Control of a kinematic robot arm

Martina Klose

December 21, 1995

1 Introduction

This paper shows how reinforcement learning can be applied to learn the control of a kinematic robot arm.
The optimization criterion is to minimize the number of time steps needed to reach a given goal position from a fixed or arbitrary start position. As the state space, containing the possible joint angle positions of the arm, is continuous, a function approximator, here an RBF net, is used.

Section 2 introduces the general problem setting, the used learning algorithm and the net architecture. The results of the different experiments are presented in section 3. Finally section 4 concludes with a discussion of the experiences made and the difficulties met during the realization of this control task.

2 Problem Setting

Here a robot arm with two segments was assumed, similar to that shown in figure 1. The segment lengths were set to $l_1 = 0.5$ and $l_2 = 0.33$.
Now the task is, given a start position defined by an initial joint angle position, to reach a goal position in minimal number of time steps. The goal position is specified in terms of a desired position of the arm's end point.
For two particular angle values q_1 and q_2 the end point's coordinates can be determined applying the forward kinematic function:

$$x1 = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2);$$

$$x2 = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2);$$

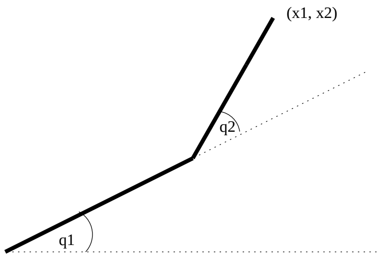

Figure 1: robot arm

Two different state spaces were investigated:
In the *restricted* state space, q_1 covered values between $-\pi/2$ and $\pi/2$ and q_2 between 0 and π. Whereas in the *unrestricted* case all angles between $-\pi$ and π resp. 0 and 2π were allowed.

At each time step the arm could increase or decrease both angles. Generally continuous changes would have been possible. Here however the actions were discretized. Again two settings of the action space were used:
In the first both angles could be changed by -0.05π, 0 and 0.05π. The second setting allowed changes of -0.2π, -0.05π, 0, 0.05π, 0.2π.
The experiments showed that it was crucial for successful learning that the arm changed its position in each time step before getting to the goal. Therefore action $\Delta q_1 = 0$ and $\Delta q_2 = 0$ was forbidden. Thus setting 1 implied 8 different actions, setting 2 24 different actions per time step.

This discretization of the actions made it impossible to reach each end position from each start position exactly. Therefore the goal was said to be reached if the distance between the arm's end point and the goal position became less than a threshold, here set to 0.1.

As the optimization criterion was to minimize the number of time steps to get from start to goal a negative reward of -0.01 was given for each step before the goal region was entered.

I decided to learn state values. Thus a form of value iteration was used, using one step of lookahead to determine the largest value of the possible successor states. No discounting was used and λ was set to 0.3.

$$\Delta\theta_t = \rho(reward + \max_{s'} V_t(s') - V_t(s_t))e_t,$$

$$where\ e_t = \lambda e_{t-1} + \nabla_{\theta_t} V_t(s_t)$$

(θ already relates to the parameter vector of the used function approximator).

Here a very simple exploration scheme was applied: Starting with an exploration rate of 1, i.e. a totally random action selection, this rate was linearly decreased to zero over 1000 resp. 2000 trials, due to the specific task.
Nevertheless the current state value was always updated with the maximal value of its possible successor states, even if the selected action did not lead to that state, as was usually the case for random action selection.

At each time step it was checked if the arm could enter the goal region by doing any of the allowed actions. If that was the case this action was selected. The present one-to-terminal state was then updated with the desired value -0.01 (resp. 0.89, 0.99 see section 3 for explanation), referring to an implicit terminal state value of 0.0.
Explicitly learning a value for the goal states, which would be 0.0 (resp. 0.9, 1.0) showed to complicate the learning process.

As the state space was continuous a function approximator was used. I applied RBF nets with 15 resp. 60 hidden units for the restricted resp. unrestricted joint angle space. As output unit a sigmoid unit with hyperbolic-tangent squashing function was used.
Besides updating the weights from the hidden to the output layer also the mean values and the variances of the RBF units were updated.
The learning rate was set to 0.01 for the weight updates and to 0.1 for the mean values and variance updates. All weights were initialized with 0, the variances with 1 and the mean values with random values belonging to the allowed angle space.

3 Results

3.1 Fixed start position

The first subtask was to get from a fixed start position, which was generally set to $q_1 = 0$, $q_2 = 0$, to a given end position, resp. a close region around it.
The results for the restricted and unrestricted cases are presented separately in the following two subsections. In both cases the two different action settings were investigated. Not surprisingly it generally took longer to learn the arm restricted to the rather tiny angle changes, than that allowed 24 partly larger movements at each time step. One reason could lay in the fact that in the second case the search space could be explored and the goal region entered much faster.

