Transfer in Inverse Reinforcement Learning for Multiple Strategies

Ajay Kumar Tanwani, Aude Billard
Learning Algorithms and Systems Laboratory (LASA)
Ecole Polytechnique Federale de Lausanne, Switzerland.
{ajay.tanwani, aude.billard}@epfl.ch

***Abstract*— We consider the problem of incrementally learning different strategies of performing a complex sequential task from multiple demonstrations of an expert or a set of experts. While the task is the same, each expert differs in his/her way of performing it. We assume that this variety across experts' demonstration is due to the fact that each expert/strategy is driven by a different reward function, where reward function is expressed as a linear combination of a set of known features. Consequently, we can learn all the expert strategies by forming a convex set of optimal deterministic policies, from which one can match any unseen expert strategy drawn from this set. Instead of learning from scratch every optimal policy in this set, the learner transfers knowledge from the set of learned policies to bootstrap its search for new optimal policy. We demonstrate our approach on a simulated mini-golf task where the 7 degrees of freedom Barrett WAM robot arm learns to sequentially putt on different holes in accordance with the playing strategies of the expert.**

I. Introduction

Inverse reinforcement learning, or rewards-driven imitation learning, is a paradigm for learning reward function from expert demonstrations [1], [2], [3], [4], [5], [6], [7]. Expert demonstrations provide a powerful means to bootstrap the learning process, subject to two notions of prime importance: 'what-to-imitate' and 'how-to-imitate', i.e., what is the intention of the expert in the demonstration and how to replicate the intended policy of the expert [8]. Inverse reinforcement learning assumes that the expert's intent is driven by rewards in a demonstration and aims to recover the control policy that can yield the same rewards as that of the expert. Rewards here are obtained by a linear combination of a set of known features representing the task.

It is well-known that humans vary widely in performing sequential decision-making tasks, possibly differing in their intentions or ways of gauging task-dependent features. This difference is a fundamental trait of natural selection that contributes to fitness and survival of an individual in changing environments. Consequently, there are often several useful ways of performing a task and how one assesses multiple criteria in a given situation yields the goodness of a decision. Despite this, most of the previous work in inverse reinforcement learning assumes single expert having the same intention in all the demonstrations – albeit with a few exceptions. In [9], the authors use an expectation-maximization approach to cluster similar strategies in the demonstrations where the number of clusters defined apriori represent the number of reward functions. Dimitrakakis and Rothkopf [10] generalize the Bayesian approach to learn multiple reward functions by considering two types of joint priors on reward functions and policies. Following above, Choi and Kim in [11] present a non-parametric Bayesian approach using the Dirichlet process mixture model to learn multiple reward functions. In this paper, we take a direct geometric approach to learn a convex set of optimal policies enclosing all expert strategies. This helps us to efficiently match any previously unseen expert strategy drawn from this set. Moreover, our method of learning multiple strategies is incremental and allows transfer of knowledge; contrary to all the batch learning approaches described above.

In this work, we are interested in learning multiple strategies of performing a task by observing several experts' demonstrations. We seek to endow our learner with the ability to mimic a variety of experts, irrespective of how different these experts are in their actions. We believe this ability is crucial to adapt to different situations/environments in an optimal way. Moreover, we exploit the fact that all the strategies share the same transition dynamics and only differ in the underlying reward function. This helps to reuse the previous experience and bootstrap incremental learning of multiple expert strategies.

II. Preliminaries

Consider the learner as an autonomous agent in a Markov Decision Process (MDP) represented by a tuple $< S, A, P_{sa}, \alpha, \gamma, \phi, w >$, where S is a finite set of N states; A is a set of M actions that the agent can take in a given state; $P_{sa} : S \times A \times S \rightarrow [0,1]$ describes the transition dynamics of the environment, i.e., $P_{sa} \triangleq Pr(s', a, s)$ is the probability of transitioning to state s' after taking action a in state s; $\alpha(s) : S \rightarrow [0,1]$ and $\sum_s \alpha(s) = 1$ is the initial state distribution from which the state s_0 is drawn; $\gamma \in \mathbb{R} \rightarrow [0,1)$ is the discount factor; $\phi(s) : S \rightarrow \mathbb{R}^k_{[0,1]}$ is the mapping from state s to a set of k task-dependent features[1]; $w \in \mathbb{R}^k_{[-1,1]}$ and $\|w\|_1 \leq 1$ defines the relative weights of the features. Different weights for the features yield different rewards while interacting with the environment, $R(s) = w^T \phi(s)$.

A policy $\pi \in \Pi$ defines the mapping from state to actions. A policy can be deterministic, $\pi(s) : S \rightarrow A$, in which case each state is mapped to a unique action, or a policy can be

[1]All the features are normalized to make their effect on the reward function comparable in a relative way.

stochastic in which case each state is mapped to a distribution over actions, $\pi(s,a) : S \times A \rightarrow [0,1]$ and $\sum_a \pi(s,a) = 1$. The policies we consider here are stationary as they depend only on current state and do not change with time. Note that a stochastic policy can be represented as a convex combination of deterministic policies and every convex combination of deterministic policies represents some stochastic policy (see Ch. 6 of [12]).

