An Autonomous Inter-task Mapping Learning Method via Artificial Neural Network for Transfer Learning

Conference Paper · December 2017

DOI: 10.1109/ROBIO.2017.8324510

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 2017 IEEE
International Conference on Robotics and Biomimetics
December 5-8, 2017, Macau SAR, China

An Autonomous Inter-task Mapping Learning Method via Artificial Neural Network for Transfer Learning

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

***Abstract*—Transfer learning could speed up reinforcement learning in many applications. Toward the fully autonomous reinforcement learning transfer agent, the mapping between the source task and target task should be learned instead of human-designed. To this end, this paper proposes an autonomous inter-task mapping learning method via artificial neural network, so as to reduce the human intervention in the transfer process. With this learned network, the reinforcement learning agent could transfer the learned knowledge from source task to target task for initialization, and set a good prior for the learning in the target task. The method is tested on the Keepaway Soccer Platform. The results indicate that the proposed method could provide a good jumpstart in the target task when weights are properly chosen for training the network.**

I. Introduction

Reinforcement learning (RL) [1] is a very widely used learning paradigm for agents to learn policy from limited feedback. Typically, the RL agent needs to interact with the environment to gain as much feedback or experience as possible to learn a higher quality policy. However, these interactions may cost too much time or expense, especially in some complex tasks.

It is similar with the case that we human face a new problem. Usually, it takes us a lot of time and energy to try all means available to find the best solution. However, when we have some experience on similar problems, we may not need to try too much means and get the solution in a short time. Transfer learning is a technique based on this idea. It reuses knowledge learned from similar source task to help reducing the amount of time or data needed for learning target task, and even improve the final performance in the target task.

However, since the source task and target task are different, the knowledge learned from source task could not be directly used in the target task, but some generalization is required. Therefore, successful transfer learning depends much on how we generalize the source task knowledge and how we apply them to the target task.

In many works, the inter-task mapping is used for generalizing the knowledge learned from source task to the target task. Typically, the inter-task mapping is hand-coded with human knowledge. However, a fully autonomous RL transfer agent [2] would have to perform all the three steps itself: a) select the appropriate source task autonomously, b) learn the relationship between the source task and target task, c) effectively transfer knowledge from the source task to the target task. In most transfer via inter-task mapping methods, the inter-task mappings are provided by human experts. This paper will focus on the second step of the fully autonomous transfer agent and try to make some contribution toward the fully autonomous transfer learning. In this paper, we propose an autonomous inter-task mapping learning method via artificial neural networks (ANN), based on the idea that elements (state variables and actions) in the target task may have relationship with many of the elements in source task. The proposed mapping could be learned directly from the tasks and do not need to be designed or provided by human experts. Then we use this learned neural network to map the knowledge learned in source task to initialize the target task. In this way, we could set a good prior for the target task so as to promote the reinforcement learning in the target task. At last, the proposed method will be tested on the Keepaway which is a subtask an soccer simulation platform.

The rest of the paper is organized as follows. Section 2 reviews some related works. The background knowledge is presented in Section 3. Our proposed method is described in Section 4. In Section 5, some experiments are carried out on Keepaway platform. Conclusion is given in Section 6.

II. Related Works

There are mainly two groups for transfer learning, one is in data mining community, such as transfer learning for classification, regression, and clustering problems [3], the other one is transfer learning for reinforcement learning in the machine learning community [2]. Our work in this paper researches on the transfer learning in reinforcement learning domain. As a technique to speedup the learning in reinforcement learning process, transfer learning has attracted a lot of 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 mappings are pre-defined by experts according to their experience or intuition. Taylor et. al

978-1-5386-3741-8/17/$31.00 © 2017 IEEE

proposed transfer via inter-task mapping (TVITM) in [4] to speedup the learning on task with action-value function RL. The inter-task mapping consists of a state-mapping and an action mapping, which are both one-to-one mappings. They construct the transfer function based on these two mappings so as to initialize the action-value function of the target task with knowledge from source task. Our previous work [5] proposed a many-to-one mapping for the transfer learning, called the linear multi-variable mapping, which uses the linear combination of the information from different related state variables and action to initialize the target task learning. However, it still requires humans to provide the parameters of the linear combination, and the optimal parameter values are not easy to be given.

