Research Journal of Applied Sciences, Engineering and Technology 6(22): 4281-4288, 2013
DOI:10.19026/rjaset.6.3545
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: March 11, 2013 Accepted: April 12, 2013 Published: December 05, 2013

Research Article

Developing Agent Based Modeling for Reverse Analysis Method

Saratha Sathasivam, Ng Pei Fen and Nawaf Hamadneh
School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM Penang, Malaysia

Abstract: In this study we want to develop Agent Based Modelling (ABM) for Reverse analysis method. 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. In this study, agent based modelling for this method will be developed by using NETLOGO as the platform. By using ABM the capability of Reverse Analysis method in the data mining application can be enhanced.

Keywords: Agent based modelling, logical rules, NETLOGO, reverse analysis

INTRODUCTION

Neural network which is inspired by biological neural networks, is assumed that intelligence emerges from interactions among a huge amount of neurons. It approach has giving an alternative knowledge discovery applications and computing to user (Michalski *et al*., 1998). It can solve sophisticated recognition and analysis problems. It obtains the ability to generalize the knowledge to similar pattern from the previous situation and results which often surmounts the human experts.

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 (Hamadneh *et al*., 2012). There is a lot of research in data mining based on neuro-symbolic integration (Blair *et al*., 1999; Hölldobler and Kalinke, 1994; Tsakonas, 2004). 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 have present a method using the recurrent Little-Hopfield network known as Reverse Analysis Method (Sathasivam, 2007). 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.

The aim of this study developed Agent Based Modelling (ABM) for doing Reverse Analysis method. Agent-based modelling is a powerful simulation modelling technique that has seen a number of applications in the last few years, including applications to real-world business problems. ABM is computer representation of systems that are comprised of multiple, interacting agents. We want to develop a user friendly approach to handle the task of Reverse Analysis. We limit our model to Horn clauses due to non-Horn clauses involve more computational complexity and triggered satisfiability problem.

LOGIC PROGRAMMING ON A HOPFIELD NETWORK

Many optimization problems, including those associated with intelligent behaviour, can be readily represented on Hopfield network by transforming the problem into variables such that the desired configuration corresponds to the minimization of the respective Lyapunov function. Indeed, the energy function can be thought of as a programming language for transforming optimization problems into a solution method applying network dynamics.

In order to keep this study self-contained we briefly review the optimization mechanism of the Hopfield model (Hopfield, 1982) 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 Using variable. $S_i(t),(i=1,2,....N)$ Neurons then are bipolar,

Corresponding Author: Saratha Sathasivam, School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM Penang, Malaysia

$S_i \in \{-1,1\}$, obeying the dynamics $S_i \to \text{sgn}(h_i)$, where, the field, $h_i = \sum_j J_{ij}^{(2)} S_j + J_i^{(1)}$, i and j running over all neurons N, $J^{(2)}_{\text{ij}}$ 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 and $-J_i$. Hence, the neuron modifies its state S_i; according to Mc-Culloch Updating Rule.

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

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

Which this 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:

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

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

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

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

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

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 (Sathasivam and Wan Abdullah, 2011).

Table 1: Synaptic strengths for $A \leftarrow B, C$ using Wan Abdullah's method

Synaptic strengths	Clause/$A \leftarrow B,C$
$J^{(3)}_{[ABC]}$	1/16
$J^{(2)}_{[AB]}$	1/8
$J^{(2)}_{[AC]}$	1/8
$J^{(2)}_{[BC]}$	-1/8
$J^{(1)}_{[A]}$	1/8
$J^{(1)}_{[B]}$	-1/8
$J^{(1)}_{[C]}$	-1/8

In principle logic programming can be seen as a problem in combinatorial optimization, which may therefore be carried out on a Hopfield neural network. This is done by using the neurons to store the truth values of the atoms and writing a cost function which is minimized when all the clauses are satisfied. For logic programming, we chose an energy function representing the number of unsatisfied clauses, given a set of logical interpretations (i.e., neural state assignments) (Hopfield, 1982), or the sum of "inconsistencies", where the inconsistency of a particular clause is given as the Boolean value of the negation of that clause.

