Automatically Mapped Transfer Between Reinforcement Learning Tasks via Three-Way Restricted Boltzmann Machines

Haitham Bou Ammar[1], Decebal Constantin Mocanu[1], Matthew E. Taylor[2], Kurt Driessens[1], Karl Tuyls[1], and Gerhard Weiss[1]

[1] Maastricht University, Department of Knowledge Engineering, Netherlands,
[2] School of Electrical Engineering and Computer Science, Washington State University,USA

Abstract. Reinforcement learning applications are hampered by the *tabula rasa* approach taken by existing techniques. Transfer for reinforcement learning tackles this problem by enabling the reuse of previously learned results, but requires an inter-task mapping to encode how the previously learned task and the new task are related. This paper presents an autonomous framework for learning inter-task mappings based on an adaptation of restricted Boltzmann machines. Both a full model and a computationally more efficient factored model are introduced and shown effective in multiple of transfer learning scenarios.

Keywords: Transfer Learning, Reinforcement Learning, Inter-task mapping, Boltzmann Machines, Least Squares Policy Iteration

1 Introduction

Reinforcement learning (RL) has become a popular framework for autonomous behavior generation from limited feedback [4,13], but RL methods typically learn *tabula rasa*. Transfer learning (TL) aims to significantly improve learning by providing informative knowledge from a previous (source) task or tasks to a learning agent in a novel (target) task. If the agent is to be fully autonomous, it must: (1) automatically select a source task, (2) learn how the source task and target tasks are related, and (3) effectively use transferred knowledge when in the target task. While fully autonomous transfer is not yet possible, this paper advances the state of the art by focusing on (2) above. In particular, this work proposes methods to automatically learn the relationships between pairs of tasks and then use this learned relationship to transfer effective knowledge.

In TL for RL, the source task and target task may differ in their formulations. In particular, when the source task and target task have different state and/or action spaces, an *inter-task mapping* [18] that describes the relationship between the two tasks is needed. While there have been attempts to discover this mapping automatically, finding an optimal way to construct this mapping is still an open

question. Existing technique either rely on restrictive assumptions made about the relationship between the source and target tasks, or adopt heuristics that work only in specific cases.

This paper introduces an autonomous framework for learning inter-task mappings based on *restricted Boltzmann machines* (RBMs) [1]. RBMs provide a powerful but general framework that can be used to describe an abstract common space for different tasks. This common space is then used to represent the inter-task mapping between two tasks and can successfully transfer knowledge about transition dynamics between the two tasks.

The contributions of this paper are summarized as follows. First, a novel RBM is proposed that uses a three-way weight tensor (i.e., TrRBM). Since this machine has a computational complexity of $\mathcal{O}(N^3)$, a factored version (i.e., FTrRBM) is then derived that reduces the complexity to $\mathcal{O}(N^2)$. Experiments then transfer samples between pairs of tasks, showing that the proposed method is capable of successfully learning a useful inter-task mapping. Specifically, the results demonstrate that FTrRBM is capable of:

1. Automatically learning an inter-task mapping between different MDPs.
2. Transferring informative samples that reduce the computational complexity of a sample-based RL algorithm.
3. Transferring informative instances which reduce the time needed for a sample-based RL algorithm to converge to a near-optimal behavior.

2 Preliminaries

This section provides background knowledge.

2.1 Reinforcement Learning

In a reinforcement learning (RL) problem, an agent must decide how to sequentially select actions to maximize its expected return. Such problems are typically formalized as a Markov decision process (MDP), 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 $\pi(s, a)$ represents the probability of selecting action a in state s. The goal of an RL agent is to find a policy $\pi^\star$ which maximizes the cumulative future rewards. It can be attained by taking greedy actions according to the optimal Q-function $Q^\star(s, a) = \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, a = a_0\right]$. 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 one such common technique, Least Squares Policy Iteration (LSPI) [4].

2.2 Transfer Learning for Reinforcement Learning

Transfer learning (TL) aims to improve learning times and/or behavior of an agent by re-using knowledge from a solved task. Most often, TL uses a source task, $\mathcal{T}_1$, and target task, $\mathcal{T}_2$. Each tasks is described by a (potentially very different) MDP. Specifically, task $\mathcal{T}_1$ is described by $\langle S_1, A_1, P_1, R_1, \gamma_1 \rangle$ and $\mathcal{T}_2$ by $\langle S_2, A_2, P_2, R_2, \gamma_2 \rangle$. To enable transfer between tasks with different state and/or action spaces, an inter-task mapping χ is required, so that information from a source task is applicable to the target task. Typically, χ is divided into two mappings: (1) an inter-state mapping χ_S, and (2) an inter-action mapping χ_A. The first relates states from the source task to the target task, while the second maps actions from the source task to the target task. This paper learns such mappings for samples in a pair of tasks. We define the inter-task mapping to relate source and target transitions (i.e., $\chi : S_1 \times A_1 \times S_1 \rightarrow S_2 \times A_2 \times S_2$). This allows the algorithm to discover dynamical similarities between tasks and construct the an inter-task mapping accordingly, enabling the transfer of near-optimal[3] transitions.

