Measuring the Distance Between Finite Markov Decision Processes

Jinhua Song, Yang Gao*, Hao Wang
State Key Laboratory for Novel Software Technology, Collaborative Innovation Center of Novel Software Technology and Industrialization
Nanjing University, Nanjing, China
songjinhua2008@gmail.com
gaoy@nju.edu.cn
wanghao@nju.edu.cn

Bo An
School of Computer Engineering
Nanyang Technological University
Singapore
boan@ntu.edu.sg

ABSTRACT

Markov decision processes (MDPs) have been studied for many decades. Recent research in using transfer learning methods to solve MDPs has shown that knowledge learned from one MDP may be used to solve a similar MDP better. In this paper, we propose two metrics for measuring the distance between finite MDPs. Our metrics are based on the Hausdorff metric which measures the distance between two subsets of a metric space and the Kantorovich metric for measuring the distance between probabilistic distributions. Our metrics can be used to compute the distance between reinforcement learning tasks that are modeled as MDPs. The second contribution of this paper is that we apply the metrics to direct transfer learning by finding the similar source tasks. Our third contribution is that we propose two knowledge transfer methods which transfer value functions of the selected source tasks to the target task. Extensive experimental results show that our metrics are effective in finding similar tasks and significantly improve the performance of transfer learning with the transfer methods.

Keywords

Markov decision process; Kantorovich metric; Hausdorff metric; reinforcement learning; transfer learning

1. INTRODUCTION

Markov decision processes (MDPs) are effective models for solving sequential decision-making problems [4] in uncertain environments. By making sequential decisions, the decision maker (called *agent*) can collect numerical feedbacks (called *rewards*) from the environment. The goal of the agent is to maximize the cumulative reward. Planning (e.g. Dynamic programming [3,23]) and learning (e.g. reinforcement learning [15,25]), which aim at finding an optimal mapping from states to actions (called *policy*), can be used to achieve the goal. When there are some MDPs that have been solved, the new MDPs can be solved better by reusing the knowledge of the solved MDPs. This idea has been used in reinforcement learning (RL) domain for many years, which is known as transfer learning in RL. Lots of transfer learning methods have been studied [18, 27], which focus on accelerating the learning process and improving the average approximation of the optimal value functions.

Although there has been lots of work on transfer learning, most approaches rely on the implicit assumption that source tasks and target tasks are very similar. However, the assumption is not necessarily true in practice. If the source task is irrelevant to the target task, transfer may cause slower learning speed or even worse performance in the target task; this phenomenon has been recognized as *negative transfer* [18, 24, 27, 28]. Therefore, it is necessary and essential to define the similarity relationship between the source tasks and the target task so as to select the best source task(s), which is the focus of this paper.

Unfortunately, there are few existing distance measures that can be used to estimate the similarity between two MDPs. Some work uses *transfer advantage* [6] to define the similarity between two MDPs [6, 9]. However, the similarity is usually difficult to be computed before the target task has been learned, thus it cannot be applied to transfer learning. Some work just measures the distance between dynamics of two MDPs, such as reward functions or transition models, and requires that the state-action spaces are the same [2, 6, 19]. Konidaris et al. [17] define *agent-space* to describe the features of an agent, and judge whether two MDPs are related by comparing the agent-spaces. However, it is difficult to automatically identify agent-space in advance.

Our contributions in this paper can be claimed as follows. First, we propose two metrics for measuring the distance between two MDPs based on the whole models of MDPs from a quantitative point of view for the first time. The metrics are based on the distance between states which consider actions, probability transition functions and reward functions of the states. We define the notion of homogeneous MDPs and extend Ferns' metric [10], which measures the distance between two states in *one* MDP, to compute the distances between states in different MDPs. After computing all the distances between states, we apply the Kantorovich metric [7] and the Hausdorff metric [5] to composite them to compute the distance between two MDPs. Kantorovich met-

*Corresponding Author

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

ric was first proposed in the field of transportation theory and is used to measure the distance between two distributions. We use Kantorovich metric with the assumption that a certain amount of "resources" from one MDP's states are allocated to the other MDP's states. However, when the state spaces of the MDPs are very large, computing Kantorovich metric may be computationally expensive. Thus, we apply another easy-to-compute metric, Hausdorff metric. Hausdorff metric is a mathematical tool for measuring the distance between two sets of points that are subsets of a metric space. The distance between two sets of states is used to represent the distance of two MDPs. Although Hausdorff metric can be computed more efficiently, it may miss some similar MDPs due to "outliers" (states which are very different from others). Therefore, the distance can be computed by Hausdorff metric if the state space is too large and by Kantorovich metric otherwise.

Our second contribution is that we use our metrics to measure the distance between RL tasks which are modeled as MDPs, and apply them in transfer learning for R-L. In this paper we focus on transfer learning in one domain and consider homogeneous MDPs which widely exist in one domain as most related works in transfer learning (e.g., [14, 21, 27, 30]). We apply the metrics for source tasks selection in transfer learning. Our third contribution is proposing two transfer methods which transfer value functions of selected source tasks to the target task. After selecting the proper source tasks, transfer methods are used to transfer knowledge to the target task. Since the metrics are based on the distances between states, value functions are used as the transfer knowledge.

Experimental results show that our metrics are effective in finding similar tasks to avoid negative transfer in transfer learning, and significantly improve the performance (i.e. reducing episodes to converge to the optimal policy by more than a third and achieving better asymptotic performance) of the baseline algorithms with our transfer methods.

2. BACKGROUND

2.1 Markov Decision Process

In this paper, we focus on finite Markov decision processes.