The *value-function* $V^\pi(s) : S \rightarrow [\frac{-1}{1-\gamma}, \frac{1}{1-\gamma}]$ measures the expected value of discounted sum of rewards that the agent gains starting from state s and following policy π:

$$V^\pi(s) = E\Big\{ \sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, a = \pi(s_t), \\ s' \sim P^\pi(.|s_t) \Big\}$$

where $P^\pi : S \times S \rightarrow [0,1]$, is the transition dynamics after fixing action in each state according to policy π. When modulated by the initial state distribution $\alpha(s)$, the value of a policy π reduces to a scalar defined by: $V^\pi = \sum_s \alpha(s) V^\pi(s)$ (note that we dropped the s in the parentheses). A policy π is optimal for the MDP if it satisfies:

$$\pi = \arg\max_{\pi \in \Pi} V^\pi$$

Similar to how the value-function gives an expectation over rewards in the long run, *feature expectation* vector, $\mu^\pi(s) : S \rightarrow \mathbb{R}^k_{[0,\frac{1}{1-\gamma}]}$, corresponds to the discounted sum of the features as the agent observes the sequence $s_0, s_1, s_2, \ldots$ starting from the state $s_0 = s$ following policy π.

$$\mu^\pi(s) = E\Big\{ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, a = \pi(s_t), \\ s' \sim P^\pi(.|s_t) \Big\}$$

Note that the reward function is linear in features, the value-function is also linear in feature expectations, parametrized by the same weight vector w, i.e., $V^\pi(s) = w^T \mu^\pi(s)$ and similarly for the initial state distribution, $V^\pi = w^T \mu^\pi$, where $\mu^\pi = \sum_s \alpha(s) \mu^\pi(s)$ [2].

The expert strategy is represented by its feature expectation μ^{π_E}. Given the expert's sequence of visited states over m runs $[s_0, s_1, s_2, \ldots]^m$, an empirical estimate of the expert's feature expectation can be computed as:

$$\hat{\mu}^{\pi_E} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^i)$$

[2]With slight abuse of notation, we later use bold-face notation to write equations in matrix form without parentheses as well. $\boldsymbol{\mu^\pi}$ for $N \times k$ matrix $[\mu^\pi(s_1) \ldots \mu^\pi(s_N)]^T$, and μ^π for column vector of dimension k, $\boldsymbol{\Phi}$ for the matrix of reward features, and $\boldsymbol{\alpha}$ for the initial-state distribution vector of dimension N.

III. Transfer in Learning Multiple Strategies

The main contribution of this paper is to incorporate the transfer of knowledge for boosting incremental learning of multiple expert strategies. We first formalize our problem statement in this section, followed by our multiple expert strategies learning algorithm and then explain the transfer of knowledge to speed up the learning process.

A. Problem Statement

Let Π_D be the set of all deterministic stationary policies available to the learner in a MDP as possible ways of executing a task. Each policy possibly gives a different feature expectation μ^π, among which the optimal ones maximize the value of a policy V^π for some w. The set of feature expectations $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_d} \subseteq \mu(\Pi_D)$ that are maximal for some w defines a convex hull $\text{Co}\{\mu(\Pi_D)\}$ in the feature expectation space. Ideally, we would like to learn all the optimal policies over this convex hull so that the learner can readily replicate any expert strategy by appropriately combining these optimal policies.

To make it concrete, suppose we can compute the set of feature expectations of all the optimal policies in Π_D, then we can approximate any expert strategy μ^{π_E} (in expectation) by constructing a mixed policy[3] that assigns a probability λ_i to the policy with feature expectation μ^{π_i}:

$$\mu^{\pi_E} = \sum_{i=1}^{|\pi_d|} \lambda_i \mu^{\pi_i}$$

Note that the deterministic stationary policies of Π_D alone do not constitute all the feasible strategies in the feature expectation space. By allowing ourselves to approximate the expert strategy with mixture of optimal policies, we do not limit the expert to be optimal or nearly-optimal in a deterministic way; otherwise we could select one optimal deterministic policy with feature expectation μ^{π_i} lying on the convex hull that is closest to μ^{π_E}. We only require the expert strategy to lie within the convex hull of feature expectations, and thereby, assume the expert to be optimal in a stochastic manner. In other words, the expert may sequentially optimize over different reward functions in his/her strategy.

However, learning all optimal policies in Π_D is in general intractable with $|\Pi_D| = A^S$. Moreover, not all the policies in the set lead to practically useful description of a task. To this end, we leverage upon the availability of the expert to address this challenge. Let us denote Π_E as the set of deterministic policies available to the expert where $|\Pi_E| \ll |\Pi_D|$ in general. Let $\Delta(\Pi_E)$ be the set of probability distributions (unknown) over the set Π_E from which the expert draws a finite number of strategies $\mu^{\pi_{E1}}, \mu^{\pi_{E2}}, \ldots, \mu^{\pi_{En}}$ as possible useful ways of demonstrating a task to the learner. The goal of the learner is to approximate the strategies demonstrated by the expert as $\mu^{\pi_{A1}}, \mu^{\pi_{A2}}, \ldots, \mu^{\pi_{An}}$ belonging to the probability distribution set $\Delta(\Pi_A)$, and after experiencing

