Symmetry detection and exploitation for function approximation in Deep RL

Anuj Mahajan ; Theja Tulabandhula
Xerox Research Centre
Bangalore-560103
India
{anujmahajan.iitd ; theja2t}@gmail.com

ABSTRACT

Symmetry based model reduction is a well studied topic in the context of Markov Decision Processes (MDPs). However, very little has been done to explore and exploit the symmetries in the environment for Deep Reinforcement Learning (DRL). Furthermore, with recent advances in the use of deep networks as function approximators for complex reinforcement learning (RL) tasks which require large amount of training data in the form of agent's experience, ensuring sample efficiency is an important problem. Thus, it is imperative to find methods which discover the underlying symmetries in the environment and leverage them to learn the optimal behavior more efficiently.

Our main contributions in this paper are as follows: we introduce a novel method to detect environment symmetries using reward trails observed during episodic experience and prove its completeness. Our second contribution is developing a framework to incorporate the discovered symmetries for functional approximation so that sample efficiency can be improved. Finally, our third contribution is showing that the use of potential based reward shaping is especially effective for our symmetry exploitation mechanism. We present results for our approach on various classical control problems (discrete and continuous) and show that the proposed method outperforms agents which do not use symmetry information for learning.

CCS Concepts

•**Computing methodologies** → **Sequential decision making;**

Keywords

Deep Reinforcement Learning, Functional Approximation, Symmetry, Representation Learning

1. INTRODUCTION

Reinforcement Learning (RL) is the task of training an agent to perform optimally in an environment using the reward and observation signals perceived upon taking actions which change the environment dynamics. Learning optimal behavior is inherently difficult because of challenges like credit assignment and exploration-exploitation trade offs that need to be made while converging to a solution. In many scenarios, like training a rover to move on martian surface, the cost of obtaining samples for learning can be high (in terms of robot's energy expenditure etc.), and so sample efficiency is an important subproblem which deserves special attention. Very often it is the case that the environment has intrinsic symmetries which can be leveraged by the agent to improve performance and learn more efficiently. For example, in the Cart-Pole domain [2, 27] the state action space is symmetric with respect to reflection about the plane perpendicular to the direction of motion of the cart (Figure 1). Thus if the state of the Cart-Pole system is given by (θ, x, ω, v) denoting the angular position of pole, position of cart and their time derivatives. Then, the state action pairs $\langle(\theta, x, \omega, v), Left\rangle and \langle(-\theta, -x, -\omega, -v), Right\rangle$ are symmetric for the purposes of learning the optimal policy (or Q function). Such is the case with many of the Atari 2600 environments like Breakout, Space Invaders as well. In fact, in many environments, the number of symmetry relations tend to increase with the dimensionality of the state space. For instance, for the simple case of grid world of dimension d (Figure 2) there exist $O(d!2^d)$ fold symmetries. This can provide substantial gains in sample efficiency while learning as we would ideally need to consider only the equivalence classes formed under the induced symmetry relations. However, discovering these symmetries can be a challenging problem owing to noise in observations and complicated dynamics of the environment. Girgin et al. [9] have given an approach to detect state similarities based on action reward sequences, but their method cannot be used even in relatively simple environments like Cart-Pole where the action mappings under the symmetry transformation are not in-

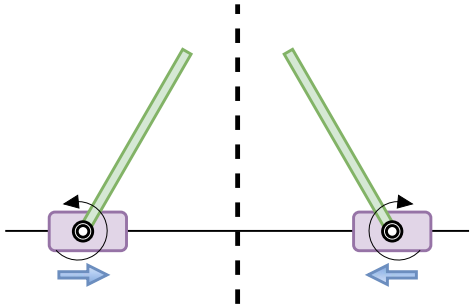

Figure 1: Symmetry in Cart-Pole environment

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

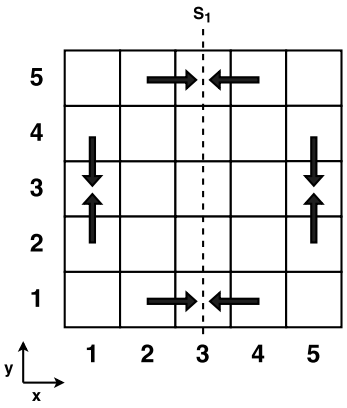

Figure 2: State action symmetries in a grid world

variant. Further the authors give no generalization for continuous state spaces and their method is not sample efficient in general.

With recent advances in deep reinforcement learning [17, 20, 14, 13], it has been demonstrated that a lot of seemingly complex tasks like game play for Atari, control in simulated physics environments etc., which pose challenges in the form of large (possibly continuous) state action spaces and the difficulty of learning good representations [3, 32]; can be handled very well with the use of deep networks as functional approximators. Training these networks, however requires large amounts of data and learning their parameters necessitates coming up with careful update procedures and defining the right objectives(see Glorot et al [11]) and is a topic of study in its own right. This points us to the fact that established methods for MDP abstraction and minimization can't be practically extended for use in function approximation for RL using deep networks which builds up the premise for this work.

To the best of our knowledge, we are the first to motivate the use of symmetry in the context of deep reinforcement learning. In this paper we investigate methods for discovering state space symmetries and their inclusion as prior information via a suitable cost objective. We also show that our method dovetails seamlessly with the framework of using potential based reward shaping [19] which can help reduce the size of the data structures required for symmetry detection and provides additional information for establishing robust similarity estimates. Experiments on various classical control settings validate our approach with agents showing significant gains in sample efficiency and performance when using symmetry information.

The rest of the paper is organized as follows: In Section 2 we discuss the related work on MDP abstraction and point out some of its shortcomings, In Section 3 we briefly discuss the ideas on which this work builds, In Section 4 we discuss our proposed method and give relevant analyses, Section 5 elaborates the experimental validation of our method for different setting and comparison with earlier approaches along with insights on the results, Section 6 discusses the implications of our work and hints future work, Finally the Appendix contains proofs, definitions and network architecture used for further reference.

