DOCTORAL THESIS

Policy Abstraction for Transfer Learning Using Learning Vector Quantization in Reinforcement Learning Framework

Ahmad Afif bin Mohd Faudzi

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Engineering

in the

Graduate School of Information Science and Electrical Engineering
Department of Electrical and Electronic Engineering

August 2015

Contents

List of Figures

List of Tables

Abbreviations

RL	**R**einforcement **L**earning
LVQ	**L**earning **V**ector **Q**uantization
TL	**T**ransfer **L**earning

Symbols

α	learning rate
γ	discount factor
k	learning episode
t	time step
a_t	action at time step t
$A(s_t)$	set of possible actions
s_t	state at time step t
S	set of possible states
r_t	reward at time step t
π	policy
$Q(s, a)$	state-action value
$\max_a Q(s, a)$	maximum state-action value
ϵ	epsilon for ϵ-greedy
τ	temperature for Boltzmann selection
$\boldsymbol{x}$	input vector
$\boldsymbol{w}$	weight vector
D	distance between input and weight vector
o	output value

Chapter 1

Introduction

1.1 Research Background

People grow up every day exposed to the infinite state space environment interacting with active biological subjects and machines. There are routines that are always expected and unpredicted events that are not completely known beforehand as well. When people interact with the future routines, they do not require the same effort as they do during the first time. Based on experience, irrelevant information that does not affect the achievement is ignored. For example, a new worker in his/her first day will carefully recognize the road to his/her office, including the road's name, signboards, and buildings as well as focusing on the traffic. After several months he/she, possibly, will focus only on buildings and traffic. Furthermore, when people interact with an unpredicted event, they will usually try to cope with the situation using their knowledge that is acquired from their past experience. For example, an accident happened and the worker's daily route was jammed, here, he/she will try to find the alternate route based on the distance and the location of his/her office. This shows that people have an ability to benefit from their previous experience and knowledge for the future. Furthermore, the knowledge is not stored in a concrete or very detailed form, but in an abstract form that is ready to be used for routine events and also to be used for assisting in unknown events. Such abilities are obviously acquired through the most significant ability of a human being, which is learning ability from its successes and failures.

Since ancient times, in order to fulfill the people' requirement, machines are created. However, machines that are created to act based on a fixed set of rules are only limited to the designed environment i.e. factory machines. Since the real world is complex and unpredictable, it is very difficult to fix a set of rules for a machine to be involved in the real world. In order to cope with that, researchers are working on getting machines or

computers to act without being explicitly set up by the designer. This field is known as machine learning [1]. In machine learning, the learning machine is trained to generalize from its experience. Generalization in this context is the ability of a learning machine to perform well on already experienced and also new, unseen examples/tasks after having experienced a learning data set.

Today, machine learning is common in our life, it is being used daily e.g. license plates and traffic signs detection and recognition [2], practical speech recognition [3], medical imaging processing [4], and spam detection [5]. In the field of machine learning, there are a number of learning techniques including supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a training technique for a data sample from a data source whose correct classification is known. On the other hand, unsupervised learning is a training technique for an unlabeled data. Through this technique, the machine can learn without an error or a reward signal. The third technique, reinforcement learning (RL) is a training technique that enables the machine to take actions that maximize the reward by interacting with its environment. Reinforcement learning is different from the other techniques as its learning process involves interaction with its environment and it only depends on the reward signal provided by the interaction. In this research, reinforcement learning is considered.

1.2 Reinforcement Learning

Reinforcement learning is among the great learning frameworks that can train an agent to find an optimal solution to a problem where the best action in any given state is to be discovered [6]. The learning framework, which is based on iterative interactions with the environment by trial-and-error, enables RL to be applied to complicated or unknown environments. Reinforcement learning has been successfully applied in various problems, e.g. robotics [7], games [8], controls [9] and economics [10]. Figure 1.1 shows the summary of recent application using RL.

The ability to perform trial-and-error exploration may assist the agent to find the optimal solution, however, it also makes RL to take a long time to obtain a proper solution [12]. The more complex and larger the environment is, the more exploration RL requires, and it will consume more learning time or computation resources. Furthermore, if the environment changes, RL abandons past experiences and requires its agent to learn from scratch, which seems neither intelligent nor efficient. Moreover, a policy that determines an action for a certain state constructs its mapping of state-action pair separately. If the environment is large and complex, tremendous state-action pairs are constructed, which make it difficult to be interpreted.

Many studies have been done to improve RL methods that provide skills or prior knowledge to improve an agent's interaction with the environment such as Option [12] and Hierarchical RL [13], and they have been proven to enhance the learning process. Besides that, an agent can also benefit from their own past experiences, i.e., the knowledge obtained from solving earlier problems. The past experiences of similar tasks can help the agent to perform better exploration by providing useful guidance based on past tasks. In this research, transferring knowledge between tasks is considered to improve the learning acceleration RL.

1.3 Transfer Learning

Traditionally, most of RL algorithms are designed to treat isolated tasks. The idea of transfer learning is to change this by developing methods to train an agent in one or more new task(s) with guidance of knowledge or experience gained from one or more previous related but different task(s) [14, 15]. The aim of transfer learning is to improve learning in the new target task by leveraging the knowledge from the previous source task.

