# Maximum Entropy Semi-Supervised Inverse Reinforcement Learning

**Julien Audiffren**
CMLA UMR 8536
ENS Cachan

**Michal Valko**
SequeL team
INRIA Lille

**Alessandro Lazaric**
SequeL team
INRIA Lille

**Mohammad Ghavamzadeh**
Adobe Research &
INRIA Lille

## Abstract

A popular approach to apprenticeship learning (AL) is to formulate it as an inverse reinforcement learning (IRL) problem. The MaxEnt-IRL algorithm successfully integrates the maximum entropy principle into IRL and unlike its predecessors, it resolves the ambiguity arising from the fact that a possibly large number of policies could match the expert's behavior. In this paper, we study an AL setting in which in addition to the expert's trajectories, a number of unsupervised trajectories is available. We introduce MESSI, a novel algorithm that combines MaxEnt-IRL with principles coming from semi-supervised learning. In particular, MESSI integrates the unsupervised data into the MaxEnt-IRL framework using a pairwise penalty on trajectories. Empirical results in a highway driving and grid-world problems indicate that MESSI is able to take advantage of the unsupervised trajectories and improve the performance of MaxEnt-IRL.

## 1 Introduction

The most common approach to solve a sequential decision-making problem is to formulate it as a Markov decision process (MDP) and then use dynamic programming or reinforcement learning to compute a near-optimal policy. This process requires the definition of a reward function such that the policy obtained by optimizing the resulting MDP produces the desired behavior. However, in many applications such as driving or playing tennis, it is required to take into account different desirable factors and it might be difficult to define an explicit reward function that accurately specifies the trade-off. It is often easier and more natural to learn how to perform such tasks by imitating an expert's demonstration. The task of learning from an expert is called *apprenticeship learning* (AL). An effective and relatively novel approach to apprenticeship learning is to formulate it as an *inverse reinforcement learning* (IRL) problem [Ng and Russell, 2000]. The basic idea is to assume that the expert is optimizing an MDP whose reward function is unknown and to derive an algorithm for learning the policy demonstrated by the expert. This approach has been shown to be effective in learning non-trivial tasks such as inverted helicopter flight control [Ng *et al.*, 2004], ball-in-a-cup [Boularias *et al.*, 2011], and driving on a highway [Abbeel and Ng, 2004; Levine *et al.*, 2011]. In the IRL approach to AL, we assume that several trajectories generated by an expert are available and the *unknown* reward function optimized by the expert can be specified as a linear combination of a set of state features. For each trajectory, we define its *feature count* as the sum of the values of each feature across the states traversed by the trajectory. The *expected feature count* is computed as the average of the feature counts of all available expert trajectories, thus implicitly encoding the behavior and preference of the expert. The goal is to find policies whose expected feature counts match that of the expert.[1] The early method of Abbeel and Ng [2004] finds such policies (reward functions) using an iterative max-margin algorithm. Its major disadvantage is that the problem is ill-defined since a large number of policies can possibly satisfy such matching condition. In this paper, we build on the work of Ziebart et al. [2008] who proposed a method based on the maximum entropy (MaxEnt) principle to resolve this ambiguity.

In many applications, in addition to the expert's trajectories, we may have access to a large number of trajectories that are not necessarily performed by an expert. For example, in learning to drive, we may ask an expert driver to demonstrate a few trajectories and use them in an AL algorithm to mimic her behavior. At the same time, we may record trajectories from many other drivers for which we cannot assess their quality (unless we ask an expert driver to evaluate them) and that may or may not demonstrate an expert-level behavior. We will refer to them as *unsupervised trajectories* and to the task of learning with them as *semi-supervised apprenticeship learning* [Valko *et al.*, 2012]. However, unlike in classification, we do not regard the unsupervised trajectories as being a mixture of expert and non-expert classes. This is because the unsupervised trajectories might have been generated by the expert themselves, by another expert(s), by near-expert agents, by agents maximizing different reward functions, or simply they can be some noisy data. The objective of IRL is to find the reward function maximized by the expert, and thus, semi-supervised apprenticeship learning is not a di-

[1]More precisely, the goal is to find reward functions such that the expected feature count of the (near)-optimal policies of the resulting MDPs match that of the expert.

rect application of semi-supervised classification.[2] Similar to many IRL algorithms that draw inspiration from supervised learning, in this paper, we build on semi-supervised learning (SSL, Chapelle et al. 2006) tools to derive a novel version of MaxEnt-IRL that takes advantage of unsupervised trajectories to improve the performance. In our first attempt, we had proposed [Valko *et al.*, 2012] a semi-supervised apprenticeship learning paradigm, called SSIRL, that combines the IRL approach of Abbeel and Ng [2004] with semi-supervised SVMs, and showed how it could take advantage of the unsupervised data and perform better and more efficiently (with less number of iterations, and thus, solving less MDPs) than the original algorithm. Nonetheless, as we will discuss in Sect. 3, SSIRL still suffers from the ill-defined nature of this class of IRL methods and the fact that all the policies (reward functions) generated over the iterations are labeled as "bad". This creates more problems in the SSL case than in the original setting. In fact, SSIRL relies on a cluster assumption that assumes the good and bad trajectories are well-separated in some feature space, but this never holds as new policies are generated and labeled as "bad". Finally, SSIRL is rather a proof of concept than an actual algorithm, as it does not have a stopping criterion, and over iterations the performance always reverts back to the performance of the basic algorithm by Abbeel and Ng [2004]. In Sec. 3, we address these problems by combining MaxEnt-IRL of Ziebart et al. [2008] with SSL. We show how unsupervised trajectories can be integrated into the MaxEnt-IRL framework in a way such that the resulting algorithm, MESSI (MaxEnt Semi-Supervised IRL), performs better than MaxEnt-IRL. In Sec. 4, we empirically show this improvement in the highway driving problem of Syed et al. [2008]. Additional experiments are available in the appendix.