2. RELATED WORK : MDP ABSTRACTION

In many realistic scenarios the tasks needed to be learned by the RL agent is composed of several sub tasks which form a hierarchy across the state space. Many RL approaches have been proposed to use this intrinsic structure for efficient learning, These methods broadly fall in the framework of temporally abstract actions (TAA) or *options* [15, 7, 1, 28] in which generalizations to primitive actions called macro actions are generated which last for more that one time step. A major caveat for such methods is that they require extensive domain knowledge specification by the agent's designer, methods in [26, 16, 8] try to automate this process by learning from agent observations but are sub optimal as they fail to discover all the abstractions [9]. Recent approaches for incorporating TAA approach using linear function approximation [29, 25] have been shown to be effective. Value function generalization for sub goal setting by Schaul et al [23] also gives better generalization over unseen sub goals in the function approximation setting. Kulkarni et al [12] have given a hierarchical DQN setting which uses abstraction and intrinsic motivation for better performance in environments giving sparse feedbacks. As we shall see our method has the potential to be augmented with these works for finding symmetries in sub-goal spaces.

Model minimization [33, 6] is also a closely related field in which symmetries are defined using equivalence relations on state and state action spaces, but a more generalized notion of symmetries is presented in [22] which allows for greater state space reduction. Methods like [10, 9] use conditionally terminating sequences formed from observed reward action sequences and directly try to estimate state equivalence and are most closely related to our methods for symmetry detection. Most these methods have prohibitively high overheads don't scale well with large or continuous state action spaces, moreover none of them address aforementioned issues in the context of using them for function approximation in deep RL.

3. PRELIMINARIES

In this section we give preliminary definitions and provide an overview of the topics we will build upon.

3.1 Reinforcement Learning

Reinforcement Learning can be modeled as an MDP with unknown underlying components, We define an MDP as a tuple $M := \langle S, A, \Psi, T, R \rangle$ where S is the set of states an agent observes, A is the set of actions, $\Psi \subset S \times A$ is the set of admissible state-action pairs which dictate the actions an agent can take in any state s, $T : \Psi \times S \to [0, 1]$ is a state transition function such that $\forall (s, a) \in \Psi$, $s' \in S$, $\sum_{s'} T(s, a, s') = 1$, and $R : \Psi \times S \to \mathbb{R}$ is the reward function. $R(s, a, s')$ is the immediate expected reward received when action a is executed in state s resulting in a transition to state s'. A policy, $\pi : \Psi \to [0, 1]$, is a mapping that defines the probability of selecting an action in a particular state. The value of a state s under policy π, $V_\pi(s)$, is the expected infinite discounted sum of rewards that the agent will gain if it starts in state s and follows π. Similarly the value of the state-action pair (s, a) is the expected infinite discounted sum of rewards that the agent will gain if it starts in state s takes action a and follows π and is denoted by $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$, where γ is the discount factor and r_t is the reward gained at time step t. The objective of

an agent is to find an optimal policy, π^*, which maximizes the state value function for all states. This is equivalent to satisfying the *Bellman optimality equation*:

$$Q_{\pi^*}(s,a) = \mathbb{E}_{s' \sim T(s,a,.)}[R(s,a,s') + \gamma \max_{a' \in \Psi(s',.)} Q_{\pi^*}(s',a')] \quad (1)$$

3.2 Symmetries in MDP

The notion of symmetries in MDP can be rigorously treated using the concept of MDP *homomorphisms*. Ravindran et. al [22](see Appendix B for some definitions) define an MDP homomorphism h from $M = \langle S, A, \Psi, T, R\rangle$ to $M' = \langle S', A', \Psi', T', R'\rangle$ as a surjection $h : \Psi \rightarrow \Psi'$, which is itself defined by a tuple of surjections $\langle f, \{g_s, s \in S\}\rangle$. In particular, $h((s,a)) := (f(s), g_s(a))$, with $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$, which satisfies two requirements: Firstly it preserves the reward function (i.e., $R'(f(s), g_s(a), f(s')) = R(s,a,s')$) and secondly it commutes with transition dynamics of M (i.e., $T'(f(s), g_s(a), f(s')) = T(s,a,[s']_{B_{h|S}})$). Here we use the notation $[\cdot]_{B_{h|S}}$ to denote the *projection* of equivalence classes B that partition Ψ under the relation $h((s,a)) = (s',a')$ on to S.

Symmetries $\chi : \Psi \rightarrow \Psi$ can then be formally defined as *automorphisms* on M that completely preserve the system dynamics with the underlying functions f, g_s being bijective. The homomorphism requirements for a symmetry reduce to:

$$T(f(s), g_s(a), f(s')) = T(s,a,s') \quad (2)$$

$$R(f(s), g_s(a), f(s')) = R(s,a,s') \quad (3)$$

The set of *equivalence classes* $C[(s,a)]$ of state action pairs formed under the relation $\chi((s,a)) = (s',a')$ (or more generally under any homomorphism h) partition Ψ and can thus be used to form a quotient MDP, represented as $M_Q = M/C$, which is smaller and can be efficiently solved.

As an example let us consider the 2D grid world in Figure 2 with states represented as ordered pair (x,y) and actions are the set $\{N, E, W, S\}$ corresponding to the directions of possible motion. Here for the symmetry with respect to the vertical axis χ_{S1}, we have $f((x,y)) = (6-x, y)$ and $g_s(N) = N, g_s(S) = S, g_s(E) = W, g_s(W) = E \ \forall s$. The equivalence class of state action pair $[((2,1),E)]_{\chi_{S1}} = \{((2,1),E), ((4,1),W)\}$

However in the RL setting, we do not know the underlying system dynamics and consequently, we do do not know C in advance, thus we cannot perform a model reduction. A workaround for this would be to estimate $C[(s,a)]$ on the go using the agent's experience and use the estimated set of equivalent classes to drive identical updates for equivalent state-action pairs during the process of learning the optimal policy.

