Neural Networks as Improving Tools for Agent Behavior

Alketa Hyso [1], Betim Çiço [2]

**[1] Department of Informatics, "Ismail Qemali" University,
Vlora, Albania**

**[2] Computer Engineering Department, Polytechnic University
Tirana, Albania**

Abstract
Current trends in software development show a move towards supporting autonomous, rational components (agents). One of the most interesting issues in agent technology has always been the modeling and enhancement of agent behavior. In this paper we are focused in the intersection of agent technology and machine learning techniques for producing intelligent agents. Our application shows that using neural network techniques we improve the reasoning mechanism of our agent supplying to it a new behavior which it did not possess from the beginning. The learning process can be applied initially to train 'dummy' agent to further improve agent reasoning. The machine learning algorithms allow for an agent to adequately respond to environment changes and improve the behavioral rules or acquire intelligent behavior. A case study will be given to demonstrate such enhancement. We simulate the behavior of a robot moving in an environment with random obstacles. Learning techniques that are added to the reasoning mechanism of this robot enrich his behavior in the dynamic environment, displaying a rational and intelligent behavior.

Keywords: *Mobile robot, Intelligent Agent, Machine Learning Techniques, Neural Network, Agent Behavior.*

1. Introduction

Agent and multi-agent system technologies, methods and theories are currently contributing in many diverse domains. Applications range from small programs that intelligently search the Web buying and selling goods via electronic commerce, to autonomous space probes. This is not only a very promising technology, it is emerging as a new way of thinking a conceptual paradigm for analyzing problems and for designing systems [1], for dealing with complexity, distribution and interactivity [2], [3], [4] , and perhaps a new perspective on computing and intelligence [5]. Numerous approaches exists, attempting to optimally reflect both the inner states as well as perceived environment of an agent in order to provide it either with reactivity or proactivity [6]. Learning is a crucial ability of intelligent agents. Machine Learning tools and techniques enable an agent to get knowledge from observations or experiences [7][8][9][10]. Agent's programming has a specific. Agent works in an environment even if the programmer does not provide him all the knowledge about the environment. If the programmer or designer has incomplete knowledge about the environment where the agent operates, the only way the agent to work in this environment is through learning the necessary knowledge. Machine learning algorithms provide the agent with additional autonomy, the result of his experience. Thus, new agents exhibit behavior that does not possess before. Agents ensure the presentation of knowledge, generation of rules and strategies used in making decisions on their part through inductive learning.

This paper aims to present how machine learning methods can be integrated with agent technology in building more intelligent agents. Simulation shows the behavior of a mobile robot seen in that perspective. The objective of this work is not to study the optimal decision-making algorithm but to simulate the new rational behavior our agent own because intersection of agent technology and machine learning techniques.

2. Multilayer Perceptron Neural Network as Function Approximation

The field of neural networks covers a very broad area. We will concentrate on using neural network as function approximation. Fig 1 presents a neural network as a function approximator.

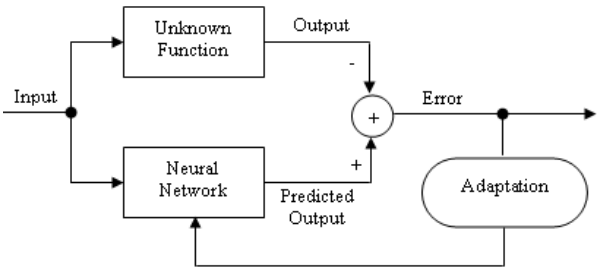


Fig 1. Neural network as function approximator

Two-layer networks, with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer, are universal approximators. We could use such networks to approximate almost any function, if we had a sufficient number of neurons in the hidden layer.

