ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

Extending Immediate Reinforcement Learning on Neural Networks to Multiple Actions

Claude Touzet

LERI - EERIE
Parc G. Besse, F - 30 000 Nîmes
Tel : ++ 33 66 38 70 34, Fax : ++ 33 66 84 05 06, Email : touzet@eerie.fr

Abstract : Between supervised and unsupervised learning, connectionism proposes a qualitative learning or *reinforcement learning* which is of interest for applications needing qualitative control. This learning technique is not new and the following advantages of a neural implementation of reinforcement learning are identified: a small memory requirement and a more effective exploration of the situations-actions space. However, the restriction of the applicability of this algorithm to problems with a limited number of actions (usually two) remains. We propose to solve this problem by correctly specifying the output coding of the action on the output cell layer of the neural network and interpreting the output values as a certainty value for doing a specified action. Experiments performed in the real world with the miniature robot Khepera confirm the possibility of extending the applicability of reinforcement learning to cases where multiple actions are possible for each situation.

1. Introduction

From a connectionist point of view, reinforcement learning algorithms are classified between supervised and unsupervised learning. Unlike supervised learning where we have some quantitative data relative to the network performance, the learning is conducted using only qualitative signals. The first neural implementation of reinforcement learning was proposed by Barto et al. in 1981 [2]. The advantages of a neural implementation of reinforcement were referenced in [9]. However, this implementation has serious limitations. For a given situation, only the action which shows the best reward probability is proposed by the artificial neural network. These limitations reduce the use of neural reinforcement to applications with few possible actions (usually two as for the inverse pendulum). In this paper, we propose to overcome these limitations. Experiments carried out in the real world by the miniature robot Khepera demonstrate the use and advantages of our approach.

2. Neural reinforcement learning

Reinforcement learning synthesises a mapping function between situations and actions by minimising a reinforcement signal. The learning is incremental, because the acquisition of the examples is carried out in real situations. A function, usually a random function, allows different situations to be tested. Heuristics, generated by a human operator, qualifies each action undertaken in a situation. The objective is to achieve the acquisition of the best rewarded behaviour for the system. Any difficulty is a result of a situation space being so large, that combined with all possible actions,

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

an exhaustive exploration of all situation-action pairs is impossible. The system must be able to generalise from a proportionally small number of examples. Moreover, the reinforcement signal is a simple qualitative criterion, for example, binary (OK or bad). We will not discuss more complex reinforcement signals, like those used by Millàn [5] ; and we are only interested in immediate reinforcement in which there is no delay in qualifying an action.

3. Neural implementation

The neural implementation of reinforcement learning [4], [7] implies the following modifications:

- The internal state is composed of the weight set of the network (W). The memory size required by the system to store the knowledge is then defined, a priori, by the number of connections in the network. It is independent of the number of explored situation-action pairs.
- The evaluation V proposes an action to be accomplished in the given situation. It is the result of the processing by the network of the input situation i plus a random component b. This component decreases during the learning process. At the same time, the network generalisation improves: the system learns.
- The update function U works on the internal state. It is a weight modification algorithm, like a gradient error descent algorithm. An error signal on the output neurons must therefore be defined (Table 1). Definition of this error is restricted to simple cases where only two actions are possible.

4. Application of reinforcement to multiple actions

In the case of classic algorithms, each reinforcement (positive or negative) associated with a situation is stored. Therefore, a negative reinforcement may be useful for avoiding the selection of the same "bad" action in the same situation the next time. It is thus possible to deal with applications which allow a large number of possible actions to be available in a given situation.

Limitations of neural reinforcement when applied to multiple actions

Contrary to classic algorithms, neural reinforcement restricts learned knowledge. Only the action with the best reward potential is proposed for a given situation. Moreover, the effective value of the expected reward is unknown. We only know that it is the action from which the greatest reward is expected. In fact, dealing with negative reinforcements is difficult. Neural implementation means that we have to "un-map" the selection of a given action in the situation. "Un-learning" is a new paradigm in connectionism. Applications (or toy applications) are currently limited to two actions and are able to unlearn by making the correct association.