## 2 Background and Notations

A Markov decision process (MDP) is a tuple $\langle S, A, r, p\rangle$, where $S$ is the state space, $A$ is the action space, $r: S \to \mathbb{R}$ is the state-reward function, and $p: S \times A \to \Delta(S)$ is the dynamics, such that $p(s'|s,a)$ is the probability of reaching state $s'$ by taking action $a$ in state $s$. We also denote by $\pi: S \to \Delta(A)$ a (stochastic) policy, mapping states to distributions over actions. We assume that the reward function can be expressed as a linear combination of state-features. Formally, let $\mathbf{f}: S \to \mathbb{R}^d$ be a mapping from the state space to a $d$-dimensional feature space and let $\mathbf{f}^i$ denote the $i$-th component of the vector $\mathbf{f}$, then we assume that the reward function is fully characterized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ such that for any $s \in S$, $r_{\boldsymbol{\theta}}(s) = \boldsymbol{\theta}^{\mathsf{T}}\mathbf{f}(s)$. A trajectory $\zeta = (s_1, a_1, s_2, \ldots, a_{T-1}, s_T)$ is a sequence of states and actions,[3] whose cumulative reward is

$$\bar{r}_\theta(\zeta) = \sum_{t=1}^{T} r_\theta(s_t) = \sum_{t=1}^{T} \boldsymbol{\theta}^{\mathsf{T}}\mathbf{f}(s_t) = \boldsymbol{\theta}^{\mathsf{T}} \sum_{t=1}^{T} \mathbf{f}(s_t) = \boldsymbol{\theta}^{\mathsf{T}}\mathbf{f}_\zeta, \quad (1)$$

where $\mathbf{f}_\zeta$ is the *feature count* of $\zeta$ that measures the cumulative *relevance* of each component $\mathbf{f}^i$ of the feature vector along the states traversed by $\zeta$.[4] In AL, we assume that only $l$ trajectories $\Sigma^* = \{\zeta_i^*\}_{i=1}^l$ are demonstrated by the expert and we define the corresponding *expected* feature count as $\mathbf{f}^* = \frac{1}{l}\sum_{i=1}^{l} \mathbf{f}_{\zeta_i^*}$. The expected feature count $\mathbf{f}^*$ summarizes the behavior of the expert since it defines which features are often "visited" by the expert, thus implicitly revealing her preferences, even without an explicit definition of the reward. This evidence is at the basis of the expected feature-count matching idea (see e.g., Abbeel and Ng 2004), which defines the objective of AL as finding a policy $\pi$ whose corresponding distribution $P_\pi$ over trajectories is such that the expected feature count of $\pi$ matches the expert's, i.e.,

$$\sum_{\zeta} P_\pi(\zeta)\mathbf{f}_\zeta = \mathbf{f}^*, \quad (2)$$

where the summation is taken over all possible trajectories. Ziebart et al. [2008] pointed out there is a large number of policies that satisfy the matching condition in Eq. 2 and introduced a method to resolve this ambiguity and to discriminate between different policies $\pi$ based on the actual reward they accumulate, so that the higher the reward the higher the probability. Building on the maximum entropy principle, Ziebart et al. [2008] move from the space of policies to the space of trajectories and define the probability of a trajectory $\zeta$ in an MDP characterized by a reward function defined by $\boldsymbol{\theta}$ as

$$P(\zeta|\boldsymbol{\theta}) \approx \frac{\exp(\boldsymbol{\theta}^{\mathsf{T}} f_\zeta)}{Z(\boldsymbol{\theta})} \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t),$$

where $Z(\boldsymbol{\theta})$ is a normalization constant and the approximation comes from the assumption that the transition randomness has a limited effect on behavior. The policy $\pi$ induced by the reward $\boldsymbol{\theta}$ is

$$\pi(a|s;\boldsymbol{\theta}) = \sum_{\zeta \in \Sigma_{s,a}} P(\zeta|\boldsymbol{\theta}), \quad (3)$$

where $\Sigma_{s,a}$ denotes the set of trajectories for which action $a$ is taken in state $s$. MaxEnt-IRL searches for the reward vector $\boldsymbol{\theta}$ that maximizes the log-likelihood of $\boldsymbol{\theta}$ of a given set of expert trajectories $\Sigma^* = \{\zeta_i^*\}_{i=1}^l$, i.e.,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\Sigma^*) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{l} \log P(\zeta_i^*|\boldsymbol{\theta}). \quad (4)$$

The resulting reward is such that the corresponding distribution over trajectories, $\sum_\zeta P(\zeta|\boldsymbol{\theta}^*)\mathbf{f}_\zeta$, matches the expected feature count $\mathbf{f}^*$, and at the same time, it strongly penalizes trajectories (and implicitly policies) that do not achieve as much reward as the expert.

## 3 Maximum Entropy Semi-Supervised Inverse Reinforcement Learning

The main objective of *semi-supervised learning* (SSL, Chapelle et al. 2006) is to bias the learning toward models that assign similar outputs to *intrinsically* similar data. Sim-

[2]If that was the case, then the AL problem would reduce to a classical supervised learning problem upon revealing the expert/non-expert labels of the additional trajectories, which is not the case.

[3]In practice, MaxEnt-IRL and our extension do not require a trajectory to include actions and only states are needed.

