REINFORCEMENT LEARNING SYSTEM IN SQUEAK SMALLTALK

Ing. Lukáš GRULICH, Doctoral Degree Programme (1)
Dept. of Intelligent Systems, FIT, BUT
E-mail: grulich@fit.vutbr.cz

Supervised by: Dr. Zdeňka Rábová

ABSTRACT

In this paper, we describe a machine learning system implemented in Squeak Smalltalk. This system is developed on the base of so-called reinforcement learning. In the first chapter there are outlined main features of the reinforcement learning, next chapters describe an agent created using this method. Finally, we present demonstration example and results of learning.

1 INTRODUCTION

The machine learning is often by an object of research in the artificial intelligence area. The ability of learning is in principle main manifestation of intelligence. We mean that does not exist more true sign of intelligence than ability of somebody to improve himself in something throught own experiences? There are many types of machine learning that are based on diverse paradigms of artificial intelligence. The proposed system is a practicle application of the reinforcement learning and it is implemented in Squeak Smalltalk.

2 REINFORCEMENT LEARNING

The rainforcement learning is a form of machine learning. According to [1], this belongs to behavioral paradigma of artificial intelligence (it is important for this paradigma to imitate outer behaviour instead of inner structure of intelligent entity). Sutton and Bart wrote (in [5]) that the rainforcement learning is "learning what to do – how to map situations to actions – so as to maximize a numerical reward signal". The characteristic properties of this form of machine learning are both **learning from trials** and **delayed rewards**. Next important feature is that the agent does not know the true goal of its working – an agent is interested only in maximalization of his reward. This approach may be advantage – this agent is very versatile. The same agent should be able to take controll of

many diverse enviroments and finding solution of many tasks. Generally, the rainforcement learning systems work on the basis of sequence of optimal actions (so actions which seems to be optimal for an agent). And how wrote Sutton and Bart (in [5]), "... actions followed by large rewards should be made more likely to recur, whereas actions followed by small rewards should be made less likely to recur ...".

3 MAIN STRUCTURE OF DESIGNED SYSTEM

Jorit Ijpenberg created and publicated in [2] the conception of a versatile self-learning system that is based on reinforcement learning. The algorithm of solution finding is realized by sequence of trials, evaluating their results, and upgrading the knowledge base. I assumed this conception with some modifications.

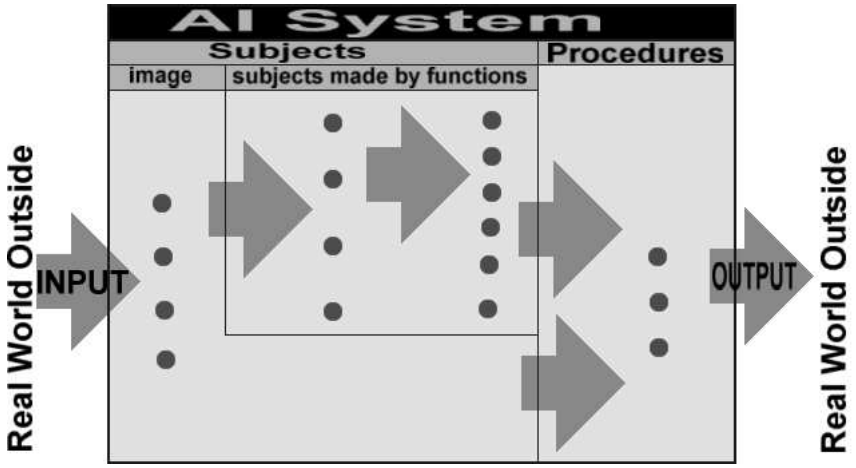

Figure 1: The abstract model of the system

3.1 DATA AND PROCEDURAL ELEMENTS OF THE SYSTEM

The system contains a finite number of **subjects** (variables) which have various types (frequently integer, or boolean). Subjects save information about elements of controlled system (enviroment). These particular subjects we can transform to another ones by **procedures** (processes). The procedures also may apply the actions, i.e. direct controll of some enviroment-features. On the figure 1, there is outlined relations between procedures (functions) and subjects (variables), and the model of data flow among subjects. Outerworld's (enviroment) data are saved in image-subjects. Others subjects that are used by the system are set by procedural transformation of one or more image-subjects. There is also reinforcement-subject (special case of image-subject). It contains the actual value of reward or penalty. This value can be true or false (it means good or bad controlling), eventually any integer value that say how much effective is controling. The process of learning is based on the maximalizaton of reward (value of reinforcement-subject).