2.3 Restricted Boltzmann Machines

Restricted Boltzmann machines (RBMs) are energy-based models for unsupervised learning. They use a generative model of the distribution of the data for prediction. These models are stochastic with stochastic nodes and layers, making them less vulnerable to local minima [15]. Further, due to multiple layers and the neural configurations, RBMs possess excellent generalization abilities. For example, they have successfully discovered informative hidden features in unlabeled data [2]. Formally, an RBM consists of two binary layers: one visible and one hidden. The visible layer represents the data and the hidden layer increases the learning capacity by enlarging the class of distributions that can be represented to an arbitrary complexity [15]. This paper follows standard notation where i represents the indices of the visible layer, j those of the hidden layer, and $w_{i,j}$ denotes the weight connection between the ith visible and j^{th} hidden unit. We further use v_i and h_j to denote the state of the ith visible and j^{th} hidden unit, respectively. According to the above definitions, the energy function is given by:

$$E(v, h) = -\sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j \tag{1}$$

where b_i and b_j represent the biases. The first term, $\sum_{i,j} v_i h_j w_{ij}$, represents the energy between the hidden and visible units with their associated weights. The second, $\sum_i v_i b_i$, represents the energy in the visible layer, while the third term represents the energy in the hidden layers. The joint probability of a state of the hidden and visible layers is defined as:

$$P(v, h) \propto \exp\left(-E(v, h)\right)$$

[3] When using function approximation techniques, RL algorithm typically learns near optimal behaviors.

To determine the probability of a data point represented by a state v, the marginal probability is used, summing out the state of the hidden layer:

$$p(v) = \sum_h P(v,h) \propto \sum_h \left(\exp\left(-\sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j \right) \right) \quad (2)$$

The above equations can be used for any given input to calculate the probability of either the visible or the hidden configuration to be activated. The calculated probabilities can then be used to perform inference to determine the conditional probabilities in the model.

To maximize the likelihood of the model, the gradient of the log-likelihood with respect to the weights from the previous equation must be calculated. The gradient of the first term, after some algebraic manipulations, can be written as:

$$\frac{\partial \log\left(\sum_h \exp\left(-E(v,h)\right)\right)}{\partial w_{ij}} = v_i \cdot P(h_j = 1|v)$$

However, computing the gradient of the second term (i.e., $\frac{\partial \log\left(\sum_{x,y} \exp(-E(x,y))\right)}{\partial w_{ij}} = P(v_i = 1, h_j = 1)$) is intractable.

2.4 Contrastive Divergence Learning

Because of the difficulty of computing the derivative of the log-likelihood gradients, Hinton proposed an approximation method called *contrastive divergence* (CD) [6]. In maximum likelihood, the learning phase actually minimizes the Kullback-Leiber (KL) measure between the input data distribution and the approximate model. In CD, learning follows the gradient of $CD_n = D_{KL}(p_0(\mathbf{x})||p_\infty(\mathbf{x})) - D_{KL}(p_n(\mathbf{x})||p_\infty(\mathbf{x}))$ where $p_n(.)$ is the distribution of a Markov chain starting from $n = 0$ and running for a small number of n steps. To derive the update rules of w_{ij} for the RBM, the energy function is re-written in a matrix form as $E(\mathbf{v}, \mathbf{h}; \mathbf{W}) = -\mathbf{h}^T \mathbf{W} \mathbf{v}$. $\mathbf{v} = [v_1, \ldots, v_{n_v}]$, where v_i is the value of the ith visible neuron and n_v is the index of the last visible neuron. $\mathbf{h} = [h_1, \ldots, h_{n_h}]$, where h_j is the value of the jth hidden neuron and n_h is the index of the last hidden neuron. $\mathbf{W} \in \mathbb{R}^{n_h \times n_v}$ is the matrix of all weights. Since the visible units are conditionally independent given the hidden units and vice versa, learning in such an RBM is easy. One step of Gibbs sampling can be carried in two half-steps: (1) update all the hidden units, and (2) update all the visible units. Thus, in CD_n the weight updates are done as follows:

$$w_{ij}^{\tau+1} = w_{ij}^{\tau} + \alpha \left(\left\langle \langle h_j v_i \rangle_{p(\mathbf{h}|\mathbf{v};\mathbf{W})} \right\rangle_0 - \langle h_j v_i \rangle_n \right)$$

