A Learning Algorithm for Continually Running Fully Recurrent Neural Networks

Authors: Ronald J. Williams and David Zipser.
Year: 1989
Journal: Neural Computation.

Report:

Abstract

"The exact form of a gradient-following learning algorithm for completely recurrent networks running in continuously sampled time is derived and used as the basis for practical algorithms for temporal supervised learning tasks. These algorithms have: (1) the advantage that they do not require a precisely defined training interval, operating while the network runs; and (2) the disadvantage that they require non local communication in the network being trained and are computationally expensive. These algorithms are shown to allow networks having recurrent connections to learn complex tasks requiring the retention of information over time periods having either fixed or indefinite length."

Introduction

- **The motivation is that to learn the complex tasks with recurrent neural networks in dynamic environment.**
- **The one of the problem with the connectionist theory, which they have been used until then is the computation expense of the neural nets.**
- **The new algorithm is introduced to overcome that computational expense through the dynamically running recurrent neural nets.**
- **The algorithms are also useful to acknowledge that in the complex tasks, the information is upgraded periodically, which may be fixed or variable length.**
- **The proposal approach follow the abstraction of "back propogation through time" but will not cause in high memory accumulation due to larger sequences.**
- **The previous works had not even been concentrated to make the general dynamic neural networks for the unrestricted architectures.**

Content

- **A major progress had been done in the field of neural networks for decreasing the computational expense and increasing the speed through forward feeding, back propagation through time etc.,**
- **In the basic algorithm, n-tuple of output is taken as the function of time as y(t) and m tuple of inputs as the x(t).**
- **Let them be the functions of z(t) varied with whether they are input or output networks.**
- **The weight matrix is taken as the matrix W with n x (m+n) and the bias weight is given for input whose value will always be 1.**
- **An other term $s_k(t)$ is introduced, which gives the total sum of the weight and the $z_t(t)$ and the next output $y_k(t+1)$ will be equal to the function of $s_k(t)$.**

- Then, an error is defined at time ‘t’ in the output set and zero otherwise.
- The total sum of the error is calculated particularly the square, to overcome the difficulty of becoming zero due to signed numbers.
- That total sum is equated to J(t) at time ‘t’ and the absolute error is defined upto the time ‘t’.
- Now, the total weights are updated based on the variation of the J(t) w.r.t to the weights.
- By equating them to zero to get the minimum error and computing we are resulted in a equation.
- Next, the assumption that to put the weights constant is withdrawn and they started to changing the weights.
- While the learning rate is dropped down, the algorithm which varies weights also gave much nearer results to the former as the algorithm had not been assured to adapt the gradient of whole error.
- And then the concept of teacher forcing is introduced, which replace the output with the desired output $d_k(t)$. Such that the networks are trained.
- It also almost gave the similar results as before with two alterations which are:
    - Where required values are used to compute future values of neural networks.
    - After p^k_{ij} values are used to calculate the corresponding weights, the p^k_{ij} values are initiated to zero.

Results

- The initial experiment is done with the random weights which gave very good results of the neural network.
- Pipelined XOR is experimented to calculate the time delays based on the current current random time steps. For this experiment, teacher forcing is not used.
- It had been pretty well for the smaller time delays and for extended delays, more network units are recommended, which increases the computational cost.
- Simple Sequence recognition is to find the sequence of the characters or words which results in “1” if the matched word is correct, else “0”. This cannot be done by feed forward mechanism, which we have used in the former experiment.
- To be a Turing Machine is another experiment, in which the neural network has to check for the balanced set of opening and closing parenthesis. For this the network units with 15 units always learned the task and it required the minimum of 12 units to learn the task.
- Learning to oscillate is an operation to create an oscillation series such as 010101….,00110011…., sine waves etc., This had worked only with teacher forcing. The reason is said to be that to create an oscillation series, the initial weights must be given correctly which may have very negligible in setting behavior.

Discussion and thoughts

- As it is an algorithm, there are many assumptions taken in the process such as the best results with randomized are taken, which is in general not a case to approach.
- Teacher forced algorithm in which the desired output is given to teach the neural network is also an arduous task.

- It is also said that the accessibility of both the weight matrix W and the error vector ‘e’ is also difficult at that time, which makes algorithm not worthy to use in actual neurophysiological networks.
- The solutions which are obscure were defended by comparing with the feedforward networks, which is to be noted.

Conclusion

- On the whole paper had been solved a good notion of dynamically changing neural networks. The algorithm looks so basic and easy to adopt.
- However, the obscurity of the solutions had been limited to evaluate the solutions in enough detail.
- As the algorithm is parallel, the computation speed is also improved from hardware parallely.