3.1.1 Restricted state space

When allowed 24 actions it was sufficient to reduce the exploration rate to 0 over 1000 trials. For the tested start-goal combinations the arm learned to get to the goal region in minimum number of steps within a few trials, usually between 100 and 150. The rather slow decrease of the exploration rate was needed to avoid the task to be unlearned again. Figure 2 shows one specific arm movement learned for the desired end position set to $x_1 = 0.5$ and $x_2 = 0.25$

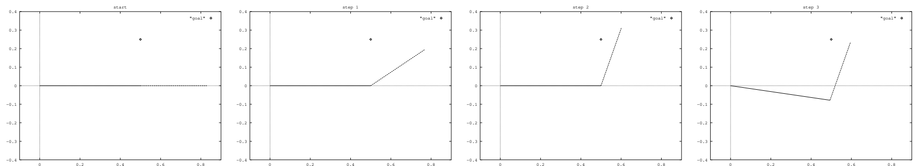

Figure 2: movements from start $(q_1, q_2) = (0, 0)$ to goal $(x_1, x_2) = (0.5, 0.25)$, denoted by a dot

The arm restricted to 8 actions per time step was also able to learn the task. However exploration had to be decreased more slowly in order to get stable results: Here the exploration rate was decreased over 2000 trials. Moreover the arm needed 1500 to 2000 trials to learn the task and repeated unlearning during the first 1500 trials was common. But finally a minimum path was detected.

In addition another modification was necessary to speed up or even allow learning of the 8 movement arm : The value function had to be shifted, i.e. the states one step prior to the goal region had to be trained with desired output 0.89 instead of -0.01, referring to a terminal state value of 0.9. This seemed to be connected with the longer trial length.

Figure 3 shows the two value functions referring to the desired end positions being $x_1 = 0.5$ and $x_2 = 0.25$. Also the path from start to end state is indicated.

Here a general problem was to learn to get to the extreme position $x_2 = 0.0$, $x_2 = -0.83$. When allowed 5 movements the arm learned a shortest path but unlearned it again. When only 3 movements were allowed, it was even hard to get into the goal region when choosing actions randomly.

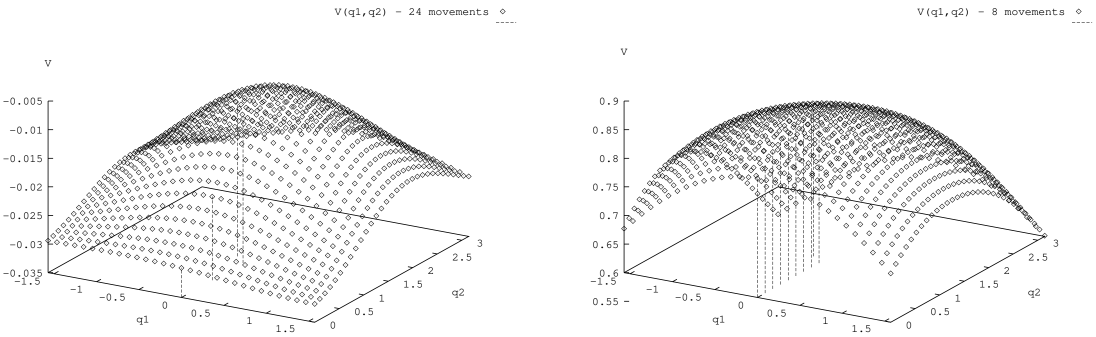

Figure 3: value functions and visited states for the case of 24 resp. 8 possible movements per time step

3.1.2 Unrestricted state space

With no constraints to the joint angle position, the arm was generally much easier to train. Due to the larger state space the exploration rate was decreased over 2000 trials.

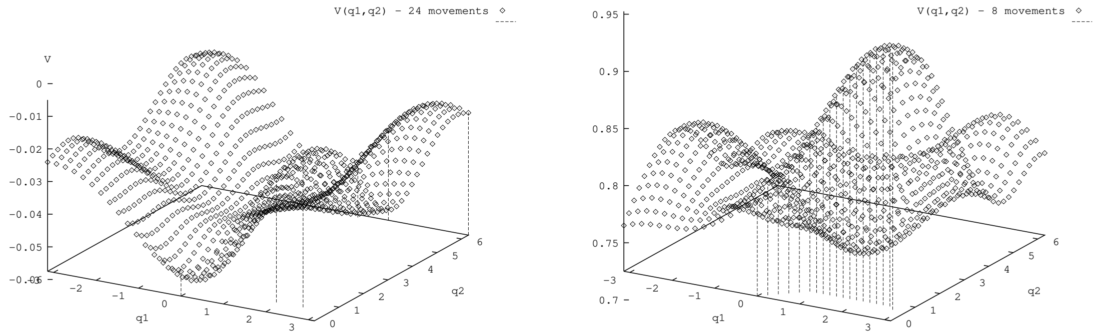

Figure 4: value functions and visited states for the case of 24 resp. 8 possible movements per time step

