Bidirectional Dynamics for Protein Secondary Structure Prediction

Pierre Baldi
Net-ID, Inc.
Los Angeles, CA 90042
+1 323 222 1151
pfbaldi@netid.com

Søren Brunak
CBS, The Technical
University of Denmark
DK-2800 Lyngby, Denmark
+45 45252477
brunak@cbs.dtu.dk

Paolo Frasconi
DIEE, Università di Cagliari
09123 Cagliari Italy
+39 070 675 5849
paolo@diee.unica.it

Gianluca Pollastri
DSI, Università di Firenze
50139 Firenze Italy
+39 055 474 226
docbazz@tin.it

Giovanni Soda
DSI, Università di Firenze
50139 Firenze Italy
+39 055 4796 260
giovanni@dsi.unifi.it

Abstract

For certain categories of sequences, information from both the past and the future can be used for analysis and predictions at time t. This is the case for biological sequences where the nature and function of a region in a sequence may strongly depend on events located both upstream and downstream. We develop a new family of adaptive graphical model architectures for learning non-causal sequence translations. These architectures employ two chains of hidden variables that propagate information from the past and from the future, respectively. This general idea can be instantiated either as a stochastic model (generalizing input output hidden Markov models), or as a neural network (generalizing recurrent neural networks). We illustrate the methodology by applying bidirectional models to the problem of protein secondary structure prediction.

1 Introduction

Connectionist models for learning in sequential domains are typically dynamical systems that use hidden states to store contextual information. In principle, these models can adapt to variable time lags and perform complex sequential mappings. In spite of several successful applications (mostly based on hidden Markov models), the general class of sequence learning problems is still far from being satisfactorily solved. In particular, learning sequential translations is generally a hard task and current models seem to exhibit a number of limitations. One of these limitations, at least for some application domains, is the causality assumption. A dynamical system is said to be *causal* if the output at (discrete) time t does not depend on future inputs. Causality is easy to justify in dynamics that attempt to model the behavior of many physical systems. Clearly, in these cases the response at time t cannot depend on stimulae that the system has not yet received as input. As it turns out, non-causal dynamics over infinite time horizons cannot be realized by any physical or computational device. For certain categories of *finite* sequences, however, information from both the past and the future can be very useful for analysis and predictions at time t. This is the case, for example, of DNA and protein sequences where the structure and function of a region in the sequence may strongly depend on events located both upstream and downstream of the region, sometimes at considerable distances. Another good example is provided by the off-line translation of a language into another one. Even in the so-called "simultaneous" translation, it is well known that interpreters are constantly forced to introduce small delays in order to acquire "future" information within a sentence to resolve semantic ambiguities and preserve syntactic correctness.

Non-causal dynamics are sometimes used in other disciplines (for example, Kalman smoothing in optimal control or non-causal digital filters in signal processing). However, as far as connectionist models are concerned, the causality assumption is shared among all the types of models which are capable of mapping input sequences to output sequences, including recurrent neural networks and input-output HMMs (IOHMMs) (Bengio and Frasconi, 1996). In this paper we develop a new family of non-causal adaptive architectures where the underlying dynamics are factored using a pair of chained hidden state variables. The two chains store contextual information contained in the upstream and downstream portions of the sequence, respectively. The output at time t is then obtained by combining the two hidden representations. Interestingly, the same general methodology can be applied to many different classes of graphical models for time series, such as recurrent neural networks, IOHMMs, tree structured HMMs, and switching state space models (Ghahramani and Jordan, 1997). For concreteness, however, in the rest of this paper we focus exclusively on recurrent networks and IOHMMs.

The main motivation of this work is an application to the problem of protein secondary structure (SS) in molecular biology. The task can be formulated as the translation of amino acid input strings into corresponding output strings that describe an approximation of the proteins' 3D folding. The paper is organized as follows. In Section 2 we shortly review the literature on SS prediction. In Sections 3 and 4 we introduce the two novel non-causal architectures: bidirectional IOHMMs (BIOHMMs) and bidirectional RNNs (BRNNs). In Section 5 we report preliminary prediction results on the SS prediction task, where our system (based on an ensemble of MLPs and BRNNs) achieves circa the same performances of the best existing systems.

