Brain Computer Interfaces: A Recurrent Neural Network Approach

Gareth Oliver and Tom Gedeon

Australian National University
{gareth.oliver,tom.gedeon}@anu.edu.au
http://www.cs.anu.edu.au

Abstract. This paper explores the use of recurrent neural networks in the field of Brain Computer Interfaces(BCI). In particular it looks at a recurrent neural network, an echostate network and a CasPer neural network and attempts to use them to classify data from BCI competition IIIs dataset IVa. In addition it proposes a new method, EchoCasPer, which uses the CasPer training scheme in a recurrent neural network. The results showed that temporal information existed within the BCI data to be made use of, but further pre-processing and parameter exploration was needed to reach competitive classification rates.

Keywords: BCI, RNN, CasPer, Echostate Network.

1 Introduction

Brain Computer Interfaces (BCI) is attempts to use the brain as a control device for a computer. Long term, BCIs have potential commercial application due to their nature as a new mode of control for computers. Given sufficient reliability they would allow the control of electronic devices with just a thought! More immediately, Brain Computer Interfaces can be used to improve the quality of life of patients suffering from advanced Amyotrophic Lateral Sclerosis (ALS). Patients with advanced ALS suffer from complete paralysis while still receiving information from the environment and having active brains. A BCI would allow such people to communicate, as well as potentially control entire devices.

This paper investigates various classifiers that take advantage of the time series nature of the EEG data, and the temporal information that should be contained within it. In particular it will investigate various recurrent neural networks, which have typically been successful in exploiting temporal information.

2 Classifiers

Four different classification methods were used. A Recurrent Neural Network, trained using the Backpropogation Through Time(BPTT) initially proposed by [1]. An Echostate Network, based on the work of [2], [3]. Thirdly a CasPer Neural

K.W. Wong et al. (Eds.): ICONIP 2010, Part II, LNCS 6444, pp. 66–73, 2010.

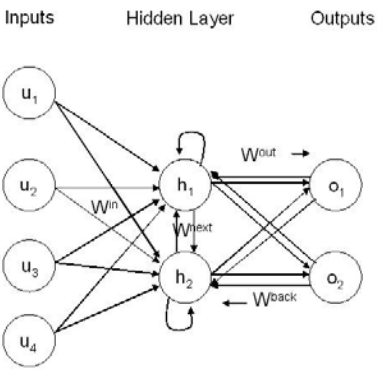

Fig. 1. Topography of a RNN with 4 Input neurons, 2 Hidden neurons and a single Output neuron

Network, following the design proposed in [4],[5]. Finally a new, hybrid extension of the CasPer neural network, EchoCasPer Neural Network, was designed to extend the principles behind the training of the CasPer neural network to a network whose structure could take advantage of the temporal information inherent in time series data.

2.1 Recurrent Neural Network

The RNN has three sets of neurons, Input (I), Hidden (H) and a single Output neuron (Fig. 1 shows the topography). The following weight matrices define the connections between these layers.

- Input to Hidden neurons: a $I \times H$ matrix $\mathbf{W}$in
- Hidden to Hidden neurons: a $H \times H$ matrix $\mathbf{W}$next
- Input and Hidden to Output neurons: a $(I + H) \times 1$ matrix $\mathbf{W}$out
- Output neurons to Hidden neurons: a $1 \times H$ matrix $\mathbf{W}$back

Using these weight matrices, the activation functions were defined as follows:

$$h(t+1) = tanh(u(t+1) \times \mathbf{W}in + h(t) \times \mathbf{W}next + o(t) \times \mathbf{W}back) \quad (1)$$

$$o(t+1) = [u(t+1), h(t+1)] \times \mathbf{W}out \quad (2)$$

To train the RNN the activation function is used to calculate the value of each hidden neuron and each output neuron at all times t. The error of the output neurons γ and hidden neurons δ are calculated for the last time step by:

$$\gamma(t) = class(t) - o(t) \quad (3)$$

$$\delta(t) = (1 - h(t)^2) \times \gamma(t) \times \mathbf{W}out \quad (4)$$

The errors of each previous time step are then calculated by:

$$\gamma(t) = class(t) - o(t) + \sum h(t+1) \times \mathbf{W}back \tag{5}$$

$$\delta(t) = (1 - h(t)^2) \times \gamma(t) \times \mathbf{W}out + \sum h(t+1) \times \mathbf{W}back \tag{6}$$

