Bidirectional Dynamics for Protein Secondary Structure Prediction

Pierre Baldi[1], Søren Brunak[2], Paolo Frasconi[3], Gianluca Pollastri[1], and Giovanni Soda[3]

[1] University of California at Irvine
[2] The Technical University of Denmark
[3] University of Florence

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 & 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

R. Sun and C.L. Giles (Eds.): Sequence Learning, LNAI 1828, pp. 80–104, 2000.

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 & Jordan, 1997). For concreteness, however, in the rest of this paper we focus exclusively on IOHMMs and recurrent neural networks.

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. This is a classic problem in bioinformatics which has been investigated for several years under the machine learning perspective, and for which significant performance improvements are still expected. Protein SS prediction can be formulated as the problem of learning a synchronous sequential translation, from strings in the amino acid alphabet to strings in the SS alphabet. The task is thus a special form of grammatical inference (Angluin & Smith, 1983). Nonetheless, to the best of our knowledge no successful applications of grammatical inference algorithms (neither symbolic, neither based on connectionist architecture) have been reported. Instead, the current best predictors are based on feedforward neural networks fed by a fixed-width window of amino acids, which by construction cannot capture relevant information contained in distant regions of the protein. Our proposal is motivated by the assumption that both adaptive dynamics and non-causal processing are needed to overcome the drawbacks of local fixed-window approaches. While our current system achieves an overall performance exceeding 75% correct prediction (at least comparable to the best existing systems) the main emphasis here is on the development of new algorithmic ideas.

The chapter is organized as follows. In Section 2, we shortly review the literature on protein 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 describe the protein datasets used in the experimental evaluation of the proposed system. Finally, in Section 6 we report preliminary prediction results on the SS prediction task using our best system which is based on ensembles of BRNNs.

2 Prediction of Protein Secondary Structure

Proteins are polypeptides 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 & 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. A significant improvement was obtained by cascading the previous architecture with a second network to clean up the output of the lower network. The cascaded architecture reached a performance of $Q_3 = 64.3\%$, with the correlations $C_\alpha = 0.41$, $C_\beta = 0.31$, and $C_\gamma = 0.41$ — see (Baldi et al., 1999) for a review of the standard performance measures used in this chapter. 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 and Sejnowski's 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 & Salamon, 1990; Krogh & 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 & Sander, 1993b, 1993a, 1994), still reaches the top levels of prediction accuracy ($Q_3 = 72\%$, measured using 7-fold cross validation). In the 1996 Asilomar competition CASP2, the PHD method reached $Q_3 = 74\%$ accuracy, thus performing much better than virtually all other methods used for making predictions of secondary structure.

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 $Q_3 = 66.3\%$ 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 $Q_3 = 71.3\%$, and correlation coefficients correlations $C_\alpha = 0.59$, $C_\beta = 0.50$, and $C_\gamma = 0.41$. Thus, in spite of a considerable amount of architectural design, the final performance is practically identical to (Rost & Sander, 1994). More recently, Cuff and Barton (1999) have compared and combined the main existing predictors. On the particular data sets used in their study, the best isolated predictor is still PHD with $Q_3 = 71.9\%$.

A more detailed review of the secondary structure prediction problem and corresponding results can be found in (Baldi & 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 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 IOHMMs and Bidirectional IOHMMS

3.1 Markovian Models for Sequence Processing

Hidden Markov models (HMMs) have been introduced several years ago as a tool for probabilistic sequence modeling. The interest in this area developed particularly in the Seventies, within the speech recognition research community, concerned at that time with the limitations of template-based approaches such as dynamic time-warping. The basic model was very simple, yet so flexible and effective that it rapidly became extremely popular. During the last years a large number of variants and improvements over the standard HMM have been proposed and applied. Undoubtedly, Markovian models are now regarded as one of the most significant state-of-the-art approaches for sequence learning. Besides speech recognition (see e.g. (Jelinek, 1997) for more recent advances), important application of HMMs include sequence analysis in molecular biology (Baldi et al., 1994; Krogh et al., 1994; Baldi & Chauvin, 1996; Baldi & Brunak, 1998), time series prediction (Andrew & Dimitriadis, 1994), numerous pattern recognition problems such as handwriting recognition (Bengio et al., 1995; Bunke et al., 1995), and, more recently, information extraction (Freitag & McCallum, 2000). HMMs are also very closely related to stochastic regular languages, making them interesting in statistical natural language processing (Charniak, 1993). The recent view of the HMM as a particular case of Bayesian networks (Bengio & Frasconi, 1995; Lucke, 1995; Smyth et al., 1997) has helped the theoretical understanding and the ability to conceive extensions to the standard model in a sound and formally elegant framework.

The basic data object being considered by the standard model is limited to a *single sequence* of observations (which may be discrete or numerical, and possibly multivariate). Internally, standard HMMs contain a *single* hidden state variable X_t which is repeated in time to form a Markov chain (see Figure 1). The Markov property states that X_{t+1} is conditionally independent of $X_1, \ldots, X_{t-1}$ given X_t. Similarly, each emission variable Y_t is independent of the rest given X_t. These conditional independence assumptions are graphically depicted using the Bayesian network of Figure 1. More details about the basic model can be found in (Rabiner, 1989).