[4]Note that if $\mathbf{f}: S \to \{0,1\}$, then $\mathbf{f}_\zeta^i$ reduces to a counter of the times $\zeta$ traverses states activating the $i$th feature.

ilarity is commonly formalized in SSL as *manifold*, *cluster*, or other *smoothness* assumptions. Previously, we had integrated [Valko *et al.*, 2012] a semi-supervised regularizer into the SVM-like structure of the feature matching algorithm of Abbeel and Ng [2004] under an implicit clustering assumption. Although the integration is technically sound, the SSL hypothesis of the existence of clusters grouping trajectories with different performance is not robust w.r.t. the learning algorithm. In fact, even when assuming that expert and non-expert policies (used to generate part of the unsupervised trajectories) are actually clustered, newly generated policies are always labeled as "negative", and as they become more and more similar to the expert's policy, they will eventually cross the clusters and invalidate the SSL hypothesis. Let us stress that MaxEnt-IRL, which we build on, is also not a classical supervised method (e.g., classification) as it does not take as input a set of *labeled* trajectories (e.g., positive/negative). Unlike imitation learning, that can be directly framed as a classification problem, IRL algorithms rely on supervised methods to solve a problem which is *not supervised* in its initial formulation. In our setting, there is no clear definition of "class" of trajectories that could be exploited by MaxEnt-IRL. Even if we knew that a unsupervised trajectory is not from the expert, we would not know how to explicitly use it in MaxEnt-IRL.

In this section, we propose an SSL solution to AL with unsupervised trajectories. To avoid the problems of the prior work, we do not reduce the problem to SSL classification. We modify maximum entropy framework to obtain an optimization problem that reflects the structural assumptions of the geometry of the data. In our context, a similar, yet different approach is applied to MaxEnt-IRL that leads to constraining the probabilities of the trajectories to be *locally consistent*.

## Pairwise Penalty and Similarity Function

We assume that the learner is provided with a set of expert trajectories $\Sigma^* = \{\zeta_i^*\}_{i=1}^l$ and a set of unsupervised trajectories $\widetilde{\Sigma} = \{\zeta_j\}_{j=1}^u$. We also assume that a function $s$ is provided to measure the similarity $s(\zeta, \zeta')$ between any pair of trajectories $(\zeta, \zeta')$. We define the *pairwise penalty* $R$ as

$$R(\boldsymbol{\theta}|\Sigma) = \frac{1}{2(l+u)} \sum_{\zeta,\zeta' \in \Sigma} s(\zeta, \zeta') \left(\boldsymbol{\theta}^\mathsf{T}(\mathbf{f}_\zeta - \mathbf{f}_{\zeta'})\right)^2, \quad (5)$$

where $\Sigma = \Sigma^* \cup \widetilde{\Sigma}$, and $\mathbf{f}_\zeta$ and $\mathbf{f}_{\zeta'}$ are the feature counts for trajectories $\zeta, \zeta' \in \Sigma$, and

$$(\boldsymbol{\theta}^\mathsf{T}(\mathbf{f}_\zeta - \mathbf{f}_{\zeta'}))^2 = (\bar{r}_{\boldsymbol{\theta}}(\zeta) - \bar{r}_{\boldsymbol{\theta}}(\zeta'))^2,$$

is the difference in rewards accumulated by the two trajectories w.r.t. the reward vector $\boldsymbol{\theta}$. The purpose of the pairwise penalty is to penalize reward vectors $\boldsymbol{\theta}$ that assign very different rewards to similar trajectories (as measured by $s(\zeta, \zeta')$). In other words, the pairwise penalty acts as a regularizer which favors vectors that give similar rewards to similar trajectories. On the other hand, when two trajectories are very different, $s(\zeta, \zeta')$ is very small, and $\boldsymbol{\theta}$ is not constrained to give them a similar reward.

In SSL, there are several ways to integrate a smoothness penalty $R(\boldsymbol{\theta})$ into the learning objective. In our case, we take inspiration from Erkan and Altun [2009] for the use of the pairwise penalty in MaxEnt, but then we add it to the dual problem rather than to the primal, as adding the penalty to the dual would not be tractable. This corresponds to adding the penalty $R(\boldsymbol{\theta})$ to the MaxEnt-IRL optimization of Eq. 4 as

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \left(L(\boldsymbol{\theta}|\Sigma^*) - \lambda R(\boldsymbol{\theta}|\Sigma)\right), \quad (6)$$