Finally the weights are updated by these error values using:

$$\mathbf{W}in_{i,j} += l \sum_{t=0}^{T} \delta_i(t) \times I_j(t) \tag{7}$$

$$\mathbf{W}out_i += l \sum_{t=0}^{T} \gamma(t) \times h_i(t) \tag{8}$$

$$\mathbf{W}next_i += l \sum_{t=0}^{T-1} o(t) \times \delta_i(t+1) \tag{9}$$

$$\mathbf{W}back_{i,j} += l \sum_{t=0}^{T-1} \delta_j(t+1) \times h_i(t) \tag{10}$$

Where l is the learning rate. To classify using a trained RNN, the error between each individual class is calculated by

$$error(class) = \sum_{t=0}^{T} (class - o(t))^2 \tag{11}$$

and the smallest is selected.

2.2 Echostate Network

The Echostate Network topography is the same as that of the RNN, see Fig. 1. As such the activation function and weights can be defined by equations 1 and 2. To ensure the echostate property[6], after initialising the weight matrices,

$$\mathbf{W}next = \frac{\mathbf{W}next}{\lambda_{max}} \tag{12}$$

$$\mathbf{W}next = \mathbf{W}next \times \alpha \tag{13}$$

where λ_{max} is the maximum absolute eigenvalue of Wnext and α is a value between 0 and 1 which governs the speed of attenuation within the hidden neurons.

To train the Echostate Network, BPTT is again used to calculate the error of each output neuron and each element of the hidden layer as per equations 3 to 6. The weights are then updated as follows:

$$\mathbf{W}in_{i,j} += l \sum_{t=0}^{T} \delta_i(t) \times I_j(t) \tag{14}$$

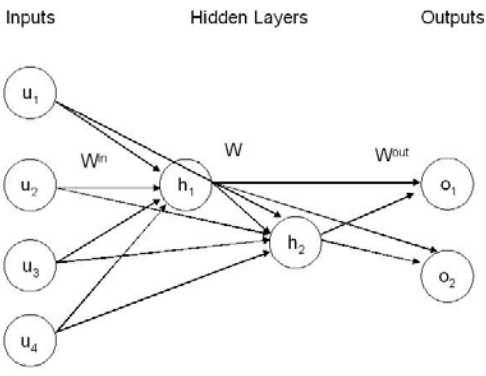

Fig. 2. Topography of a CasPer Neural Network with 4 Input neurons, 2 Hidden neurons and a single Output neuron

$$\mathbf{W}out_i += l \sum_{t=0}^{T} \gamma(t) \times h_i(t) \tag{15}$$

Notice that the weights connecting the hidden layer remain fixed. Once again classification is done by calculating the error for each individual class by equation 11.

2.3 CasPer Neural Network

A full description of the CasPer Neural Network training method can be seen in [4]. The topography of a CasPer Neural Network is shown in in Fig. 2. The hypobolic tangent function was used as the activation function for neurons within the CasPer Neural Network. The classification is done by calculating the error for each individual class and selecting the minimum by:

$$error(class) = (class - o)^2 \tag{16}$$

2.4 EchoCasPer Neural Network

The topography of the EchoCasPer Neural Network can be seen in Fig. 3. It consists of Input(I), Hidden(H), Casper(C) Neurons and an Output Neuron, (Fig. 3 shows an example topography). The connections are defined by the weight matrices:

- Input layer to the Reservoir: $I \times R$ matrix $\mathbf{W}$in
- Reservoir to itself: $R \times R$ matrix $\mathbf{W}$next
- To i^{th} neuron of the CasPer layer: $(I + R + i) \times 1$ matrix $\mathbf{W}$i
- To the Output neuron: $(I + R + C) \times O$ matrix $\mathbf{W}$out

The Wnext matrix has the Echostate property enforced on it as in the Echostate Network section. The activation functions are defined for each time t by:

$$r(t+1) = tanh((r)(t) \times \mathbf{W}next + (u)(t+1) \times \mathbf{W}in) \tag{17}$$

