Investigation on using Kernels with Q-learning for gait adaptation on visually guided walking robots

Francesco Tenore, Roman Genov

Abstract.
In this work we apply a reinforcement learning algorithm (Q-learning) to a simulated walking robot to "teach" it to step over hurdle-type obstacles. Two ways to achieve this are described in detail. The first method uses a gait model-based approach; the second explores the state-action space thoroughly by using a non-zero probability of a random transition. Finally we use a kernel (radial basis) function to speed up the learning process.

1. Introduction

In the standard reinforcement-learning model, an agent is connected to its environment via perception and action. On each interaction step the agent receives an input, which encodes information regarding the current state, *S*, of the environment. The agent then chooses an action, *a*, to generate as output. The action changes the state, and the value of this state transition is communicated to the agent through a discrete delayed reinforcement signal, *r*. The agent should choose actions that maximize the long-run sum of values of the reinforcement signal. It learns this over time through trial and error, and can be guided by a wide variety of reinforcement algorithms [1].

2. An overview of Q-learning

In our case, the algorithm that was chosen was Q-learning. Q-learning [2], is a form of reinforcement that does not require a model of its environment and can be used on-line. It works by estimating a quality index of state-action pairs and provides robots with the capability of learning to act optimally in a Markovian environment [3].

Assuming the robot can discriminate the set ***S*** of states and can take the set ***A*** of actions, the way the algorithm works is the following.

i) A matrix *Q* of *S* rows and *a* columns is initialized to 0;

ii) current state *S* is acknowledged;

iii) an action *a* is chosen according to the ***policy***:

$$f(S) = \arg\max_a Q(S,a) \qquad (1)$$

The *value function* is the quality index that corresponds to the choice of such a policy:

$$V_f(S) = \max_a Q(S,a) \qquad (2)$$

iv) action *a* is carried out in the environment and the next state becomes *S'*;

v) update the value function and therefore the Q-matrix through the *learning rule*:

$$Q(S_t, a_t) \leftarrow (1-a)Q(S_t, a_t) + a(r + gV_f(S_{t+1})) \qquad (3)$$

where α is called the *learning rate*, and is typically a low positive number between close to 0, and γ is the *discount factor*, also fixed between 0 and 1, and finally *r* is the *reinforcement signal* and is set to –1 if the action taken resulted in a failure, whereas it is set to a +1 value if the result of the action was a success.

Learning rule (3) shows how the update of the matrix occurs. Since α has a small value (0.1 in our simulation), one can see that the Q-matrix entry at time *t+1* comes largely (90%) from the value of the previous state (*t*), and partially (10%) from the sum of the reinforcement signal and the value function.

3. Gait model-based learning

3a. Implementation

We assume a finite number of actions and a finite number of states. In the future it will be desirable to work with an infinite number of states, thereby requiring a continuous function to describe the state. In this simulation we show a robot walking at a normal pace if no obstacle is in its field of view. When an obstacle appears, the robot will start changing its gait according to its policy. The obstacle is displayed as a rectangle into which the robot's feet cannot step into or a punishment will be enforced (reinforcement signal equal to –1). In a real world environment this is equivalent to stepping over an obstacle: the rectangle's longer sides represent the two limits for performing a correct jump. I it stepped within this rectangle the robot's last step should either be increased (if possible) or decreased enough to become the next-to-last step.

Therefore the state-space encodes the distance from the obstacle, which in the real-world scenario will be given to us through a stereo-vision chip mounted on a camera on the robot's legs. There are a total of 75 states, and 8 actions corresponding to the length of the steps that the robot can make, the longest being action 1. In simulation, after every obstacle, a new one appears after a random number of steps. The distance to it is also partially random. To ensure learning, we empirically choose to reach 30 consecutive successes. As output, the simulation shows the relative distance covered, the number of iterations (obstacles) that were required to achieve the 30 consecutive successes and a graphical

representation that shows the complete set of trials required to accomplish the task.
A part of these trials are shown in Figure 1.