DEFINITION 2.1. *A finite Markov decision process can be represented as a 4-tuple* $M = \{S, A, P, R\}$*, where* S *is a finite set of states;* A *is a finite set of actions;* $P : S \times A \times S \to [0, 1]$ *is the probability transition function; and* $R : S \times A \to \Re$ *is the reward function. In this paper, we denote the probability of the transition from state* s *to another state* s' *when taking action* a *by* $P^a_{ss'}$ *and the immediate reward received after the transition by* r^a_s.[1]

A policy is defined as a mapping, $\pi : S \times A \to [0, 1]$. To measure the quality of a policy, a value function, $V^\pi(s)$, is used to estimate the expected long-term cumulative reward from state s under a policy π. It is formally defined as:

$$V^\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \big| s_0 = s \right],$$

where γ is a discount factor, r_t is the reward at time-step t, and E_π is the expectation with respect to the policy π. It can be expressed as the Bellman equation:

$$V^\pi(s) = \sum_{a \in A} \pi(s, a) \sum_{s' \in S} P^a_{ss'} \left[r^a_s + \gamma V^\pi(s') \right].$$

Similarly, action value function (called Q-value), the expected long-term cumulative reward of taking action a in state s under a policy π, is defined as:

$$Q^\pi(s, a) = \sum_{s' \in S} P^a_{ss'} \left[r^a_s + \gamma \sum_{a' \in A} \pi(s', a') Q^\pi(s', a') \right].$$

The goal is to find an optimal policy π^* which maximizes the expectation of the long-time discounted cumulative reward from any starting state $s \in S$:

$$\pi^* = \arg\max_\pi E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right].$$

2.2 Transfer Learning

The idea of transfer learning comes from psychology and cognitive science research. It is motivated by the finding [22] that humans can solve a problem faster by reusing knowledge learned from solving related problems. In the MDP domain, agents are expected to improve their performance through transferring the knowledge retained from similar MDPs.

Consider the situation that an agent is solving a sequence of MDPs. If these MDPs are related and the agent has solved n MDPs, the agent may use the knowledge learned from the prior MDPs when solving the $(n + 1)$-th MDP. Transfer learning aims to select the learned MDP(s), choose the transfer knowledge (e.g. instances [19], Q-values [29] and policies [16]), and use the knowledge to improve the performance of solving the new MDP. Related MDPs may have some common knowledge which can help the new MDP to be solved better, and transferring from MDPs which differ too much from the new MDP may cause negative transfer. Thus, the notion of MDP similarity (or distance) is essential to transfer learning. However, many transfer approaches ignore this problem. Few work define the similarity, but their definitions can only be used under strict conditions or cannot be applied to transfer learning. In the next section, we define measures that can be used to estimate the similarity between two MDPs.

3. METRICS FOR MEASURING THE DISTANCE BETWEEN MDPS

Our goal is to define metrics to measure the distance between two MDPs based on the whole models of MDPs including state transitions and reward structures. Note that an MDP contains a set of states, and each state s has its own action set A_s, probability transition function P_s and reward function R_s. Therefore, the distance between two MDPs is based on the distance between their states, and the distance between two states depends on their action sets, probability transition functions and reward functions. To measure the distance between two states in different MDPs, we extend Fern's metric of states [10], which considers actions, probability transition functions and reward functions of those states. Then we apply Kantorovich metric and Hausdorff metric to measure the distance between the two state sets of two MDPs.

[1] In fact, the reward could also be related to s'. However in many practical applications, it is usual the case that the reward is determined by the state s and the action a [10]. We follow this convention in this paper.

3.1 Measuring States' Distance in Different MDPs

In this section, we introduce a metric to measure the distance between two states of two MDPs by extending an existing metric for measuring the distance of two states of one MDP [11–13].

DEFINITION 3.1. *For any $s, s' \in S$, their distance $d(s, s')$ is defined as*

$$d(s, s') = \max_{a \in A} \{|r_s^a - r_{s'}^a| + cT_K(d)(P_s^a, P_{s'}^a)\}, \quad (1)$$

where $c \in (0, 1)$, P_s^a (resp. $P_{s'}^a$) is the probabilistic transition when action a is taken at state s (resp. s'), and $T_K(d)(P_s^a, P_{s'}^a)$ is the Kantorovich distance between the two probabilistic transitions.

Notice that the above metric is only defined on states in the same MDP and it may not be directly applied to compute the distance between two states in different MDPs. It is known that a specific task can be modeled as different MDPs with different state representations and action representations [25, 26], and it is difficult for the agent to recognize whether the MDPs have the same state-action spaces. Therefore, the above metric is not applicable when the state and action representations are different. Thus, we extend it to measure the distance between states in different MDPs. Our key idea is that if two MDPs can be merged into one, then we directly use this metric to measure the distance between states in the "new MDP". To decide whether MDPs can be merged, we first give the definition of homogeneous MDPs.

DEFINITION 3.2. *MDPs are homogeneous if they satisfy the following conditions:*

(1) their state representations are equivalent (the state-spaces can be different);

(2) there is a one-to-one correspondence between their action spaces;

(3) they are reward-linked.

Equivalent state representations mean that the states in different MDPs have the same kind of representations (i.e. same state variables [27]). For example, states in different Gridworlds can be represented by the agents' positions. Thus these Gridworlds meet the first condition. If the action spaces are actually the same but they are named differently in different MDPs, it is possible to find a correspondence of actions. Actions *up*, *down*, $left$, *right* in one Gridworld can be mapped to *north*, *nouth*, *west*, *east* in another Gridworld. *Reward-linked* [17] indicates that the rewards are equal when reaching the same subgoals (or goals) in two MDPs, such as arriving at the exports in different mazes and eating pac-dots in different Pac-man Games [1].

If MDPs are homogeneous, they can be merged into one "new MDP", and metric d (Definition 3.1) can be used to measure the distance between their states. Consider the example that an indoor mobile robot is required to perform a task in different rooms, which are considered as different tasks and can also be modeled as different homogeneous MDPs. At the same time, the different tasks can be also treated as subtasks of one task and modeled in one MDP. Homogeneous MDPs are common in the MDP domain and we extend the metric in Definition 3.1 to compute the distance between states in different MDPs.

