Transfer learning via linear multi-variable mapping under reinforcement learning framework

Conference Paper · July 2017

DOI: 10.23919/ChiCC.2017.8028754

3 authors, including:

Qiao Cheng
National University of Defense Technology
12 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Xiangke Wang
National University of Defense Technology
94 PUBLICATIONS **780** CITATIONS
SEE PROFILE

Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

Transfer Learning via Linear Multi-variable Mapping under Reinforcement Learning Framework

Qiao Cheng, Xiangke Wang, Lincheng Shen
College of Mechatronics and Automation,
National University of Defense Technology,
Changsha, Hunan Province,410073, P. R. China
E-mail: {qiao.cheng,xkwang,lcshen}@nudt.edu.cn

Abstract: Though popular in many agent learning tasks, reinforcement learning still faces problems, such as long learning time in complex environment. Transfer learning could shorten the learning time and improve the performance in reinforcement learning by reusing the knowledge acquired from different but related source task. Due to the difference in state space and/or action space of the target and source task, transfer via inter-task mapping is a popular method. The design of the inter-task mapping is very critical to this transfer learning method. In this paper, we propose a linear multi-variable mapping (LMVM) for the transfer learning to make a better use of the knowledge learned from the source task. Unlike the inter-task mapping used before, the LMVM is not a one-to-one mapping but a one-to-many mapping, which is based on the idea that the element in target task is related with several similar elements from source task. We test transfer learning via our new mapping on the Keepaway platform. The experimental results show that our method could make the reinforcement learning agents learn much faster than those without transfer and those transfer with simpler mappings.

Key Words: Transfer Learning, Reinforcement Learning, Linear Multi-variable Mapping, Inter-task Mapping, Keepaway

1 Introduction

Reinforcement learning (RL)[1] is gaining more and more attention in artificial intelligent area. It works well in helping agents to solve sequential decision making problems. However, there are a lot of problems remaining in this area. For example, it usually takes too much time for the RL agents to converge to a good policy, especially in some large or complex environments. This gives rise to a lot of techniques which speedup the reinforcement learning process. Transfer learning (TL) [2] is one of these techniques. The main idea of transfer learning is that the knowledge acquired from related source tasks could provide some prior for the target task to speedup the learning and/or improve the final performance in reinforcement learning.

The idea behind transfer learning is inspired from the usual way that we human been solve new problems. When facing a new problem, we could use experience learned before to solve it. Similarly, agents could also benefit from experience of both themselves and other agents. Usually, the agent uses information learned from a simpler source task to speedup the learning in a more complex target task. Thus, the state space and/or action space of the target task is different from that of source task. A popular method called transfer via inter-task mapping [3] uses mapping of the state space and/or the action space between the source task and the target task to enable the transfer learning on task with action-value function RL. Therefore, the inter-task mapping is very important to the success of the transfer learning.

There are many methods about how to do the mapping from the source task to the target task. Traditionally, the inter-task mapping provides every state feature or action in the target task a correspondent state feature or action in the source task. However, the number of feature in the target task usually bigger than that of the source target. Hence, some different features in the target task have to map from the same feature in the source task. Apart from this, many simple mappings don't take the relationship among different features into consideration. Therefore, we contribute by proposing a linear multi-variable mapping (LMVM) for the transfer learning, which takes the former two problems into consideration at the same time.

The rest of the paper is organized as follows. Related work is presented in the next section. Background is given in Section 3. Our linear multi-variable mapping for transfer learning is described in Section 4. Section 5 details the experiments using the new mapping for transfer learning in Keepaway platform. In section 6, conclusion and future works are discussed.

2 Related Works

As a technique to speedup the learning in reinforcement learning process, transfer learning has attracted many attention. There are many methods to transfer knowledge from source task to the target task, one of which is through inter-task mapping. Most of the inter-task mapping is pre-defined by experts according to their experience or intuition. Taylor et. al proposed transfer via inter-task mapping (TVITM) in [3] to speedup the learning on task with action-value function RL. They use a pair of inter-task mapping (state mapping χ_X and action mapping χ_A) to represent the relationship between two tasks. Based on these two mappings, they construct the transfer function so as to initialize the action-value function Q_{target} of the target task by Q_{source}. The mapping simply maps similar feature from the source task to the target task without any transition, and some features in the target task are mapped from the same feature in the source task.