Some variants of standard HMMs are conceived as extensions of the internal structure of the model. For example, factorial HMMs (Ghahramani & Jordan, 1997) contain more than just one hidden state variable (see for example the middle of Figure 1) and can also introduce complex probabilistic relationships amongst state variables. However, the data interface towards the external world essentially remain the same and the basic data objects being processed are still single sequences.

Another direction that can be explored to extend the data types that can be dealt with Markovian models is the direction of modeling. A standard HMM

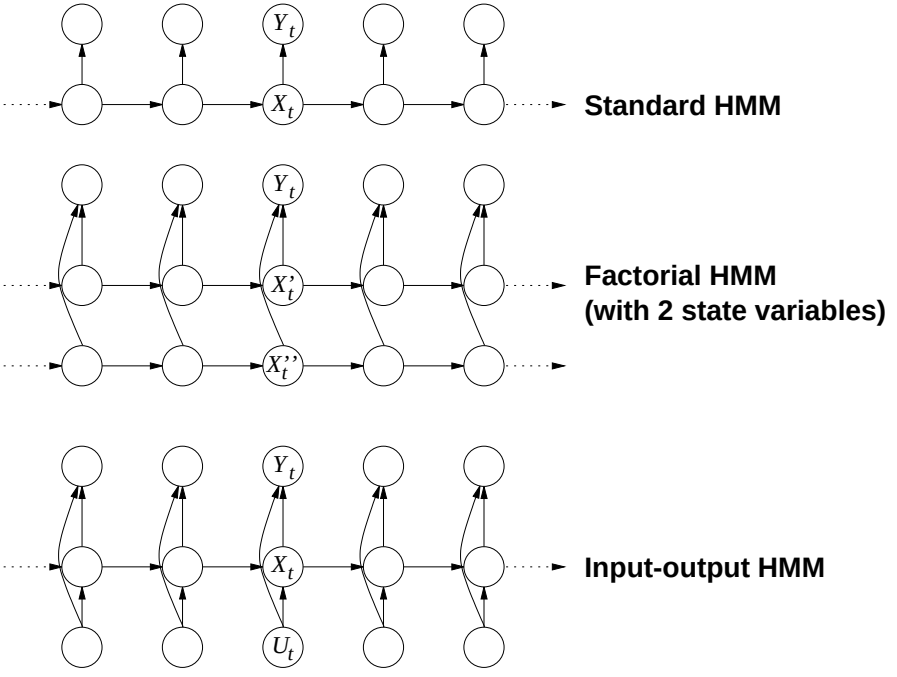

Fig. 1. Bayesian networks for standard, factorial, and input-output HMMs.

is a generative model. When used in classification tasks (as in isolated word recognition), one model per class is introduced and each model is typically trained on positive examples only. This means that no competition among classes is introduced when estimating the parameters and there is no associative probabilistic mapping from a sequence to a class variable. Learning is thus unsupervised because the model is trained to estimate an *unconditional* probability density (in a sense, we might say that a weak form of supervision occurs because class membership of training examples is only used to select the model to which the sequence is presented but each model is not aware of other classes). A fully supervised method for training HMMs has been early proposed by Brown (1987). The method modifies the function to be optimized and, instead of using maximum likelihood estimation, it relies on maximum mutual information (MMI) between the model and the class variable. In this way, the set of models employed in a classification task allows to estimate the *conditional* probability of the class *given* the input sequence, thus effectively introducing supervision in the learning process, as noted by Bridle (1989) and Bengio & Frasconi (1994). However, the MMI approach only allows to associate a single output variable to the input sequence. A more general supervised learning problem for sequences consists of associating a whole output sequence to the input sequence. This is the common sequence learning setting when using other machine learning approaches, such

as recurrent neural networks. This extension cannot be achieved by a modification of the optimization procedure but requires a different architecture, called input-output HMM (IOHMM).

As shown at the bottom of Figure 1, three classes of variables are considered. The emission Y_t is called in this case `output` variable. Hidden states in IOHMMs are conditionally dependent on the *input* variable U_t. In this way, state transition probabilities, although stationary, are non-homogeneous because the probability of transition is controlled by the current input. The resulting model is thus similar to a stochastic *translating* automaton (i.e. it can be used to estimate the conditional probability of the output sequence given the input sequence), but IOHMMs can also deal with *continuous* input and output variables. This can be easily achieved by employing feedforward neural networks as models of the conditional probabilities $P(Y_t|X_t, U_t)$ and $P(X_t|X_{t-1}, U_t)$ that define the parameterization of the model. This technique is described in detail in (Baldi & Chauvin, 1996) and in (Bengio & Frasconi, 1996), along with details about inference and learning algorithms. These topics are summarized later on in Sections 3.3 and 3.4 for the more general case of the bidirectional architecture presented in this paper.

3.2 The Bidirectional 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. 2. In particular, the following factorization of the joint distribution holds:

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

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

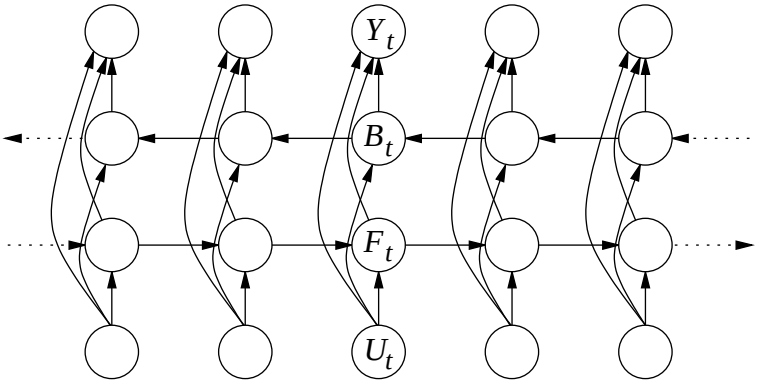

Fig. 2. Bayesian networks for the bidirectional IOHMM.

