Improving Recurrent CSVM Performance for Robot Navigation on Discrete Labyrinths

Nancy Arana-Daniel[1], Carlos López-Franco[1], and Eduardo Bayro-Corrochano[2]

[1] Electronics and Computer Science Division, Exact Sciences and Engineering Campus, CUCEI, Universidad de Guadalajara, Av. Revolucion 1500, Col. Olímpica, C.P. 44430, Guadalajara, Jalisco, México
[2] Cinvestav del IPN, Department of Electrical Engineering and Computer Science, Zapopan, Jalisco, México
{nancy.arana,carlos.lopez}@cucei.udg.mx,edb@gdl.cinvestav.mx

Abstract. This paper presents an improvement of a recurrent learning system called LSTM-CSVM (introduced in [1]) for robot navigation applications, this approach is used to deal with some of the main issues addressed in the research area: the problem of navigation on large domains, partial observability, limited number of learning experiences and slow learning of optimal policies. The advantages of this new version of LSTM-CSVM system, are that it can find optimal paths through mazes and it reduces the number of generations to evolve the system to find the optimal navigation policy, therefore either the training time of the system is reduced. This is done by adding an heuristic methodoly to find the optimal path from start state to the goal state.can contain information about the whole environment or just partial information about it.

Keywords: Robot navigation, LSTM-CSVM, optimal path, heuristic.

1 Introduction

This paper presents an improvement of the recurrent learning system LSTM-CSVM [1] for robot navigation applications. The design of LSTM-CSVM system is based on Evoke algorithm [evoke], both of them are constructed by two cascaded modules: (1) a recurrent neural network Long-Short Term Memory (LSTM) that receives the sequence of external inputs, and (2) a parametric function that maps the internal activations of the first module to a set of outputs. The second module is different for these systems: Evoke algorithm uses a real Support Vector Machine to produce precise final outputs, meanwhile LSTM-CSVM uses a Clifford generalization of SVM algorithm known as Clifford Support Multivector Machine (CSVM) [3] to do the same job, but improving the real-computation time. So, the main advantage of using CSVM approach as second module is that real-SVM algorithm is transformed into a MIMO SVM without adding complexity. This is done by embedding the optimization problem into a geometric algebra framework, which allows us to represent the entries to the CSVM, the optimization variables and the outputs as multivectors and, in this way we can

E. Bayro-Corrochano and J.-O. Eklundh (Eds.): CIARP 2009, LNCS 5856, pp. 834–842, 2009.

represent multiple classes according to the dimension of the geometric algebra in which we work.

In previous work [1] it is proved that LSTM-CSVM approach gets lower training and testing errors by showing different experiment results on applications like time series and comparing them with the results obtained from algorithms such as Evolino, Echo State Networks and LSTM. Furthermore, it is shown the performance of LSTM-CSVM on tasks like robot navigation through a maze using reinforcement learning and neuroevolution approaches to solve this tasks.

The results presented in this paper provide evidence about the improvement of the performance of LSTM-CSVM for robot navigation, in particular for maze navigation by adding an heuristic approach which allows us to find the lowest cost path from the robots start state to the goal state. Other advantage of this new version is that the number of generations to evolve the system is reduced, therefore either the training time of the system is reduced.

2 Long Short Term Memory (LSTM)

Learning to extract and represent information from a long time ago has proven difficult, both for model-based and for model-free approaches. The difficulty lies in discovering the correlation between a piece of information and the moment at which this information becomes relevant at a later time, given the distracting observations and actions between them [4].

LSTM is a recurrent neural network architecture, originally designed for supervised timeseries learning [5]. It is based on an analysis of the problems that conventional recurrent neural networks and their corresponding learning algorithms, e.g. Elman networks with standard one step backpropagation, Elman networks with backpropagation through time (BPTT) and real-time recurrent learning (RTRL), have when learning timeseries with long-term dependencies. These problems boil down to the problem that errors propagated back in time tend to either vanish or blow up (see [4]). LSTM's solution to this problem is to enforce constant error ow in a number of specialized units, called Constant Error Carrousels (CECs). This turns out to correspond to the CECs having linear activation functions which do not decay over time. In order to prevent the CECs from filling up with useless information from the timeseries, access to them is regulated using other specialized, multiplicative units, called input gates. Like the CECs, the input gates receive input from the timeseries and the other units in the network, and they learn to open and close access to the CECs at appropriate moments. Access from the activations of the CECs to the output units (and possibly other units) of the network is regulated using multiplicative output gates. Similar to the input gates, the output gates learn when the time is right to send the information stored in the CECs to the output side of the network. The forget gates learn to reset the activation of the CECs (in a possibly gradual way) when the information stored in the CECs is no longer useful. The combination of a CEC with its associated input, output, and forget gate is called a memory cell.