As an example, consider the clause, $k_1 = A \leftarrow B, C$, which translates as $k_1 = A \vee \neg (B \wedge C) = A \vee \neg B \vee \neg C$. The "inconsistency" of this clause is then:

$$E_1 = ½ (1-S_A) ½ (1+S_B) ½ (1+S_C) \quad (5)$$

where, $S_A = 1$ if A is true and -1 otherwise, etc. It can be verified that E_1 is 0 only when k_1 is satisfied and 1 otherwise. By comparing with (3), the contribution of to the energy function is given in Table 1, noting that at least a third-order network is required. Another clause, $k_2 = D \leftarrow B$, say, would contribute:

$$E_2 = ½ (1-S_D) ½ (1+S_B) \quad (6)$$

to the energy:

$$E_P = \sum_i E_i \quad (7)$$

for the program $P = \bigcup_i k_i$ and +¼ to $J^{(2)}_{[B,D]}$ and-¼ and +¼, respectively to $J^{(1)}_B$ and $J^{(2)}_D$. E_P then is proportional to the number of unsatisfied clauses and is minimized at 0 when all clauses are satisfied. This method of doing logic programming in neural network is addressed as *Wan Abdullah's method*. Note that, the

energy function is additive in the sense that new clauses can easily be added to the knowledge base just by adding respective synaptic values without having to undergo recalculation.

In theory the definition of the energy as chosen guarantees that the minimal energy states are those with least logical inconsistency, irrespective of how clauses are related. Some studies on issues like this and others have been carried out and indicate that CNF clauses are better suited than the usual Horn clauses used in conventional logic programming (Browne and Sun, 2001; Garcez *et al*., 2002).

HEBBIAN LEARNING OF LOGICAL CLAUSES

The Hebbian learning for a two-neuron synaptic connection can be written as:

$$\Delta J_{ij}^{(2)} = \lambda^{(2)}(2V_i - 1)(2V_j - 1) \quad (8)$$

(or, for bipolar neurons, $\Delta J_{ij} = \lambda S_i S_j$), where λ is a learning rate. For connections of other orders, we can generalize this to:

$$\Delta J_{ij....m}^{(n)} = \lambda^{(n)}(2V_i - 1)(2V_j - 1)....(2V_n - 1) \quad (9)$$

This gives the changes in synaptic strengths depending on the activities of the neurons. In an environment where selective events occur, Hebbian learning will reflect the occurrences of the events. So, if the frequency of the events is defined by some underlying logical rule, the respective logic should be entrenched in the synaptic weights.

Wan Abdullah has shown that the logic acquired through Hebbian learning as such is equivalent to that in the synaptic strengths obtained using Wan Abdullah's method provided that Eq. (10) is true:

$$\lambda^{(n)} = \frac{1}{(n-1)!} \quad (10)$$

This has also been verified through computer simulation studies (Sathasivam and Wan Abdullah, 2011). This implies that from the synaptic strengths obtained via Hebbian learning in an environment dictated by some underlying logical rules, these logical rules can be obtained if we can come up with a way of reversing Wan Abdullah's method.

REVERSE ANALYSIS METHOD FOR 3-CNF

We now adopt CNF clauses instead of Horn clauses. We can generalize Wan Abdullah's method in Section 2 to 3-CNF formulae. The following Table 2 shows logical clauses in 3-CNF form and the corresponding connection strengths.

