Multi Agent System for Artificial Neural Network Training

S. M. Dharmakeerthi[1], A. S. Karunananda[2,]

[1,2] Department of Computational Mathematics,University of Moratuwa, Sri Lanka.

[1]sanethdha@gmail.com, [2]asoka@itfac.mrt.ac.lk

Abstract **- Artificial neural networks are heavily used in the areas of pattern recognition, feature extraction, function approximation, scientific classification, control systems, noise reduction and prediction. Feed-forward and back-propagation neural networks are the most commonly used artificial neural networks. Many researchers face difficulties when selecting a proper ANN architecture and training parameters. The manual ANN training process is not the best practical solution because it is a time consuming task. Also most of the people conduct the manual process in an ad-hoc manner without having a proper basis for changing parameters. This research project has developed a multi-agent based approach to automate the optimization of neural network architecture and its training for feed-forward and back-propagation neural network. The ontology of the agent system comprises of commonly used heuristics for training of neural networks. Our experiments show that the more rational results can be obtained from the system with both simple datasets like XOR as well as with real life data sets. We can conclude that the neural network optimization and training tasks can be successfully accomplished by the agent based approach by analysing the results of the evaluation.**

Keywords- *Artificial Neural Network, Neural Network Training, Multi Agent Systems*

1. INTRODUCTION

Artificial neural network (ANN) is a very prominent artificial intelligent (AI) technique in the area of pattern recognition, function approximation, scientific classification and control systems [1]. Feed-forward and back-propagation neural network (FBPNN) is commonly used to perform the above activities [2] [3]. Selecting the appropriate neural network architecture is difficult to most of the people. The key problem is in deciding the number of hidden layers and the number of neurons that each hidden layer should contain. If the appropriate number of layers and the correct number of neurons in the hidden layers are not selected properly, the neural network will not be able to model the problem in a correct way and in turn results in poor output [4]. Since learning cycles on ANN take long time, performing repeated training and evaluating ANN while changing the hidden layer architecture manually would not be the best practical solution.

This project, MASAnnt (Multi Agent System for Artificial Neural Network Training) proposes a multi agent based approach to automate the neural network optimization and training tasks. In this proposed project, a group of expert agents works collaboratively to perform the neural network optimization and training. As an output, it will deliver well trained neural network with most rational neural network architecture without any human interaction. Most of the artificial neural network users are non-AI experts. Even a user who does not have much knowledge of artificial neural network can easily use this toolkit to perform their tasks. Hence those who do research in the areas like biology, zoology, chemistry, geology and medical science can be highly benefited by using this toolkit. Multi agent technology is a popular AI technology. Agents are fascinated by their simple characteristics and most of the time these agents do small jobs. They use only a very few resources and keep themselves idle after finishing the task. These agents can perform their tasks in a parallel and independent manner. The most significant feature of these agents is their ability to solve problems through communication, coordination and negotiation. Multi agent system is highly valuable when needed to model complex systems. So it is an ideal system of a model human brain which has a high density of complexity (high degree of interconnectivity and high degree of complexity) [7] [8].

The authors see the vital need of a system that can provide the most suitable and rational artificial neural network architecture for a given input and output data set or for a given problem.

2. CURRENT MOVEMENTS IN ARTIFICIAL NEURAL NETWORK TRAINING

An artificial neural network is a popular information processing mechanism which is inspired by the biological neural system of our brain which processes the information and makes inference. It works in parallel and can

successfully solve nonlinear problems. ANN is made by a collection of highly interconnected procession units working collectively to solve a specific problem. Like a human brain, ANN also learns by example. Similar to the human brain, ANN also makes adjustments to the connection in between neurons [9].

Artificial neural networks are different from conventional programming because conventional programs use algorithmic approach, whereas the ANN uses a non-algorithmic approach. Mainly ANN is used in the areas of classification (pattern recognition, feature extraction), noise reduction (where removing of the noise elements is input data) and prediction (forecasting based on historical data).