2 Prediction of protein secondary structure

Proteins are polypetides chains carrying out most of the basic functions of life at the molecular level. The chains can be viewed as linear sequences over the 20-letter amino acid alphabets that fold into complex 3D structures essential to their function. One step towards predicting how a protein folds is the prediction of its secondary structure. The secondary structure consists of local folding regularities often maintained by hydrogen bonds, and traditionally subdivided into three classes: alpha helices, beta sheets, and coils, representing all the rest. The sequence preferences and correlations involved in these structures have made secondary structure prediction one of the classical problems in computational molecular biology. Moreover, this is one application where machine learning methods, particularly neural networks, have had considerable impact yielding the best performing algorithms to date (Rost and Sander, 1994).

The basic architecture used in the early work of Qian and Sejnowski 1988 is a fully connected MLP with a single hidden layer that takes as input a *local* fixed-size window of amino acids (the typical width is 13), centered around the residue for which the secondary structure is being predicted. Although this approach has proven to be quite successful, using a local fixed width window has well known drawbacks. First, the size of the input window must be chosen a priori and a fair choice may be difficult. Second, the number of parameters grows with the window size. This means that permitting certain far away inputs to exert an effect on the current prediction is paid in terms of parametric complexity. Hence, one of the main dangers of the Qian–Sejnowski architectures is the overfitting problem.

Most of the subsequent work on predicting protein secondary structure using NNs has been based on architectures with a local window, although a lot of effort has been put on devising several improvements. Rost and Sander 1993b; 1993a started with Qian and Sejnowski's architecture, but used two methods to address the overfitting problem. First, they used early stopping. Second, they used ensemble averages (Hansen and Salamon, 1990; Krogh and Vedelsby, 1995) by training different networks independently, using different input information and learning procedures. But the most significant new aspect of their work is the use of multiple alignments, in the sense that profiles (i.e. position-dependent frequency vectors derived from multiple alignments), rather than raw amino acid sequences, are used in the network input. The reasoning behind this is that multiple alignments contain more information about secondary structure than do single sequences, the secondary structure being considerably more conserved than the primary sequence. Although tests made on different data sets can be hard to compare, the method of Rost and Sander (which resulted in the PHD prediction server (Rost and Sander, 1993b; 1993a; 1994)) still reaches the top levels of prediction accuracy (about 72%, measured using 7-fold cross validation).

Another interesting recent NN approach is the work of Riis and Krogh 1996, who address the overfitting problem by careful design of the NN architecture. Their approach has four main components. First, they reduce the number of free parameters by using an adaptive encoding of amino acids, that is, by letting the NN find an optimal and compressed representation of the input letters. Second, the authors design a different network for each of the three classes, using biological prior knowledge. For example, in the case of alpha-helices, they exploit the helix periodicity by building a three-residue periodicity between the first and second hidden layers. Third, Riis and Krogh use ensembles of networks and filtering to improve the prediction. The networks in each ensemble differ, for instance, in the number of hidden units used. Finally, the authors use multiple alignments together with a weighting scheme. Instead of profiles, for which the correlations between amino acids in the window are lost, predictions are made first from single sequences and then combined using multiple alignments. Most important, perhaps, the basic accuracy achieved is 66.3% when using seven-fold cross-validation on the same database of 126 non-homologous proteins used by Rost and Sander. In combination with multiple alignments, the method reaches an overall accuracy of 71.3%. Thus, in spite of a considerable amount of architectural design, the final performance is practically identical to (Rost and Sander, 1994).

A more detailed review of the secondary structure prediction problem and corresponding results can be found in (Baldi and Brunak, 1998). The important information however is that there is an emerging consensus of an accuracy upper bound, slightly above 70-75%, to any prediction method based on *local* information only. By leveraging evolutionary information in the form of multiple sequence alignments, performance seems to top at the 72-74% level, in spite of several attempts with sophisticated architectures. Thus it appears today that to further improve prediction results one must use distant information, in sequences and alignments, which is not contained in *local* input windows. This is particularly clear in the case of beta sheets where stabilizing bonds can be formed between amino acids far apart. Using long-ranged information, however, poses two formidable related challenges: (1) avoiding overfitting related to large-input-window MLPs (2) being able to detect the *sparse* and weak long-ranged signal and combine it with the significant local information, while ignoring the bulk of less relevant distant information.