3.3 Function Approximation

Tabular RL methods that use Bellman's expectation equation for iterative updates don't scale well with the number of states and actions. For many environments having continuous state and/or action spaces, these methods are prohibitively slow. Thus in practice it is common to use function approximators for estimating the optimal action value function ($Q(s,a;\theta) \approx Q_{\pi^*}(s,a)$). These have an additional advantage of generalizing to unseen configurations.Many successful RL implementations have been shown to work with linear functional approximators using hand crafted features [27], but their performance is highly sensitive depends on the choice of the features. Mnih et al. [17] developed a Deep Q Network (DQN) agent that successfully used deep neural networks for learning control policies for complex domains. Their success can be attributed to the good representations learned by the underlying networks and their generalization power. Many other variants of DQN and other deep network based agents have followed since [31, 30]. As we show, our work can be applied to any form of functional approximation. We focus on value function approximation using deep feed forward networks in our study to illustrate the merits of our solutions.

3.4 Reward Shaping

Reward shaping is a technique that augments the reward function R of a MDP $M = \langle S, A, \Psi, T, R\rangle$ with a shaping function $F : \Psi \times S \rightarrow \mathbb{R}$, with the goal of providing the agent with additional useful information about the environment in the form of rewards. This is done so that it can learn an optimal policy quickly. Thus, the agent now sees a modified reward $R'(s,a,s') = R(s,a,s') + F(s,a,s')$ when it takes actions. The notion of reward shaping in RL has its roots in behavioral psychology [24]. It has been shown that the shaping rewards is helpful in learning optimal policies if the shaping rewards are carefully designed to ensure that the policy invariance property holds [21]. Ng et al. [19] have shown that if the reward shaping function is potential based i.e., is of the form: $F(s,a,s') = \gamma\Theta(s') - \Theta(s) \quad \forall s,a,s'$ for some $\Theta : S \rightarrow \mathbb{R}$, then the policy invariance property is guaranteed. The advantage of the reward shaping technique in our context is two fold. Firstly, it helps in faster convergence to the optimal policy due to additional information. Secondly, since our method is dependent on estimating similarities across state action pairs based on observed reward trails, it helps distinguish these pairs by making the rewards sufficiently distinct and consequently preventing spurious similarity estimates.

4. METHODOLOGY

In this section we present our approach for simultaneously discovering and exploiting symmetries in the RL framework. Our goal is to incorporate the state action space symmetry information early on during the learning so that *regret* can be minimized and the optimal policy can be learned in a sample efficient manner. The main components involved in our method are: (a) symmetry detection, and (b) learning with symmmetry based priors. We elaborate each of these components next.

4.1 Symmetry Detection

Given an MDP $M = \langle S, A, \Psi, T, R\rangle$, we define set $\Pi_{sa,j} = \{(\sigma, N_\sigma)\}$ where σ is a sequences of rewards of length j and N_σ is the number of times it is seen starting with state s taking action a during the execution of policy π. We use the notation $|\Pi_{sa,j}| = \sum_{|\sigma|=j} N_\sigma$ and $\Pi_{sa,j} \cap \Pi_{s'a',j} = \{(\sigma, \min(N_\sigma, N'_\sigma))\}$(' corresponds to s', a'). We define the notion of similarity between two state action pairs $\langle s,a\rangle$ and $\langle s'a'\rangle$ as follows:

$$\chi_{i,l_0}(\langle s,a\rangle, \langle s'a'\rangle) = \frac{\sum_{j=l_0}^{i} |\Pi_{sa,j} \cap \Pi_{s'a',j}|}{(\sum_{j=l_0}^{i} |\Pi_{sa,j}| * \sum_{j=l_0}^{i} |\Pi_{s'a',j}|)^{1/2}} \quad (4)$$

To efficiently compute the similarities between all the state action pairs, we use an auxiliary structure called the reward

history tree P (similar to [9]), which stores the prefixes of reward sequences of length upto i for the state action pairs observed during policy execution. A reward history tree $P(N, E)$ is a labeled rooted tree (root being a null node). Here, N is the set of nodes with each node labeled with a reward observation. And E is the set of directed edges defined in the following way: Let sequence of reward labels obtained while traversing to n starting from the root be denoted by σ_n, then directed edge $(n, n') \in E$ iff σ_n appended with the reward label of n' form a prefix for some observed reward sequence. Additionally each node n maintains a list of state, action, occurrence frequency tuples $[\langle s, a, o\rangle]$, Thus node n, will store a tuple $\langle \hat{s}, \hat{a}, \hat{o}\rangle$ if the reward sequence σ_n was observed starting from $\hat{s}$ taking action $\hat{a}$ for $\hat{o}$ times during policy execution. The similarities can then be computed by doing a breadth first traversal on P and maintaining two arrays $A_u(\langle s, a\rangle)$ and $A_p(\langle s, a\rangle, \langle s'a'\rangle)$ that store the occurrences and co-occurrences of the observed reward sequences. Estimate can be computed by: $\chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) = \frac{A_p(\langle s,a\rangle, \langle s'a'\rangle)}{(A_u(\langle s,a\rangle) \cdot A_u(\langle s',a'\rangle))^{1/2}}$.

We consider state pairs

$$\chi_{sym} := \{\langle s, a\rangle, \langle s'a'\rangle | \chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) \geq \Delta\}$$

as similar for the given length (i and l_0) and threshold parameters (Δ). Our objective of using symmetry for faster learning thus involves balancing two opposing phenomena. If we are too early in trusting our similarity estimates, we might be hurting the learning process by updating for falsely estimated (s, a) similar pairs. On the other hand, if we wait too long to collect samples for getting good estimates we might be losing out on the opportunity of using symmetry early on and minimizing regret. Thus the parameters Δ, i and l_0 need to be judiciously chosen for getting good estimates early enough so as to exploit them.