where τ is the iteration, α is the learning rate, $\left\langle \langle h_j v_i \rangle_{p(\mathbf{h}|\mathbf{v};\mathbf{W})} \right\rangle_0 = \frac{1}{N} \sum_{n=1}^{N} v_i^{(n)} P(h_i^{(n)} = 1|\mathbf{h}; \mathbf{W})$, and $\langle h_j v_i \rangle_n = \frac{1}{N} \sum_{n=1}^{N} v_i^{(n)G_l} P(h_j^{(n)G_l}|\mathbf{h}^{(n)G_l}; \mathbf{W})$ with N the total number of input instances and G_l indicating that the states are obtained after l iterations of Gibbs sampling from the Markov chain starting at $p_0(.)$. In this work, a variant of the CD algorithm is used to better learn the neural configuration of the proposed FTrRBM model and is explained in Section 3.2.

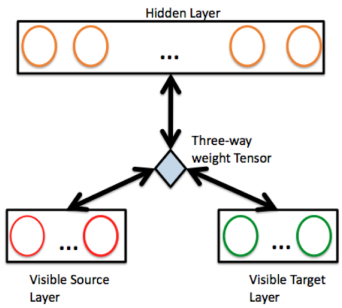


Fig. 1. The overall structure of the proposed full model. The three-way weight tensor is shown in the middle with its connections to each of the visible source layer, visible target layer, and the hidden layer shown in red, green, and orange respectively.

3 RBMs for Transfer Learning

The core hypothesis of this paper is that RBMs can automatically build an inter-task mapping using source task and target task samples, because an RBM can discover the latent similarities between the tasks, implicitly encoding an inter-task mapping. To construct an algorithm to test this hypothesis, the TrRBM framework consists of three layers as shown in Figure 1. The first is the source task visible layer (describing source samples), the second is the target task visible layer (describing target samples), and the third is the hidden layer that encodes similarities between the two tasks. This hidden layer therefore encodes a type of inter-task mapping, which will be used to transfer samples from the source to the target.

The next section presents a derivation of the full model. However, this model is computationally expensive, and a factored version of the model is developed in Section 3.2.

3.1 Transfer Restricted Boltzmann Machine

TrRBM is used to automatically learn the inter-task mapping between source and target tasks. TrRBM consists of three layers: (1) a visible source task task layer, (2) a visible target task layer, and (3) a hidden layer. The number of units in the visible layers is equal to dimensionality of each of the source and target task transitions ($i.e.$, $\langle s_t^i, a_t^i, s_t^{i\prime}\rangle$) for $i \in \{1, 2\}$, respectively. Since the inputs might be of continuous nature, the units in the visible units are set to be Gaussians with means, which are learned as described later in this section. These three layers are connected with a three-way weight tensor. Formally, the visible source layer is $\mathcal{V}_S = [v_S^{(1)}, \ldots, v_S^{(V_S)}]$, where $v_S^{(i)}$ is a Gaussian $\mathcal{N}(\mu_S^{(i)}, \sigma_S^{(i)2})$, with a mean $\mu_S^{(i)}$ and a variance $\sigma_S^{(i)2}$. The visible target layer $\mathcal{V}_T = [v_T^{(1)}, \ldots, v_T^{(V_T)}]$, where $v_T^{(k)}$ is a Gaussian $\mathcal{N}(\mu_T^{(j)}, \sigma_T^{(j)2})$, with a mean $\mu_T^{(j)}$ and a variance $\sigma_T^{(j)2}$, and the hidden layer $H = [h^{(1)}, \ldots, h^{(H)}]$ consists of sigmoidal activations.

Next, the mathematical formalizations of TrRBM are represented.[4]

Energy of the Full Model The energy function, analogous to Equation 1, that is needed to determine the update rules is as:

$$E_1(\mathcal{V}_S, \mathcal{V}_T, \mathcal{H}) = -(v_{S,\alpha} - a_\alpha)\Sigma_S^{\alpha\alpha}(v_S^\alpha - a^\alpha) - b_\beta h^\beta \\ - (v_{T,\gamma} - c_\gamma)\Sigma_T^{\gamma\gamma}(v_T^\gamma - c^\gamma) - W_{\alpha\beta\gamma}\Sigma_T^{\gamma\gamma} v_T^\gamma h^\beta \Sigma_S^{\alpha\alpha} v_S^\alpha$$

where $v_{.,\alpha}$ is the covector of v_s^α, a_α is the covector of the visible layer biases a^α, b_β is the covector of the hidden layer biases b^β, and c_γ is the covector of the visible layer biases c^γ. $\Sigma_{\cdot}^{\alpha\alpha}$ is a second order diagonal tensor representing the variances of the different layers, and $W_{\alpha\beta\gamma}$ is the third order tensor representing the weight connections.