The two types of ANN training mechanisms are supervised learning and unsupervised learning. Feed-forward and back-propagation is the most popular algorithm in supervised learning, where errors are propagated backwards through the hidden layers and the weights are adjusted accordingly. This is the most commonly used artificial neural network technique.

2.1 Manual Processes in Neural Network Training

In most of artificial neural network projects the training process is done manually. Trial and error method is used to determine the training parameters where several ANN models are developed by comparing the results with one and another [10]. Previous experiences and the intuition of the person are essential in selecting the correct structure. Therefore trial and error method is not a practical method as the ANN learning phase takes considerable time and needs human interaction at the end of each cycle and may cause many difficulties to the new researchers and to the ordinary people.

If the selection of the amount of hidden neurons are too large, the training time will be very high and may results in the neural network with over fit data [4] [6]. This may end up with local minima. If the hidden neuron amount is less than the required amount it cannot be converted to a good stage [5]. Learning rate of a neural network is to do adjustments to weights. If this is high, it might have quick converge or might diverge from solution. If too small, it will take longer time to converge to a final solution [11]. Momentum is helpful to skip from local minima. Incorrect momentum value some time slows down the ANN training, and in some cases can become unstable [12]. There are several activated functions available (threshold, piecewise linear and sigmoidal) so researchers have to select the appropriate function according to the nature of the data set. If E_{max} or error goal is too high we will get a half trained neural network. If it is too low, the ANN has a risk of never converging into a solution.

Accordingly, If the correct structure and parameters are not selected it will be impossible to get the required results from the artificial neural network and also affect the computational efficiency of the system.

2.2. Approaches to Neural Network Training

Amaresh and his team [13] of Anusandhan University, India has done a research and has proposed an experimental structure optimization technique for ANN using adaptive PSO (Particle Swarm Optimization), GA (Genetic Algorithm) and Performance Analysis Based on Boolean Identities like AND, OR and XOR. This includes the optimizing the architecture and the weights of ANN with minimum training error and with optimal number of hidden layers and nodes. They have run both the PSO and GA optimization until max error 0.001 is reached. In this research, the optimization is limited to the number of hidden layers and their neurons. Other training parameters like learning rate, learning momentum and activation functions were not the main focus of this project.

Miloš and Miroslav [10] have proposed an application of Taguchi Method for the optimization of the ANN model where the training is done by Levenbrg-Marquardt algorithm. In their case study, there were 4 input neurons and one output neuron to get the results. Though they have addressed more training parameters the training was done using a selected predefine parameter set. Always the number of hidden layer count is fixed to 2. So this approach did not provide a generalized solution.

XLMiner™ [14] is a data mining add-in for Excel that provides the data classification service using artificial neural network. In the neural network training it does the optimization task to get the suitable number of hidden layers and the number of neurons should contain in the hidden layer. In this case, not much intelligent work is happening, it keeps adding layers and neurons until it is satisfied with the results. This system forgets about the tuning of other main two artificial neural network parameters; learning rate and momentum.

Nayer [15] and a team from the University of Waterloo present an approach to get the optimal number of hidden nodes in a neural network. According to their research paper they have experimentally provided that best performance of the neural network occurs when the hidden node count is equal to $\log_2(T)$, where T is the number of training samples.

Apart from the above mentioned tools there are lots of artificial neural network tools available. Neuroph Studio, FANN (Fast Artificial Neural Network Library) and Neural Network Toolbox are few examples. The major disadvantage of these tools is that the user should have a better understanding about how the neural network function and need to conduct the tuning process manually.

3. MULTI AGENT BASED APPROACHES TO MODEL COMPLEX SYSTEMS

Multi agent systems have opened a new paradigm in computer world, mostly in the field of artificial intelligence. Agents are simple software programs that act autonomously in distributed environments. It has shown prominent achievements in problem solving in complex systems [18].