The limitations associated with the fixed-size window approach can be mitigated using other connectionist models for learning sequential translators, such as recurrent neural networks (RNNs) or input-output hidden Markov models (IOHMMs). Unlike feedforward nets, these models employ state dynamics to store contextual information and they can adapt to variable width temporal dependencies. Unfortunately there are theoretical reasons suggesting that, despite an adequate representational power, RNNs cannot possibly learn to capture long-ranged information because of the vanishing gradient problem (Bengio *et al.*, 1994). However, it is reasonable to believe that RNNs fed by a *small* window of amino acids can capture some distant information using much less adjustable weights than MLPs. The usual definition of RNNs only allows "past" context to be used but, as it turns out, useful information for prediction is located both downstream and upstream of a given residue. The architectures described in the next sections remove these limitations.

3 Bidirectional IOHMMS

3.1 The architecture

A bidirectional IOHMM is a non-causal model of a stochastic translation defined on a space of finite sequences. Like IOHMMs, the model describes the conditional probability distribution $P(\boldsymbol{Y}|\boldsymbol{U})$, where $\boldsymbol{U} = U_1, U_2, \cdots, U_T$ is a generic input sequence and $\boldsymbol{Y} = Y_1, Y_2, \cdots, Y_T$ the corresponding output sequence. Although in the protein application described below both $\boldsymbol{U}$ and $\boldsymbol{Y}$ are symbolic sequences, the theory holds for sequences of continuous (possibly multivariate) sequences as well. The model is based on two Markov sequences of hidden state variables, denoted by $\boldsymbol{F}$ and $\boldsymbol{B}$, respectively. For each time step, F_t and B_t are discrete variables with realizations (states) in $\{f^1, \cdots, f^n\}$ and $\{b^1, \cdots, b^m\}$, respectively. As in HMMs, F_t is assumed to have a causal impact on the next state F_{t+1}. Hence, F_t stores contextual information contained on the left of t (propagated in the *forward* direction). Symmetrically, B_t is assumed to have a causal impact on the state B_{t-1}, thus summarizing information contained on the right of t (propagated in the *backward* direction). As in other Markov models for sequences, several conditional independence assumptions are made, and can be described by a Bayesian network as shown in Fig. 1. In particular,

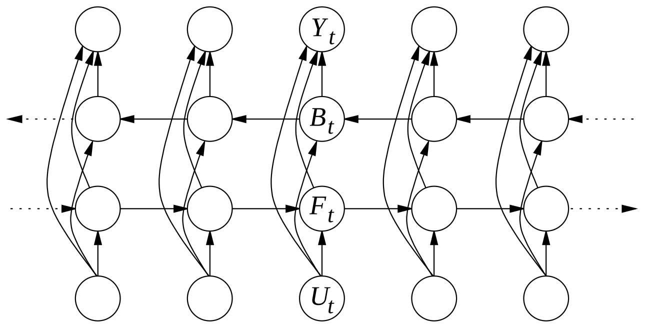

Figure 1: Bayesian networks for the bidirectional IOHMM.

the following factorization of the joint distribution holds:

$$P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{F}, \boldsymbol{B}) = \prod_{t=1}^{T} P(Y_t|F_t, B_t, U_t)P(F_t|F_{t-1}, U_t) \cdot P(B_t|B_{t+1}, U_t)P(U_t) \qquad (1)$$

Two boundary variables, B_{T+1} and F_0 are needed to complete the definition of the model. For simplicity we assume these variables are given, i.e. $P(B_{T+1} = b^1) = P(F_0 = f^1) = 1$, although generic (trainable) distributions could be specified. The suggested architectures can be viewed as a special form of factorial IOHMMs (with obvious relationships to factorial HMMs and hidden Markov decision trees (Ghahramani and Jordan, 1997)) where the state space is factorized into the state variables F_t and B_t.