There are some researchers who design mechanisms to select good mappings from several pre-defined mappings. Anestis et.al [6] propose two algorithms to select the best mapping from multiple mappings for both model-learning and model-free reinforcement learning algorithms to transfer from multiple inter-task mappings. In another of their work [7], they propose a method which autonomously selects mappings from the set of all possible inter-task mappings.

There are also some methods for learning inter-task mapping automatically. Taylor et. al [8] proposed a method called Modeling Approximate State Transitions by Exploiting Regression (MASTER). It automatically learns a mapping from one task to another by using experience gathered from task environments. Haitham B. Ammar et. al [9] proposed a framework to learn inter-task mappings between different MDPs automatically by an adaptation of restricted Boltzmann machines.

The most similar to our work is the work of Luiz et.al [10] which use a Neural Network to map actions from one domain to another domain, but the mapping between the states is assumed. They need to observe the results of the two actions in source domain and target domain so as to learn the weights of the network.

III. BACKGROUD

A. Reinforcement Learning

Reinforcement learning tasks are commonly described by Markov decision process (MDP) $< S, A, T, R >$, where S is the set of states, A is the set of agent actions. $T : S \times A \times S \rightarrow [0, 1]$ is the state transition probability function, which indicates the task dynamic. $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function which measures the performance of the agent. For 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, so 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 maximizes the total amount of reward it receives [1]. In general, the goal is defined by the expected return, and value function specifies what is good in the long run, so almost all reinforcement learning algorithms are based on estimating value functions [1]. The state-action value function (Q-function) for taking action a in state s under policy π could be defined as in Equation (1), where $\gamma \in [0, 1)$ is the discount factor.

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

Usually, the estimates of value functions are represented as a table with one entry for each state or state-action pair. When faced with large tasks and tasks with continuous space, the generalization is needed so as to produce a good approximation from limited samples. There are many function approximators for RL tasks, 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 then give the estimated value function. We could denote them in the form of $Q(s, a) = f(s, a, \varphi)$. Then f is the function approximator we choose, and φ is the argument of approximator f. The reinforcement learning process in these cases could be regarded as the learning process of argument φ.

B. Transfer Learning

Transfer learning uses learned knowledge from source task to provide priors for the target task learner. In source task, the learning could be denoted as $(S^{(s)}, s_0^{(s)}, A^{(s)}, T^{(s)}, R^{(s)}, Q_0^{(s)}) \Rightarrow Q_{final}^{(s)}$. Similarly, the target task could be denoted as $(S^{(t)}, s_0^{(t)}, A^{(t)}, T^{(t)}, R^{(t)}, Q_0^{(t)}) \Rightarrow Q_{final}^{(t)}$. Typically, the source task and the target task are related. As the RL agent learns by exploring the environment, the related knowledge from source task which serves as a prior could help to reduce the scope of exploration or set a better start point for the exploration in target task, therefore the learner could learn faster than just exploring randomly in environment as in RL without transfer. In reinforcement learning task based on value function, the transfer learner uses source knowledge $Q_{final}^{(s)}$ to bias the initial value function $Q_0^{(t)}$ in the target task. Since the Q value functions are approximated by function approximators, the bias could be done on the function approximators structure.

However, the target task usually has some differences with the source task, that is $S^{(s)} \neq S^{(t)}$ and/or $A^{(s)} \neq A^{(t)}$. Therefore, no matter bias the target task by Q value or approximator structure from source task, certain mapping ρ is needed to map the knowledge in source task to target task. The mapping should consider both the state mapping and the action mapping between source task and target task, we denote them by χ_S and χ_A, respectively. Therefore, the inter task mapping could be denoted as $\rho = \Phi(\chi_S, \chi_A)$. Since Q value could be represented by the structure of function approximator, we attempt to learn a mapping between the approximator structure of source task and that of target task, and map the source knowledge with the learned mapping to

initialize the function approximator in the target task.