3.1 Complex Systems

Conventional software development technologies cannot provide good solutions in the domain of complex systems. Complex systems are equipped with high level of uncertainty, hence the behaviour of a complex system is difficult to predict. High degree of interactions among diverse entities (Agents) results in this uncertainty [19]. These diverse agents have rich interactions among them and are interdependent on each other. These agents are self-controlled based on defined rules and laws. They have their own autonomy and also have self-organization capabilities. Because of the high degree of nonlinear interactions they have come up with emergence solutions. It is not possible for a complex system to return to the previous system. Therefore it is said that complex systems are far from equilibrium (non-equilibrium).

3.2 Attempts to Solve Complex Problems using Multi Agents Technology

Many successful researches have been conducted to give solutions to the problems in complex systems. Multi agent technology has shown remarkably good results in problem solving in complex systems which is discussed below.

A research [20] has successfully provided a multi agent solution to vulnerability analysis of power grid in north China, by replaying existing reductionism mechanism. This power grid is a typical multi-level complex system. SCALA [7] toolkit (Cooperative System of Software Autonomous Agent) has been developed by Irene to design complex systems using multi agent technology which is executed on top of JACK. Another [21] successful multi agent system was developed by University of Michigan-Dearborn for complex vehicle fault diagnostics and health monitoring. Two set of agents works autonomously in this system. Signal agents do the monitoring and fault diagnosing the signals coming from sensors. Special case agents are used for detecting faults on specific components. Research [22] done by Nanjing University of Aeronautics and Astronautics shows that multi agent technology can provide a successful solution to complex mechatronics. This project has combined the agent concept with a physical control system and produce controller agents. By using these controller agents they devolved a multi controller system (multi agent system).

By analyzing research done by George Rzevski from The Open University, UK [19] proves that multi agent systems can model and solve problems in complex environments in many domains. One research provides a dynamic scheduling solution for rent-a-car business domain [23]. This was considered as the first industrial multi-agent system developed for dynamic scheduling. This was successfully implemented in the largest taxi service in London [24].

Like mentioned above they have carried out a lot of research and development to provide solutions to the problems in complex environments like real-time scheduling (taxis, air taxis, car rentals, seagoing tankers, trucks, space craft's), dynamic data mining, dynamic knowledge discovery and semantic search.

By deeply analyzing the research done by people in an around the world to provide solutions to problems in complex systems, it can be clearly stated that multi-agent technology have a solid basis for addressing the issues in complex environments.

4 MULTI AGENT TECHNOLOGY FOR NEURAL NETWORK TRAINING

At a glance anyone might wonder whether a solution can be given to the neural network training problem using an expert system. The

simple answer is yes; we can develop a set of rules for all parameters and develop an expert system to perform this task. However, with the same set of rules different experts might execute those rules in a different manner. So in this case it is highly valuable to have a set of experts, to let them communicate and see each other's inferences. Definitely a much better solution can be brought across rather having one expert.

When analyzing closely neural network training also create a complexity. There is a high level of unpredictability of the final result of the neural network training. This uncertainty is the result of the higher interactions among these training parameters. These parameters have their own autonomy. They are governed by their own rules but they have a high level of interdependency. In the middle of the training, the neural network cannot reverse to the previous stage as it is irreversible. The neural network evolves with its environment and these parameters collectively provide an emerging solution in the training cycle. Because of this training an ANN creates a complex system. Hence, the multi agent technology can be considered as the most suitable approach to automate the neural network architecture optimization.

We postulate that multi agent technology can be used to automate the artificial neural network optimization and training task. Interaction among agents enables the emergence of quality training sessions which cannot be archived by an ad-hoc training sessions conducted by humans. It is straight forward to recognize training parameters such as number of hidden layers, number of neurons in each the hidden layer, momentum, learning rate, E_{max} (Error goal) and the activate function of an ANN as a set of agents. Inherent features of agents includes coordination, communication and negotiation and are able to mimic the ANN optimizing and training process by manipulating theses parameters.