Note that the definition of χ_{i,l_0} enables finding state action symmetries even when the actions are not invariant under the symmetry transform ie. $g_s(a) \neq a \forall s, a$ (indeed, this is the case with the Cart-Pole problem where $\forall s \in S$ $g_s(Left) = Right, g_s(Right) = Left$). Previous work in [9] is unable to do this. Finally we present the theorem which establishes the completeness of the similarity measure $\chi_{l_0,i}$

THEOREM 1. *Let $(s, a), (s', a')$ be equivalent pairs under symmetry $\chi\langle f, g_s\rangle$ which induces the coarsest partition on Ψ. Assuming uniform distribution over starting states for each episode run, we have:*

$$\lim_{|episodes| \to \infty} \chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) = 1 \tag{5}$$

for all symmetric pairs, $\forall l_0, i \leq |S|$.

Informally completeness asserts that any state action pair which is equivalent under the given symmetry should be identifiable using the similarity measure 4. The proof of this theorem is given in Appendix A.1. We are interested in the χ that induces the coarsest partition because it leads to the highest sample efficiency as ideally we wish to update all the pairs in a partition in parallel for a given observation. Finally, note that our method and theorem 1 do not strictly require symmetries at the MDP level. The method can find more general homomorphic reductions and the associated equivalence classes (we omit this generalization for brevity and conciseness).

4.2 Symmetry Inclusion Priors

Let $Q(s, a; \theta)$ be the function approximator being used (in our experiments, we use a deep network similar to that of DQN). Having found some symmetric state action pairs χ_{sym}, Our next task would be to use this information while training the function approximator network. Specifically we want the network to have identical outputs for symmetric state-action pairs. This can be achieved by constraining the network to learn identical representations of the symmetric pairs in its top layer. Such a restriction may also help generalize for other unseen symmetric pairs (which can be numerous). An intuitive way of moving towards this goal would be to directly use χ_{sym} for inducing hard constraints while minimizing an appropriate loss based on one-step TD (temporal difference) targets:

$$L_{i,TD}(\theta_i) = \mathbb{E}_{\mathbb{B}}[((r + \gamma \max_{a'} Q(s', a'; \theta_i)) - Q(s, a; \theta_i))^2] \tag{6}$$

Where $\mathbb{B}$ is the set of observed (s, a, r, s') tuples following a ϵ-greedy behavioral policy which ensures sufficient exploration. Thus the minimization problem becomes:

$$\min_{\theta_i} \; L_{i,TD}(\theta_i) \;\text{ s.t.}$$
$$Q(s, a; \theta_i) = Q(s', a'; \theta_i) \; \forall (\langle s, a\rangle, \langle s'a'\rangle) \in \chi_{sym}. \tag{7}$$

However it becomes difficult to optimize the above problem if there are too many constraints and methods like [5] might be required. Moreover since the estimated similarity pairs are not guaranteed to be true, it may be better to solve a *softer* version of the problem by introducing the symmetry constraints as an additional loss term:

$$L_{i,Sym}(\theta_i) = \mathbb{E}_{\chi_{sym}}[(Q(s', a'; \theta_i) - Q(s, a; \theta_i)^2] \tag{8}$$

The overall loss thus becomes:

$$L_{i,total}(\theta_i) = L_{i,TD}(\theta_i) + \lambda L_{i,Sym}(\theta_i), \tag{9}$$

where λ is a weighing parameter for the symmetric loss.

4.3 Learning with Symmetric Priors

Differentiating the total loss (Eq. 9) with respect to the weights, we arrive at a combination of gradients coming from the two loss objectives:

$$\nabla_{\theta_i} L_{i,TD}(\theta_i) = \mathbb{E}_{\pi,s}[(r + \gamma \max_{a'} Q(s', a'; \theta_i) - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i)] \tag{10}$$

$$\nabla_{\theta_i} L_{i,Sym}(\theta_i) = \mathbb{E}_{\pi,\chi_{sym}}[(Q(s', a'; \theta_i) - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s', a'; \theta_i)] \tag{11}$$

In practice we use stochastic gradient descent for loss minimization using mini batches $\mathbb{B}$. Eq. 10 represents the familiar Q-learning gradient for function approximation. Also Eq. 11 is defined so because we perform our weight updates in two steps. The first step involves using Eq. 10 on a randomly chosen mini-batch $\mathbb{B}$; this updates the network with fresh experience for observed state action pair (s, a) and its associated reward. In the second step, we propagate this experience to the symmetric states (s', a') by moving their state-action value estimates towards $Q(s, a; \theta_i)$ (the targets) but while doing so we wish to keep the targets fixed(approximately). Otherwise, the network may move the

output of both (s, a) and (s', a') in order to minimize symmetric loss, which is clearly undesirable (we experimentally confirmed that this did adversely impact learning rates). Below we present the proposed symmetric version of the DQN algorithm 1.

Algorithm 1 Sym DQN

```
Initialize: Memory 𝔻 ← {}, P ← {{root}, {}}
Initialize action-value function Q with random weights(θ)
for episode ≤ M do
    Initialize start state
    for t = 1 to T do
        With probability ϵ select action a_t
        Otherwise select a_t = argmax_a Q(s_t, a, θ)
        Execute action a_t and observe reward r_t state s_{t+1}
        Store transition (s_t; s_t; r_t; s_{t+1}) in 𝔻
        Sample random minibatch 𝔹 from 𝔻
        Set targets y_j ← r_j + max_a Q(s_{j+1}, a, θ)
        Perform gradient descent step 10 using 𝔹
        Find 𝔹_s the batch of symmetric pairs of 𝔹 from P
        Perform gradient descentstep 11 using 𝔹_s
    Update P with the episode
```

SymDQN 1 essentially matches DQN except for steps $13 - 15$ which describe the symmetry detection and exploitation steps, The modularity of our algorithm enables easy modification of existing RL algorithms for symmetry incorporation, For instance GORILA framework proposed by Nair et.al [18] massively parallelized the DQN algorithm to reduce learning time on Atari environment can be easily modified to accommodate symmetry learning procedure and would be expected to give even better speed ups due to symmetry sharing.

