Learning from Demonstration Using MDP Induced Metrics*

Francisco S. Melo[1] and Manuel Lopes[2]

[1] INESC-ID/Instituto Superior Técnico
TagusPark - Edifício IST
2780-990 Porto Salvo, Portugal
fmelo@inesc-id.pt
[2] University of Plymouth
Plymouth, Devon, PL4 8AA, UK
manuel.lopes@plymouth.ac.uk

Abstract. In this paper we address the problem of learning a policy from demonstration. Assuming that the policy to be learned is the optimal policy for an underlying MDP, we propose a novel way of leveraging the underlying MDP structure in a kernel-based approach. Our proposed approach rests on the insight that the MDP structure can be encapsulated into an adequate state-space metric. In particular we show that, using MDP metrics, we are able to cast the problem of learning from demonstration as a classification problem and attain similar generalization performance as methods based on inverse reinforcement learning at a much lower online computational cost. Our method is also able to attain superior generalization than other supervised learning methods that fail to consider the MDP structure.

1 Introduction

In this paper we address the problem of learning a policy from demonstration. This problem has garnered particular interest in recent years, as the ability of non-technical users to program complex systems to perform customized tasks rests heavily on fast and efficient solutions to this particular problem. The literature on this topic is too extensive to review here, and we refer to [3, 13] for reviews.

We formalize the problem of learning a policy from demonstration as a standard supervised learning problem: given a demonstration of some target policy consisting of (possibly perturbed) samples thereof, we must recover the target policy from the observed samples. These samples consist of the actions that the agent (learner) should take at specific situations; the learning agent must generalize the observed actions to situations never observed before.

* Work partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (INESC-ID multiannual funding) through the PIDDAC Program funds and by the projects PTDC/EEA-ACR/70174/2006 and Handle (EU-FP7-ICT-231640).

Several works follow a similar approach to the problem of learning from demonstration. Examples go back to the pioneer work of Pomerleau [18], in which an artificial neural network is used to have a vehicle learn how to steer from sample steering motions by a human operator. Other related approaches include [4,22].

When envisioning real world scenarios – *e.g.,* when a human user must teach an agent some target task, – the algorithm must exhibit good generalization ability, and hence all available information on the structure of the problem should be exploited. Our contribution in this paper addresses the latter issue. We resort to *MDP metrics* [7,27] as a way to add additional structure to the problem of learning from demonstration. In particular, and unlike other supervised learning approaches to this problem, we assume that the target policy is the optimal policy for some underlying *Markov decision process* (MDP), whose dynamics are known. We thus propose the use of this underlying MDP structure to improve the generalization ability of our learning algorithm.

While many supervised learning methods depend critically on the definition of *features* that capture useful similarities between elements in input-space, MDP metrics naturally achieve this in the class of problems considered in this paper, by identifying two states as "similar" if policies can safely be generalized from one to the other.[1] In this sense, our approach is also close to *inverse reinforcement learning* [17,19] and related approaches (see, e.g., [1] and references therein). However, our method avoids one bottleneck associated with the IRL-based methods. In fact, IRL-based methods are typically iterative and require solving one different MDP per iteration (see Section 3 for details) rendering these methods computationally expensive in tasks involving large domains.

The remainder of the paper is organized as follows. Section 2 formalizes the problem of learning a policy from demonstration. Section 3 reviews IRL-based and supervised learning approaches to this problem. We introduce our method in Section 4 and illustrate its application in Section 5. Section 6 concludes.

2 Formalizing Learning from Demonstration

We now formalize the problem of learning a policy from demonstration. Throughout the paper, a *policy* is understood as a mapping $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, where $\mathcal{X}$ is some finite set of states, $\mathcal{A}$ is a finite set of actions and $\Delta(\mathcal{A})$ is the set of all probability distributions over $\mathcal{A}$. A policy π thus maps each state $x \in \mathcal{X}$ to a distribution over actions. Intuitively, it can be thought of as a decision rule that an agent/decision-maker can follow to choose the actions in each state. All throughout the paper we focus on problems with finite $\mathcal{X}$, common in scenarios of routing, man-machine interfaces or robot control using high-level skills/options.

[1] This is particularly evident in scenarios in which the input space is discrete. In fact, problems in continuous domains typically assume the input space as some subset of $\mathbb{R}^p$, for which natural metrics exist. The same does not occur in discrete domains, where the notion of two states being close is less immediate.

A *demonstration* is a set $\mathcal{D}$ of state-action pairs, $\mathcal{D} = \{(x_i, a_i), i = 1, \ldots, N\}$, where the states are independently randomly sampled from $\mathcal{X}$, and the corresponding actions are sampled according to the target policy π_{target}. Formally, denoting by $\mathsf{Uni}(\cdot)$ the uniform probability distribution over $\mathcal{X}$, we have that the samples $(x_i, a_i), i = 1, \ldots, N$ are i.i.d. according to

$$\mathbb{P}\left[(X, A) = (x, a)\right] = \mathsf{Uni}(x)\pi_{\text{target}}(x, a),$$

where $(X, A) = (x, a)$ denotes the event that the pair (x, a) is sampled.[2]

We assume that the target policy π_{target} used to generate the demonstration is either the optimal policy for an underlying *Markov decision process* (MDP) or a perturbation thereof. An MDP is herein represented as a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, where $\mathcal{X}$ and $\mathcal{A}$ are as defined above, $\mathsf{P}(x, a, y)$ denotes the probability of moving to state y after choosing action a at state x and $r(x)$ is the reward that the agent receives upon reaching state x. The constant γ is a discount factor. We recall that, given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, the value associated with a policy π in state $x \in \mathcal{X}$ is

$$V^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(X_t) \mid X_0 = x\right],$$

where X_t denotes the state of the MDP at time instant t and the expectation $\mathbb{E}_\pi\left[\cdot\right]$ is taken with respect to the distribution over trajectories of the chain induced by π. The optimal policy for an MDP is thus the policy π^* that verifies, for all $x \in \mathcal{X}$ and all π, $V^{\pi^*}(x) \geq V^\pi(x)$. We assume the transition probabilities for the MDP, encapsulated in the matrix P, to be known to the learner. The problem of learning a policy from demonstration can thus be roughly described as that of determining a policy that best approximates the target policy, π_{target}, given a demonstration $\mathcal{D}$ and knowledge of the MDP transition probabilities P.