[3]A mixed policy is executed by randomly selecting the policy π_i at $t = 0$ with probability λ_i ($\lambda_i \geq 0, \sum_i \lambda_i = 1$), and following it for the rest of the time.

a finite number of them, be able to approximate any new expert strategy drawn from $\Delta(\Pi_E)$[4]. The learner does so by finding the set of deterministic policies Π_A that is used to generate a mixed policy for matching any expert strategy by drawing from the associated distribution such that the performance of the learner is at least as good as that of the expert with a tolerance of ϵ_0:

$$|V^{\pi_E} - V^{\pi_A}| \leq \epsilon_0 \tag{1}$$

where $\epsilon_0 \geq 0$, $\pi_A \sim \Delta(\Pi_A)$, $\pi_E \sim \Delta(\Pi_E)$ and the expert's weight vector is unknown in the demonstrated strategy.

B. Learning Multiple Expert Strategies

Given an expert strategy μ^{π_E}, the learner seeks a policy π^A whose performance is close to that of the expert's policy π^E as given by Eq. (1). Based on the reward function used by the expert, there are two main approaches to recover the learner's policy: 1) learn the expert's reward function from demonstrations of the strategy explicitly and then compute the optimal policy for this reward function [1], [3], [7], or 2) match the feature expectations of the learner and the expert's policy irrespective of the reward function used [2], [13], [5]. We follow the latter approach in this work and present our results with the well-known *projection* algorithm [2].

The projection algorithm returns the learner's policy π^A for a given expert strategy such that $\|\mu^{\pi_E} - \mu^{\pi_A}\|_2 \leq \epsilon_1$, thereby yielding the same performance as that of the expert. From (1):

$$\begin{aligned} |V^{\pi_E} - V^{\pi_A}| &= w^T(\mu^{\pi_E} - \mu^{\pi_A}) \\ &\leq \|w\|_2 \|\mu^{\pi_E} - \mu^{\pi_A}\|_2 \\ &\leq 1 \cdot \epsilon_1 \end{aligned}$$

where the first inequality follows from Cauchy-Schwarz inequality: $|x^T y| \leq \|x\|_2 \|y\|_2$ and $\epsilon_1 \geq \epsilon_0$. The problem of matching a given expert strategy with respect to the unknown weight vector is, hence, transformed to a vector matching problem over feature expectations. The projection algorithm iteratively computes an optimal policy π^i with feature expectation μ^{π_i} for reward function, $R(s)^i = (w^i)^T \phi(s)$ in each iteration, $i = 1 \ldots T$. The weight vector w^i of the reward function is updated in each iteration such that the successive projected mapping $\bar{\mu}^i$ moves closer to the expert strategy μ^{π_E}, where $\bar{\mu}^i$ is the projection of μ^{π_E} on the line joining $\bar{\mu}^{i-1}$ and μ^{π_i}. Learning converges when the projected mapping is ϵ_1-close to the expert strategy μ^{π_E} and the weight vector changes no more (see Algorithm 1). At the end, the point μ^{π_E} is guaranteed to be close to the convex hull of feature expectation set of intermediate policies, $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_T}$, with μ^{π_A} being the closest point in that convex hull to μ^{π_E}.

Here we extend the idea of projection algorithm for learning multiple expert strategies. After computing the feature expectation set $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_T}$ corresponding to T iterations of the projection algorithm for expert strategy $\mu^{\pi_{E1}}$, the initial weight vector for $\mu^{\pi_{E2}}$ is selected along the line connecting $\mu^{\pi_{E2}}$ and the closest possible feature expectation achievable from the set $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_T}$ to $\mu^{\pi_{E2}}$. For the jth expert strategy, the initial weight is computed as: $w = \mu^{\pi_{Ej}} - u$, where u is obtained from the feature-expectation set as following:

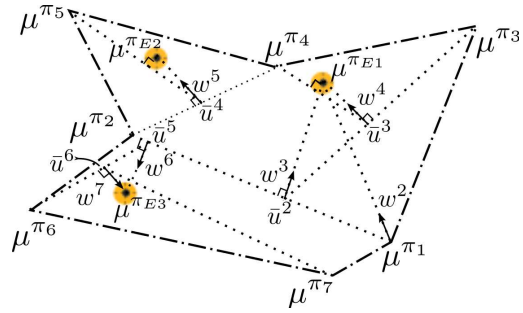

Fig. 1: Projection algorithm for multiple expert strategies

$$\begin{aligned} &\min_\mu \|\mu - \mu^{\pi_{Ej}}\|_2 \qquad \text{s.t.} \\ &\mu = \textstyle\sum_{i=1}^{(T\times j)} \lambda_i \mu^{\pi_i}, \qquad \textstyle\sum_{i=1}^{(T\times j)} \lambda_i = 1, \qquad \lambda_i >= 0 \end{aligned} \tag{2}$$

Note that if $\|w\|_2 < \epsilon_1$ after the above optimization, the algorithm terminates in the first iteration as $\mu^{\pi_{Ej}}$ can already be estimated from the existing feature expectation set of the learner.