3 Clifford Support Multivector Machine (CSVM)

For the case of the Clifford SVM for classification we represent the data set in a certain Clifford Algebra [6] $\mathcal{G}_n$ where $n = p + q + r$, where any multivector base squares to 0, 1 or -1 depending if they belong to p, q, or r multivector bases respectively. We consider the general case of an input comprising D multivectors, and one multivector output, i.e. each ith-vector has D multivector entries $\boldsymbol{x_i} = [\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}}, ..., \boldsymbol{x_{iD}}]^{\boldsymbol{T}}$, where $\boldsymbol{x_{ij}} \in \boldsymbol{\mathcal{G}_n}$ and D is its dimension. Thus the ith-vector dimension is D$\times 2^n$, then each data ith-vector $\boldsymbol{x_i} \in \boldsymbol{\mathcal{G}_n^D}$. And each of ith-vectors will be associated with one output of the 2^n posibilities given by the multivector output $y_i = y_{i_s} + y_{i_{\sigma_1}} + y_{i_{\sigma_2}} + ... + y_{iI} \in \{\pm 1 \pm \sigma_1 \pm \sigma_2 \ldots \pm I\}$ where the first subindex s stands for scalar part. The dual expression of the optimization problem to solve by the CSVM[1] is: ,

$$\begin{aligned} max \quad & \boldsymbol{a^T 1} - \frac{1}{2}\boldsymbol{a^T H a} \\ subject\,to \quad & \\ & 0 \leq (\alpha_s)_j \leq C,\ 0 \leq (\alpha_{\sigma_1})_j \leq C, ..., \\ & 0 \leq (\alpha_{\sigma_1\sigma_2})_j \leq C, ..., 0 \leq (\alpha_I)_j \leq C \\ & for \quad j = 1, ..., l, \end{aligned} \tag{1}$$

where H represents the Gramm matrix and it is defined by the Clifford product of the input vectors x in terms of the matrices of t-grade $\boldsymbol{H_t} = \langle \boldsymbol{x^\dagger x} \rangle_{\boldsymbol{t}}$, and $\boldsymbol{a}$, has the dimensions $(l \times 2^n) \times 1$, l is the total number of training data, 2^n is the dimension of the geometric algebra and each entry for the vector is given by:

$$\begin{aligned} \boldsymbol{a_s^T} &= [(\alpha_s)_1(y_s)_1,\ (\alpha_s)_2(y_s)_1,\ ...,\ (\alpha_s)_l(y_s)_l] \\ \boldsymbol{a_{\sigma_1}^T} &= [(\alpha_{\sigma_1})_1(y_{\sigma_1})_1,\ (\alpha_{\sigma_1})_1(y_{\sigma_1})_1,\ ...,\ (\alpha_{\sigma_1})_l(y_{\sigma_1})_l] \\ &..., \\ \boldsymbol{a_I^T} &= [(\alpha_I)_1(y_I)_1,\ (\alpha_I)_1(y_I)_1,\ ...,\ (\alpha_I)_l(y_I)_l] \end{aligned} \tag{2}$$

where $(\alpha_s)_j, (\alpha_{\sigma_1}), ..., (\alpha_{\sigma_1\sigma_2})_j, ..., (\alpha_I)_j \leq$ are the Langrange multipliers.

The threshold $\boldsymbol{b} \in \mathcal{G}_n^D$ can be computed by using KKT conditions with the Clifford support vectors as follows $\boldsymbol{b} = (b_s + b_{\sigma_1}\sigma_1 + ... + b_{\sigma_1\sigma_2}\sigma_1\sigma_2 + ... + b_I I)$ $= \sum_{j=1}^{l} (\boldsymbol{y_j} - \boldsymbol{w^{\dagger T} x_j})/l$.