5. EXPERIMENTS

In order to validate our approach we compare the performance of different agents using deep feed forward neural networks for state-action value function approximation(see Appendix C for architectures used). In particular, we use two test environments. The first environment is a goal state grid world traversal task. The second environment is the well known Cart-Pole balancing task. These tasks are representative of many different environments in RL. For example, the Taxi drop problem and other grid based environments are similar to the first task and many physics based control tasks such as acrobot, pendulum etc. are similar to the Cartpole task.

5.1 Grid World

In the grid world domain we consider the problem of finding the shortest path to a goal in a nXn square grid world. The state space is described by the coordinates $(x, y), 1 \leq x, y \leq n$. The action set is $\{N, E, W, S\}$ corresponding to the directions in which the agent can move. Upon taking a desired step, we have a 90% chance of landing on the expected state and a 10% chance of landing on a randomly chosen adjacent state.On hitting the boundaries the agent ends up in the same state. The episode starts at a randomly chosen state and the agent is allowed to take a predefined maximum number of moves to reach the goal. The discount factor is set to be $\gamma = 0.9$, exploration $\epsilon = 0.1$ and the weigh parameter is set to $\lambda = 1$. We use fully connected neural nets as function approximators with two hidden layers. ReLU non-linearity is used for the non linear activation function at each hidden node. We use TD(0) Learning as the control algorithm to compare the effect of adding symmetries 9. The results are presented for various grid world sizes with results averaged over 50 iterations. The goal state(x_G, y_G) is chosen randomly at the start of each iteration. For the 2D grid world we test two kinds of reward shaping settings as discussed below.

In the first informative reward setting, we introduce a reward shaping function based on the potentials: $\Theta(x, y) = (|x - x_G| + |y - y_G|)$ (referred to as $Pot1$). The parameters are set to be $\Delta = 0.8, l_0 = 1, i = 7, \lambda = 0.4$. In the second informative reward setting the potential used is: $\Theta(x, y) = (|x - x_G| + |y - y_G|)\gamma^{|x-x_G|+|y-y_G|}$ (referred to as $Pot2$). The parameters are set to be $\Delta = 0.8, l_0 = 1, i = 5, \lambda = 0.4$. The maximum episode length for grid sizes $9, 13$ is set to $480, 800$ respectively. The average reward per time step as the number of episodes are varied for both the potential settings are plotted in Fig. 3(a,b,c,d). A comparison with the previous work by Girgin [9] and a naive agent which uses no symmetry information is also plotted.

From the plot, we can see that the all the agents converge to the optimal policy. Furthermore, the agent running our algorithm (labeled Sym) learns the optimal policy much faster than baseline (labeled $Naive$) and previous work. The Sym agent converges to the optimal policy within 50 episodes whereas the other agents require nearly 120 episodes to learn the same. We can also see the effects of the amount of information offered by the environment on the process of learning(a,b vs c,d). The reward shaping potentials $Pot2$ are more informative than $Pot1$ as they also convey nearness to the goal state. The effect of this disparity can be seen in the plots as the agents learning under $Pot2$ tend to learn faster than those which are given rewards augmented with $Pot1$. Finally, it is also evident that the effects of adding symmetry priors for learning become more significant as the size of the grid increases(a,c vs b,d).

Figure 4 shows the average number of similar pairs found per observed pair as the number of episodes increase for different threshold values Δ. We can observe that the number of similar pairs found per observed pair increases quickly until the optimal policy is found and thereafter increases only slightly, seemingly reaching an upper limit asymptotically. Further it is interesting to see that the number of such similarities is greater than eight which is the number of symmetries of the 2D grid world. This is because for the task of finding the shortest path all the states which are equidistant(in L_1 norm) from the goal state are approximately equivalent under the assumed shaping potential. Thus a more efficient reduction in the state action space is possible. Moreover the similarity measure $\chi_{l_0,i}$ successfully finds these similarities since only the rewards are being considered for the measurement.

In the next part of experimenting with Grid-World domain we wish to analyze how our method performs when the dimensionality of the grid is increased, to this end we set up a $3D$ grid world in which the states are given by (x, y, z) $1 \leq x, y, z \leq n$, The action set was extended to contain two more motions up and down: $\{N, E, W, S, U, D\}$ The transition once again have a 90% chance of landing on the expected state and 10% chance of selecting a random adjacent state. Notice that this domain has $3!2^3 = 48$ fold symmetry. $Pot2$ type potentials are used : $\Theta(x, y, z) = (|x - x_G| + |y - y_G| + |z - z_G|)\gamma^{|x-x_G|+|y-y_G|+|z-z_G|}$. The param-