C. Optimal Policy Transfer

There are two main issues in learning multiple expert strategies with the feature-matching approach: 1) it is computationally very expensive to find an optimal policy for a given reward function with weight w, and 2) the number of deterministic policies in the set Π_A can grow arbitrarily large for matching all the expert strategies. Consequently, the learner seeks to: 1) reuse the previously learned policies to achieve faster learning with a new reward function parametrized by w, and 2) store only distinct policies (we call them ϵ-better policies) that are possibly optimal for a wide range of weights. Previous work in [14] uses such transfer of knowledge to optimize average-reward per time step in hierarchical Semi-Markov Decision Processes. A more generic overview of transfer in reinforcement learning can be found in [15].

Let $\Pi_A^{(j)}$ be the set of stored optimal deterministic policies after learning the jth expert strategy. Given a new reward function with weight w, the learner chooses as initial policy π^{init} the one with the highest value in the set $\Pi_A^{(j)}$:

$$\pi^{init} = \arg \max_{\pi \in \Pi_A^{(j)}} (w^T \mu^\pi) \tag{3}$$

The initial policy π^{init} is the optimal policy for the given reward function if there exists no other policy whose performance is ϵ-better than the initial policy. The set of ϵ-better policies is characterized in the following Lemma:

[4]For simplicity, we assume that the new expert strategy during testing belongs to the convex set of already experienced expert strategies $\mu^{\pi_{E1}}, \mu^{\pi_{E2}}, \ldots, \mu^{\pi_{En}}$.

Lemma 1: Given a finite state space S, action set A, initial state distribution $\boldsymbol{\alpha}$, reward function $\boldsymbol{R}$, the optimal policy π with transition matrix $\boldsymbol{P^\pi}$ is ϵ-better than an initial policy π^{init} with transition matrix $\boldsymbol{P^{\pi_{init}}}$, if it satisfies:

$$\boldsymbol{\alpha}^T \left((I-\gamma \boldsymbol{P^\pi})^{-1} - (I-\gamma \boldsymbol{P^{\pi_{init}}})^{-1}\right) \boldsymbol{R} \geq \epsilon \quad (4)$$

Proof: The value of an ϵ-better policy is at least ϵ better than the value of π^{init}:

$$\begin{aligned} V^\pi - V^{\pi_{init}} &\geq \epsilon \\ ((\mu^\pi)^T - (\mu^{\pi_{init}})^T)w &\geq \epsilon \quad (5) \end{aligned}$$

$$\begin{aligned} \mu^\pi &= \sum_s \mu^\pi(s)\alpha(s) \\ &= \sum_s E(\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|s_0 = s, s' \sim P^\pi(.|s_t))\alpha(s) \\ &= \sum_s (\phi(s) + \gamma \sum_{s'} P^\pi \mu^\pi(s'))\alpha(s) \\ \mu^\pi &= \underbrace{(\boldsymbol{\Phi} + \gamma \boldsymbol{P^\pi}\boldsymbol{\mu^\pi})}_{\boldsymbol{\mu^\pi}}{}^T \boldsymbol{\alpha} \\ \mu^\pi &= \boldsymbol{\Phi}^T((\boldsymbol{I} - \gamma \boldsymbol{P^\pi})^{-1})^T \boldsymbol{\alpha} \\ (\mu^\pi)^T &= \boldsymbol{\alpha^T}(\boldsymbol{I} - \gamma \boldsymbol{P^\pi})^{-1}\boldsymbol{\Phi} \quad (6) \end{aligned}$$

Substituting Eq. (6) into Eq. (5) for $(\mu^\pi)^T$ and $(\mu^{\pi_{init}})^T$:

$$(\boldsymbol{\alpha^T}(\boldsymbol{I} - \gamma \boldsymbol{P^\pi})^{-1}\boldsymbol{\Phi} - \boldsymbol{\alpha^T}(\boldsymbol{I} - \gamma \boldsymbol{P^{\pi_{init}}})^{-1}\boldsymbol{\Phi})w \geq \epsilon \quad (7)$$

Rearranging gives the required result in (4) [5].

■

Lemma 1 gives the space of policies that are better than π^{init} for the given reward function with weight w. We now further narrow down this space by imposing constraints due to other policies in the set $\Pi_A^{(j)}$.

Definition 1: Given a set of optimal deterministic policies, $\pi^1, \pi^2, \ldots, \pi^T \in \Pi_A$, with feature expectations, $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_T} \in \mu(\Pi_A)$, corresponding to reward functions with weights, $w^1, w^2, \ldots, w^T$, the optimal policy π for reward function with weight w and feature expectation μ^π is an ϵ-better policy in Π_A if:

$$\begin{aligned} w^T(\mu^\pi - \mu^{\pi_i}) &\geq \epsilon \quad (8) \\ (w^i)^T(\mu^\pi - \mu^{\pi_i}) &\leq 0 \qquad i = 1, 2, \ldots, T \quad (9) \end{aligned}$$

The first set of constraints follows from the definition of the feature expectation μ^π of the optimal policy π for weight w:

$$\begin{aligned} \mu^\pi &= \arg \max_{\mu \in \mu(\Pi_D)} (w^T \mu) \\ \Rightarrow \qquad w^T\mu^\pi &\geq w^T\mu \qquad \forall \mu \in \mu(\Pi_D) \end{aligned}$$

[5]Note that the term $((\boldsymbol{I} - \gamma \boldsymbol{P^\pi})^{-1})^T \boldsymbol{\alpha}$ gives the state-visitation frequencies $\sum_a x(s,a)$ following policy π, where $x(s,a)$ is a feasible solution of the dual linear MDP. Consequently, one can easily switch between primal and dual variables.