where $\lambda$ is a parameter trading off between the log-likelihood of $\boldsymbol{\theta}$ w.r.t. the expert's trajectories and the coherence with the similarity between the trajectories in $\widetilde{\Sigma}$ and $\Sigma^*$. This modification is motivated by the reduction of the number of parameters: indeed, following the derivation in Erkan and Altun [2009] would lead to a different algorithm with $l^2$ additional parameters, hence greatly increasing the computational complexity of the problem. As it is often the case in SSL, the choice of the similarity is critical to the success of learning, since it encodes a prior on the characteristics two trajectories should share when they have similar performance, and it defines how unsupervised data together with the expert data are used in regularizing the final reward. Whenever enough prior knowledge about the problem is available, it is possible to hand-craft specific similarity functions that effectively capture the critical characteristics of the trajectories. On the other hand, when the knowledge about the problem is rather limited, it is preferable to use generic similarity functions that can be employed in a wide range of domains. A popular choice of a generic similarity is the radial basis function (RBF) $s(\zeta, \zeta') = \exp(-\|\mathbf{f}_\zeta - \mathbf{f}_{\zeta'}\|^2/2\sigma)$, where $\sigma$ is the bandwidth. Although hand-crafted similarity functions usually perform better, we show in Sect. 4 that even the simple RBF is an effective similarity for the feature counts. More generally, the effectiveness of the regularizer $R(\boldsymbol{\theta}|\Sigma)$ is not only related to the quality of the similarity function but also to the unsupervised trajectories. To justify this claim, let $P_u$ be a probability distribution over the set $\overline{\Sigma}$ of all the feasible trajectories (i.e., trajectories compatible with the MDP dynamics); we assume that the unsupervised trajectories are drawn from $P_u$. Note that the generative model of the unsupervised trajectories could be interpreted as the result of a mixture of behaviors obtained from different reward functions. If $P(\theta)$ is a distribution over reward functions (e.g., a set of users with different preferences or skills), then $P_u$ is defined as $P_u(\zeta) = P(\zeta|\theta)P(\theta)$. When $s$ only provides local similarity (e.g., RBF), it is indeed the distribution $P_u$ that defines the way the similarity is propagated across trajectories, since $s$ is only applied to trajectories in $\Sigma$ drawn from $P_u$. In order to have an intuition of the effect of the unsupervised trajectories through the similarity function, let us consider an (uninteresting) case when $P_u$ is uniform over $\overline{\Sigma}$ and RBFs are used with a small bandwidth $\sigma$. In this case, Eq. 6 would basically reduce to a regularized version of MaxEnt-IRL with a $L_2$-regularization on the parameter $\boldsymbol{\theta}$ that, as a result, is forced to be *uniformly* smooth over all trajectories in $\overline{\Sigma}$. However, in the typical case, we expect $P_u$ to be non-uniform, which enforces the regularization towards $\boldsymbol{\theta}$'s that give similar rewards only to similar trajectories among those that are more likely to be present. As a result, we expect that the more $P(\cdot|\boldsymbol{\theta}^*)$ (i.e., the trajectory distribution induced by the reward maximized by the expert) is supported[5] in $P_u$, the more effective

[5]We use *support* as in measure theory, so that trajectories that are

**Algorithm 1** MESSI - MaxEnt SSIRL

```
Input: $l$ expert trajectories $\Sigma^* = \{\zeta_i^*\}_{i=1}^l$, $u$ unsupervised trajectories $\widetilde{\Sigma} = \{\zeta_j\}_{j=1}^u$, similarity function $s$, number of iterations $T$, constraint $\theta_{\max}$, regularizer $\lambda_0$
Initialization:
Compute $\{\mathbf{f}_{\zeta_i^*}\}_{i=1}^l$, $\{\mathbf{f}_{\zeta_j}\}_{j=1}^u$ and $\mathbf{f}^* = 1/l \sum_{i=1}^l \mathbf{f}_{\zeta_i^*}$
Generate a random reward vector $\boldsymbol{\theta}_0$
for $t = 1$ to $T$ do
    Compute policy $\pi_{t-1}$ from $\boldsymbol{\theta}_{t-1}$ (backward pass Eq. 3)
    Compute counts $\mathbf{f}_{t-1}$ of $\pi_{t-1}$ (forward pass Eq. 8)
    Update the reward vector as in Eq. 7
    If $\|\boldsymbol{\theta}_t\|_\infty > \theta_{\max}$, project back by $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t \frac{\theta_{\max}}{\|\boldsymbol{\theta}_t\|_\infty}$
end for
```

the penalty $R(\boldsymbol{\theta}|\Sigma)$. Since the regularization depends on $\boldsymbol{\theta}$, in this case, MESSI forces the similarity only among the trajectories that are likely to perform well. Finally, similar to standard SSL, if $P_u$ is rather adversarial, i.e., if the support of $P(\zeta|\theta^*)$ is not supported in $P_u$, then penalizing according to the unsupervised trajectories may even worsen the performance. In the experiments reported in the next section, we will define $P_u$ as a mixture of $P(\cdot|\theta^*)$ (used to generated the expert trajectories) and trajectories generated by other distributions, and show that the unfavorable case has only a limited effect on the MESSI's performance.

## Implementation of the Algorithm

Alg. 1 shows the pseudo-code of MESSI: *M*ax*E*nt *S*emi-*S*upervised *I*RL. MESSI solves the optimization problem of Eq. 6 by gradient descent. At the beginning, we first compute the empirical feature counts of all the expert's $\{\mathbf{f}_{\zeta_i^*}\}$ and unsupervised trajectories $\{\mathbf{f}_{\zeta_j}\}$, and randomly initialize the reward vector $\boldsymbol{\theta}_0$. At each iteration $t$, given the reward $\boldsymbol{\theta}_t$, we compute the corresponding expected feature count $\mathbf{f}_t$ and obtain $\boldsymbol{\theta}_{t+1}$ by the following update rule:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + (\mathbf{f}^* - \mathbf{f}_t)\\ &+ \frac{\lambda}{\theta_{\max}(l+u)} \sum_{\zeta,\zeta'\in\Sigma} s(\zeta,\zeta')\,(\boldsymbol{\theta}_t^\mathsf{T}(\mathbf{f}_\zeta - \mathbf{f}_{\zeta'})^2),\end{aligned} \tag{7}$$

where $\mathbf{f}^*$ is the average feature count of the expert trajectories in $\Sigma^*$ and $\theta_{\max}$ is a parameter discussed later. A critical step in MESSI is the computation of the expected feature count $\mathbf{f}_t$ corresponding to the given reward vector $\boldsymbol{\theta}_t$. In fact, $\mathbf{f}_t$ is defined as $\sum_\zeta P(\zeta|\boldsymbol{\theta}_t)\mathbf{f}_\zeta$ and it requires the computation of the "posterior" distribution $P(\zeta|\boldsymbol{\theta}_t)$ over all the possible trajectories. This becomes rapidly infeasible since the number of possible trajectories grows exponentially with the number of states and actions in the MDP. Thus, we follow the same approach illustrated in Ziebart et al. [2008]. We note that the expected feature count can be written as