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 & Jordan, 1997)) where the state space is factorized into the state variables F_t and B_t.

3.3 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 & 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 & 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)).

3.4 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 such as Pearl's algorithm (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 $\{U_t, F_t, B_t, Y_t\}, \{U_t, F_t, B_t, F_{t-1}\}$, and $\{U_t, F_t, B_t, B_{t+1}\}$. 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 is formulated in the framework of maximum likelihood and is solved by the expectation maximization (EM) algorithm. EM is a family of algorithms for maximum likelihood estimation in the presence of missing (or hidden) variables (in our case, forward and backward states F_t and B_t are the hidden variables). Let $\mathcal{D}_c$ denote the complete data (input and output sequences $\boldsymbol{U}$ and $\boldsymbol{Y}$, plus the hidden state sequences $\boldsymbol{F}$ and $\boldsymbol{B}$) and let $\mathcal{D}$ denote the incomplete data (only the input and output sequences). Furthermore, let L_c denote the complete data likelihood, i.e.

$$L_c(\boldsymbol{\theta}; \mathcal{D}_c) = \prod_{\text{training sequences}} P(\boldsymbol{Y}, \boldsymbol{F}, \boldsymbol{B} | \boldsymbol{U}, \boldsymbol{\theta}).$$

Since forward and backward states are not observed, $\log L_c(\boldsymbol{\theta}; \mathcal{D}_c)$ is a random variable that cannot be optimized directly. However, given an initial hypothesis $\hat{\boldsymbol{\theta}}$ on the parameters and observed variables $\boldsymbol{U}$ and $\boldsymbol{Y}$, it is possible to compute the expected value of $\log L_c(\boldsymbol{\theta}; \mathcal{D}_c)$. Thus, an EM algorithms iteratively fills in missing variables according to the following procedure:

E-step Compute the auxiliary function

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) \doteq E[\log L_c(\hat{\boldsymbol{\theta}}; \mathcal{D}_c) | \boldsymbol{\theta}, \mathcal{D}]$$

M-step Update the parameters by maximizing $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ with respect to $\boldsymbol{\theta}$, i.e.

$$\hat{\boldsymbol{\theta}} \leftarrow \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$$

and repeat until convergence.

A well known result is that the above procedure will converge to a local maximum of the likelihood function (Dempster et al., 1977). A variant of the above procedure, called *generalized* EM (GEM) consists of computing a new parameter vector $\hat{\boldsymbol{\theta}}$ such that $Q(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) > Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ (instead of performing full optimization). GEM will also converge to a local maximum, although convergence rate may be slower.

In the case of belief networks, the E-step essentially consists of computing the expected sufficient statistics for the parameters. In our case these statistics are the following expected counts:

- $N^f_{j,l,u}$: expected number of (forward) transitions from f^l to f^j when the input is u ($j, l = 1, \ldots, n$; $u = 1, \ldots, K$);
- $N^b_{k,\ell,u}$: expected number of (backward) transitions from b^ℓ to b^k when the input is u ($k, \ell = 1, \ldots, m$; $u = 1, \ldots, K$);
- $N^y_{i,j,k,u}$: expected number of times output symbol i is emitted at a given position t when the forward state at t is f^j, backward state at t is b^k, and the input at t is u.

Basically, expected sufficient statistics are computed by inference using the junction tree algorithm as a subroutine. If local conditional probabilities were modeled by multinomial tables, then the M-step would be straightforward: each entry in the table would be replaced by the corresponding normalized expected count (Heckerman, 1997). However, in our case 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 the neural network weights $\boldsymbol{\theta}$ to be adapted to perfectly fit the normalized expected sufficient statistics. 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 activation (before the softmax) and the corresponding expected sufficient statistic. 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 $U_t = u_t$ (where u_t is fixed according to the input 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 contribution in this sequence at position t to the expected sufficient statistics (obviously, $\hat{z}_{i,j,k,t} = 0$ if the observed output at position t, $\overline{y}_t \neq y^i$). Then, the error function for training this network is given by

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

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:

$$F_t = \phi(F_{t-1}, U_t) \tag{3}$$

$$B_t = \beta(B_{t+1}, U_t) \tag{4}$$

where $\phi()$ and $\beta()$ are adaptive nonlinear transition functions. The vector $U_t \in \mathbb{R}^K$ encodes the input at time t (for example, using one-hot encoding in the case of amino acids). Equations 3 and 4 are completed by the two boundary conditions $F_0 = B_{T+1} = 0$. Transition functions $\phi()$ and $\beta()$ are realized by two MLPs $\mathcal{N}_\phi$ and $\mathcal{N}_\beta$, respectively. In particular, for MLP $\mathcal{N}_\beta$ we have:

$$b_{i,t} = \sigma\left(\sum_{j=1}^{N_\beta} \omega_{ij} h_{j,t}\right) \quad i = 1, \ldots n \tag{5}$$

where σ is the logistic function, N_β is the number of hidden units in MLP $\mathcal{N}_\beta$, $b_{i,t}$ is the i-th component of B_t and

$$h_{j,t} = \sigma\left(\sum_{\ell=1}^{N_\beta} w_{j,\ell} b_{\ell,t+1} + \sum_{\ell=1}^{k} v_{j,\ell} u_{\ell,t}\right) \quad j = 1, \ldots, N_\beta \tag{6}$$

is the output of j-th hidden unit. In the above equations, $\omega_{ij}, w_{j,\ell}$ and $v_{j,\ell}$ are adaptive weights. Network $\mathcal{N}_\phi$ is described by similar equations except that the forward state F_t is used instead of the backward state B_t.