Instead of using simple pre-defined mapping, there are some researchers design mechanisms to select good mappings. Anestis et.al [4] propose two algorithms for both model-learning and model-free reinforcement learning algo-

This work is supported by National Natural Science Foundation (NNSF) of China under Grant 61403406 and 61403410.

rithms to transfer from multiple inter-task mappings. The main job is about mechanisms to select the best mapping from multiple mappings, not about how to design a mapping. In another of their paper [5], they propose a method which autonomously selects mappings based on an on-line probabilistic evaluation from the set of all possible inter-task mappings.

There are also some methods for learning inter-task mapping automatically. Taylor et. al [6] propose a method called Modeling Approximate State Transitions by Exploiting Regression (MASTER). It automatically learn a mapping from one task to another by using experience gathered from task environments. However, this method is very computationally expensive. Haitham B. Ammar et. al [7] proposed a framework to learn inter-task mappings between different MDPs automatically by an adaptation of restricted Boltzmann machines. This methods makes the inter-task mapping could be learned instead of hand-coded by human experts, but it still need a lot of computations.

However, there are also methods that use other ways to express the inter-task mappings. George et.al [8] propose a framework for transfer in reinforcement learning via shared features, which is based on the idea that related tasks share some common features. Thus, in their transfer learning methods, no direct mapping from the source task features to target task is needed. The inter-task mapping is conveyed by the shared features.

3 Background

3.1 Reinforcement Learning

Reinforcement learning (RL) is learning to make sequential decision about how to map states to actions so as to maximize a numberical reward signal [1]. An RL problem is typically formalized as a Markov Decision Process (MDP) $< S, A, T, R >$, where S is the set of environment states, A is the set of actions that the agent may execute. $T : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function. $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function which measures the performance of the agent. When in the environment state $s \in S$, the agent takes an action $a \in A$, then the environment state transits into state $s' \in S$ with a probability of $T(s, a, s')$. An immediate reward $R(s, a, s')$ will be given to the agent. A policy $\pi : S \times A \times S \rightarrow [0, 1]$ defines how the agent choose action in each state, where the probability of selecting action a in state s is $\pi(s, a)$. The agent needs to improve its policy so as to reach the optimal policy π^*, which should satisfy optimal state-action function (Q-function) as in Equation (1), where $\gamma \in [0, 1)$ is the discount factor.

$$Q^*(s, a) = \max_{\pi} E[\sum_{k=0}^{\infty} \gamma^k r_{k+1} | s_0 = s, a_0 = a, \pi] \quad (1)$$

When the state and/or action space is continuous, it is not suitable to represent the Q-functions and/or policies in a table format, so sampling or function approximation techniques are needed. There are some popular function approximations, such as Cerebellar Model Articulator Controller (CMAC), Radial Basis Functions (RBF), Artificial Neural Networks (ANN). These function approximators take in the state and action, and give the estimated value function.

3.2 Transfer Learning

Transfer learning (TL) use knowledge acquired from the source task to improve learning in the target task. Typically, the source task and the target task are different but related. As the RL agent learns by exploring the environment, the related knowledge from source task which serves as a prior could help to smaller the scope of exploration or set a better start point for the exploration in target task, therefore it will learn faster than just exploring randomly in environment as in RL without transfer.

Since the source task and target task are different, they may differ in their state spaces and/or action spaces. To get successful transfer, the inter-task mapping is needed. There are many ways to have the inter-task mapping for transfer learning. As detailed in Section 2, methods for selecting best inter-task mapping and learning inter-task mapping exist already. However, our work is going to focus on the designing of the inter-task mapping. A typical inter-task mapping is given in [3], which is the basis of our work.

