Training Bidirectional Recurrent Neural Network Architectures with the Scaled Conjugate Gradient Algorithm

Michalis Agathocleous[1], Chris Christodoulou[1(✉)], Vasilis Promponas[2], Petros Kountouris[3], and Vassilis Vassiliades[1,4]

[1] Department of Computer Science, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
{magath06,cchrist}@cs.ucy.ac.cy
[2] Dept. of Biological Sciences, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
vprobon@ucy.ac.cy
[3] The Cyprus Institute of Neurology and Genetics, Nicosia, Cyprus
petrosk@cing.ac.cy
[4] Inria, Nancy - Grand Est, France
vassilis.vassiliades@inria.fr

Abstract. Predictions on sequential data, when both the upstream and downstream information is important, is a difficult and challenging task. The Bidirectional Recurrent Neural Network (BRNN) architecture has been designed to deal with this class of problems. In this paper, we present the development and implementation of the Scaled Conjugate Gradient (SCG) learning algorithm for BRNN architectures. The model has been tested on the Protein Secondary Structure Prediction (PSSP) and Transmembrane Protein Topology Prediction problems (TMPTP). Our method currently achieves preliminary results close to 73 % correct predictions for the PSSP problem and close to 79 % for the TMPTP problem, which are expected to increase with larger datasets, external rules, ensemble methods and filtering techniques. Importantly, the SCG algorithm is training the BRNN architecture approximately 3 times faster than the Backpropagation Through Time (BPTT) algorithm.

Keywords: Scaled Conjugate Gradient · Bidirectional Recurrent Neural Networks · Protein Secondary Structure Prediction · Transmembrane Protein Topology Prediction · Computational intelligence · Bioinformatics

1 Introduction

Even though a number of Machine Learning (ML) algorithms have been designed to process and make predictions on sequential data, the mining of such data types is still an open field of research due to its complexity and divergence [1]. Analysis and development of optimisation algorithms for specific ML techniques must

A.E.P. Villa et al. (Eds.): ICANN 2016, Part I, LNCS 9886, pp. 123–131, 2016.
DOI: 10.1007/978-3-319-44778-0_15

take into account (a) how to capture and exploit sequential correlations, (b) how to design appropriate loss functions, (c) how to identify long-distance interactions, and (d) how to make the optimisation algorithm fast [2]. One of the most successful classes of models which has been designed to deal with these questions is Recurrent Neural Networks (RNNs) [3]. The most common learning algorithm for such models is the Backpropagation Through Time (BPTT) [4,5], which is based on the gradient descent algorithm. Unfortunately, this kind of algorithms have a poor convergence rate [6]. Moreover, they depend on parameters which have to be specified by the user and are usually crucial for the performance of the algorithm. In order to eliminate these drawbacks, more efficient algorithms must be developed. One such algorithm is the Scaled Conjugate Gradient (SCG) [6], a second-order learning algorithm, that has been found to be superior to the conventional BPTT algorithm in terms of accuracy, convergence rate and the vanishing-gradient problem [7]. In addition, the original form of the algorithm [6] does not depend on any parameters.

Predictions on sequential data are particularly challenging when both the upstream and downstream information of a sequence is important for a specific element in the sequence. Application examples include problems from Bioinformatics such as Protein Secondary Structure Prediction (PSSP) [8–10] and other related problems (e.g., Transmembrane Protein Topology Prediction (TMPTP) [11]). In such sequence-based problems the events are dynamic and located downstream and upstream, i.e., left and right in the sequence. A ML model designed for such data must learn to make predictions based on both directions of a sequence. To predict these events, researchers utilise Bidirectional Recurrent Neural Network (BRNN) architectures [8]. The BRNN has proved to be a very efficient architecture for the PSSP problem with accuracy of approximately 76 % [8], while for the TMPTP problem to the best of our knowledge the BRNN architecture has not been used so far. The BRNN architectures are currently trained with an extension of the BPTT algorithm [5] with the error propagated in both directions of the BRNN. However, the SCG algorithm has not been developed for this architecture.

This paper introduces the mathematical analysis and development of the SCG learning algorithm for the BRNN architecture. The implemented model and learning algorithm is then tested on the PSSP and TMPTP problems.