C. Artificial Neural Network

Artificial Neural Network (ANN) is a kind of computational structure based on the mechanism of biological neural network. Therefore, it is also comprised of densely interconnected artificial neurons (or nodes). The ANNs have a lot of attractive characteristics such as nonlinearity, high parallelism, robustness, fault and noise tolerance, learning and generalization capabilities [11]. There are various types of ANNs, one of the most widely used one is called the backpropagation (BP) ANNs. The BP network consists of (i) an input layer, (ii) an output layer, (iii) one or more hidden layers. The backpropagation networks take the input data by the nodes in the input layer, and get the output from the nodes in the output layer. The error computed at the output layer is propagated backward from the output layer to the hidden layer, and finally to the input layer. The weights are updated during this back propagation process. This is why it called backpropagation network.

IV. Autonomous Mapping Learning Method for Transfer Learning

We propose an autonomous mapping learning method for transfer learning, which uses artificial neural network to learn the inter-task mapping between source task and target task. For the description convenience, we assume the RL learner uses the CMAC as the function approximator. Therefore, the RL agent learn by updating its tile weights of CMAC. The transfer learning process with the proposed mapping learning method could be divided into three parts: a) the structure design of the neural network, b) the training of the network, and c) the use of the trained network. The whole procedures are listed in Algorithm 1, and the detail of each part will be given in the following subsections.

Algorithm 1 ANN-based autonomous inter-task mapping learning method

```
run the source task for p episodes
run the target task for q episodes, q ≪ p
input data ← tile weights from q episodes of source task
output data ← tile weights from q episodes of target task
train the ANN with BP algorithm
    Repeat (for each training):
        foreward data from input layer to output layer
        compute the error at output layer
        backpropagate the error, and update the network
    until error < δ
input of ANN ← weights of p episodes source task
initial tile weights of target task ← output of the ANN
start the reinforcement learning in the target task
```

A. Structure Design of the ANN

We use a three-layer artificial neural network to learn the inter-task mapping, just as shown in Figure 1. The three layers are input layer, hidden layer and output layer. The input data is denoted by x_i, and the output layer data is denoted by o_k. The weights matrix between input layer and hidden layer is V, and the weights matrix between hidden layer and output layer is W. The goal is to learn the matrix V and W.

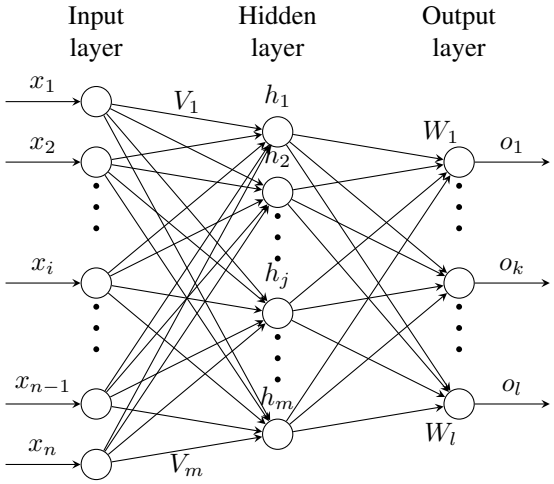

Fig. 1: The ANN structure with its denotations

In our work, the source task and target task are learned by reinforcement learning with CMAC as the function approximator of Q values. The function approximation is trained by changing how much each tile contributes to the Q value, as shown in Equation (2), where f_i denotes the tile value and c_i denotes the weights for tile i.

$$Q(s,a) = \sum_i c_i f_i(s,a) \tag{2}$$

In CMAC, when there are a lot of state features, we usually use one-dimensional tilings, that is each tile is just activated by one state variable and independent of other state variable. Suppose T is the set of tiles that activated by state $s = [v_1, ..., v_k, ...v_n]$, then T could be divided into n subset $T = \{T_1, ..., T_k, ..., T_n\}$, and each T_k is the set of tiles that activated by state variable v_k.