The evaluation of which transfer might improve learning can be indicated using three measures as mentioned by Torrey et al [14]. *First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent. Second is the amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratch. Third is the final performance level achievable in the target task compared to the final level without transfer.*

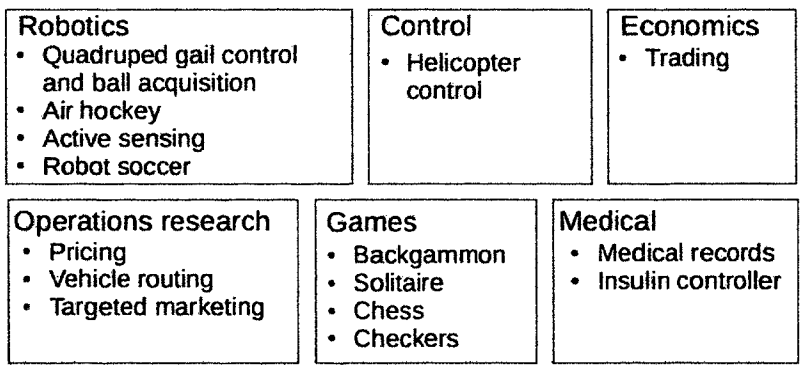


FIGURE 1.1: Summary of applications using reinforcement learning [11].

Several points need to be considered before performing transfer learning. First is the similarities between the source and the target task or how far they are allowed to differ. If both tasks are too different, transfer might decrease performance. Second is the representation and type of information transferred between the source and target tasks. Depending on task similarity, different types of knowledge may transfer better or worse. If tasks closely related, low-level information may be transferred, while high-level representation may transfer over less similar tasks. The types of information transferred across tasks in RL might be an action set, task features, action-value function or policy. In this research, policy is transferred between tasks.

1.3.1 Developed transfer learning methods for RL

Recently, the development of transfer learning (TL) methods in RL has been receiving great attention [14, 15]. Torrey and Shavlik [14] divide the transfer approaches in RL into several categories. Some categories and examples which are related to this research will be explained. The first category is *starting-point methods* which is to set the initial solution in a target task based on knowledge from a source task. Instead of random or zero setting, this method can start the learning at a point that receives higher reward. Bowling et al. use a starting-point method by initialize the target task with the final Q-values of the source task [16].

Next category is the *imitation methods* which use the learned source-task policy to guide the initial exploration of the agent in the target task. Instead of random exploration that RL algorithms normally do, the agent is guided to perform exploration more efficiently based on a source-task policy. Fernández and Veloso [17] use this approach by giving the agent a three-way choice between exploiting the current target-task policy, exploiting a source-task policy, and exploring randomly. A second parameter is introduced, in addition to the ϵ of ϵ-greedy exploration, to decide the probability of selecting each choice.

The third category is the alteration methods which involves altering the state space, action space, or reward function of the target task based on source-task knowledge. One of the techniques to alter the target-task state space is by state abstraction. Andre and Russell [18] do this by introducing three-part decomposition of value function that, for instance, can reduce data up to 85.4%. They then transfer to more complex target task, which performs significantly better than without transfer.

The transfer learning method in this research can be categorized in the second and the third categories.

1.4 Problem Statement and Proposed Learning Methods

1.4.1 Issues in transfer learning for reinforcement learning

In this research, the type of knowledge that is transferred between tasks is a policy; the rule that decides what action should be taken in certain states. A trained policy from the source task is expected to be provide appropriate actions for the states of target task which are related. Fernández et al. also transfer learned policy/policies across tasks and shows that a learned policy acquired from different but related problems can guide the agent better during the exploration of new tasks [17]. Nevertheless, the fact that we do not know for sure whether the transferred knowledge may or may not work in unknown different environments still needs to be considered.

Transfer learning has been applied to Q-learning [19], which is one of RL methods, by policy reuse such as initializing the Q-values of a target task with previously learned Q-values gained from a source task [16]. Q-value is an evaluation value regarding doing an action in a certain state. The transfer might improve the initial exploration. However, if the target task and the source task are significantly dissimilar, direct transfer of Q-values can cause *negative transfer* which is much worse than learning without policy reuse [20]. In order to perform the transfer learning efficiently, at least two issues need to be solved. The first is what the appropriate representation of transferred knowledge is, and the second is how to prepare for the target tasks whose environments are not completely known.

Regarding the first issue, through RL, even if we obtained the best policy for the source task, we do not know for sure that it will work with the target task. The policy might be incorrect for the target task, or its representation might not be so appropriate to be reused. If the old environment is small, and we use a lookup table to represent the policy, perhaps we manage to interpret and understand the learned policy. Even so, when the environment is large and complicated, it will be difficult. However, if we can simplify it by extracting some rules from the policy, it will be easier to understand and it will be possible to apply the policy to different tasks. These are good or even essential when we want to apply the learning system to real-world problems.

The representation also affects the size of transferred knowledge. For instance, in Q-learning, an optimal policy is obtained after the agent visited every possible state-action pair infinitely often and usually is represented as an explicit lookup table with a distinct table entry for every distinct state value. The state is a function of all environment features (state variable or dimension) and the size of the state-space grows exponentially with the dimensionality of the environment and the number of possible values in

each dimension. It is inefficient and not a realistic way to transfer a policy that has a tremendous data to be used in a new environment considering its load size and the relevance of all detailed data. We may reduce the state-space by using an approximation function like a neural network (NN) to represent the policy, but then it will suffer from the problem of 'black box' nature of the NN, and we cannot explain how it works, that is the first issue mentioned above.