In our finite action setting, we can treat each action $a \in \mathcal{A}$ as a *class label*, and a policy is essentially a *discriminant function* that assigns class labels to the states in $\mathcal{X}$. In other words, the policy $\hat{\pi}$ computed by our learning algorithm can be interpreted as classifier that assigns each label $a \in \mathcal{A}$ to a state $x \in \mathcal{X}$ according to the probabilities $\mathbb{P}\left[A = a \mid X = x\right] = \hat{\pi}(x, a)$. We henceforth use the designations "classifier" and "policy" interchangeably. We define the 0-1 *loss function* $\ell : \mathcal{A} \times \mathcal{A} \to \{0, 1\}$ as $\ell(a, \hat{a}) = 1 - \delta(a, \hat{a})$, where $\delta(\cdot, \cdot)$ is such that $\delta(a, \hat{a}) = 1$ if $a = \hat{a}$ and 0 otherwise. Our algorithm must then compute the classifier $\hat{\pi}$ that minimizes the *misclassification rate*,

$$\mathcal{E} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\hat{\pi}}\left[\ell(a_i, a)\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{a \in \mathcal{A}} \ell(a_i, a)\hat{\pi}(x_i, a), \tag{1}$$

[2] There may be applications in which the above representation may not be the most adequate (*e.g.,* in some robotic applications). However, this discussion is out of the scope of the paper and we instead refer to [3].

where (x_i, a_i) is the ith sample in the dataset $\mathcal{D}$. To this purpose, we resort to a kernel-based learning algorithm from the literature [14].[3]

Our contribution arises from considering a kernel obtained from a metric over $\mathcal{X}$ that is *induced by the MDP structure*. This *MDP metric* provides a natural way of "injecting" the MDP structure in the learning algorithm, leading to an improved performance when compared to other metrics that ignore the MDP. To put the work of this paper in context, the following section reviews the main ideas behind IRL-based approaches to the problem of learning a policy from demonstration. Also, it shows the supervised learning method used in our experiments to assess the usefulness of MDP metrics in the setting of this paper.

3 IRL and Supervised Learning Approaches

In this section we review some background material on (supervised) learning from demonstration and IRL. As will become apparent from our discussion, the underlying MDP used in IRL approaches is a rich structure that, if used adequately, leads to a significant improvement in estimating the target policy. This observation constitutes the main motivation for the ideas contributed in the paper. The section concludes with a brief revision of MDP metrics.

3.1 Inverse Reinforcement Learning

To describe the main ideas behind IRL-based approaches to the problem of learning a policy from demonstration, we introduce some additional concepts and notation on Markov decision processes. Given a policy π and an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, we have

$$V^\pi(x) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_\pi(x, y) V^\pi(y)$$

where $\mathsf{P}_\pi(x, y) = \sum_{a \in \mathcal{A}} \pi(x, a) \mathsf{P}(x, a, y)$. For the particular case of the optimal policy π^*,

$$V^*(x) = r(x) + \max_{a \in \mathcal{A}} \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(x, a, y) V^*(y).$$

The Q-function associated with policy π is defined as

$$Q^\pi(x, a) = r(x) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(x, a, y) V^\pi(y)$$

[3] We note that, in an MDP, the impact of misclassified actions on the performance of the agent heavily depends on which state such misclassification occurred. If information regarding the "relevance" of states in terms of performance is available, then such information can be integrated into the error $\mathcal{E}$ by weighting differently the loss at different states. In any case, we emphasize that the performance criterion for our learning algorithm is concerned with classification accuracy rather than with a reward-based performance.

Inverse reinforcement learning (IRL) deals with the inverse problem to that of an MDP: *Given a policy π_{target} and the model $(\mathcal{X}, \mathcal{A}, \mathsf{P}, \gamma)$, IRL seeks to determine a reward function r^* such that π_{target} is an optimal policy for the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r^*, \gamma)$.*

IRL was first formalized in the seminal paper by Ng and Russel [17]. Among other things, this paper characterizes the solution space associated with a target policy π_{target} as being the set of rewards $\mathbf{r}$ verifying

$$\big(\mathsf{P}_\pi - \mathsf{P}_a\big)\big(\mathbf{I} - \gamma \mathsf{P}_\pi\big)^{-1}\mathbf{r} \succeq \mathbf{0}, \tag{2}$$

where we have denoted by $\mathbf{r}$ the column vector with xth component given by $r(x)$. One interesting aspect of the condition (2) is that, although trivially met by some reward functions, it still provides a restriction on the reward space arising solely from considering the structure of the MDP. In other words, by considering the structure of the MDP it is possible to restrict the rewards that are actually compatible with the provided policy.