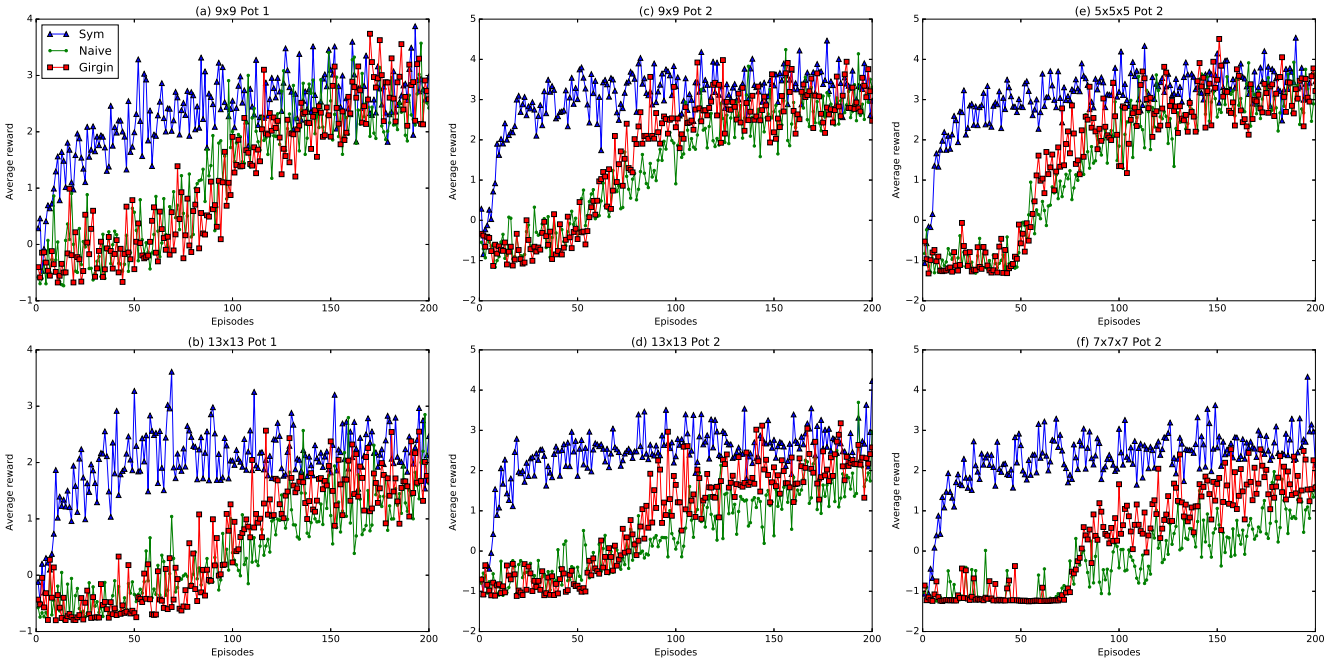

Figure 3: Gridworld plots:(a)9x9 Pot 1,(b)13x13 Pot 1,(c)9x9 Pot 2,(d)13x13 Pot 2,(e)5x5x5 Pot 2,(f)7x7x7 Pot 2. (a) to (d) are averaged over 50 iterations, (e),(f) are averaged over 25 iterations.

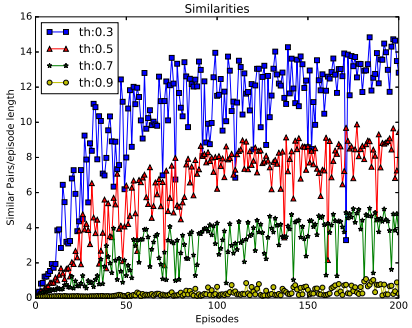

Figure 4: Variation of similar pairs found for different threshold values.

eters are set to be $\Delta = 0.8, l_0 = 1, i = 5, \lambda = 0.4$.The maximum episode length for grid sizes $5, 7$ is set to $1000, 1500$ respectively. The average reward per time step as the number of episodes are varied are plotted in Fig. 3(e,f). From the plots we can observe that the performance of the agent using our method(Sym) seems to be unaffected by the increase in dimensionality and it is still able to learn the optimal policy within 50 episodes, whereas other agents need increased number of episodes to converge with them failing to reach the optimal even in 200 episodes for grid of size 7, This makes intuitional sense as for a grid of size n in d dimensions, we have a total of $O(2dn^d)$ Q values to learn, whereas if we were to use symmetries for model reduction, we would be only be required to learn $O(\frac{2dn^d}{d!2^d})$ values.

5.2 Cart-Pole

The Cart-Pole domain first introduced by Barto et al. [2] is a classic control problem. The goal of the agent in this task is to balance a pole hinged on a cart by just moving the cart. The state (θ, x, ω, v) denoting the angular position of pole, position of cart and their time derivatives is continuous and is typically bound by a box in 4 dimensions. The action space is the move set $\{Left, Right\}$. The agent gets a reward of $+1$ after every time step it manages to keep the pole balanced within the box bounds.

As discussed before the state action space exhibits symmetry about the plane of reflection perpendicular to cart's track. Notice the difficulty in finding similar state-action pairs using only the observed rewards in the conventional reward setting: very long reward histories will be needed and the estimates will have many false positives, We define a reward shaping function for this domain as follows. To keep the symmetry finding tractable we discretize the state space into L levels along each dimension using uniformly spaced intervals, This is done only for reward assignment(agent still gets continuous input observations for the state). Thus, if the bounding box for position were $[-X_b, X_b]$, then each interval is of width $w = \frac{2X_b}{L}$ and the discrete dimension $x_d = k$ if $\frac{(2k+1)w}{2} \geq x \geq \frac{(2k-1)w}{2}$. The shaping function is defined as : $F(\theta, x, \omega, v) = 1 - \frac{(\theta_d^2 + x_d^2 + \omega_d^2 + v_d^2)}{(L-1)^2}$. Intuitively, this shaping motivates the agent to keep its coordinates near stable Cart-Pole configurations. We modify the 'CartPole-v0' environment available in the OpenAI Gym platform [4] for our experiments. The algorithms we experimented with are the DQN [17] agent and its proposed symmetric variant Symmetric DQN (Algorithm 1) agent. We use a discretization level $L = 9$ for the experiments, The maximum episode length is set to 1500 the replay memory size is set 100000 and the mini batch size of 128 is used. Agents are run un-