5. DESIGN OF MASAnnt

Figure 1 shows a high level design of the developed system. It contains four main modules; System Control Agent, Training Unit, Expert Agent Unit and Ontology. Except the Ontology each module has at least one agent. System control agent work as the main request agent and the agents that are functioning in Expert agent unit are designed as resource agents. Agents inside the Training unit work as task agents. The ontology that is shown in the Figure 1 can be accessible by every agent in the system. All agents are designed to run on top of the JADE agent development framework.

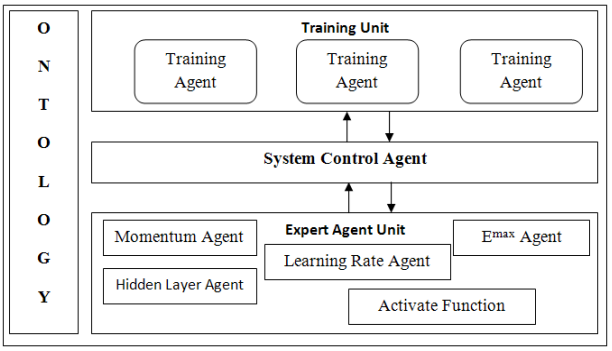

Figure 1: High Level Design of MASAnnt

5.1 System Control Agent

This is the main control unit of the system. Users will directly interact with this unit to do their training tasks. System control agents carry out the start and termination of the training sessions and request the parameter proposals from expert agents. This agent has several behaviours.

1) GUI behaviour

The main task of this behaviour is to enable the visibility of the activities of the system to the end users. The GUI contains necessary components to control the system. There are several components in the GUI to show the agent communication and the message space. Also it shows the latest updates on the status of each and every agent in the system. Separately it shows how the training process is functioning and how the error rate behaves. All past training attempts and their results are also visible in the GUI. All these information will be in single frame so the end user does not need to do any task to see this information. The system control agent will initiate the GUI and then it makes sure that each and every agent can access it. To accomplish that, system control agent will share a reference of the GUI in the common ontology. Because of that other agents can easily access the GUI and do the necessary updates. But key updates of the training process will be done by the system control agent itself.

2) Negotiation behaviour

This is the key behaviour in the system. All the intelligent results are highly depended on the functionality of this behaviour. System control agent conducts and manages the negotiation process. This behaviour includes sending the training data set to expert agents, monitoring the

negotiation, controling if needed and sending the final results to the training unit to conduct the training session. If the negotiation goes beyond the predefined threshold time the system control agent takes necessary actions to finish it quickly and start the training task with available data. All the threshold values are stored in the common ontology so users can do the changes without any effect.

3) Message space behaviour

The system control agent will create a common profile and keep it under its control. This will ensure that each and every agent write messages and read them when needed. This is very much useful in agent negotiation and indicated the status of agents to others. So the expert agents can update their proposing values and suggestion in the negotiation phase. Because of this message space, it reduces the communication overhead at the negotiation phase.

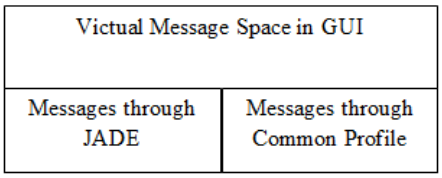

Figure 2: Message Space

As shown in the Figure 2 there will be a virtual message space in the GUI that contains all the messages that are communicated through the common profile as well as directly using JADE. Because at the end, users need not be aware of how the agent communication is conducted. For the users, the most important thing is what the agents are communicating.

5.2 Training Unit

This unit has a set of training agents. These agents will execute training tasks with the training parameter sets provided by the System Control Agent. More than one training agent can be executed in parallel if sufficient hardware resources are available. Currently this is only designed for one training agent. But later this can be designed to work with remote training agents then we can execute more parallel training sessions. Training agent should send training information to the system control agent to display in the GUI.

5.3 Expert Agent Unit

There are set of expert agents in this unit where each is an expert on one training parameter such as one is an expert on hidden layers, one may be with momentum, one for activation function, one for E_{max} and one for the learning rate. With the communication among these agents, they will negotiate about the ANN training parameters that need to be setup.