In [3], the value function is approximated by the weights of tiles in CMAC. The transfer learning is to copy the weights of tiles from source task to initialize the tile weights in the target task, while the initial weights in target task are all set to 0 when no transfer is done. However, the target task has more state variables and actions. Thus, the inter-task mapping is used to map appropriate tile weights in the source task to the tiles in the target task. The CMAC takes a state vector and an action, then returns the activated tiles. Thus, tile weights in source task could be copied to tiles that activated by a similar state variable and action in target task as in source task. After all these, the knowledge from source task will set a prior for the target task.

3.3 Keepaway Platform

Keepaway soccer platform [9] is a subproblem of Robocup simulated soccer, which is a popular testbed for machine learning. There are two teams in the platform, namely keepers and takers. The keepers try to maintain possession of the ball within a limited region, while the takers try to gain possession of the ball from the keepers or kick it out of bounds. Figure (1) is an screenshot for a 4vs.3 keepaway task scenario. Let n and m denotes the number of keepers and takers, respectively. $K_1, ..., K_n$ are keepers ordered by the closeness to the ball. K_1 is the keeper with the ball, K_2 is the closest keeper to K_1, K_3 the next closest, and so on up to K_n. Similarly, $T_1, ..., T_m$ are takers ordered by the closeness to the K_1. The keeper with the ball choose actions from the action set where the number of actions is n. The action space includes actions {Hold, PassToKeeperK_2,...., PassToKeeperK_n}.

The state space is composed of distance variables and angle variables. Let C denotes the center of the playing region, $dist(a, b)$ denotes the distance between a and b, and $ang(a, b, c)$ denotes the angle between a and c with vertex at b. The state space includes six types of variables with a total number of $(4n+2m-3)$. The six types of state variables are listed below.

- $dist(K_i, C)$, $(i \in [1, n])$. These n variables are the distances from the keeper to the center.
- $dist(T_j, C)$, $(j \in [1, m])$. These m variables are the

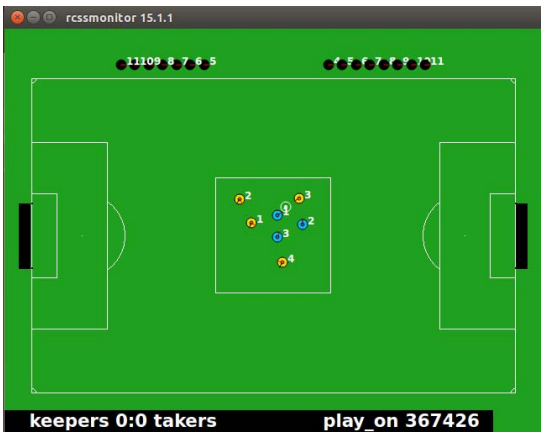

Fig. 1: 4vs.3 Keepaway scenario

distances from the taker to the center.

- $dist(K_1, K_i)$, $(i \in [2, n])$. These $(n-1)$ variables are the distances from the keeper with ball to other keepers.
- $dist(K_1, T_j)$, $(j \in [1, m])$. These m variables are the distances from the keeper with ball to takers.
- $dist(K_i, T_w)$, $(i \in [2, n], w = \arg\min_{j \in [1,m]} dist(K_i, T_j))$. These $(n-1)$ variables are the minimal distances from the keeper without the ball to takers.
- $ang(K_i, K_1, T_w)$, $(i \in [2, n], w = \arg\min_{j \in [1,m]} ang(K_i, K_1, T_j))$. These $(n-1)$ variables are the minimum angles between the keeper without ball (K_i) and the takers (T_w) with vertex at the keeper with ball (K_1).

