Reinforcement Learning Transfer via Sparse Coding

Haitham B. Ammar
Maastricht University, The Netherlands
haitham.bouammar@maastrichtuniversity.nl

Karl Tuyls
Maastricht University, The Netherlands
k.tuyls@maastrichtuniversity.nl

Matthew E. Taylor
Lafayette College, USA
taylorm@lafayette.edu

Kurt Driessens
Maastricht University, The Netherlands
kurt.driessens@maastrichtuniversity.nl

Gerhard Weiss
Maastricht University, The Netherlands
gerhard.weiss@maastrichtuniversity.nl

ABSTRACT

Although reinforcement learning (RL) has been successfully deployed in a variety of tasks, learning speed remains a fundamental problem for applying RL in complex environments. Transfer learning aims to ameliorate this shortcoming by speeding up learning through the adaptation of previously learned behaviors in similar tasks. Transfer techniques often use an inter-task mapping, which determines how a pair of tasks are related. Instead of relying on a hand-coded inter-task mapping, this paper proposes a novel transfer learning method capable of autonomously creating an inter-task mapping by using a novel combination of sparse coding, sparse projection learning and sparse Gaussian processes. We also propose two new transfer algorithms (*TrLSPI* and *TrFQI*) based on least squares policy iteration and fitted-Q-iteration. Experiments not only show successful transfer of information between similar tasks, inverted pendulum to cart pole, but also between two very different domains: mountain car to cart pole. This paper empirically shows that the learned inter-task mapping can be successfully used to (1) improve the performance of a learned policy on a fixed number of environmental samples, (2) reduce the learning times needed by the algorithms to converge to a policy on a fixed number of samples, and (3) converge faster to a near-optimal policy given a large number of samples.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Transfer Learning, Reinforcement Learning, Sparse Coding, Inter-task mapping, Sparse Gaussian Processes, Optimization

1. INTRODUCTION

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

Transfer learning is the field that studies how to effectively leverage knowledge learned from one or more source tasks when learning one or more target tasks. Although TL has been widely studied within the supervised learning framework (e.g., [9, 10]), TL in RL domains only recently started to gain interest and is very much in flux [12, 18, 21].

Reinforcement learning (RL) is a popular framework that allows agents to solve sequential decision making problems with minimal feedback. Unfortunately, RL agents may learn slowly in large or complex environments due to the computational effort needed to converge to an acceptable performing policy. TL is one technique that copes with this difficulty by providing a good starting policy or prior for the RL agent.

Typically, the source and target task are different but related. In RL settings, the source and target tasks have different representations of state and action spaces, requiring a mapping between the tasks. An inter-task mapping matches each state/action pair of the source task to its correspondance in the target task. Two issues stand out in current TL for RL research. First, although there have been a number of successes in using an inter-task mapping for TL, the mappings are typically *hand-coded* and may require substantial human knowledge [19]. Two fundamental open questions are, first, to what extent it is possible to learn the mapping automatically and, second, how should an inter-task mapping be best leveraged to produce successful transfer?

We tackle these problems and make a number of contributions. The primary contribution is a novel method to automatically learn an inter-task mapping between two tasks based on sparse coding, sparse projection learning, and sparse Gaussian processes. More specifically, we define an inter-task mapping to be a function that relates state-action successor state triplets from the source task to the target task. This inter-task mapping is more than a one-to-one mapping between the state and/or action spaces of the MDPs, as it also includes non-linear terms that are automatically discovered by global approximators, which ultimately enhance the efficacy of transfer. This inter-task learning framework can be split into three essential parts. The first is a dimensional mapping of both the source and target task state-action spaces of the MDPs. The second is the automatic discovery of a high dimensional informative space of the source task. This is achieved through sparse coding, as described in Section 4.1.2, ensuring that transfer is conducted in a high representational space of the source task. In order to use a similarity measure among different patterns, the data should be present in the same space, which is why the target task triplets need to be projected to the high representational space of the source (done via sparse projection learning, described in Section 4.2). The third and final step is to approximate the inter-task mapping via a non-parametric regression technique, explained in Section 4.3.

Another contribution is to introduce two new algorithms for transfer between tasks of continuous state spaces: Transfer Least Squares Policy Iteration (TrLSPI) and Transfer Fitted-Q-Iteration (TrFQI). Experiments illustrate the feasibility and suitability of the presented approach, and furthermore show that this introduced method can successfully transfer between two different RL benchmarks: transfer is successful between the inverted pendulum and the cart pole, as well as between the mountain car and the cart pole.

2. RELATED WORK

This section presents a selection of related work, focused on transfer learning for reinforcement learning tasks.

TL in RL has been of growing interest to the agents community, due in part to its many empirical successes at significantly improving the speed and/or quality of learning [18]. However, the majority of existing work assumes that 1) the source task and target task are similar enough that no mapping is needed, or 2) an inter-task mapping is provided to the agent.

For example, many authors have considered transfer between two agents which are similar enough that learned knowledge in the source task can be directly used in the target task. For instance, the source and target task could have different reward functions (e.g., *compositional learning* [13]) or have different transition functions (e.g., changing the length of a pole over time in the cart pole task [12]). More difficult are cases in which the source task and target task agents have different state descriptions or actions. Some researchers have attempted to allow transfer between such agents without using an inter-task mapping. For example, a shared *agent space* [3] may allow transfer between such pairs of agents, but requires the agents to share the same set of actions, and requires an agent-centric mapping. Other approaches assume that an inter-task mapping is provided, such as Torrey *et al.* [21] who transfer advice between agents and Taylor *et al.* [19] who transfer Q-value functions by leveraging an existing inter-task mapping.