The sample-based approach to the problem of learning from demonstration considered in this paper, however, is closest to [19,15]. In these works, the demonstration provided to the algorithm consists in perturbed samples from the optimal policy associated with the target reward function. Specifically, in [19], the distribution from which the samples are obtained is used as the *likelihood function* in a Bayesian setting. The paper then estimates the posterior distribution $\mathbb{P}[\mathbf{r} \mid \mathcal{D}]$ using a variant of the Monte-Carlo Markov Chain algorithm. In [15], on the other hand, the authors adopt a gradient approach to recover the reward function that minimizes the empirical mean squared error with respect to the target policy.

For future reference, we review the latter method in more detail. Roughly speaking, the working assumption in [15] is that there is one reward function, r_{target}, that the agent must estimate. Denoting the corresponding optimal Q-function by Q^*_{target}, the paper assumes the demonstrator will choose an action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$ with probability

$$\mathbb{P}[A = a \mid X = x, r_{\text{target}}] = \frac{e^{\eta Q^*_{\text{target}}(x,a)}}{\sum_{b \in \mathcal{A}} e^{\eta Q^*_{\text{target}}(x,b)}}, \tag{3}$$

where η is a non-negative constant, henceforth designated as *confidence parameter*. Now given a demonstration $\mathcal{D} = \{(x_i, a_i), i = 1, \ldots, N\}$ obtained according to the distribution in (3), the algorithm proceeds by estimating the reward function r that minimizes the error

$$\mathcal{E} = \frac{1}{N}\sum_i \big(\tilde{\pi}(x_i, a_i) - \hat{\pi}_r(x_i, a_i)\big)^2,$$

where $\tilde{\pi}(x_i, a_i)$ is the empirical frequency of action a_i in state x_i and $\hat{\pi}_r$ is the policy estimate, corresponding to the optimal policy for the MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$. Minimization is achieved by (natural) gradient descent, with respect to the parameters of $\hat{\pi}_r$ – the corresponding reward function $\mathbf{r}$. The method thus proceeds by successively updating the reward

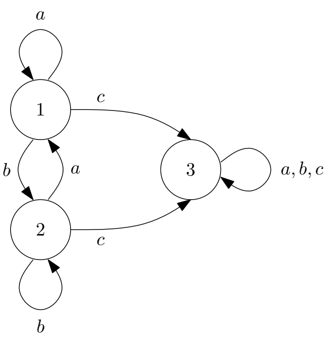


Fig. 1. Transition diagram for a simple MDP with 3 states, 3 actions and deterministic transitions. Even if the reward for this MDP is unknown, the demonstration of an optimal action in state 1, say action c, immediately implies that this action is also optimal in state 2. In fact, if c is optimal in state 1, then $r(3) > r(1)$ and $r(3) > r(2)$ – otherwise c would not be optimal in state 1. But then, the action in state 2 should also be c.

$$r^{(k+1)} = r^{(k)} - \alpha_t \tilde{\nabla}_r \mathcal{E}(r^{(k)}),$$

where $\tilde{\nabla}_r \mathcal{E}(r^{(k)})$ is the natural gradient of $\mathcal{E}$ with respect to r computed at $r^{(k)}$.

We conclude by noting that the computation of the natural gradient $\tilde{\nabla}_r \mathcal{E}(r^{(k)})$ at each iteration k requires solving the corresponding MDP $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r^{(k)}, \gamma)$. This need to solve an MDP at each iteration makes this method computationally expensive for large problems.

3.2 Supervised Learning

There is a significant volume of work on supervised learning approaches to the problem of learning a policy from demonstration. That literature is far too extensive to be reviewed here, and we refer to [3,13] for more detailed accounts.

A "pure" supervised learning approach addresses the problem of learning a policy from demonstration as a standard classification problem: *Given a set $\mathcal{D}$ of state-action pairs, $\mathcal{D} = \{(x_i, a_i), i = 1, \ldots, N\}$, sampled as detailed in Section 2, we want to compute a discriminant function $\hat{\pi}$ that minimizes the misclassification rate $\mathcal{E}$ in* (1).

Interestingly, even when the target policy is assumed optimal for some MDP, most supervised learning approaches to the problem of learning a policy from demonstration eventually ignore the underlying MDP structure in the learning process (see, for example, [4]). This means that such supervised learning algorithms do not actually require any particular knowledge of the underlying structure of the problem to function. Of course, when such information is available, it can provide valuable information for the learning algorithm and the learning algorithm should be able to accommodate such information.

In Section 4 we propose a principled approach to leverage the MDP structure in a kernel-based learning algorithm. Our proposed approach rests on the insight that the MDP structure can be encapsulated into an *input-space metric* that the learning algorithm can use to generalize along the (implicit) MDP structure. We propose using a metric that "projects" the structure of the MDP in such a way that the target policy in two states that are "close" is likely to be similar. This notion is illustrated in the example of Fig. 1.

We postpone to Section 4 the detailed description of how to encapsulate the MDP structure in the input-space metric and conclude this section by describing the learning algorithm used in our experiments.

Going back to the formulation in Section 2, we are interested in computing a classifier $\hat{\pi}$ (a distribution over the set of class labels – in this case, the action set $\mathcal{A}$) that varies along the state-space $\mathcal{X}$. In particular, we are interested in a learning algorithm that takes advantage of the metric structure in $\mathcal{X}$ to be described next. Possible methods for this include kernel logistic regression [28], beta regression [14] and others. We adopt the trivial extension of the latter to the multi-class case due to its computational efficiency and the possibility of including prior information.