4 Transfer via Linear Multi-variable Mapping

4.1 Transfer via Inter-task Mapping

Transfer via inter-task mapping (TVITM) [3] is defined for value function reinforcement learners. Given the source task: $(S^{(s)}, s_0^{(s)}, A^{(s)}, T^{(s)}, R^{(s)}, Q_0^{(s)}) \Rightarrow Q_{final}^{(s)}$ and the target task: $(S^{(t)}, s_0^{(t)}, A^{(t)}, T^{(t)}, R^{(t)}, Q_0^{(t)}) \Rightarrow Q_{final}^{(t)}$, where $S^{(s)} \neq S^{(t)}$ and/or $A^{(s)} \neq A^{(t)}$. In TVITM, we let $Q_0^{(t)} = \rho(Q_{final}^{(s)})$, where $\rho(Q)$ is the transfer function and is built from the inter-task mappings. Although $S^{(s)} \neq S^{(t)}$ and/or $A^{(s)} \neq A^{(t)}$, the source task and target task should be related so as to succeed in transfer learning. The relations could be represented as a pair of inter-task mappings, denoted χ_S and χ_A. The state mapping χ_S maps each state variable in the target task to the most similar state variable in the source task:

$$\chi_S(s_i^{(t)}) = s_j^{(s)} \tag{2}$$

Similarly, the action mapping χ_A maps each action in the target task to the most similar action in the source task:

$$\chi_A(a_i^{(t)}) = a_j^{(s)} \tag{3}$$

Depend on the particular function approximators, the transfer function ρ could be constructed from χ_S and χ_A. However, each state variables and/or action in target task is mapped to only one correspondence in source task. As a result, there are several state variables in target task mapped to the same state variable in source task, and the same situation for action mapping. Thus, there would be actions that have no difference for the learner, which is not that much reasonable. Therefore, we propose the linear multi-variable mapping (LMVM) to different the state-action values for different state variables and actions in target task.

4.2 Linear Multi-variable Mapping

Based on the TVITM, we design a linear multi-variable mapping. The idea behind our method is that the state in a new task is different from that in the old task, even if they have the same denotation or meaning, the roles that the state plays in two tasks are different. Therefore, a simple one-to-one mapping from the state variable with the same or similar denotation to the new state variable would weaken the effect of the transfer learning. To make the best use of the information in the old task, all the related information should be collected as much as possible. Thus, a state variable in the new task should map from all the related state variables in the old task. To integrate all the information of every related state variables, a linear model is a desirable one.

In LMVM, we first divide the state variables into different groups according to their meaning or similarity. For the state space of target task $S^{(t)}$ and the state space of the source task $S^{(s)}$, we divide each state space into K subspace, that is $S^{(s)} \rightarrow (S_0^{(s)}, S_1^{(s)}, ..., S_K^{(s)})$, and $S^{(t)} \rightarrow (S_0^{(t)}, S_1^{(t)}, ..., S_K^{(t)})$. In this division, $S_k^{(s)}$ and $S_k^{(t)}$ are subspace with the same meaning or the most similar meaning, but with different number of state variables. Our idea is that state variables in subspace $S_k^{(t)}$ are related with every state variables in subspace $S_k^{(s)}$, not just related to one state variables in $S^{(s)}$. For $s_p^{(t)} \in S_k^{(t)}$, the state variable mapping is:

$$s_p^{(t)} = \chi_S^{(-1)}(S_k^{(s)}) \tag{4}$$

Let $M_k = |S_k^{(s)}|$ that denotes the number of variables in subspace $S_k^{(s)}$. Under state variable mapping χ_S, any function $\Phi(\bullet)$ for state variables should follow Equation (5).

$$\chi_S : \Phi(s_p^{(t)}) = \sum_{i=1}^{M_k} (\xi_i \Phi(s_i^{(s)})) \tag{5}$$

where ξ is weighting factor for states variables, and $\sum_{i=1}^{M_k}(\xi_i) = 1$. Let $\mathbf{x}_p = [\xi_1, ..., \xi_{M_k}]$, then $\chi_S : \Phi(s_p^{(t)}) = \mathbf{x}_p \Phi(S_k^{(s)})$.