DEFINITION 3.3. *Given two homogeneous MDPs $M_1 = \{S_1, A, P_1, R_1\}$ and $M_2 = \{S_2, A, P_2, R_2\}$, the distance $d'(s, s')$ between any state $s \in S_1$ and any state $s' \in S_2$ is defined as:*

$$d'(s, s') = \max_{a \in A} \{|(r_1)_s^a - (r_2)_{s'}^a| + cT_K(d')((P_1)_s^a, (P_2)_{s'}^a)\}, \quad (2)$$

where $(r_1)_s^a$ (resp. $(r_2)_{s'}^a$) and $(P_1)_s^a$ (resp. $(P_2)_{s'}^a$) are the immediate reward and the probabilistic transition function in M_1 (M_2), and $T_K(d')((P_1)_s^a, (P_2)_{s'}^a)$ is the Kantorovich distance between the two probabilistic transitions.

DEFINITION 3.4. *For two state probability transition functions $(P_1)_s^a$ and $(P_2)_{s'}^a$ in different MDPs, their Kantorovich distance is*

$$\begin{aligned} T_K(d') = & \min_{\lambda_{kt}, k=1\ldots|S_1|, t=1\ldots|S_2|} \sum_{k=1}^{|S_1|} \sum_{t=1}^{|S_2|} \lambda_{kt} d'(s_k, s_t') \\ \text{s.t.} \quad & \sum_t \lambda_{kt} = (P_1)_{ss_k}^a, \forall k, \\ & \sum_k \lambda_{kt} = (P_2)_{s's_t'}^a, \forall t, \\ & \lambda_{kt} \geq 0, \forall k, t, \end{aligned} \quad (3)$$

where $s_k \in S_1$, $s_t' \in S_2$, $|S_1|$ represents the number of states in S_1, and $|S_2|$ denotes the number of states in S_2.

The solution, fixed point $d_f'(s, s')$, for Equation (2) always exists and is unique, which can be derived from the property of metric d [12]. Similar to metric d, fixed point $d_f'(s, s')$ can be computed in an iterative manner. The process of computing the distances between states in different MDPs is shown in Algorithm 1. For each state $s_i \in S_1$ and each state $s_j' \in S_2$, according to Equation (2), $d'(s_i, s_j')$, the maximal sum of the difference between the immediate rewards and Kantorovich distance between the probability transition functions, is computed using the distances of the last iteration (Lines 4-6). δ is the mean absolute error (MAE) of distances in two iterations (Line 7) and is initialized to a value larger than error threshold ξ (Line 1). d_0 stores the distances of the last iteration (Line 9). This process stops when the MAE is not larger than the error threshold (Line 10).

The worst case running time of computing the fixed point, $d_f'(s, s')$, is $O(|A||S_1|^2|S_2|^2|S_1+S_2|\log(S_1+S_2)\lceil\frac{\ln\eta}{\ln c}\rceil)$. However, when the two MDPs are deterministic ones, $T_K(d')$ in Equation (2) can be simplified to $d'(s_{next}, s'_{next})$ where $s_{next}(s'_{next})$ is the next state from $s(s')$. In this situation, the computational cost is $O(|A||S_1||S_2|\lceil\frac{\ln\eta}{\ln c}\rceil)$.

After computing all the distances between states in different MDPs, we need to composite them to compute the distance between the two MDPs. It is not appropriate to simply accumulate or average the distances between all different states. Consider the situation of two same MDPs, where the distance between the MDPs must be zero. However, the entries of the state distance matrix may not always be zero when the two states are not similar. Therefore, we focus on matchings of states and provide two proper metrics, the Kantorovich metric and the Hausdorff metric, for measuring the distances between MDPs. The Kantorovich metric essentially tries to find out an optimal state matching between the two MDPs, and the Hausdorff metric does not rely on any specific matching but rather bounds all "reasonable" matchings. We should note that, to our best knowledge,

Algorithm 1: Computing the distance between states of two homogeneous MDPs

Input: two MDPs $M_1 = \{S_1, A, P_1, R_1\}$ and $M_2 = \{S_2, A, P_2, R_2\}$, parameter c, error threshold ξ.
Output: distance matrix $d'(s_i, s'_j)$.

Initialization. $\delta > \xi$, distance matrixes $d_0(s_i, s'_j) = 0$, $d'(s_i, s'_j) = 0$.
repeat
  foreach $(s_i, s'_j, a_k) \in S_1 \times S_2 \times A$ **do**
    compute $T_K(P^{a_k}_{s_i}, P^{a_k}_{s'_j})$ according to Equation (3) using distance matrix d_0;
    $tempd = |r^{a_k}_{s_i} - r^{a_k}_{s'_j}| + cT_K(P^{a_k}_{s_i}, P^{a_k}_{s'_j})$;
    if$\big(tempd > d'(s_i, s'_j)\big)$ $d'(s_i, s'_j) = tempd$.
  $\delta = \Big(\sum_{i=1}^{|S_1|}\sum_{j=1}^{|S_2|} |d'(s_i, s'_j) - d_0(s_i, s'_j)|\Big) \Big/ (|S_1| * |S_2|)$;
  foreach $(s_i, s'_j) \in S_1 \times S_2$ **do**
    $d_0(s_i, s'_j) = d'(s_i, s'_j)$.
until $\delta \le \xi$;

other alternatives are hardly applicable. In the remaining of this section, we give details on the two metrics.

