ASSIMILATING BOLTZMANN MACHINE AND REVERSE ANALYSIS METHOD

SARATHA SATHASIVAM, NG PEI FEN

Abstract : Logical knowledge resides in the synaptic connections of a network. By examining the connection strengths obtained during the learning process, logical rules that have been acquired may be deduced. Following this, we introduce a method to do this, which we call reverse analysis: given the values of the connections of a network, we hope to know what logical rules are entrenched in it. Boltzmann machine are examined for its ability to accelerate the performance of doing data mining by using technique named as Reverse Analysis method. In this paper, agent based modelling for this method will be developed by using NETLOGO as the platform. By using Boltzmann machine the capability of Reverse Analysis method in the data mining application can be enhanced.

Keywords: Reverse Analysis, Agent Based Modelling, Boltzmann machine, logical rules

Introduction : Neural networks are becoming very popular with data mining practitioners, particularly in medical research, finance and marketing fields. This is because they have proven through comparison, their predictive power with statistical techniques using real data sets such as clustering technique, K-means algorithm and others. There is a lot of research in data mining based on neuro-symbolic integration [1-4]. Most approaches have been based on the feed-forward network architecture in which the back-propagation algorithm is generally available for any functional induction, whereas Sathasivam [5] have present a method using the recurrent Little-Hopfield network known as Reverse Analysis Method. By examining the connection strengths obtained after learning process, logical rules that have been acquired may be deduced which we call Reverse Analysis: given the values of the connections of a network, we hope to know what logical rules are entrenched in it. In this paper, we develop agent based modelling (ABM) for doing Reverse Analysis method. We also will analyze the usage of Boltzmann machine in enhancing the performance of doing data mining in Hopfield network.

LOGIC PROGRAMMING ON A HOPFIELD NETWORK

In order to keep this paper self-contained we briefly review the optimization mechanism of the Hopfield model [6], and how logic programming can be carried out on such a network through this mechanism. The system consists of N formal neurons, each of which is described by an Ising variable $S_i(t), (i = 1, 2, ...N)$. Neurons then are bipolar, $S_i \in \{-1, 1\}$, obeying the dynamics $S_i \to \mathrm{sgn}(h_i)$, where the field, $h_i = \sum_j J_{ij}^{(2)} S_j + J_i^{(1)}$, i and j running over all neurons N, $J_{ij}^{(2)}$ is the synaptic strength from neuron j to neuron i, and $-J_i$ is a fixed bias (negative of the threshold) applied externally to neuron i . Hence, the neuron modifies its state S_i; according to Mc-Culloch Updating Rule.

Restricting the connections to be symmetric and zero-diagonal, $J_{ij}^{(2)} = J_{ji}^{(2)}$, $J_{ii}^{(2)} = 0$, allows one to write a Lyapunov or energy function,

$$E = -\frac{1}{2}\sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (1)$$

which decreases monotonically with the dynamics. The dynamics thus allows the handling of combinatorial optimization problems, where neurons are mapped onto the combinatorial variables and the energy function is equated to the cost function of the optimization problem.

The two-connection model can be generalized to include higher order connections. This modifies the "field" into

$$h_i = ... + \sum_j \sum_k J_{ijk}^{(3)} S_j S_k + \sum_j J_{ij}^{(2)} S_j + J_i^{(1)} \qquad (2)$$

where "....." denotes still higher orders, and an energy function can be written as follows:

$$E = ... - \frac{1}{3}\sum_i \sum_j \sum_k J_{ijk}^{(3)} S_i S_j S_k - \frac{1}{2}\sum_i \sum_j J_{ij}^{(2)} S_i S_j - \sum_i J_i^{(1)} S_i \qquad (3)$$

provided that $J_{ijk}^{(3)} = J_{[ijk]}^{(3)}$ for i, j, k distinct, with [...] denoting permutations in cyclic order, and $J_{ijk}^{(3)} = 0$ for any i, j, k equal, and that similar symmetry requirements are satisfied for higher order connections. The updating rule maintains

$$S_i(t+1) = \mathrm{sgn}[h_i(t)] \qquad (4)$$

In logic programming, a set of Horn clauses which are logic clauses of the form $A \leftarrow B_1, B_2, .., B_N$

where the arrow may be read "if" and the commas "and", is given and the aim is to find the set(s) of interpretation (i.e., truth values for the atoms in the clauses which satisfy the clauses (which yields all the clauses true). In other words, the task is to find 'models' corresponding to the given logic program [6, 7, 8, 9].