Also, consider the mapping

$$Y_t = \eta(F_t, B_t, U_t) \tag{7}$$

where $Y_t \in \mathbb{R}^s$ is the output prediction and $\eta()$ is realized by a third MLP $\mathcal{N}_\eta$. In the case of classification, s is the number of classes and $\mathcal{N}_\eta$ has a softmax output layer so that outputs can be interpreted as conditional class probabilities. The neural network architecture resulting from eqs. 3,4, and 7 is shown in Fig. 3, 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

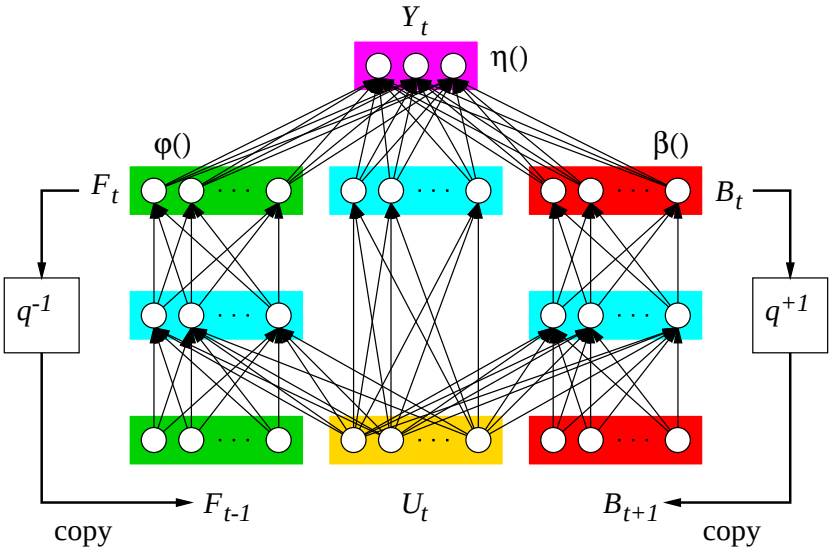

Fig. 3. A bidirectional RNN.

to the input. This is graphically represented in Fig. 3 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. 3, 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 networks $\mathcal{N}_\beta$, $\mathcal{N}_\phi$ and $\mathcal{N}_\eta$ do not change over time. Stationarity will be assumed throughout the paper. It is worth 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. 2. 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. Starting from $F_0 = 0$, all the states F_t are updated from left to right, following eq. 3. Similarly, states B_t are updated from right to left using the boundary condition $B_{T+1} = 0$ and following eq. 4. After forward and backward propagation have taken place, predictions Y_t can be computed using eq. 7. 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 MLPs $\mathcal{N}_\beta$ and $\mathcal{N}_\phi$ 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 maximum-likelihood. For simplicity we limit our discussion to classification. In this case, the cost function (or negative log likelihood) has the form of a cross-entropy:

$$C(\mathcal{D}; \boldsymbol{\theta}) = \sum_{\text{training sequences}} \sum_{i=1}^{s} \overline{y}_{i,t} \log y_{i,t}$$

where $\overline{y}_{i,t}$ is the target output (equal to 1 if the class at position t is i, and equal to 0 otherwise) and $y_{i,t}$ is the i-th output of $\mathcal{N}_\eta$ at position t. Optimization is based on gradient descent, where gradients are computed by a noncausal version of backpropagation through time. The intuitive idea behind backpropagation through time is to flatten-out cycles in the recurrent network by unrolling the recursive computation of state variables over time. This essentially consists of replicating the transition network for each time step, so that a feedforward network with shared weights is obtained. The same unrolling procedure can be applied in more general cases than just sequences. For example it can be applied to trees and graphs, provided that the resulting unrolled network is acyclic (Goller & Kuechler, 1996; Frasconi et al., 1998). In the case of BRNNs, it is immediate to recognize that the unrolling procedure yields an acyclic graph (if we only look at the main variables Y_y,B_t,F_t, and U_t, the unrolled network has the same topology as the Bayesian network shown in Figure 2). The unrolled network can be divided into slices associated with different position indices t, and for each slice there is exactly one replica of each network $\mathcal{N}_\phi$, $\mathcal{N}_\beta$, and $\mathcal{N}_\eta$.