Inference in the Full Model Because there are no connection between the units in each of the three layers, inference is conducted in parallel. Formally,

$$\mu_S^{\alpha::} = W_{\beta\gamma}^{\alpha::} v_T^\gamma h^\beta + a^{\alpha::} \\ \mu_T^{::\gamma} = W_{\alpha\beta}^{::\gamma} h^\beta v_S^\alpha + b^{::\gamma} \\ s^{:\beta:} = W_{\alpha\gamma}^{:\beta:} v_T^\alpha v_S^\alpha + c^{:\beta:}$$

where $\alpha ::$ are slices in the tensor field in the α direction, $: \beta :$ are slices in the tensor field in the β direction, and $:: \gamma$ are these in the γ direction. Therefore,

$$p(\mathcal{V}_S|\mathcal{V}_T, \mathcal{H}) = \times_\alpha \mathcal{N}(\mu_S^{\alpha::}, \Sigma^{:\alpha\alpha}) \\ p(\mathcal{V}_T|\mathcal{V}_S, \mathcal{H}) = \times_\gamma \mathcal{N}(\mu_T^{::\gamma}, \Sigma^{:\gamma\gamma}) \\ p(\mathcal{H}|\mathcal{V}_T, \mathcal{V}_S) = \times_\beta \, sigmoid(s^{:\beta:})$$

where $\times_{\cdot}$ are tensor products in the corresponding $\cdot$ fields.

Update Rules of the Full Model In this section, the update rules to learn the weights and the biases of TrRBM are described. These rules are attained by deriving the energy function of Equation 1 with respect to weight tensor yielding the following:

$$\nabla_{W_{\alpha\beta\gamma}} = \left\langle v_T^\alpha h_\beta v_{S,\alpha} \right\rangle_0 - \left\langle v_T^\alpha h_\beta v_{S,\alpha} \right\rangle_\lambda \\ \Delta a^\alpha \propto \langle v_S^\alpha \rangle_0 - \langle v_S^\alpha \rangle_\lambda \\ \Delta b^\beta \propto \langle h^\beta \rangle_0 - \langle h^\beta \rangle_\lambda \\ \Delta c^\gamma \propto \langle v_T^\gamma \rangle_0 - \langle v_T^\gamma \rangle_\lambda$$

[4] **A word on notation:** Because of space concerns, we resorted to Einstein's 1916 tensor index notation and conventions [5] for the mathematical details. As we realize this is not standard in ML literature, a more expansive derivation using a normal notation can be found in the online appendix.

where $\langle . \rangle_0$ is the expectation over the original data distribution and $\langle . \rangle_\lambda$ is the expected reconstruction determined by a Markov Chain of length λ, attained through Gibbs sampling that started at the original data distribution.

3.2 Factored Transfer Restricted Boltzmann Machine

TrRBM, as proposed in the previous section, is computationally expensive. Because TL is a speedup technique, any TL method must be efficient or it will not be useful in practice. This section presents a factored version of the algorithm, FTrRBM. In particular, the three-way weight tensor is factored into sums of products through a factoring function, thus reducing the computational complexity from $\mathcal{O}(N^3)$ for TrRBM to $\mathcal{O}(N^2)$ for FTrRBM.

Energy of the Factored Model As mentioned previously, the three-way weight tensor among the different layers is now factored. Therefore, the energy function is now defined as:

$$\begin{aligned} E(\mathcal{V}_S, \mathcal{V}_T, \mathcal{H}) = &-(v_{S,\alpha} - a_\alpha)\Sigma_S^{\alpha\alpha}(v_S^\alpha - a^\alpha) - b_\beta h^\beta \\ &- (v_{T,\gamma} - c_\gamma)\Sigma_T^{\gamma\gamma}(v_T^\gamma - c^\gamma) - w_{f\alpha}^{(V_S)}\Sigma_S^{\alpha\alpha} v_S^\alpha w_{f\beta}^{(h)} w_{f\gamma}^{(V_T)} \Sigma_T^{\gamma\gamma} v_T^\gamma \end{aligned}$$

where f is the number of factors used to decompose the weight tensor.

