Effective Strategies for Complex Skill Real-time Learning Using Reinforcement Learning

WEI Ying-zi\(^1,2,3\), ZHAO Ming-ya\(^4\)

1. Robotics Laboratory, Shenyang Institute of Automation, Chinese Academy of Science, Shenyang 110016, China;
2. Shenyang Institute of Technology, Shenyang 110168, China;
3. Graduate School of the Chinese Academy of Sciences, Beijing 100039, China
yzwei@ms.sia.ac.cn, myzhao@ms.sia.ac.cn

Abstract
Following the principle of human skill learning, robot acquiring skill is a process similar to human skill learning. Reinforcement learning is an on-line actor critic method for robot to develop its skill. The reinforcement function has become the critical component for its effect of evaluating the action and guiding the learning process. A difference form of augmented reward function is considered carefully.

In this paper, we present a strategy for the task of complex skill learning. Automatic robot shaping policy is to dissolve the complex skill into a hierarchical learning process. Variable resolution discretization of input space is introduced to improve the generalization capability of CMAC-based RL. Conventional \( \epsilon \)-greedy policy has the shortage of unnecessary randomization. Boltzmann distribution selection is also introduced to the balance of exploration and exploitation. We describe our ideas of reinforcement learning methods and also illustrate with an example the utility of method for learning skilled robot control on line.

1 Introduction

Robots have been used successfully in manufacturing settings, where the environment is very structured and the tasks performed are repetitive and relatively simple. The research is inspired by the vision of robots moving beyond the structured, controlled settings, and still performing successfully robot shaping to accomplish complex tasks. When human perform a task, their actions reflect the skill associated with that task. Human has the ability to apply past knowledge and experience in performing given tasks, which called human skill. The operators cannot completely describe their skill, but can demonstrate it. The skill is gained and incrementally improved through learning and practicing. Using the idea of human skill learning, robot performing a task in an unstructured environment is mainly to make it acquire the skill associated with that task.

Real time learning is essential for robot to work in unstructured settings. Reinforcement learning (RL) is an on-line actor critic method in machine learning and artificial intelligence. The reinforcement learning paradigm aims at mimicking the animal behavior derived from its interaction with the world during a goal search. A robot learns how to operate in the environment by receiving a reinforcement signal that indicates the usefulness of its action to reach the goal. RL is an attractive solution for developing robot skills because it does not require a model of the environment, nor does it require the designer to specify the exact behavior of the robot in every situation that may occur. The interaction of a robot with the environment is its major source of skill. This is distinct from a supervised learning approach where selected input-output data set is used to train a learner. Another nice feature of the Reinforcement Learning approach is that all non-linearity like bandwidth limitation of the actuators, torque limitation, etc. are encapsulated from the viewpoint of the learner. In recent years, RL not only attracts a lot of researcher in psychology but also in computer science, control engineering and operation research. In computer science, lots of research work had been done on the theory, algorithms and application of reinforcement learning.

This paper is organized as follows. In section 2, we describe the related theory of learning method, and an augmented form of reward function is given in section 3. From section 4 to 6, we present the ideas of implementing for acquiring complex skill in detail. Section 7 and 8 provide an example to test the idea and result is given.

2 Related Work

2.1 Markov Decision Processes

Markov Decision Processes (MDPs) are a standard, general formalism for modeling stochastic, sequential decision problems. A (finite-state) Markov decision process (MDP), is a tuple \( M = (S, \{a\}, T, \gamma, R) \), where: \( S \) is a finite set of states; \( A = \{a_1, a_2, \ldots, a_k\} \) is a set of 2 actions; \( T = (P_{sa} \gamma) \) where \( a \in A \) are the next-state
transition probabilities, with \( P_{as}(s') \) giving the probability of transitioning to state \( s' \) upon taking action \( a \) in state \( s \); \( \gamma \in (0,1) \) is the discount factor; and \( R \) specifies the reward distributions. For simplicity, we assume rewards are deterministic, in which case \( R \) is a bounded real function called the reward function. In the literature, reward functions are typically written \( R: S \times A \rightarrow \mathbb{R} \), with \( R(s,a) \) being the reward received upon taking action \( a \) in state \( s \). We introduce a more general form, \( R: S \times A \times S \rightarrow \mathbb{R} \), with \( R(s,a,s') \) being the reward received upon taking action \( a \) in state \( s \) and transitioning to state \( s' \).