The primary contrast between these methods and the current work is that we are interested in *learning* a mapping between states and actions in pairs of tasks, rather than assuming that it is provided, or rendered unnecessary because of similarities between source task and target task agents, a requirement for fully autonomous transfer.

There has been some recent work on learning such mappings. For example, semantic knowledge about state features between two tasks may be used [4, 8], background knowledge about the range or type of state variables can be used [16, 20], or transition models for each possible mapping could be generated and tested [17]. However, there are currently no general methods to learn an inter-task mapping without requiring 1) background knowledge that is not typically present in RL settings, or 2) an expensive analysis of an exponential number (in the size of the action and state variable sets) of inter-task mappings. This paper overcomes these problems by automatically discovering high-level features and using them to transfer knowledge between agents without suffering from an exponential explosion. Others have focused on transferring samples between tasks. For instance, Lazaric et al. [6] transfers samples in batch reinforcement learning using a compliance measure. The main difference to this work is that we neither assume any similarities between the transition probabilities, nor restrict the framework to similar state and/or action feature representations.

In contrast to all existing methods (to the best of our knowledge), this paper allows for differences between all variables describing Markov decision processes for the source and target tasks and learns an *inter-task mapping*, rather than a mapping based on state features. Furthermore, the framework introduced in this paper can use *state-dependent* action mappings, allowing additional flexibility.

3. PRELIMINARIES

This section briefly covers reinforcement learning and sparse coding, introducing the necessary notation for the rest of the paper. Section 4 will draw the connection between the two and introduce the main contribution of the paper.

3.1 Reinforcement Learning (RL)

In an RL problem, an agent must decide how to sequentially select actions to maximize its expected return [1, 15]. Such problems are typically formalized as Markov decision processes (MDPs), defined by $\langle S, A, P, R, \gamma \rangle$. S is the (potentially infinite) set of states, A is the set possible actions that the agent may execute, $P : S \times A \times S \rightarrow [0, 1]$ is a state transition probability function, describing the task dynamics, $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function measuring the performance of the agent, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : S \times A \rightarrow [0, 1]$ is defined as a probability distribution over state action pairs, where $\pi(s, a)$ represent the probability of selecting action a in state s. The goal of an RL agent is to improve its policy, potentially reaching the optimal policy $\pi^\star$ which maximizes cumulative future rewards. It can be attained by taking greedy actions according to the optimal Q-function $Q^\star(s, a) = \max_\pi E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, a = a_0]$. In tasks with continuous state and/or action spaces, Q and π cannot be represented in a table format, typically requiring sampling and function approximation techniques. This paper uses two such techniques, *Least Squares Policy Iteration* (LSPI) and *Fitted-Q-Iteration* (FQI), discussed later.

3.2 Sparse Coding

Sparse coding (SC) [7] is an unsupervised feature extraction technique that finds a high-level representation for a set of unlabeled input data by discovering a succinct, over-complete basis for the provided data set.

Given a set of m k-dimensional vectors, $\boldsymbol{\zeta}$, SC aims to find a set of n basis vectors, $\mathbf{b}$, and activations, a, with $n > k$ such that $\boldsymbol{\zeta}^{(i)} \approx \sum_{j=1}^{n} a_j^{(i)} \mathbf{b}_j$, where i and j represent the number of input data patterns and number of bases, respectively. SC begins by assuming a Gaussian and a sparse prior on the reconstruction error ($\boldsymbol{\zeta}^{(i)} - \sum_{j=1}^{n} a_j^{(i)} \mathbf{b}_j$) and on the activations, solving the following optimization problem:

$$\min_{\{\mathbf{b}_j\}, \{a_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\boldsymbol{\zeta}^{(i)} - \sum_{j=1}^{n} \mathbf{b}_j a_j^{(i)}||_2^2 + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} ||a_j^{(i)}||_1 \tag{1}$$

$$s.t. \ ||\mathbf{b}_j||_2^2 \leq c, \forall j = \{1, 2, \ldots, n\}$$

The problem presented in Equation 1 is considered to be a "hard" optimization problem as it is not jointly convex (in the activations and bases). However, fast and efficient optimization algorithms exist [7] and were used, as described in Section 4.1.

4. LEARNING AN INTER-TASK MAPPING

In this section, we cast the problem of learning the inter-task mapping, χ, as a supervised learning problem. We define χ to be a mapping of state-action-state triplets from the source task to the target task. Learning such a mapping requires related triplets from both tasks as data points for training. Unfortunately, obtaining such corresponding triplets is itself a hard problem — it is not trivial for a user to describe which triplets in the source task correspond to which triplets in the target task.

To automatically construct these data points, we propose a novel framework utilizing sparse coding together with an L_1 projection scheme.

We approach the problem by automatically transforming the source task space (i.e., state-action-state space) into a higher representational space through SC, followed by a projection of the target task triplets onto the attained bases. We then use a simple Euclidean distance measure[1] to gauge similarity (Section 4.1). At this stage, the data set is ready to be provided to the learning algorithm so that it may construct the inter-task mapping.

The following sections further clarify each of the above steps and explain the technicalities involved.