2 Methodology

2.1 The BRNN Architecture

The BRNN architecture of Baldi et al. [8] consists of two RNNs and a Feed Forward Neural Network (FFNN). The novelty of this architecture is the contextual information contained in the two RNNs, the Forward RNN (FRNN) and the Backward RNN (BwRNN). The prediction at step t, for a segment in a sequence, is processed based on the information contained in a sliding window W_a. The FRNN iteratively processes the $(W_a - 1)/2$ residues located on the left

side of the position t to compute the forward (upstream) context (F_t). Similarly, the BwRNN iteratively processes the $(W_a - 1)/2$ residues located on the right side of the position t to compute the backward (downstream) context (B_t). Hence, the two RNNs are used to implement F_t and B_t. These RNNs correlate each sequence separately and hold an internal temporary knowledge to form the network's internal memory [3].

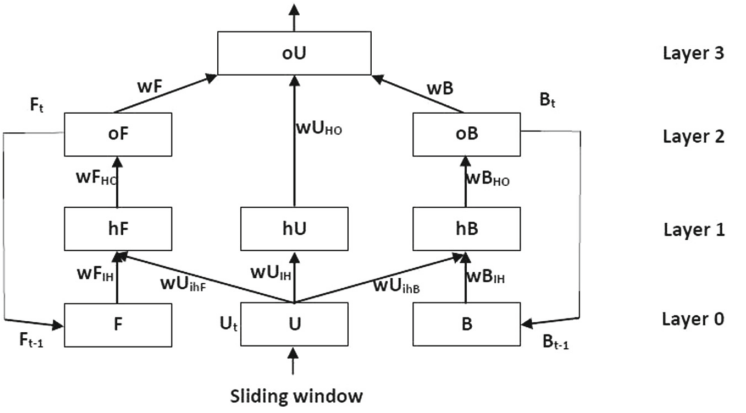

Fig. 1. The BRNN architecture

The BRNN architecture in Fig. 1 is inspired by the work of Baldi et al. [8]. Layer 0 in Fig. 1 is not an active layer, layers 1 and 2 have a hyperbolic tangent transfer function, while layer 3 is a softmax output layer which is calculated based on the result of Eq. 1. Box U stands for input nodes, F for the set of forward states and B for the set of backward states. The links between boxes oF and F and between boxes oB and B represent the recursive connections providing the information of the given number of states of current input U.

$$oU_i = softmax\left(\frac{1}{2\psi}\sum_{j=1}^{N_\phi} wF_{ij}\cdot f_{i,t} + \sum_{j=1}^{N_\beta} wB_{ij}\cdot b_{i,t} + \sum_{j=1}^{N_{hU}} wU_{ij}\cdot hU_{i,t}\right) \quad (1)$$

where ψ is the number of training patterns. N_ϕ, N_β and N_{hU} are the dimensions of oF, oB and hU layers, respectively. i stands for the position of a neuron in oU. wF_{ij}, wB_{ij} are the connection weights between layer 2 and 3, and wU_{ij} between layer 1 and 3. Finally, $f_{i,t}$, $b_{i,t}$ and $hU_{i,t}$ are the outputs of each neuron at time t of oF, oB and oU, respectively.

2.2 Development of the SCG Algorithm for BRNNs

As the Scaled Conjugate Gradient (SCG) learning algorithm [6] has not been previously developed for the BRNN architecture, we have mathematically analysed

and developed the corresponding learning formulas and optimisation procedure. These formulas were based on an unfolded BRNN. Since stationarity is assumed, the connection weights do not change over time and the unfolding architecture of the BRNN is as indicated by the work of Baldi et al. [8]. We have used the straightforward cost function given by 2:

$$E = \frac{1}{2\psi} \sum_{p=1}^{\psi} \sum_{i=1}^{s} (\overline{y}_{p,i,t} - y_{p,i,t})^2 \tag{2}$$

where s the number of neurons in the output layer, $\overline{y}_{p,i,t}$ the target output and $y_{p,i,t}$ the system output for an input pattern p.

The partial derivatives with respect to the weights are hidden in the system output as usual. Consequently, the general formula of the partial derivative of E with respect to any weight in Fig. 1 can be written as below:

$$\frac{\partial E}{\partial w} = \frac{1}{\psi} \left(\frac{\partial E}{\partial y_{p,i,t}} \cdot \frac{\partial y_{p,i,t}}{\partial oU_t} \cdot \frac{\partial oU_t}{\partial w} \right) = \frac{1}{\psi} \left((\overline{y}_{p,i,t} - y_{p,i,t}) \cdot y_{p,i,t} \cdot (1 - y_{p,i,t}) \cdot \frac{\partial oU_t}{\partial w} \right) \tag{3}$$

Finally, based on Eq. 3, we have calculated the derivatives which are used in the SCG algorithm for training the weights of the BRNN architecture in Fig. 1. After the partial derivatives of the cost function of Eq. 2 with respect to individual weights were calculated, they were directly applied to the SCG algorithm in the work of Møller [6] and represent the formulas of updating weights in the unfolded version of the BRNN.

