Reinforcement Learning of a Simple Control Task Using the Spike Response Model

Murilo Saraiva de Queiroz, *murilo@vettalabs.com*
Vetta Laboratories
Antônio de Pádua Braga, *apbraga@cpdee.ufmg.br*
Roberto Coelho de Berredo, *rberredo@uai.com.br*
Dept. of Electronics Engineering, Federal University of Minas Gerais, MG, Brazil

Abstract

In this work, we propose a variation of a direct reinforcement learning algorithm, suitable for usage with spiking neurons based on the Spike Response Model (SRM). The SRM is a biologically-inspired, flexible model of spiking neuron based on kernel functions that describe the effect of spike reception and emission on the membrane potential of the neuron. In our experiments, the spikes emitted by a SRM neuron are used as input signals in a simple control task. The reinforcement signal obtained from the environment is used by the direct reinforcement learning algorithm, that modifies the synaptic weights of the neuron, adjusting the spiking firing times in order to obtain a better performance at the given problem. The obtained results are comparable to those from classic methods based on value function approximation and temporal difference, for simple control tasks.

1. Introduction

Many of the traditional artificial neural networks are based on simple neuron models known as rate models [1]. These models are extremely simple, bearing little or no resemblance to biological neurons. One of the most important features of a real neuron is that its behavior is not only based on the output potential, as modeled in classic artificial neural networks, but also on the precise spike timing: this capability makes it possible to achieve impressive sensorial acuity based on a very small number of action potentials, in intervals of the order of tens of *nanoseconds* in some species [2] [3]. Understanding spiking neuron models is extremely important to neuroscience, and can be quite interesting from the point of view of artificial neural network applications, since in nature the dynamic behavior of spiking neurons is fundamental to solve complex tasks in noisy environments [4].

Recently, applications based on spiking neurons have been attaining interest (e.g. [5] [6]), in [7], Hebbian-based learning is used in a variation of the leaky-integrator spiking neuron model for motion and velocity detection systems. In [8] and [9], the association between the stochastic nature of the spike firing times and (Hebbian) learning is considered. The latter work uses a stochastic model of neuron, which fires spikes with Poisson statistics, and uses the correlation of the reward signal with firing times to stochastically estimate the gradient for a local search algorithm, using an approach similar to the one presented here, without the control-related aspects.

We base our work in a powerful formal spiking neuron model is the *Spike Response Model*, or SRM [10] [1]. The SRM can be understood as a generalization of the integrate-and-fire model; mainly characterized by its parameters dependence on the time of the last output spike, instead of being a function only of the membrane potential as in more complex models. In the SRM model explicit functions model the action potentials, the reset/refractory periods, and the neuron response to input spikes. We can see these functions as *response kernels*; which was the inspiration for the name of the model. Our main motivation for choosing this model is our interest in studying how reinforcement learning can be done using low-level models closer to biological neurons.

Classic reinforcement learning methods are based on the estimation of a value function, which supposedly provides the best action to be taken in a given spate. Direct reinforcement learning, however, estimates the gradient direction that improves the performance of the used policy. We use this algorithm to modify the synapses of a SRM neuron, modifying the delays applied to the input spikes. The timing of the output spikes emitted determines the actions to be taken in the control task.

The results of our experiments show that the overall qualitative performance of the proposed direct reinforcement learning algorithm based on the Spike Response Model is comparable to the classic algorithms, but with a higher computational cost. We believe that this difference comes from the costs of simulating the SRM neurons with their spike trains and also from the biological nature of

the model, overheads not present in standard reinforcement learning algorithms like Q-Learning and SARSA.

2. Understanding Spiking Neurons

The electrical output of neurons is usually characterized by the presence of stereotyped action potential waveforms known as *spikes*. Spikes are clearly defined events generated in response to incoming stimuli arriving through neuron dendrites. The action potentials travel through cells' axon, and their firing times is practically the only information transmitted, since the waveforms of the spikes are practically identical. An important observation is that this neural code is not always fixed: for instance, in motor neurons submitted to a continuous stimulus, the spiking rate diminishes with time, demonstrating what is called neural adaptation [11].