The decision function can be seen as sectors reserved for each involved class, i.e. in the case of complex numbers ($\mathcal{G}_{1,0,0}$) or quaternions ($\mathcal{G}_{0,2,0}$) we can see that the circle or the sphere are divide by means spherical vectors. Thus the decision function can be envisaged as

$$\boldsymbol{y} = csign_m\Big[f(\boldsymbol{x})\Big] = csign_m\Big[\boldsymbol{w^{\dagger T} x} + \boldsymbol{b}\Big] = csign_m\Big[\sum_{j=1}^{l}(\alpha_j \circ \boldsymbol{y_j})(\boldsymbol{x_j^{\dagger T} x}) + \boldsymbol{b}\Big] \tag{3}$$

where $csign_m\Big[f(\boldsymbol{x})\Big]$ is the function for detecting the sign of $f(\boldsymbol{x})$ and m stands for the different values which indicate the state valency, e.g. bivalent, tetravalent and the operation "$\circ$" is defined as

[1] The reader can get a detailed explanation about computations of CSVM in [3].

$$(\alpha_j \circ \boldsymbol{y_j}) =< \alpha_j >_0 < \boldsymbol{y_j} >_0 + < \alpha_j >_1 < \boldsymbol{y_j} >_1 \sigma_1 + ... + < \alpha_j >_{2^n} < \boldsymbol{y_j} >_{2^n} I \quad (4)$$

simply one consider as coefficients of the multivector basis the multiplications between the coefficients of blades of same degree. The major advantage of our approach is that we redefine the optimization vector variables as multivectors. This allows us to utilize the components of the multivector output to represent different classes. The amount of achieved class outputs is directly proportional to the dimension of the involved geometric algebra. The key idea to solve multi-class classification in the geometric algebra is to avoid that the multivector elements of different grade get collapsed into a scalar, this can be done thanks to the redefinition of the primal problem involving the Clifford product instead of the inner product of the real approach.

For the nonlinear Clifford valued classification problems we require a Clifford valued kernel $K(\boldsymbol{x}, \boldsymbol{y})$. In general we build a Clifford kernel $K(x_m, x_j)$ by taking the Clifford product between the conjugated of $\boldsymbol{x_m}$ and $\boldsymbol{x_j}$ as follows

$$K(x_m, x_j) = \Phi(\boldsymbol{x}_{\boldsymbol{m}}^{\dagger})\Phi(\boldsymbol{x_j}), \quad (5)$$

note that the kind of conjugation operation † of a multivector depends of the signature of the involved geometric algebra $\mathcal{G}_{p,q,r}$. The results of this paper were obtained using the quaternion-valued Gabor kernel function as follows $\boldsymbol{i} = \sigma_2\sigma_3$, $\boldsymbol{j} = -\sigma_3\sigma_1$, $\boldsymbol{k} = \sigma_1\sigma_2$. The Gaussian window Gabor kernel function reads

$$K(\boldsymbol{x_m}, \boldsymbol{x_n}) = g(\boldsymbol{x_m}, \boldsymbol{x_n}) exp^{-\boldsymbol{i}\mathbf{w}_0^T(\boldsymbol{x_m} - \boldsymbol{x_n})} \quad (6)$$

where $g(\boldsymbol{x_m}, \boldsymbol{x_n}) = \frac{1}{\sqrt{2\pi}\rho} exp^{-\frac{||\boldsymbol{x_m} - \boldsymbol{x_n}||^2}{2\rho^2}}$ and the variables $\boldsymbol{w_0}$ and $\boldsymbol{x_m} - \boldsymbol{x_n}$ stand for the frequency and space domains respectively. Unlike the Hartley transform or the 2D complex Fourier this kernel function separates nicely the even and odd components of the involved signal

4 LSTM-CSVM

LSTM-CSVM is an Evoke based system [2]: the underlying idea of these systems is that it is needed two cascade modules: a robust module to process short and long-time dependencies (LSTM) and an optimization module to produce precise outputs (CSVM, Moore-Penrose pseudoinverse method, SVM respectively). The LSTM module addresses the disadvantage of having relevant pieces of information outside the history window and also avoids the problem of the "vanishing error" presented by algorithms like Back-Propagation Through Time (BPTT) or Real-Time-Recurrent Learning (RTRL)[2]. Meanwhile CSVM maps the internal activations of the fist module to a set of precise outputs, again, it is taken advantage of the multivector output representation to implement a system with less process units and therefore less computational complex.