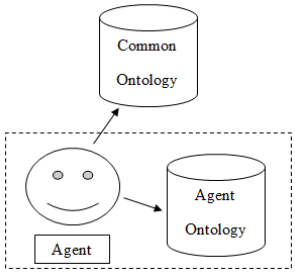

Figure 3: Expert Agent and Ontology

As shown in Figure 3 each expert agent has its own local ontology. Accordingly, agent code only contains how to do the negotiation and the communication. All the agent specific knowledge is located inside the local ontology. So user can add or remove any knowledge without modifying the agent code. This will simplify the agent code. When a new data set is received from system control agent these expert agents quickly move to active mode and suggests the initial parameters which are needed to start the first training session. Depending on the results of the previous training session they again start another negotiation session to decide the next set of parameters. These agents will continue this cyclic process until they get terminated by a system control agent.

5.4 Ontology

All the common knowledge required for agents to work is stored in here. This contains names of agents, the syntax needed for communication, negotiation rules and threshold values. Information about all previous training sessions are also saved in here. This is accessible for all the agents and this common ontology does not have any contradictory facts with any expert agent local ontology. Also this does not contain any information for deciding the neural network parameters. As mentioned in the above section expert agent specific information will be stored in their own ontologies.

6. IMPLEMENTATION OF MASAnnt

JADE was used as the multi agent framework. JADE was selected because it provides fast and easy framework for agent application development. The Java based development

environment was used to implant the prototype on top of JADE. NetBeans was used as the development framework.

6.1 Implementation of User Interface

All the UI components are embedded into one single window as shown in Figure 4, so the users need not navigate through many windows. This enhances the usability of this toolkit. Component 1 shows the colour codes used to show the agent status. Component 2 is the only input field in this application which is used to specify the training data set. In the 3rdcomponent, it shows the current proposed values for the training parameter and also shows the status of each expert agent by using the colour codes explained in component 1. The component 4 and 5 illustrates the message space and the system log respectively. Component 6 gives a real-time visual understanding on how the error is reduced. Component 7 keeps information on previous training sessions.

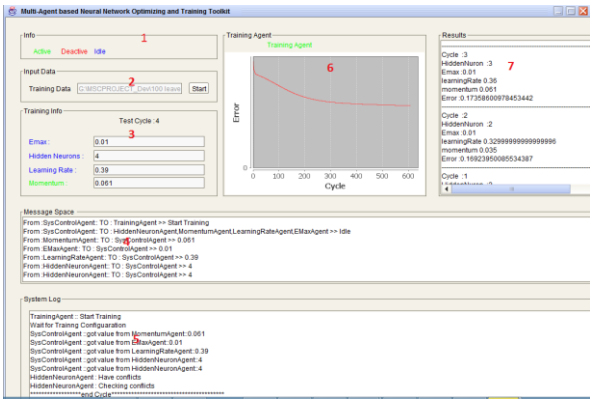

Figure 4 User Interface of MASAnnt

Component 1 shows the colour codes used to show the agent status. Component 2 is the only input field in this application which is used to specify the training data set. In the 3rd component, it shows the current proposed values for the training parameter and also shows the status of each expert agent by using the colour codes explained in component 1. The component 4 and 5 illustrates the message space and the system log respectively. Component 6 gives a real-time visual understanding on how the error is reduced. Component 7 keeps information on previous training sessions.

6.2 Implementation of System Control Agent Behaviours

First the control agent will process the data set and make them into a system understandable format. It will assign a unique key to each type of data set. After all data processing is over system control agent will then activate all the expert agents. Then system control agent will send a request message to all expert agents to get their proposals on the training set parameters. Control agent will search in the DFService to receive the information about all expert agents available in the system. Then it adds those expert agents as receivers and sends the message. Then expert agent will be in the loop for a predefine time. Within this time period expert agent can do the negotiation among them and propose the values.