$$Q(s,a) = \sum_k \sum_{j \in T_k} c_j f_j(v_k, a) \tag{3}$$

In the source task, there are n_s state variables, that is $s_s = [v_{s1}, ..., v_{sk}, ...v_{sn_s}]$. The number of actions in the action space is m_s. Similarly, the number of state variables and actions in the target task is n_t and m_t, respectively. We set the number of the input nodes to be $N_i = n_s * m_s$, and the number of output nodes to be $N_o = n_t * m_t$. Each input node corresponds to the tile that activated by state variable v_{sk} with action a_{sk} in the source task. Similarly, each output node corresponds to the tile that activated by state variable v_{tk} with action a_{tk} from the target task. The number of hidden nodes is set to be $N_h = \frac{1}{2}(N_i + N_o)$.

B. Training the ANN for the Mapping

After the design of the ANN structure, we should next extract the training data from the stored source task weights and target task weights. We run the source task and target task both for q episodes, and store the weights learned in these two tasks. Suppose the target task takes p episodes to get the optimal policy, and we let $q \ll p$. The nodes of input layer take the tile weights from the source task, while the nodes of output layer use the tile weights from the target task as the expected value to compute the errors. Then the errors at the output layer are used to update the weight matrix V and W.

Since BP algorithm is one of the most widely used method, and it's also very easy to understand and execute, we choose the BP algorithm to train the ANN.

For the hidden layer, we have

$$\begin{cases} h_j = f(net_j), & j = 1, 2, ..., m \\ net_j = \sum_{i=0}^{n} v_{ij} x_i, & j = 1, 2, ..., m \end{cases} \tag{4}$$

For output layer, we have

$$\begin{cases} o_k = f(net_k), & k = 1, 2, ..., l \\ net_k = \sum_{j=0}^{m} w_{jk} h_j, & k = 1, 2, ..., l \end{cases} \tag{5}$$

where we use sigmoid function as the $f(x)$ function, that is

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

Let d denotes the expected output in the output layer, then the error in the output layer is

$$E = \frac{1}{2}(\mathbf{d} - \mathbf{o})^2 = \frac{1}{2}\sum_{k=1}^{l}(d_k - o_k)^2 \tag{7}$$

Through back propagate this error, we could get the training rules as follows.

$$\delta_k^o = -\frac{\partial E}{\partial net_k} = (d_k - o_k) o_k (1 - o_k) \tag{8}$$

$$\delta_j^y = -\frac{\partial E}{\partial net_j} = (\sum_{k=1}^{l} \delta_k^o w_{jk}) h_j (1 - h_j) \tag{9}$$

$$\Delta w_{jk} = \eta \delta_k^o h_j \tag{10}$$

$$\Delta v_{ij} = \eta \delta_j^h x_i \tag{11}$$

Use the data from q episodes of source task as input, and q episodes of target task as output, we could train the network with Equations (8,9,10,11), and get the inter-task mapping expressed by the artificial neural network.

C. Transfer Learning with the ANN Learned Mapping

In reinforcement learning without transfer, the initialization of target task is simply done by setting all the tile weights to be zero. Therefore, every state variable have the same influence on the choosing of action at the beginning of the game, and every action has the same priority when the learner choose which action to execute. The agent need to spend a lot of time on exploring. The transfer learning here is used to help the agents avoid this shortcoming of reinforcement learning.

In this stage, we will finally reuse the knowledge learned from source task. The learned knowledge from the source task is the tile weights stored after a long time of learning in the source task. Use these weights as input data of the trained ANN, we could map the learned knowledge from the source task to the target task by using the output of the ANN to initialize the tile weights in the target task. Therefore, the initialization of the target task learning would be more informational, and the agents could be set up for the reinforcement learning at a better starting point.

V. Experiments

To verify the effectiveness of the proposed inter-task mapping learning method, we have done some experiments on the Keepaway soccer platform [12].