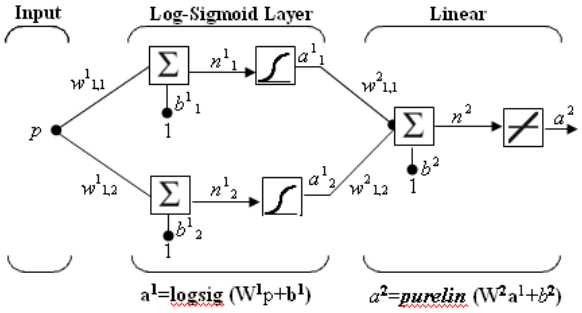


Fig 2. Example function approximation network

Hornik, Stinchcombe and White, [11] present a proof that multilayer perceptron networks are universal approximators. Pinkus, [12] gives a review of the approximation capabilities of neural networks; Niyogi and Girosi, [13] develop bounds on function approximation error when the network is trained on noisy data.

3. The Application

We will represent an application that simulate the new intelligent rational behavior that our robot gains because of the intersection of agent technology and machine learning techniques. Our robot operates in an environment with obstacles. Its goal is to find its way out in this environment. The obstacles' sensors help it to memorize information about the world. Every time those sensors feel an unknown obstacle, the robot acts according to the reflections. We will supply a neural network in robot's reasoning mechanism. It will be trained to learn to measure the distance during its locomotion. So this new ability and information that come from sensors help the robot to create a map, which represents the environment partially. The robot shows an intelligent rational behaviour – it makes its movement plan based on knowledge provided by the map.

The stages we have to follow to realize what we previously mentioned are:

a) monitor the behaviour of the robot in the environment with obstacles based only in the perceptions of the sensors. b) Apply the machine learning technique - MultilayerPerceptron, which is a neural network that trains the robot using backpropagation to measure the distance. c) Creating the map of the obstacles and incorporating this knowledge model in the reasoning mechanism of the agent. d) Monitor the new behavior of the agent.

3.1 The Agent's Reactive Behavior.

Let's consider the behaviour of the robot in the environment with obstacles based only in the perceptions of the sensors. The obstacles are generated randomly; i.e. in random numbers, sizes and distance between them, and their shapes are rectangular.

For convenience, we assume that the mobile robot enters on the left side and exits from the right side of the environment. This robot will be equipped with obstacle's sensors to memorize information of the world.

The robot percepts the environment through sensors and acts on that based on these perceptions.

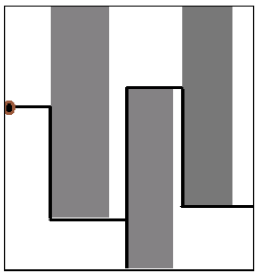


Figure 3. The agent's reactive behavior in a unknown environment

The environment is not completely accessible. The agent should reason about the set of states it can pass, not just for one single state. The agent knows that when the sensor of obstacle gives a signal, it has reached an obstacle, but it doesn't know if the obstacle comes from the bottom or the top. The agent will try the action 'move straight forward' as long as it doesn't encounter an obstacle. When it encounters an obstacle, it moves 'one step downwards and again straight forward'. The agent repeats this sequence

until it finds a free state or reaches the bottom bound. If it finds a free state, it moves 'straight forward'. The set of actions 'straight-downwards-straight' or 'straight-upwards-straight' will reach a free state, if it exists in the environment generated randomly. This is the way our agent acts to realize its goal. Figure 3 presents the reactive behavior of the robot in a unknown environment.

The Agent Movement Algorithm 1

input: *state* (the map of the obstacles), *origin* of the agent, *exit* destination, *percept*, *rules* (that will follow by the agent according to the perception and state), *g* (goal, initially null), *problem formulation*, *s* - an action sequence (the plan of movement).

state ← UPDATE-STATE (*state, origin, exit)*
//the goal is the first obstacle in the map.
g ← FORMULATE-GOAL (*state*)

//the environment is unknown by the agent
Problem ←FORMULATE-PROBLEM *(state, g*)

//solution that will be chosen by the agent.
actionPlan1 : While (agent does not achieve the *exit*) {

if (the agent percepts an obstacle) {
state ←UPDATE-STATE (*state, percept*)

