contact maps by recurrent neural network architectures. 2002. ISMB 2002 Conference. Submitted.

[36] G. Pollastri, P. Baldi, P. Fariselli, and R. Casadio. Prediction of coordination number and relative solvent accessibility in proteins. *Proteins*, 2001. In press.

[37] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary strucure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 2001. In press.

[38] I. N. Shindyalov, N. A. Kolchanov, and C. Sander. Can three-dimensional contacts of proteins be predicted by analysis of correlated mutations? *Protein Engineering*, 7:349–358, 1994.

[39] K. T. Simons, C. Strauss, and D. Baker. Prospects for ab initio protein structural genomics. *J. Mol. Biol.*, 306:1191–1199, 2001.

[40] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[41] S. Thrun. The role of exploration in learning control. In D. White and D. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, 1992.

[42] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Folding and Design*, 2:295–306, 1997.

[43] M. Vendruscolo. Protein folding using contact maps and contact vectors. This volume.