Boltzmann Machine

Boltzmann Machines are neural networks whose behaviour can be described statistically in terms of simple interactions between the units consist in that network. In order to implement the Boltzmann machine in a parallel networks, a simple modification of Hopfield's updating rule will be required. The Boltzmann machine operates by picking a hidden unit at random, let say unit i, and spinning the state of neuron i from S_i to $-S_i$ at temperature T (during the annealing cycle) with probability:

$$p_i = \frac{1}{1+e^{(-\frac{\Delta E_i}{T})}} \quad (5)$$

where ΔE_i, is given by:

$$E_i = \sum_{j=i}^{n} u_j w_{ij} - \theta_i \quad (6)$$

The temperature parameter T is important for the method of simulated annealing. If the spinning procedure is applied continually to the units, the units will change state and it will ensure the relative probability of two global states is determined exclusively by their energy difference when the system reached the thermal equilibrium and follows a Boltzmann distribution:

$$\frac{P_\alpha}{P_\beta} = e^{-\frac{(E_\alpha - E_\beta)}{T}} \quad (7)$$

where P_α is the probability of being in the α-th global state, and E_α is the energy of that state.

Reverse Analysis Method For 3-Cnf

This method consists of these following steps:

i) Enumerate number of neurons and patterns in the database.
ii) Initialize number of trials, energy relaxation loops, number of patterns.
iii) Extract the events from the database and represent in binary/bipolar pattern, where 0 indicates false state and 1 indicates true state (for bipolar -1 represent false state and 1 represent true state).
iv) Calculate the connection strengths for the events using Hebbian learning as described in section 3.
v) List out all the connection strengths obtained for third order connections($\Delta T_{[ABC]}, \Delta T_{[AB]}, \Delta T_{[AC]}, \Delta T_{[BC]}, \Delta T_A, \Delta T_B, \Delta T_C$), second order connections($\Delta T_{[AB]}, \Delta T_{[AC]}, \Delta T_{[BC]}, \Delta T_A, \Delta T_B, \Delta T_C$) and first order connections ($\Delta T_A, \Delta T_B, \Delta T_C$).
vi) Capture all the nonzero values (connection strengths) for third order connection.
vii) By using the method in section 3, list out all the corresponding clauses for (vi).
viii) Calculate the connection strengths for the extracted clauses in step (vii) and deduct the value of the corresponding clauses connection strengths from (v).
ix) Repeat the similar steps to extract the clauses corresponding to the first order and second order connections.

Model Of Neural Networks Simulator

Agent-based Modelling (ABM) [7] which also called individual-based modelling is a new computational modelling paradigm which is an analyzing systems that representing the 'agents' that involving and simulating of their interactions. Their attributes and behaviours will be group together through their interactions to become a scale. Programmer can design ABM in NETLOGO by using button, input, output, slides and other functions that make ABM easy to understand and use. In addition, ABM reveals the appearance of the systems from low to high level outcomes and it make improvement by surpassing the traditional modelling limitations such as allowing agent learning and adaption, limited knowledge and access to information. In general, when we build an ABM to simulate a certain phenomenon, we need to identify the actors first (the agents). We then need to consider the processes (rules) governing the interactions among the agents. Figure 1shows the layout of the ABM built.