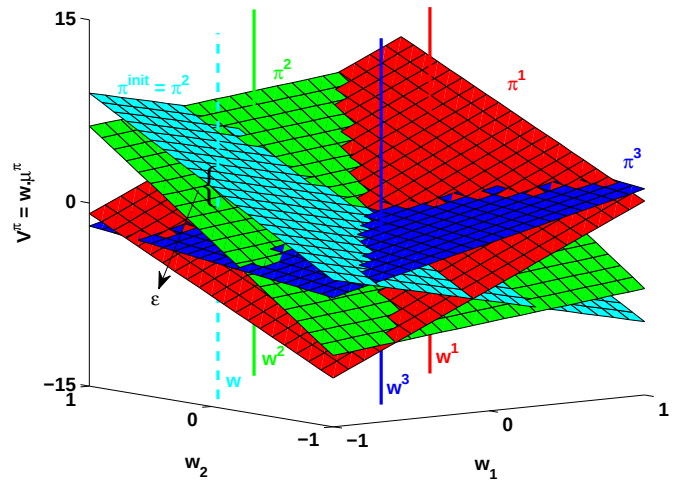

Fig. 2: 'Value-Surface' with $k = 2$ (best viewed in color). For a new reward function with weight w, value-surface gives the initial policy with the best weighted value. The surface is updated only if there exists a ϵ-better policy at w whose weighted value is less than the value of other optimal policies at $w^1, w^2, \ldots, w^T$.

For π to be stored, its value $w^T\mu^\pi$ has to be ϵ-better than the values of all the policies in the set Π_A at weight w: $w^T\mu^\pi \geq w^T\mu^{\pi_i} + \epsilon$ for $i = 1 \ldots T$. Rearranging yields the constraints in (8). Since the value V^π is linear in weights, the policy gives a weighted value of $(w^i)^T\mu^\pi$ at some other weight w^i. The weighted value $(w^i)^T\mu^\pi$ must be less than the optimal value $(w^i)^T\mu^{\pi_i}$ for π^i to be the optimal policy corresponding to weight w^i; otherwise π would be the optimal policy for weight w^i, i.e., $(w^i)^T\mu^\pi \leq (w^i)^T\mu^{\pi_i}$ for $i = 1 \ldots T$. Rearranging gives the constraints in (9). Further, adding constraints (8) and (9) and using Cauchy-Schwarz inequality gives a lower bound on the distance between w and other weight vectors in the set $w^1, w^2, \ldots, w^T$ for w to have an ϵ-better policy[6]:

$$\begin{aligned} (w - w^i)^T(\mu^\pi - \mu^{\pi_i}) &\geq \epsilon \\ \|w - w^i\|_2 \|\mu^\pi - \mu^{\pi_i}\|_2 &\geq \epsilon \\ \|w - w^i\|_2 &\geq \frac{\epsilon(1-\gamma)}{\sqrt{k}} \qquad i = 1 \ldots T \quad (10) \end{aligned}$$

Every policy adds a set of constraints for a new reward function with weight w to satisfy. The set $\mu^{\pi_1}, \mu^{\pi_2}, \ldots, \mu^{\pi_T}$ defines a convex hull Co$\{\mu(\Pi_A)\}$ in the feature expectation space and the resulting piecewise planar 'value-surface' gives the best policy value for each possible weight (see Fig. 2).

Note that Lemma 1 combined with the constraints in Definition 1 can be used to find an ϵ-better policy with a linear program; albeit very slow. In our implementation, we verify the existence of ϵ-better policy in three steps in this order: 1) satisfy (10) to check if there does not exist

[6]Remember that: $\mu^\pi \in \mathbb{R}^k_{[0, \frac{1}{1-\gamma}]} \Rightarrow \|\mu^\pi - \mu^{\pi_i}\|_2 \leq \frac{\sqrt{k}}{1-\gamma}$.

any w^i in the vicinity of w for which we already have the optimal policy, 2) there exists a μ such that the constraints in Definition 1 are satisfied, i.e.,

$$\begin{aligned} \text{Solve for } \mu \text{ s.t.} \quad & w^T(\mu - \mu^{\pi_{init}}) \geq \epsilon, && (11) \\ & (w^i)^T(\mu - \mu^{\pi_i}) \leq 0, \qquad i = 1, 2, \ldots, T \\ & 0 \preceq \mu \preceq \tfrac{1}{1-\gamma} \end{aligned}$$

Note that the use of $\mu^{\pi_{init}}$ at w also satisfies all μ^{π_i} in (8), and 3) find the optimal policy using the well-known value-iteration algorithm starting from π^{init} (any reinforcement learning algorithm can be used) and use Lemma (1) to decide whether to store or discard the optimal policy. If the verification fails at any of the above three steps, π^{init} is declared the optimal policy for w. The overall algorithm of learning multiple strategies from demonstrations is presented in Algorithm 1.