One of the most undesirable difficulties during the training of a RNN is the vanishing gradient problem. A mechanism introducing shortcut connections between the forward and backward states of the sequence was used (as in [8]), forming shorter paths along the sequence where gradients can be propagated. Therefore, the gradient information at each step t includes a strong signal from the whole sequence to encounter for the vanishing gradient problem and consequently avoid long range dependencies elimination.

Furthermore, we have introduced a couple of minor modifications on the SCG algorithm to increase the convergence rate and the ability of the algorithm to search for the best solution in a complicated error surface of such a network:

1. An **Adaptive Step Size Scaling Parameter** S_{cSS} was introduced at step 7 of the SCG algorithm (see [6]). We have modified the algorithm's update weight vector rule to Eq. 4:

$$w_{k+1} = w_k + S_{cSS} a_k p_k \tag{4}$$

where a_k is the step size and p_k is the search direction. During the first iterations this scalar is high, assuming that the algorithm has identified a direction to a minimum. Hence, we force the algorithm to use bigger step size in a specific direction to approach a minimum faster. The adaptive scaling parameter is exponentially decreasing as the algorithm approaches a minimum to avoid losing the lowest point of the curve. Furthermore, this parameter is redefined each time

the SCG algorithm restarts. Empirically, the use of this parameter is mandatory for training a complicated BRNN architecture with the SCG algorithm.

2. Restart Algorithm Condition: The original SCG algorithm is restarted if the number of learning iterations surpasses the number of the network's parameters. However, this condition is successful only if the algorithm is used to optimize a quadratic function. Clearly, in the case of the BRNN architecture this condition fails because the error surface is more complicated than a quadratic function. Hence, we have chosen to restart the algorithm only if the training process develops slowly (improvement in training error $<10^{-7}$) and after the constant number of 20 iterations. Furthermore, our algorithm, before a restart, stores all the weight vectors and the respective training errors. Finally, after the algorithm reaches the final training iteration, it returns a trained model with the weight vector which was assigned to the lowest training error. Consequently, this version of the algorithm is widely exploring the respective error surface and is less likely to get stuck for a long time in a local minimum.

2.3 Application Domains and Data

High quality datasets for training and validation purposes are mandatory when constructing a prediction model. Therefore, we have chosen two well known bioinformatics problems which are suited to the BRNN architecture.

Protein Secondary Structure Prediction: The prediction of a protein's Secondary Structure (SS) from its Primary Structure (PS) is an important intermediate step to the identification of a protein's three-dimensional (3D) structure, which is crucial because it specifies the protein's functionality. Experimental methods for the determination of a protein's 3D structure are expensive, time consuming and frequently inefficient [8]. A protein is typically composed of 20 different amino acid types, which are chemically connected, folding into a 3D structure by forming short-, mid- and long-range interactions. When an experimentally determined 3D structure is available, each amino acid residue can be assigned to a SS class, usually under a commonly accepted scheme: helix (H), extended (E) and coil/loops (L). We use the CB513 [13], a non-redundant dataset, which has been heavily used as a PSSP benchmark dataset. Multiple sequence alignment (MSA) profiles have been shown to enhance machine learning-based PSSP, since they incorporate useful evolutionary information for the encoding of each position of a protein. More specifically, each protein sequence position is replaced by a 20-dimensional vector, which corresponds to the frequencies of 20 different amino acid types as calculated from a PSI-BLAST [12] search against the NCBI-NR (NCBI: http://www.ncbi.nlm.nih.gov/) database.

Transmembrane Protein Topology Prediction: Knowledge of the structure and topology of Transmembrane (TM) proteins is important since they are involved in a wide range of important biological processes and more than half of all drugs on the market target membrane proteins [11]. However, due to

experimental difficulties, this class of proteins is under-represented in structural databases. Similarly to the PSSP problem, a TMPTP dataset consists of the proteins' PS and each amino acid can be assigned to a topology class: inside a cell (I), outsite a cell (O) and inside a cell's membrane (T). Such a dataset has been introduced by Nugent and Jones [11] which contained 131 sequences (TM131) with all available crystal structures, verifiable topology and N-terminal locations. As in the PSSP problem, MSA profiles have been used to represent a sequence's PS.