In this paper, we focus on tasks that have a small action space. Therefore, we don't divide the action space in the way we divide state space. Currently, the action space are divided into just one subspace. Similarly, the action mapping for $a_q^{(t)} \in A^{(t)}$ is:

$$a_q^{(t)} = \chi_A^{(-1)}(A^{(s)}) \tag{6}$$

Let $N = |A^{(s)}|$ that denotes the number of variables in subspace $A^{(s)}$. Under the action mapping χ_A, any function $\Psi(\bullet)$ for actions should follow Equation (7).

$$\chi_A : \Psi(a_q^{(t)}) = \sum_{j=1}^{N} \beta_j \Psi(a_j^{(s)}) \tag{7}$$

where β is the weighting factor for actions, and $\sum_{j=1}^{N} \beta_j = 1$. Let $\mathbf{y}_q = [\beta_1, ..., \beta_N]$, then $\chi_A : \Psi(a_q^{(t)}) = \Psi(A^{(s)})\mathbf{y}_q^T$.

Based on the inter-task mapping, the state-action value in target task could be initialized by the transfer function with the final state-action value in the source task.

$$Q_0^{(t)}(s_p^{(t)}, a_q^{(t)}) = \rho(Q_{final}^{(s)}(S_k^{(s)}, A^{(s)})) \quad (8)$$

We use a linear way to construct the transfer function ρ.

$$Q_0^{(t)}(s_p^{(t)}, a_q^{(t)}) = \sum_{w,i,j} \eta_w Q_{final}^{(s)}(s_i^{(s)}, a_j^{(s)}) \quad (9)$$

$$= \mathbf{x}_p Q_{final}^{(s)}(S_k^{(s)}, A^{(s)}) \mathbf{y}_q^T \quad (10)$$

where $s_p^{(t)} \in S_k^{(t)}, a_q^{(t)} \in A^{(t)}, s_i^{(s)} \in S_k^{(s)}, a_j^{(s)} \in A^{(s)}$, and η is weighting factor. The value of η is determined by the inter-task mapping χ_S and χ_A. As long as we could choose appropriate parameters ξ and β, we could make a better use of information from the multi-variable mapping.

5 Transfer via LMVM in Keepaway

To show the transfer via LMVM in detail and verify the effectiveness of LMVM, we use Keepaway platform as the testbed, and transfer knowledge from 3vs.2 Keepaway (i.e., 3 keepers and 2 takers) to 4vs.3 Keepaway (i.e., 4 keepers and 3 takers).

5.1 Transfer Learning in Keepaway

As detailed in Section 2, there are six types of state variables in the state space. Therefore, we could divide the state space into six group according to the types. Take the group of state variables $dist(K_1, T_j)$ as an example, let $\mathbf{D} = [dist(K_1, T_1), dist(K_1, T_2)]$ denotes this subspace in source task and $\mathbf{A} = [a_0, a_1, a_2, a_3]^T$ denotes the action space of the source task. The LMVM for state variables in this group could be defined according to Equation (5). Table 1 gives the state variable mapping of this group, which maps state variable from 3vs.2 Keepaway to 4vs.3 Keepaway by LMVM and the inter-task mapping given in [3]. Similarly, according to Equation (7), the action space mapping is given in Table 2.

s in 4v3		s mapped from 3v2	
variable	Denotion	LMVM	TVITM in [3]
$dist(K_1, T_1)$	s_1	$\mathbf{x}_1\Phi(\mathbf{D})$	$dist(K_1, T_1)$
$dist(K_1, T_2)$	s_2	$\mathbf{x}_2\Phi(\mathbf{D})$	$dist(K_1, T_2)$
$dist(K_1, T_3)$	s_3	$\mathbf{x}_3\Phi(\mathbf{D})$	$dist(K_1, T_2)$

Table 1: An Example of State Variable Mapping in Keepaway

a in 4v3	a mapped from 3v2	
	LMVM	TVITM in [3]
a_0	$\Psi(\mathbf{A})\mathbf{y}_1^T$	a_0
a_1	$\Psi(\mathbf{A})\mathbf{y}_2^T$	a_1
a_2	$\Psi(\mathbf{A})\mathbf{y}_3^T$	a_2
a_3	$\Psi(\mathbf{A})\mathbf{y}_4^T$	a_2

Table 2: Action Mapping in Keepaway

After defining the LMVM for χ_S and χ_A, we could construct the LMVM transfer function ρ as follows:

$$Q(s_i, a_j) = \mathbf{x}_i Q(\mathbf{D}, \mathbf{A}) \mathbf{y}_j^T \quad (11)$$

The mapping for state variables in other subspaces are dealt in a similar way. Since the six subspaces of Keepaway state space have very similar patterns, we use the same set of parameters $\mathbf{x}_i$ and $\mathbf{y}_j$ for different subspaces in our experiments.