3.2 Hausdorff Metric Based Measure

Hausdorff metric measures the distance between two subsets of a metric space in mathematics. It computes the largest distance of all the distances from a point in one set to the nearest point in the other set. When Hausdorff metric is applied to two MDPs, we consider the state sets as the subsets of the metric space $\{M, d'\}$. For each state in one MDP, finding the most similar state in the other MDP is equivalent to mapping the states from one MDP to the other. After getting the mapped states of both MDPs, Hausdorff metric is applied to find the largest distance between the mapped states. We use this distance to represent the distance between two MDPs. The formal definition of the distance between two MDPs using Hausdorff metric is defined as follows.

Definition 3.5. *Given two MDPs $M_1 = \{S_1, A, P_1, R_1\}$ and $M_2 = \{S_2, A, P_2, R_2\}$, their Hausdorff distance $\Phi(S_1, S_2)$ can be computed as follows.*

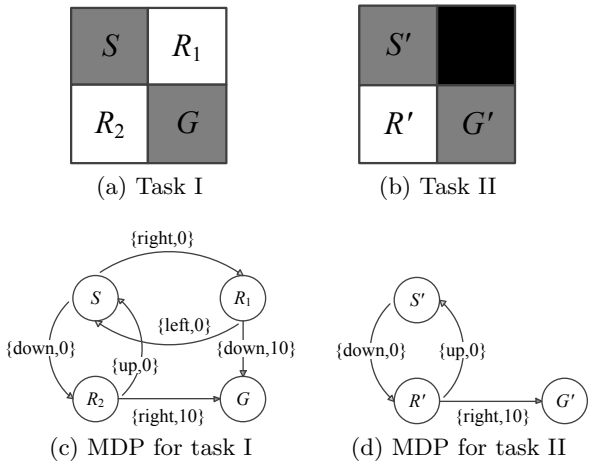
$$\Phi(S_1, S_2) = \max\left\{\max_{s\in S_1}\min_{s'\in S_2} d'(s, s'), \max_{s'\in S_2}\min_{s\in S_1} d'(s, s')\right\} \tag{4}$$

where d' is the metric defined in Definition 3.3.

Hausdorff metric can be efficiently computed and it only needs to compare $(2|S_1||S_2| - 1)$ times.

3.3 Kantorovich Metric Based Measure

Using Hausdorff metric to define the distance between MDPs leads to the possibility that similar MDPs are mistakenly treated as dissimilar ones if at least one of the them has an "outlier". Here the notion of an "outlier" is a state which is very different from others. Hausdorff metric computes the largest distance of all the distances from a state in one MDP to the nearest state in the other MDP. Therefore, if one MDP has outliers, the MDPs' distance is the distance of one of the outliers. In this case, even if other states are mapped to each other, the two MDPs cannot be considered similar. An example is presented as follows. Figure 1 shows two simple maze tasks and their MDPs, where S and S' are start states, G and G' are target states, and the black grid is an obstacle. We can see that the only difference between the two tasks is that Task II has an obstacle but Task I does not. The MDPs below the mazes show the state transitions by arrows. Actions and rewards of the transitions are labeled in the braces. The distances between states of different MDPs are computed using Equation (2). The results are shown in Table 1. It is easy to get that $\Phi = 20$. R_1 is an "outlier" in task I. The distance between two MDPs computed by Hausdorff metric is equal to $\min_s d'(s, R_1)$ and it is too large to consider the two MDPs similar.



(a) Task I (b) Task II

(c) MDP for task I (d) MDP for task II

Figure 1: Example of measuring the distance between two MDPs

Table 1: Distance between states

distance	S'	R'	G'
S	10	20	20
R_1	20	20	20
R_2	20	5	20
G	20	20	0

To address the limitation of the above measure, we propose a new measure based on Kantorovich metric, which is originally used to find an optimal allocation of transportation resources. In a transportation problem, there are piles of sand (supply nodes) and holes (demand nodes) of the same volume, and the goal is to find an optimal way to move the sand into the holes. Kantorovich metric is an LP method for solving this problem to reach the goal of getting the minimum cost. When it is applied to two MDPs, one MDP's states are considered as the supply nodes and the other's are considered as demand nodes. Suppose that there is a transportation arc from each supply node to each demand node. A flow is an assignment of quantities to be transported along each arc. In general, the total supply and the total demand are equal, hence we set them to 1. The supply quantity of each supply node is $1/|S_1|$ with the assumption that each state is equally important to the MDP, where $|S|$ is the size of the state space of the MDP. The demand quantity of each demand node is $1/|S_2|$ as well. The unit cost of each arc equals to the distance of the corresponding states. Then transportation expense of each arc is

the unit cost multiplied by the value of the flow along that arc. The goal is to get the optimal assignment of the flows to minimize the total transportation expenses. It can be seen that the distance of the outlier (if exists) only affects the transportation expense of some flows, so it has less impact on the total transportation expenses.

Now we give the formal definition of the distance between two MDPs based on Kantorovich metric:

DEFINITION 3.6. *Given two MDPs* $M_1 = \{S_1, A, P_1, R_1\}$ *and* $M_2 = \{S_2, A, P_2, R_2\}$*, their Kantorovich distance* $\Psi(S_1, S_2)$ *can be computed as*

$$\begin{aligned} \Psi(S_1, S_2) = \min_{l_{ij}} & \sum_{i=1}^{|S_1|} \sum_{j=1}^{|S_2|} l_{ij} d'(s_i, s'_j) \\ \text{s.t.} \quad & \sum_{j=1}^{|S_2|} l_{ij} = \frac{1}{|S_1|}, \forall i, \\ & \sum_{i=1}^{|S_1|} l_{ij} = \frac{1}{|S_2|}, \forall j, \\ & l_{ij} \geq 0, \forall i, j, \end{aligned} \tag{5}$$

where $d'(s_i, s'_j)$ *is the state distance between* s_i *and* s'_j *and can be obtained according to Equation (2). Obviously, the value of* $\Psi_i(S_1, S_2)$ *can be computed by solving an LP.*

The cost of computing the Kantorovich metric is $O(|S_1 + S_2||S_1||S_2| \log(S_1 + S_2))$.