At each state $x \in \mathcal{X}$, the desired classifier $\hat{\pi}$ is simply a multinomial distribution over the set of possible actions. Using a standard Bayesian approach, we compute a posterior distribution over each of the parameters of this multinomial using the corresponding conjugate prior, the Dirichlet distribution. For notational convenience, let us for the moment denote the parameters of the multinomial at a state $x \in \mathcal{X}$ as a vector $\mathbf{p}(x)$ with ath component given by $\mathbf{p}_a(x)$. Using this notation, we want to estimate, for each $x \in \mathcal{X}$, the posterior distribution $\mathbb{P}\left[\mathbf{p}(x) \mid \mathcal{D}\right]$. Let x_i be some state observed in the demonstration $\mathcal{D}$, and let $n_a(x_i)$ denote the number of times that, in the demonstration, the action a was observed in state x_i. Finally, let $n(x_i) = \sum_a n_a(x_i)$. We have

$$\begin{aligned}\mathbb{P}\left[\mathbf{p}(x_i) \mid \mathcal{D}\right] &\propto \mathsf{Multi}(\mathbf{p}_1(x_i), \mathbf{p}_2(x_i), \ldots, \mathbf{p}_{|\mathcal{A}|}(x_i))\mathsf{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{A}|}) \\ &= \frac{n(x_i)!}{\prod_{a\in\mathcal{A}} n_a(x_i)!} \prod_{a\in\mathcal{A}} \mathbf{p}_a(x_i)^{n_a} \frac{1}{B(\boldsymbol{\alpha})} \prod_{a\in\mathcal{A}} \mathbf{p}_a(x)^{\alpha_a - 1}\end{aligned}$$

In other words, the posterior distribution of $\mathbf{p}(x_i)$ is also a Dirichlet distribution with parameters $n_a + \alpha_a, a = 1, \ldots, |\mathcal{A}|$. In order to generalize $\mathbb{P}\left[\mathbf{p}(x) \mid \mathcal{D}\right]$ to unvisited states $x \in \mathcal{X}$, and following the approach in [14], we assume that the parameters of the Dirichlet distribution depend smoothly on x. From the metric structure of $\mathcal{X}$ we define a kernel on $\mathcal{X}$, $\mathbf{k}(\cdot, \cdot)$, and use standard kernel regression to extrapolate the parameters of the Dirichlet from the training dataset to unvisited states [23]. Specifically, for any query point $x^* \in \mathcal{X}$ and all $a \in \mathcal{A}$, we have

$$\hat{n}_a(x^*) = \sum_i \mathbf{k}(x^*, x_i) n_a(x_i) + \alpha_a. \tag{4}$$

Finally, the posterior mean of the distribution over the parameters $\mathbf{p}(x^*)$ – that we will henceforth use as our classifier at x^*, $\hat{\pi}(x^*, \cdot)$ – is given by

$$\hat{\pi}(x^*, a) = \mathbb{E}\left[\mathbf{p}_a(x^*) \mid \mathcal{D}\right] = \frac{\hat{n}_a(x^*)}{\sum_b \hat{n}_b(x^*)}, \tag{5}$$

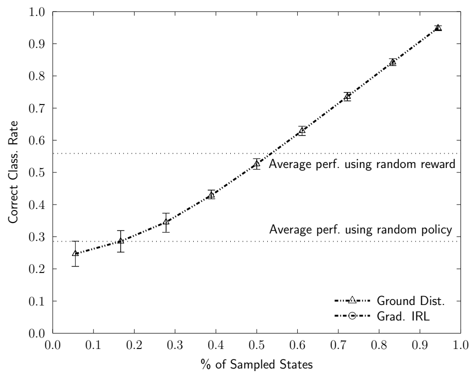


Fig. 2. Classification rate of the method above for different demonstration sizes. The horizontal lines correspond to the performances of a random policy and a policy obtained from the underlying MDP with a random reward.

for all $a \in \mathcal{A}$. In the next section, we introduce our main contribution, proposing a principled way to encapsulate the MDP structure in a suitable metric for $\mathcal{X}$ that can then be used to define the kernel $\mathbf{k}(\cdot, \cdot)$.

To conclude this subsection, we briefly summarize the main differences between IRL-based approaches and "pure" SL-based approaches. IRL-based approaches use the knowledge of the MDP model to search the space of reward functions for one specific reward function that yields an optimal policy that best matches the provided demonstration. For each reward function "tested", these methods solve the corresponding MDP, and so each iteration of IRL-based methods requires solving an MDP. However, the very structure of the MDP significantly reduces the space of possible policies, allowing these methods to exhibit very good generalization.

"Pure" SL-based methods, on the other hand, typically do not assume any underlying MDP model, and as such do not require solving any such model. This renders these methods more computationally efficient in large domains. However, the lack of clear domain knowledge often causes poorer generalization than IRL-based methods.

To illustrating the difference in generalization capabilities of the two classes of methods above, we applied the kernel-based algorithm just described to a problem of learning from demonstration in a grid-like world (a more detailed description of the experimental setting is postponed to Section 5). Figure 2 compares the performance of the algorithm against those of a random policy and a policy obtained by solving the underlying MDP with a random reward. Note that, even using a random reward, the MDP-based solution is able to attain about a policy with 57% of correct actions, clearly establishing the advantage of using the MDP structure.

3.3 Bisimulation and MDP Metrics

Let us start by considering again the MDP depicted in Fig. 1. In this MDP, there is a close relation between states 1 and 2 since their actions and corresponding transitions are similar. In such a scenario, information about the policy, say, in state 1 will typically be also useful in determining the policy in state 2.

This notion of "similarity" has recently been explored in the MDP literature as a means to render solution methods for MDPs more efficient [7,21,27]. In fact, by identifying "similar" states in an MDP $\mathcal{M}$, it may be possible to construct a smaller MDP $\mathcal{M}'$ that can more easily be solved. In this paper we instead use MDP metrics to identify "similar" states and safely generalize the policy observed in the demonstration.