On the second issue, since the target tasks' environments are not usually completely known and unpredicted, the transferred knowledge from the source task will never be perfect for the target task. There are two approaches that possibly treat this issue. One is adaptation. The agent changes itself so that it can work well in the changed environment. The agent changes itself after the change in the environment has actually happened. The other is precaution, which is a measure taken in advance for the agent so that it can work in all possible environments. The agent is prepared for all the possible changes in the environment before the changes actually occur. In this research, both approaches are considered.

1.4.2 Proposed solutions

In this research, transfer learning is considered in order to improve the learning speed in RL, and the type of knowledge that is transferred across the tasks is policy. In order to perform policy-transfer efficiently, the need for appropriate representation and learning methods are presented in this research. Therefore in this research, policy abstraction is proposed, as a core of learning methods, that generates appropriate representation of transferred knowledge and enables transfer learning to work in partially known or unpredicted target tasks.

Abstraction

Abstraction is an operation that changes the representation of an object by removing less critical details while preserving desirable properties [21]. In RL, state-space reduction is one of the popular techniques to accelerate learning. Most of conventional methods perform state-space reduction by focusing on the similarity between the states [22, 23]. In this research, in addition to this, the ideal behavior is also taken into account during abstraction.

In this research, as shown in Figure 1.2, abstraction is performed on the learned policy that was trained by RL. The proposed abstraction method not only classify the states that have the same ideal actions to their classes but also constructing subclasses based

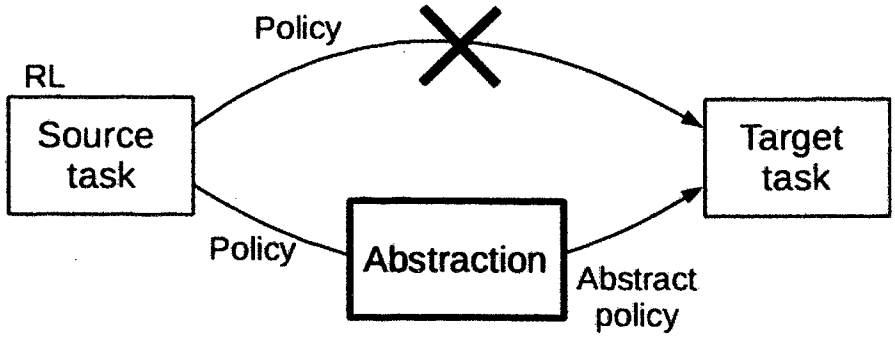


FIGURE 1.2: Policy abstraction for transfer learning.

on the similarities of the states. A modified learning vector quantization (LVQ) that can autonomously increase or decrease, as required, the number of its network neurons is proposed to perform this classification. Francisco et al proposed VQQL, which combines vector quantization and Q-learning but they focus more on state space generalization [24]. The generated groups with ideal actions that represent the extracted rules from original learned policy is called abstract policy.

The application of abstract policy is expected to perform as good as the ordinary policy while it simple representation provide fewer data and enables it to be interpreted by human designer.

Adaptation

Abstract policy that is obtained in certain source tasks is not always useful for related target tasks. While applying abstract policy, adaptation ability is required for agent to change the abstract policy in order to fit to target tasks. Adaptation is an operation that enables the agent to change itself so that it can work well after the environment has actually changed. Rajendran and Bergamo proposed abstract policy learning and reused the abstract policy to improve initial performance of an RL learner in a similar new problem [25–27]. Both studies showed good results in terms of the learning acceleration and state space reduction. However, they did not consider any other environments and interpretation. The modified LVQ algorithm is extended to perform adaptation while performing abstraction. In addition to this, a non-liner transformation of the reward is proposed to treat the difficulties to achieve the optimal behavior by limiting the training responding to small reward so that it can prevent the learning system suffer from an undesirable effect by small reward. Figure 1.3 illustrates the adaptation.

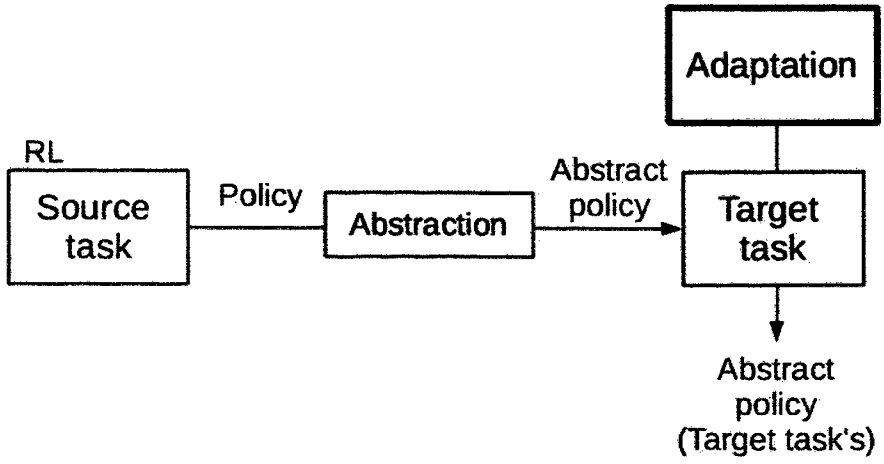


FIGURE 1.3: Adaptation for transfer learning.