In 1999, Bartlett and Baxter [12] proposed an interesting model of synaptic plasticity based on correlation (and thus Hebbian in essence). Their idea is to apply a *reinforcement learning* [13] algorithm to modify the synaptic efficiency with the explicit objective of improving performance on a given problem. Note that the objective now is not to mimic cellular mechanisms, but to use spiking neurons in practical applications. Analyzed from this point of view, their results were stimulating: the application of *direct reinforcement learning* [14] [15] in probabilistic spiking neurons is effective for simple tasks involving pattern classification and motor learning.

In [12], Bartlett and Baxter show how to apply direct reinforcement learning to modify synaptic weights of simple spiking neurons. Direct reinforcement learning is a technique that modifies the parameters of a random policy based on an estimate of the average long-term reward gradient. This approach does not manipulate value functions directly, avoiding some drawbacks of the techniques based on the generalized policy iteration.

This approach was used by them to solve simple pattern classification and motor learning problems. The simple spiking neuron (called here SSN) used is a variation of the MCP neuron [16], which was modified to produce action potentials stochastically. Like in many applications based on the MCP neuron, the input of the SSN at time t is a vector u_{t-1} of the binary values representing the presence or absence of an input spike at the current time step.

The potential of the SSN is given by

$$v_t = \sum_j w_j u_{t-1}^j \tag{1}$$

This expression is identical to the MCP neuron potential. In the MCP neuron, the output is graded and continuous valued, computed by the application of a transfer function to the neuron potential. Differently, the SSN uses a stochastic threshold model, where the neuron potential affects only the probability of firing a spike (i.e., taking the action $a_t = 1$) at time t:

$$\Pr(a_t = 1) = \sigma(v_t) = 1/(1+e^{-v_t}) \tag{2}$$

Consequently, the expression for the random, parameterized policy generated by the SSN is [12]

$$\pi_t(u_t, a_t, w_j) = \begin{cases} \sigma(v_t) & \text{if } a_t = 1 \text{ or} \\ 1-\sigma(v_t) & \text{if } a_t = 0. \end{cases} \tag{3}$$

The action $a_t = 1$ is represented by firing a spike at time t and, correspondingly, the absence of a spike represents the action $a_t = 0$, which has probability $1-\sigma(v_t)$.

In order to apply direct reinforcement learning, we need to compute ([17])

$$\frac{\nabla \pi_t(u_t, a_t, w)}{\pi_t(u_t, a_t, w)} \tag{4}$$

In the SSN case we have

$$\frac{\nabla \pi_t(u_t, a_t, w_j)}{\pi_t(u_t, a_t, w_j)} = \frac{\frac{\partial}{\partial w_j}\pi_t}{\pi_t} = \begin{cases} \frac{\sigma'(v_t)u_{t-1}^j}{\sigma(v_t)} & \text{if } a_t = 1, \\ \frac{-\sigma'(v_t)u_{t-1}^j}{1-\sigma(v_t)} & \text{if } a_t = 0 \end{cases} \tag{5}$$

$$= (a_t - \sigma(v_t))u_{t-1}^j, \tag{6}$$

observing the negative sign of $-\sigma'(v_t)u_{t-1}^j$, since that it is necessary to *reduce* the probability of a spike if we want to reinforce the action $a_t = 0$, and that

$$\sigma'(v_t) = \sigma(v_t)(1-\sigma(v_t)) \tag{7}$$

and

$$\frac{\partial}{\partial w_j} v_t = u_{t-1}^j. \tag{8}$$

Therefore, the complete equations for direct reinforcement learning for the simple spiking neuron are ([17])