$$\mathbf{f}_t = \sum_\zeta P(\zeta|\boldsymbol{\theta}_t)\mathbf{f}_\zeta = \sum_{s\in S} \rho_t(s)\mathbf{f}(s), \tag{8}$$

where $\rho_t(\cdot)$ is the expected visitation frequency of the states $s \in S$ obtained from following the policy induced by $\boldsymbol{\theta}$. As a result we first need to compute the policy $\pi_t$ as in Eq. 3. This can be done using a value-iteration-like algorithm as in the *backward pass* of Alg. 1 in Ziebart et al. [2008].[6] Once the stochastic policy $\pi_t$ is computed, starting from a given initial distribution over $S$, we recursively apply $\pi_t$ and using the transition model $p$ compute the expected visitation frequency $\rho$ (*forward pass*). Although the backward pass step of MaxEnt-IRL allows the estimation of $\pi_t$, it requires computing the exponential of the reward associated to the feature count of any given path. This may introduce numerical issues that may prevent the algorithm from behaving well. Thus, we introduce two modifications to the structure of the MaxEnt-IRL algorithm. Since these issues exist in the original MaxEnt-IRL and are not caused by the SSL penalty, the following two modifications could be integrated into the original MaxEnt-IRL as well. We first normalize all the features so that for any $s$, $\mathbf{f}(s) \in [0,1]^d$. We then multiply them by $(1-\gamma)$ and move to the discounted feature count. In other words, given a trajectory $\zeta = (s_1, \ldots, s_T)$, if the features $\mathbf{f}$ are multiplied by $(1-\gamma)$, we have that the discounted feature count of $\zeta$, $\mathbf{f}_\zeta = \sum_{t=0}^\infty \gamma^t \mathbf{f}(s_t)$, is bounded in $[0,1]$, i.e., $\|\mathbf{f}\|_\infty \leq 1$. As a result, all the expected feature counts involved in the optimization problem will be bounded. Although normalizing features already provides more stability to the backward pass, the cumulative rewards $\bar{r}_{\boldsymbol{\theta}}(\zeta)$ still depend on $\boldsymbol{\theta}$, which would often tend to be unbounded. In fact, if the expert trajectories often visit feature $\mathbf{f}^i$ while completely avoiding feature $\mathbf{f}^j$, then we want apprenticeship learning to find a reward vector $\boldsymbol{\theta}$ that favors policies that reach $\mathbf{f}^i$ much more often than $\mathbf{f}^j$. However, from Eq. 3 we note that the probability of an action $a$ in a state $s$ is obtained by evaluating the *exponential* rewards accumulated by different trajectories that take action $a$ in state $s$. So, in order to obtain a policy $\pi(\cdot|\cdot;\theta)$ that has a strong tendency of reaching $\mathbf{f}^i$ and avoiding $\mathbf{f}^j$, the reward vector $\boldsymbol{\theta}$ should be very positive for $\boldsymbol{\theta}^i$ and very negative for $\boldsymbol{\theta}^j$. This may lead to very extreme reward vectors such that $\boldsymbol{\theta}_t^i \to \infty$ and $\boldsymbol{\theta}_t^j \to -\infty$ as the algorithm proceeds, which would rapidly lead to numerical instability. In order to prevent this effect, we introduce a constraint $\theta_{\max}$ such that $||\boldsymbol{\theta}_t||_\infty \leq \theta_{\max}$. From an algorithmic point of view, the introduction of the constraint $\theta_{\max}$ requires ensuring that at the end of each iteration of the gradient descent, the $\boldsymbol{\theta}$ is always projected back into the set of $||\boldsymbol{\theta}||_\infty \leq \theta_{\max}$. Finally, it is sensible to use the constraint to tune the range of the regularizer $\lambda$ in advance, which should be set to $\lambda = \lambda_0/\theta_{\max}$. The constraint $\theta_{\max}$ can also be seen as the minimum entropy (or uncertainty) we would like to observe in the resulting policy. In fact, a very small $\theta_{\max}$ forces $\pi(\cdot|s;\theta)$ to have a very large entropy, where most of the actions have similar probability. Therefore, the parameter $\theta_{\max}$ could be used to encode how much expert trajectories, that are realizations of the expert's policy, should be trusted, and how much residual *uncertainty* should be preserved in the final policy $\pi(\cdot|s;\theta_T)$.

likely to be drawn from $P(\cdot|\boldsymbol{\theta}^*)$ are present in $P_u$.

[6]This step is the main computational bottleneck of MESSI. It is directly inherited from MaxEnt-IRL and is common to most IRL methods. However, using the transition samples in $\Sigma$ we may apply an approximate dynamic programming algorithm, like fitted value iteration, to reduce the computational complexity and remove the need for knowing the exact dynamics of the MDP. One of such approaches was used by Klein et al. [2012].

## 4 Experimental Results

We report the empirical results of MESSI on standard testbeds in IRL and investigate to what extent unsupervised trajectories are exploited by MESSI to learn better policies. Additional experiments on a grid world problem and a pit problem are available in the appendix.