Algorithm 1 *Transfer in Learning Multiple Strategies*

Input: $< S, A, P_{sa}, \alpha, \gamma, \phi, \{\mu^{\pi_{E1}}, \mu^{\pi_{E2}}, \ldots, \mu^{\pi_{En}}\}, \epsilon >$

procedure LEARNER_TRAINING
Initialize $i := 1$, w^i s.t. $\|w^i\|_1 = 1$, $\Pi_A = \{\}$
$\bar{\mu}^i = \arg\max_{\mu \in \mu(\Pi_D)} \left((w^i)^T \mu\right)$
for $j = 1$ **to** $|\mu^{\pi_{En}}|$ **do**
  if $\Pi_A \neq \{\}$ **then**
    Solve (2) for $\mu := \min_{\mu \in \text{Co}\{\mu(\Pi_A)\}} \|\mu - \mu^{\pi_{Ej}}\|_2$
    $w^i = \mu^{\pi_{Ej}} - \mu$
    $\bar{\mu}^{i-1} = \mu$
  end if
  repeat
    if $i > 1$ **then**
      $\pi^{init} := \arg\max_{\pi \in \Pi_A} \left((w^i)^T \mu\right)$
      Verify three steps for existence of ϵ-better policy
      if three steps are verified **then**
        Add π^i to Π_A
      else
        $\pi^i = \pi^{init}$
      end if
      $\bar{\mu}^i = \bar{\mu}^{i-1} + \frac{(\mu^{\pi_i} - \bar{\mu}^{i-1})^T (\mu^{\pi_{Ej}} - \bar{\mu}^{i-1})}{(\mu^{\pi_i} - \bar{\mu}^{i-1})^T (\mu^{\pi_i} - \bar{\mu}^{i-1})} (\mu^{\pi_i} - \bar{\mu}^{i-1})$
    end if
    $w^{i+1} = \mu^{\pi_{Ej}} - \bar{\mu}^i$
    $i := i + 1$
  until $\|w^i - w^{i-1}\|_2$ is unchanged
end for
return set of learner policies Π_A

procedure LEARNER_TESTING
loop
  Expert demonstrates a strategy $\mu^{\pi_E} \sim \Delta(\Pi_E)$
  Learner finds a strategy $\mu^{\pi_A} \sim \Delta(\Pi_A) : \mu^{\pi_A} = \sum_{i=1}^{|\Pi_A|} \lambda_i \mu^{\pi_i}$, where λ_i is obtained by solving (2) with $(T \times j) = |\Pi_A|$
end loop

IV. Experimental Study

Experimental study is first performed on a grid world problem, followed by our sequential decision making task of playing mini-golf. The goal here is to asses the performance of optimal policy transfer in learning multiple expert strategies with different values of ϵ against the 'no transfer' case where each expert strategy is learned separately with the projection algorithm. The performance is evaluated using three metrics: 1) empirical error – distance between the estimated feature expectation of the expert and the learner averaged over n strategies, i.e., $\frac{1}{n}\sum_{j=1}^{n} \|\hat{\mu}^{\pi_{Ej}} - \hat{\mu}^{\pi_{Aj}}\|_2$, 2) CPU learning time, and 3) number of policies stored. We use the same discount factor of 0.9 in all our experiments. Moreover, we only iterate our algorithm for an expert strategy up to a maximum of 50 iterations.

A. Grid World

We first illustrate our approach in a conceptually simple grid world environment of 100×100 cells. Each cell represents a different state of the learner. In a given state, the learner can take 9 different actions corresponding to a move in all eight neighbouring directions or a stay in the same cell. Transition dynamics are stochastic with 0.7 probability of moving in the direction of desired action instead of a random one. Initial state distribution is uniform over all the states. Five features – radial basis functions with centres chosen randomly among states and width drawn in the interval $[1, 20]$ – are used to populate the feature space. Ten different reward functions are generated to simulate multiple experts by randomly assigning different weights to every feature in the interval $[-1, 1]$. We log the visited states sequence of 125 time steps from the optimal policy of every reward function in a demonstration and vary the number of sample demonstrations to study its effect on learning multiple strategies.

Fig. 3 (left) shows that the average empirical error over all strategies decreases sharply with the increase in the number of demonstrations, while it increases slightly with higher values of ϵ for a given number of sample demonstrations. The other two plots clearly indicate the advantage of optimal policy transfer with a magnitude of performance improvement in terms of required time and number of policies to learn all strategies. Note that the optimal policy transfer is useful even for the case of learning a single expert strategy.