$$w_{j,t+1} = w_{j,t} + \gamma r_{t+1} z_{j,t+1} \tag{9}$$

$$z_{j,t+1} = \beta z_{j,t} + (a_t - \sigma(v_t))u_{t-1}^j \tag{10}$$

Where r is the reinforcement signal from the environment (using the standard convention of positive values for desirable policies and negative values for indicating punishment). The parameter γ, the learning rate, is a parameter between 0 and 1 that determines how much the estimated gradient affects the neuron weights at each step. Similarly, the discount factor β, also between 0 and 1, is related to the magnitude of the adjusts applied to the estimated gradient. The values used for these parameters, obtained from simple experiments, were $\gamma = 0.9$ and $\beta = 0.1$; they are not

critical to the performance in the selected problems, as far as we could observe. A detailed theoretical analysis of the discount factor β can be found at [17].

This update rule has several interesting characteristics. Its Hebbian component, $a_t u_{t-1}^j$, increases the synapse efficiency when an action a_t follows a given input u_{t-1}^j. When no output spike is produced, the component $-\sigma(v_t)u_{t-1}^j$ reduces the synapse strength, as expected [12].

3. Direct RL Using the SRM

The main contribution to the spiking neuron research is now discussed: a method applying direct reinforcement learning with a neuron model far more sophisticated than the MCP, the SRM_0, where the subscript zero indicates a simplified version of the full SRM model [10, 1]. The difference is that the exact form of η during the spike is replaced by an impulse $\delta(t-t_i^{(f)})$, as shown in figure 3, and the fact that the last firing time of the neuron does not affect the new postsynaptic potential. The subscript zero is intended to remind this of the latter "zero order" characteristic.

As mentioned above, the spiking neurons used with direct reinforcement learning in the original work were simple variations of the classic MCP neuron [16], with minor modifications to stochastically produce spikes based on the neuron potential. In contrast, the SRM_0 uses the information contained in the spike firing times in a way much closer to complex, biologically inspired models such the classic one proposed by Hodgkin and Huxley [18].

In this section we present the deduction of the equations required to update the synaptic weights of the spiking neuron, implementing reinforcement learning. In the next one, an algorithm using these results is proposed, and the obtained results are analyzed.

Our implementation used a fully connected, feedforward spiking neural network with connections composed of multiple delayed synapses. Each individual connection from one neuron to another is actually composed by several sub-synapses, each one with its own weight (synaptic efficiency) and specific transmission delay [19]. It has been demonstrated that multiple synapses are biologically plausible [20] and have a time dependent dynamic plasticity, which can be used to enhance the efficiency of a spiking neuron to perform computations [21, 22].

Therefore, a spike produced by a neuron i is independently transmitted over several sub-synapses to the neuron j. The different delays and weights of each sub-synapse changes how the input spike will affect the potential of neuron j. As an example, Figure 1 illustrates a single connection composed by three delayed sub-synapses.

The total potential of a neuron j, connected to a neuron i through a connection with m delayed sub-synapses, is then given by [23]

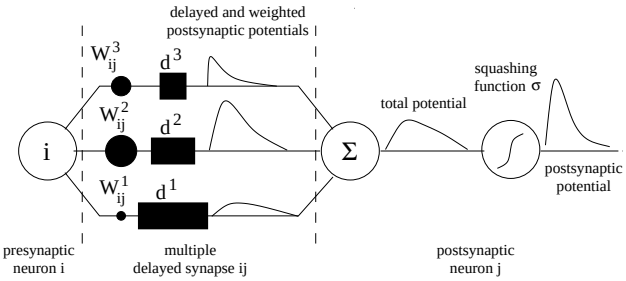

Figure 1. Single connection composed by multiple weighted and delayed synapses. Note how different weights and delays generate different forms of postsynaptic potentials. From [23] .

$$v_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k \varepsilon(t - t_i - d^k) \tag{11}$$

Here, w_{ij}^k and d^k are, respectively, the weight and the delay associated to the k-th synapse, Γ_j is the set of all the neurons i presynaptic to the neuron j and t_i is the time of the input spike i. The input to the network is represented by a set of firing times within a coding interval ΔT, where each neuron is required to fire at most once [23]