As established in [9], "similarity" between MDP states is best captured by the notion of *bisimulation*. Bisimulation is an equivalence relation $\sim$ on $\mathcal{X}$ in which two states x and y are similar if $r(x) = r(y)$ and

$$\mathbb{P}\left[X_{t+1} \in U \mid X_t = x, A_t = a\right] = \mathbb{P}\left[X_{t+1} \in U \mid X_t = y, A_t = a\right],$$

where U is some set in the partition induced by $\sim$. Lax bisimulation is a generalization of bisimulation that also accounts for action relabeling. Both bisimulation and lax bisimulation led to the development of several *MDP metrics* in which, if the distance between two states x, y is zero, then $x \sim y$ [7,27]. In this paper we adopt one such MDP metric, introduced in [27] and henceforth denoted as $\delta_{d_{\text{MDP}}}$, that is built over an initial metric on $\mathcal{X}$ that we refer as the *ground distance* (see Appendix A for details). We point out, however, that this choice is not unique.

While MDP metrics such as the one above were designed to improve efficiency in MDP solution methods, in this paper we are interested in their use in the problem of learning a policy from demonstration. In this context, MDP metrics arise as a natural way to "embed" the MDP structure in a supervised learning algorithm to improve its generalization performance while avoiding solving multiple MDPs. As will soon become apparent from our results, the use of an MDP metric indeed provides a significant and consistent improvement in performance over other metrics that ignore the MDP structure.

4 A Kernel-Based Approach Using MDP Metrics

We now introduce the main contributions of the paper, namely how to use MDP metrics such as the one discussed above to the particular problem considered in this paper.

The first aspect to consider is that, when learning a policy from demonstration, there is no reward information. While most MDP metrics naturally include a component that is reward-dependent, the particular setting considered here implies that the metric used should not include one such term. Secondly, the metric $\delta_{d_{\text{MDP}}}$ used already implicitly provides the learning algorithm with the necessary information on action relabeling. Therefore, in our algorithm we use the kernel

$$\mathbf{k}\big((x,a),(y,b)\big) = \exp\big(-\delta_{d_{\mathrm{MDP}}}((x,a),(y,b))/\sigma\big),$$

where σ denotes the kernel bandwidth, and (4) and (5) become

$$\hat{n}_a(x^*) = \sum_{i,b} \mathbf{k}\big((x^*,a),(x_i,b)\big) n_b(x_i) + \alpha_a, \tag{6}$$

$$\hat{\pi}(x^*,a) = \frac{\hat{n}_a(x^*)}{\sum_b \hat{n}_b(x^*)}. \tag{7}$$

The complete algorithm is summarized in Algorithm 1.

Algorithm 1. MDP-induced Kernel Classification.

```
Require: State-action space metric $\delta_{d_{\mathrm{MDP}}}$;
Require: Dataset $\mathcal{D} = \{(x_i, a_i), i = 1, \ldots, N\}$;
1: Given a query point $x^*$,
2: for all $a \in \mathcal{A}$ do
3:     Compute $\hat{n}_a(x^*)$ using (6)
4: end for
5: for all $a \in \mathcal{A}$ do
6:     Compute $\hat{\pi}(x^*, a)$ using (7)
7: end for
8: return $\hat{\pi}$;
```

4.1 Active Learning from Demonstration

Algorithm 1 is a Bayesian inference method that computes the posterior distribution over the parameters $\mathbf{p}(x)$ of the multinomial distribution (see Section 3.2). It is possible to use this posterior distribution in a simple active sampling strategy that can lead to a reduction in the sample complexity of our method [24].

One possibility is to compute the variance associated with the posterior distribution over the parameters $\mathbf{p}(x)$ at each state $x \in \mathcal{X}$, choosing as the next sample the state for which this variance is largest. The intuition behind this idea is that states with larger variance were observed less frequently and/or are distant (in the sense of our MDP metric) from other more often sampled states.

Using this active learning strategy, we can reduce the number of samples required for learning, since more informative states will be chosen first. Note also that this active learning approach implicitly takes into account the similarity between the states, in a sense treating similar states as one "aggregated" state and effectively requiring less samples.

5 Results

In this section we discuss several aspects concerning the performance of the proposed approach. Our results feature demonstrations with significantly less samples than the size of the state-space of the underlying MDP, allowing a clear evaluation of the generalization ability of the tested methods. Also, the scenarios used are typically not too large, allowing for easier interpretation of the results.

Our first test aims at verifying the general applicability of our approach. To this purpose, we applied the algorithm in Section 3.2 using the MDP metric described in Section 4 to 50 randomly generated MDPs. The state-space of the MDPs varies between 20 and 60 states, and the action space between 4 and 10 actions. From each state, it is possible to transition to between 20% and 40% of the other states. For each MDP, we randomly sample (without replacement) half of the states in the MDP and provide the learning algorithm with a demonstration consisting of these states and the corresponding optimal actions.

We compare the classification accuracy of the algorithm in Section 3.2 when using the MDP metric against the performance of that same algorithm when using other metrics. In particular, we used the *zero-one metric*, in which each the distance between any two state-action pairs (x, a) and (y, b) is either zero – if $(x, a) = (y, b)$ – or 1, and the *transition distance*, in which the distance between any two state-action pairs (x, a) and (y, b) corresponds to the minimum number of steps that an agent would take to move from state x to state y when taking as a first action the action a. As for the MDP metric, we used the distance $\delta_{d_{\mathrm{MDP}}}$ obtained by using the transition distance just described as ground distance. The bandwidth of the kernel was adjusted experimentally in each case to optimize the performance. The correct classification rate for the different approaches is summarized in Table 1.