6.3 Training Agent

When the training agent received the training parameter set it will initialize the neural network that is configured to that agent and start the training session. At the end it will provide results to the control agent. By configuring, this agent can add any external neural network framework. Since this implementation separate the parameter deciding and network training. Either of this can be changed any time.

6.4 Expert Agent

Every expert agent has two Java classes. One is for agent functionality and other is for the local ontology.

The source codes of all the agents are identical except for the reference code of their ontologies. Because of that all agents function in the same way. The difference of their decision making is powered by their ontologies. Each agent conducts its local ontology to conduct the negotiation.

6.5 Ontology

Currently the ontology was developed as a Java class and it contains all the syntax needed for the communication to share information among other agent to store the history of the training sessions. Communication ontology contains all needed syntaxes to the communication. Common message will be wrapped in this class to ensure maintainability. Common ontology contains all history of training sessions kept in a result list. Agent can easily find historical data by navigating through the list available in the result list object. Training parameter ontology is used for temporary keeping the values of training parameters. Expert agent updates this ontology to share their proposals in the negotiation stage.

7. EVALUATION OF MASAnnt

One of the major requirements of this prototype was the usability. As mentioned in earlier chapters it can be used by any person who has little knowledge about artificial neural networks. According to the requirements, the system only requires the training data set as the input. User interface only has one input location and the insertion of text file contains the training data set. So the user does not have to select any artificial neural network parameter for the training. Therefore, the input requirement is successfully completed in this prototype.

Adhering to the requirement this prototype displays all information about the training session conducted and also shows the live view of the training session. The output requirements were also successfully completed in this prototype. Later in this paper the results and the behaviour of the developed prototype will be discussed in depth.

7.1 Evaluation with XOR

XOR gate was used to conduct the initial evaluation. Since the neural network training takes longer time for converge, authors have used this XOR data pattern, so the results can be seen in a less time duration.

Table 1: XOR training data

Inputs	0.0,0.0	0.0,1.0	1.0,0.0	1.0,1.0
Outputs	0.0	1.0	1.0	0.0

Table 2: XOR training results

cycle	Hidden Neurons	Emax	Learning Rate	momentum	Error
1	2	0.01	0.3	0.035	0.2573
4	6	0.01	0.3600	0.0402	0.0205
6	6	0.01	0.4000	0.0428	0.0104
8	8	0.01	0.4400	0.0453	0.0099

- Evaluation with XOR

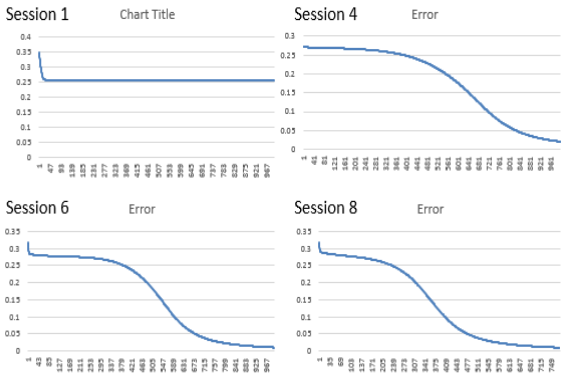

Figure 5: Error reduction in XOR

By analysing the results in Table 2 and Figure 5 it can be easily concluded that the system was able to provide a more rational solution. It is able to reduce the error rate to reach the E_{max} value. At the training session 8 the system was able to converge to the target. This test was a good evidence that the developed prototype is functioning correctly. The test reached the error goal without any problem and since the training data set was small, functionality of it could be easily monitored.

8. EVALUATION WITH IRIS DATA SET

This evaluation was conducted by using the available UC Irvine Machine Learning Repository (The University of California, Irvine). This is a well-known data set in the area of pattern recognition literature.

8.1 50, 000 training cycles

50,000 training cycles were used in each training session, if the targeted E_{max} value could not be reach by the system; it starts a new session with a new set of parameters.