// The agent chooses the action based on perception and rules)
rule ← RULE-MATCH (*state, percept, rules*)

//the agent moves one step downwards and then straight forward'...
action ← RULE – ACTION [*rules*]

//the state is updated after the action
state ← UPDATE-STATE (*state, action*)
}

// the goal is the next obstacle of the map
g ← FORMULATE-GOAL (*state*)

}

The lack of a map of the environment forces it to act just using its reflexes. The plan that our agent follows is not rational. We would have a rational behavior of the robot if its motion is based on a map of the environment created by the robot.We have to improve the robot's reasoning mechanism to fulfill new requirements:

- the agent should build the path through the obstacles before beginning its movement.

For this purpose, we equip the reasoning mechanism of the robot with a neural network, and we train it to calculate the distance of its locomotion, this based on learning techniques.

3.2 Training the Robot to Measure its Locomotion Distance

We shall take into consideration that our physical agent is able to move and calculate the time. After getting the signal from a given sensor, the agent moves through a straight line without obstacles, starting from a position of "rest", to which corresponds a 0 moment of time. Its moving accelerates with a given acceleration that our agent is not able to calculate. We suppose that in equal distances our agent must leave a trace. We should make the agent learn the process of calculating the distance it covers, only based on its capacity to calculate the time.

In order to solve the above mentioned problem, we use sources of data extracted during the movement of the agent. These sources include data about movement: time intervals, the speed of movement in that starting moment, acceleration and distance. Acceleration: 2 m/s2; Time in seconds (0; 1; 2; 3; 4; 5; 6; 7) and distance in meters (0; 1; 4; 9; 16; 25; 36; 49).

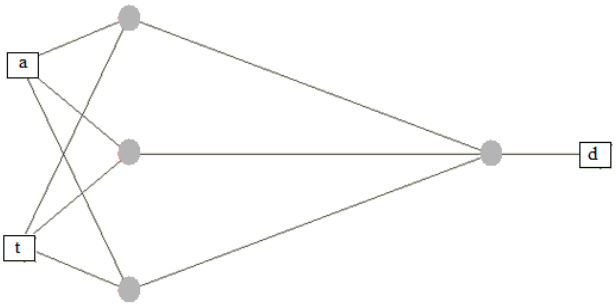


Fig 4. Architecture of neural network

We apply machine learning method – multilayer perceptron neural network, by means of which we can find a model, which can be incorporated within our agent in the next step. Fig 4 present the architecture of the neural network. Our neural network has an input layer that designate the values of the elements in the hidden layer which has with 3 nodes with sigmoidal activation function. Whereas, nodes in the hidden layer designate the value for the output layer element with a linear activation function.

Among the learning algorithm the one that gives the best solution is the backpropagation algorithm (BP) of neural networks which is supported by Multilayer Perceptron method [14] [15].

BP learns the weights for a multilayer network. It employs gradient descent in an attempt to reach the outputs.

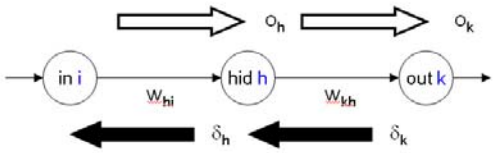


Fig 5. Back-propagation

The learning process has two stages: Forward stage: calculating of outputs based on an input pattern and Backward stage: updating of weights by calculating the errors. Fig 5 shows the stages of back-propagation algorithm.

The back-propagation algorithm in a multilayer network is presented here:

- Initialize all weights to small random numbers. Initialize each Δw_i to 0.
- Until satisfied, Do

//after a fixed number of iterations (epochs) or once the error falls below some thresholds

- For each training example Do

//(training example - a pair of the form $< \vec{x}, \vec{t} >$,

// where $\vec{x}$ is the vector of input values,

// and $\vec{t}$, is the vector of target output values.