We use the above example to illustrate the whole process of computing the distance between two MDPs (see Figure 1 and Table 1). After computing the distances between states, Kantorovich metric is applied to compute the distance between two MDPs. The illustration of the flows is shown in Figure 2. The numbers beside the nodes are supply nodes' supply quantities or demand nodes' demand quantities. The numbers on the bold arrows are the values of flows leaving from the supply states to the demand states, and the values of other flows are zero which are omitted in the figure. The values of flows are l_{ij} in Equation (5). The distance between two MDPs computed by Kantorovich metric is $\Psi = 8.75$, and is much smaller than Φ. We can see from Figure 2, in Kantorovich metric the outlier's distance multiplies a weight to reduce the effect of that.

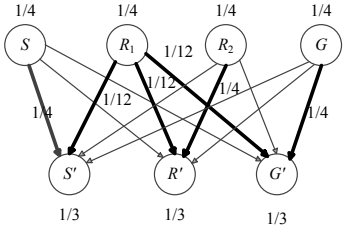


Figure 2: The transition between two MDPs

4. TRANSFER METHODS

Transfer learning for RL aims to improve the agent's learning performance by transferring the knowledge from prior learned task(s). The metrics for measuring the distance between MDPs can be used for automated source tasks selection in transfer learning, after which the knowledge learned in the selected source tasks can be transferred to the target task. The knowledge transfer mechanism adopted is value function transfer. In this section, we propose two transfer methods, one for Kantorovich metric (Ψ) only and the other for both metrics, Kantorovich metric and Hausdorff metric (Φ). We define the transfer method in two scenarios: transferring from one source task (single source transfer) and transferring from multiple source tasks (multiple source transfer).

4.1 Weight Transfer

We first introduce the transfer method (called *weight transfer*) which is only used when metric Ψ is adopted to measure the distance between MDPs. For each state of the target task, each weight l_{ij} defined in Definition 3.6 indicates how similar it is to the corresponding state in the source task. The more similar the two states are, the larger l_{ij} is. Therefore, we can transfer value functions (or Q-values) from states of the source task(s) to states of the target task according to the values of l_{ij}.

The transfer method for transferring from multiple source tasks is defined as follows.

DEFINITION 4.1. *Given the target task's MDP* $M : \{S, A, R, P\}$ *and* n $(n \geq 1)$ *source MDPs* $M_1 : \{S_1, A, R_1, P_1\}, M_2 : \{S_2, A, R_2, P_2\}, \ldots, M_n : \{S_n, A, R_n, P_n\}$*, the value functions of* M *can be initialized by the* n *source tasks' value functions. For every* $s_i \in S$*,*

$$V(s_i) = \frac{1}{n} \sum_{k=1}^{|n|} \sum_{j=1}^{|S_k|} \frac{l_{ij}^k}{L_j^k} V^k(s_j^k), \tag{6}$$

where s_j^k *is the* j*-th state in* S_k*,* l_{ij}^k *is the weight of* $d'(s_i, s_j^k)$ *in* $\Psi(S, S_k)$*,* $L_j^k = \sum_{i=1}^{|S|} l_{ij}^k$*, and* $V^k(s_j^k)$ *is the value function of the state* s_j^k *in source task* M_k *which has been computed.*

Similarly, Q-values of M *can be initialized by the following formula:*

$$Q(s_i, a) = \frac{1}{n} \sum_{k=1}^{|n|} \sum_{j=1}^{|S_k|} \frac{l_{ij}^k}{L_j^k} Q^k(s_j^k, a), \tag{7}$$

where $Q^k(s_j^k, a)$ *is the Q-value of the state-action pair* (s_j^k, a) *in source task* M_k *which has been learned.*

We choose the most similar n $(n \geq 1)$ source tasks, and transfer their average weighted value functions (or Q-values) to the target task. The value of n depends on specific problems, and the weights for the transfer are proportional to l_{ij}. When $n = 1$, Equations (6) and (7) can be used to single source transfer.

The above transfer methods may be applied to algorithms which learn value functions or action value functions to solve the RL tasks. However, we only consider value iteration [23] and Q-learning [31] in this paper.

Algorithm 2 shows the pseudo code of transferring value functions from m $(m \geq 1)$ source tasks $M_k (k = 1, \ldots, m)$ to the target task M (the process of transferring Q-values is similar). The whole process of transfer learning consists of three steps: (1) compute the distance $\Psi(S, S_l)$ $(l = 1, \ldots, m)$ between the target task and each source task according to Equation (5), and then sort the source tasks according to the computed distances in ascending order; (2) select the most similar n source tasks M_{k_s} $(s = 1, \ldots, n)$ and load the

weight matrix $l_{ij}^{k_s}$ and their value functions (or Q-values); (3) transfer their value functions (or Q-values) to the target task (Lines 3-8), then use value iteration (or Q-learning) algorithm to solve the target task (Line 9).

Algorithm 2: Transfer value functions and learning

Input: discount rate γ, the number of transferred source tasks n, coefficient matrix l_{ij}^k and value functions $V^k(s^k)$ of the selected source tasks $M_k(k = 1 \dots n)$.

1 Initialization. weight $\omega = 0$,
2 value functions of the target task: $V(s) = 0$.
3 **foreach** $s_i \in S$ **do**
4 **foreach** M_k **do**
5 **foreach** $s_j^k \in S_k$ **do**
6 $\omega = \frac{l_{ij}^k}{L_j^k}$;
7 $V(s_i) = V(s_i) + \omega V^k(s_j^k)$;
8 $V(s_i) = \frac{V(s_i)}{n}$.
9 Solve the target task with value iteration.