[2] The reader can get more information about BPTT and RTRL-vanishing error versus LSTM-constant error flow in [5].

LSTM-CSVM works as follows: a sequence of input vectors $(\boldsymbol{u(0)...u(t)})$ is given to the LSTM which in turn feeds the CSVM with the outputs of each of its memory cells.

The CSVM aimed at finding the expected nonlinear mapping of training data. The input and output equations are:

$$\phi(t) = f(\boldsymbol{W}, \boldsymbol{u}(t), \boldsymbol{u}(\text{t-1}),...,\boldsymbol{u}(0),...,).$$
$$y(t) = b + \sum_{i=1}^{k} w_i K(\phi(t), \phi_i(t)). \tag{7}$$

where $\phi(t) = [\psi_1, \psi_2, ..., \psi_n]^T \in R^n$ is the activation in time t of n units of the LSTM, this serves as input to the CSVM, given the input vectors$(\boldsymbol{u(0)...u(t)})$ and the weight matrix $\boldsymbol{W}$. Since the LSTM is a recurrent net, the argument of the function $f(.)$ represents the history of the input vectors.

In the first phase of the training, data were propagated through the LSTM- module of the system, it was training using reinforcement learning with advantage-λ learning to produce a vector of activations $\phi(t)$. Once all k sequences have been seen, the weights w_{ij} of the CSVM are computed using support vector regression/ classification from ϕ to the desired outputs d_i, with $\{\phi, d_i\}$ as training set.

In the second phase, a validation set is presented to the network, but now the inputs are propagated through the LSTM and the newly computed output connections to produce $y(t)$. The error in the classification/prediction or the residual error, possibly combined with the error on the training set, is then used as the fitness measure to be minimized by evolution. By measuring error on the validation set rather that just the training set, LSTM will receive better fitness for being able to generalize. Those LSTM that are most fit are then selected for reproduction where new candidate LSTM are created by exchanging elements between chromosomes and possibly mutating them. New individuals replace the worst old ones and the cycle repeats until a sufficiently good solution is found. Those LSTM that are most fit are then selected for reproduction where new candidate RNNs are created by exchanging elements between chromosomes and possibly mutating them. New individuals replace the worst old ones and the cycle repeats until a sufficiently good solution is found. We evolved the rows of the LSTM's weight matrix using the evolutionary algorithms known as Enforced Sub-Populations (ESP) [7] algorithm. This approach differs with the standard methods, because instead of evolving the complete set of the net parameters, it allows to evolve subpopulations of the LSTM memory cells. For the mutation of the chromosomes, the ESP uses Cauchy density function.

5 LSTM-CSVM with Cost Heuristic for Robot Navigation through Mazes

We built an LSTM-CSVM system in order to deal with the path planning problem for one robot moving through a maze of obstacles to a goal. The recurrent neural system is used as the function approximator for a model-free, value

function-based reinforcement learning algorithm. The state of the environment is approximated by the current observation, (which is the input to the network) together with the recurrent activations in the network, which represent the agent's history. In this case, the recurrent activations in the specialized memory cells, are supposed to learn to represent relevant information from long ago.

The number of input units of the system is stablished by the size of the observation vector, which is four and represents if there is (1) or there is not (0) an obstacle on a particular adjoining position (North, South, East or West). The LSTM module has four memory cells and their activation output values feed one CSVM module which give the final four outputs as a multivector, each element of it represents one of the posible actions to take by the agent on the current state.

We added to the reinforcement system LSTM-CSVM a heuristic approach: on each step t of the learning, we compute a heuristic reinforcement signal $h(t)$. The heuristic reinforcement signal idea is to provide a comparative measure of output action goodnes for each state. These computations are made by a module that we called "'the critic"' [8]. At each time t, the environment provides the path-finder with the input pattern $u(t)$ (environment observation), together with the environmental reinforcement signal $z(t)$. The input pattern is fed to both the LSTM-CSVM and the critic. Nevertheless the LSTM-CSVM does not receive directly the environmental reinforcement signal but the *heuristic reinforcement signal* $h(t)$ elaborated by the critic. The latter is that the LSTM-CSVM produces instantaneously an output patter $y(t)$ that is a multivector which represents the action to execute by the agent. The environment receives this action $y(t)$ and, at time $t + 1$, sends to the LSTM-CSVM both an evaluation $z(t + 1)$ of the appropriateness of the action $y(t)$ for the observation $u(t)$ and a new stimulus $u(t + 1)$.