Precaution

The precaution is an operation that prepares the agent before the environment actually changes. In order to improve the possibility to use the abstract policy with more target tasks, a more generally applicable abstract policy, common abstract policy is derived. The advantage of the common abstract policy compared to a policy or an abstract policy is that it has higher generality and that more importantly, it only holds the common information of past similar environments and thus can support the learning agent better in similar target tasks. As shown in Figure 1.4, the common abstract policy is generated using the proposed LVQ by finding and extracting the similarities in the previously obtained policies.

Koga et al proposed a similar stochastic abstract policy, a single set of policies, that represent the extracted similarities of a number of past experiences [28]. However, there are still some room left for improvement in the aspect of the extraction of similarities among tasks and the representation of abstract policy.

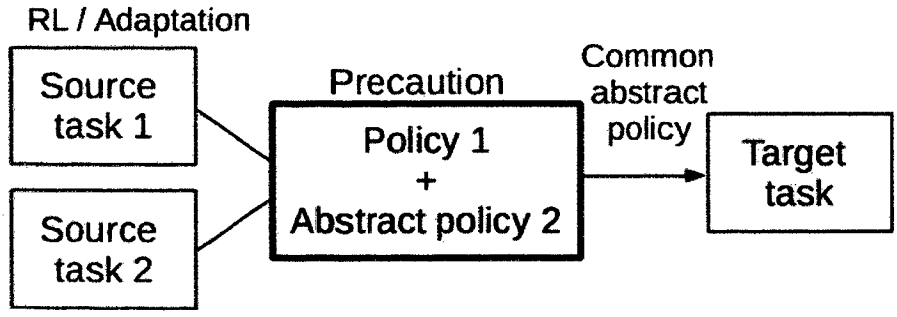


FIGURE 1.4: A precaution for transfer learning.

In this research, in order to verify the proposed methods, several simulations are performed. A three-dimension maze problem with a camera-mounted agent to move from a random start state to the goal state while avoiding obstacles is designed, and several related environments are prepared. The results show that the representation of acquired abstract policy is interpretable and the use of abstract policy for transfer learning accelerate the learning.

1.5 Research Objectives

The objectives of this research are summarized as:

i To study and develop an abstraction method for transfer learning in order to improve training in reinforcement learning in terms of learning speed and interpretability of policies.

ii To treat and resolve the issues that arise when transfer learning is performed for reinforcement learning problems, which are the appropriate representation of transferred knowledge, and how to prepare for the target tasks whose environments are not completely known.

1.6 Contributions

The direct contributions of this research work are summarized as:

i Policy abstraction that considers not only state value, but also the ideal behavior is proposed to extract an abstract policy from an ordinary policy.

ii A modified learning vector quantization (LVQ) algorithm is proposed to perform abstraction that can autonomously add and delete, as required, the number of its network neurons.

iii Adaptation is proposed to enables the agent to adapt to the new task while performing abstraction.

iv An extended modified learning vector quantization (LVQ) algorithm is proposed that introduces a non-liner transformation of the reward to improve the adaptation by limiting the training responding to small reward behavior so that it can prevent the learning system suffer from an undesirable effect by small reward behaviors.

v A precaution is proposed that introduces a more generally applicable abstract policy, common abstract policy in order to improve the possibility to use the abstract policy with more target tasks.

1.7 Overview of the Thesis

This thesis consists of six chapters. The outline of the thesis from the next chapter is presented as follows:

Chapter 2 introduces the background and the general theories of reinforcement learning which is the main framework in this thesis. The learning framework of RL is also explained. This chapter also introduces the learning vector quantization algorithm which is the core of the proposed solutions for the issues mentioned in the Chapter 1.

Chapter 3 proposes a modified LVQ algorithm with a dynamic network structure as an abstraction method to solve the first issue, the appropriate representation of transferred knowledge. The abstraction is performed by extracting an abstract policy out of a learned policy that was prepared by the RL conventional method, Q-learning. The LVQ network that can dynamically add and delete its weight vectors that represent the abstract policy is expected to generate an appropriate representation of transferred knowledge in terms of the transferred data size and also whether the transferred data is interpretable or not.

Chapter 4 introduces an extended modified LVQ algorithm to work in the RL framework for the adaptation and the precaution as solutions for the second issue which is how to prepare for the new tasks whose environments are not completely known or predicted. The LVQ algorithm proposed in the previous chapter is extended so that it can make the agent adapt to the new task while performing abstraction. The precaution is realized by deriving a more generally applicable abstract policy, common abstract policy, by finding and extracting the similarities in the previously obtained policy and the current one.

Chapter 5 proposes another modification of the LVQ algorithm to improve the adaptation operation in Chapter 4. The difficulties for the LVQ algorithm to work in RL framework are discussed. As a solution, two modifications are introduced. The first is a new way to define the reward value that is calculated by the agent autonomously based on its behavior, and the second is a function that converts the defined reward to another reward before it is actually used during learning. This function limits the training in response to small rewards so that it can prevent the learning system suffer from an undesirable effect by small reward behaviors.

Chapter 6 gives general conclusions and recommendations for further research.

Chapter 2

Reinforcement Learning and Learning Vector Quantization

The first section in this chapter introduces the history and the framework of reinforcement learning (RL). Q-learning, one of its famous learning method that is used in this research is described as well. In the second section, Learning Vector Quantization (LVQ) which is the proposed method in this research is introduced.

2.1 Reinforcement Learning

2.1.1 History of reinforcement learning

Sutton and Barton describes the history of RL in their book very detailed [6]. Next are the summary of it. The history of reinforcement learning has three main threads; one thread concerns the problem of optimal control and its solution using value functions and dynamic programming, the second thread concerns learning by trial and error and started in the psychology of animal learning and the last thread concerns temporal-difference learning. These threads came together in the late 1980s to produce the present modern field of reinforcement learning.