In order to have a sensible comparison between MESSI and MaxEnt-IRL we first need to define how unsupervised trajectories are actually generated (i.e., the distribution $P_u$). We consider three cases, where the unsupervised trajectories obtained from near-expert policies, random policies, and policies optimizing different reward functions. Let $\boldsymbol{\theta}^*$ denote the reward associated with the expert and $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ denote two other rewards. We define $P_{u^*} = P(\cdot|\boldsymbol{\theta}^*)$, $P_1 = P(\cdot|\boldsymbol{\theta}_1)$, and $P_2 = P(\cdot|\boldsymbol{\theta}_2)$, and draw unsupervised trajectories from $P_{\mu_1} = \nu P_{u^*} + (1-\nu)P_1$, $P_{\mu_2} = \nu P_{u^*} + (1-\nu)P_2$, and $P_{\mu_3} = \nu P_1 + (1-\nu)P_2$, where $\nu \in [0,1]$ is a parameter. We also consider MESSIMAX, equivalent to MESSI when the unsupervised trajectories are obtained from $P_{u^*}$. This is a favorable scenario used as an upper-bound to the best attainable performance of MESSI. Because of the conceptual and practical issues discussed in Sec. 3, we do not compare with SSIRL, but rather compare to MaxEnt-IRL when all trajectories (i.e., $\Sigma^* \cup \widetilde{\Sigma}$) are used and to a semi-supervised baseline inspired by the EM algorithm (see e.g., Sect. 2.3 in Zhu 2005) that we created to illustrate that viewing our problem as semi-supervised classification has conceptual flaws (as argued in Sect. 3). The EM-MaxEnt starts from an arbitrary reward $\theta^0$ and at each round $K$ performs two steps:

1. *Expectation step*: given $\boldsymbol{\theta}^{K-1}$, we compute the probabilities $P(\zeta|\boldsymbol{\theta}^{K-1})$ for each $\zeta \in \Sigma$.
2. *Maximization step*: we solve a modified version of Eq. 4 where trajectories are weighted by their probabilities,

$$\boldsymbol{\theta}^K = \arg\max_{\boldsymbol{\theta}} \sum_{\zeta \in \Sigma} P(\zeta|\boldsymbol{\theta}^{K-1}) \log P(\zeta_i^*|\boldsymbol{\theta}). \qquad (9)$$

In practice, at each round, Eq. 9 is solved using a gradient descent as in Alg. 1. Therefore, we introduce an additional parameter $\eta$ which determines the number of gradient steps per-round in the maximization step. The resulting algorithm is referred to as $\eta$-EM-MaxEnt.

**Parameters.** For each of the experiments, the default parameters are $\theta_{\max} = 500$, $\lambda_0 = 0.05$, the number of iterations of gradient descent is set to $T = 100$, one expert trajectory is provided ($l = 1$), and the number of unsupervised trajectories is set to $u = 20$ with $\nu = 0.5$. In order to study the impact of different parameters on the algorithms, we first report results where we fix all the above parameters except one and analyze how the performance changes for different values of the selected parameter. In particular, in Fig. 1, while keeping all the other parameters constant, we compare the performance of MaxEnt-IRL, different setups of MESSI, and MESSIMAX by varying different dimensions: **1)** the number of iterations of the algorithms, **2)** the number of unsupervised trajectories, **3)** the distribution $P_{\mu_1}$ by varying $\nu$, and **4)** the value of parameter $\lambda$. We then report a comparison with the MaxEnt-IRL with all trajectories and EM-MaxEnt algorithm in Fig. 2.

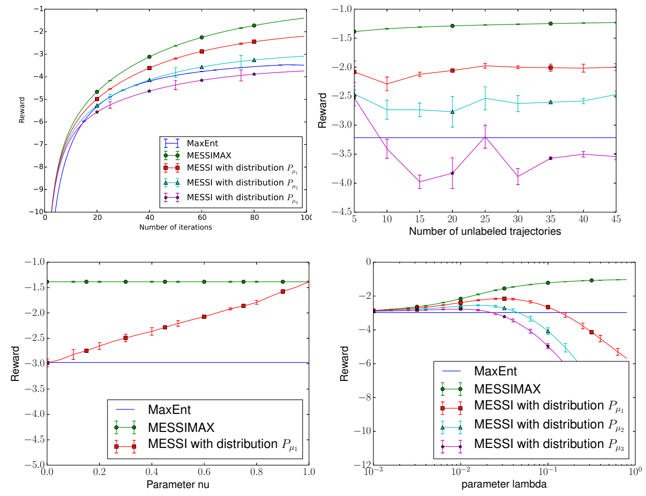

Figure 1: Results as a function of (from left to right): number of iterations of the algorithms, number of unsupervised trajectories, parameter $\nu$, and parameter $\lambda$.

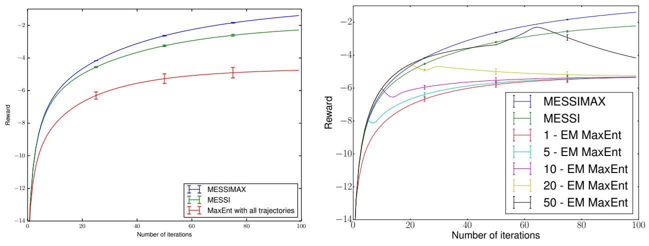

Figure 2: Comparison of MESSI with MaxEnt-IRL using all trajectories (*left*) and $\eta$-EM-MaxEnt with $\eta$ =1, 5, 10, 20, 50 (*right*).

**The highway problem.** We study a variation of the car driving problem in Syed [2008]. The task is to navigate a car in a busy 4-lane highway. The 3 available actions are to move *left*, *right*, and *stay* on the same lane. We use 4 features: *collision* (contact with another car), *off-road* (being outside of the 4-lane), *left* (being on the two left-most lanes of the road), and *right* (being on the two right-most lanes of the road). The expert trajectory avoids any collision or driving off-road while staying on the left-hand side. The true reward $\boldsymbol{\theta}^*$ heavily penalizes any collision or driving off-road but it does not constrain the expert to stay on the left-hand side of the road. $\boldsymbol{\theta}_1$ penalizes any collision or driving off-road but with weaker weight, and finally $\boldsymbol{\theta}_2$ does not penalize collisions. We use the RBF kernel with $\sigma = 5$ as the similarity function and evaluate the performance of a policy as *minus* the expected number of collisions and off-roads.