4.1 Sparse Coding Transfer for RL

As described in Section 3.2, SC is an efficient way to discover high-level features in an unlabeled data set. First, SC learns how best to match the dimensions of the two different MDPs. Second, SC discovers higher-level features for the low dimensional task's state-action-state space.

4.1.1 Mapping the Source and Target Dimensions

To generate triplet matchings through SC, the first step is to match the dimensions of the state-action-state spaces of the source and target MDPs, which are likely to be different. In principle, after this step any existing TL in RL technique can be used. However, this paper goes further and proposes a new transfer framework based on the discovered bases and activations, described in Section 5.

This "dimensional mapping" process is summarized in Algorithm 1. In short, it sparse codes random triplets $\langle s_0, a_0, s_0^{'}\rangle$ from the task with the lowest dimensionality, constrained to learn a number of bases (d_1) equal to the higher dimension of the unmodified task (regardless of which is the source and target task). We use existing algorithms [7] to solve the Equation from step 2 of Algorithm 1.

4.1.2 High Information Representation

After mapping the source and target dimensions as described in the previous section, SC is again used to discover a succinct higher feature bases of the activations than the unified dimensional spaces that were discovered in Section 4.1.1. If successful this step will discover new features in the source task that could better represent relations with the target task than the bases discovered in Section 4.1.1. Algorithm 2, similar in spirit to Algorithm 1, describes this process.

Algorithm 2 sparse codes the activations, which represent the original source task triplets of the MDPs, to a higher representational space, d_n.[2] This stage should guarantee that we project the

Algorithm 1 Sparse Coding TL for RL

```
Require: Triplets $\{\langle s_0, a_0, s_0^{'}\rangle\}_{i=1}^{m}$ and $\{\langle s_1, a_1, s_1^{'}\rangle\}_{j=1}^{f}$ from
    both MDPs
1: Determine $d_0$ and $d_1$, the dimensions of the state-action-state
    spaces for both MDPs, where $d_0 \leq d_1$
2: Sparse code the lower dimensional triplets by solving:
    $\min_{\{\mathbf{b}_j\},\{a_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\langle s_0, a_0, s_0^{'}\rangle^{(i)} - \sum_{j=1}^{d_1} \mathbf{b}_j a_j^{(i)}||_2^2 + \beta \sum_{i=1}^{m}\sum_{j=1}^{d_1} ||a_j^{(i)}||_1$
    $s.t.\ ||\mathbf{b}_j||_2^2 \leq c, \forall j = \{1, 2, \ldots, d_1\}$
3: Solve the above equation by using an existing algorithm [7]
4: Return the Activation matrix ($\mathbf{A} \in \mathbb{R}^{m\times d_1}$) and the Bases ($\mathbf{B} \in$
    $\mathbb{R}^{d_1\times d_0}$)
```

Algorithm 2 Succinct High Information Representation of MDPs

```
Require: Activations $\mathbf{A}$ from Algorithm 1, Target number of new
    high dimensional bases $d_n$
1: Represent the activations in the $d_n$ bases by solving the follow-
    ing problem (again using the algorithm in [7]):
2:
    $\min_{\{\mathbf{z}_j\},\{c_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\langle \mathbf{a}_{1:d_1}\rangle^{(i)} - \sum_{j=1}^{d_n} \mathbf{z}_j c_j^{(i)}||_2^2$
    $+\beta \sum_{i=1}^{m}\sum_{j=1}^{d_n} ||c_j^{(i)}||_1$
    $s.t.\ ||\mathbf{z}_j||_2^2 \leq o, \forall j = \{1, 2, \ldots, d_n\}$
3: return Return new activations $\mathbf{C} \in \mathbb{R}^{m\times d_n}$ and bases $\mathbf{Z} \in$
    $\mathbb{R}^{d_n\times d_1}$
```

triplets of the source task MDP into a high feature space where a similarity measure can be used to find a relation between the source and target task triplets. Note that there are no restrictions on the number of bases: unneeded bases will end up with an activation of zero.

Algorithm 2 discovers new features in the source or target state-action-state spaces. As TL typically transfers from a low dimensional source task to a high dimensional target task, SC determines new bases that are of a higher dimensionality than the original representation used for states and actions in the source task. These newly discovered bases can describe features not anticipated in the original design of the MDP's representation. These new features can highlight similarities between the source and target task thus helping and guiding the transfer learning scheme. The re-encoded triplets—described as a linear combinations of the bases and activations (i.e., **AB**)—do not yet relate to the triplets of the other task. The target task triplets still need to be projected towards these new sparse coded source task bases features. This is done as described in Section 4.2.

4.2 L_1 Sparse Projection Learning

Once the above stages have finished, the source task triplets are described via the activations **C** (generated in Algorithm 2). However, target task triplets still have no relationship to the learned activations. Since we are seeking a similarity correspondence between

[1]A simple Euclidian distance might not be optimal, but optimizing this measure is planned as future research. Experiments show that more than reasonable results can be attained using this simple approximation.