### 2.2 Reinforcement Learning

Many RL algorithms estimate value functions, which are defined with respect to policies and reflect the expected value of the return. The action-value function of policy represents the expected discounted return obtained when starting from state \( s \), taking \( a \), and henceforth following \( \pi \). The optimal action-value function is

\[
\pi: Q^\pi(s,a) = E_t \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]
\]

The optimal action-value function is

\[
\pi^*(s,a) = \max_{\pi} Q^\pi(s,a), \forall s, a \in \mathcal{A}.
\]

An optimal policy is one for which this maximum is attained. If the optimal action-value function is learned, then an optimal policy can be implicitly derived as a greedy one with respect to that value function.

Most RL algorithms iteratively improve estimates of value functions based on samples of transitions obtained on-line. For example, at each time step \( t \), the typical tabular learning algorithm updates the value of the current state-action pair \((s_t, a_t)\) based on the observed reward \( r_{t+1} \) and the next state-action pair \((s_{t+1}, a_{t+1})\), as:

\[
Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s,a) \right], \quad \alpha \in (0,1)
\]

The aim of the learning system is to improve the controller so as to maximize some function of the reinforcement, e.g., its instantaneous reward, or the expected sum of its rewards, or its average. There are a few key concerns when applying reinforcement learning in robot settings: first, RL can be very slow in large domains; and second, RL may be made distracted from whatever it really should be trying to do when robot is learning complex skill.

### 3 Augmented Form of Reward Function

The reinforcement is usually closely coupled to the performance metric for a task. In fact many reinforcement learning investigations consider performance task and reward as one and the same. Since learning system strive to maximize the reward signal provided to them, performance is maximized when their reward closely parallels performance. A reward function is defined as performance if and only if maximum reward implies optimal performance. But in many tasks a performance based reward is always sparse and delayed. It may take many steps for a robot to reach a non-negative reward. In a forage task for instance, measured performance is negative or zero until an attractor is delivered. This delay makes it more difficult for a robot to assign credit or blame to actions taken in the past than if the immediate rewards are provided. It is necessary for a robot being provided with more immediate feedback during the whole process of reinforcement learning.

In this paper we propose a method for accelerated learning by extending and structuring reward functions to take advantage of domain knowledge and experience. This \( R = R + F \) is the form of augmented rewards. \( R \) is the traditional reward and \( F \) is supplying additional rewards to a learning robot to guide its learning process, beyond those supplied by the underlying markov decision process (MDP).

To speed up learning, we provided positive rewards whenever the agent made progress towards the goal. To encourage moving towards a goal, the additional reward function \( F(s,a,s') = r \) may be some positive reward whenever \( s' \) is closer (in whatever appropriate sense) to the goal than \( s \), and \( F(s,a,s') = 0 \) otherwise. The augmented reward often seek to improve the performance by providing more immediate feedback to accelerate RL process.

---

**Figure 1:** A block diagram representation of RL with augmented reward

---

The reinforcement is usually closely coupled to the performance metric for a task. In fact many reinforcement learning investigations consider performance task and reward as one and the same. Since learning system strive to maximize the reward signal provided to them, performance is maximized when their reward closely parallels performance. A reward function is defined as performance if and only if maximum reward implies optimal performance. But in many tasks a performance based reward is always sparse and delayed. It may take many steps for a robot to reach a non-negative reward. In a forage task for instance, measured performance is negative or zero until an attractor is delivered. This delay makes it more difficult for a robot to assign credit or blame to actions taken in the past than if the immediate rewards are provided. It is necessary for a robot being provided with more immediate feedback during the whole process of reinforcement learning.

In this paper we propose a method for accelerated learning by extending and structuring reward functions to take advantage of domain knowledge and experience. This \( R = R + F \) is the form of augmented rewards. \( R \) is the traditional reward and \( F \) is supplying additional rewards to a learning robot to guide its learning process, beyond those supplied by the underlying markov decision process (MDP).

To speed up learning, we provided positive rewards whenever the agent made progress towards the goal. To encourage moving towards a goal, the additional reward function \( F(s,a,s') = r \) may be some positive reward whenever \( s' \) is closer (in whatever appropriate sense) to the goal than \( s \), and \( F(s,a,s') = 0 \) otherwise. The augmented reward often seek to improve the performance by providing more immediate feedback to accelerate RL process.