**Results.** In the following, we analyze the results obtained by averaging 50 runs along multiple dimensions.

*Number of iterations.* As described in Sec. 2, MaxEnt-IRL solves the ambiguity of expected feature count matching by encouraging reward vectors that strongly discriminate from expert's behavior and other policies. However, while in MaxEnt-IRL, all trajectories that differ from the expert's are (somehow) equally penalized, MESSI aims to provide similar rewards to the (unsupervised) trajectories that resemble (according to the similarity function) the expert's. As a result, MESSI is able to find rewards that better encode the unknown expert's preferences when provided with relevant unsupervised data. This intuitively explains the improvement of both MESSI and MESSIMAX w.r.t. MaxEnt-IRL. In particular, the benefit of MESSIMAX, which only uses trajectories drawn from a distribution coherent with the expert's distribution, becomes apparent after just a few dozens of iterations. On the other hand, the version of MESSI using trajectories drawn from $P_{\mu_3}$, which is very different from $P_{u^*}$, performs worse than MaxEnt-IRL. As discussed in Sec. 3, in this case, the regularizer resulting from the SSL approach is likely to bias the optimization away from the correct reward. Likewise, the versions of MESSI using trajectories drawn from $P_{\mu_1}$ and $P_{\mu_2}$ lose part of the benefit of having a bias toward the correct reward, but still perform better than MaxEnt-IRL. Another interesting element is the fact that for all MaxEnt-based algorithms the reward $\boldsymbol{\theta}_t$ tends to increase with the number of iterations (as described in Sec. 3). As a result, after a certain number of iterations, the reward reaches the constraint $\|\boldsymbol{\theta}\|_\infty = \theta_{\max}$, and when this happens, additional iterations no longer bring a significant improvement. In the case of MaxEnt-IRL, this may also cause overfitting and degrading performance.

*Number of unsupervised trajectories.* The performance tends to improve with the number of unsupervised trajectories. This is natural as the algorithm obtains more information about the features traversed by (unsupervised) trajectories that resemble the expert's. This allows MESSI to better generalize the expert's behavior over the entire state space. However in some experiments, the improvement saturates after a certain number of unsupervised trajectories are added. This effect is reversed when $P_{\mu_3}$ is used since unsupervised trajectories do not provide any information about the expert and the more the trajectories, the worse the performance.

*Proportion of good unsupervised trajectories.* MESSI may perform worse than MaxEnt-IRL whenever provided with a completely non-relevant distribution (e.g., $P_{\mu_3}$). This is expected since unsupervised trajectories together with the similarity function and the pairwise penalty introduce an inductive bias that forces similar trajectories to have similar rewards. However, our results show that this decrease in performance is not dramatic and is indeed often negligible as soon as the distribution of the unsupervised trajectories supports the expert distribution $P_{u^*}$ enough. In fact, MESSI provided with trajectories from $P_{\mu_1}$ and $P_{\mu_2}$ starts performing better than MaxEnt-IRL when $\nu \geq 0.15$. Finally, note that for $\nu = 1$, MESSI and MESSIMAX are the same algorithm.

*Regularization* $\lambda$. The improvement of MESSI over MaxEnt-IRL depends on a proper choice of $\lambda$. This is a well-known trade-off in SSL: If $\lambda$ is too small, the regularization is negligible and the resulting policies do not perform better than the policies generated by MaxEnt-IRL. On the contrary, if $\lambda$ is too large, the pairwise penalty forces an extreme smoothing of the reward function and the resulting performance for both MESSI and MESSIMAX may decrease.

*Comparison to MaxEnt-IRL with all trajectories.* As expected, when unsupervised trajectories $\widetilde{\Sigma}$ are used directly in MaxEnt-IRL, its performance is negatively affected, even in the case of $P_{\mu_1}$ and $\nu = 0.5$ as shown in Fig. 2. This shows how MESSI is actually using the unsupervised trajectories in the proper way by exploiting them only in the regularization term rather than integrating them directly in the likelihood.

*Comparison to $\eta$-EM-MaxEnt.* In our setting, $\eta$-EM-MaxEnt-IRL is worse than MESSI in all but a few iterations, in which it actually achieves a better performance. The decrease in performance is due to the fact that the unsupervised trajectories are included in the training set even if they are unlikely to be generated by the expert. This is because the estimation of $\theta^*$ is not accurate enough to compute the actual probability of the trajectories and the error made in estimating these probabilities is amplified over time, leading to significantly worse results. Finally, one may wonder why not to interrupt $\eta$-EM-MaxEnt when it reaches its best performance. Unfortunately, this is not possible since the quality of the policy corresponding to $\theta_t$ cannot be directly evaluated, which is the same issue as in SSIRL. On the contrary, MESSI improves monotonically and we can safely interrupt it at any iteration.

Our experiments verify the SSL hypothesis that whenever unsupervised trajectories convey some useful information about the structure of the problem, MESSI performs better than MaxEnt-IRL: **1)** If the expert trajectory is suboptimal and the unsupervised data contain the information leading to a better solution: For example, in the highway experiment, the expert drives only on the left-hand side, while an optimal policy would also use the right-hand side when needed. **2)** In the case where the information given by the expert is incomplete and the unsupervised data provide the information about how to act in the rest of the state space: On the other hand, if none of the unsupervised trajectories contains useful information, MESSI performs slightly worse than MaxEnt-IRL. However, our empirical results indicate that the distribution of the trajectories $P_u$ only needs to partially support $P_{u^*}$ ($\nu \geq 0.15$) for MESSI to perform significantly better than MaxEnt-IRL.