Inference in the Factored Model Inference in the factored version is done in a similar manner to that of the full model with different inputs for the nodes. In particular, because there are no connections between the units in the same layer, inference is done in parallel for each of the nodes. Mathematically these are derived as:

$$\begin{aligned} \mu_S^{:\alpha} &= w_f^{(V_S):\alpha} w_{f\beta}^{(h)} h^\beta w_{f\gamma}^{(V_T)} v_T^\gamma + a^{:\alpha} \\ \mu_T^{:\gamma} &= w_f^{(V_T):\gamma} w_{f\beta}^{(h)} h^\beta w_{f\alpha}^{(V_S)} v_S^\alpha + b^{:\gamma} \\ s^{:\beta} &= w_f^{(h):\beta} w_{f\alpha}^{(V_S)} v_S^\alpha w_{f\gamma}^{(V_T)} v_T^\gamma + c^{:\beta} \end{aligned}$$

Learning in the Factored Model Learning in the factored model is done using a modified version of Contrastive Divergence. The derivatives of the energy function are computed again, this time yielding:

$$\Delta w^{(V_S)}_{:f\alpha} \propto \left\langle \Sigma^{\alpha:}_S v^{\alpha:}_S w^{(h)}_{:f\beta} h^{\beta} w^{(V_T)}_{:f\gamma} \Sigma^{\gamma\gamma}_T v^{\gamma}_T \right\rangle_0 - \left\langle \Sigma^{\alpha:}_S v^{\alpha:}_S w^{(h)}_{:f\beta} h^{\beta} w^{(V_T)}_{:f\gamma} \Sigma^{\gamma\gamma}_T v^{\gamma}_T \right\rangle_\lambda$$

$$\Delta w^{(V_T)}_{:f\gamma} \propto \left\langle \Sigma^{\gamma:}_S v^{\gamma:}_T w^{(h)}_{:f\beta} h^{\beta} w^{(V_S)}_{:f\alpha} \Sigma^{\alpha\alpha}_S v^{\alpha}_S \right\rangle_0 - \left\langle \Sigma^{\gamma:}_S v^{\gamma:}_T w^{(h)}_{:f\beta} h^{\beta} w^{(V_S)}_{:f\alpha} \Sigma^{\alpha\alpha}_S v^{\alpha}_S \right\rangle_\lambda$$

$$\Delta w^{(h)}_{:f\beta} \propto \left\langle h^{:\beta} w^{(V_T)}_{:f\gamma} \Sigma^{\gamma\gamma}_T v^{\gamma}_T w^{(V_S)}_{:f\alpha} \Sigma^{\alpha\alpha}_S v^{\alpha}_S \right\rangle_0 - \left\langle h^{:\beta} w^{(V_T)}_{:f\gamma} \Sigma^{\gamma\gamma}_T v^{\gamma}_T w^{(V_S)}_{:f\alpha} \Sigma^{\alpha\alpha}_S v^{\alpha}_S \right\rangle_\lambda$$

The update rules for the biases are the same as for the full model.

with f being the index running over the number of factors, $\langle . \rangle_0$ is the expectation from the initial probability distribution (i.e., data), and $\langle . \rangle_\lambda$ is the expectation of the Markov chain, starting at the initial probability distribution, and sampled λ steps using a Gibbs sampler.

Unfortunately, learning in this model cannot be done with normal CD. The main reason is that if CD divergence was used as is, FTrRBM will learn to correlate *random* samples from the source task to *random* samples in the target. To tackle this problem, as well as ensure computational efficiency, a modified version of CD is proposed. In *Parallel Contrastive Divergence* (PCD), the data sets are first split into batches of samples. Parallel Markov chains run to a certain number of steps on each batch. At each step of the chain, the values of the derivatives are calculated and averaged to perform a learning step. This runs for a certain number of epochs. At the second iteration the same procedure is followed but with randomized samples in each of the batches. Please note that randomizing the batches is important to avoid fallacious matchings between source and target triplets.

4 Using the Inter-task Mapping

Using FTrRBMs for transfer in RL is done using two phases. First, the inter-task mapping is learned through source and target task samples. Second, samples are transferred from the source to the target, to be used as starting samples for a sample-based RL algorithm (which proceeds normally from this point onward).

Algorithm 1 Overall Transfer Framework

1: **Input:** Random source task samples $\mathcal{D}_S = \{\langle s_S^{(i)}, a_S^{(i)}, s_s'^{(i)} \rangle\}_{i=1}^m$, random target task samples $\mathcal{D}_T = \{\langle s_T^{(j)}, a_T^{(j)}, s_T'^{(j)} \rangle\}_{j=1}^n$, optimal source task policy $\pi_S^\star$
2: Use $\mathcal{D}_S$ and $\mathcal{D}_T$ to learn the intertask mapping using FTrRBM.
3: Sample source task according to $\pi_S^\star$ to attain $\mathcal{D}_S^\star$.
4: Use the learned RBM to transfer $D_S^\star$ and thus attain $\mathcal{D}_T^0$.
5: Use $\mathcal{D}_T^0$ to learn using a sample-based RL algorithm.
6: **Return:** Optimal target task policy $\pi_T^\star$.