1. Input the training example to the network and compute the network outputs
2. For each output unit k

$\delta_k \leftarrow o_k(1-o_k)\cdot(t_k - o_k)$

//ok – output value, and tk – target output value

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1-o_h)\cdot \sum_{k\in outputs} w_{h,k}\delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j}(n) = \eta\delta_j x_{i,j} + \alpha\Delta w_{i,j}(n-1)$$

// η - learning rate, α - momentum, (n)-iteration.

As input parameters we consider time and acceleration, while, as an output we consider parameter the covered distance. Number of iterations (for backpropagation), Nit = 100. We perform the training of the agent according to this algorithm by adjusting these parameters of algorithm such as the number of training cycles, learning rate or even the given model of the neural network. In the end, the neural network adjusts the proper weights.

The model that we gain is a non-linear function of the type:

$$d = -20761.777221\cdot\frac{1}{1+\exp^{-(a\cdot 3.983986+t\cdot(-0.077077)+(-7.592238))}} +$$

$$335.640279\cdot\frac{1}{1+\exp^{-(a\cdot 0.226480+t\cdot 0.378405+(-1.142265))}} +$$

$$-705.166631\cdot\frac{1}{1+\exp^{-(a\cdot 0.912341+t\cdot 0.335880+(-2.606184))}} +$$

$$a\cdot 4967.103182 + t\cdot(-363.815049) + 2483.551591$$

In a second phase we incorporate the neural network within the robot's reasoning mechanism and monitor the behavior of the agent. Now, our robot can measure the distance of movement based on the law which it discovered through the training process. Figure 6, presents the traces that our robot leaves during his movement, based on the covered distances every second.

The upper traces (1) are left by our agent based on the measurement according the knowledge it gained from the training process (approximation function). They are based on the distance that our agent thinks it covers every second. The traces below are the actual values of the distance covered– every second. We see that for the first 50 m that correspond 7 seconds, the distance measured by the agent is equal with the distance it accomplishes. But, if the agent continues moving we can see that there will be faults in the measurement it does. This shows that the function of calculation of the covered distance is valid only for the range of data included in the training process of the agent. This function is not valid for the uncovered data by the training process.

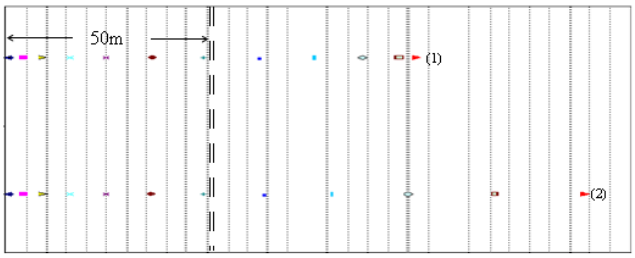


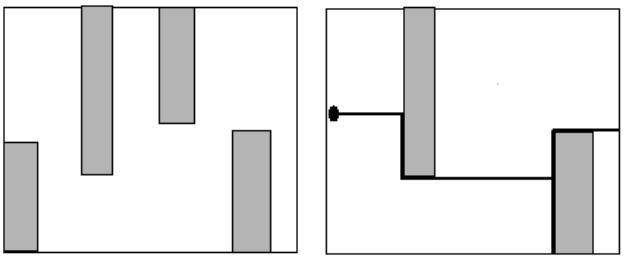
Fig 6. The traces, that our agent leaves based on the distance that our agent thinks it covers every second (1), and covered distances every second (2).

We see that the agent displays a new behavior: it is able to realize an approximate calculation of distance; by developing an intelligent behavior of the agent.