Table 1. Average performance over 50 random worlds with between 20 and 50 states and 4 and 10 actions

	Correct Class. Rate (%)
Zero-one Distance	57.95 ± 4.33
Transition Distance	60.67 ± 5.10
$\delta_{d_{\mathrm{MDP}}}$	73.92 ± 6.98

As is clear from the results above, our approach clearly outperforms all other approaches, computing the correct action on 73% of the total number of states. This establishes that, as expected, the use of the MDP metric indeed provides an important boost in the performance of the method.

Our second test aims at comparing the performance of our approach against that of IRL-based algorithm (namely, the algorithm in Section 3.1). We run this comparison in two dimensions. We compare how both methods behave as the size of the demonstration increases and as the noise in the demonstration varies. For this comparison, we used a scenario consisting of eight clusters weakly connected

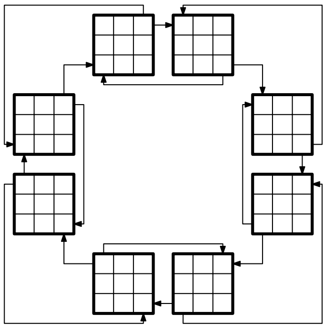


Fig. 3. Cluster-world scenario used in the tests

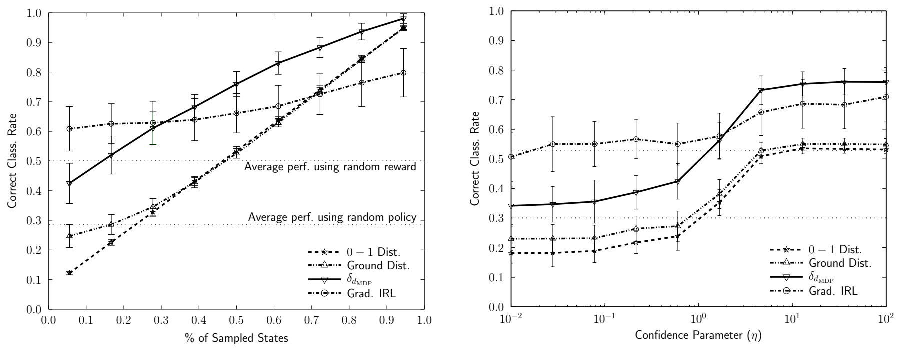
with one another. Each cluster contains a total of 9 states in a total of 72 states (see Fig. 3). Unlike more standard grid-world scenarios, the weakly connected clusters of states imply that there is a great asymmetry between the states in terms of transitions (unlike what occurs in standard grid-world domains). For this MDP, we considered 5 actions, corresponding to motions in the 4 directions, and the "NoOp" action.

We provided the algorithms with demonstrations of increasing size and evaluated the percentage of correct actions. As before, the demonstration was obtained by sampling (without replacement) a percentage of the total number of states and, for these, providing the corresponding optimal actions according to the target policy. Figure 4(a) compares our approach with the gradient-based IRL approach (GIRL). We also depict the performance of the supervised learning method when using other metrics and the two baselines discussed in Section 3.

The results show that all methods improve as the size of the demonstration increases. For very small demonstrations, GIRL gives better results as its baseline is higher. With a dataset larger than about 30% of the total number of states, our method clearly outperforms all other up to a "full demonstration", when all kernel-based methods perform alike. We also note that the worse performance of GIRL even in large demonstrations is probably due to the existence of local minima.[4]

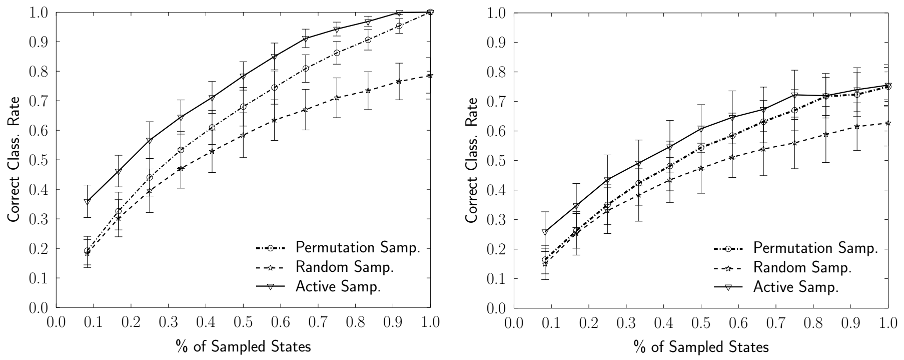
Figure 4(b) compares the performance of all methods as the noise in the demonstration varies. As in the first experiment, we used a demonstration of half the total number of states in the MDP. The actions were sampled from the distribution in (3) and the noise was adjusted by changing the value of the confidence parameter η. Low values of η correspond to noisy policies (where many suboptimal actions are sampled) while high values of η correspond to near-optimal policies. As expected all methods behave worse as the noise level increases, approaching the baseline performance levels for high levels of noise. As the noise decreases, the performance approaches the levels observed in Fig. 4(a).

[4] The results shown contain only runs that did have an improvement from the initial condition.



(a) (b)

Fig. 4. Classification rate of the different methods for (a) different demonstration sizes; (b) different confidence levels on the demonstration, with a demonstration size of alf the number of states. The horizontal lines are the baseline references (see main text). The results depicted are averaged over 50 independent Monte-Carlo trials, with the error bars corresponding to sample standard deviation.



(a) Noise-free demonstration. (b) Noisy demonstration.

Fig. 5. Classification rate of the different exploration methods with the size of the demonstration (as a ration of the number of states). The results depicted are averaged over 50 independent Monte-Carlo trials, with the error bars corresponding to sample standard deviation.