The error signal is first computed for the leaf nodes (corresponding to the output variables Y_t):

$$\delta_{i,t} \doteq \frac{\partial C}{\partial a_{i,t}} = \overline{y}_{i,t} - y_{i,t} \quad i = 1, \ldots, s;\ t = 1, \ldots, T$$

where $a_{i,t}$ denotes the activation of the i-th output unit of $\mathcal{N}_\eta$ (before the softmax). Then error is then propagated over time (in both directions) by following any reverse topological sort of the unrolled net. For example, the delta-errors of $\mathcal{N}_\phi$ at position t are computed from the delta-errors at the input of $\mathcal{N}_\eta$ at t, and from the delta-errors at the first n inputs of $\mathcal{N}_\phi$ at position $t+1$. Similarly, the delta-errors of $\mathcal{N}_\beta$ at position t are computed from the delta-errors at the input of $\mathcal{N}_\eta$ at t, and from the delta-errors at the last m inputs of $\mathcal{N}_\beta$ at position $t-1$. Obviously, the computation of delta-errors 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. There are three possibilities: First, if i and j are any pair of connected neurons within the same position slice t, $\delta_{i,t}$ denotes the delta-error of unit i at t, and $x_{j,t}$ denotes the output of neuron j at t, then

$$\frac{\partial C}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_{i,t} x_{j,t}.$$

Second, if i is in the hidden layer of $\mathcal{N}_\beta$ at slice t and j is in the output layer of $\mathcal{N}_\beta$ at slice $t+1$, then

$$\frac{\partial C}{\partial \omega_{ij}} = \sum_{t=1}^{T} \delta_{i,t} b_{j,t+1}.$$

Finally, if i is in the hidden layer of $\mathcal{N}_\phi$ at slice t and j is in the output layer of $\mathcal{N}_\phi$ at slice $t-1$, then

$$\frac{\partial C}{\partial \omega_{ij}} = \sum_{t=1}^{T} \delta_{i,t} f_{j,t-1}.$$

4.3 Embedded Memories and Other Architectural Variants

One of the principal difficulties when training standard RNNs is the problem of vanishing gradients. In (Bengio et al., 1994), it is shown that one of the following two undesirable situations necessarily arise: either the system is unable to robustly store past information about its inputs, or gradients vanish exponentially. Intuitively, in order to contribute to the output at time t, the input signal at time $t-\tau$ must be propagated in the forward chain through τ replicas of the NN that implements the state transition function. However, during gradient computation, error signals must be propagated backward along the same path. Each propagation step involves a multiplication between the error vector and the Jacobian matrix associated with the transition function. Unfortunately, when the

dynamics develop attractors that allow the system to reliably store past information, the norm of the Jacobian is < 1. Hence, when τ is large, gradients of the error at time t with respect to inputs at time $t - \tau$ tend to vanish exponentially. Similarly, in the case of BRNNs, error propagation in both the forward and the backward chains is subject to exponential decay. Thus, although the model has in principle the capability of storing remote information, such information cannot be learnt effectively. Clearly, this is a theoretical argument and its practical impact needs to be evaluated on a per case basis.

In practice, in the case of proteins, the BRNN can reliably utilize input information located within about ± 15 amino acids (i.e., the total effective window size is about 31). This was empirically evaluated by feeding the model with increasingly long protein fragments. We observed that the average predictions at the central residues did not significantly change if fragments were extended beyond 41 amino acids. This is an improvement over standard NNs with input window sizes ranging from 11 to 17 amino acids (Rost & Sander, 1994; Riis & Krogh, 1996). Yet, there is presumably relevant information located at longer distances that our model have not been able to discover so far.

To limit this problem, we propose a remedy motivated by recent studies of NARX networks (Lin et al., 1996). In these networks, the vanishing gradients problem is mitigated by the use of an explicit delay line applied to the output, which provides shorter paths for the effective propagation of error signals. The very same idea cannot be applied directly to BRNNs since output feedback, combined with bidirectional propagation, would generate cycles in the unrolled network. However, as suggested in (Lin et al., 1998), a similar mechanism can be implemented by inserting multiple delays in the connections among hidden state units rather than output units. The modified dynamics in the case of BRNNs are defined as follows:

$$\begin{aligned} F_t &= \phi(F_{t-1}, F_{t-2}, \ldots, F_{t-s}, I_t) \\ B_t &= \beta(B_{t+1}, B_{t+2}, \ldots, B_{t+s}, I_t). \end{aligned} \tag{8}$$

The explicit dependence on forward or backward states introduces *shortcut* connections in the graphical model, forming shorter paths along which gradients can be propagated. This is akin to introducing higher order Markov chains in the probabilistic version. However, unlike Markov chains where the number of parameters would grow exponentially with s, in the present case the number of parameters grows only linearly with s. To reduce the number of parameters, a simplified version of Eq. 8 limits the dependencies to state vectors located s residues apart from t:

$$\begin{aligned} F_t &= \phi(F_{t-1}, F_{t-s}, I_t) \\ B_t &= \beta(B_{t+1}, t+s, I_t). \end{aligned} \tag{9}$$

Another variant of the basic architecture which also allows to increase the effective window size consists in feeding the output networks with a window in the forward and backward state chains. In this case, the prediction is computed as

$$O_t = \eta(F_{t-k}, \ldots, F_{t+k}, B_{t-k}, \ldots, B_{t+k}, I_t). \tag{10}$$