[2]In our experiments we have set d_n to be 100, a relatively high number.

the source and target task triplets, the target task triplets should be represented in the same high informational space of the source task. Therefore, the next step is to learn how to project the target task triplets onto the **Z** basis representation. The overall scheme is described in Algorithm 3, where the activations are learned by solving the L_1 regularized least squares optimization problem in step 2. This optimization problem guarantees that the activations are as sparse as possible and is solved using the interior point method [2]. The next step will be to order the data points from both the source and the target tasks, which are then used to approximate the inter-task mapping.[3]

4.3 Approximating an Inter-Task Mapping

To finalize the problem of approximating χ, corresponding triplets from the source and target task should be provided to a regressor. We approach this problem by using a similarity measure in the high feature space, **Z**, to identify similar triplets from the two tasks. This similarity measure identifies triplets from the source task that are most similar to those of the target task, and then map them together as being inputs and outputs for the regression algorithm, respectively, as shown on line 2 of Algorithm 4. Since the similarity measure is used in the sparse coded spaces, the distance is calculated using the attained activation (**C** and **Φ**) rather than the triplets themselves. Therefore, the scheme has to trace the data back to the original dimensions of the state-action pairs of the MDPs.

Algorithm 3 Mapping Target Task Triplets

```
Require: Sparse Coded Bases Z generated by Algorithm 2, Target
    MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{i=1}^{f}$
1: for $i = 1 \rightarrow f$ do
2:     Represent the target data patterns in the sparse coded bases,
       Z, by solving:
       $\hat{\phi}^{(i)}(\langle s_t, a_t, s_t^{'}\rangle) = \arg\min_{\phi^{(i)}} ||\langle s_t, a_t, s_t^{'}\rangle^{(i)} - \sum_{j=1}^{d_n} \phi_j^{(i)} \mathbf{z}_j||_2^2 + \beta||\phi^{(i)}||_1$
3: return Activations Φ
```

Algorithm 4 Similarity Measure & Inter-Task mapping approximation

```
Require: Sparse Coded Basis Z, Sparse Coded Activations of the
    source task $\mathbf{C} \in \mathbb{R}^{m\times d_n}$, Projected Target Task activations
    $\mathbf{\Phi} \in \mathbb{R}^{m\times d_n}$
1: for all $\phi$ do
2:     Calculate the closest activation in C minimizing the Euclidean/similarity distance measure.
3: Create a data set $\mathcal{D}$ from minimum-distance triplets
4: Approximate the Inter-task mapping, $\chi$, from $\mathcal{D}$ with an appropriate learning algorithm
5: return The approximated Inter-task mapping, $\chi$
```

5. TRANSFER SCHEME

This section describes the novel transfer scheme.

5.1 Implementation Details

Here we introduce implementation details and background used in Section 5.2. Due to space constraints, we briefly describe Least Squares Policy Iteration (LSPI), Fitted-Q-Iteration (FQI) and Gaussian Processes (GPs).

5.1.1 Least Squares Policy Iteration

LSPI [5] is an approximate RL algorithm using the actor/critic framework. LSPI is composed of two parts: the policy is evaluated with *Least Squares Temporal Difference Q-learning* (LSTDQ) and then it is improved. LSTDQ is used to update the weights that parameterize the policy to minimize an error criterion. Once this step has finished, LSPI uses the weights to improve the policy by taking greedy actions with respect to the new Q-function.

5.1.2 Fitted-Q-Iteration

FQI [1] is another approximate RL technique that works by approximating the Q-function as a linear combination of weights and state-action basis functions. FQI operates within two spaces: 1) the parameter space and 2) the Q-function space. During each iteration of the algorithm, the Q-function is updated via the Bellman operator in its corresponding space. Then the function is projected back to the parameter space. These steps are repeated to find high quality policies in practice, although convergence to an optimal Q-function is not guaranteed.

5.1.3 Gaussian Processes

GPs are supervised learning techniques used to discover a relation between a given set of input vectors, $\mathbf{x}$, and output pairs, $\mathbf{y}$. A full mathematical treatment can be found elsewhere [11, 14]. Unlike many regression techniques, which perform inference in the weight space, GPs perform inference directly in the function space. Learning in a GP setting involves maximizing the marginal likelihood to find the hyper-parameters best describing the data. Maximizing the likelihood may be computationally complex. Therefore, we use a fast learning technique, *Sparse Pseudo-input Gaussian Processes* (SPGP) [14], to quickly model the complex inter-task mapping.

5.1.4 Sparse Pseudo-Inputs Gaussian Processes

SPGPs aim to reduce the complexity of learning and prediction in GPs by parametrizing the regression model with $M \ll N$ (N is the number of input points) pseudo-input points, while still preserving the full Bayesian framework. The covariance of the GP model is parametrized by the location of the $M \ll N$ pseudo-inputs and training aims at finding the parameters and the locations of the pseudo-points that best describe the data. Existing results [14] show a complexity reduction in the training cost (i.e., the cost of finding the parameters of the covariances) and in the prediction cost (i.e., prediction on a new set of inputs) compared to GP regression. The results further demonstrate that the SPGP framework can match normal GPs approximation power with small M (i.e., few pseudo-inputs).

5.2 Transfer Details

This section proposes two novel transfer algorithms for pairs of tasks with continuous state spaces and discrete action spaces, titled Transfer Least Squares Policy Iteration (TrLSPI) and Transfer Fitted-Q-Iteration (TrFQI), which makes use of a learned source task policy.

Transfer Least Squares Policy Iteration.

TrLSPI is described in Algorithm 5 and can be applied to any TL in RL problem having continuous states and discrete action spaces. It is also sample efficient as it preserves the advantages of the normal LSPI algorithm. TrLSPI can be split into two sections. The first (lines 1–4 of Algorithm 5) determine χ (see Section 4), using

[3]We use the s and t indices to describe the source task (typically of lower dimensions) and the target task.