A. Keepaway Platform

Keepaway soccer platform [12] is a popular testbed for machine learning. There are two teams, 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 2 is an screenshot for a Keepaway scenario with 4 keepers and 3 takers (called 4vs.3 Keepaway). Suppose there are n keepers and m takers, $K_1, ..., K_n$ are keepers, 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 in the closeness to the K_1.

Fig. 2: 4vs.3 Keepaway scenario

The keeper with the ball choose actions from the action set {Hold, PassToKeeperK_2,...., PassToKeeperK_n}, so there are n actions.

The state variables include distance variables and angle variables. The state space has $(4n + 2m - 3)$ state variables in total, including six types of variables. The six types are: n distances variables from a keeper to the center, m distances variables from a taker to the center, $(n-1)$ distances variables from a keeper to K_1, m distances variables from a taker to K_1, $(n-1)$ variables for minimal distances from a keeper without ball to a taker, $(n-1)$ variables for minimal angles between a keeper and a taker whose the vertex is at K_1.

The action space and state space are different for different player settings. The more players, the bigger action space and

state space, as well as the longer time needed for learning. Therefore, we could use an easier task (fewer players) as the source task and a task with more players as the target task to conduct the transfer learning.

B. Experiments Settings

We use the 3vs.2 Keepaway scenario as the source task, and 4vs.3 scenario as the target task. There are 13 state variables and 3 actions in the source task, and 19 state variables and 4 actions in the target task. Therefore, the structure of the neural network is set as follows: 39 nodes in the input layer, 76 nodes in the output layer, and 57 nodes in the hidden layer.

We have done three sets of experiments for transfer with the proposed mapping. Each set of experiment transfers with ANN trained from weights of different episodes' Keepaway. The different episode numbers for getting the training weights are set to be 200, 500 and 1000, respectively. In the first set of ANN-mapping transfer learning experiments, we use weights obtained by running 200 episodes in both 3vs.2 scenario and 4vs.3 scenario to train the ANN mapping, so we denote this set of experiments with ANN-200. The same go with ANN-500 and ANN-1000. For comparison, we also run the experiments with two other inter-task mapping in the same condition. One is the inter-task mapping proposed in [4], we denote it by TVITM, the other one is the linear inter-task mapping proposed in [5], we denote it by Linear-TL. In all these five set of experiments, simulation are run with 10 different source knowledge obtained from different episodes of 3vs.2 scenario Keepaway, and the episodes are ranging from 200 to 8000. Each source knowledge case is run for 10 trials. The results are the average of these 10 trials.

As listed in Mathew's survey [2], there are five metrics to measure the benefits of transfer: jumpstart, asymptotic performance, total reward, transfer ratio, and time to threshold. In our experiments, we choose the jumpstart as the metric. Jumpstart is the initial performance of an agent in the target task that has been improved by transfer from a source task. Since the goal of the keeper in the platform is to keep the ball as long as possible, the duration of each episode is the metric for the performance of the learners. Therefore, the jumpstart here is refer to the initial episode duration that the transfer learning has improved.

C. Experiment Results

The average initial episode duration of 10 trials in target task without transfer learning is 5.14 ± 0.10 seconds, and the initial episode duration for each trial is obtained by averaging the first 1000 episodes. We use the same method to get the initial episode duration for each set of experiments. The experiment results of ANN-mapping based transfer learning is shown in the table I, while the comparison experiments of other inter-task mapping is show in table II.