The shape of the postsynaptic potential is given by the ε-kernel, as shown in figure 2. In this work, we consider a strictly excitatory α-function, given by [23]

$$\varepsilon(t) = \frac{t}{\tau} e^{(1-\frac{t}{\tau})} \tag{12}$$

In order to apply direct reinforcement learning for training an SRM_0 neuron with multiple delayed synapses, the first step is to define the probability of firing a spike. Like in the implementation of the SSN, we use a squashing function quite similar to that defined by Equation 2, with a new parameter g with determines the *sigmoid gain*.

$$\Pr(a_t = 1) = \sigma(v_t) = 1/(1 + e^{-gv_t}) \tag{13}$$

A parameterized, random policy $\pi_t(u_t, a_t, w)$ based on the potential v_t of the SRM_0 neuron can be defined, identical to the policy of the simple spiking neuron, defined in Equation 3.

From Equations 14 and 15, applying the chain rule to derive the Direct Reinforcement Learning equations for the SRM0 neuron, we obtain Equation 16.

$$\frac{\partial \pi_t}{\partial w_{ij}^k} = \frac{\partial \pi_t}{\partial v_j(t)} \frac{\partial v_j(t)}{\partial w_{ij}^k} \tag{14}$$

$$\frac{\partial v_j(t)}{\partial w_{ij}^k} = \frac{\partial(\sum_{i\in\Gamma_j}\sum_{k=1}^{m} w_{ij}^k \varepsilon(t-t_i-d^k)}{\partial w_{ij}^k} = \varepsilon(t-t_i-d^k) \tag{15}$$

$$\frac{\nabla\pi_t(u_t,a_t,w_{ij}^k)}{\pi_t(u_t,a_t,w_{ij}^k)} = \frac{\frac{\partial}{\partial w_{ij}^k}\pi_t}{\pi_t} = g(a_t-\sigma(v_t))\varepsilon(t-t_i-d^k) \tag{16}$$

Note the presence of the parameter g, the sigmoid gain, in Equation 16. It appears because, if σ is defined as in Equation 13, then $\sigma'(v_t) = g\sigma(v_t)(1-\sigma(v_t))$.

Finally, the equations for updating the weights of a SRM_0 neuron with multiple delayed synapses using direct reinforcement learning are

$$w_{j,t+1}^k = w_{j,t}^k + \gamma r_{t+1} z_{j,t+1} \tag{17}$$

$$z_{j,t+1} = \beta z_{j,t} + g(a_t-\sigma(v_t))\varepsilon(t-t_i-d^k) \tag{18}$$

Equation 18 should be compared to the equivalent expression for the simple spiking neuron case, defined by Equation 10. The ε-kernel, which determines the shape of the postsynaptic potential as a function of the input spikes firing times, has a fundamental role in the new rule, affecting the amount of change in each weight.

4. RL Applied to Spiking Neurons

Our main target is to find out how to use spiking neurons to perform useful computations, inspired by the success of classic artificial neural networks (ANN) and, obviously, by the interesting capabilities of real neurons. Once we have chosen a specific model of spiking neurons, the evident next step is to ask how to use artificial neurons based on this model to solve problems.

More than carefully engineer a fixed solution based on spiking neurons, what we need is, once more as we do in classic ANN, to find out suitable *learning rules*, thus making our system capable to adapt itself in order to achieve good performance. Many results show that variations of the well-known Hebbian learning paradigm can perform convincingly well when applied to spiking neuron models [24] [25] [26].

Figure 4 shows SRM-RL, our most important contribution: a direct reinforcement learning algorithm for spiking neurons based on the SRM_0 model, which iteratively implements the equations described in the previous section. This simplified version assumes only one spiking neuron; extending it for multiple output neurons is straight-forward.