Table 1: Performance in Cart-Pole Domain

			DQN		SymDQN	
$\mathbf{l_0}$	**i**	$\boldsymbol{\Delta}$	**mean**	**max**	**mean**	**max**
1	5	0.8	112.1 ± 63.44	561.19 ± 200.60	$\mathbf{262.74 \pm 63.97}$	$\mathbf{783.45 \pm 288.95}$
1	5	0.5	91.38 ± 57.81	503.43 ± 223.65	$\mathbf{233.92 \pm 74.68}$	$\mathbf{682.27 \pm 221.83}$
1	4	0.8	105.93 ± 61.82	587.43 ± 186.78	$\mathbf{236.14 \pm 81.60}$	$\mathbf{652.82 \pm 248.91}$
2	5	0.8	80.89 ± 43.41	466.28 ± 194.65	$\mathbf{232.65 \pm 92.86}$	$\mathbf{831.32 \pm 342.33}$
2	5	0.5	117.14 ± 53.80	609.48 ± 214.90	$\mathbf{257.12 \pm 51.36}$	$\mathbf{620.70 \pm 203.83}$
2	4	0.8	88.65 ± 67.93	491.34 ± 197.59	$\mathbf{229.48 \pm 78.20}$	$\mathbf{703.88 \pm 267.35}$

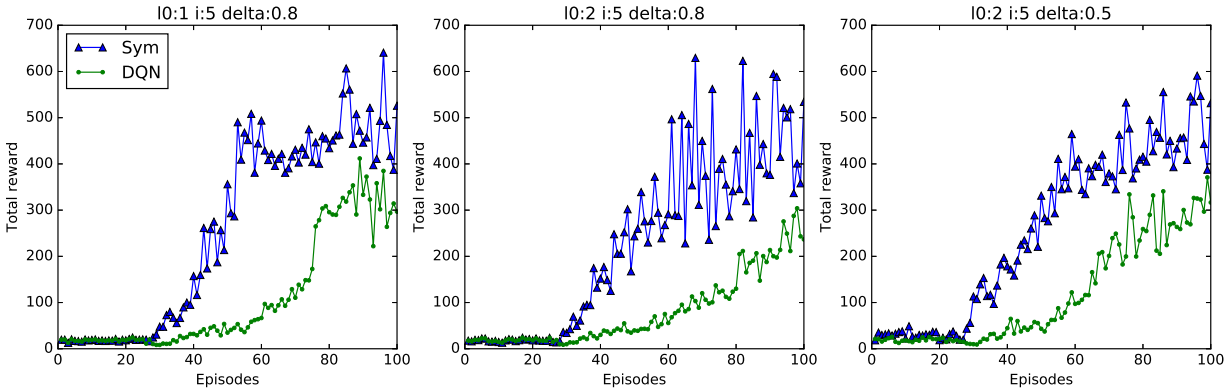

Figure 5: Variation of total reward with episodes.Parameter setting left to right:(1)$\mathbf{l_0 = 1, i = 5, \Delta = 0.8}$, (2)$\mathbf{l_0 = 2, i = 5, \Delta = 0.8}$, (3)$\mathbf{l_0 = 2, i = 5, \Delta = 0.5}$, all experiments used $\lambda = 1, \gamma = 0.99, \epsilon$ was decayed to 0.1 starting from 1 at rate 0.98

der a completely random policy for the first 25 episodes to ensure proper initialization of the replay memory.

Figure 5 shows the variation of total reward obtained with the number of episodes averaged over 15 iterations for three different parameter settings, It can be observed that the agent using algorithm 1(labeled Sym) learns the optimal behavior faster than the control(labeled DQN). Table 1 gives the mean and the maximum values of the total rewards obtained in an episode for the two algorithms averaged over 15 iterations, SymDQN 1 clearly performs much better in both the metrics. The difference in the mean values of the total rewards is a good measure of the relative regret a naive agent can recover when it uses reward based symmetry information from the environment.

6. CONCLUSION

In this paper we have proposed a novel framework for discovering environment symmetries and exploiting them for the paradigm of function approximation, The framework consists of two main components: The similarity detecting procedure which calculates similarity estimates between state action pairs from reward observations, and the similarity incorporating component which promotes learning of symmetric policies by imposing a symmetry cost. Our approach is scalable and requires minimal additional time and space overheads, Further we have proved the completeness of our similarity measure. Through extensive experimentation using deep neural networks on Grid world and Cart-Pole domains, we have shown that our method outperforms previously proposed methods and other agents not using symmetry information. Further we have shown that the benefits of using symmetry information while learning get more profound as the dimensionality of the environment increases as there is scope for multiple symmetry occurrences. Finally we also noticed the important role reward shaping played for the method.

We believe that incorporation of discovered environment symmetries as complementary knowledge in deep reinforcement learning is a promising direction of research and we hope to have justifiably motivated it through this work. It is a very natural way of leveraging prior experience and is frequently observed in human behavior. For future work we would like to explore methods for learning the symmetries functionally as closed form expressions using deep networks so that the tree based approach can be elegantly extended to more general scenarios.

APPENDIX

A. PROOFS

A.1 Proof of Theorem 1

PROOF. We give the proof sketch using a coupling argument. Given MDP M, Let $\{s_t^e, a_t^e, r_t^e\}_{t=0}^{T^e}$ be the random variables observed denoting states, actions and rewards respectively while following a symmetric policy π in an episode e of length T, Let $\hat{N_{sa}}$ be the number of times M transits though the state action pair s, a and $\hat{N_{\sigma,sa}}$ denote the frequency of observing reward sequence σ after such a transit after some number of episode runs(Thus $(\sigma, \hat{N_{\sigma,sa}}) \in \Pi_{sa,T^e}$). Clearly all the estimates in $\lim_{|episodes| \to \infty} \frac{\hat{N_{\sigma,sa}}}{\hat{N_{sa}}}$ will converge in distribution to the true marginals.