3.2 Parameterization

The parameters of a Bayesian network specify the local conditional distribution of each variable given its parents. In the case of BIOHMMs, the local conditional distributions are $P(Y_t|F_t, B_t, U_t)$, $P(F_t|F_{t-1}, U_t)$, and $P(B_t|B_{t+1}, U_t)$. Unconditional distributions for root nodes (i,e, $P(U_t)$) do not need to be modeled if we assume that there are no missing data in the input sequences. A quite common simplification is to assume that the model is *stationary*, i.e. the above conditional distributions do not vary over time. Stationarity can be seen as a particular form of parameter sharing that significantly reduces the degrees of freedom of the model. In the discrete case parameters can be explicitly represented using conditional probability tables. Unfortunately the tables can become very large when nodes have many parents, or variables have large state spaces. Hence, a more constrained reparameterization is often desirable and can be achieved using the neural network techniques. In (Baldi and Chauvin, 1996), the general approach is demonstrated in the context of HMMs for protein families using, for the emission probabilities, a single hidden layer shared across all HMM states. In the case of BIOHMMs the approach can be extended by introducing three separate feedforward neural networks for modeling the local conditional probabilities $P(B_t|B_{t+1}, U_t)$,$P(F_t|F_{t-1}, U_t)$,$P(Y_t|F_t, B_t, U_t)$.

Alternatively, a modular approach using a different MLP for each state can be pursued (Baldi *et al.*, 1994; Bengio and Frasconi, 1996). The modular approach is also be possible with BIOHMMs, although in this case the number of subnetworks would become $n + m + nm$ (one subnetwork for each state b^i, f^j, and one for each pair (b^i, f^j)).

3.3 Inference and learning

The basic theory for inference and learning in graphical models is well established, and can readily be applied to the present architecture. For conciseness, we focus on the main aspects only. A major difference between BIOHMMs and IOHMMs or HMMs is that the Bayesian network for BIOHMMs is not singly connected. Hence direct propagation algorithms (e.g., Pearl's algorithm (Pearl, 1988)) cannot be used for solving the inference problem. Rather, we adopt the general junction tree algorithm (Jensen *et al.*, 1990). Given the regular structure of the network, the junction tree can be constructed by hand. Cliques are $\{U_t, F_t, B_t, Y_t\}, \{U_t, F_t, B_t, F_{t-1}\}\{U_t, F_t, B_t, B_{t+1}\}$. Assuming B_t and F_t have the same number of states (i.e., $n = m$) space and time complexities are $O(kTn^3)$, where k is the number of input symbols ($k = 20$ in the case of proteins). However, it should be noted that often in a sequence translation problem input variables are all observed (this is the case, at least, in the protein problem) and thus we know a priori that the nodes U_t always receive evidence, both in the learning and recall phases. Therefore, we can reduce the complexity by a factor k since only those entries which are known to be non-zero need to be stored and used in the absorption computations. In the case of proteins, this simple trick yields a speed up factor of about 20, the size of the input amino acid alphabet. The advantage is even more pronounced if, instead of a single amino acid, the input U_t is obtained by taking a window of amino acids, as explained later on.

Learning by maximum likelihood is implemented using a generalized EM algorithm. Basically, sufficient statistics in the E-step are computed by inference using the junction tree. The M-step deserves more attention because of the neural network reparameterization of the local conditional probabilities. In fact, maximizing the function $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, requires finding a perfect fit of the neural network parameters $\boldsymbol{\theta}$ to the expected values computed using the "old" parameters $\hat{\boldsymbol{\theta}}$. Even in the absence of local minima, a complete maximization would require an expensive inner gradient descent loop, inside the outer EM loop. Hence, we resorted to a generalized EM algorithm, where a single gradient descent step is performed inside the main loop. The expected sufficient statistics are used as "soft" targets for training the neural networks. In particular, for each output unit, the backpropagation delta-error term is obtained as the difference between the unit activations (before the softmax) and the corresponding expected sufficient statistics. For example, consider the network for estimating the conditional probability of the output Y_t, given the forward state F_t, the backward state B_t, and the input symbol U_t. For each sequence and for each time step t, let $a_{i,t}$ be the activation of the i-th output unit of this network when fed by $F_t = f^j$, $B_t = b^k$ and $U_t = u_t$ (u_t is obtained from the training sequence). Let $z_{i,j,k,t} = \exp(a_{i,t})/(\sum_\ell \exp(a_{\ell,t})$. We have $z_{i,j,k,t} = P(Y_t = y^i | F_t = f^j, B_t = b^k, U_t = u_t, \boldsymbol{\theta})$. Moreover, let $\hat{z}_{i,j,k,t} = P(Y_t = y^i, F_t = f^j, B_t = b^k, U_t = u_t, \text{training data}, \hat{\boldsymbol{\theta}})$ denote the expected sufficient statistics (obviously, $\hat{z}_{i,j,k,t} = 0$ if the current target $y_t \neq y^i$). Then, the error function for training this network is given by