5.2 Experiment Setting

In our experiments, we carry out experiments on Keeepaway Platform to transfer knowledge from 3vs.2 Keepaway to 4vs.3 Keepaway. There are two set of experiments, one is to transfer via LMVM, the other one is to transfer with the TVITM in [3].

In keepaway, the goal of the keepers is to keep the ball as long as possible. Thus, we use the average episode duration to measure the performance of learning. In 3vs.3 keepaway, the average episode duration is roughly 19.2 seconds, while only reach about 9.3 seconds in 4vs.3 keepaway. To measure the effectiveness of transfer learning, we set a threshold of 9.0 seconds, and record the learning time it takes to reach this threshold in transfer learning. More specific, when the keepers in 4v3 keepaway have learned to maintain the episode for an average of 9.0 seconds over 1000 episodes, we regard that the keepers have sufficiently learned the 4v3 task. By recording this time over many trials we can compare the performance of two inter-task mapping in transfer learning.

Though transfer learning may help shorten the learning time in the target task, it will be not worth enough if the knowledge from the source task takes too much time to get. Therefore, we use knowledge learned after different episodes of 3vs.2 keepaway to test the transfer learning performance.

All the experiments use the Sarsa(λ) algorithm with CMAC function approximator. Both the source task and the target task are run in a 25×25 region in keepaway. The weights of tiles in the source task and target task without transfer learning are initialized with 0.

In the set of experiments for LMVM transfer, we set the parameters $\mathbf{x}_i$ and $\mathbf{y}_j$ as given in Equation (12) and Equation (13), respectively. This set of X and Y is choose by some intuition. For example, in the subspace for variables $dist(K_1, T_j)$ in 4vs.3 keepaway, the state variable $dist(K_1, T_1)$ is the distance to the nearest taker, so we give more weight to the state-value function related with $dist(K_1, T_1)$ than weight given to $dist(K_1, T_2)$ when mapping from 3vs.2 keepaway, say $\xi_1 = 2/3$ and $\xi_2 = 1/3$. To verify this intuition, we also set $\mathbf{x}_i$ and $\mathbf{y}_j$ as in Equations (14)-(17). Then we use the weights learned after 2000 episodes in 3vs.2 keepaway as the source task knowledge and use each pair of ($\mathbf{X}$,$\mathbf{Y}$) to run 3 trials. Table 3 shows the average time that LMVM transfer with different pairs of ($\mathbf{X}$,$\mathbf{Y}$) takes to reach the threshold. It indicates that ($\mathbf{X}_0$, $\mathbf{Y}_0$) works most effective among these three pairs, which is consistent with our intuition. Therefore, we will use ($\mathbf{X}_0$, $\mathbf{Y}_0$) to do experiments comparing with the inter-task mapping given in [3].

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} 2/3 & 1/3 \\ 1/2 & 1/2 \\ 1/3 & 2/3 \end{bmatrix} \tag{12}$$

$$\mathbf{Y}_0 = [\mathbf{y}_1 \quad \mathbf{y}_2 \quad \mathbf{y}_3 \quad \mathbf{y}_4] = \begin{bmatrix} 1/2 & 1/4 & 1/3 & 1/4 \\ 1/4 & 1/2 & 1/3 & 1/4 \\ 1/4 & 1/4 & 1/3 & 1/2 \end{bmatrix} \tag{13}$$

$$\mathbf{X}_1 = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \tag{14}$$

$$\mathbf{X}_2 = \begin{bmatrix} 1/3 & 2/3 \\ 1/2 & 1/2 \\ 2/3 & 1/3 \end{bmatrix} \tag{15}$$

$$\mathbf{Y}_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 \end{bmatrix} \tag{16}$$

$$\mathbf{Y}_2 = \begin{bmatrix} 1/4 & 1/3 & 1/4 & 1/2 \\ 1/4 & 1/3 & 1/2 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/4 \end{bmatrix} \tag{17}$$