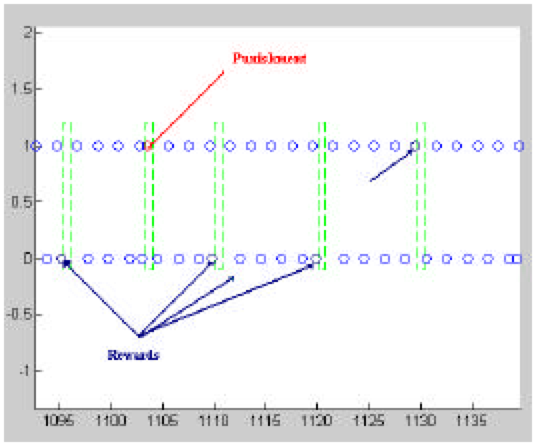


Figure 1. Zoom-in of a part of the robot's "journey". The blue circles represent the robot's footsteps; the green rectangles are the obstacles.

Note that in this representation, the top circles are the places where the *left* foot stepped and the bottom circles correspond to the *right* foot. Also, it should be noted that the probability of having 30 consecutive successes without gait adaptation, and therefore having a constant step size equal to its maximum possible size, is over 11 orders of magnitude greater than what we obtain.

In the following figure we show a 3-D plot of what the Q-matrix obtained. The "canyon" shown in the figure corresponds to state-action pairs that prevent the robot from surmounting the obstacle. This diagonal part of the matrix is essentially the part that the robot must avoid in order to obtain a success. In the figure, the actions are numbered from 1 to 8 the states from 1 to 75 and the z coordinate is the quality index for each position in the matrix.

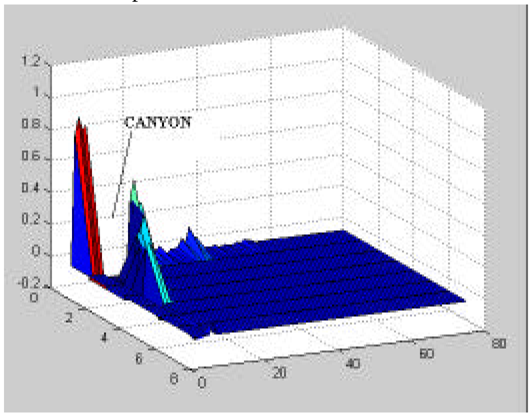


Figure 2. This 3-D plot of the Q-matrix shows the negative values on a diagonally shaped area of the matrix.

This, however, was the original version of the program, which did NOT have any degree of randomness in the choice of the action. So we did not explore the state-action space but used a predefined model in which the step size is always initially chosen to be maximum (action 1). Also, since the Q-matrix was not entirely explored a small number of iterations is required for learning.

3.b Results with predefined gait model.

In order for the robot to have learned, we empirically said that this occurs once the simulation has reached 30 consecutive successes. A histogram that plots the results is shown in Figure 3. An average of 120 iterations are required to reach this goal. This value is highly influenced by the only outlier (the standard deviation is equal to approximately 35). It was therefore quite interesting to also see what the result for the median would give. This was seen to be 110, significantly lower than the mean.
In this case, experiments with noise were also conducted. In these experiments, it was assumed that every action would have some noise to it and therefore the robot would not always place its "foot" where it was intended to be. Also, it was decided that the maximum step size coincided with the robot's normal step and was therefore not subject to noise. Under these conditions, the addition of noise proportional to step size linearly increased the number of obstacles required for learning.

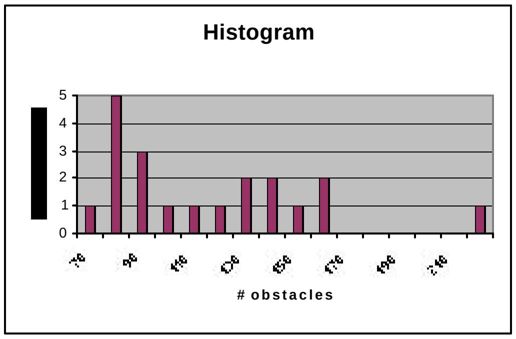


Figure 3. Plot of the number of learning successes versus the number of iterations required (in 20 total simulations).