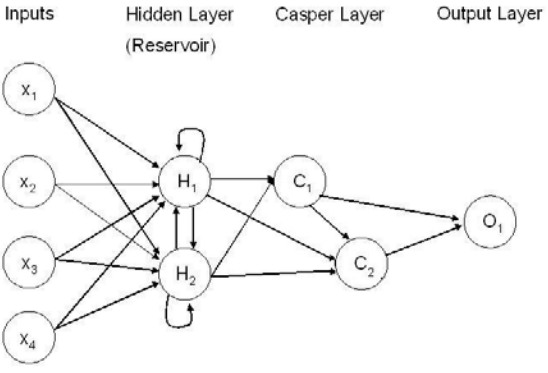

Fig. 3. Topography of a EchoCasPer Neural Network with 4 Input neurons, 2 Hidden neurons, 2 CasPer layer Neuron and a single Output neuron

$$c_i(t+1) = tanh([(u)(t+1), r(t+1), c_1(t+1)...c_{i-1}(t+1)] \times \mathbf{W}i) \tag{18}$$

$$o(t+1) = ([(u)(t+1), r(t+1), c(t+1)] \times \mathbf{W}out) \tag{19}$$

The training takes place in a series of iterations equal to C. At the beginning of each iteration the weight matrices need to be updated for the new neuron. **W**i needs to be initialised, and a new column needs to be added to **W**out. Additionally, matrices of learning rates, L1, L2 and L3 need to be assigned to each weight going to and from the neurons in the CasPer layer. The weights in the reservoir are not adaptable. The learning rates should be defined so that $L1 > L2 >> L3$.

All weights going to into the newly added neuron should be given a learning rate of L1 so that they adapt the fastest. Weights going from the newly added neuron to the output layer should be given a learning rate of L2. All other weights going to and from neurons in the CasPer or output layer should be given a learning rate of L3 so that the weights are mostly fixed.

Once the network is set up for the next set of training, it is trained to within a level of convergence using a backpropagation through time algorithm (BPTT). This is done by first running the EchoCasPer network for a training data for n time iterations, collecting the activations of the neurons at each time period. The errors for the CasPer layer(δ), Output Layer (γ) neurons are given by:

$$\gamma(t) = class - o(t) \tag{20}$$

$$\delta_i(t) = (1 - c_i(t)) \times \gamma(t) \times \mathbf{W}out_{I+R+i} + \sum_{m=i}^{C} \delta_m(t) \times \mathbf{W}m_i \tag{21}$$

Using the calculated errors the weights can be updated for each time step using the appropriate learning rates by:

$$\mathbf{W}out+ = \sum_{t=0}^{T} gradout[u(t), h(t), c(t)] \times \gamma(t) - sign(\mathbf{W}out)\mathbf{W}out^2 e^{-0.01it} \quad (22)$$

$$\mathbf{W}i+ = \sum_{t=0}^{T} grad_i[u(t), h(t), c_1(t)..c_i(t)] \times \delta_i(t) - sign(\mathbf{W})\mathbf{W}^2 e^{-0.01it} \quad (23)$$

Here the matrices grad and gradout contain the learning rates for the appropriate weight. The final term in each of the above expression is a regularisation term, used to prevent overfitting. Sign returns the sign of the passed number, so that the regularisation is performed in the correct direction. The decay term is a parameter that controls the amount of regularisation applied and it is the number of iterations of training have passed since the last neuron was added.

Once the weights have been updated, the error is calculated and convergence is checked. The convergence can be checked with:

$$0.01 < \frac{Elast - Ecurr}{Elast} \times (15 + pnodes) \quad (24)$$

Where p is an input parameter, increasing the steps trained before convergence and nodes is the number of neurons currently inserted in the hidden layer of the network. By having the convergence dependent on the number of neurons in the network the neural network will train longer. After convergence is reached the next neuron is added to the CasPer layer. Classification can be done once again by equation 11.

3 Pre-processing

The data for each classifier was pre-processed using two pre-processing techniques. Firstly the channels across the motor cortex were selected from the total number of channels, using expert knowledge to determine the correct ones to remove, reducing the number of channels to 43. Secondly a FIR filter was constructed to reduce frequencies outside of the beta (11 to 25Hz) rhythm as this is the primary band in which responses to motor tasks occur.

4 Experiment

The dataset used in these tests comes from the the Brain Computer Interface III, which is a competition that was held in association with Brain-Computer Interface Technology: Third International Meeting of june 2005 [7]. The specific dataset used was that of dataset IVa. This was provided by Fraunhofer FIRST, Intelligent Data Analysis Group. The dataset is a two class classification problem, with varying amount of training data and induced noise.

The data was recorded by a 118 electrode EEG. Only the right foot and right hand tasks were given as the training and test data for this classification

Table 1. BCI Compeon III Dataset IVa

Subject	Test	Training	Induced Noise
aa	168	112	yes
al	224	56	no
av	84	196	no
aw	56	224	yes
ay	28	252	no

problem. Each of the 5 subjects was broken into a separate test and training datasets, with characteristics described in the below table.