In the first thread, in the mid-1950s, Richard Bellman developed an approach for solving optimal control problems that now often called the Bellman equation. He also introduced the discrete stochastic version of the optimal control problem known as Markovian decision processes (MDPs), and Ron Howard in 1960 devised the policy iteration method for MDPs. Dynamic programming is widely considered as the only feasible way of solving

general stochastic optimal control problems. Even it suffers from 'the curse of dimensionality', it is still far more efficient and more widely applicable than any other general method.

The other thread that centered on the idea of trial-and-error learning, began in psychology, where 'reinforcement' theories of learning are common. Thorndike in 2011, called the idea of this learning as the 'Law of Effect' that includes two most important aspects; first, it is *selectional*, meaning that it involves trying alternatives and selecting among them by comparing their consequences. Second, it is *associative*, meaning that the alternatives found by selection are associated with particular situations. In 1954, the earliest computational investigations of trial-and-error learning were done, by Minsky and by Farley and Clark. However, Farley and Clark shifted from trial-and-error learning to generalization and pattern recognition, that is, from reinforcement learning to supervised learning. Harry Klopf then revived back the trial-and-error thread to reinforcement learning within artificial intelligence before it influenced Barto and Sutton.

The final thread is concerning temporal-difference learning. In 1972, Klopf brought trial-and-error learning together with an important component of temporal-difference learning. Klopf also linked the idea with trial-and-error learning and related it to the massive empirical database of animal learning psychology. Sutton and Barto refined these ideas and developed a psychological model of classical conditioning based on temporal-difference learning. In 1997, a recent summary of the links between temporal-difference learning and neuroscience ideas is provided by Schultz, Dayan, and Montague. In 1981, Sutton and Barto developed a method for using temporal-difference learning in trial-and-error learning, known as the actor-critic architecture. A key step was taken by Sutton in 1988 by separating temporal-difference learning from control, treating it as a general prediction method.

Finally, the temporal-difference and optimal control threads were fully brought together in 1989 with Chris Watkins's development of Q-learning.

2.1.2 Reinforcement learning framework

Reinforcement learning (RL) [6] is an adaptive learning framework for the problem of learning from interaction to achieve a goal. In RL, the *agent* is the learner and decision-maker and the *environment* is the everything outside the agent that the agent interacts with. During interaction, the agent selecting actions and the environment responding to those actions. New situations are perceived by the agent as the results of the executed actions. The environment also gives rise to rewards, special numerical values that the

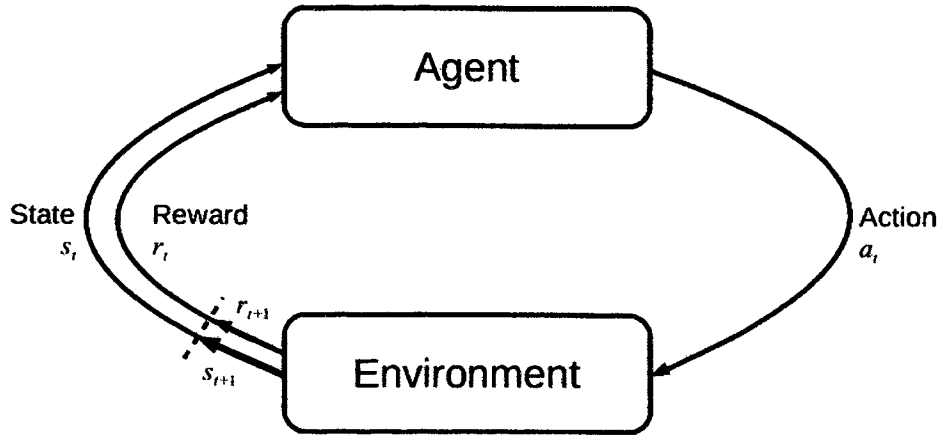


FIGURE 2.1: The agent-environment interaction in reinforcement learning.

agent tries to maximize over time. A complete specification of an environment defines a *task*, one instance of the reinforcement learning problem.

Figure 2.1 shows the agent-environment interaction. The agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, \ldots$. At each time step t, the agent receives some representation of the environment's *state* $s_t \in S$, where S is the set of possible states, and on that basis selects an *action* $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t. After one time step, the agent receives a numerical *reward* $r_{t+1} \in R$ and finds itself in a new state, s_{t+1}. At each time step, the agent updates its *policy* π_t, a mapping from states to probabilities of selecting each possible action, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, is to maximize the total amount of reward it receives over the long run [6].

This framework can be applied to many different problems in many different ways. For example, the actions can be low-level controls, such as the voltages applied to the motors of movable camera [29], or high-level decisions, such as producing a recognition result [30]. Similarly, the states can be completely determined by low-level sensations, such as direct sensor readings, or they can be more high-level and abstract, such as symbolic descriptions of objects in a room. In general, actions can be any decisions we want to learn how to make, and the states can be anything we can know that might be useful in making them. Anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment. Sometimes, not everything in the environment is unknown to the agent. The reinforcement learning framework is a considerable abstraction of the problem of goal-directed learning from interaction. It proposes that any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment: one signal to

represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states), and one signal to define the agent's goal (the rewards).