4.2 State transfer

The weight transfer method can be only used for metric Ψ since it needs the coefficient matrix l_{ij}^k. Now we propose another method, called *state transfer*, which can be used for both Ψ and Φ. This method is also based on the idea that transfer can occur between similar states. For each state in the target task, its nearest state can be found in the source task after the distances of states are computed, and we directly transfer the value functions (or Q-values) of the nearest state in the source task to it.

Definition 4.2. *Given the target task's MDP* $M_1 = \{S_1, A, R_1, P_1\}$ *and the source task's MDP* $M_2 = \{S_2, A, R_2, P_2\}$, *the value function of each state* s *in* M_1 *is initialized by the following equations:*

$$s' = \arg\min_{s'} d'(s, s'),$$
$$V(s) = V(s').$$

Similarly, Q-values of M_1 *can be initialized by:*

$$Q(s, a) = Q(s', a).$$

In multiple source transfer situation, we transfer the average value functions (or Q-values) of the source tasks' corresponding states to the target task.

5. EXPERIMENTAL EVALUATION

We evaluate our metrics in transfer learning for RL. We consider the maze problem in which many transfer learning methods have been applied [8, 9, 20]. An example of the mazes used in our experiment is shown in Figure 3. 'S' is the start position and 'G' is the goal position. The black squares are called obstacles that block agent to go through, and other things are the same as Gridworld [25]. The objective of the learning agent is to find an optimal path to reach the goal position from the start position. In each position, there are four actions, *up*, *down*, $left$, and *right*, for the agent to choose. After choosing an action, agent deterministically gets to the corresponding neighbor position except when a movement is blocked by an obstacle or the edge of the maze,

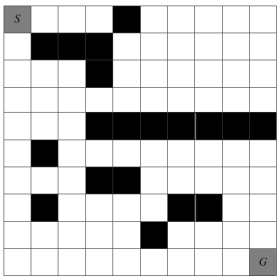


Figure 3: A 10×10 maze

in which case the agent will stay in its current position, i.e., the (illegal) movement causes no effect. Reward is 100 when arriving at the goal position and 0 in other cases.

We adopt value iteration and Q-learning as the basic learning methods. The transfer learning setting is as follows: There is a library of source tasks and when given a target task we hope to transfer knowledge from the library to improve the learning performance of the target task. In this paper, the maze tasks are homogeneous. The tasks in the library have been learned by the basic learning methods, and their models and learned knowledge (i.e. value functions or Q-values) have been stored. The target task will be also learned by the basic learning methods.

The parameters of the methods are set as follows.

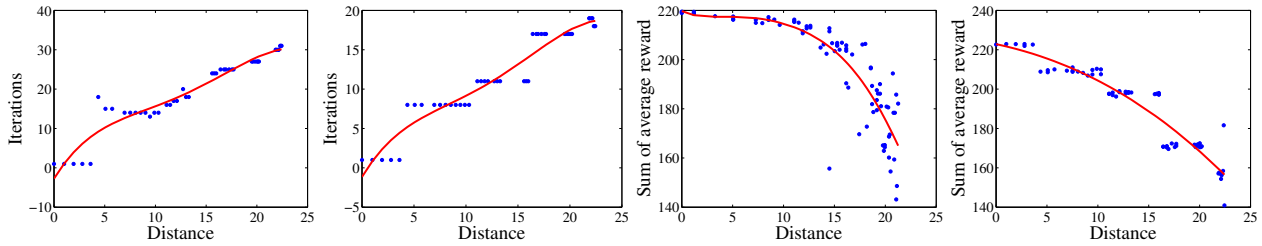
1) We set parameter $c = 0.5$, $\delta = 1$, and the threshold $\xi = 0.01$ when computing the distance of two states in different MDPs (see Algorithm 1).

2) The discount factor γ is set to 0.9.

3) The Q-learning agent behaves in an ϵ-*greedy* way, i.e., it selects the best action (indicated by the largest Q-values) with probability $1 - \epsilon$, and selects a random action with the rest probability of ϵ. The exploration probability ϵ is set to 0.1 and the learning factor α is set to 0.2.

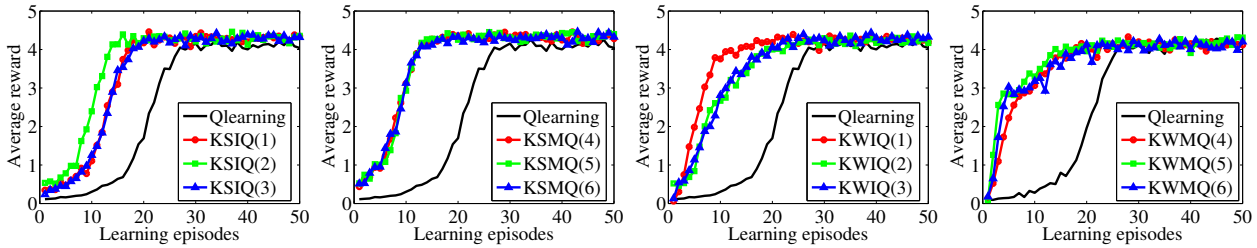
We have conducted two sets of experiments: one is for evaluating the property of the proposed metrics, i.e., the relation between transfer performance and task distance, and the other is for investigating whether the similar source task(s) selected by our metrics can accelerate the learning of the target task.

For the first set of experiments, we randomly generate a 10×10 maze (the target task) and a series of 10×10 mazes (the source tasks) with different distances to the target maze. We use metrics Ψ and Φ to measure the distance between the target task and each source task, after which we record the distances of each pair of the tasks. Then, we transfer the knowledge of all the source tasks to the target task using single source transfer methods. We have two metrics, two transfer methods and two learning methods. To keep it simple, we use letters "K", "H", "W", "S", "V" and "Q" to represent metric Ψ, metric Φ, weight transfer method, state transfer method, value iteration and Q-learning. The combination of these letters represents the combination of methods, e.g., KSV means using metric Ψ, state transfer and value iteration to solve the transfer learning task. In Figure 4, four scatter diagrams and their rough fitting curves are shown.



