ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

# Analysis of a Reinforcement Learning algorithm using Self-Organizing Maps

Vicente Buendía-Ramón, Emilio Soria-Olivas, José D. Martín-Guerrero, Pablo Escandell-Montero, José M. Martínez-Martínez

University of Valencia - Department of Electronic Engineering
CL. Dr. Moliner, 50, 46100 Burjassot, Valencia - Spain

**Abstract**. The scenario of this work is defined by the need of many Machine Learning algorithms to tune a number of parameters that define its behavior; the resulting performance can be evaluated with different indices. The relationship between parameters and performance may be neither linear nor straightforward. This work proposes a qualitative approach to the afore-mentioned relationship by using Self-Organizing Maps due to their visual information processing. The approach is evaluated in the framework of Reinforcement Learning algorithms.

## 1 Introduction

One of the most challenging issues in Machine Learning is the relationship between the parameters of the models (e.g., the learning rate in a neural network) and the performance measures of those models (e.g., sensitivity/specificity in a classification problem). There are many previous works on this topic, being the conclusions of most of them only approximate due to the non-linearity and time variance of the models [1].

This work proposes a qualitative approach to obtain the above-mentioned relationship by using Self-Organizing Maps (SOMs). Such an approach enables a non-expert to design an experimental setup that allows finding a given performance. The proposed methodology can be applied to any Machine Learning model. In this work, a classical Reinforcement Learning (RL) model is used.

### 1.1 Self-Organizing Maps

SOM algorithm consists of a set of neurons usually arranged in a one or two-dimensional grid[2]. Although higher dimensional grids are also possible, they are hardly ever used because of their problematic visualization. Every neuron has a fixed position in the grid, and is represented by an n-dimensional weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_n]$, where $n$ is the dimensionality of the input space. A user pattern $\mathbf{x}$ is randomly chosen from the data set on each training step. Then, the neuron whose weight vector is the most similar to the user pattern is searched, being this neuron the so-called Best Matching Unit (BMU). The weight vectors of the BMU and its neighborhood are updated as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \alpha(t)h(t)(\mathbf{x} - \mathbf{w}^t) \tag{1}$$

where $t$ stands for the iteration number, $\alpha(t)$ is the learning rate, and $h(t)$ the neighborhood kernel, whose center is located at the BMU. The neighborhood kernel determines which neurons around the BMU are updated, and how this update affects each neuron.

One of the SOM applications is the so-called Visual Data Mining that basically consists of extracting knowledge from those models. A straightforward way to do this is by clustering the map using any clustering algorithm (e.g., K-Means [1]) on the distances between synaptic weights of the neurons. Afterwards, the different components of the input vectors to the SOM can be simultaneously analyzed thus obtaining relationships among the different features that are used as inputs to the model.

### 1.2 Reinforcement Learning

RL algorithms are based on the interaction between an agent and its environment as shown in Fig. 1. The agent is the learner which interacts with the environment, making decisions according to observations made from it. The environment is every external condition that cannot be modified by the agent [3].

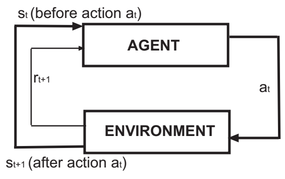

Fig. 1: Characterization of the Reinforcement Learning model; $a_t$ is the action taken by the agent at time $t$, $s_t$ is the state of the environment at time $t$, and $r_{t+1}$ denotes the reward at time $t+1$.

The long-term reward ($R_t$) is the sum of all the immediate rewards throughout a complete decision process. It is the objective function that the agent is interested in maximizing by taking the right actions [3]. Its mathematical expression is:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

where $\gamma$ is called the *discount-rate* parameter, which ranges between 0 and 1. This factor determines the present value of future rewards.

The goal is to maximize $R_t$ by means of an optimal policy, which tells the agent the best action to take for an optimal performance. Therefore, an estimation of the expected $R_t$ as the result of an action $a_t$ from a state $s_t$ is required. This estimation is usually called *action-value function*:

$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a] \tag{3}$$

where $Q^{\pi}(s,a)$ is the expected $R_t$ starting at state $s$ and taking action $a$, following policy $\pi(s,a)$. Therefore, the optimal policy is given by the following expression [3]:

$$\pi^*(s,a) = \arg\max_{a \in A} Q^{\pi}(s,a) \tag{4}$$

where $A$ stands for the set of possible actions. Optimal policies thus share the same optimal action-value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$. An iterative and very popular procedure to determine Q(s,a) is Q-Learning [3]:

$$Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \alpha \left[ r_{t+1} + \gamma \max_{a \in A}(Q_t(s_{t+1},a)) - Q_t(s_t,a_t) \right] \tag{5}$$

As shown in expression (5), Q-Learning has two parameters to take into account: $\alpha$ is the learning rate that measures how fast the Q-values change, and $\gamma$ measures the relevance of future rewards. Moreover, there are two other parameters that play a relevant role: $\epsilon$ balances the trade-off between a conservative policy (taking the best known action) and the exploration of new actions (take new random actions in order to likely find an even better action eventually); $\lambda$ is related to "traces" that are also related with the relevance of future rewards.