3.3 The Process Of Mapping

We include in our robot the reasoning mechanism - a software agent that creates and recreates, the environment map the way the robot perceives the environment. The map will present the distance between the obstacles encountered during the agent's motion, and the direction of motion (up or down), which the robot will follow to overcome the obstacles ahead. The distance measured by the neural network at the moment it encounters the obstacle, is recorded on the map. Robot moves up or down to avoid the obstacle. When it moves down the obstacle, it registers in a state variable the value (-1), but

when it moves up the obstacle, it registers in this state variable the value (+1). The moment the robot passes this barrier this value is stored in the map, attached to the distance. The robot follows this procedure for every obstacle it encounters until it exits. This way, it creates the map of obstacles and recreates it each time it detects changes in its environment;



(a) (b)

Fig 7. The real environment a) , the environment according to robot perception.

Table 1 the map that the agent creates

obstacle	distance	direction
1	11	-1
2	37	+1
...	....	...
...	....	...

Fig 7 presents the real environment (a), the environment according to robot perception (b), and Table1 presents the map that it creates .

3.4 The Behavior of the Agent Based on the Map.

Since the agent has a map of its environment, it has information about it. It acts based on a plan, made before beginning to move. It follows the same trajectory of motion as in the case of reflex motion, but when facing an obstacle it follows the direction of motion that is stored on the map. So another movement plan is included in the navigation agent - part of the robot's reasoning mechanism. Let's present the following robot's movement algorithm in the known environment with obstacles, based on the map.

The Agent Movement Algorithm 2

actionPlan2: While (agent does not achieve the *exit*) {

//the agent moves in a known environment according to the map
move to achieve the goal
if (the agent percepts an obstacle that is in the map) {
state ←UPDATE-STATE (*state*, *percept*, *map*)
//the agent moves upwards or downwards according to the map.
action ← RULE – ACTION [*rules*]

//the state is updated after the action
state ← UPDATE-STATE (*state*, *action*)
}

// the goal is the next obstacle of the map
g ← FORMULATE-GOAL (*state*)

}

In a dynamic environment it will show reflexive behavior intertwined with pro-active behavior. The agent has one plan for the unknown environment and another for the known one. It will choose which plan to apply and combines them autonomously according to the situation. This way, the agent autonomously creates a movement plan every time it is set in a dynamic environment.

4. Conclusions

This paper aims to present how machine learning methods can be integrated with agent technology in building more intelligent agents. Using machine learning techniques makes it possible to develop agents able to learn from and adapt to their environment. In this way, the performance of these agent systems can be improved. We demonstrate the use of learning techniques to provide a map, which helps generate new behaviors. By using a neural network as a reasoning mechanism and by analyzing input and output, our agent will acquire a new intelligent behavior which it did not possess from the beginning.
This paper is a brief summary of what we have done in the practice of combining agents and machine learning techniques. We will make more efforts to explore this new trend of research. The agents are rational decision-making system: they are able to show their reflexive and pro-active behavior and intertwines these kinds of behavior according to the situation by making the best decisions offered at the moment.

References

[1] Sterling, L and Taveter, K. (2009), 'The Art of Agent-Oriented Modeling,' MIT Press, p 27-112
[2] Buse, D. P., and . Wu, Q. H. (2004). "Mobile agents for remote control of distributed systems," IEEE Transactions on Industrial Electronics, vol. 51, no. 6, pp. 1142–1149.
[3] Dimeas, A. L., and Hatziargyriou,N. D. (2005). "Operation of a multiagent system for microgrid control," IEEE Transactions on Power Systems, vol. 20, no. 3, pp. 1447–1455.
[4] Davidson,E. M., S McArthur,. D. J. J., McDonald,R. T., Cumming, and Watt,I. (2006) "Applying multi-agent system technology in practice: automated management and analysis