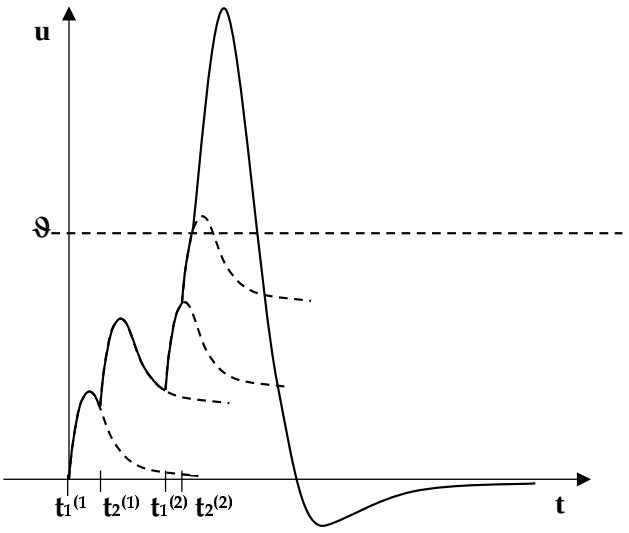

Figure 2. Evolution of the membrane potential. The threshold θ is crossed after the arrival of the post-synaptic potential produced by the fourth spike, and a new action potential is generated. Note the negative overshoot and slow recovery after spike emission, corresponding to the refractory period. Based on [1].

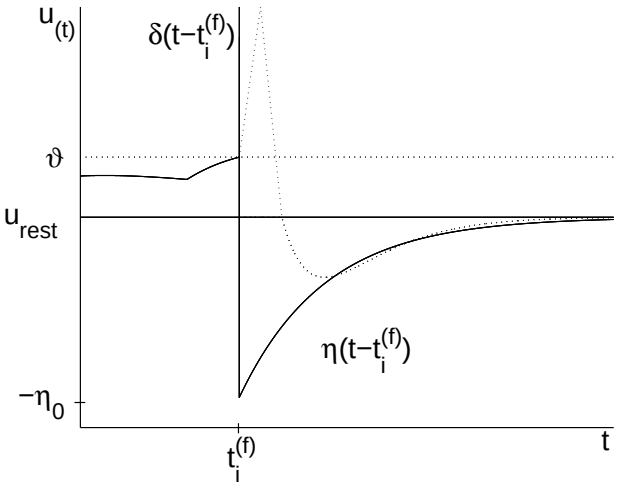

Figure 3. η-kernel of a SRM_0 neuron (thick line). The spike occurs when the potential u crosses the threshold θ. In the spike-afterpotential period the membrane potential is reduced to a negative level $-\eta_0$. After the negative overshoot there is a period of slow recuperation, due to refractoriness. In many models the exact form of η during the spike is replaced by an impulse $\delta(t-t_i^{(f)})$, as shown. Redrawn from [10].

for all sub-connections $k = 1..m$ of all inputs $i \in \Gamma$ **do**
  $z_i^k \leftarrow 0$
  $w_i^k \leftarrow$ small random value
end for
for all episodes at time t=0,1,... **do**
  for all input neurons $i \in \Gamma$ **do**
    $t_i \leftarrow$ time of the spike from input neuron i
  end for
  $v \leftarrow \sum_{i\in\Gamma}\sum_{k=1}^{m} w_i^k \varepsilon(t - t_i - d^k)$
  $a \leftarrow$ Action chosen according to $\Pr(a = 1) = \sigma(v)$
  Take action a, observe reward r
  for all sub-connections $k = 1..m$ of all inputs $i \in \Gamma$ **do**
    $z_i^k \leftarrow \beta z_i^k + g(a - \sigma(v))\varepsilon(t - t_i - d^k)$
    $w_i^k \leftarrow w_i^k + \gamma r z_i^k$
  end for
end for

Figure 4. SRM-RL algorithm for a single output neuron. Based on the algorithm for direct reinforcement learning using spiking neurons proposed in [12].