Algorithm 5 TrLSPI

```
Require: Source MDP triplets $\{\langle s_s, a_s, s_s^{'}\rangle\}_{i=1}^{m}$, Target MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{j=1}^{f}$, Number for re-samples $n_s$, close to optimal policy for the source system $\pi_s^\star$, State action basis functions for the target task $\psi_1, \ldots, \psi_k$
 1: Map the Dimensions using Algorithm 1
 2: Discover High Informational Representation using Algorithm 2
 3: Sparse Project the target task triplets using Algorithm 3
 4: Use a similarity measure to attain the data set and approximate $\chi$ using Algorithm 4
 5: Randomly sample $n_s$ source task triplets $\langle s_s, a_s, s_s^{'}\rangle_{i=1}^{n_s}$ greedily in the optimal policy $\pi_s^\star$, set of state-dependent basis function $\psi_1, \ldots, \psi_k : S_t \times A_t \to \mathbb{R}$
 6: for $i = 1 \to n_s$ do
 7:     Find the corresponding target task triplets as $\langle s_t^{(i)}, a_t^{(i)}, s_t^{(i)\prime}\rangle = \chi(\langle s_s^{(i)}, a_s^{(i)}, s_s^{(i)\prime}\rangle)$
 8: Use the black box generative model of the environment to produce the rewards on the transferred triplets
 9: Use LSTDQ to evaluate transformed triplets
10: Improve policy until convergence using LSPI
11: return Learned policy $\pi_t^\star$
```

source and target task triplets.[4] The second section (lines 5–9) provides triplets (using $\pi_s^\star$) as a start for the evaluation phase of the LSPI algorithm (LSTDQ), allowing it to improve the target task policy. If the tasks are similar, and if the inter-task mapping is "good enough," then those triplets will 1) bias the target task controller towards choosing good actions and 2) restrict its area of exploration, both of which help to reduce learning times and increase performance.

Algorithm 5 leverages source task triplets to attain a good starting behavior for the target task. The performance of the policy necessarily depends on the state space region where those triplets were provided. In other words, it is not possible to achieve near-optimal performance with a small number of triplets that are in regions far from the goal state.[5] Therefore, to learn a near-optimal policy, it must collect triplets from the target with the current policy, or a large number of source task triplets should be provided. A full model of the system is not required but the algorithm does require a black box generative model for sampling.

Transfer Fitted-Q-Iteration.

The second novel algorithm, Transfer Fitted-Q Iteration (TrFQI), is also capable of transferring between MDPs with continuous state space and countable actions and preserves the advantages of the standard FQI algorithm. The key idea is to provide a good starting sample distribution on which the FQI algorithm can learn. This distribution is provided to the target task agent via χ, which in turn maps the source task triplets (sampled according to $\pi_s^\star$) into the target task. Algorithm 6 presents the pseudocode and can also be split into two parts. First, lines 1-8 use the inter-task mapping to project the source task triplets to the target task (same steps followed in the first part of TrLSPI). Second, lines 9-10 provide those triplets to the FQI Algorithm to learn a policy.[6]

[4]The source task policy may be optimal or near-optimal, depending on the RL algorithm used in the source task.

[5]This is a problem that is inherit to LSPI and not a property of the TrLSPI algorithm.

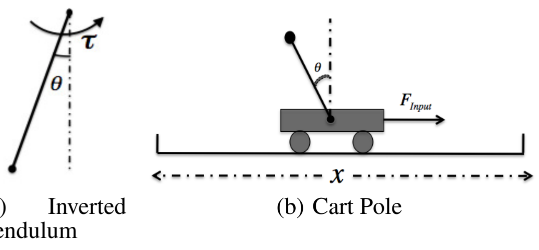
[6]It is also worth noting that the generalization of this algorithm depends on the type of function approximators used to approximate the Q-function. This is a property of FQI and not of TrFQI. Therefore, if the algorithm has to attain near-optimal behavior, either a large amount of triplets should be provided, or it must again have access to a black box generative model of the MDP for re-sampling.

Algorithm 6 TrFQI

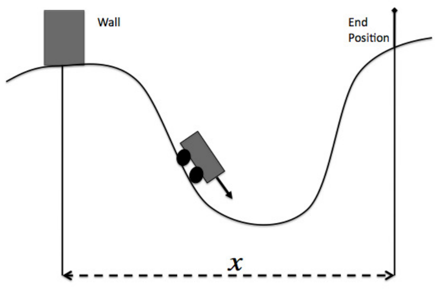
```
Require: Source MDP triplets $\{\langle s_s, a_s, s_s^{'}\rangle\}_{i=1}^{m}$, Target MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{j=1}^{f}$, Number for re-samples $n_s$, close to optimal policy for the source system $\pi_s^\star$, State action basis functions for the target task $\psi_1, \ldots, \psi_k$
 1: Map the Dimensions using Algorithm 1
 2: Discover High Informational Representation using Algorithm 2
 3: Sparse Project the target task triplets using Algorithm 3
 4: Use a similarity measure to attain the data set and approximate $\chi$ using Algorithm 4
 5: Randomly Sample $n_s$ Source task triplets $\langle s_s, a_s, s_s^{'}\rangle_{i=1}^{n_s}$ greedily in the optimal policy $\pi_s^\star$, set of state-dependent basis function $\psi_1, \ldots, \psi_k : S_t \times A_t \to \mathbb{R}$
 6: for $i = 1 \to n_s$ do
 7:     Find the corresponding target task triplets as $\langle s_t^{(i)}, a_t^{(i)}, s_t^{(i)\prime}\rangle = \chi(\langle s_s^{(i)}, a_s^{(i)}, s_s^{(i)\prime}\rangle)$
 8: Use the black box generative model of the environment to produce the rewards on the transferred triplets
 9: Apply FQI
10: return Learned policy $\pi_t^\star$
```

6. EXPERIMENTS & RESULTS

We have conducted two experiments to evaluate the framework. The first was the transfer from the Inverted Pendulum (IP), Figure 1(a), to the Cart Pole (CP), Figure 1(b). The second experiment transfers between Mountain Car (MC), Figure 1(c) and CP. This section describes the experiments conducted.