Table 3: XOR training results-50, 000 training cycles

cycle	Hidden Neurons	Emax	Learning Rate	momentum	Error
1	4	0.01	0.3	0.035	0.3020
4	8	0.01	0.39	0.087	0.1979
8	16	0.01	0.630	0.19099	0.1170
16	32	0.113	0.7500	0.2429	0.107

If the system cannot reduce the error rate to the given E_{max} value within a pre-define number of training sessions, system will automatically

increase the E_{max}. Since this Iris data set has lot of values it takes longer time to train. So threshold number of session was set to 10 in this evaluation.

8.2 100,000 training cycles

Another training session was conducted using 100,000 cycles in each session.

Table 4: XOR training results-100, 000 training cycles

cycle	Hidden Neurons	Emax	Learning Rate	momentum	Error
1	4	0.01	0.3	0.035	0.3020
4	10	0.01	0.39	0.087	0.1979
12	32	0.0225	0.6300	0.19099	0.1170
16	46	0.1139	0.7500	0.24299	0.1073

It is clear that by increasing the number of training cycles in a session have a positive impact on the results. By analysing the results in Table 3 and Table 4, it can be concluded that the developed prototype is capable of doing the neural network optimization task. Starting with a higher error rate, it is able to reduce the error rate to the targeted value.

9. CONCLUSION

Starting from the initial training parameter values, expert agents changes their parameters to reduce the error rate by negotiating with each other. The evaluation shows that the MASAnnt was able to reduce the error rate to the given target E_{max} value. The toolkit only requires the training of the data set to be performed. So as expected initially, this MASAnnt can be used by any person; the user need not be an artificial neural network expert. Because of that researchers from any domain can use this tool to conduct their research easily.

Defining the expert agent's rules was the major problem faced in this research. It was hard to find the rules that could define the values of the training parameters. After a lot of reading and experiments a simple rule set was implemented in the expert agent ontology. Most of the rules behaved in a heuristic manner to select the training parameters. These rules simulated the training session conducted by an artificial neural network expert.

Since the agents were developed in a manner that the agent knowledge could be easily edited, the agent ontology can be continuously improved by doing more research. All improvements that are adding to the agent ontology are highly influential to the quality of the MASAnnt. Also new expert agents can be introduced to the system to enhance the results. Since the artificial neural network has longer training time, this toolkit should be deployed in a high-end server, web service or in a similar technology for the users to get the service without having to bear any additional computational cost.

REFERENCES

[1] L. N. Long and A. Gupta, "Scalable Massively Parallel Artificial Neural Networks," in *Aerospace Conference- American Institute of Aeronautics and Astronautics*, 2005.

[2] J. Heaton, "A Feedforward Neural Network," Heaton Research, Inc., 2012. [Online]. Available: http://www.heatonresearch.com/node/704. [Accessed 10 September 2012].

[3] P. McCollum, "An Introduction to Back-Propagation Neural Networks," Seattle Robotics Society, [Online]. Available: http://www.seattlerobotics.org/encoder/nov98/neural.html. [Accessed 2 September 2012].

[4] DTREG : Software For Predictive Modeling and Forecasting, "Multilayer Perceptron Neural Networks," DTREG : Software For Predictive Modeling and Forecasting, [Online]. Available: http://www.dtreg.com/mlfn.htm. [Accessed 5 September 2012].

[5] S.-i. Kazuhiro and K. Maizuru, "A Two Phase Method for Determining the Number of Neurons in the Hidden Layer of a 3-Layer Neural Network," in *SICE Annual Conference 2010*, 2010.

[6] A. Kretinnin, Y. Bulygin and S. Valyuhov, "Intelligent Algorithm for Forecasting of Optimum Neurons Quantity in Perceptron with One Hidden Layer," in *International Joint Conference on Neural Networks (IJCNN 2008)*, 2008.

[7] I. Degirmenciyan-Cartault, "A Multi-Agent Approach for Complex System Design," 2002. [Online]. Available: http://ftp.rta.nato.int/public//PubFullText/RTO/EN/RTO-EN-022///EN-022-05.pdf. [Accessed 2012 August 27].