Now the assumption of uniform distribution over starting state allows us to define a coupled policy execution: For every episode e starting with s_0^e, a_0^e we begin an episode e' starting with $f(s_0^e), g_{s_0^e}(a_0^e)$. Further, for each action $a_t^e = \pi(s_t^e)$ taken in e we take action $a_t^{e'} = g_{s_t^e}(a_t^e)$ in e' and force the transition to $s_{t+1}^{e'} = f(s_{t+1}^e)$.Clearly both the sequences obey the transition dynamics of M as the former is driven by it and the latter conforms due to given symmetry equivalence 2 and thus form a coupling $\mathbb{C}(\{s_t^e, a_t^e, r_t^e\}_{t=0}^{T^e}, \{s_t^{e'}, a_t^{e'}, r_t^{e'}\}_{t=0}^{T^{e'}})$. Since we had started from symmetric state action pairs and take symmetrical transition from then on by 3 the reward sequence observed must be exactly identical $(r_0^e, ..r_{T^e}^e) = (r_0^{e'}, ..r_{T^e}^{e'})$, hence all reward based observations for symmetric pairs must be identical in particular: $\hat{N_{\sigma,sa}} = \hat{N_{\sigma,s'a'}}$, Finally from the definition 4 we have $\chi_{i,l_0}(\langle s, a\rangle, \langle s'a'\rangle) = 1$ in the coupled execution at all the times. Since $\mathbb{C}$ and the uncoupled case converge in distribution to same behavior in $\lim_{|episodes|\to\infty}$ the theorem is proved. □

B. DEFINITIONS

Let X, Y be sets, element $x \in X$, we then have the following constructs:

Definition 1. Partition: $B := \{b_i | b_i \subseteq X\}$ is a partition of X iff $(\cup_i b_i = X) \wedge (b_i \cap b_j = \emptyset | i \neq j)$. We denote the block to which x belongs by $[x]_B$

Definition 2. Coarseness: Let B_1, B_2 be two partitions of X then we say B_1 is coarser that B_2 written $B_1 \geq_c B_2$ iff $\forall x, x' \in X$, $[x]_{B_2} = [x']_{B_2} \Rightarrow [x]_{B_1} = [x']_{B_1}$

Definition 3. Projection: Let B be a partition of $Z \subseteq X * Y$, let $B(x)$ denote the distinct blocks of B containing pairs of which x is a component, we define $B|X$ the projection of B onto X as the partition of X which follows: $x, x' \in X \wedge ([x]_{B|X} = [x']_{B|X}) \iff B(x) = B(x')$

Definition 4. Equivalence Relation: An equivalence relation R on a X is a subset of XxX, denoted xRy to mean $(x, y \in R)$ which has following attributes:

- Reflexive: $xRx, \forall x \in X$
- Symmetric: $xRy \iff yRx, \forall x, y \in X$
- Transitive: $xRy \wedge yRz \Rightarrow xRz, \forall x, y, z \in X$

Definition 5. The equivalence class of x in an equivalence relation given by $[x]_R := \{y | xRy, y \in X\}$

C. ARCHITECTURES USED

All the networks were fully connected and used ReLU non linearity for hidden layers, input for Grid-World was a sparse coding of $|S| + |A|$

- Grid-World $9x9, 13x13$: $input \to 120 \to 40 \to 1$
- Grid-World $7x7x7, 5x5x5$: $input \to 300 \to 120 \to 1$
- Cart-Pole : $4 \to 100 \to 100 \to 2$

REFERENCES

[1] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, (5):834–846, 1983.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] A. Cotter, M. R. Gupta, and J. Pfeifer. A light touch for heavily constrained SGD. In *Proceedings of the 29th Conference on Learning Theory*, pages 729–771, 2016.

[6] T. Dean and R. Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, pages 106–111, 1997.

[7] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[8] S. Girgin, F. Polat, and R. Alhajj. Learning by automatic option discovery from conditionally terminating sequences. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 141:494, 2006.

[9] S. Girgin, F. Polat, and R. Alhajj. State similarity based approach for improving performance in RL. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 817–822, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[10] S. Girgin, F. Polat, and R. Alhajj. Improving reinforcement learning by using sequence trees. *Machine Learning*, 81(3):283–331, 2010.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[12] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *arXiv preprint arXiv:1604.06057*, 2016.

[13] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[15] A. McGovern and R. Sutton. Macro actions in reinforcement learning. Technical report, Technical Report TR 98-70, U. Mass. Amherst, 1998.

[16] E. A. Mcgovern. *Autonomous discovery of temporal abstractions from interaction with an environment.* PhD thesis, University of Massachusetts Amherst, 2002.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[18] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[19] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

[20] E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *International Conference on Learning Representations*, 2016.

[21] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *International Conference on Machine Learning*, volume 98, pages 463–471. Citeseer, 1998.

[22] B. Ravindran and B. AG. Symmetries and model minimization of Markov decision processes. 2001.

[23] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1312–1320, 2015.

[24] B. F. Skinner. *The behavior of organisms: an experimental analysis*. BF Skinner Foundation, 1990.

[25] J. Sorg and S. Singh. Linear options. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 31–38. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[26] M. Stolle and D. Precup. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 212–223. Springer, 2002.

[27] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[28] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[29] C. Szepesvari, R. S. Sutton, J. Modayil, S. Bhatnagar, et al. Universal option models. In *Advances in Neural Information Processing Systems*, pages 990–998, 2014.

[30] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. *AAAI Conference on Artificial Intelligence*, 2016.

[31] Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[32] D. Williams and G. Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[33] M. Zinkevich and T. Balch. Symmetry in Markov decision processes and its implications for single agent and multi agent learning. In *In Proceedings of the 18th International Conference on Machine Learning*. Citeseer, 2001.