4.1 Learning Phase

When FTrRBM learns, weights and biases are tuned to ensure a low reconstruction error between the original samples and the predicted ones from the model. The RBM is initially provided with random samples from both the source and the target tasks. Triplets from the source task (i.e., $\{\langle s_1^{(i)}, a_1^{(i)}, s_1'^{(i)} \rangle\}_{i=1}^m$) and target task (i.e., $\{\langle s_2^{(j)}, a_2^{(j)}, s_2'^{(j)} \rangle\}_{j=1}^n$) are inputs to the two visible layers of the RBM. These are then used to learn good hidden and visible layer feature representations. Note that these triplets should come from random sampling—the RBM is attempting to learn an inter-task mapping that covers large ranges in both the source and target tasks' state and actions spaces. If only "good" samples were used the mapping will be relevant in only certain narrow areas of both source and target spaces.

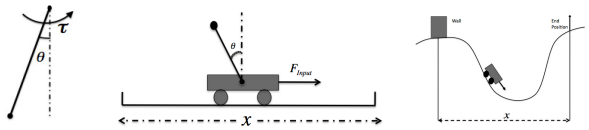
4.2 Transfer Phase

After learning, the FTrRBM encodes an inter-task mapping from the source to the target task. This encoding is then used to transfer (near-)optimal sample transitions from the source task, forming sample transitions in the target task. Given a (near-)optimal source task policy, $\pi_S^\star$, the source task is sampled greedily according to $\pi_S^\star$ to acquire optimal state transitions. The triplets are passed through the visible source layer of FTrRBM and are used to reconstruct initial target task samples at the visible target layer, effectively transferring samples from one task to another. If the source and target task are close enough[5], then the transferred transitions are expected to aid the target agent in learning an (near-)optimal behavior. They are then used in a sample based RL algorithm, such as LSPI to learn an optimal behavior in the target task (i.e., $\pi_T^\star$). The overall process of the two phases is summarized in Algorithm 1.

[5] Note that defining a similarity metric between tasks is currently an open problem and beyond the scope of this paper.

5 Experiments and Results

To asses the efficiency of the proposed framework, experiments on different RL benchmarks were performed. Two different transfer experiments were conducted using the tasks shown in Figure 2[6].



(a) Inverted Pendulum (b) Cart Pole (c) Mountain Car

Fig. 2. Experimental domains

5.1 Inverted Pendulum to Cart-Pole Transfer

To test the transfer capabilities of FTrRBMs, transfer was conducted from the Inverted Pendulum (IP) (i.e., Figure 2(a)) to the (CP) (i.e., Figure 2(b)).

Source Task The state variables of the pendulum are $\langle \theta, \dot{\theta} \rangle$. The action space is a set of two torques $\{-10, 10\}$ in Newton meters. The goal of the agent is again to balance the pole in an upright position with $\langle \theta = 0, \dot{\theta} = 0 \rangle$. A reward of $+1$ is given to the agent when the pole's angle is in $-\frac{\pi}{12} < \theta < \frac{\pi}{12}$ and -1 otherwise.

Target Task The target task was a CP with, $l = 0.5$. The target action space, transition probability and reward functions are different from the source task. The action space of the target agent was changed to $A_T = \{-10, 0, 10\}$ and the reward function was changed to $cos(\theta)$, giving the agent a maximum value of $+1$ when the pole is the upright position.

Experiment 3000 random source task samples and 2000 target task samples were used to learn the inter-task mapping. The RBM contained 80 hidden units and 25 factors. Learning was performed as before, with FTrRBM converging in about 3.5 minutes. Transfer was accomplished and tested similarly to the previous experiment. The results are reported in Figure 3. It is again clear that transfer helps the target agent in his learning task. LSPI again converged with fewer

[6] The samples required for learning the inter-task mapping were not given as extra samples for the random learner in the target. Please note, that even if these were included the results as seen from the graphs will still be in favor of the proposed methods.

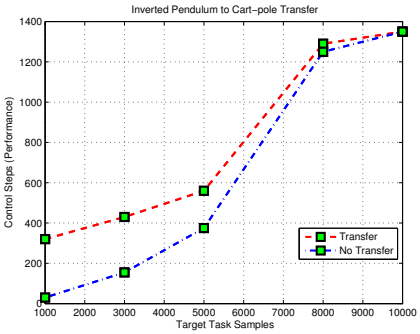


Fig. 3. Transfer versus no transfer comparison on different tasks

iterations when using transfer. LSPI convergence time also decreased on different transferred samples. For example, LSPI converged with only 9 iterations on 5000 transferred samples compared to 12 using random ones and with 17 compared to 19 on 8000 transferred and random samples, respectively. The time needed to reach near optimal behavior was reduced from 22 to 17 minutes by using a transferred policy to initialize LSPI. Therefore,
ConclusionI: FTrRBM is capable of learning a relevant inter-task mapping between a pair of dissimilar tasks.