Unlearning in neural networks

The first generalisation which comes to mind when the number of possible actions increases is to learn the inverse mapping [2]. The problem encountered is to determine from among all those left which action is the most in opposition. Because the output of the network is numerical, we can change the sign of the output values. However, this is a harmful way of unlearning. Nobody knows what has been deleted.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

Representation on a neural network is distributed, so it is not possible to delete only one mapping without interfering with the rest of the learned knowledge. Moreover, a negative reinforcement does not always mean that the error is important, and to learn the inverse action can be defective. With the same goal in mind, Ackley [1] proposes the use of the complement of the generated output.

Generalisation of neural reinforcement to multiple actions

Our propositions are presented in the three points below:

1/ Reduction in the number of negative reinforcements. This can be done by modifying the heuristics.

2/ Decomposition to elementary actions (Fig. 1) of agonist-antagonist type. In this way, motor actions proposed by the neural network are more numerous. The choice between proposed actions is based on the certainty values of the output neurons: we select here the action corresponding to the maximum output value.

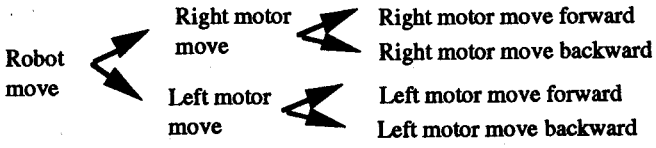

Fig. 1. Example of decomposition in elementary agonist-antagonist actions.

In this case of a robot with two wheels (and two motors), the moving action can be designed as a unique command specifying the direction. It can also be separated into one command for each motor. At the end, it can be decomposed as a competition between forward and backward movement on each motor. We have tested each decomposition on a task of obstacle avoidance with a real robot: this means only one neuron for the complete control of the direction, or two neurons (one for each motor), or four neurons (two in competition for each motor). Experiments show that learning only converges towards a correct behaviour when we have a complete decomposition of actions, i. e. four neurons.

3/ Even if generalisation leads to negative reinforcement, the output error may not be very significant. We want to limit the weight modifications, so we interpret the output values as certainty values of the network in its propositions. Fig. 2 shows how we exchange the values of an agonist-antagonist pair. This exchange limits the size of the error. Interpreting the network output values as certainty values is a common practice in connectionism, in particular in the domain of fuzzy reasoning [8].

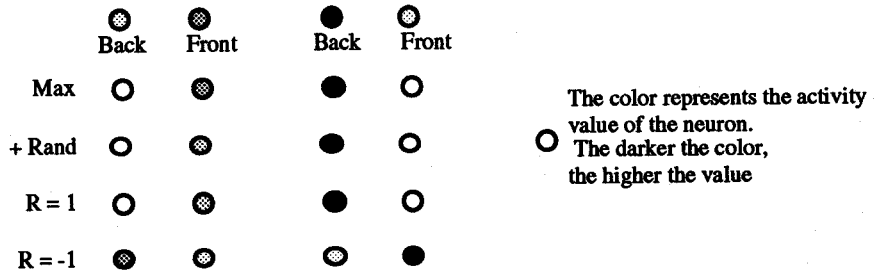

Fig. 2. Application of principle 3.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

Through competition between the neurons in each pair, the neuron of maximum value is selected and a random value is added. If the reinforcement signal is positive then the error is equal to the added random value. There is no error for the other neuron of the pair. If the reinforcement signal is negative then values in each pair of neurons are exchanged. This is a way to modulate weight modifications.

The modified version of neural reinforcement learning for application to multiple actions is shown in table 1. In this example, two output neurons correspond to two competitive actions: one neuron for agonist actions and the other one for antagonist actions. The real output value allows a selection to be made from among all possible actions. The coding is, a priori, continuous: two actions, the effect of which are similar are coded by similar real values. Following this guideline of coding enhances the quality of generalisation .

Table 1 Learning algorithm

1. Random initialisation of the network weights (W_0) .
2. Repeat :
   a. Let i be an input situation for the neural network, and (o_1, o_2) the outputs computed by propagation of activities. The action a perform do is given by : $a = o_{select} + b$, where b is a random signal and o_{select} = Max. (o_1, o_2).
   b. Execute the action a in the world.
   c. Let r be the immediate reward associated with the execution of a. The weights are updated by an algorithm which minimises the output error. It is necessary to determine a desired output value d for each output neuron, depending on r. If r = 0 then there is no modification, if r = +1 then $d_{select} = a$ (and then $error_{select} = b$). Only weights connected to the selected neuron are modified. If r = -1 then $d_1 = o_2$ and $d_2 = o_1$ (exchange of values).