5 Datasets

The assignment of the SS categories to the experimentally determined 3D structure is nontrivial and is usually performed by the widely used DSSP program (Kabsch & Sander, 1983). DSSP works by assigning potential backbone hydrogen bonds (based on the 3D coordinates of the backbone atoms) and subsequently by identifying repetitive bonding patterns. Two alternatives to this assignment scheme are the programs STRIDE and DEFINE. In addition to hydrogen bonds, STRIDE uses also dihedral angles (Frishman & Argos, 1995). DEFINE uses difference distance matrices for evaluating the match of interatomic distances in the protein to those from idealized SS (Richards & Kundrot, 1988). While assignment methods impact prediction performance to some extent (Cuff & Barton, 1999), here we concentrate exclusively on the DSSP assignments. A number of data sets were used to develop and test our algorithms. We will refer to each set using the number of sequences contained in it. The first high quality data used in this study was extracted from the Brookhaven Protein Data Bank (PDB) (Bernstein & et al., 1977) release 77 and subsequently updated. 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 & Sander, 1983).
- The protein 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 chains, a representative subset with low pairwise sequence similarities was selected by running the algorithm #1 of Hobohm et al. (1992), using the local alignment procedure search (rigorous Smith-Waterman algorithm) (Myers & Miller, 1988; Pearson, 1990) using the pam120 matrix, with gap penalties -12, -4.

Thus we obtained a data set consisting of 464 distinct protein chains, corresponding to 123,752 amino acids, roughly 10 times more than what was available in (Qian & Sejnowski, 1988). Another set we used is the EMBL non-redundant PDB subsets that can be accessed by ftp at the site ftp.embl-heidelberg.de. Data details are in the file /pub/databases/pdb_select/README. The extraction is based on the file /pub/databases/pdb_select/1998_june.25.gz containing a set of non-redundant (25%) PDB chains. After removing 74 chains on which the DSSP program crashes, we obtained another set of 824 sequences, overlapping in part with the former ones. In addition, we also used the original set of 126 sequences of Rost and Sander (corresponding to 23,348 amino acid positions) as well as the complementary set of 396 non-homologue sequences (62,189 amino acids) prepared by Cuff and Barton (1999). Both sets can be downloaded at

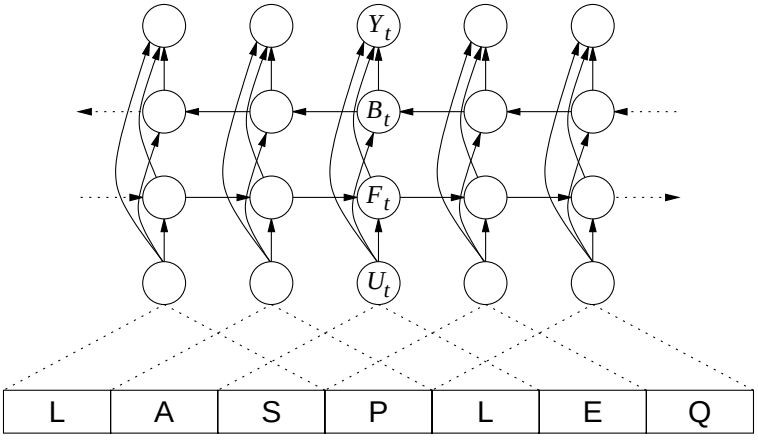

Fig. 4. Unfolded bidirectional model for protein SS prediction. The model receives as input an explicit window of amino acids of size w ($w = 3$ in the figure).

http://circinus.ebi.ac.uk:8081/pred_res/. Finally, we also constructed two more data sets, containing all proteins in PDB which are at least 30 amino acids long, produce DSSP output without chain breaks, and have a resolution of at least 2.5 Å. Furthermore the proteins in both sets have less than 25% identity to any of the 126 sequences of Rost and Sander. In both sets, internal homology is reduced again by Hobohm's #1 algorithm, keeping the PDB sequences with the best resolution. For one set, we use the standard 25% threshold curve for homology reduction. For the other set, however, we raise, the threshold curve by 25%. The set with 25% homology threshold contains 826 sequences, corresponding to a total of 193,249 amino acid positions, while the set with 50% homology threshold contains 1180 sequences (282,303 amino acids). Thus, to the best of our knowledge, our experiments are based on the currently largest available corpora of non-redundant data. In all but one experiment (see below), profiles were obtained from the HSSP database (Schneider et al., 1997) available at http://www.sander.embl-heidelberg.de/hssp/.

6 Architecture Details and Experimental Results

We carried out several preliminary experiments to tune up and evaluate the prediction system. DSSP classes were assigned to three secondary structure classes α, β, and γ as follows: α is formed by DSSP class H, β by E, and γ by everything else (including DSSP classes F, S, T, B, and I). This assignment is slightly different from other assignments reported in the literature. For example, in (Riis & Krogh, 1996), a contains DSSP classes H, G, and I. In the CASP competition (Moult & et al., 1997; CASP3, 1998), α contains H, and G, while β contains E, and B. In a first set of experiments, we used the 824 sequences dataset and

reserved 2/3 of the available data for training, and 1/3 for testing. We trained several BRNNs of different sizes and different architectural details. In all experiments, we set $n = m$ and we tried different values for n and k (see Eq. 10). The number of free parameters varied from about 1400 to 2600. Qualitatively we observed that using $k > 0$ can improve accuracy, but increasing n beyond 12 does not help because of overfitting. Results for this method, without using profiles, are summarized in the first rows of Table 1. By comparison, we also trained several feedforward NNs on the same data. The best feedforward NN achieved $Q_3 = 67.2\%$ accuracy using a window of 13 amino acids. By enriching the feedforward architecture with adaptive input encoding and output filtering, as in (Riis & Krogh, 1996), 68.5% accuracy was achieved (output filtering actually increases the length of the input window). Hence, the best BRNN outperforms our best feedforward network, even when additional architectural design is included. Subsequent experiments included the use of profiles. Table 1 reports the best results obtained by using multiple alignments, both at the input and output levels. Profiles at the input level consistently yielded better results. The best feedforward networks trained in the same conditions achieve $Q_3 = 73.0\%$ and 72.3%, respectively.