(a) Inverted Pendulum (b) Cart Pole



(c) Mountain Car

Figure 1: Experimental domains

6.1 Inverted Pendulum to Cart Pole Transfer

The source task was the inverted pendulum problem. The state variables describing the systems are the angle and angular velocity $\{\theta,\ \dot{\theta}\}$. The control objective of the IP is to balance the pendulum in an upright position with an angle, $\theta = 0$ and angular velocity $\dot{\theta} = 0$. The allowed torques are $+50$, 0 and -50 Nm. The reward function is $cos(\theta)$ which yields its maximum value of $+1$ at the up-right position.

In cart pole, the goal is to swing up the pole and keep it balanced in the upright position (i.e., $\theta = \dot{\theta} = 0$). The allowed actions are (+1) for full throttle right and (-1) for full throttle left. The reward function of the system consisted of two parts: (1) $cos(\theta)$, which yields its maximum value of $+1$ at the upright position of the pole, and (2) -1 if the cart hits the boundaries of the track. The angle and position were restricted to be within $|\theta| < \frac{\pi}{9}$ and $|x| < 3$.

In order to transfer between IP and CP, we first learn an optimal policy in the source task, $\pi^{\star}_{IP}$, with LSPI. $\pi^{\star}_{IP}$ was then used to randomly sample different numbers of initial states of task, to be used by χ. We started with 5000 and 2500 randomly sampled states (using a random policy) for the IP and the CP, respectively. These triplets were used by the algorithm described in Section 4 to learn the inter-task mapping χ[7]. After χ had been learned, different numbers of samples were collected from the source task using $\pi^{\star}_{IP}$. Specifically, we have sampled $500, 1000, \ldots 20000$ states as input to the TrLSPI and the TrFQI algorithms to measure performance and convergence times.

6.1.1 TrLSPI Results

Our results show both an increase in the performance on a fixed number of samples and a decrease in the convergence times in both a predefined number of samples and to attain an optimal policy. We measured the performance as the number of steps during an episode to control the pole in an upright position on a given fixed amount of samples. Figure 2 summarizes the results attained on a different number of transferred samples and compares them with those attained through normal LSPI learning scheme. It shows an increase in the number of control steps (i.e., steps the pole was in an up-right position) in the case of the transferred samples compared to a random sampling scheme. It can be seen that when using 2000 samples (i.e., a small number of samples) our transfer scheme was able to attain an average of 520 control steps while random initialization reached only 400. This performance increases with the number of samples to reach 1200 steps at 10000 transferred samples.

Another measure was the time required to learn a near-optimal target task policy using only a fixed number of samples. There was a decrease in the convergence times, represented by the number of iterations in LSPI, when provided a fixed amount of transferred samples. LSPI was able to converge faster once provided the transferred samples compared to a random sample data set. For example, it took LSPI 5 iteration to converge provided 5000 transferred samples but 7 iterations in the random case. Further, the algorithm converged within 12 iteration provided 20000 transferred samples while it took it about 19 for the random case. Finally, LSPI was able to converge to an acceptable policy (i.e., an 800 control steps) within 22.5 minutes after being provided a random data set, compared to 16 minutes with the transferred data set[8]. Calculating χ took an additional 3.7 minutes.

[7]We believe but have not confirmed that the samples to learn χ should be provided using random policies in both the source and the target task as we need to cover most large areas of the state-action spaces in both tasks.

[8]Our experiments were performed on a 2.8 Ghz Intel Core i7.

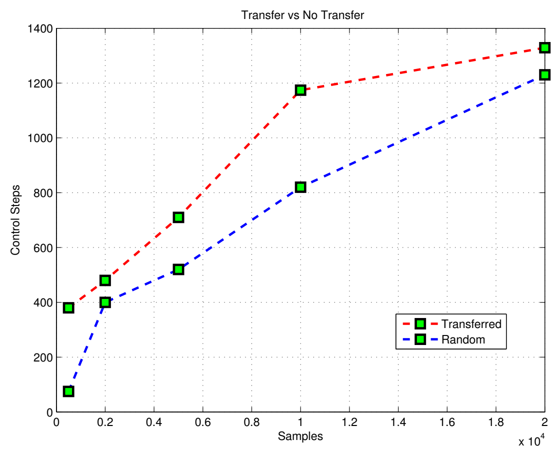


Figure 2: Cart Pole results from LSPI and TrLSPI after learning on Inverted Pendulum: the performance is measured after collecting 500 different initial states in the target tasks.