2.1.3 Q-learning

Q-learning [6, 19] is one of the commonly used reinforcement learning techniques. It has been proven that for an y finite Markov decision process (MDP), Q-learning is reliable to be used to find an optimal action-selection policy [19]. Q-learning operates by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy afterwards. A policy is a rule that determines the agent action in the state the agent is in. Q-learning is an off-policy learning technique, which means during learning the agent may take an action that is different from the action determined by the learned policy. After the training complete, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the advantages of Q-learning is that it is a model-free technique that is able to compare the expected utility of the available actions without requiring a model of the environment.

Algorithm

The MDP problem model contains an agent, a finite set of states S and a finite set of states A. The agent can move from a state $s \in S$ to another state by performing an action $a \in A$. Performing an action in a specific state provides the agent with a reward which is a numerical value. The goal of the agent is to maximize its total reward. In each state, the agent learns which action is the optimal one that has the highest long-term reward. All available state-action pairs are evaluated, and the evaluation value is called a Q-value.

The flow chart of the Q-learning algorithm is as shown in the Figure 2.2. Before learning has started, Q returns a value initialized by the designer. At each step of time step t, the agent observes the state s_t, then chooses and performs an action a_t. As the process moves to state s_{t+1}, the agent receives a reward or punishment r_{t+1}. Then, $Q(s_t, a_t)$ is updated. The update function of the Q-learning algorithm is defined by Eq. (2.1). It adopts the old value and makes a correction based on the new information.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\{r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\} \tag{2.1}$$

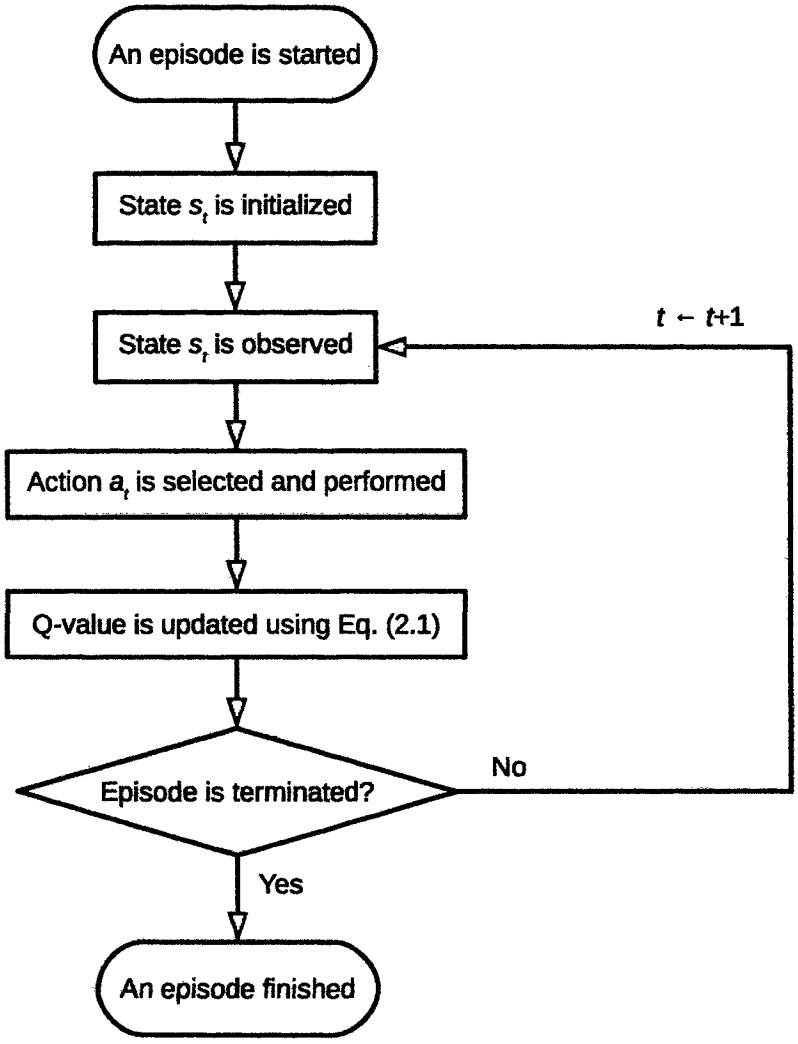


FIGURE 2.2: Flowchart of Q-learning algorithm in one episode.

Chapter 1

Introduction

1.1 Research Background

People grow up every day exposed to the infinite state space environment interacting with active biological subjects and machines. There are routines that are always expected and unpredicted events that are not completely known beforehand as well. When people interact with the future routines, they do not require the same effort as they do during the first time. Based on experience, irrelevant information that does not affect the achievement is ignored. For example, a new worker in his/her first day will carefully recognize the road to his/her office, including the road's name, signboards, and buildings as well as focusing on the traffic. After several months he/she, possibly, will focus only on buildings and traffic. Furthermore, when people interact with an unpredicted event, they will usually try to cope with the situation using their knowledge that is acquired from their past experience. For example, an accident happened and the worker's daily route was jammed, here, he/she will try to find the alternate route based on the distance and the location of his/her office. This shows that people have an ability to benefit from their previous experience and knowledge for the future. Furthermore, the knowledge is not stored in a concrete or very detailed form, but in an abstract form that is ready to be used for routine events and also to be used for assisting in unknown events. Such abilities are obviously acquired through the most significant ability of a human being, which is learning ability from its successes and failures.