---

**Figure 1:** A block diagram representation of RL with augmented reward

---

389
To encourage taking action \( a_i \) in some set of states \( S_0 \), we set \( F(s,a,s') = \tau \) whenever \( s = s_0 \), \( s' \notin S_0 \), and \( F(s,a,s') = 0 \), otherwise. This provides a new way to incorporate prior knowledge and experience into the learning process for RL controller. This idea is utilized specially for RL based on the function approximator of CMAC to incorporate prior knowledge and experience into the learning process in this paper. For instance, we assume a production rule is:

If \( s \in S_0 \) then \( a=a_1 \), \( F=\tau \). (4)

So the rules and experience are included. This knowledge reduces the effective number of state-action pair that RL must explore. Since search time scales exponentially with the state space size, this shaping reward function can lead to substantial reductions in learning time.

If the robot cycles through a sequence of states \( (s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n \rightarrow s_{i+1} \rightarrow \ldots) \), we must be sure it is not artificially rewarded. So if \( R'=R+F \), the following must be satisfied

\[
F(s_1,a_1,s_2) + \ldots + F(s_{n-1},a_{n-1},s_n) + F(s_n,a,s_n) 
\]

\( F(s,a,s') \) must have the form of a difference of potentials:

\[
F(s,a,s') = \tau \Phi(s') - \Phi(s). 
\]

4 Fast Function Approximator

Reinforcement learning is a general approach for learning from interaction with a stochastic, unknown environment. RL has proven quite successful in handling large, realistic domains, by using function approximation techniques. However, RL algorithms using function approximation (FA) are still facing the embarrassment of limited generalization capability and slow convergence.

A major milestone in the course of RL development has been the proposal of neural-based implementations, which reduced learning time and memory requirements to realistic values, allowing true RL applications. In MDPs with large or continuous state spaces, value functions can be represented by function approximators. In this paper, the agent uses Cerebellar Model Articulation Controller (CMAC) neural network to approximate the Q-values for each action. The CMAC was proposed by Albus (1975) as a model of the information processing activities within the cerebellum. A CMAC has the advantage of having not only local generalization, but also being low in computation.

We adopt the coarse-coding structure where each point in input space excites a set of locally-tuned, overlapping and offset receptive fields in each dimension of the input vector. These receptive fields are defined by quantizing functions which operate on the input values. Hashing is used for reducing the storage requirements for the weights, whereby their locations are mapped into a smaller amount of physical memory, i.e. perform a mapping from the \( A^* \) set of weights to the \( A' \) set of physical memory locations.

CMAC requires defining the interval of discretization of the state space. Several quantizing functions in each input dimension are adopted. Our idea is: the closer to the goal, the higher resolution discretization of state space is adopted. The objective of this method is to make search to jump out of region near non-goal states and reach the optimum swiftly. By utilizing multi-resolution quantizing coding in the continuous state space, the value functions of MDP are approximated with variable resolutions so that the generalization ability of the CMAC-based RL controller can be improved and the search efficiency increases. All these ideas help much to achieve real time control.

5 Automatic Robot Shaping

Complex and uncertainty in modern robots and other autonomous systems make it difficult to design controllers for such systems that can achieve desired level of precision and robustness. A difficulty with reinforcement learning in complex tasks is that performance may converge slowly, or not at all. The problem is aggravated when only occasional (sparse) reinforcement is provided.

Shaping is a term used in behavior psychology to describe a process in which an animal is trained to perform a complex behavior in stages. The animal is first trained to perform a very simple task, and is then retrained to perform similar, although slightly more difficult, tasks in gradual degrees until the desired behavior is attained. A skill developmental process is also an open-ended cumulative process: A robot cannot learn complex skills successfully without first learning necessary simpler skills, e.g., without learning how to hold a pen, the robot will not be able to learn how to write. The strategy for complex skill learning follows the principle of divide-and-conquer in which a difficult problem is broken into several smaller and easier-to-solve sub-problems. Automatic robot shaping is introduced into the form of valuation function as:

\[
\text{val}(s,a) = W_1 Q(s,a) + W_2 \alpha Q(s,a) 
\]

Where \( W_1 \) and \( W_2 \) are weights that represent the amount of consideration that should be given to each value. \( Q(s,a) \) represents the current Q-value of state \( s \) and action \( a \), and \( \alpha Q(s,a) \) represents the Q-values learned in the primary task. \( W_2 \) should gradually decay to a small digital so that the subsequent policy is based mainly on the new Q-values. An elementary skill needs to be learned to solve each elementary task. This ensures that the robot will eventually learn the complex skill. The robot must be able to switch task swiftly when it find the situation appropriate. So the weights must be designed to
satisfaction of the requirement above mentioned.
The element skill k may switch from one to another very slowly in order to achieve \( k^* \) steadily. However, such a slow sequential composite learning cannot be desirable. To speed up the learning, the criterion should be modified. With the criterion, next motion begins without waiting for the completion of the previous motion to attain optimum. This is effective not only for the speed-up but also for smooth motion switching. The resultant behavior is smooth and speedy, i.e., dexterous.