B. Mini-Golf

Mini-golf, short for Miniature golf, is a competitive but enjoyable sport in which the players compete to strike a golf ball with a putter into a hole. The game is played on a small field with various fixed obstacles and unique variations. Different fields are marked with increasing order of the difficulty level and the players are required to complete each hole before moving on to the next one. The goal is to sink the ball into the hole from the tee area in as few shots as possible. Depending on the various features of the field, the task of estimating how to hit the ball in a given situation is a difficult task that requires a lot of skill from the expert. The

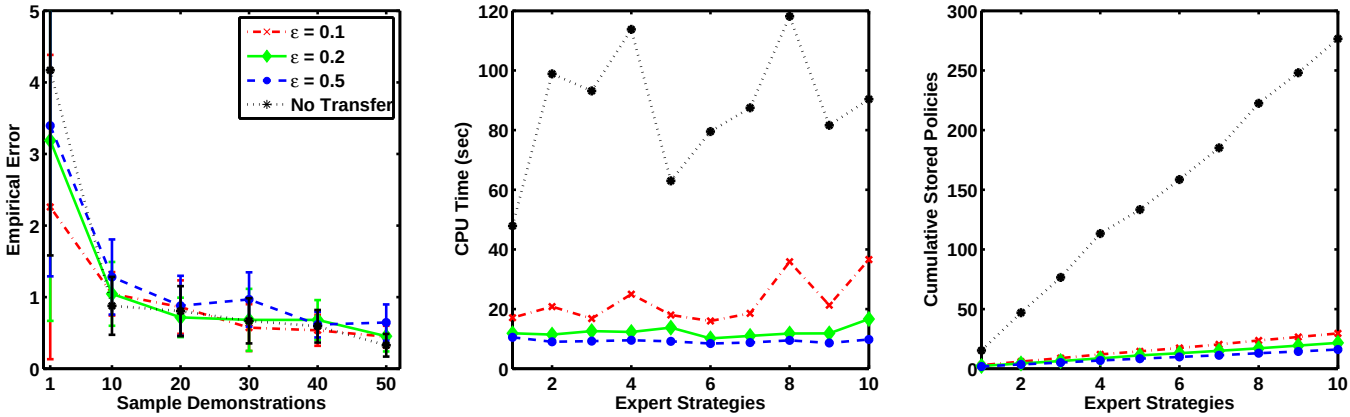

Fig. 3: Grid world results. Results are averaged over 5 iterations

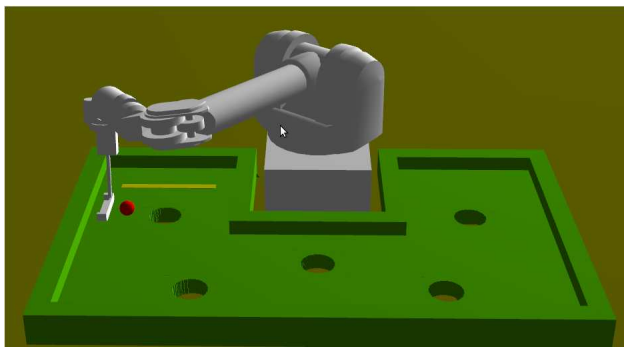

Fig. 4: Simulated mini-golf playing field

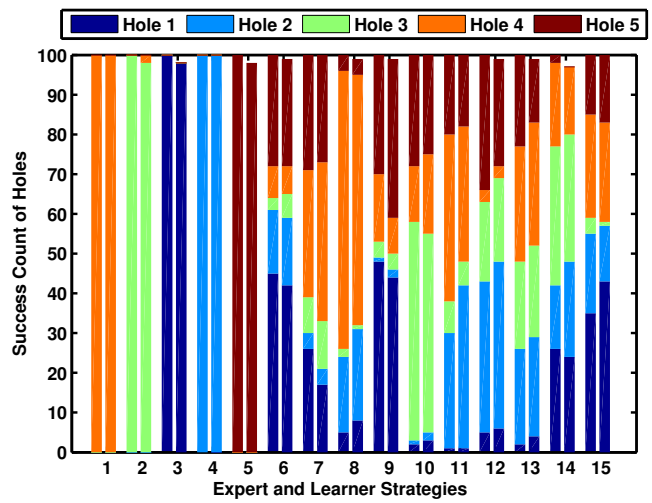

Fig. 5: Comparison of first 15 expert and learner strategies for 100 episodes with $\epsilon = 0.1$. For every strategy number, the first bar gives the success count of holes for the expert, the second bar gives the learner's response to the expert's strategy. First five strategies correspond to the training set, other mixed strategies are from the testing set.

expert has to plan a number of aspects such as reflections of the boundaries, number of shots and intermediate ball positions for every hole separately. In this section, we use the knowledge of different experts to teach the learner how to putt the golf ball into different holes.

1) Learning Problem: We are interested in learning all the useful playing strategies for the learner from the expert. The learner is a 7-degrees of freedom Barrett WAM robot arm and the expert is a computer program that knows how to sink the ball in different holes. The simulated mini-golf environment is shown in Fig. 4. To simulate various strategies of the expert, we have 5 different holes in one field. To find useful playing strategies, the expert computes 100 optimal policies for randomly chosen weights and selects one optimal policy for each hole based on its success count and policy-value. For brevity, we fix 100 demonstrations of length equal to 50 time steps for each optimal policy to estimate the feature expectation of expert's strategies, $\hat{\mu}^{\pi_{Ej}}, j = 1 \ldots 5$ (same setting is used to empirically estimate the learner strategies). The learner is required to learn the set of deterministic policies Π_A from which it can approximate any randomly chosen distribution over the 5 expert strategies. In other words, sink the ball in each hole same number of times as the expert does in his/her strategy.