5.1 The Critic

The goal of the critic is to transform the environmental reinforcement signal into a more informative signal, namely the heuristic reinforcement signal. This improvement is based on past experience of the path-finder when interacting with the environment, as represented bye the reinforced baseline b:

$$h(t) = z(t) - b(t - 1) \tag{8}$$

The critic receives the input pattern X(t) and predicts the reinforcement baseline b(t) with which to compare the associated environment reinforcement signal z(t + 1). We use the technique called predicted comparison [9], it computes the reinforcement baseline as a prediction based on the environmental reinforcement received when the sameor similarinput patterns occurred. That is, the critic has to predict the environmental reinforcement signal $z(t + 1)$ to be received by the path-finder when the stimulus $u(t)$ is present. In order to undertake this task, the critic is built as a second network with a supervised learning algorithm to learn to associate each stimulus with the corresponding environmental reinforcement

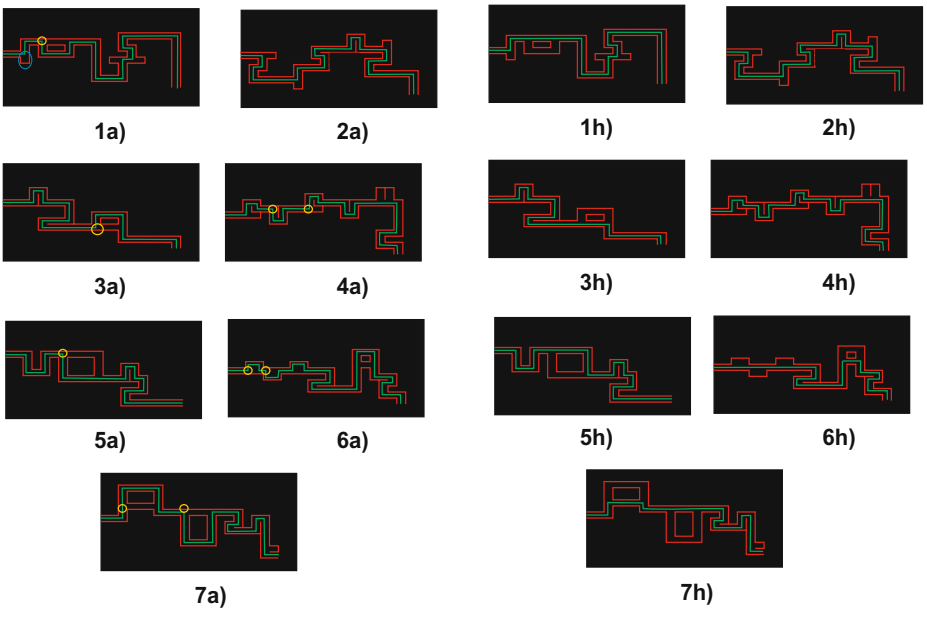

Fig. 1. Left side maze images (1a to 7a) shows path results obtained with LSTM-CSVM without heuristic. Right side shows optimal paths obtained with heuristic approach of LSTM-CSVMv(1h to 7h).

signal. In particular, we have used the "on-line" version of the backpropagation algorithm with a momentum term. the finding path improves, the mapping from stimuli to reinforcement changes.

5.2 Results

We evolved the recurrent LSTM-CSVM system during 30 generations using a Cauchy noise parameter of $\alpha = 10^{-}3$. The first phase of the experiments consisted of simulated mazes discretized on 5 by 5 pixels cells. The length of each maze varies from 100 to 300 cells. Left side of figure 1 shows the paths found with LSTM-CSVM approach (which was evolved during 60 generations using a Cauchy noise parameter of $\alpha = 10^{-}3$) meanwhile right side shows the shortest paths obtained with heuristic approach of LSTM-CSVM. It is important to note that on results obtained with LSTM-CSVM approach the agent seems to have learned a policy which tells the agent to take action go-north every time it finds a two junction like shown on (Fig. 1-3a), 7a)). The policy also tells the agent to take the action go-south on two junction like shown on (Fig. 1-1a), 5a)). These choices of actions produce suboptimal paths.