5.2 Mountain Car to Cart-Pole Transfer

A second experiment shows the transfer performance of FTrRBMs between pairs even less similar tasks than in the previous section. The target task remained the same cart-pole as before, while the source task was chosen to be the Mountain-Car (MC) problem. Although very different, successful transfer results between these tasks had previously been shown elsewhere [3].

Source Task The system is described with two state variable $\langle x, \dot{x} \rangle$. The agent can choose between two actions $\{-1, 1\}$. The goal of the agent is to drive the car up the hill to the end position. The car's motor is not sufficient to drive the car directly to its goal state — the car has to oscillate in order to acquire enough momentum to drive to the goal state. The reward of the agent is -1 for each step the car did not reach the end position. If the car reaches the goal state, the agent receives a positive reward of $+1$ and the episode terminates. Learning in the source task was performed using SARSA. 4000 random source task samples and 2000 target task samples were used to lean the inter-task mapping as before. The RBM contained 120 hidden units and 33 factors and converged in about 3.5 minutes to the lowest reconstruction error. The results of transfer (performed as before) are reported in Figure 4. It is clear that transfer helps even when tasks are highly dissimilar. As before, LSPI converged with fewer iterations when using transfer than without using transfer. For example, LSPI converged with only 10

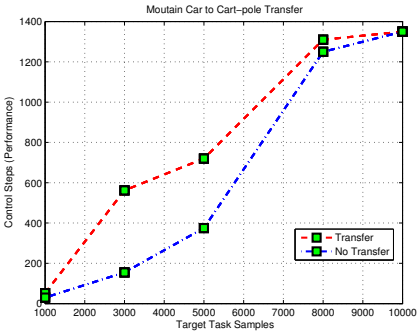


Fig. 4. Transfer versus no transfer comparison on highly dissimilar tasks.

iterations on 5000 transferred samples compared to 12 using random ones. LSPI converged to an optimal behavior in about 18 minutes compared to 22 minutes for the non-transfer case. Therefore,
ConclusionII: FTrRBM is capable of learning a relevant inter-task mapping between a pair of highly dissimilar tasks.

5.3 FTrRBM Robustness

To provide a comprehensive comparison between FTrRBM and the work in [3], two additional experiments were conducted. The source task was either the IP or the MC, while the target task was the CP system. 1000 and 500 target samples were used to learn an intertask mapping using either FTrRBM or the work of Bou Ammar et al. [3]. Having these intertask mappings, (near-) optimal source task samples[7] were then transferred to provide an initial batch for the target RL agent to learn on. Performance, measured by the number of successful control steps in the target, was then reported in Figures 5 and 6.

Figure 5 shows two comparison graphs. The left graph reports the performance when using 1000 target samples to learn the intertask mapping. These clearly demonstrate that FTrRBM performs better than sparse coded intertask mappings, where for example, FTrRBM attains about 570 control steps compared to 400 in the sparse coded case at 5000 transferred samples. As the number of control steps increases, the performance of both methods also increases, to reach around 1300 control steps for FTrRBM compared to 1080 in the sparse coded case at 10000 transferred samples. The right graph shows the results of the same experiments, however, when using only 500 target samples to learn the intertask mapping. Again these results show that apart from the first two points, FTrRBM outperforms the Sparse coded intertask mapping.

In Figure 6 the results of the same experiments on highly dissimilar tasks are shown. In the left graph, 1000 target samples were used to learn an intertask

[7] The optimal policy in the source was again attained using SARSA.

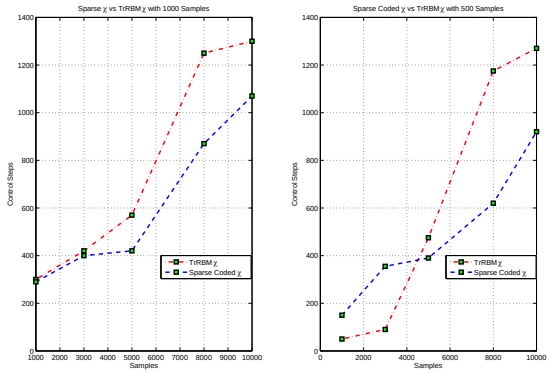


Fig. 5. Performance comparison of transferring from IP to CP using FTrRBM or Sparse Coded inter-task mappings (i.e., Bou Ammar et.al [3]). The left graph shows the results of transfer when using 1000 target samples to learn the intertask mapping, while the right presents the results when using 500 samples to learn such a mapping.