## 2 Experimental Setup

### 2.1 Description of the problem

The problems analyzed in this paper were variants of a classical RL problem, the Gridworld, in which there are a start position and a goal position. The agent tries to find the goal position as soon as possible.

In particular, Gridworlds of size $10 \times 10$ were considered. Four different variants (Fig. 2) were taken into account. Gridworld 1 was the simplest one, with only a start cell and a goal cell. Gridworld 2 also had some walls (cells which could not be accessed) and mousetraps (cells with negative reward). Gridworld 3 had cells in which the agent could only go one way, and Gridworld 4 had cells in which some actions made the agent go to non adjacent cells.

The agent was guided towards the goal by means of the Watkins' TD algorithm Q($\lambda$)[3] that makes use of the parameters $\alpha$, $\lambda$, $\epsilon$ and $\gamma$. After initial tests to set the ranges of the different parameters, five values linearly spaced into its range were tested for each parameter, thus leading to 625 different combinations. Every parameter combination was run over 30 experiments of 300 episodes each. The criteria to evaluate learning performance was based in Kaelbling criteria [4]:

- *Convergence* episode: Number of the episode from which the standard deviation of the number of steps for episode falls to value ten times lower than the maximum standard deviation. It's desired a low convergence episode however the results will show it's difficult to obtain a good convergence and also good values in the other performance measures.

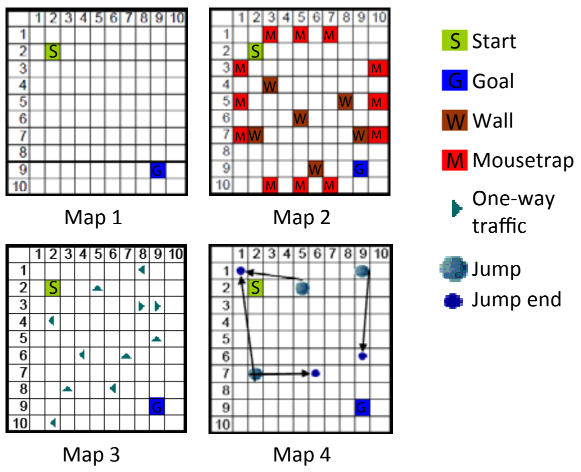

Fig. 2: Gridworld environments. Canonical example and three variants in which an agent has to learn the best behavior.

- *Consumption*: Number of steps/actions followed within an episode, starting from the convergence episode. Once the algorithm is stable, it is desirable a low number of steps to reach the goal.
- Standard deviation *(STD)*: Standard deviation of the number of steps an episode takes, starting from the convergence episode. A stability measure.
- Percentage of *Deficient* episodes: Percentage of episodes that do not reach convergence before a certain threshold that is set in order to allow the calculation of the other criteria.
- Percentage of episodes in which the optimal path *(Optimum)* is reached.

Therefore, the aim was to relate four parameters with five performance measures. SOM was used to show the relationships among them, as it is explained in detail in Section 2.2.

## 2.2 SOM visualization of results

Each Gridworld test data (all input parameter value combinations and their resulting performance measures) was separately used to train SOMs with different initializations and neighbor functions. Those maps showing the lowest Kaski and Lagus index[2] were finally chosen since the objective was to find maps that carried out an appropriate representation of the data and were easily understandable.