6 Alternative Action Selection Policies

The \( \epsilon \)-greedy action selection is used because of its simplicity and often ensures a sufficient exploitation/exploration balance. A policy is called greedy with respect to some action value function \( Q(s, a) \) if in each state it selects one of the actions that have the maximum value:

\[
\pi(s, a) > 0 \quad \text{iff} \quad a \in \arg\max_{a \in A} Q(s, a) \quad (8)
\]

Upon trying to take a step to acquire the skill, we have an 80% chance of taking the greedy action towards the goal, and a 20% chance of a random action. However, this method has to face the occurrence that obviously hopeless actions are chosen again and again with the same probability as the more promising alternatives. To overcome the main drawback of this method, Boltzmann distribution for selecting actions is introduced to fix the above-mentioned problem in the random exploration. The action \( a \) in state \( s \) is selected with probability:

\[
P_\epsilon(a \mid s) = \frac{e^{Q(s, a)/\tau}}{\sum_{a \in A} e^{Q(s, a)/\tau}} \quad (9)
\]

Action selection based on both Boltzmann distribution and greedy selection was implemented, in which greedy selection policy is to maximize reward and Boltzmann distribution is then utilized to improve random exploration.

7 Examples

Our learning task is to learn to balance bicycle and reach target. Learning to ride a bicycle means to stay upright while on bicycle and then go to a goal. At each time step the agent receives information about the state of the bicycle, the angle and angular velocity of the handle bars, the angle, angular velocity and acceleration of the angle from the bicycle to vertical. For details of the bicycle system we refer to [1]. The agent chooses two basic actions. What torque should be applied to the handlebars, \( T \in \{-2N, 0N, +2N\} \), and how much the centre of mass should be displaced from the bicycle's plan, \( d \in \{-2cm, 0cm, +2cm\} \) — a total of 9 possible actions. Noise is laid on the choice of displacement, to simulate an imperfect balance, \( d = d_{\text{mean}} \cdot \text{choice} \cdot m \cdot p \), where \( p \) is a random number within \([-1, +1]\) and \( m \) is the noise level measured in centimeters. We use \( m = 2cm \). The bicycle starts out at the original heading west. The goal is a circular spot (10 meter radius) positioned 1000 meters to the north of the starting point.

Figure 2: Average number of seconds the robot can balance on the bicycle after 10 runs

Figure 3: Average route after 10 drives to the goal from scratch

Figure 2 illustrates the number of seconds the robot can keep bicycle from falling as incremental function of the number of trial. When it can keep upright for 1000 seconds, the task of balance is considered achieved. Figure 3 shows the route of riding bicycle to goal started
from scratch.

The requirement to approach the goal makes the task even harder, since the balancing and the goal-aiming are to be performed simultaneously. We argue that by decomposing the task into sub-tasks and task level, the overall rate of learning is increased compared to monolithic learners.

In the first experiments, the robot should first learn to keep bicycle from falling, and it was rewarded for riding towards the goal but was not punished for riding away from it. The robot learned to ride in tiny circles near the start state because no penalty was incurred for riding away from the goal with explicitly considering only finite-state domains. After robots have acquired the skill of balance, the loss of balance will be penalized. The dynamic criterion of actor critic offers more freedom for learning system acquiring more heuristic information in different stages of learning than stationary criterion.

Though it is still wandering around the optimum, bicycle is much closer to shortest path than results before.

8 Conclusions

Q learning algorithm based on the CMAC approximator is applied to learn the parameter of the bicycle control strategy. In particular, variable resolution discretization of input space not only improves the generalization ability of CMAC but also saves much search time. Automatic robot shaping is mainly to solve the complexity of learning. The augmented reward function offers immediate feedback information for improving the learning process. While augmented reward functions usually provide for quicker learning, problems include the possibility of local maxima and/or global maxima that do not correspond with optimal performance. Robots using local heuristic reward for training may not be able to optimize performance of the overall system. So the weight of heuristic information must be considered carefully.

All these ideas are implemented in the simulation on the robot learning control of bicycle. Simulation result shows the efficiency in complex skill learning. Though we only presented the simulation result for bicycle riding skill, the algorithm had also been shown to be applicable for other skills.

Acknowledgments

Sincerest thanks to Dr. ZHAO Ming-yang for his advice and contributions to this work. The authors gratefully acknowledge the support of Fund of Robotics Laboratory, Chinese Academy of Sciences in the completion of the reported work.

References