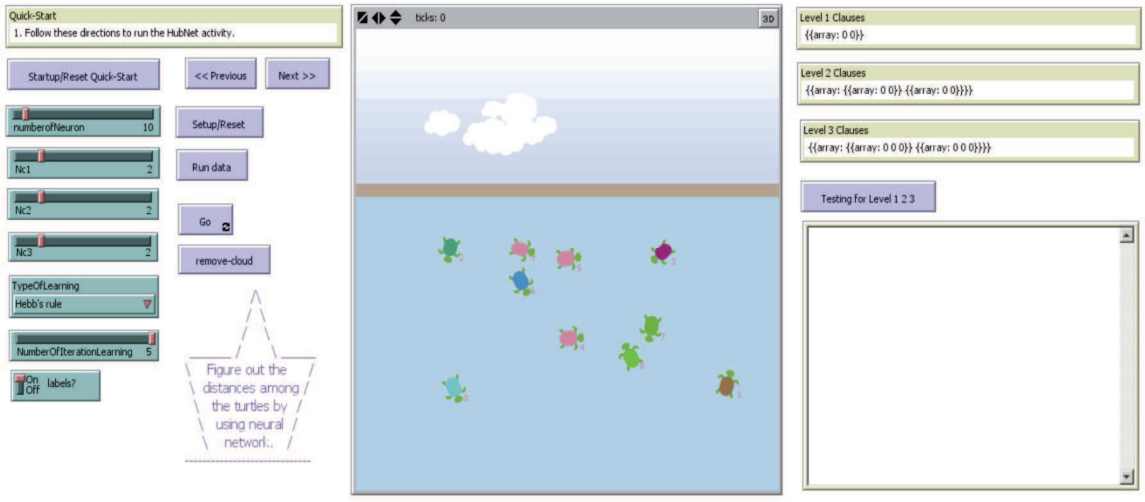

Figure 1: Layout of the ABM

Experimental Results And Discussion

We test the agent based modelling with computer simulation. We generate a set of 4 random clauses and subject 10 neurons network through Hebbian learning from events satisfying the generated clauses. The clauses are then extracted from the resulting network using reverse analysis algorithm and then compared to original set of generated clauses. We find that in 200 trials, we have 100% agreement. In the output we can see how the turtles are linked together. With this we can unearth the relationships between the data sets (Figure 2). By using ABM, user can analyze the graphical design of the links more efficiently and systematically.

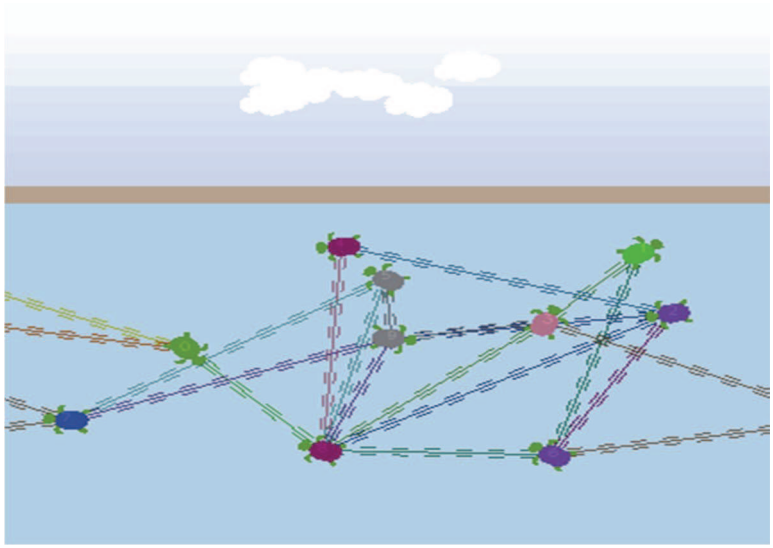

Figure 2: Testing Result

Conclusion : In this paper, we had developed agent based modelling to carry out reverse analysis method integrate with Boltzmann Machine by using NETLOGO as platform. So, ABM has enhanced the output efficiency of the Reverse Analysis method. The user can analyze the output and interpret the linked between the clauses extracted easily and robustly. This upgrades the capability of Reverse Analysis method in the data mining application.

Acknowledgement

This research is financed by Science Fund grant (305/PMATHS/613142) from the Ministry of Higher Education, Malaysia and USM grant (304/PMATHS/6312053).

References

1. Michalski, R. S., Bratko, I., & Kubat, M. 1998. Machine Learning and Data Mining: Methods and Applications. New York: John Wiley & Sons, Ltd.
2. Tsakonas, A. (2004). Towards Neural-Symbolic Integration: The Evolutionary Neural Logic Paradigms. 2nd IEEE International Conference on Intelligent Systems, pp 156-161.
3. Blair, H.A. & Dushin, F. & Jake, D.W. (1999). Continuous Models of Computation for Logic Programs. The Logic Programming Paradigm: A 25 Year Perspective, pp 231-255.
4. H"olldobler, S. & Kalinke, Y. (1994). Towards a Massively Parallel Computational Model for Logic Programming. Proceedings. ECAI94 Workshop on Combining Symbolic and Connectionist Processing, pp 68-77.
5. S. Sathasivam. 2007. Logic Mining in Neural Network. PhD Thesis, Malaysia.
6. Hopfield, J.J. 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings. Natl. Acad. Sci. USA. , 79(8), pp 2554-2558.
7. Saratha Sathasivam & W.A.T. Wan Abdullah, Logic Mining in Neural
8. Network, Computing, Volume 91, Issue, (2011), pp 119-133.

***

School of Mathematical Sciences, Universiti Sains Malaysia,
11800 USM Penang, Malaysia ,
saratha@usm.my, ngpeifen@yahoo.com