We note, however, that the performance of the kernel-based methods suffers a significant improvement at a certain noise threshold (for $\eta \approx 1$). We also note that larger demonstrations allowing states to be sampled with replacement will lead all methods to better filter out the noise and improve the classification rate. Finally, Fig. 5 shows the results of the active learning approach described in Section 4.1. We compare several possible sampling techniques: *random sampling*, where the states to be demonstrated are sampled uniformly with replacement; permutation sampling, where the states are sampled uniformly without replacement; and *active sampling*, where the states are sampled according to the variance of the posterior distribution over the parameters.

Our results, although specific to the particular scenario considered, illustrates the fact that our active sampling strategy manages to outperform other sampling techniques in terms of sample efficiency. The obtained gain depends on the particular problem, where problems with more symmetries/more clustered states are expected to yield larger gains for the active learning approach.

6 Concluding Remarks

Our results illustrate how the use of the MDP structure indeed provides valuable information when tackling the problem of learning a policy from demonstration. This can be verified easily by comparing, for example, the two baselines in Fig. 2. In these results, using a random policy lead to a correct classification rate of around 30%. However, if we instead use a *random reward function* and solve the corresponding MDP, the correct classification rate goes up to 57%. This clearly indicates that, in this particular example, the MDP structure significantly restricts the set of possible policies. Although these numbers correspond to a rather small environment in which the set of policies is naturally small, some preliminary experiments indicate that this conclusion holds in general (although with different numbers, of course). This conclusion is also supported by the discussion in Section 3.1 about the results in [17].

A second remark is concerned with the computational complexity associated with the particular MDP metric used in our results. As already pointed out by [7], MDP metrics that rely on the Kantorovich metric to evaluate the distance between two distributions $\mathsf{P}(x, a, \cdot)$ and $\mathsf{P}(y, b, \cdot)$ – such as the one used here – are computationally demanding. There are other alternative metrics that can be used (such as the total variation distance [7]) that are significantly more computationally efficient and can thus alleviate this issue. Nevertheless, it is worth pointing out that the MDP metric needs only to be computed once and can then be applied to any demonstration. This point is particularly important when envisioning, for example, robotic applications, since the MDP metric can be computed offline, hard-coded into the robot and used to learn different tasks by demonstration to be able to adapt to different users.

Another important point to make is that there are other methods that share a similar principle to the gradient-based IRL method in Section 3.1 and used for comparison in Section 5. These methods, while not explicitly concerned in recovering a reward description of the demonstrated task, use nevertheless an underlying MDP structure within a supervised learning or optimization setting [2, 20, 26, 25, 29]. Unfortunately, these approaches are close to IRL-based methods in that they still suffer from the need to solve multiple MDPs (see also the discussion in [16] for a more detailed account of the similarities and differences between the methods above). It is also worth noting that most aforementioned methods are designed to run with significantly larger datasets than those used in the experiments [16]. In conclusion they share the same advantages and disadvantages of the GIRL method in Section 3.2.

Finally, we conclude by noting that there is no immediate (theoretical) difficulty in extending the ideas in this paper to continuous scenarios. We did not discuss this extension in the paper since MDP metrics in continuous MDPs require a significantly more evolved machinery to describe [8] that would unnecessarily complicate the presentation. On the otherhand, although some continuous MDP metrics have been identified and shown to have good theoretical properties [8], they are expensive to compute and no computationally efficiently alternatives are known. One important avenue for future research is precisely the identifying continuous MDP metrics that are efficiently computable. It would also be interesting to explore, within the setting of our work, the impact of several recent works that proposed classification-based MDP solution methods [5,6,10,11,12].

Acknowledgements

The authors acknowledge the useful suggestions by Andreas Wichert and the anonymous reviewers.

References

1. Abbeel, P.: Apprenticeship learning and reinforcement learning with application to robotic control. Ph.D. thesis, Dep. Computer Science, Stanford Univ (2008)
2. Abbeel, P., Ng, A.: Apprenticeship learning via inverse reinforcement learning. In: Proc. 21st Int. Conf. Machine Learning, pp. 1–8 (2004)
3. Argall, B., Chernova, S., Veloso, M.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)
4. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. J. Artificial Intelligence Research 34, 1–25 (2009)
5. Fern, A., Yoon, S., Givan, R.: Approximate policy iteration with a policy language bias. In: Adv. Neural Information Proc. Systems 16 (2003)
6. Fern, A., Yoon, S., Givan, R.: Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. J. Artificial Intelligence Research 25, 75–118 (2006)
7. Ferns, N., Panangaden, P., Precup, D.: Metrics for finite Markov decision processes. In: Proc. 20th Conf. Uncertainty in Artificial Intelligence, pp. 162–169 (2004)
8. Ferns, N., Panangaden, P., Precup, D.: Metrics for Markov decision processes with infinite state-spaces. In: Proc. 21st Conf. Uncertainty in Artificial Intelligence, pp. 201–208 (2005)
9. Givan, R., Dean, T., Greig, M.: Equivalence notions and model minimization in Markov Decision Processes. Artificial Intelligence 147, 163–223 (2003)
10. Lagoudakis, M., Parr, R.: Reinforcement learning as classification: Leveraging modern classifiers. In: Proc. 20th Int. Conf. Machine Learning, pp. D424–D431 (2003)
11. Langford, J., Zadrozny, B.: Relating reinforcement learning performance to classification performance. In: Proc. 22nd Int. Conf. Machine Learning, pp. D473–D480 (2005)
12. Lazaric, A., Ghavamzadeh, M., Munos, R.: Analysis of a classification-based policy iteration algorithm. In: Proc. 27th Int. Conf. Machine Learning (to appear, 2010)

13. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction levels for robotic imitation: Overview and computational approaches. In: From Motor Learning to Interaction Learning in Robots, pp. 313–355 (2010)
14. Montesano, L., Lopes, M.: Learning grasping affordances from local visual descriptors. In: Proc. 8th Int. Conf. Development and Learning, pp. 1–6 (2009)
15. Neu, G., Szepesvári, C.: Apprenticeship learning using inverse reinforcement learning and gradient methods. In: Proc. 23rd Conf. Uncertainty in Artificial Intelligence, pp. 295–302 (2007)
16. Neu, G., Szepesvári, C.: Training parsers by inverse reinforcement learning. Machine Learning (2009) (accepted)
17. Ng, A., Russel, S.: Algorithms for inverse reinforcement learning. In: Proc. 17th Int. Conf. Machine Learning, pp. 663–670 (2000)
18. Pomerleau, D.: Efficient training of artificial neural networks for autonomous navigation. Neural Computation 3(1), 88–97 (1991)
19. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. In: Proc. 20th Int. Joint Conf. Artificial Intelligence, pp. 2586–2591 (2007)
20. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum margin planning. In: Proc. 23rd Int. Conf. Machine Learning, pp. 729–736 (2006)
21. Ravindran, B., Barto, A.: Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes. In: Proc. 5th Int. Conf. Knowledge-Based Computer Systems (2004)
22. Saunders, J., Nehaniv, C., Dautenhahn, K.: Teaching robots by moulding behavior and scaffolding the environment. In: Proc. 1st Annual Conf. Human-Robot Interaction (2006)
23. Schölkopf, B., Smola, A.: Learning with kernels: Support vector machines, regularization, optimization and beyond. MIT Press, Cambridge (2002)
24. Settles, B.: Active learning literature survey. Tech. Rep. CS Tech. Rep. 1648, Univ. Wisconsin-Maddison (2009)
25. Syed, U., Schapire, R.: A game-theoretic approach to apprenticeship learning. In: Adv. Neural Information Proc. Systems, vol. 20, pp. 1449–1456 (2008)
26. Syed, U., Schapire, R., Bowling, M.: Apprenticeship learning using linear programming. In: Proc. 25th Int. Conf. Machine Learning, pp. 1032–1039 (2008)
27. Taylor, J., Precup, D., Panangaden, P.: Bounding performance loss in approximate MDP homomorphisms. In: Adv. Neural Information Proc. Systems, pp. 1649–1656 (2008)S
28. Zhu, J., Hastie, T.: Kernel logistic regression and the import vector machine. In: Adv. Neural Information Proc. Systems. pp. 1081–1088 (2002)
29. Ziebart, B., Maas, A., Bagnell, J., Dey, A.: Maximum entropy inverse reinforcement learning. In: Proc. 23rd AAAI Conf. Artificial Intelligence, pp. 1433–1438 (2008)

A Description of a Lax-Bisimulation Metric

Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be an MDP, where the state-space $\mathcal{X}$ is endowed with a distance d. In other words, $(\mathcal{X}, d)$ is a metric space, and we henceforth refer to d as the *ground distance*. The function d can be any reasonable distance function on $\mathcal{X}$, *e.g.,* the Manhattan distance in the environment of Fig. 1.

Given one such distance d and any two distributions p_1 and p_2 over $\mathcal{X}$, the *Kantorovich distance* (also known as the earth mover's distance) between p_1 and p_2 is defined as the value of the linear program

$$\begin{aligned} \max_{\theta_x} \quad & \sum_x \big(p_1(x) - p_2(x)\big)\theta_x \\ \text{s.t.} \quad & \theta_x - \theta_y \leq d(x,y), && \text{for all } x, y \in \mathcal{X} \\ & 0 \leq \theta_x \leq 1 && \text{for all } x \in \mathcal{X} \end{aligned}$$

and denoted as $\mathsf{K}_d(p_1, p_2)$. Now given any two state-action pairs (x, a) and (y, b), we write

$$\delta_d\big((x,a),(y,b)\big) = k_1|r(x) - r(y)| + k_2\mathsf{K}_d\big(\mathsf{P}(x,a,\cdot), \mathsf{P}(y,b,\cdot)\big) \tag{8}$$

where k_1 and k_2 are non-negative constants such that $k_1 + k_2 \leq 1$. The function δ_d is, in a sense, a "one-step distance" between (x, a) and (y, b). It measures how different (x, a) and (y, b) are in terms of immediate reward/next transition. In order to measure differences in terms of long-term behavior, we denote by $\mathsf{H}_d(U, V)$ the Hausdorff distance (associated with d) between two sets U and V,[5] and define the MDP metric d_{MDP} as the fixed point of the operator $\mathbf{F}$ given by

$$\mathbf{F}(d)(x,y) = \mathsf{H}_{\delta_d}(\{x\} \times \mathcal{A}, \{y\} \times \mathcal{A}). \tag{9}$$

As shown in [27], the metric d_{MDP} can be obtained by iterating $\mathbf{F}$ and whenever $d(x, y) = 0$ then x and y are lax-bisimulation equivalent. Also, using the metric d_{MDP} it is possible to relabel the actions at each state to match those of "nearby" states.

We conclude by noting that, as discussed in Section 4, in our results we use $\delta_{d_{\text{MDP}}}$ with $k_1 = 0$ on the right-hand side of (8).

[5] Given a metric space $(\mathcal{X}, d)$, the Hausdorff distance between two sets $U, V \subset \mathcal{X}$ is given by

$$\mathsf{H}_d(U,V) = \max\left\{\sup_{x\in U}\inf_{y\in V} d(x,y), \sup_{y\in V}\inf_{x\in U} d(x,y)\right\}.$$