(a) KWV (b) KSV (c) KWQ (d) KSQ

Figure 4: The relation between transfer performance and distance computed by metric Ψ



(a) KSIQ (b) KSMQ (c) KWIQ (d) KWMQ

Figure 5: Transferring from the most similar tasks using metric Ψ ($\epsilon = 0.1$)

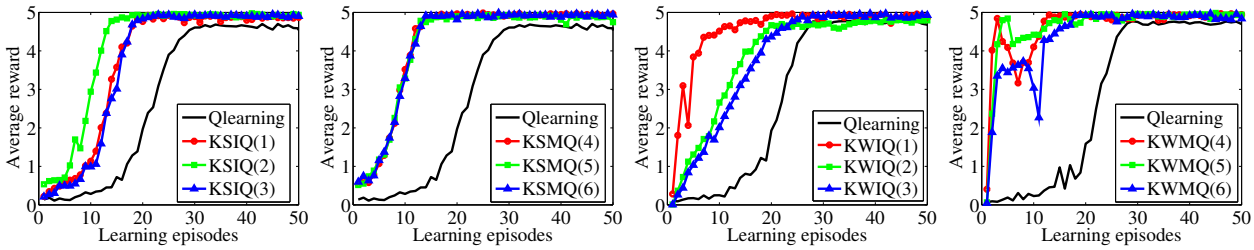
The horizontal axis represents the distance between the source task and the target task. In subfigures 4(a) and 4(b), vertical axis represents the number of iterations, and in subfigures 4(c) and 4(d) vertical axis represents the sum of the average rewards of each step (we just call it average rewards in the following sections) of the first 50 episodes. The target task's learning process is repeated 50 times and the number of iterations (or the sum of the average rewards) is the average values of these processes. For value iteration, we use the number of iterations to measure the learning performance of the task, the smaller the better. For Q-learning, we use the sum of average rewards of the first 50 episodes to measure the learning performance of the task, the more the better. We use "50 episodes" since the processes of Q-learning have often converged within 50 episodes in this experiment.

We can see from the figures that the performance of transfer is related to the distance computed by metric Ψ. In subfigures 4(a) and 4(b), the number of iterations to learn a task increases as the distance increases. The sum of average rewards decreases as the distance increases in subfigures 4(c) and 4(d). This means the performance of transfer learning improves as the similarity increases. The source tasks with distance less than 5 have the similar distributions (numbers and positions) of obstacles with the target task while the tasks with distance near 25 have quite different ones. Therefore, metric Ψ can describe task similarity in transfer learning correctly, and negative transfer can be avoided by filtering the dissimilar tasks. However, metric Φ does not have such property. The reason may be that it may miss some actually similar MDPs due to "outliers". Although metric Φ does not has such property, the similar tasks selected by it can improve the learning performance in the target task, which will be demonstrated in the next set of experiments.

For the second set of experiments, the target task is a randomly generated 10×10 maze and the source tasks consist of 100 10×10 mazes which have several obstacles that are generated randomly in the grids. After computing the distance between the target task and each source task by our metrics Ψ and Φ, the source tasks are sorted according to the distances in ascending order, and labeled by sequence numbers according to the order. The smaller the number is, the more similar the task is to the target task. The target task is learned in many situations. First it is learned by value iteration (or Q-learning), and then single source transfer and multiple source transfer are used to alter its learning process. For single source transfer scenario, the most similar source task's knowledge is transferred using our single source transfer method. The multiple source transfer scenario is similar, in which knowledge from the most similar n (we consider some different n in this section) source tasks is transferred. The Q-learning agent behaves in an ϵ-*greedy* way, and we test two situations, $\epsilon = 0.1$ and $\epsilon = 0.01$.

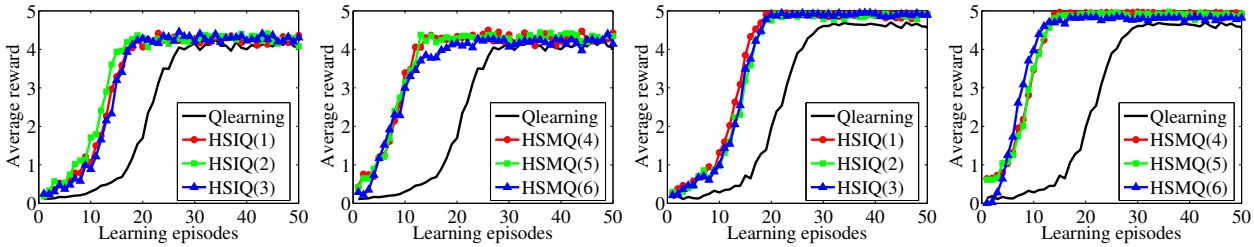
We consider all the combinations of metrics, transfer methods and learning methods. We use the combinations of letters to represent the combination of methods. We add letter "I" and "M" to represent single source transfer and multiple source transfer. The "I" and "M" are put before the learning method ("Q" or "V"), e.g., KWIQ means using metric Ψ, single source weight transfer and Q-learning to solve the transfer learning task. The results of Q-learning are shown in Figures 5, 6 and 7.

The horizontal axis x represents the number of learning episodes and the vertical axis y represents the average rewards in episode x. The learning process is repeated 50 times and the average rewards are the average values of these processes. In the figures, for single source transfer, the number n in the parentheses after the combinations of letters means transferring from source task with sequence number



(a) KSIQ (b) KSMQ (c) KWIQ (d) KWMQ

Figure 6: Transferring from the most similar tasks using metric Ψ ($\epsilon = 0.01$)



(a) HSIQ ($\epsilon = 0.1$) (b) HSMQ ($\epsilon = 0.1$) (c) HSIQ ($\epsilon = 0.01$) (d) HSMQ ($\epsilon = 0.01$)

Figure 7: Transferring from the most similar tasks using metric Φ

n. For multiple source transfer, the number n in the parentheses means transferring from n most similar source tasks. For example, KWIQ(1) means using metric Ψ, single source weight transfer and Q-learning, and transferring from the source task with sequence number 1 to solve the transfer learning task.