Table 1. Experimental results using a single BRNN and 1/3 of the data as test set. h_ϕ, h_β and h_η are the number of hidden units for the transition networks $\mathcal{N}_\phi$, $\mathcal{N}_\beta$ and the output network $\mathcal{N}_\eta$ respectively. We always set $h_\phi = h_\beta$.

Profiles	n	k	h_ϕ	h_η	W	Accuracy
No	7	2	8	11	1611	Q_3 =68.7%
No	9	2	8	11	1899	Q_3 =68.8%
No	7	3	8	11	1919	Q_3 =68.6%
No	8	3	9	11	2181	Q_3 =68.8%
No	20	0	17	11	2601	Q_3 =67.6%
Output	9	2	8	11	1899	Q_3 =72.6%
Output	8	3	9	11	2181	Q_3 =72.7%
Input	9	2	8	11	1899	Q_3 =73.3%
Input	8	3	9	11	2181	Q_3 =73.4%
Input	12	3	9	11	2565	Q_3 =73.6%

In a second set of experiments (also based on the 824 sequences), we combined several BRNNs to form an ensemble, as in (Krogh & Vedelsby, 1995), using a simple averaging scheme. Different networks were obtained by varying architectural details such as n, k, and the number of hidden units. Combining 6 networks using profiles at the input level we obtained the best accuracy $Q_3 = 75.1\%$, measured in this case using 7-fold cross validation. We also tried to include in the ensemble a set of 4 BRNNs using profiles at the output level but performance in this way slightly decreased to 75.0%. A study for assessing the capabilities of the model in capturing long ranged information was also performed. Results indicate

that the model is sensitive to information located within about ±15 amino acids. Although this value is not very high, it should be remarked that typical feedforward nets reported in the literature do not exploit information beyond $\tau = 8$. To further explore the long-range information problem we conducted another set of experiments using BRNNs with simplified embedded memories (see Eq. 9). In this case, as for the results reported in Table 1, we used a single model (rather than a mixture) and the test set method (1/3 of the available data) for measuring accuracy. We tried all values of s from 1 to 10, but in no case we could observe a significant performance improvement on the test set. Interestingly, our experiments showed that using shortcuts reduces the convergence difficulties associated with vanishing gradients: accuracy on the training set increased from 75.7% using no shortcuts to 76.9% with $s = 3$. On the other hand, the gap between training set and test set performance also increased. Thus overfitting offset the convergence improvement, probably because long-range information is too sparse and noisy.

Another experiment was conducted by training on all the 824 sequences and using the official test sequences used at the 1998 CASP3 competition. In this case, we adopted a slightly different class assignment for training (DSSP classes H, G, and I were merged together). The CASP3 competition was won by one of the two programs entered by D. Jones, which selected 23 out of 35 proteins obtaining a performance of $Q_3 = 77.6\%$ per protein, or $Q_3 = 75.5\%$ per residue (Jones, 1999). We evaluated that system on the whole set of 35 proteins by using Jones' prediction server at http://137.205.156.147/psiform.html. It achieved $Q_3 = 74.3\%$ per residue and 76.2% per protein. On the same 35 sequences our system achieved $Q_3 = 73.0\%$ per residue and $Q_3 = 74.6\%$ per protein. A test set of 35 proteins is relatively small for drawing general conclusions. Still, we believe that this result confirms the effectiveness of the proposed model, especially in consideration of the fact that Jones' system builds upon more recent profiles from TrEMBL database (Bairoch & Apweiler, 1999). These profiles contain many more sequences than our profiles, which are based on the older HSSP database, leaving room for further improvements of our system.

Table 2. First confusion matrix derived with an ensemble of 6 BRNNs with 2/3-1/3 data splitting. First row provides percentages of predicted helices, sheets, and coils within (DSSP-assigned) helices.

	pred α	pred β	pred γ
α	78.61%	3.13%	18.26%
β	5.00%	61.49%	33.51%
γ	10.64%	9.37%	79.99%

To further compare our system with other predictors, as in (Cuff & Barton, 1999), we also trained an ensemble of BRNNs using the 126 sequences in the Rost and Sander data set. The performance on the 396 test sequences prepared by Cuff

Table 3. Same as above. First row provides percentages of (DSSP-assigned) helices, sheets, and coils within the predicted helices.