From the experiment results in table I, we can see that the transfer learning with ANN learned mapping could make the RL agents have some jumpstart at the beginning of the learning. However, the results for ANN-200 is not very good,

3vs.2 episodes	initial episode duration in 4vs.3 (seconds)		
	ANN-200	ANN-500	ANN-1000
200	**5.26±0.06**	**5.28±0.04**	**5.29±0.04**
500	4.79±0.06	**5.28±0.06**	**5.32±0.02**
1000	4.55±0.02	**5.33±0.08**	**5.35±0.02**
1500	4.58±0.13	**5.58±0.07**	**5.52±0.02**
1800	5.01±0.11	**5.59±0.07**	**5.70±0.03**
2000	4.99±0.13	**5.51±0.10**	**5.73±0.02**
2500	4.97±0.08	**5.35±0.11**	**5.55±0.05**
3000	5.09±0.10	**5.17±0.09**	**5.31±0.11**
4000	**5.22±0.09**	5.08±0.08	**5.33±0.09**
8000	**5.21±0.15**	**5.21±0.07**	**5.33±0.03**

TABLE I: Results for transfer learning with ANN learned inter-task mapping

3vs.2 episodes	initial episode duration in 4vs.3 (seconds)	
	TVITM	Linear-TL
200	5.55±0.64	5.75±0.08
500	5.89±0.19	6.67±0.16
1000	5.94±0.11	6.68±0.17
1500	6.26±0.26	6.34±0.12
1800	6.58±0.16	6.37±0.11
2000	6.53±0.14	6.09±0.08
2500	6.48±0.20	5.98±0.06
3000	6.83±0.20	6.10±0.17
4000	6.39±0.12	5.99±0.08
8000	6.67±0.18	5.90±0.11

TABLE II: Results for two comparison methods

it even works worse than no-transfer case. It is interesting that the ANN-200 only outperforms the no-transfer case when transfer with 200 episodes of source knowledge and when the source task has been run for a very long time (like 4000 episodes and 8000 episodes). This is probably that the ANN maps knowledge to the weights structure of 200 episodes too well, which makes weights from other episodes map much less well. Since we just use a very short time of learning in the target task and source task as the training data of ANN, the mapping of state variables and actions between these two tasks is not too adequate. The exploration in the task with a short time just covers a very small part of the state-action space, thus the contribution to the action-choosing of each state variable has not been fully expressed in the weights file store from the task. On the other hand, after the source task has been learned very thoroughly, the influence of the inadequately mapped structure could be alleviated, so it will work better. The results of ANN-500 and ANN-1000 case is better, they could outperform the no-transfer case in almost every experiments. In addition, the ANN-1000 set is better than the ANN-500 set in almost every case. This is probably because more episodes learned in target task could give the ANN trainer a better ground for capture the structure of inter-task mapping. However, the two other comparison inter-task mapping have much better jumpstart. It seems that reducing the human intervention may weaken the performance of transfer learning. Therefore, this ANN learned mapping is suitable in situation in which human intervention is very hard or too expensive to get or the design of the inter-task mapping

is too difficult for human to work out.

We also computed the time to threshold of these transfer learning trials. However, the time to threshold of these transfer learning trials is longer than trials without transfer learning. This result shows that this transfer learning method is suitable at the beginning of the target task, but do not have much effect for the learning in the target task after a certain time.

VI. Conclusion

In this paper, we propose to use artificial neural network to learn the inter-task mapping between source task and target task, so as to transfer the knowledge learned in the source task to the target task for initialization. The results show some improvement in jumpstart metric, though not very good in other metrics, such as the time to threshold. However, it could reduce the human involvements in designing the inter-task mapping, which would be suitable for situations that human involvements are too much expensive or the design of the inter-task mapping is too difficult. A way to avoid the drawback of this method is to use this method at the beginning of the target task, but use other methods in the later period of the learning in target task, which would be our future work. Another future work could be use deeper-layered ANNs or use other ANN training methods to learn the inter-task mapping.

Acknowledgment

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

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] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[4] 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.

[5] Q. Cheng, X. Wang, and L. Shen, "Transfer learning via linear multi-variable mapping under reinforcement learning framework," in *Control Conference (CCC), 2017 36th Chinese*, pp. 8795–8799, IEEE, 2017.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] L. A. Celiberto Jr, J. P. Matsuura, R. L. De Mantaras, and R. A. Bianchi, "Using cases as heuristics in reinforcement learning: a transfer learning application," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 1211, 2011.

[11] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[12] 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.