Let us now see how we can obtain clauses from synaptic strengths. Since three-atom clauses will give highest the third order connections, firstly three-atom clauses are captured by inspecting positive values of third order connection strengths. For example, if $J_{[ABC]}^{(3)}$ is positive nonzero, there is a possibility for the possible clauses:

$(A \vee B \vee \neg C)$, or $(A \vee C \vee \neg B)$, or $(B \vee C \vee \neg A)$, or $(\neg A \vee \neg B \vee \neg C)$

If $J_{[BC]}^{(2)}<0$ and $J_{[AC]}^{(2)}>0$ and $J_{[AB]}^{(2)}>0$, then the clause $A \vee \neg B \vee \neg C$ is derived. If any decision cannot be done by using second order connection strength information, then we will look at the first order connection strengths. If $J^{(1)}{}_A$, $J^{(1)}{}_B$ and $J^{(1)}{}_C$ show positive values of connection strengths, then the clause $A \vee B \vee C$ is induced. After obtaining the clauses, the contributions of this clause to various connection strengths (third order connection strengths, second order connection strengths and first order connection strengths) are subtracted from the total connection strengths contributed by the data set. Then the process can be carried out with other three-atom clauses until all the remaining positive values of third order connection strengths is within a tolerance value, determined by the user, to minimize spuriously learnt clauses. The similar steps then carried out for the extraction of two-atom then one-atom clauses.

Next, we capture three-atom clauses by inspecting negative values of third order connection strengths. The corresponding second order connection strengths are inspected to determine which atom is the head of the clause. For example, if value of $J^{(3)}{}_{[ABC]}$ is negative nonzero, there is a possibility for the possible clauses:

Table 2: Logical clauses connections strengths using Wan Abdullah's method

	Connection Strengths						
3-CNF formula	$J_{ABC}^{(3)}$	$J_{AB}^{(2)}$	$J_{AC}^{(2)}$	$J_{BC}^{(2)}$	$J_{A}^{(1)}$	$J_{B}^{(1)}$	$J_{C}^{(1)}$
$A \square \square B \square \square C$	1/16	1/8	1/8	-1/8	1/8	-1/8	-1/8
$A \square B \square \neg C$	-1/16	-1/8	1/8	1/8	1/8	1/8	-1/8
$\square A \square \square B \square \square C$	-1/16	-1/8	-1/8	-1/8	-1/8	-1/8	-1/8
$A \square B \square C$	1/16	-1/8	-1/8	-1/8	1/8	1/8	1/8

$(A \vee B \vee \neg C)$, or $(A \vee C$ v $\neg B)$, or $(B \vee C \vee \neg A)$, or $(\neg A \vee \neg B \vee \neg C)$

If $J^{(2)}_{[BC]}>0$ and $J^{(2)}_{[AC]}>0$ and $J^{(2)}_{[AB]}<0$, then the clause $A \vee B \vee \neg C$ is derived. Similarly, if any decision could not been made by using second order connection strength information, then first order connection strengths are considered. If $J^{(1)}_{A}$, $J^{(1)}_{B}$ and $J^{(1)}_{iC}$ show negative values of connection strengths, then the clause $\neg A \vee \neg B \vee \neg C$ is induced. After obtaining the clauses, the contributions of this clause to various connection strengths are subtracted from the connection strengths. Then, the process is carried on with other three-atom clauses until all the remaining negative values of third order connection strengths is within a tolerance value, determined by the user (e.g., 0.0001), to minimize spuriously learnt clauses. We then carry out the extraction of two-atom clauses then one- atom clauses in similar way.

AGENT BASED MODELLING

After knowing the efficiency of Reverse Analysis method, we will use NETLOGO as a platform to develop Agent Based Modelling (ABM). We will designed an agent based modelling to implement Reverse Analysis. NETLOGO is a multi-agent programming language and integrated modelling environment. There has been continuous development for connected learning and computer-based modelling. Furthermore, it is a well suited method for modelling complex systems that can give instructions to hundreds or thousands of "agents" to operate independently like turtle. It is because it is fully programmable and formed by simple language structure. It can instruct mobile agents move over a grid of stationary agents.

While for the link agents, they connect the mobile agents to make networks, graphs and aggregates which can let the users get more understanding on output of system. Moreover, its runs are exactly reproducible cross-platform. The model can be viewed in either 2D or 3D form. Programmer can choose any interesting agent shapes to design the agent based modelling and some interface builders like buttons, sliders, switches, choosers, monitors, notes, output area in the agent based modelling can be developed too. Those interface builders are ready to use and programmer do not need to write more programming language from them. Later in the next section will carry out the definition and more explanation of simulator and benefits of agent based modelling in NETLOGO.