Since ancient times, in order to fulfill the people' requirement, machines are created. However, machines that are created to act based on a fixed set of rules are only limited to the designed environment i.e. factory machines. Since the real world is complex and unpredictable, it is very difficult to fix a set of rules for a machine to be involved in the real world. In order to cope with that, researchers are working on getting machines or

computers to act without being explicitly set up by the designer. This field is known as machine learning [1]. In machine learning, the learning machine is trained to generalize from its experience. Generalization in this context is the ability of a learning machine to perform well on already experienced and also new, unseen examples/tasks after having experienced a learning data set.

Today, machine learning is common in our life, it is being used daily e.g. license plates and traffic signs detection and recognition [2], practical speech recognition [3], medical imaging processing [4], and spam detection [5]. In the field of machine learning, there are a number of learning techniques including supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a training technique for a data sample from a data source whose correct classification is known. On the other hand, unsupervised learning is a training technique for an unlabeled data. Through this technique, the machine can learn without an error or a reward signal. The third technique, reinforcement learning (RL) is a training technique that enables the machine to take actions that maximize the reward by interacting with its environment. Reinforcement learning is different from the other techniques as its learning process involves interaction with its environment and it only depends on the reward signal provided by the interaction. In this research, reinforcement learning is considered.

1.2 Reinforcement Learning

Reinforcement learning is among the great learning frameworks that can train an agent to find an optimal solution to a problem where the best action in any given state is to be discovered [6]. The learning framework, which is based on iterative interactions with the environment by trial-and-error, enables RL to be applied to complicated or unknown environments. Reinforcement learning has been successfully applied in various problems, e.g. robotics [7], games [8], controls [9] and economics [10]. Figure 1.1 shows the summary of recent application using RL.

The ability to perform trial-and-error exploration may assist the agent to find the optimal solution, however, it also makes RL to take a long time to obtain a proper solution [12]. The more complex and larger the environment is, the more exploration RL requires, and it will consume more learning time or computation resources. Furthermore, if the environment changes, RL abandons past experiences and requires its agent to learn from scratch, which seems neither intelligent nor efficient. Moreover, a policy that determines an action for a certain state constructs its mapping of state-action pair separately. If the environment is large and complex, tremendous state-action pairs are constructed, which make it difficult to be interpreted.

Many studies have been done to improve RL methods that provide skills or prior knowledge to improve an agent's interaction with the environment such as Option [12] and Hierarchical RL [13], and they have been proven to enhance the learning process. Besides that, an agent can also benefit from their own past experiences, i.e., the knowledge obtained from solving earlier problems. The past experiences of similar tasks can help the agent to perform better exploration by providing useful guidance based on past tasks. In this research, transferring knowledge between tasks is considered to improve the learning acceleration RL.

1.3 Transfer Learning

Traditionally, most of RL algorithms are designed to treat isolated tasks. The idea of transfer learning is to change this by developing methods to train an agent in one or more new task(s) with guidance of knowledge or experience gained from one or more previous related but different task(s) [14, 15]. The aim of transfer learning is to improve learning in the new target task by leveraging the knowledge from the previous source task.

The evaluation of which transfer might improve learning can be indicated using three measures as mentioned by Torrey et al [14]. *First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent. Second is the amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratch. Third is the final performance level achievable in the target task compared to the final level without transfer.*

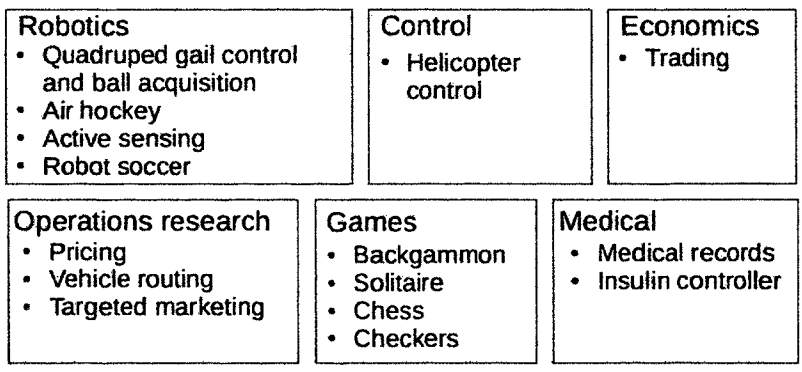


FIGURE 1.1: Summary of applications using reinforcement learning [11].

Chapter 3

The Abstraction

This chapter discussed the first issue that is highlighted in this research when we want to perform learning which is what the appropriate representation of transferred knowledge is. The type of knowledge that is transferred between source and target tasks in this research is policy. An abstract policy that is extracted from a learned policy of a source task by abstraction process is introduced as transferred knowledge. A modified learning vector quantization (LVQ) algorithm that can autonomously increase or decrease as required the number of its network neurons is proposed to perform this abstraction. Simulation results show that the transferred knowledge represented by abstract policy has fewer data and simple enough to be interpreted and that the transfer successfully improves the learning in the target task.

3.1 Introduction

In order to guarantee the efficiency of transfer learning, one of the factors that need to be considered is what the appropriate representation of transferred knowledge is [15]. The representation can range from very low-level information about a specific task to general heuristics that attempt to guide learning. Depending on how similar the tasks, different representation of may cause positive or negative transfer. For example, low-level information may transfer across closely related tasks, while high-level concepts may transfer across pairs of less similar tasks. Based on the type of transfer knowledge and how similar the source and target tasks, an appropriate representation is need to be found.