of SCADA and digital fault recorder data," IEEE Transactions on PowerSystems, vol. 21, no. 2, pp. 559–567.
[5] Poole, D., and Mackworth, A. (2010), 'Artificial Intelligence: Foundations of Computational Agents,' Cambridge University Press.
[6] Wooldridge, M., (2000), 'Reasoning about Rational Agents, ' MIT Press.
[7] Zhang. et al, 2005, Agents and Data Mining: Mutual Enhancement by Integration, (Eds) Gorodetsky V, Liu J, Skormin V.A, Autonomous Intelligent Systems: Agents and Data Mining, p 50-61, Springer.
[8] Symeonidis, A. et al , 2005, Methodology for Predicting Agent Behavior by the Use of Data Mining Techniques, International Workshop AIS-ADM 2005 Russia, Proceedings.
[9] Remondino, M., et al, 2005, Data Mining Applied to Agent Based Simulation, Proceedings 19th European Conference on Modelling and Simulation Merkuryev Y, Zobel R, Kerckhoffs E, ECMS.
[10]Russell, S.dhe Norvig, P.: "Artificial Inteligence: A Modern Approach." Prentice –Hall 1995, p 525-647.
[11] Hornik, K. et al 1989, 'Multilayer feedforward networks are universal approximators,' Neural Networks, vol. 2, no. 5, 359–366.
[12] Pinkus A., 1999, 'Approximation theory of the MLP model in neural networks,' Acta Numerica, p 143–195.
[13] Niyogi, P., et al, 1999, 'Generalization bounds for function approximation from scattered noisy data,'Adv. Comp. Math., vol. 10, 51–80.
[14] Werbos, P. J., 'Beyond regression: New tools for prediction and analysis in the behavioral sciences,' Ph.D. Thesis, Harvard University, Cambridge, MA, 1974. Also published as The Roots of Backpropagation, John Wiley & Sons, New York, 1994.
[15] Russell, S., et al, P., 1995, Artificial Intelligence A Modern Approach, Prentice Hall, p 581.

First Author. Alketa Hyso: 1988-1993 Electronic Engineering - Polytechnic University of Tirana, Electrical Engineering Faculty; 2003-2005 Master in Computer Engineering - Polytechnic University of Tirana, Electrical Engineering Faculty, Computer Engineering Branch; 2006-2011 PhD student Polytechnic University of Tirana. Professional Experience: 1994-1995 teacher in High Technical School in Vlora, 1996-1997 Vlora Hospital, position Head of Technical Branch. 2001-to date Technological University "Ismail Qemali" Vlore, Albania, position lector of informatics. Head of Editorial Board EPICT Albania. Papers and conferences: "Agents as Rational Decision-Making Systems", bci'07 - 3rd Balkan Conference in Informatics and Albanian Journal of Natural and Technical Sciences , "Application of Machine Learning Techniques and Data Mining to Agents", 3rd Annual South-East Europian Doctoral Student Conference; "Comparison of Different Types of Intelligent Agents: A case study and a simulation" , Albanian Journal of Natural and Technical Sciences ; "Neural Networks as Improving Tools for Agent Behavior", IADIS International Conference in Intelligent Systems and Agents 2009; "Agents as Tools for Solving Complex Control Problems", 4th Annual South-East Europian Doctoral Student Conference; "Agents in the Control Domain Process" 5th Annual South-East Europian Doctoral Student Conference;

Second Author. Betim Çiço: Prof. Dr. Betim CICO was graduated as electronic engineer in Polytechnic University of Tirana (PUT) in 1970 with excelent results. After one year of working as electronic engineer in Shijak Radiostation, Albanian Radio-Television, he worked for 26 years as scientific researcher at the Institute of Nuclear Physics(INP), mainly in the field of microprocessors and computers, computer based on-line systems, Head of the Electronics Department, part time professor in PUT. In 1998 he moved as Head of Computer Engineering Section Electronic Department, Faculty of Electrical Engineering at PUT. During 1999-2011 member of the main governing body (Senate) at PUT. Participated in 50 International Trainings, Workshops, Conferences etc. Author of 40 Scientific Papers and 20 INP Technical Reports. Supervisor of three PhD Theses. During 1999–2001 member of the Project Group, MOES, for the implementation of the Education Management Information System (EMIS) Component under the Transition Education Reform Project In Albania. Designer of several LAN–s in Tirana.