MODEL OF NEURAL NETWORKS SIMULATOR

Firstly, a simulator of Hopfield networks that using a conventional computer had created instead of every time build up a new network design or store a new set of memories. It saves lots of energies and times for the programmer to rebuild new system from time to time. Thus, a computer program which emulates exactly what the user want needs to construct in order to simulate the action of Hopfield Network. It wills easy the programmer to modify the program and store a new set of data. Thus, an agent based modelling had designed for the user to run the simulator.

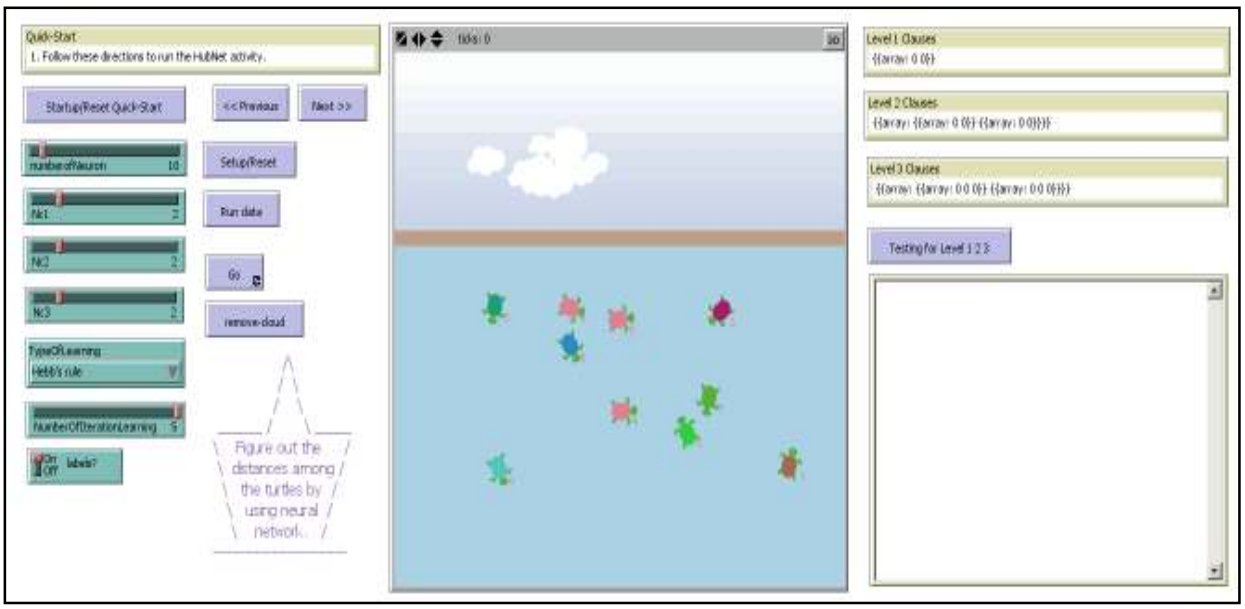

Fig. 1: Layout of the ABM

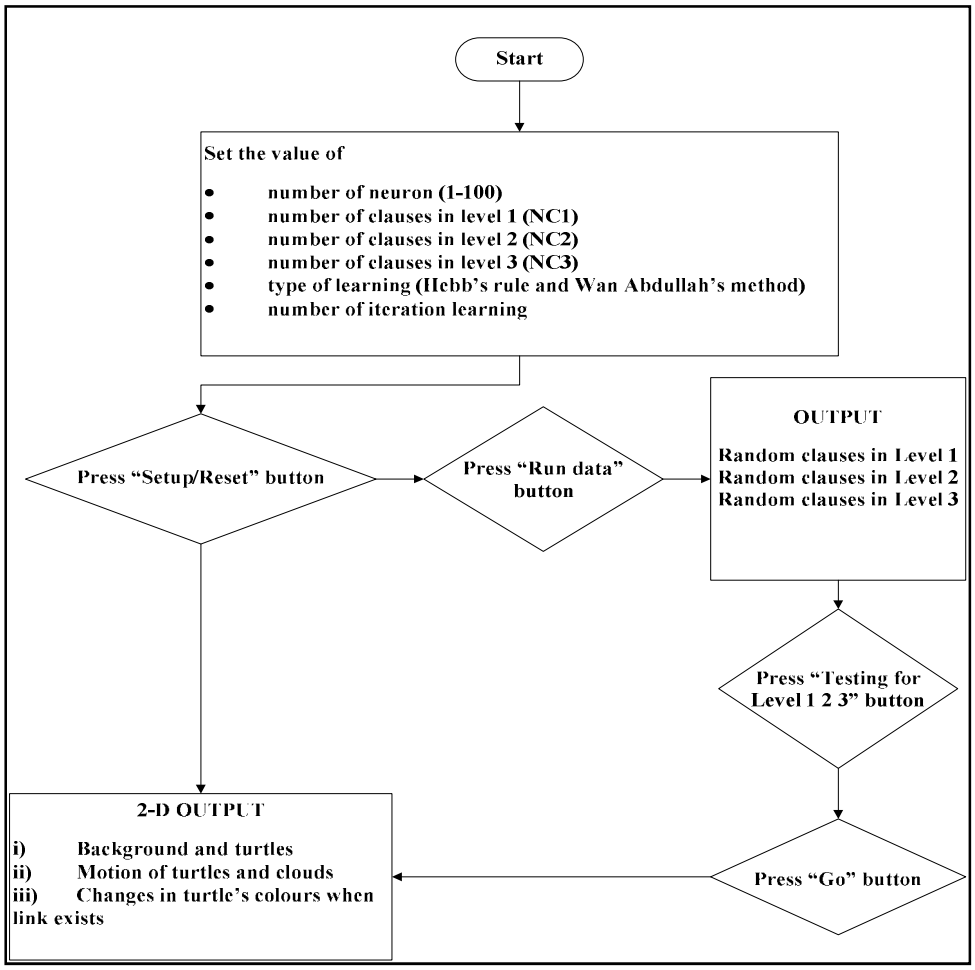

Fig. 2: Flow chart of ABM

Moreover, Agent-Based Modelling (ABM) 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 (Hopfield, 1982; Sathasivam and Wan Abdullah, 2011). 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 (Barry *et al*., 2009; Shamseldin and Adviser-Bowling, 2009) 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. It is because, the agent based modelling paradigm are commonly used in dynamics and complex communities such as telecommunication, health care and others that involving large populations which used explicitly capture social networks.

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 1 shows the layout of the ABM built and Fig. 2 shows the flow chart for the ABM.
Explanation for the flow chart:

Phase 1: Entering values:

- Press the start up/Reset Quick-Start button for the new user.
- Key in NN, NC1, NC2 and NC3. All these values are chosen by try and error technique.
- Later, choose type of learning which is either Hebb's rule or Wan Abdullah's method.
- Next, slide the slider to choose Number of iteration learning and switch on the labels of turtles.
- After all the value had been set, press the setup button.
- Then, press run button to run the program. The program will generate random program clauses.

Example, if user declared NC1 as 4, then 4 first order clauses will be generated.

Phase 2: Training:

- Initialize initial states for the neurons in the clauses:
  Next, based on the idea for the Hopfield network that originated from the behaviour of neurons (particles) in a magnetic field, every particles will 'communicates' to each other to a completely linked forms with each of them are trying to reach an energetically favourable state that means a minimum of the energy function. This state is known as activation. Thus, all neurons in this state will rotate and thereby encourage each other to continue the rotation. So, let the network evolves until minimum energy is reached. The minimum energy corresponds to the energy needed to reach global minima values.
  - Test the final state (state obtained after the neurons relaxed). The system will determine the final state as a stable state if the state remains unchanged for more than five runs.
- The clauses are then extracted from the resulting network by using reverse analysis algorithm and then compared to original set of generated clauses in the aspect of global minima ratio and hamming distance.
- We will run for 200 trials and 100 combinations of neurons.
- Lastly, the system will print out the output for each run and click on the test button for report the result of comparison generated clauses and clauses that obtained.