$\mathbf{X}$,$\mathbf{Y}$ pair	$\mathbf{X}_0$,$\mathbf{Y}_0$	$\mathbf{X}_1$,$\mathbf{Y}_1$	$\mathbf{X}_2$,$\mathbf{Y}_2$
average time (hours)	5.95	6.81	6.33

Table 3: Comparison for different $\mathbf{X}$,$\mathbf{Y}$ pairs

5.3 Experiment Results

The average time for 4vs.3 keepaway task to reach the threshold without transfer is 21.32 hours. We run the experiments in 10 groups, each group use a source knowledge gained from different episodes of learning in 3vs.2 keepaway task. We run each group of experiments for 10 trials, the time that it takes to reach the threshold is recorded. Table 4 shows the experiment results. The first column is the number of episodes that learned in 3vs.2 keepaway task. The second column are results for transfer learning via LMVM, while the third column are results for transfer learning via inter-task mapping given in [3].

3vs.2 episodes	average time taken in 4vs.3 (hours)	
	LMVM	TVITM given in [3]
100	18.78±3.00	20.39±2.60
200	19.66±2.26	22.21±3.46
500	16.52±2.76	17.45±2.67
1000	16.35±2.00	15.65±2.27
1400	8.59±1.67	10.93±2.52
1700	6.00±1.38	8.69±2.18
2000	6.12±1.00	6.91±0.91
2300	7.28±2.50	7.30±1.23
2600	5.49±1.49	8.80±2.42
4000	11.90±2.12	13.54±2.81

Table 4: Results for transferring from 3v2 to 4v3 using LMVM and inter-task mapping given in [3]

From these results, we can see that transfer learning via inter-task mapping given in [3] learns faster than the learner without transfer in most of the cases, while transfer via LMVM learns faster than learner without transfer in all cases. The linear multi-variable mapping takes less time to reach the threshold than the inter-task mapping given [3] in almost every set of the experiments. In conclusion, the linear multi-variable mapping succeed in making a better use of the knowledge than that of the inter-task mapping presented in [3].

6 Conclusion

In this paper, we proposed an linear multi-variable mapping for transfer learning based on the work in [3]. The new mapping aims at making a better use of the knowledge in the source task, and provide a better performance in transfer learning. The experiment results show that the linear multi-variable mapping could outperform the one in [3]. However, due to the limited time, we should have carried out more experiments to evident whether there are some other benefits we could get from the newly designed mapping. Apart from this, the choose of the optimal values for parameters $\mathbf{X}$ and $\mathbf{Y}$ is an open question that could be left for the future works.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[2] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[3] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *Journal of Machine Learning Research*, vol. 8, no. Sep, pp. 2125–2167, 2007.

[4] A. Fachantidis, I. Partalas, M. E. Taylor, and I. Vlahavas, "Transfer learning via multiple inter-task mappings," in *European Workshop on Reinforcement Learning*, pp. 225–236, Springer, 2011.

[5] A. Fachantidis, I. Partalas, M. E. Taylor, and I. Vlahavas, "Transfer learning with probabilistic mapping selection," *Adaptive Behavior*, vol. 23, no. 1, pp. 3–19, 2015.

[6] M. E. Taylor, G. Kuhlmann, and P. Stone, "Autonomous transfer for reinforcement learning," in *In The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008.

[7] H. B. Ammar, D. C. Mocanu, M. E. Taylor, K. Driessens, K. Tuyls, and G. Weiss, "Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 449–464, Springer, 2013.

[8] G. Konidaris, I. Scheidwasser, and A. Barto, "Transfer in reinforcement learning via shared features," *Journal of Machine Learning Research*, vol. 13, no. May, pp. 1333–1371, 2012.

[9] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu, "Keepaway soccer: From machine learning testbed to benchmark," in *Robot Soccer World Cup*, pp. 93–105, Springer, 2005.