$$C = \sum_{\text{training sequences}} \sum_t \sum_{i,j,k} \hat{z}_{i,j,k,t} \log z_{i,j,k,t}. \quad (2)$$

Similar equations hold for the other two networks modeling $P(F_t|F_{t-1}, U_t)$, and $P(B_t|B_{t+1}, U_t)$.

4 Bidirectional recurrent neural nets

4.1 The architecture

The basic idea underlying the architecture of BIOHMMs can be adapted to recurrent neural networks. Suppose in this case F_t and B_t are two vectors in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Then consider the following (deterministic) dynamics, in vector notation:

$$B_t = \beta(B_{t+1}, U_t) \quad (3)$$
$$F_t = \phi(F_{t-1}, U_t) \quad (4)$$

where $\beta()$ and $\phi()$ are nonlinear functions realized by two MLPs and $U_t \in \mathbb{R}^k$ encodes the input at time t (for example, using one-hot encoding in the case of amino acids). Equations 3 are completed by the two boundary conditions $F_0 = B_{T+1} = 0$. Also, consider the mapping

$$Y_t = \eta(F_t, B_t, U_t) \quad (5)$$

where $\eta()$ is also realized by an MLP (with a softmax output layer in the case of classification). The neural network architecture resulting from eqs. 3,5 is shown in Fig. 2, where for simplicity all the MLPs have a single hidden layer (several variants are conceivable by varying the number and the location of the hidden layers). Like in Elman's simple recurrent networks, the hidden state F_t is copied back to the input. This is graphically represented in Fig. 2 using the causal *shift operator* q^{-1} that operates on a generic temporal variable X_t and is symbolically defined as $X_{t-1} = q^{-1}X_t$. The shift operator with the composition operation forms a multiplicative group. In particular, q, the inverse (or non-causal) shift operator is defined $X_{t+1} = qX_t$ and $q^{-1}q = 1$. As shown in Fig. 2, a non-causal copy is performed on the hidden state B_t. Clearly, if we remove the backward chain $\{B_t\}$ we obtain a standard first-order RNN. A BRNN is stationary if the connection weights in the networks realizing $\beta()$, ϕ and η do not change over time. Stationarity will be assumed throughout the paper. It is worth

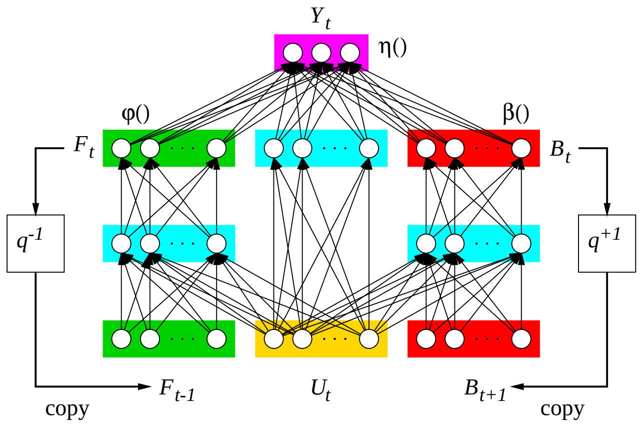

Figure 2: A bidirectional RNN.

noting that using MLPs for implementing $\beta()$ and $\phi()$ is just one of the available options as a result of their well known universal approximation properties. Similar generalizations of second-order RNN (Giles *et al.*, 1992) or recurrent radial basis functions (Frasconi *et al.*, 1996) are easily conceivable following the approach here described.