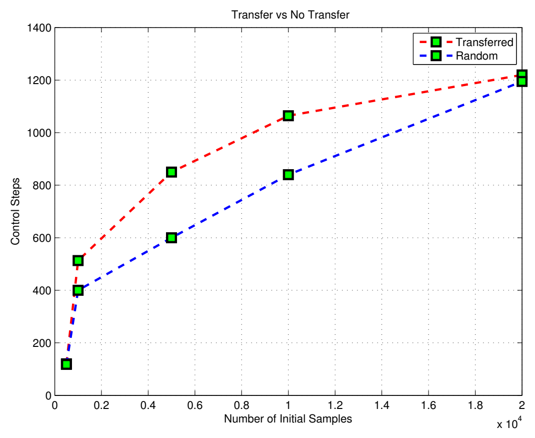


Figure 3: Cart Pole results with FQI and TrFQI after learning on Inverted Pendulum: the performance is measured after collecting 500 different initial states in the target tasks.

6.1.2 TrFQI Results

We performed similar experiments using the other proposed algorithm TrFQI. Similar results were observed as could be seen from Figure 5. The transferred samples produce more control steps on the target task compared to learning on random samples. As an illustration, the algorithm was able to achieve 800 control steps when using 5000 transferred states but it needed about 10000 random samples to attain the same performance. We also report a decrease in the number of training iterations in the TrFQI compared to FQI at a fixed number of samples and to attain an optimal policy. We have observed good performance at 50 iteration of training on transferred samples compared to 70 iterations for the random case. Moreover, TrFQI was able to reach a suboptimal acceptable policy with about 85 iteration once using transferred samples compared to a 150 iterations for the random case.

6.2 Mountain Car to Cart Pole Transfer

The source task was the MC problem, a benchmark RL task. The car has to drive up the hill (Figure 1(c)). The difficulty is that gravity is stronger than the car's motor—even at maximum throttle

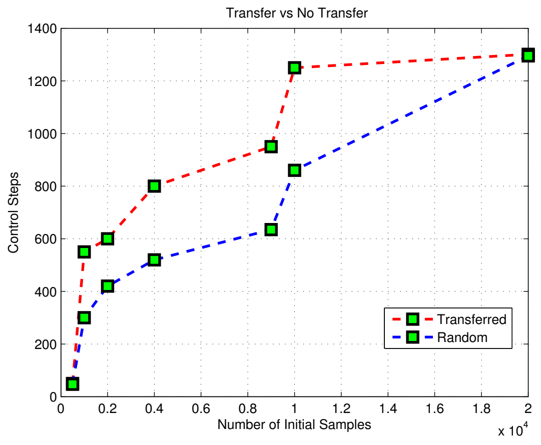


Figure 4: Cart Pole results using LSIP and TrLSPI after learning on Mountain Car: the performance is measured after collecting 500 different initial states in the target tasks.

the car can not directly reach the top of the hill. The dynamics of the car are described via two continuous state variables $(x,\ \dot{x})$ representing the position and velocity of the center of gravity of the car, respectively. The input action takes on three distinct values: maximum throttle forward (+1), zero throttle (0), and maximum throttle reverse (-1). The car is rewarded by $+1$ once it reaches the top of the hill, -1 if it hits the wall, and zero elsewhere.

The target task is the Cart Pole problem, as described in the previous experiment.

SARSA(λ) [15] is used to learn $\pi^{\star}_{MC}$ in the source task. The policy was then used to randomly sample different numbers of source task states, to be used by χ. We started with 5000 and 2500 randomly sampled states for the Mountain Car and the Cart Pole, respectively. These samples were used by the algorithm described in Section 4 to learn the inter-task mapping χ. After χ has been learned, different numbers of samples were collected from the source task using $\pi^{\star}_{MC}$. Specifically, we have sampled $500, 1000, \ldots 20000$ states as input to the TrLSPI and the TrFQI algorithms to measure performance and convergence times.

6.2.1 *TrLSPI Results*

Figure 4 clearly shows an increase in the number of control steps in the case of the transferred samples compared to a random sampling scheme. When using 2000 samples, our transfer scheme was able to attain an average of 600 control steps. Achieving a similar performance required roughly 4000 random samples. This performance increases with the number of samples to finally reach about 1300 control step on 20000 samples for both cases. We also report a decrease in the convergence times, represented by the number of iterations in LSPI, provided a fixed amount of transferred samples. LSPI was able to converge faster once provided the transferred samples compared to a random sample data set. For example, it took LSPI 7 iteration to converge provided 5000 transferred samples but 12 iterations in the random case. Further the algorithm converged within 14 iterations provided 20000 transferred samples while it took it about 19 for the random case. Finally, LSPI was able to converge to an acceptable policy within a 22.5 minutes after being provided a random data set, compared to 17 minutes with the transferred data set. Calculating χ took an addition 3.7 minutes.

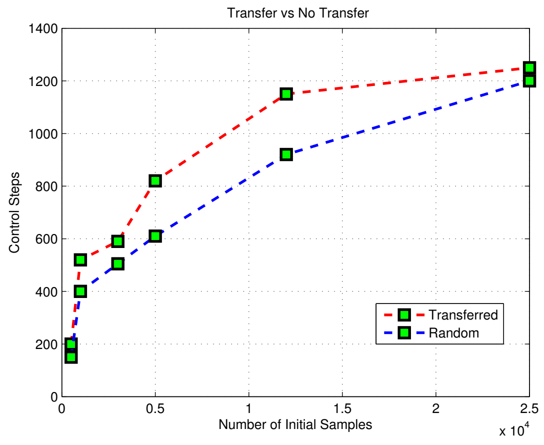


Figure 5: Transfer Results on the Cart Pole task using TrFQI after learning on Mountain Car: the performance is measured after collecting 500 different initial states in the target tasks.