5. Experimental Results

For the tests we selected the *mountain car* problem, as stated in [13]:

> Consider the task of driving an underpowered car up a steep mountain road. The difficulty is that gravity is stronger than the car's engine, and even at full throttle the car cannot accelerate up the steep slope. The only solution is to first move away from the goal and up the opposite slope on the left. Then, by applying full throttle the car can build up enough inertia to carry it up the steep slope even though it is slowing down the whole way. This is a simple example of a continuous control task where things have to get worse in a sense (farther from the goal) before they can get better. Many control methodologies have great difficulties with tasks of this kind unless explicitly aided by a human designer.
>
> The reward in this problem is -1 on all time steps until the car moves past its goal position at the top of the mountain, which ends the episode. At time t, there are three possible actions a_t: full throttle forward ($+1$) and full throttle reverse (-1). The car moves according to a simplified physics. Its position and velocity are updated by
>
> $$x_{t+1} = bound[x_t + x'_{t+1}] \tag{19}$$
>
> $$x'_{t+1} = bound[x'_t + 0.001a_t + -0.0025cos(3x_t)] \tag{20}$$
>
> where the *bound* operation enforces
> $-1.2 \leq x_{t+1} \leq 0.5$
> and
> $-0.07 \leq x'_{t+1} \leq 0.07$.
>
> When reached the left bound, was reset to zero. When it reached the right bound, the goal was reached and the episode was terminated. Each episode started from a random position and velocity uniformly chosen from these ranges.

This environment was used for the sake of simplicity, since it only allows two actions and is computationally inexpensive. The state space of this problem consists of two continuous state variables: the velocity and the position of the car. For each dimension, nine random-mean, Gaussian receptive fields are defined, converting the inputs into 81 discrete states. The discrete state index is seen by the neuron as the delay of the input spike. A spike emission is in a given instant t is interpreted as the action -1 (full throttle reverse), and its absence, as the action 1.

In this framework, the performance of SRM-RL was compared with other available algorithms, namely Q-learning, Q(λ), Sarsa, Sarsa(λ) and the classic algorithm based on adaptive neuron-like elements (ANE) [27].

Each algorithm was executed one hundred times, with each run finishing after one hundred episodes (training epochs). Our performance indicator for this task is the average number of steps needed to reach the goal (the top of the mountain).

Figure 5 displays the obtained results. The results for Sarsa and Sarsa(λ) for this problem are almost identical to the results of, respectively, Q-learning and Q(λ), and both were omitted for the sake of simplicity. The parameters used for Sarsa and Sarsa(λ) were $\varepsilon = 0.0$ (probability of random action), $\alpha = 0.5$ (step size parameter), $\lambda = 0.9$ (trace-decay parameters) and $\gamma = 1$ (discount-rate parameters). The parameters and the implementation of these algorithms were the same used in [13].

As can be observed, the results obtained for this task are quite satisfactory. The performance of SRM-RL with $gain = 4$ is comparable to standard Q-learning. With higher gain, the average number of steps is much lower, being comparable to those obtained using algorithms based on temporal difference (Q(λ)) and custom value function approximation (ANE), and close to the optimal value.

It is interesting to notice the difference in the behavior of the tabular algorithms (Q-learning and Q(λ)) and the others. The variation in the number of steps is much lower in ANE and in SRM-RL, suggesting better convergence.

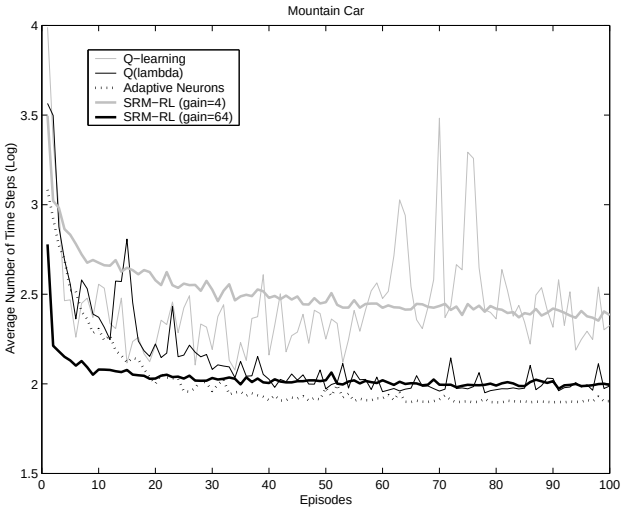

Figure 5. Performance of SRM-RL in the mountain car problem. The results are the logarithm of the average obtained over one hundred executions. SRM-RL was executed using two different values for the sigmoid gain parameter.