Phase two of agent navigation through mazes was applied to real mobile robot system, it comprises mobile-base robot, a stereoscopic camera and a laser sensor. The task consisted to move the robot through a real 2D labyrinth. The labyrinths

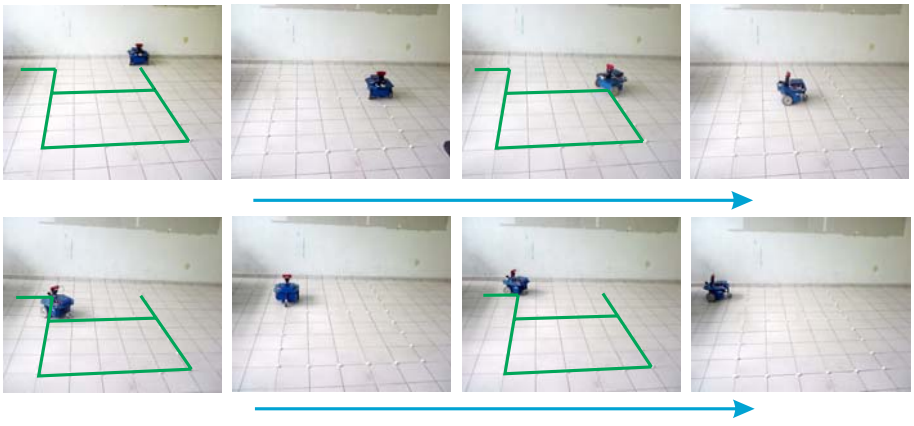

Fig. 2. Image sequence of a mobile robot following the shortest path finding by the heuristic approach of LSTM-CSVM. Blue arrows show the reading order of the images.

used on this phase were much more shorter than the simulated on phase one. The mazes were discretized on 10 by 10 pixels cells and the length of each maze varied from 20 to 30 cells. The LSTM-CSVM with heuristic approach was evolved during 15 generations and the one path found is shown on (Fig. 2). Note that the mobile robot takes the shortest path to the final goal position.

6 Conclusion

In the robot path finding problem , the consequences of an action can emerge later in time. Thus, actions must be selected based on both their short- and long-term consequences. Neverthless many learning algorithms have limited long-term memory. The approach presented in this paper overcomes this limitation thanks to its LSTM module which is responsible to capture the short- and long-term correlation between input data sequences meanwhile the CSVM module allows to produce precise outputs. By adding the heuristic approach to the LSTM-CSVM system we can be able to overcome the problem of time consuming of traditional reinforcement learning in the initial learning phase, this heuristic methodology also allows to find shortest paths as is shown in the results subsection and along the numerous experiments that we had conducted.

References

1. Bayro-Corrochano, E., Arana-Daniel, N., Vallejo-Gutierrez, R.: Recurrent Clifford Support Machines. In: Proceedings IEEE World Congress on Computational Intelligence, Hong-Kong (2008)
2. Schmidhuber, J., Gagliolo, M., Wierstra, D., Gomez, F.: Recurrent Support Vector Machines, Technical Report, no. IDSIA 19-05 (2005)

3. Bayro-Corrochano, E., Arana-Daniel, N., Vallejo-Gutierrez, R.: Geometric Preprocessing, geometric feedforward neural networks and Clifford support vector machines for visual learning. Journal Neurocomputing 67, 54–105 (2005)
4. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient ow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer, S.C., Kolen, J.F. (eds.) A field guide to dynamical recurrent neural networks. IEEE Press, Los Alamitos (2001)
5. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory, Technical Report FKI-207-95 (1996)
6. Hestenes, D., Li, H., Rockwood, A.: New algebraic tools for classical geometry. In: Sommer, G. (ed.) Geometric Computing with Clifford Algebras. Springer, Heidelberg (2001)
7. Gómez, F.J., Miikkulainen, R.: Active guidance for a finless rocket using neuroevolution. In: Proc. GECCO, pp. 2084–2095 (2003)
8. Millán, J.R., Torras, C.: A Reinforcement Connectionist Approach to Robot Path Finding in Non-Maze-Like Environments. J. Mach. Learn. 8, 363–395 (1992)
9. Sutton, R.S.: Temporal credit assignment in reinforcement learning. Ph.D. Thesis, Dept. of Computer and Information Science, University of Massachusetts, Amherst (1984)