Phase 3: 2D graphic show:

- After press the test button, click on the go button to run the animation that show a group of turtles based on the number of NN swimming in the river. An internal link will create between them to show the connections among themselves.

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. A sample trial is shown below in Fig. 3 and 4 displaying this agreement. In the output we can see how the turtles are linked together. With this we can unearth the relationships between the data sets. For example let us consider the following example (Sathasivam and Abdullah, 2008):

Reverse analysis method has been tested in a small data set as shown in Table 3. The logical rules induced from the method seem to agree with the frequent observations.

By using Reverse Analysis method approached we was discussed previously, we obtained the following rules:

Bread ← Peanut Butter, Cheese
Burger ← Fillet, Cheese

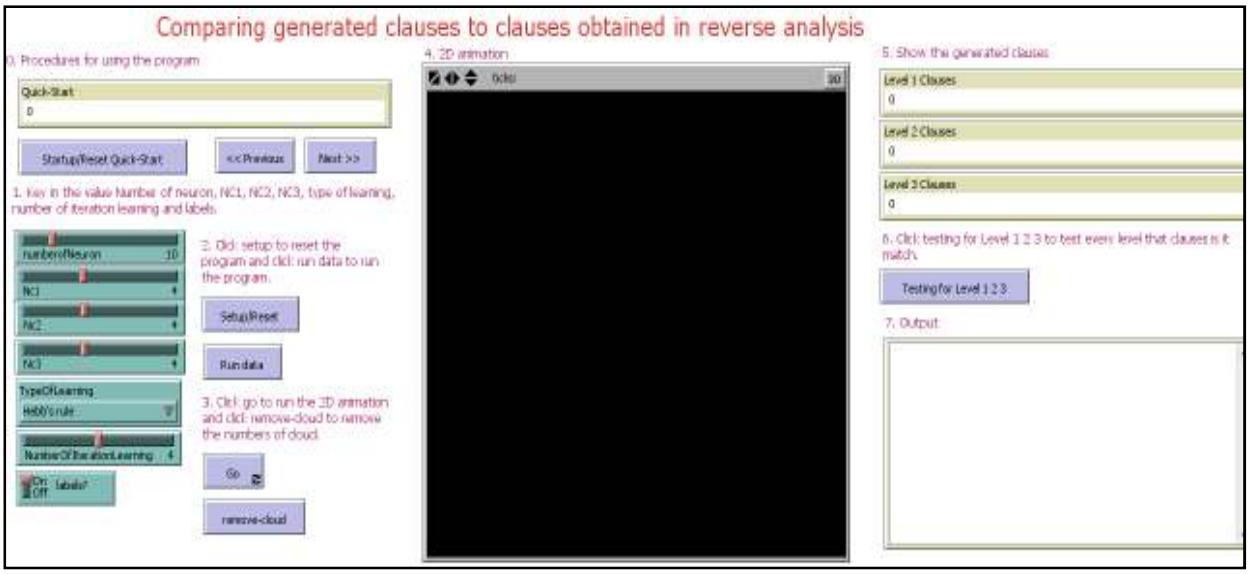

Fig. 3: Output for comparison generated clauses and clauses that obtained

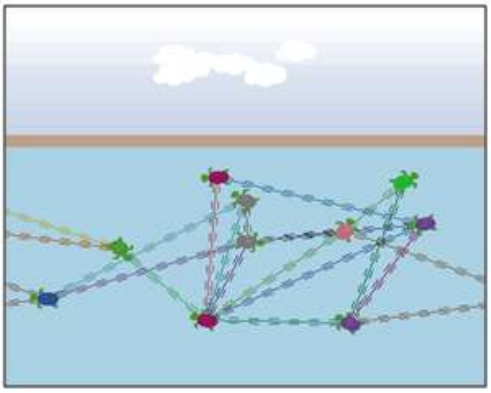

Fig. 4: Testing result

Table 3: Customers daily purchased from a supermarket

	Bread	Peanut Butter	Cheese	Fillet	Burger
Sara	√				
Mike		√	√	√	
John	√				√
Lena		√			√
Susan	√		√	√	