	α	β	γ
pred α	80.77%	3.74%	15.49%
pred β	5.11%	73.17%	21.72%
pred γ	11.71%	15.63%	72.66%

and Barton is $Q_3 = 72.0\%$. This is slightly better than the 71.9% score for the single best predictor (PHD) amongst (DSC, PHD, NNSSP, and PREDATOR) reported in (Cuff & Barton, 1999). This result is also achieved with the CASP class assignment. Finally, we also trained an ensemble of 6 BRNNs using the set containing 826 sequences with less than 25% identity to the 126 sequences of Rost and Sander. When tested on the 126 sequences, the system achieves $Q_3 = 74.7\%$ per residue, with correlation coefficients $C_\alpha = 0.692$, $C_\beta = 0.571$, and $C_\gamma = 0.544$. This is again achieved with the harder CASP assignment. In contrast, the $Q_3 = 75.1\%$ described above was obtained by 7 fold cross-validation on 824 sequences and with the easier class assignment ($H \to \alpha$, $E \to \beta$, the rest $\to \gamma$). The same experiment was performed using the larger training set of 1,180 sequences having also less than 25% identity with the 126 sequences of Rost and Sander, but with a less stringent redundancy reduction requirement. In this case, and with the same hard assignment, the results are $Q_3 = 75.3\%$ with correlation coefficients $C_\alpha = 0.704$, $C_\beta = 0.583$, and $C_\gamma = 0.550$. The corresponding confusion matrices are given in Tables 2 and 3. Table 4 provides a summary of the main results with different datasets.

Table 4. Summary of main performance results.

Training sequences	Test sequences	Class assignment	Performance
824 (2/3)	824 (1/3)	Default	$Q_3 = 75.1\%$
824	35	CASP	$Q_3 = 73.0\%$
126	396	CASP	$Q_3 = 72.0\%$
826	126	CASP	$Q_3 = 74.7\%$
1180	126	CASP	$Q_3 = 75.3\%$

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. 4). 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 resulting model has about 10^5 parameters in total. 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 & 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 (Schneider et al., 1997). 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 & 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.

7 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. One improvement of our prediction system could be obtained 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

Andrew, F., & Dimitriadis, M. (1994). Forecasting probability densities by using hidden markov models with mixed states. In Weigend, A. S., & Gershenfeld, N. (Eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley.

Angluin, D., & Smith, C. H. (1983). A survey of inductive inference: Theory and methods. *ACM Comput. Surv.*, *15*(3), 237–269.

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

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

Baldi, P., Brunak, S., Chauvin, Y., & Nielsen, H. (1999). Assessing the accuracy of prediction algorithms for classification: an overview. Submitted for publication.

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

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

Bengio, Y., & Frasconi, P. (1995). An input output HMM architecture. In Tesauro, G., Touretzky, D., & Leen, T. (Eds.), *Advances in Neural Information Processing Systems 7*, pp. 427–434. The MIT Press.

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

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, *5*(2), 157–166.

Bengio, Y., & Frasconi, P. (1994). Credit assignment through time: Alternatives to backpropagation. In Cowan, J. D., Tesauro, G., & Alspector, J. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 6, pp. 75–82. Morgan Kaufmann Publishers, Inc.

Bengio, Y., LeCun, Y., Nohl, C., & Burges, C. (1995). LeRec: A NN/HMM hybrid for on-line handwriting recognition. *Neural Computation*, *7*(6), 1289–1303.

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

Bridle, J. S. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D.S.Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Vol. 2, pp. 211–217. Morgan Kaufmann.

Brown, P. (1987). *The Acoustic-Modeling problem in Automatic Speech Recognition*. Ph.D. thesis, Dept. of Computer Science, Carnegie-Mellon University.

Bunke, H., Roth, M., & Schukat-Talamazzini, E. (1995). Off-line Cursive Handwriting Recognition Using Hidden Markov Models. *Pattern Recognition*, *28*(9), 1399–1413.

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

Charniak, E. (1993). *Statistical Language Learning*. MIT Press.

Cuff, J. A., & Barton, G. J. (1999). Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, *34*, 508–519.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, *39*, 1–38.

Frasconi, P., Gori, M., Maggini, M., & Soda, G. (1996). Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, *23*, 5–32.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, *9*(5), 768–786.

Freitag, D., & McCallum, A. (2000). Information extraction with hmm structures learned by stochastic optimization. In *Proc. AAAI*.

Frishman, D., & Argos, P. (1995). Knowledge-based secondary structure assignment. *Proteins*, *23*, 566–579.

Ghahramani, Z., & Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning*, *29*, 245–274.

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

Goller, C., & Kuechler, A. (1996). Learning task-dependent distributed structure-representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, pp. 347–352.

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

Heckerman, D. (1997). Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, *1*(1), 79–119.

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

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press.

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

Jones, D. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, pp. 195–202.

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

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., & Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, pp. 1501–1531.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In Tesauro, G., Touretzky, D., & Leen, T. (Eds.), *Advances in Neural Information Processing Systems 7*, pp. 231–238. The MIT Press.

Lin, T., Horne, B. G., & Giles, C. L. (1998). How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. *Neural Networks*, *11*(5), 861–868.

Lin, T., Horne, B. G., Tino, P., & Giles, C. L. (1996). Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, *7*(6), 1329–1338.

Lucke, H. (1995). Bayesian belief networks as a tool for stochastic parsing. *Speech Communication*, *16*, 89–118.

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

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

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

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

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

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Richards, F. M., & Kundrot, C. E. (1988). Identification of structural motifs from protein coordinate data: secondary structure and first-level supersecondary structure. *Proteins*, *3*, 71–84.

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

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

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

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

Schneider, R., de Daruvar, A., & Sander, C. (1997). The hssp database of protein structure-sequence alignments. *Nucleic Acids Research*, *25*, 226–230.

Smyth, P., Heckerman, D., & Jordan, M. I. (1997). Probabilistic independence networks for hidden markov probability models. *Neural Computation*, *9*(2), 227–269.