3 Results and Discussion

The developed SCG learning algorithm for BRNNs has been implemented and tested on both PSSP and TMPTP problems. To train the BRNNs, we have used the already mentioned CB513 and TM131 datasets. More specifically, the model's input vector was a sliding window on a protein's PS. The target output class was the SS class for PSSP and topology class for TMPTP which was assigned to the segment at the middle point of a sliding window.

A single BRNN has been trained each time. At this stage, we carried out multiple experiments to tune up our model and extract preliminary results, which are shown in Tables 1 and 2. One of the most important parameters with a big impact on the results is the sliding window size. Particularly, we have used 3 window size parameters. Parameter W_a stands for the sliding window size on the PS sequence. The first $(W_a - 1)/2$ residues of the sliding window are used as input to F_t and similarly the last $(W_a - 1)/2$ residues are used an input to B_t. The W_c window parameter represents the number of W_a residues which are located at the center of the window and are used as input to hU. Finally, the W_{fb} window parameter represents the number of residues that are used as input to F_t and B_t at each step. Each one of the 3 window size parameters is multiplied by 20 which is the length of each amino acid MSA representation. Furthermore, we had to tune up the parameters that determine the network's architecture. The parameter n is the length of the context vectors oF and oB. In addition, the parameter h_n is the number of hidden units in hU layer and similarly h_{fb} is the number of hidden units in hF and hB layers. We have also used the already mentioned adaptive step size scaling parameter $ScSS$. Finally, we have used the S_{fb} and S_{out}, which are the numbers of additional consecutive context vectors in the future and the past of F_t/B_T and O_t, respectively. In all experiments, the 2/3 of the datasets were used to train the model and the 1/3 for validation purposes. The performance of our model has been evaluated by the Q_3 metric, which corresponds to the percentage of the correctly predicted residues [14].

Firstly, we have trained the BRNN architecture on the PSSP problem. For the purpose of tuning up the network's parameters we have used a subset of CB513 dataset, which contained 150 randomly selected protein sequences. The results can be seen in Table 1. After we have tuned up the network architecture, we have noticed that in order to maximize the algorithm's performance the three windows W_a, W_{fb} and W_c must have values of 25, 3 and 3, respectively. Thus,

shorter sliding window does not provide the network with enough information and longer sliding window cannot be captured by the network. As it can be seen from experiments *2* and *5* in Table 1, the correct tuning up of the sliding window parameter W_a can increase the algorithm's performance more than 3 %. Furthermore, we have noticed that the S_{cSS} parameter should be set to 100 to increase the convergence rate for this problem. As it can be seen from experiments *1*, *3* and *5* in Table 1, this parameter can increase the performance of the algorithm near to 4 %. The final Q_3 metric has also increased by 2 % after we dropped the S_{fb} and S_{out} parameters from 3 to 2, as it can be seen from experiments *6* and *7* in Table 1. Finally, the best Q_3 result was **73.90 %** which has been achieved in 500 training iterations, the 1/3 of BPTT learning iterations.

Table 1. Experimental results using 1/3 of the CB513 subset as a test set (see text for description of the parameters)

A/A	W_a	W_{fb}	W_c	S_{cSS}	n	h_n	h_{fb}	S_{fb}	S_{out}	$Q_3(\%)$
1	25	3	3	10	11	11	11	2	2	69.64
2	31	3	3	100	11	11	11	2	2	70.26
3	25	3	3	1000	11	11	11	2	2	69.10
4	25	3	3	100	9	9	9	2	2	67.56
5	25	3	3	100	11	11	11	2	2	73.03
6	25	3	3	100	14	14	14	2	2	**73.90**
7	25	3	3	100	14	14	14	3	3	71.26

Similarly, we have used the TM131 dataset to train the model with the results shown in Table 2. The network needed for this problem was much bigger compared to the one used for the PSSP problem. Importantly, the W_{fb} window had to be always set to 1, as larger size windows reduced the algorithm's performance. Furthermore, the S_{cSS} parameter was set to 10 to increase more than 2 % the algorithm's Q_3 accuracy, as it can be seen from experiments *2* and *3* in Table 2. Surprisingly, the network cannot converge with any value more than 0 for the S_{out} parameter. The best Q_3 achieved was **78.85 %**. This Q_3 accuracy was achieved with no external rules, ensemble methods or filtering techniques, which will be used in our final methodology and we expect to increase the performance of our system. Consequently, our results are lower than the 89 % correct predictions of Nugent and Jones [11] on the same dataset. This observation shows that, at least with regards to the output layer, context networks seem to be less important compared to the PSSP problem. This fact was actually expected, since TM regions are on average much longer than SS elements in globular proteins.