5. Experimentation

Khepera is a miniature robot [6] having a diameter of 6 cm and a weight of 86 g. Two wheels allow the robot to move around. 8 infra-red sensors help the robot to perceive its environment. The detection range is between 5 and 2 cm. Sensor data are integers between 0 (nothing in front) and 1023 (obstacle nearby). Inter-individual variability among sensors is high (it can be 50 %). The location of the sensors on the robot body ensures greater performance in frontal detection. The computational power of the robot is equivalent to that of a Mac. Energy autonomy is 30 minutes.

Behaviours of both obstacle avoidance and forward moving

Khepera must be able to move autonomously in a real environment. Contrary to a simulation environment, here we have sensor noise, control error and dynamically changing environments. Moreover, the solution to the problem, i. e. the map between situation and actions, is completely unknown.

Coding

Each sensor is associated with a neuron. There are 1023^8 input situations (approximately 10^{24}). Each of the four output neurons has a precise semantic : Left

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

motor forward, Left motor backward, Right motor forward, Right motor backward. Each motor can take 20 values, so there are 10 values for each neuron and 400 possible actions per situation.

Heuristics

The heuristics defines the reinforcement signal value r. They have been subject to many modifications. For the learning of an obstacle avoidance behaviour (Table 2), we compare the past and present sensor values. If there is more light than the last measurement then r = 1 (it is OK); otherwise if the present sensor values presents a level which is too high then
r = -1 (it is bad).

Table 2 Heuristics for obstacle avoidance behaviour

Let $i_j(t)$ be the sensor value of j at time t and r the reinforcement signal.
if $(\Sigma\, i_{all}(t-1)) - (\Sigma\, i_{all}(t)) \geq 60$ then r = 1
else if $(\Sigma\, i_{front}(t) > 3000)$ or $(\Sigma\, i_{back}(t) > 2000)$ then r = -1
else r = 0

Threshold values like (2000, 3000, 60) have been determined after extensive experimentation. As one can see after reading this heuristics, backward moving in front of an obstacle is a correct policy. However, this behaviour is not the one we are looking for, so we decide to forbid backward moving mechanically. This is easier than modifying the heuristics.

Learning

We were able to point out that the quality of the learning was enhanced by a gradual increase in the complexity of the encountered situation. For example, at the beginning the environment contains only walls. Then, after the walls are correctly avoided, cylindrical obstacles are added. Their sensor images are more complex. At the end, almost everything can be used. It is important not to stop completely the random signal on the selection of actions. This variability in the selection allows dead-end situations to be dealt with.

Results

After 3000 learning steps, Khepera exhibits a behaviour which appears to solve the problem. However, an objective qualification of the behaviour is difficult. According to which criterion should we measure the performance? The learned behaviour can be interpreted from the network weights, or from a more direct indicator of the task, or from the measurement of the performance in relation to the heuristics. Today, there is no answer to this problem, so we will present all these measures.

Interpreting the learned behaviour from the network weights

The analysis of the behaviour from the network weights can be envisaged only for small networks (with only one layer of weights). In our case, the results remind us of Braitenberg's experiments [3]. In Fig. 3, we present the weights after learning. They are dependent on the initial conditions, but in most of the experiments the resulting behaviour is coherent. The diversity of the values is the expression of the

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

heterogeneity of the sensors. A later analysis indicates that there is a 50% performance variation between the sensors 2 and 3.

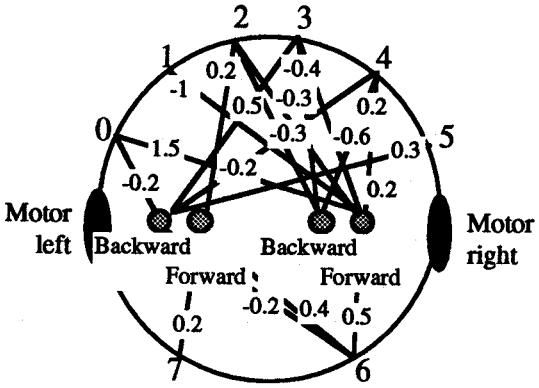

Fig. 3 Network weights after the learning of an obstacle avoidance behaviour. Only absolute weight values superior to 0.2 have been reported.