We can see from the figures that the similar tasks selected by our metrics can speed up the learning process of the target task. Moreover, when $\epsilon = 0.01$, transfer learning can improve the asymptotic performance as shown in Figure 6 and Figures 7(c) and 7(d). The curves of transferring from different source tasks in each figure are very close since the distances are close. Therefore, our metrics can perform well in helping finding the similar tasks in transfer learning to improve the learning performance in the target task. We can also see from Figures 5(a), 6(a), 7(a) and 7(c) that, although source tasks selected by Hausdorff metric can benefit learning in the target task, the really similar tasks (e.g. the green curves in Figures 5(a) and 6(a)) may be missed. However, Kantorovich metric can find them.

The results of value iteration are shown in table 2. .

Table 2: The number of iterations in single source transfer

No	*HSIV*	*KSIV*
1	1	1
2	11	11
3	14	15
4	14	14

The column "No" in the table represents the sequence number of the transferred source task. In addition to this column, other columns are numbers of iterations using the transfer learning methods. Value iteration for the target task without transfer can converge after 20 iterations. We can see from the table that HSIV and KSIV can accelerate the process of value iteration. We also test the multiple source transfer scenario, and multiple source transfer methods cannot improve the learning performance. As a result, single source state transfer methods can be applicable to value iteration, and our metrics can perform well in helping finding the similar tasks when using single source state transfer methods.

6. CONCLUSION

In this work, we propose two metrics for measuring the distance between MDPs. We extend Kantorovich metric and Hausdorff metric to define the distance between MDPs. We show that the metrics can be applied in transfer learning for RL where tasks are modeled as MDPs. The metrics can help with selecting similar tasks to effectively transfer and avoid negative transfer. Since the metrics are based on the distances between states, value functions (or action-value functions) are used as the transfer knowledge and we propose two value function transfer methods. Experimental results show that the metrics are effective in finding similar tasks to avoid negative transfer, and the learning algorithms are significantly improved by our metrics and transfer methods.

In the future, we hope to generalize our metrics to other settings, where MDPs are not homogeneous. Our metrics are defined on MDPs with finite state space, and we are trying to establish an extension for infinite state space.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge the support for this work from the National Natural Science Foundation of China (Grant Nos. 61432008, 61503178, 61321491) and the Natural Science Foundation of Jiangsu Province of China (BK20150587).

REFERENCES

[1] Pac-man. http://en.wikipedia.org/wiki/Pac-Man, 2014.

[2] H. B. Ammar, E. Eaton, M. E. Taylor, D. C. Mocanu, K. Driessens, G. Weiss, and K. Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the 28th AAAI Conference on Artificial Intelligence*, 2014.

[3] R. Bellman. Dynamic programming. *Princeton University Press*, 1957.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[5] D. Braun, J. Mayberry, A. Powers, and S. Schlicker. The geometry of the hausdorff metric. http://faculty.gvsu.edu/schlicks/Hausdorff_paper.pdf, 2003.

[6] J. L. Carroll and K. Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings of the IEEE International Joint Conference on Neural Network*, pages 803–808, 2005.

[7] Y. Deng and W. Du. The kantorovich metric in computer science: A brief survey. *Electronic Notes in Theoretical Computer Science*, 253(3):73–82, 2009.

[8] K. Ferguson and S. Mahadevan. Proto-transfer learning in markov decision processes using spectral methods. In *Proceedings of the 23rd ICML Workshop on Transfer Learning*, pages 895–900, 2006.

[9] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 720–727, 2006.

[10] N. Ferns. Metrics for markov decision processes. Master's thesis, McGill University, 2003.

[11] N. Ferns, P. Castro, P. Panangaden, and D. Precup. Methods for computing state similarity in markov decision processes. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*, pages 174–181, 2006.

[12] N. Ferns, P. Panangaden, and D. Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*, pages 162–169, 2004.

[13] N. Ferns, P. Panangaden, and D. Precup. Metrics for markov decision processes with infinite state spaces. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 201–208, 2005.

[14] A. Haitham, Bou, E. Eric, R. Paul, and T. Matthew. Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1206–1214, 2014.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[16] G. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

[17] G. Konidaris, I. Scheidwasser, and A. G. Barto. Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 13(1):1333–1371, 2012.

[18] A. Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[19] A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 544–551, 2008.

[20] H. Li, X. Liao, and L. Carin. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 10:1131–1186, 2009.

[21] M. G. Madden and T. Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21(3–4):375–398, 2004.

[22] D. N. Perkins and G. Salomon. Transfer of learning. *International encyclopedia of education*, 2:6452–6457, 1992.

[23] M. L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. *Journal of the Operational Research Society*, 46(6):792–793, 1995.

[24] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *NIPS Workshop on Inductive Transfer*, 2005.

[25] R. S. Sutton and A. G. Barto. Reinforcement learning: an introduction. *IEEE Transactions on Neural Network*, 9(5):1054–1054, 1998.

[26] M. E. Taylor and P. Stone. Representation transfer for reinforcement learning. In *AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development*, 2007.

[27] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[28] M. E. Taylor and P. Stone. An introduction to intertask transfer for reinforcement learning. *AI Magazine*, 32(1):15, 2011.

[29] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[30] L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Skill acquisition via transfer learning and advice taking. In *Proceedings of the 17th European Conference on Machine Learning*, pages 425–436, 2006.

[31] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.