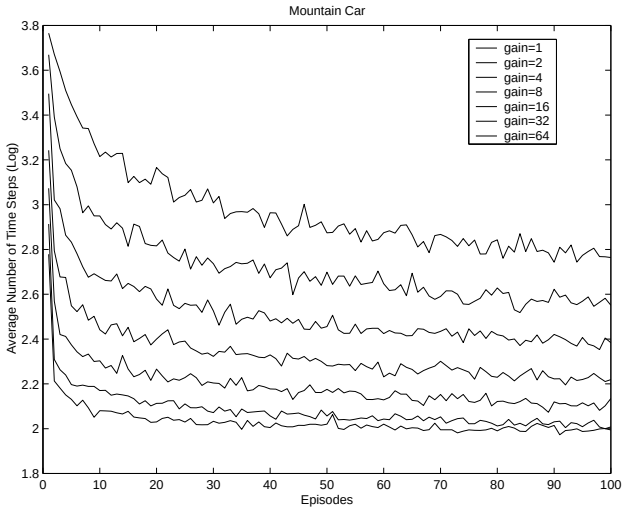

Figure 6. Performance of SRM-RL as a function of the gain, in the mountain car problem. The curves are show in the same order shown in the chart legend.

The sigmoid gain in the SRM-RL algorithm is directly associated to the exploration-exploitation dilemma. Lower values of gain implies that the steepness of $\sigma(v_t)$, the probability distribution function used to select actions (see Equation 13), will be small. This leads to more frequent selection of actions contrary to the agent's belief, favoring exploration.

In contrast, if very high values of gain are used, the function $\sigma(v_t)$ approximates the step function, and the agent always chooses the best action according to its knowledge, maximizing exploitation.

Figure 6 shows the results of several experiments in the mountain car environment, using different values of gain. Continuous exploration limits the performance of the algorithm, but gives better results in non-stationary problems. High values of gain, however, can be dangerous, since they increase the chance of being trapped in a local minimum. A trade-off solution for this problem is to slowly decay the frequency of exploration, allowing the agent to approximate optimal performance after some time.

6. Conclusion

The billions of real neurons in the brain are very complex dynamic entities capable of processing information encoded in stereotyped action potentials, informally named "spikes". These neurons rely not only on average potential values as described by the traditional models, but also on precise spike firing time information. Experimental evidence found by neuroscientists shows that neurons based on this paradigm achieve fast and highly reliable performance in several different domains [28, 4].

Our proposed algorithm uses direct reinforcement learning with a more sophisticated model of spiking neuron, SRM_0. The potential of SRM_0 neurons is computed using a function that determines the behavior in time of the received postsynaptic potentials (spikes), based on the firing times of the input spikes. We show also how to use direct reinforcement learning to modify the weights of the multiple delayed synapses that composes a connection between two spiking neurons in the network architecture used [23, 19].

The presented results show that this approach have performance comparable to classic algorithms based on temporal difference and value function approximation, for solving a simple control problem.

References

[1] W. Gerstner, W. M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

[2] J. A. Simmons, A view of the world through the bat's ear: The formation of acoustic images in echolocation, Cognition 33 (1989) 155–199.

[3] G. Rose, W. W. Heiligenberg, Temporal hiperacuity in the electric sense of fish, Nature 318 (1985) 178–180.

[4] R. d. R. v. S. F. Rieke, D. Warland, W. Bialek, Spikes: Exploring the neural code, The MIT Press, 1997.