4. Exploration of the state space

4a. Implementation
The next step was to add the degree of randomness (mentioned at the end of section 3a) to allow the exploration of the state-space. The probability distribution was chosen so as to favor longer steps to shorter ones, thereby inherently allowing for a faster stride. Its cumulative distribution function is shown in Figure 4.

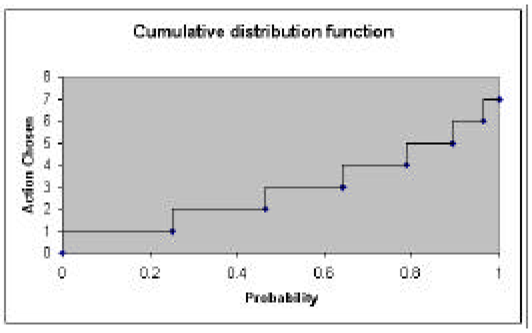


Figure 4. The cdf is used to decide which action to take.

Note, however, that there is no reinforcement signal that rewards a faster pace as opposed to a slower one. This might be added in future versions of the simulation.

This exploration approach is implemented in the following way.

Stage 1. As the robot starts learning, or every time it runs into an obstacle, two possibilities can occur in deciding how to take an action. Either it chooses the policy (the action that gives the maximum Q-value in a given state), or it chooses the action randomly. This second part occurs with a probability of 0.35. In essence, we take a randomly generated number in the interval [0, 1] and if the number is higher than 0.65 then we generate another random number, which will decide which action to take, based on the probability distribution, shown in figure 4, that favors long steps to shorter ones.

Stage 2. Once the robot reaches a certain number of consecutive successes, which we choose empirically to be 4, the probability of selecting a random action as opposed to the one dictated by the policy decreases linearly with the number of successes; we also weight the probability with the value function.

Stage 3. Having reached the second threshold of 10 consecutive successes (also chosen empirically), we argue that the matrix is converging and therefore we should exploit only the policy. If, in this last scenario, the robot bumps into the obstacle then the process goes back to *stage 1*.

In this manner one can predict that the entire matrix, or almost all of it, will be constituted by non-zero elements.

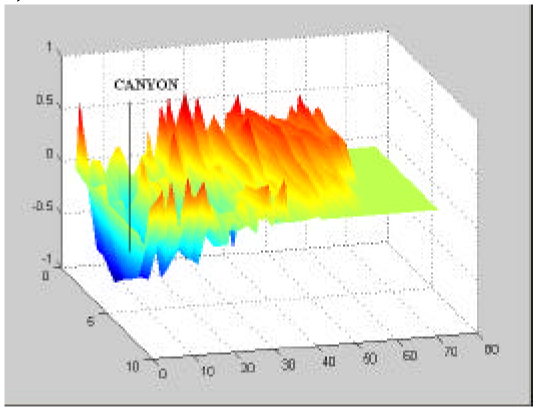
As expected, the negative elements appear only on a particular "diagonal", just as shown in section 3a, which corresponds to the set of state-action pairs from which the obstacle cannot be passed. This situation is shown in Figures 5a and 5b (3D and 2D plots).

The number of iterations required for a complete exploration and to reach the goal of 30 consecutive successes is about **one to two orders** of magnitude larger compared to the situation in which the gait model, not exploration, is used. This is mainly due to the randomness in the decision of the action: it takes more iterations to explore a greater part of the Q-matrix.