The preliminary results on the PSSP and TMPTP problems have shown that a BRNN trained with our version of the SCG learning algorithm can capture patterns and make predictions on complicated sequences where the information in both upstream and downstream direction is important. Furthermore, the SCG

Table 2. Experimental results using 1/3 of the TM131 as a test set (see text for description of the parameters)

A/A	W_a	W_{fb}	W_c	S_{cSS}	n	h_n	h_{fb}	S_{fb}	S_{out}	$Q_3(\%)$
1	25	1	25	10	40	40	40	3	3	54.20
2	25	1	25	10	40	40	40	3	0	72.56
3	25	1	25	100	40	40	40	3	0	70.36
4	25	1	25	10	37	37	37	3	0	73.06
5	25	1	25	10	30	30	40	3	0	77.73
6	25	1	25	10	25	25	25	3	0	**78.85**
7	25	1	15	10	40	40	40	3	0	70.09

learning algorithm needs much less training iterations than the conventional BPTT learning algorithm. This is very important if we take into account the latest developments in the field which demand very big datasets and network architectures, which consequently increase exponentially the training time. In addition, many of these methods are used in ensemble methods (as in Baldi et al. [8]) where the training time is increased even further. Furthermore, our experiments have shown that in the absence of the S_{cSS} parameter, the SCG algorithm training the BRNN architecture could not converge (results not shown).

Importantly, our final methodology will be based on our previous work in [9,10]. Our current preliminary results, for the PSSP problem, are slightly lower than the 76 % Q3 accuracy of Baldi et al. [8], as no big datasets, external rules, ensemble methods or filtering techniques have yet been used, through which we expect (based on the results of our previous work [9,10]) to increase the final Q3 accuracy for both the PSSP and TMPTP problems. Moreover, we do not have in this paper a direct comparison with the results of Baldi et al. [8] because the dataset used is different. Consequently, the final results of training a BRNN with SCG on the PSSP and TMPTP problems and the direct comparison with similar methods will be presented at the conference.

References

1. Schuster, M., Paliwal, K.K.: IEEE Trans. Signal Proces. **45**, 2673–2681 (1997)
2. Dietterich, T.G.: Machine learning for sequential data: a review. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SSPR&SPR 2002. LNCS, vol. 2396, pp. 15–30. Springer, Heidelberg (2002)
3. Elman, J.L.: Cogn. Sci. **14**, 179–211 (1990)
4. Werbos, P.J.: Proc. IEEE **78**(10), 1550–1560 (1990)
5. Frasconi, P., Gori, M., Sperduti, A.: IEEE Trans. Neural Netw. **9**, 768–786 (1998)
6. Møller, M.F.: Neural Netw. **6**, 525–533 (1993)
7. Hochreiter, S., Schmidhuber, J.: Neural Comput. **9**, 1735–1780 (1997)

8. Baldi, P., Brunak, S., Frasconi, P., Soda, G., Pollastri, G.: Bioinformatics **15**, 937–946 (1999)
9. Kountouris, P., Agathocleous, M., Promponas, V., Christodoulou, G., Hadjicostas, S., Vassiliades, V., Christodoulou, C.: IEEE ACM Trans. Comput. Biol. Bioinform. **9**, 731–739 (2012)
10. Agathocleous, M., Christodoulou, G., Promponas, V., Christodoulou, C., Vassiliades, V., Antoniou, A.: Protein secondary structure prediction with Bidirectional recurrent neural nets: can weight updating for each residue enhance performance? In: Papadopoulos, H., Andreou, A.S., Bramer, M. (eds.) AIAI 2010. IFIP AICT, vol. 339, pp. 128–137. Springer, Heidelberg (2010)
11. Nugent, T., Jones, D.T.: BMC Bioinf. **10**, 159 (2009)
12. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, A., Zhang, Z., Miller, W., Lipman, D.J.: Nucleic Acids Res. **25**, 3389–3402 (1997)
13. Cuff, J.A., Barton, G.J.: Proteins **34**, 508–519 (1999)
14. Richards, F., Kundrot, C.: Proteins **3**, 71–84 (1988)