[5] A. Delorme, S. J. Thorpe, Face identification using spike per neuron: resistance to image degradations, Neural Networks

14 (6–7) (2001) 795–803.

[6] R. Malaka, S. Buck, Solving nonlinear optimization problems using networks of spiking neurons, in: IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Vol. 6, 2000, pp. 486–491.

[7] C. C., G. Bugmann, T. G. Clarksona, A spiking neuron model: Applications and learning, Neural Networks 15 (2002) 891–908.

[8] H. S. Seung, Learning in spiking neural networks by reinforcement of stochastic synaptic transmission, Neuron 40 (2003) 1063–1073.

[9] X. Xie, H. S. Seung, Learning in neural networks by reinforcement of irregular spiking, PHYSICAL REVIEW E 69, 041909.

[10] W. Gerstner, The spike response model, mNN4 (September 1999).

[11] H. R. Wilson, Spikes Decisions and Actions – Dynamical Foundations of Neuroscience, 1st Edition, The Oxford University Press, Oxford, 1999.

[12] P. L. Bartlett, J. Baxter, Hebbian synaptic modifications in spiking neurons that learn, Tech. rep., Australian National University (November 1999).

[13] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.

[14] J. Baxter, P. L. Bartlett, Direct gradient-based reinforcement learning: I. gradient estimation algorithms, Tech. rep., Research School of Information Sciences and Engineering, Australian National University (July 1999).

[15] L. W. J. Baxter, P. L. Bartlett, Direct gradient-based reinforcement learning: Ii. gradient descent algorithms and experiments, Tech. rep., Research School of Information Sciences and Engineering, Australian National University (September 1999).

[16] W. S. McCulloch, W. A. Pitts, A logical calculus of the ideas imminent in nervous activity, Bull. Math. Byophys. 5 (1943) 115–133.

[17] J. Baxter, P. Bartlett, Direct gradient-based reinforcement learning, Tech. rep., Research School of Information Sciences and Engineering, Australian National University (July 1999).

[18] A. L. Hodgkin, A. F. Huxley, A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes, J. Physiol. (Lond.) 117 (1952) 500–544.

[19] T. Natschläger, B. Ruf, Spatial and temporal pattern analysis via spiking neurons, Network: Comput. in Neural Syst. 9 (1998) 319–332.

[20] E. Wolf, F.-Y. Zhao, A. Roberts, Non-linear summation of excitatory sinaptic inputs to small neurones: a case study in spinal motoneurones of the young *Xenopus* tadpole, Journal of Physiology 511.3 (1998) 871–886.

[21] W. Maass, A. M. Zador, Dynamic stochastic synapses as computational units, Neural Computation 11 (4) (1999) 903–917.

[22] T. Natschläger, W. Maass, A. Zador, Efficient temporal processing with biologically realistic dynamic synapses, Network: Computation in Neural Systems 12 (2001) 75–87.

[23] S. M. Bohte, H. L. Poutr, J. N. Kok, Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks, IEEE Transactions on Neural Networks 13 (2).

[24] W. Gerstner, W. M. Kistler, Mathematical formulations of Hebbian learning, Biological Cybernetics 87 (2002) 404–415.

[25] M. M. P. Häfliger, L. Watts, A spike based learning neuron in analog vlsi, in: M. I. J. M. C Mozer, T. Petsche (Eds.), Advances in Neural Information Processing Systems, The MIT Press, 1997, pp. 692–698.

[26] D. D. L. J. Rubin, H. Sompolinsky, Equilibrium properties of temporally asymmetric hebbian plasticity, Phys. Rev. Lett. 86 (2001) 364–367.

[27] R. S. S. Andrew G. Barto, C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man, and Cybernetics SMC-13 (1983) 834–846.

[28] W. Maass, Computing with spikes, Tech. rep., Institut fuer Grundlagen der Informationsverarbeitung - Technische Universitaet Graz (2002).