4.2 Inference and learning

As for standard RNNs, it is convenient to describe inference and learning in BRNNs by unrolling the network on the input sequence. The resulting graphical model has exactly the same form as the BIOHMM network shown in Fig. 1. Actually, the BRNN can be interpreted as a Bayesian network, except for some differences as explained below. First, causal relationships among nodes linked by a directed edge should be regarded as deterministic rather than probabilistic. In particular, $P(Y_t|F_t, B_t, U_t)$ should be regarded as a delta-Dirac distribution centered on a value corresponding to $Y_t = \eta(F_t, B_t, U_t)$. Similarly, $P(F_t|F_{t-1}, U_t)$ and $P(B_t|B_{t+1}, U_t)$ are replaced by Dirac distributions centered at $F_t = \phi(F_{t-1}, U_t)$ and $B_t = \beta(B_{t+1}, U_t)$, respectively. The second difference is that state variables in this case are vectors of real variables rather than symbols in a finite alphabet.

The inference algorithm in BRNNs is straightforward having in mind the network unrolled in time. Starting from $F_0 = 0$, all the states F_t are updated from left to right. Similarly, states B_t are updated from right to left. After forward and backward propagations have taken place, predictions Y_t can be computed. The main advantage with BRNNs (compared to BIOHMMs) is that inference is much more efficient. The intuitive reason is that in the case of BRNNs, the hidden states F_t and B_t evolve independently (without affecting each other). However, in the case of BIOHMMs, F_t and B_t, although conditionally independent given U_t, become dependent when Y_t is also given (as it happens during learning). This is reflected by the fact that cliques relative to B_t and F_t in the junction tree contain triplets of state variables, thus yielding a time complexity proportional to n^3 for each time step (if both variable have the same number of states n). In the case of BRNNs, assuming that the MLPs realizing $\beta()$ and $\phi()$ have $O(n)$ hidden units, time complexity is only proportional to n^2 for each time step and this can be further reduced by limiting the number of hidden units.

The learning algorithm is based on gradient descent so the only difference with respect to standard RNNs is to compute the gradient taking into account non-causal temporal dependencies. Because the unrolled network is acyclic, a generalized backpropagation algorithm can be derived as follows. The error signal is first injected into the leaf nodes (corresponding to the output variables Y_t). Then error is propagated over time (in both directions) by following any reverse topological sort of the unrolled net. Obviously, this step also involves backpropagation through the hidden layers of the MLPs. Since the model is stationary, weights are shared among the different replicas of the MLPs at different time steps. Hence, the total gradients are obtained by summing all the contributions associated to different time steps.

5 Experimental results

5.1 Data Set

We began this study using a high quality data set extracted from the Brookhaven Protein Data Bank (PDB) (Bernstein and et al., 1977) release 77 (July 1996). We excluded entries if:

- They were not determined by X-ray diffraction, since no commonly used measure of quality is available for NMR or theoretical model structures.
- The program DSSP could not produce an output, since we wanted to use the DSSP assignment of protein secondary structure (Kabsch and Sander, 1983).
- The protein that had physical chain breaks (defined as neighboring amino acids in the sequence having C^α-distances exceeding 4.0Å).
- They had a resolution worse than 1.9Å, since resolutions better than this enables the crystallographer to remove most errors from their models.
- Chains with a length of less than 30 amino acids were also discarded.

From the remaining set of chains a representative subset with low pairwise sequence similarities were selected by running the algorithm #1 of Hobohm *et al.* (1992), using the local alignment procedure search (rigorous Smith-Waterman algorithm) (Myers and Miller, 1988; Pearson, 1990) using the pam120 matrix, with gap penalties -12, -4. Since the beginning of this study we have progressively updated the data set and the reported experiments are based on a set consisting of 824 distinct protein chains, corresponding to 184973 amino acids, roughly 10 times more than what was available in (Qian and Sejnowski, 1988).