From the rules, we can interpret that a customer who purchased peanut butter and cheese has a high probability of purchasing bread also. Meanwhile, a customer who purchased fillet and cheese has a high probability of purchasing burger.

We developed agent based modelling for this case. We can see how the turtles (bread, peanut butter, cheese etc.,) are linked to each other graphically (Fig. 2).

The logical rules that were induced by using Reverse Analysis method can help the departmental store in monitoring their stock according to the customers demand. Significant patterns or trends in the data set have been identified by using reverse analysis. The departmental store can apply the patterns to improve its sales process according to customers shopping trends. Furthermore, the knowledge obtained may suggest new initiatives and provide information that improves future decision making. By using ABM, user can analyze the graphical design of the links more efficiently and systematically.

CONCLUSION

In this study, we had developed agent based modelling to carry out reverse analysis method by using NETLOGO as platform. The benefits of ABM over other modelling techniques can be captured in three statements:

- **ABM captures emergent phenomena:** Able to produce model for the set of logic programs
- **ABM provides a natural description of a system:** Able to integrate/link Reverse Analysis method
- **ABM if flexible:** We can change the training parameters according to the user. 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.

ACKNOWLEDGMENT

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

REFERENCES

Barry, P.S., M.T. Koehler and B.F. Tivnan, 2009. Agent-directed simulation for systems engineering. Proceeding of the Spring Simulation Multiconference.

Blair, H.A., F. Dushin, D.W. Jakel, A.J. Rivera and M. Sezgin, 1999. Continuous Models of Computation for Logic Programs: Importing Continuous Mathematics Into Logic Programming's Algorithmic Foundations. Retrieved from: http:// citeseerx. ist.psu. edu/ viewdoc/ summary? Doi = 10.1.1.15.9528.

Browne, A. and R. Sun, 2001. Connectionist inference models. Neural Netw., 14(10): 1331-1355.

Garcez, A.S.A., K. Broda and D.M. Gabbay, 2002. Neural-symbolic Learning Systems: Foundations and Applications. Springer Verlag, London, pp: 271, ISBN: 1852335122.

Hamadneh, N., S. Sathasivam, S.L. Tilahun and O.H. Choon, 2012. Learning logic programming in radial basis function network via genetic algorithm. J. Appl. Sci., 12(9): 840-847.

Hölldobler, S. and Y. Kalinke, 1994. Towards a new massively parallel computational model for logic programming. Proceeding of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proc. Nat. Acad. Sci., 79(8): 2554-2558.

Michalski, R.S., I. Bratko and A. Bratko, 1998. Machine Learning and Data Mining: Methods and Applications. John Wiley and Sons, Inc., Chichester, pp: 472.

Sathasivam, S., 2007. Logic Mining in Neural Network. Ph.D. Thesis, Malaysia.

Sathasivam, S. and W.A.T.W. Abdullah, 2008. Logic Learning in Hopfield Networks. arXiv preprint arXiv:0804.4075.

Sathasivam, S. and W.A.T. Wan Abdullah, 2011. Logic mining in neural network: Reverse analysis method. Computing, 91(2): 119-133.

Shamseldin, R.A. and S.R. Adviser-Bowling, 2009. Network optimization of dynamically complex systems. Int. J. Exp. Design Process Optimisation, 1(1): 79-103.

Tsakonas, A., 2004. Towards neural-symbolic integration: The evolutionary neural logic networks. Proceeding of the 2nd International IEEE Conference Intelligent Systems.