After initial testings, the following parameters for the topography of each network were used. The RNN had 7 Hidden Layer Neurons and a learning rate of 0.01. The Echostate Network also had 7 Hidden Layer Neurons and a learning rate of 0.01 with an alpha value of 0.5. The CasPer Neural Network had 5 Hidden Layer Neurons and values of 0.001, 0.0001 and 0.00001 for the learning rates L1, L2 and L3 respectively. The EchoCasPer Network had 7 Hidden Layer and 5 CasPer Layer Neurons, values of 0.1, 0.01 and 0.001 for L1, L2 and L3 respectively and finally an alpha value of Alpha.

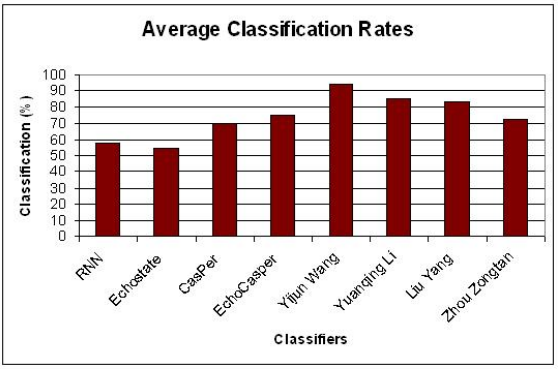

Fig. 4. Average classification rates over all subjects

Fig. 4 shows the average results of each classifier, in addition to four of the results of the best classifiers in the BCI competition. More details of their algorithms can be found in [7],[8],[9]. It can be clearly seen that, of the designed classifiers, the EchoCasPer Network performed the best, followed by the CasPer Network. All of these results fell short of the literature classifiers.

5 Conclusion

Both the RNN and the echostate network performed relatively poorly. The CasPer neural network was found to be significantly more successful than these

two but fell short of the best literature classifiers. In addition, the large number of inputs from the time series data resulted in extremely long training times, which also limited the amount of fine tuning of the multiple parameters could take place. The EchoCasPer network was designed as a way to allow the iterative method of dynamically creating the topography of a neural network used by the CasPer algorithm be used in a network that maintains a state. The resulting network made use of a reservoir from the echostate network to store the state while allowing a neural network to be built above it that preserved the CasPer properties. The results on the dataset were close to the literature results, and so shows promise if parameters and preprocessing methods are further fine-tuned. Additionally the classification rate was a 5% improvement over the CasPer algorithm. This implied that the EchoCasPer network was able to make use of some of the temporal information in the resevoir so further investigation is warrented.

References

1. Werbos, P.J.: Backpropagation through time: What it does and how to do it. Proceedings of IEEE 78, 1550–1560 (1990)
2. Ozturk, M.C., Xu, D., Principe, J.C.: Analysis and design of echostate networks. Neural Computation 19, 111–138 (2006)
3. Mezzano, T.: Echo State Networks application on maze problems. PhD thesis, Katholieke Universiteit Leuven (2007)
4. Treadgold, N.K., Gedeon, T.D.: A cascade network employing progressive rprop. In: International Work Conference on Artificial and Natural Neural Networks, pp. 733–742 (1997b)
5. Treadgold, N.K., Gedeon, T.D.: Extending and Benchmarking the CasPer Proceedings of the 10th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence (1997a)
6. Jaeger, H.: Short term memory in echostate networks. Technical report (2002)
7. Blankertz, B., Muller, K.R., Krusienski, D., Schalk, G., Wolpaw, J.R., Schlogl, A., Pfurtscheller, G., Millan, J.R., Schroder, M., Birbaumer, N.: The bci competition iii: Validating alternative approachs to actual bci problems. IEEE Transactions on Neural Systems and Rehabilitation Engineering 14, 153–159 (2006)
8. Wang, Y., Zhang, Z., Li, Y., Gao, X., Gao, S., Yang, F.: Bci competition 2003data set iv: An algorithm based on cssd and fda for classifying single-trial eeg. IEEE Transactions on Biomedical Engineering 51, 1081–1086 (2004)
9. Zhu, X., Guan, C., Wu, J., Cheng, Y., Wang, Y.: Expectation maximization method for eeg-based continuous cursor control. EURASIP Journal on Advances in Signal Processing (2007)