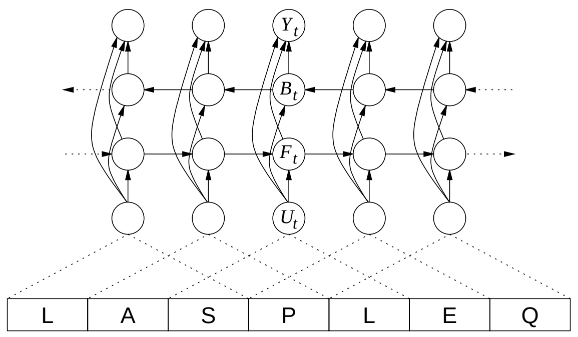

Figure 3: Unfolded bidirectional model for protein SS prediction. The model receives as input an explicit window of amino acids of size w ($w = 3$ in the figure).

5.2 Architecture details and results

Although in principle bidirectional models can memorize all the past and future information using the state variables F_t and B_t, we also tried to employ a window of amino acids as input at time t. In so doing, the input U_t for the model is a window of w amino acids centered around the t-th residue in the sequence (see Fig. 3). As explained in the previous sections, both with BIOHMMs and BRNNs the prediction Y_t is produced by an MLP fed by U_t (a window of amino acids) and the state variables F_t and B_t. Hence, compared to the basic architecture of Qian and Sejnowski, our architecture is enriched with more contextual information provided by the state variables. The main advantage of the present proposal is that w can be kept quite small (even reduced to a single amino acid), and yet relatively distant information propagated through the state variables. Because of stationarity (weight sharing) this approach allows a better control over the number of free parameters, thus reducing the risk of data overfitting.

In a set of preliminary experiments, we have tried different architectures and model sizes. In the case of BIOHMMs the best result was obtained using $w = 11$, $n = m = 10$, 20 hidden units for the output network and 6 hidden units for the forward and backward state transition networks. The correct residue prediction rate is 68%, measured by reserving 1/3 of the available sequences as a test set. This result was obtained without using output filtering or multiple alignments. Unfortunately, $n = 10$ seems too small a number for storing enough contextual information. On the other hand, higher values of n are currently prohibitive for today's computational resources since complexity scales up with n^3.

In the case of BRNNs we were able to obtain slightly better performances, with significant computational savings. A set of initial experiments indicated that redefining the output function as $Y_t = \eta(F_{t-k}, \ldots, F_{t+k}, B_{t-k}, B_{t+k}, U_t)$ and using $w = 1$ yields the best results. In subsequent experiments we have trained 4 different BRNNs with $n = m$ varying from 7 to 9, and k varying from 2 to 4. The number of free parameters varies from about 1400 to 2100. An RNN can develop quite complex nonlinear dynamics and, as a result, n BRNN state units are able to store more context than n BIOHMM discrete states. The performances of the 4 networks are basically identical, achieving about 68.8% accuracy measured on the test set. While these results do not lead to an immediate improvement, it is interesting to remark that using a static MLP we obtained roughly the same accuracy only after the insertion of additional architectural design as in (Riis and Krogh, 1996): adaptive input encoding and output filtering. More precisely, the MLP has $w = 13$, with 5 units for adaptive encoding (a total of about 1800 weights) and achieves 68.9%. Interestingly, although the 4 BRNNs and the static MLP achieve roughly the same overall accuracy, distributions of errors on the three classes are quite different. This suggests that combining predictions from filtered MLP and BRNNs could improve performance. Indeed, by constructing an ensemble with the five networks, accuracy increased to 69.5%. Finally we enriched the system using an output filtering network on the top of the ensemble and adding multiple alignment profiles as provided by the HSSP database (Sander and Schneider, 1991). In this preliminary version of the system we have not included commonly used features like entropy and number of insertions and deletions. The performance of the overall system is 73.3%.

In a second set of experiments we measured accuracy using 7-fold cross validation. The usage of more training data in each experiments seems to have a positive effect. The performance of the five networks ensemble is 69.6% without alignments and 73.7% using alignments. We must remark that these results are not directly comparable with those reported by Rost and Sander 1994 because our dataset contains more proteins and the assignment of residues to SS categories is slightly different (in our case the class *coil* includes everything except DSSP classes 'H' and 'E').