In this research, a high-level representation of transferred knowledge is proposed. It is expected to work as good as the low-level representation for closely related tasks

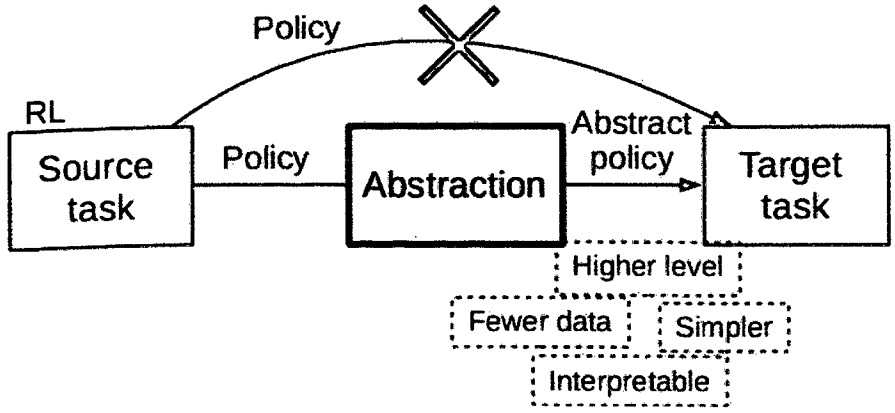


FIGURE 3.1: Policy abstraction for transfer learning.

and better for less similar tasks. As shown in Figure 3.1, instead of transferring the ordinary learned policy, rules that extracted through abstraction from the learned policy is transferred. The generated rules are expected to have fewer data compared to ordinary policy and simple enough to be interpreted. In this research, a modified learning vector quantization (LVQ) algorithm that can autonomously add or delete its network neurons is proposed to perform this abstraction and extracted rules is called abstract policy.

If the source and target tasks are very different, transfer learning might not be work so well. While allowing transfer to happen between less similar source and target tasks gives more flexibility to a designer. In this research, the tasks are similar in the terms of the tasks' objective, action set, environment objects and the number of state's variables. While tasks are different in the terms of possible states caused by the different settings of environment objects.

The rest of this chapter is organized as follows. In the Section 3.2, the idea behind the abstraction and the proposed LVQ algorithm is described. Then, it follows by the detailed explanation of the LVQ algorithm. The simulations and their results are explained in Section 3.4. Finally, Section 3.5 states the summary.

In simulation, a 3-D maze problem with a camera-mounted agent is employed. The agent is trained to move from the start state towards the goal state by avoiding some obstacles. The results show that the abstraction is successful and the abstract policy represented by weight vectors is simple and easy to interpret.

The rest of this chapter is organized as follows. In the Section 3.2, the issues and solution are described. Then, it follows by the detailed explanation of the proposed algorithm that were used in this paper. The simulations and their results are explained in Section 3.4. Finally, Section 3.5 states the summary.

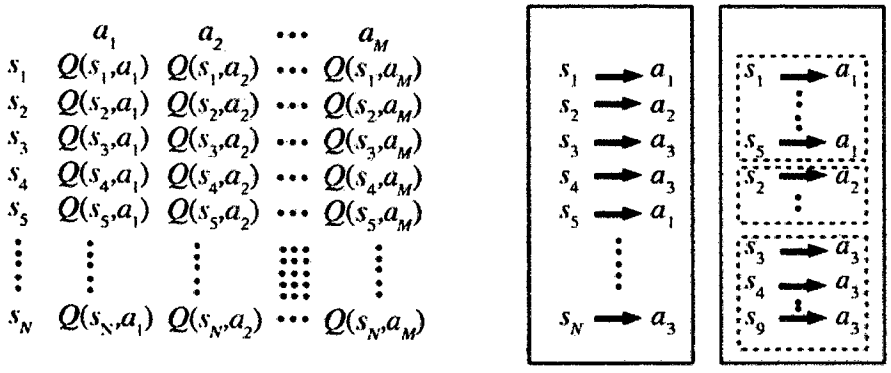
3.2 Learning Vector Quantization (LVQ) as an Abstraction Method

In this chapter, the source task is trained using Q-learning which is one of the conventional RL methods. After the training completed, a learned policy is obtained. Here, instead of being transfer directly to the target task, the learned policy is abstracted using a modified LVQ and abstract policy is generated.

Q-learning uses lookup table to represent its policy which is actually the state-action values. As shown in Figure 3.2a, the table size is $N \times M$, where N is the the number of different possible states, and M is the number of different possible actions. During action selection for a certain state, agent refers to the table and lookup corresponding action values for that state, and choose the maximum. The agent will evaluate the state-action values repeatedly or in other words update its policy during learning. As illustrated in Figure 3.2b, after learning complete, the agent will have a learned policy that has the ideal actions for all possible states. It is also means that after the learning finished, all possible states are classified to the number of actions set, e.g. three classes.

Abstraction is an operation that reduce the complexity of a problem by ignoring irrelevant properties while preserving all the important ones necessary to still be able solve a given problem. In this research, the abstraction is performed by grouping different states that correspond same actions. As shown in Figure 3.3, all states that are classified to M classes by Q-learning are re-classified by abstraction to several more subclasses. Each subclass has its own ideal action.

The classification is performed based on the states' values and ideal actions. Here, during abstraction, the essential properties that are preserved are the ideal actions, while the less relevant information are the states' values. The states that are close to each other and



(A) During training (B) After training

FIGURE 3.2: Policy representation using lookup table