4.1. Results in exploration

Since these experiments require larger amounts of time, only 8 simulations were run. The resulting number of iterations required had an average of 13,000, a standard deviation of 12,000 (one large outlier at 40,000) and a median of 10,000.

a)



b)

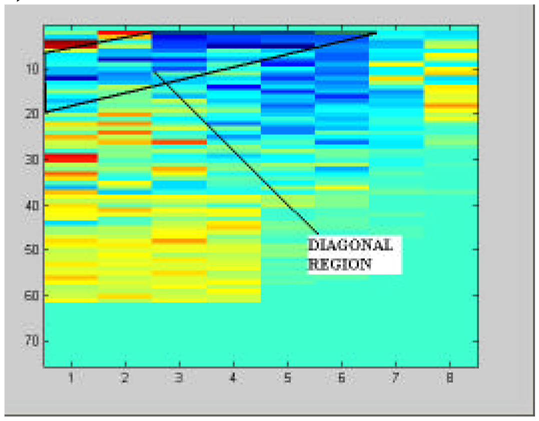


Figure 5. a) 3-D plot of Q-matrix after exploration of the state-space. The exploration accounts for the smoothness.
b) 2-D plot showing the position of the diagonal region into which the robot should not step.

If the threshold points in stages 1 and 2 had been different, different results would have been obtained. Research in this part of the project will be dealt with in future improvements.

5. The radial basis function

The radial basis function is introduced to decrease the number of iterations required for learning in the exploration case (section 4a). Essentially, every time the robot passes an obstacle, whether correctly or incorrectly, the matrix updates not only the last entry, but also its nearest neighbors according to the radial basis function. This exploits the fact that the finer one quantizes the state-action space, the more probable it is that the elements surrounding a given state-action pair will have the same topological properties, thereby allowing for a quicker update of the entire matrix and, in the limit, learning in the continuous domain.
The radial basis function decreases like the square of an exponential. The formula used to update the surrounding elements will be:

$$\forall i, j$$
$$Q(S_i, a_j) \leftarrow Q(S,a) \cdot \exp(-\|(S,a) - (S_i, a_j)\|^2) + Q(S_i, a_j) \quad (4),$$

where, in our case, the square norm represents the square of the distance between any Q-value and the element updated by reinforcement alone (*Q(S,a)*).
The problem with this implementation was the fact that it did not converge, therefore a modification of the update formula (probably with a normalization factor in front) will be required for future evolutions of the function.

6. Conclusions

To implement the reinforcement algorithm correctly it is necessary to add a certain probability that a random action in a given state can occur. It has been shown in this paper, however, that if the probability is not added, and instead a gait model is used, the number of iterations, and therefore obstacles, required for learning is sensibly smaller.
The question arises as to whether or not it is necessary to explore the state-space, and therefore which of these two options will be chosen to be implemented on the real robot. Undoubtedly, the first option will be tested first, being easier to work with. If no real problems are encountered with this methodology it might not be worthwhile to test out the exploration. Noise or other factors could become important (although tests with noise have been conducted in the gait model scenario and the number of iterations required for learning did not increase significantly) issues that might force us to explore the rest of the state-space.
The kernel approach so far has not been proved useful but through certain modifications (such as an update of only a close neighborhood of the reinforced element) and a closer study of the behavior of the matrix, more detailed conclusions might be made.
Improvements that could be added to these basic functions include a reinforcement signal that awards higher speed of accomplishment of the task. Also, a quasi-continuous state-space could be implemented. This is an approach in which the input (distance to obstacle, i.e. current position) is continuous and therefore lies between two states of the Q-matrix. We would then use a radial basis function to update these two states according to their distance from the position. By extension, also other close states can be updated, albeit to a much smaller extent.

References

[1]. Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore, “Reinforcement Learning: A Survey”; http://www.cs.brown.edu/people/lpk/rl-survey/rl-survey.html

[2]. C. Watkins, P. Dayan, “Q Learning" *Machine Learning*, vol. 8(3), pp. 279-292, 1992.

[3]. Y. Takahashi, M. Takeda, M. Asada, “Continuous valued Q-learning for vision-guided behavior acquisition”; Proceedings of the 1999 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Taipei, Taiwan, ROC. August 1999