[8] P. Stone, "Multiagent Systems," 24 September 1997. [Online]. Available: http://www.cs.cmu.edu/afs/cs/usr/pstone/public/papers/97MAS-survey/node2.html. [Accessed 26 August 2012].

[9] C. Stergiou and D. Siganos, "NEURAL NETWORKS," Imperial College London, [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Accessed 7 November 2012].

[10 J. M. Miloš and R. R. Miroslav, "Optimal Selection of ANN Training and Architectural Parameters Using Taguchi Method: A Case Study," *FME Transactions,* vol. 39, no. 2, pp. 79-86, 2011.

[11 R. Berteig, "Basic Concepts for Neural Networks," Cheshire Engineering Corporation, 2003. [Online]. Available: http://www.cheshireeng.com/Neuralyst/nnbg.htm. [Accessed 1 Octomber 2012].

[12 University of Wisconsin, "A Neural Network Approach For Interpolating Species Density Patterns From Remotely Sensed & GIS data: An Example Using The Desert Tortoise," University of Wisconsin, [Online]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/tort.html. [Accessed 1 Octomber 2012].

[13 S. Amaresh, K. P. Sushanta and P. Sabyasachi, "Optimization of ANN Structure Using Adaptive PSO & GA and Performance Analysis Based on Boolean Identities," *International Journal of Computer & communication Technology,* vol. 2, no. VIII, pp. 70-77, 2011.

[14 XLMiner, "XLMiner Data Mining Add-in For Excel," Frontline Systems, Inc, 2012. [Online]. Available: http://www.solver.com/xlminer-data-mining. [Accessed 5 November 2012].

[15 N. Wanas, G. Auda, M. S. Kamal and F. Karray, "ON THE OPTIMAL NUMBER OF HIDDEN NODES IN A NEURAL NETWORK," in *Canadian Conference on Electrical and Computer Engineering*, Waterloo, 1998.

[16 Advanced Agent-Robotics Technology Lab, "Multi-Agent Systems," Carnegie Mellon University, [Online]. Available: http://www.cs.cmu.edu/~softagents/multi.html. [Accessed 5 2 2013].

[17 G. Rzevski, "Modelling Large Complex Systems Using Multi-Agent Technology," in *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2M12.

[18 X. Dong, G. Xiong, J. Hou, F. Dong and T. R. Nyberg, "Vulnerability analysis of power grid based on multi-agent complex systems," in *IEEE International Conference on Service Operations, Logistics, and Informatics (SOLI)*, 2011.

[19 Y. Lu Murphey and Z. Chen, "A Multi-Agent System for Complex Vehicle Fault Diagnostics and Health Monitoring," in *15th IEEE International Conference on Engineering of Complex Computer Systems*, Oxford, 2010.

[20 X. Wu, P. Lou and D. Tang, "A Multi-agent Controller on Embedded System for Complex Mechatronics," in *Chinese Control and Decision Conference (CCDC 2009)*, Guilin, 2009.

[21 G. Rzevski, S. Andreev, P. Shveykin, P. Skobelev and I. Yankov, "Multi-Agent Scheduler for Rent-a-car companies," in *Forth International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Linz, 2009.

[22 G. Rzevsk, A. Glaschenko, A. Ivaschenko and P. Skobelev, "Multi-Agent Real-Time Scheduling System for Taxi Companies," in *8th Int. Conf. on Autonomous Agents and Multiagent Systems*, Budapest, 2009.

[23 G. Rzevski, P. Skobelev and V. Andreev, "MagentaToolkit: A Set of Multi-agent Tools for Developing Adaptive Real-Time Application," in *Third International Conference on Industrial Applications of Holonic and Multi-Agent Sys*, Regensburg, 2007.

[24 D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning respresentations by back-propergation errors," *NATURE,* vol. 323, pp. 533-536, 1986.

[25 B. Krose and P. v. d. Smagt, An Introduction to Neural Networks, University of Amsterdam, 1996.