Action	SubjectA (boolean)	SubjectB (boolean)
ProcedureX	Value Activation	Value Activation
ProcedureY	Value Activation	Value Activation

Table 1: The structure of inter-object table (simplified)

3.2 DATA REPRESENTATION

All the information (so whole knowledge base) are saved in the **inter-object table** (simplified version is on the table 1). In this table, there are relations between **control subjects** (collumns of the table) and **procedures** (rows of the table). Table cells contain several data items. The most important items are these two:

- *Activation* – This value is updated when the procedure is activated, and in every cycle of learning is multiplicated by learning factor. This is, in principle, weighting of actions – recently activated procedures have greates weight (and sense) than actions, that was activated earlier. The learning factor determines, how much is the system capable to learn from delayed response.
- *Value* – The prediction of reinforcement value. This number is computed by Activation value and reinforcement subject values within a trial. Value of this item is also important for selecting optimal action.

The process of learning is follows (simplified algorithm):

1. Watch the enviroment variables and update corresponding image-subjects.
2. Update other subjects by their update-code.
3. Select the optimal action and run it. The "optimal action" is (for the agent) an action (procedure) that has the highest value of the "Value" field in the inter-object table for actual control-subject configuration.
4. Update the table (items Activation and Value).
5. If the learning has not finished, go to 1 (learning is finished when the trial has been finished – by success or failure).

4 IMPLEMENTATION OF THE SYSTEM

I have implemented the learning-system (agent) that is described above in Squeak Smalltalk enviroment. The main parts of implementation are:

1. Implementation of subjects

   MLSubject is the main class of subject's implementation. It contains three data fields: the subject's name, actual value and block (variable of Squeak's block type) which is an instrument for updating of subject value.

   Next class is **MLImageSubject** (derived from **MLSubject**). It contains also an item called **updater** that serves to get a value of some enviroment-property.

2. Implementation of the procedures

   The procedures (actions) have been implemented by the **MLProcedure** class and represent any action (inner – in agent, or outer – in enviroment). **Name** of action, **evaluation** of procedure and **block** that implements action are important fileds of this class.

3. The main part – Agent

   The kernel of system – the agent – is implemented by class **MLAgent**. This class creates and manages the inter-object table, and implements an interface between inner structure (subjects and procedures) and user that use this system. Via this class an user creates the inter-object table, sets the parametrs of learning, etc.

5 DEMO AND RESULTS

We choose a simple problem for demonstration – searching the way in a labyrinth. Here is a (simplified) code of this demo:

1. Creating and initializing of both systems – agent and enviroment (labyrinth)

```
system := controledSystem new.
system initialize.
agent := controledSystem new.
agent enviroment: system.
```

2. Setting a link to the reinforcement value (by enviroment method "getRF")

```
agent addRewardSubject: [:enviroment | enviroment getRF.].
```

3. Adding all required subjects (in this case the subject is "subjectX", its value is acquired by enviroment-method called "getX"

```
agent addNamedImageSubject: 'subjectX' source: [:enviroment |
    enviroment getX.
].
```

4. Adding all required procedures (action named "left", implemented by enviroment method "goLeft")

```
agent addProcedure: 'left' action: [:val :ag |
(ag enviroment) goLeft: 1.].
```

5. Calling the **learn**-method to start the learning (200 trials)

```
agent learn: 200.
```

Results of labyrinth goal searching are shown in table 2. How we should see, the learning-factor is very importat for a successful work – for many values of it, the agent converges very slowly.

	Number of trials		
Faktor učení	100	200	300
0,99	0 (0 %)	0 (0 %)	0 (0 %)
0,95	3 (3 %)	11 (6 %)	22 (7 %)
0,9	10 (10 %)	28 (14 %)	63 (21 %)
0,8	18 (18 %)	69 (35 %)	192 (64 %)

Table 2: The results of demo (number of successful goal finding)

6 CONCLUSIONS

We summarized the most important features of created system (generic agent) in this article. Most of the theoretical and implementation details could not be described because of possible size of this paper. The main advantage of the developed agent is its versatility. In future, we plan more extensions of this concept: the knowledge representation must be improved, it is also possible to use some genetic algorithms to create/destroy subjects, etc. Anyway, this current implementation is just a prototype, but it allows us to do many interesting experiments.

REFERENCES

[1] Russel, N.: Artificial Intelligence: A Modern Approach, Prentice Hall, 2003.

[2] Ijpenberg, J.: A description of a new AI system with superior learning capabilities, 1999.

[Online: http://www.geocities.com/ainew.geo/index.html.]

[3] Vysoký, P.: Fuzzy řízení, ČVUT Praha, 1996. In czech.

[4] Reinforcement Learning Repository, University of Massachusetts.

[Online: http://www-anw.cs.umass.edu/rlr]

[5] Sutton, R. S., Barton, A. G.: Reinforcement Learning: An Introduction, Cambridge, 1998.

[Online: http://www-anw.cs.umass.edu/ rich/book/the-book.html]