2) State, Action and Feature Space: The state-space corresponds to the $2-$dimensional position of the ball in the grid, $|S| = 81 \times 56 = 4536$. The action-set corresponds to 4 hitting directions at right angles to one another and 6 different hitting speeds, $|A| = 24$. The feature space is 13-dimensional, where first 8-dimensions give distance of the ball to each wall segment, and other 5-dimensions give distance of the ball to each hole. The features are scaled such that $\phi(s) \leq 1$. Intuitively speaking, an ideal strategy chooses the intermediate ball positions in a way that keeps the ball maximally away from all other holes and wall segments, while sinks the ball in the desired hole in least number of shots. The initial state distribution is uniform on the tee area marked with the yellow line in Fig. 4. An episode of play corresponds to 50 shots. The ball position is randomly reset on the tee area every time the episode ends or the ball sinks into a hole.

TABLE I: Performance comparison of projection algorithm for learning multiple strategies with and without optimal policy transfer. Results are averaged over 5 iterations

Learning Multiple Strategies	CPU Time (sec)	Stored Policies	Empirical Error (Training)	Empirical Error (Testing)
No Transfer	333.53	250	0.901 ± 0.117	0.931 ± 0.096
$\epsilon = 0.1$	310.49	14.2	0.972 ± 0.089	0.778 ± 0.03
$\epsilon = 0.2$	188.66	12	0.971 ± 0.068	0.797 ± 0.032
$\epsilon = 0.5$	78.05	8.2	1.025 ± 0.073	0.794 ± 0.038

3) Results and Discussions: We design our experiments as follows: the learner is required to learn the 5 expert strategies from their estimated feature expectations using our proposed algorithm in the training phase. During testing, the expert then draws 50 mixed strategies each corresponding to a random distribution over pure expert strategies, and the learner is asked to replicate the expert's strategy.

Table I gives a performance comparison of the projection algorithm for learning multiple strategies with and without optimal policy transfer. The algorithm with 'no transfer' fails to converge for each of the 5 expert strategies in 50 iterations, leading to a large number of stored policies. Increasing values of ϵ depict a similar trend as in the grid world problem, however, the CPU learning times are more closer to one another. This is because the value-iteration algorithm takes somewhat shorter time in this case to compute an optimal policy even if it is initialized randomly. In more realistic scenarios where sample collection process is expensive and optimal policy needs to be computed online, the difference in learning times would be largely amplified. By reducing the time to compute optimal policy, our approach would scale gracefully with moderately high dimensions. A direct comparison with learning multiple expert strategies on the real robot is, however, subject to our future work.

Fig. 5 gives a measure of the ability of the learner to replicate previously unseen expert strategies. It is seen that after learning the 5 expert strategies corresponding to sinking the ball in each hole separately during training, the learner is able to successfully replicate all the mixed strategies of the expert in the testing phase.

V. Conclusions

We presented the learner as an autonomous agent that can learn multiple ways of doing a task by observing the expert, while making use of the previously gathered experience. We tested our algorithm on the mini-golf task to verify the proficiency of the learner against different playing strategies of the expert.

In this work, we evaluate the ability of the learner to match any complex strategy demonstrated by the expert. We are also interested in the online version of our formulated problem where the expert's choice of subsequent strategy selection guides the learning process of the learner to reach equilibrium. While having discrete state-action space with known transition dynamics can often be restrictive for real-world tasks, we plan to relax these assumptions with continuous states and actions for model-based/model-free interaction with the environment in our future work.

Acknowledgement

This work was supported in part by EU Project First-MM (FP7/2007 – 2013) under grant agreement number 248258 and by the National Center of Competence in Research in Robotics (NCCR Robotics).

References

[1] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[3] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *In Proceedings of the 23rd International Conference on Machine Learning (ICML06*, 2006.

[4] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the 20th international joint conference on Artifical intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591.

[5] U. Syed and R. Schapire, "A game-theoretic approach to apprenticeship learning," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 1449–1456.

[6] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, ser. AAAI'08. AAAI Press, 2008, pp. 1433–1438.

[7] G. Neu and C. Szepesvári, "Apprenticeship learning using inverse reinforcement learning and gradient methods," *CoRR*, vol. abs/1206.5264, 2012.

[8] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," Handbook of Robotics, chapter 59, 2008.

[9] M. Babes, V. Marivate, M. Littman, and K. Subramanian, "Apprenticeship learning about multiple intentions," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11. ACM, 2011, pp. 897–904.

[10] C. Dimitrakakis and C. A. Rothkopf, "Bayesian multitask inverse reinforcement learning," in *Proceedings of the 9th European conference on Recent Advances in Reinforcement Learning*, ser. EWRL'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 273–284.

[11] J. Choi and K.-E. Kim, "Nonparametric bayesian inverse reinforcement learning for multiple reward functions," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 314–322.

[12] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[13] U. Syed, R. Schapire, and M. Bowling, "Apprenticeship Learning Using Linear Programming," in *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, 2008, pp. 1032–1039.

[14] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning," *Machine Learning*, vol. 73, no. 3, pp. 289–312, 2008.

[15] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. 1, pp. 1633–1685, 2009.