The resulting SOMs for Gridworlds 1, 2 and 3 are presented in Figs. 3, 4 and 5, respectively. SOM for Gridworld 4 is not presented since results were very

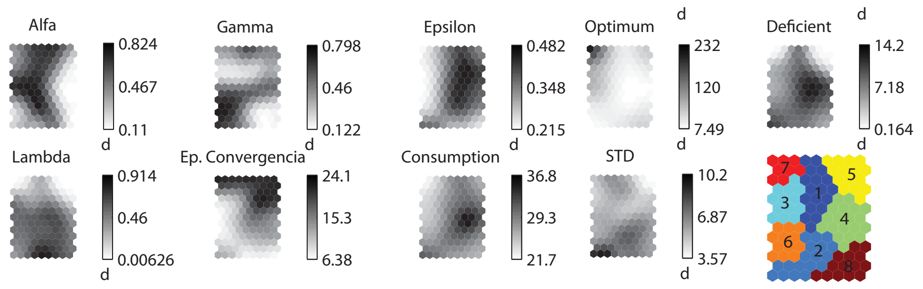

Fig. 3: SOM for Gridworld 1. The component planes for the different features (model parameters and performance measures) as well as a K-Means clustering of the SOM are shown.

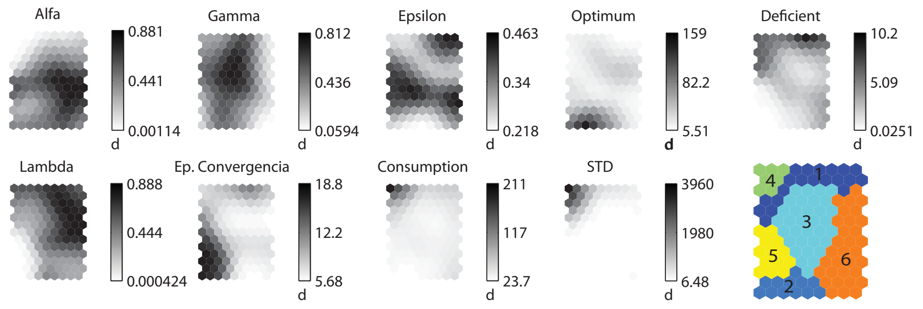

Fig. 4: SOM for Gridworld 2. The component planes for the different features (model parameters and performance measures) as well as a K-Means clustering of the SOM are shown.

similar to Gridworld 1. A K-Means clustering was carried out over the SOM in order to emphasize the relevant areas of the map, and hence, make an easier interpretation of the results.

Figs. 3, 4 and 5 show up relevant relationships among the different features. Since Gridworlds 1 and 4 presented a similar behavior, only Gridworld 1 is shown, being especially remarkable the differences with Gridworld 2; there are also some differences with Gridworld 3. It should be emphasized the similarity among the Component Planes of *Consumption*, *STD* and *Deficient* episodes in all the cases; therefore, only one of the indices would be necessary to explain the others. Component Planes also show an inverse relationship between $\lambda$ and *Convergence*. There is also an inverse relationship between $\alpha$ and *Deficient* (and hence, also *Comsumption* and *STD*). This kind of analysis can be done at the cluster level, thus extraction a set of rules about the relationship among the different features. For example, fig. 3 shows how cluster 7 is related to a good parameter selection as it means low consumption, medium-low STD, low deficient and high optimum percentages of episodes.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

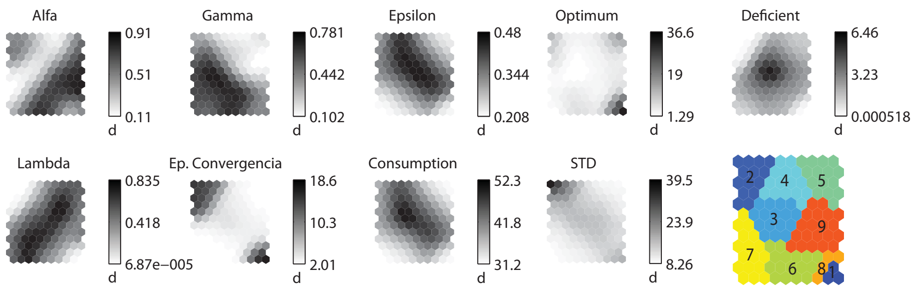

Fig. 5: SOM for Gridworld 3. The component planes for the different features (model parameters and performance measures) as well as a K-Means clustering of the SOM are shown.

## 3 Conclusions and further work

This work has presented the SOM as a tool for visual data mining. It is used in order to determine the rules that relate the different parameters of a Reinforcement Learning model to its corresponding performance measures. The methodology can be extended to other Machine Learning models, thus helping non-experts to optimize the paramaters of a certain model.

## 4 Acknowledgments

This work has been partially supported by the Spanish Ministry of Education and Science under projects with reference numbers TIN2007-61006 and CSD2007-00018.

## References

[1] E. Alpaydin. *Introduction to Machine Learning.* The MIT Press, 2004.

[2] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Heidelberg, Germany, 2001.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[4] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.