The last experiment is based on a set of 35 proteins from the 1998 edition of "Critical Assessment of Protein Structure Prediction" (Moult *et al.*, 1997; CASP3, 1998). This unique experiment attempts to gauge the current state of the art in protein structure prediction by means of blind prediction. Sequences of a number of target proteins, which are in the process of being solved, are made available to predictors before the experimental structures are available. Although we tried our system only after the competition was closed, we believe that result obtained on this dataset are still interesting. Our system achieved 71.78% correct residue prediction on the 35 sequences. A direct comparison with other systems is difficult. The best system (labeled JONES-2 in the CASP3 web site) achieves 75.5% correct residue prediction on a subset of 23 proteins (performance of JONES-2 on the remaining 12 proteins is not available). It should be also remarked that, in the CASP evaluation system, DSSP class 'G' (3-10 helix) is assigned to 'H' and DSSP class 'B' (beta bridge) is assigned to 'E'. Moreover, accuracy is measured by averaging the correct prediction fraction over single proteins, thus biasing sensitivity towards shorter sequences. Using this convention, our accuracy is 74.1% on 35 proteins while JONES-2 achieves 77.6% on 23 proteins. If we focus only on the 24 proteins for which our network has the highest prediction confidence (the criterion is based on the entropy at the softmax output layer of the network), then the performance of our system is 77.5%, although it is likely that in so doing we are including sequences which are easy to predict. More importantly, JONES-2 results have been obtained using profiles from TrEMBL database (Bairoch and Apweiler, 1999). These profiles contain many more sequences than our profiles which are based on the older HSSP database. We believe that this leaves room for further improvements.

6 Conclusion

In this paper we have proposed two novel architectures for dealing with sequence learning problems in which data is not obtained from physical measurements over time. The new architectures remove the causality assumption that characterize current connectionist approaches to learning sequential translations. Using BRNNs on the protein secondary structure prediction task appears to be very promising. Our performance is very close to the best existing systems although our usage of profiles is not as sophisticated. In the next future we plan to improve our prediction system by using profiles from the TrEMBL database.

Acknowledgements
The work of PB is in part supported by an NIH SBIR grant to Net-ID, Inc. The work of SB is supported by a grant from the Danish National Research Foundation. The work of PF and GS is partially supported by a "40%" grant from MURST, Italy.

References

A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Res*, (27):49–54, 1999.

P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* MIT Press, Cambridge, MA, 1998.

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

P. Baldi, Y. Chauvin, T. Hunkapillar, and M. McClure. Hidden Markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, 91:1059–1063, 1994.

Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Trans. on Neural Networks*, 7(5):1231–1249, 1996.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, 5(2):157–166, 1994.

F. C. Bernstein and et al. The protein data bank: A computer based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

CASP3. Third community wide experiment on the critical assessment of techniques for protein structure prediction. Unpublished results available in `http://predictioncenter.llnl.gov/casp3`, December 1998.

Paolo Frasconi, Marco Gori, Marco Maggini, and Giovanni Soda. Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23:5–32, 1996.

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–274, 1997.

C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, and Y. C. Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992.

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

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

F. V. Jensen, S. L. Lauritzen, and K. G. Olosen. Bayesian updating in recursive graphical models by local computations. *Comput. Stat. Quarterly*, 4:269–282, 1990.

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

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 231–238. The MIT Press, 1995.

John Moult *et al.* Critical assessment of methods of protein structure prediction (CASP): Round II. *Proteins*, 29(S1):2–6, 1997. Supplement 1.

E. W. Myers and W. Miller. Optimal alignments in linear space. *Comput. Appl. Biosci.*, 4:11–7, 1988.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann, 1988.

W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, (183):63–98, 1990.

N. Qian and T. J. Sejnowski. Predicting the secondary structure of glubular proteins using neural network models. *J. Mol. Biol.*, 202:865–884, 1988.

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

B. Rost and C. Sander. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proc. Natl. Acad. Sci. USA*, 90(16):7558–7562, 1993.

B. Rost and C. Sander. Prediction of protein secondary structure at better than 70 % accuracy. *J. Mol. Biol.*, 232(2):584–599, 1993.

B. Rost and C. Sander. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins*, (19):55–72, 1994.

C. Sander and R. Schneider. HSSP: Homology derived secondary structure of proteins. *Proteins*, 9:56–68, 1991.