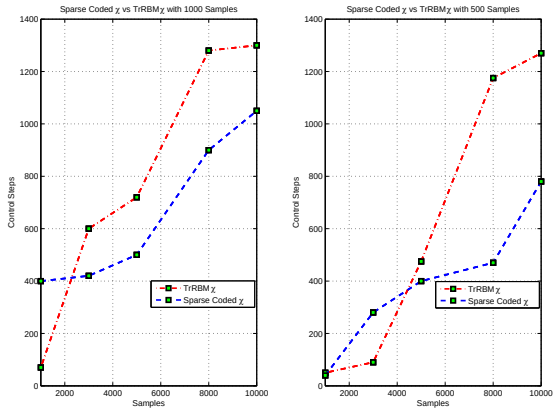


Fig. 6. Performance comparison of transferring from MC to CP using FTrRBM or Sparse Coded inter-task mappings (i.e., Bou Ammar et al. [3]). The left graph shows the results of transfer when using 1000 target samples to learn the intertask mapping, while the right presents the results when using 500 samples to learn such a mapping. It is clear that FTrRBM transfer outperforms that described in [3].

mapping using either FTrRBM or the approach of [3]. The results clearly manifest the superiority of FTrRBM compared to the sparse coded approach, where at 5000 transferred samples FTrRBM attains 600 control steps, with 410 steps for the sparse coded intertask mapping. This performance increases to reach about 1300 control steps for FTrRBM with 1050 for the sparse coded approach

on 10000 transferred samples. In the right graph the same experiments were repeated by using 500 samples to learn the intertask mapping. It is again clear that FTrRBM outperforms the approach of [3].

6 Related Work

Learning an inter-task mapping has been of major interest in the transfer learning community [17] because of its promise of a fully autonomous speed-up method for lifelong learning. 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* [12]) or have different transition functions (e.g., changing the length of a pole over time in the cart pole task [11]). 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* [7] may allow transfer between such pairs of agents, but requires the agents to share the same set of actions and an agent-centric mapping. The primary contrast between these methods and the current work is that this paper is 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 [8,10], background knowledge about the range or type of state variables can be used [14,19], or transition models for each possible mapping could be generated and tested [16].

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. The closest work to the proposed method is that of Bou Ammar *et al.* [3]. This method is based on sparse coding, sparse projection, and sparse Gaussian processes to learn an inter-task mapping between MDPs with arbitrary variations. However, the authors rely on an Euclidean distance correlation between source task and target task triplets, which may fail in highly dissimilar tasks. Our work overcomes these problems by adopting a more robust feature extraction technique and by avoiding the need for such a distance correlation as shown in the experiments of Section 5.3. Others have focused on transferring samples between tasks. For instance, Lazaric et. al [9] transfers samples in batch rein-

forcement 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 robustly learns an *inter-task mapping*, rather than a mapping based on state features.

7 Discussions and Conclusions

This paper proposes a theoretically grounded method for learning an inter-task mapping, based on RBMs. The merits of the approach were validated through experimental evidence. The proposed technique successfully learned a useful inter-task mapping between, less similar, and highly dissimilar pairs of tasks. Furthermore, a comparison between the proposed technique and that of [3] showed that FTrRBM outperforms sparse coded inter-task mappings when less samples are available.

Although successful, the approach is not guaranteed to provide useful transfer. To clarify, the reward was not included in the definition of the inter-task mapping, but when transferring near-optimal behaviors sampled according to near-optimal policies such rewards are implicitly taken into account and thus, attaining successful transfer results.

Despite that these experiments showed transfer between tasks with different reward functions, negative transfer may occur if the rewards of the source task and target tasks were highly dissimilar. Such a mismatch may lead to an incorrect mapping because the reward is not considered in the presented method. A solution to this potential problem is left for future work, but will likely require incorporating the sampled reward into the current approach.

A second potential problem may occur during the learning phase of FTrRBM, which could be traced back to quality of the random samples. If the number of provided samples is low and are very sparse[8], the learned mapping may be uninformative. The solution of this problem is also left for future work, but could possibly be tackled by using a deep belief network to increase the level of abstraction.

References

1. H. Ackley, E. Hinton, and J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, pages 147–169, 1985.
2. Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[8] Sparse in this context means that the samples arrive from very different locations in the state-space and areas of the state space are not sufficiently visited.

3. H. Bou-Ammar, M. Taylor, K. Tuyls, K. Driessens, and G. Weiss. Reinforcement learning transfer via sparse coding. In *Proceedings of the eleventh conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Valencia, Spain, 2012.
4. L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.
5. A. Einstein. The Foundation of the General Theory of Relativity. *Annalen Phys.*, 49:769–822, 1916.
6. G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, Aug. 2002.
7. 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.
8. 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.
9. 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.
10. 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.
11. O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *IJCAI*, pages 670–672, 1985.
12. S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. In *Machine Learning*, pages 323–339, 1992.
13. R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 1998.
14. E. Talvitie and S. Singh. An experts algorithm for transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.
15. G. W. Taylor, G. E. Hinton, and S. T. Roweis. Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research*, 12:1025–1068, 2011.
16. 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.
17. M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.
18. 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.
19. 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.