6.2.2 *TrFQI Results*

The analogous experiments using TrFQI produced similar results, as shown in Figure 5. Transferred samples where able to produce a higher number of control steps in the target task when compared to learning on random samples in the target tasks. As an illustration, the algorithm was able to attain a performance of 800 control steps when using 5000 transferred states, but needed 9000 random samples to attain the same threshold. Using transferred and random samples both allow FQI to converge to roughly the same performance (1200) when provided a large number of samples. We also report a decrease in the number of training iterations at a fixed number of samples and to attain an optimal policy. We have observed good performance at 50 iteration of training on transferred samples compared to 80 iterations for random samples. Moreover, TrFQI was able to reach a suboptimal policy with about 91 iteration once using transferred samples compared to a 150 iterations for the random case.

7. ANALYSIS & DISCUSSION

It is clear from the results presented that the learner's performance increased using our proposed framework, relative to a random selection scheme. Policy performance improved, as measured by the number of control steps achieved by the agent on the target task. The number of learning iterations required also decreased, as measured by the number of iterations required by the algorithm to converge to a policy on a fixed number of transferred samples. This leads us to conclude that TrFQI and TrLSPI both:

1. provided a better distribution of samples compared to random policy in the target task,
2. required fewer iterations to converge to a fixed policy when provided a fixed number of transferred samples, and
3. reached a near-optimal performance policy faster than when using random selection scheme.

Furthermore, these results show that the proposed framework

4. successfully learned an inter-task mapping between two sets of different RL tasks.

We speculate that the framework is applicable to any model-free TL in RL problem with continuous state spaces and discrete action spaces, covering many real world RL problems. The framework has the advantage of automatically finding the inter-task functional mapping using SC and any "good" regression technique. One potential weakness is that our framework should work correctly when the two tasks at hand are *semantically* similar, as the rewards of the two systems were not taken into account in the explained scheme. For instance, consider the transfer example between the cart pole and "cart fall" tasks. The control goals of these two tasks are opposite whereby in the cart pole the pole has to be balanced in the upright position while in the cart fall the pole has to be dropped as fast as possible. In other words, the agents have the same transitions in the two tasks but have to reach two opposite goal. Our mapping scheme of Section 4, once applied, will produce a one-to-one mapping from the source to the target task relating the same transitions from both of the tasks together. Clearly the optimal policies of the two tasks are opposite. In this case the target task would be provided with a poor bias, potentially hurting the learner (i.e., producing negative transfer). We think that our approach will be able to avoid this scheme once the rewards are added to the similarity measure generating the training set to approximate the inter-task mapping χ, but such investigation is left to future work.

8. CONCLUSIONS & FUTURE WORK

This paper has presented a novel technique for transfer learning in reinforcement learning tasks. Our framework may be applied to pairs of reinforcement learning problems with continuous state spaces and discrete action spaces. The main contributions of this paper are (1) the novel method of automatically attaining the inter-task mapping, χ and (2) the new TrLSPI and TrFQI algorithms for tasks with continuous state spaces and discrete actions. We approached the problem by framing the approximation of the inter-task mapping as a supervised learning problem that was solved using sparse pseudo input Gaussian processes. Sparse coding, accompanied with a similarity measure, was used to determine the data set required by the regressor for approximating χ. Our results demonstrate successful transfer between two similar tasks, inverted pendulum to cart pole, and two very different tasks, mountain car to cart pole task. Success was measured both in an increase in learning performance as well as a reduction in convergence time. We speculate that the process usefully restricts exploration in the target task and that the transferred state quality resulting from our scheme.

There are many exciting directions for future work. First, we intend to compare different distance metrics and demonstrate their effects on the overall performance of the algorithm. Second, the distance measure will be improved by incorporating the rewards in the framework, helping to avoid the problem of negative transfer.

9. REFERENCES

[1] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[2] S. jean Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 2007, 2007.

[3] G. Konidaris and A. Barto. Autonomous shaping: knowledge transfer in reinforcement learning. In *In Proceedings of the 23rd Internation Conference on Machine Learning*, pages 489–496, 2006.

[4] G. Kuhlmann and P. Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of The Eighteenth European Conference on Machine Learning*, September 2007.

[5] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4:1107–1149, December 2003.

[6] A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 544–551, New York, NY, USA, 2008. ACM.

[7] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *In NIPS*, pages 801–808. NIPS, 2007.

[8] Y. Liu and P. Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–20, July 2006.

[9] S. J. Pan and Q. Yang. A survey on transfer learning.

[10] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.

[11] C. E. Rasmussen. In *Gaussian processes for machine learning*. MIT Press, 2006.

[12] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *IJCAI*, pages 670–672, 1985.

[13] S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. In *Machine Learning*, pages 323–339, 1992.

[14] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances In Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

[15] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 1998.

[16] E. Talvitie and S. Singh. An experts algorithm for transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[17] M. E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 283–290, May 2008.

[18] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.

[19] 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.

[20] M. E. Taylor, S. Whiteson, and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 156–163, May 2007.

[21] L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *In Proceedings of the Sixteenth European Conference on Machine Learning*, pages 412–424, 2005.