Indicator of the task

The measure of the moving distance per unit of time during the learning shows that at the beginning, movements are small and take random directions. At the end of the learning, movements are longer and there is no more "hesitation" about how to deal with an obstacle.

Measure of the performances in relation to the heuristics

Fig. 4 displays the number of positive and negative reinforcements during learning. This measure appears to be unbiased, but it only measures the algorithm performance in minimising errors. The link with the task to be solved is tenuous. It depends on the quality of the heuristics. Our experience in this domain shows that writing heuristics is difficult and it needs to be validated using real experiments.

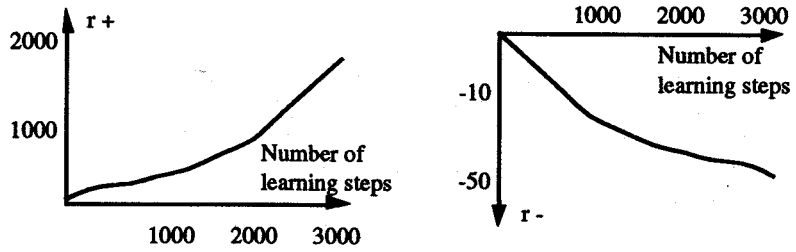

Fig. 4 Positive and negative reinforcements during learning. In particular, we can see the comparatively small number of negative reinforcements.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 153-159

6. Conclusion

In this paper, we describe a way to deal with negative reinforcement in the case of neural network implementation. The available actions are decomposed until elementary actions of agonist-antagonist type are reached. Each elementary action is coded by an output neuron and the output values are interpreted as certainty values of the network in its action proposals. In this way, weight modifications depend on the certainty of the network. Experiments in learning an obstacle avoidance behaviour with the miniature robot Khepera show how to find the right policy in a situations-actions space of size (10^{24} * 400). These experiments carried out in the real world allow real problem solving applications to be envisaged by neural reinforcement learning. This work is currently limited to immediate reinforcement learning but we think that these principles will be useful for delayed reinforcement learning.

Acknowledgement

The experiments were performed during a two-months positions as a guest professor in the Neural Networks Postgrade Course at the LAMI-EPFL (Switzerland). We thank all the K-Team members for their interest in this research and the use of one of the first Khepera robots.

References

1. Ackley D. & Littman M., "Interactions Between Learning and Evolution," Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X, C. G. Langton & Co Eds. Addison-Wesley, 487-509, 1991.
2. Barto A. G. & Anandan P., "Pattern Recognizing Stochastic Learning Automata," IEEE Transactions on Systems, Man and Cybernetics, SMC-15 : 360-375, 1985.
3. Braitenberg V., "Vehicles: Experiments in Synthetic Psychology," MIT Press, 1986.
4. Hertz J., Krogh A. & Palmer R. G. "Introduction to the Theory of Neural Computation," SFI Studies in the Sciences of Complexity, Addison-Wesley, Redwood City, 1991.
5. Millan J. & Torras C., "A Reinforcement Connectionist Approach to Robot Path Finding in Non-Maze-Like Environments," Machine Learning 8, n° 3/4, 363-395, 1992
6. Mondada F., Franzi E. & Ienne P., "Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms," Third International Symposium on Experimental Robotics, Kyoto, Japan, October 1993.
7. Sehad S. & Touzet C., "Reinforcement Learning and Neural Reinforcement Learning," this volume (ESANN94).
8. Touzet C. & Giambiasi, "Application of Connectionist Models to Fuzzy Inference Systems," in Parallelization in Inference Systems, Lecture Notes in Artificial Intelligence 590, B. Fronhöfer & G. Wrightson Eds., Springer Verlag, 1992.
9. Touzet C., "Neural Reinforcement Learning: Advantages and Limitations," (in French) 2nd European Congress on System Science, Prague, 1993.