As in the restricted case the arm with the greater variety of movements learned much faster: Depending on the specific end point between 50 and 200 trials were needed.
The arm limited to 8 movements needed between 50 and 800 trials. Moreover

the value function had to be shifted again. Here the implicit goal state value had to be set to 1.0. Therefore 0.99 was used as the desired value of the states one step prior to the goal region.
All tested end position could be learned to be reached in minimum number of steps.

Figure 4 shows the value functions referring to one of the possibly hardest tasks, i.e. to get to $x_1 = -0.83$, $x_2 = 0.0$. Also the visited states are shown. Remark that the goal states do not have the highest value, what should not surprise as they are not explicitly learned.

3.2 Random start positions

An extension of the task with a fixed start position, was to allow the arm to start from arbitrary start positions. Here only the arm with 24 movements was investigated in the restricted and unrestricted angle space. Exploration was decreased over 1000 trials.

In order to test the performance resp. the improvement of the arm's behavior, training was interrupted after every 100 trials. The net was then tested on a fixed test set consisting of 121 resp. 400 start positions for the restricted resp. unrestricted case. These test positions were evenly spaced over the state space. During testing, trials which took longer than 50 time steps were stopped.

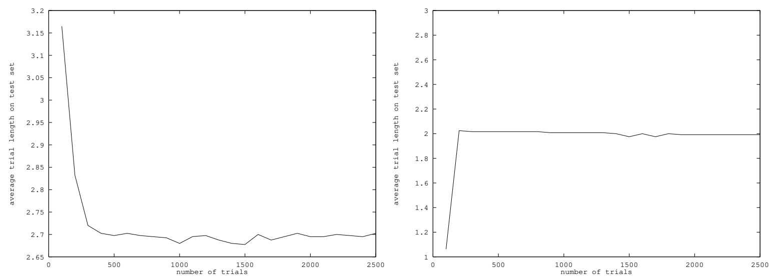

Figure 5: average path length on the test set in the unrestricted and restricted case.

In the unrestricted case the arm learned to reach the goal region within a small number of steps very fast. The largest improvements appeared during the first 300 trials. After that the average path length fluctuated on a rather constant level.
After the first 100 trials the restricted arm still needed more than 50 steps for a couple of tested starting positions. But after 200 trials it reached a minimum

and stays at this level.
Figure 5 shows the average trial length over time, when the goal state is $x_1 = 0.5$, $x_2 = 0.25$. Remark that the average was taken only over those testing trials which reached the goal. This explains the increasing curve shown in the second graph.

For the arm in the unrestricted state space it was no problem to learn to reach different goal regions from arbitrary start positions. In the restricted case the extreme position $x_1 = 0.0$, $x_2 = -0.83$ yielded problems as expected.

4 Conclusion

This experiments showed that one can use RBF nets using a combination of TD(λ) and backpropagation to learn the control of a kinematic robot arm.
However I had to face some difficulties: The performance of the net was very sensitive to the parameter settings, e.g. the learning rates.
But also the function to be learned had to be modified for the different subtasks, in that for some problems the value function needed to be shifted towards larger values.
Also the amount of necessary exploration had to be changed from subtask to subtask. Here a more sophisticated exploration scheme could perhaps help.
Generally speaking, it was not possible for me to find a setting, including the function values and the exploration scheme which was applicable for each task.

It should further be mentioned, that RBF nets showed to be favorable to usual Backprop nets with sigmoidal hidden units, due to there more locally behavior.

The evaluation of the results was sometimes a bit vague, e.g. as I could not check all start and end combinations. Moreover especially in the second subtask including random start positions it was rather difficult to judge in how far the trained net reached optimality.

Nevertheless the results of the unrestricted arm were rather promising. Here further work could try to extend the task to continuous actions. Moreover more reasonable optimality criteria with respect to real conditions could be applied. I tried to penalize any single movement instead of any time step, which worked equally well for the cases I tested.
Concerning relevance to real applications it would be helpful to learn the state-action values instead of the state values. Then one would not need to evaluate each successor state in order to select the next action resp. successor state.