## 5 Conclusion

We studied apprenticeship learning with access to unsupervised trajectories in addition to those generated by an expert. We presented a novel combination of MaxEnt-IRL with SSL by using a pairwise penalty on trajectories. Empirical results showed that MESSI takes advantage of unsupervised trajectories and can perform better and more efficiently than MaxEnt-IRL. This work opens a number of interesting directions for future research. While MaxEnt-IRL is limited to almost deterministic MDPs, the causal entropy approach [Ziebart *et al.*, 2013] allows to deal with the general case of stochastic MDPs. A natural extension of MESSI is to integrate the pairwise penalty into causal entropy. Other natural directions of research are experimenting with other similarity measures for trajectories (e.g., the Fisher kernel) and studying the setting in which we have access to more than one expert.

## Acknowledgments

This work was supported by the French Ministry of Higher Education and Research and the French National Research Agency (ANR) under project ExTra-Learn n.ANR-14-CE24-0010-01.

## References

[Abbeel and Ng, 2004] P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[Boularias *et al.*, 2011] A. Boularias, J. Kober, and J. Peters. Relative Entropy Inverse Reinforcement Learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 182–189, 2011.

[Chapelle *et al.*, 2006] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[Erkan and Altun, 2009] A. Erkan and Y. Altun. Semi-Supervised Learning via Generalized Maximum Entropy. In *Proceedings of JMLR Workshop*, pages 209–216. New York University, 2009.

[Klein *et al.*, 2012] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse Reinforcement Learning through Structured Classification. In *Advances in Neural Information Processing Systems 25*, pages 1016–1024. 2012.

[Levine *et al.*, 2011] S. Levine, Z. Popovic, and V. Koltun. Nonlinear Inverse Reinforcement Learning with Gaussian Processes. In *Advances in Neural Information Processing Systems 24*, pages 1–9, 2011.

[Ng and Russell, 2000] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670, 2000.

[Ng *et al.*, 2004] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Inverted Autonomous Helicopter Flight via Reinforcement Learning. In *International Symposium on Experimental Robotics*, 2004.

[Syed *et al.*, 2008] U. Syed, R. Schapire, and M. Bowling. Apprenticeship Learning Using Linear Programming. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1032–1039, 2008.

[Valko *et al.*, 2012] M. Valko, M. Ghavamzadeh, and A. Lazaric. Semi-Supervised Apprenticeship Learning. In *Proceedings of the 10th European Workshop on Reinforcement Learning*, volume 24, pages 131–241, 2012.

[Zhu, 2005] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[Ziebart *et al.*, 2008] B. Ziebart, A. Maas, A. Bagnell, and A. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

[Ziebart *et al.*, 2013] Brian D. Ziebart, J. Andrew (Drew) Bagnell, and Anind Dey. The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, February 2013.

## A The Grid-world Problem

In this set of experiments, we use a $16 \times 16$ square grid-world, a variation of the grid-world domain in (Abbeel and Ng, 2004). The agent has $4$ possible actions (*up*, *down*, *left*, *right*) with 70% chance of success and 30% chance of taking a random different action. The grid is divided into 64 macro-states of size $2 \times 2$ each, which define the feature space $\mathbf{f}$, so that each macro-state has its own characteristic feature that is active when a state within a macro-state is traversed.

A reward vector $\boldsymbol{\theta}^*$ is generated at random and is set as the *true* reward of the problem. This reward maps every feature to a strictly negative value, except for 3 randomly chosen features. $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are also generated at random.

In this problem, we use the RBF kernel $s(\zeta_i, \zeta_j) = \exp(-\|\mathbf{f}_{\zeta_i} - \mathbf{f}_{\zeta_j}\|/10)$ to define the similarity between two trajectories $\zeta_i$ and $\zeta_j$. This similarity function is completely general and does not exploit any prior knowledge about the problem. We evaluate the performance of a policy as its expected reward w.r.t. the true reward, i.e., $\boldsymbol{\theta}_{\text{true}}^{\top}\mathbf{f}_T$ .

Results are reported in Figs. 3, 4, 5, and 6. The result of the comparison with the $\eta$-EM-MaxEnt are reported in Fig. 7.

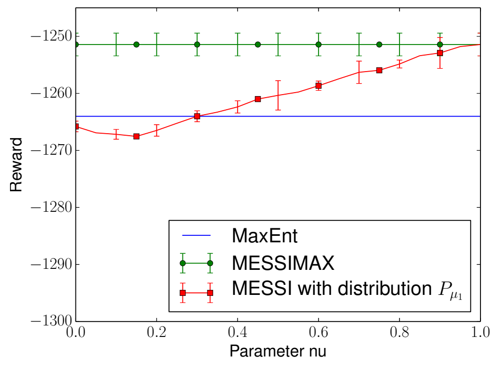

Figure 5: Results on the grid-world problem as a function of parameter $\nu$.

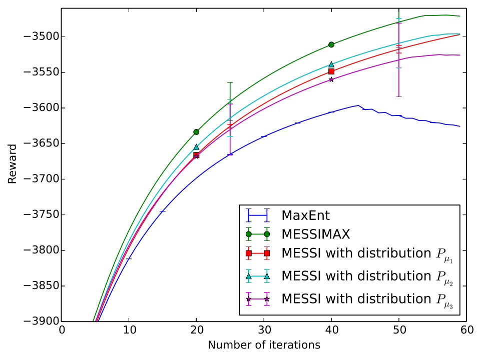

Figure 3: Results on the grid-world problem as a function of number of iterations of the algorithms.

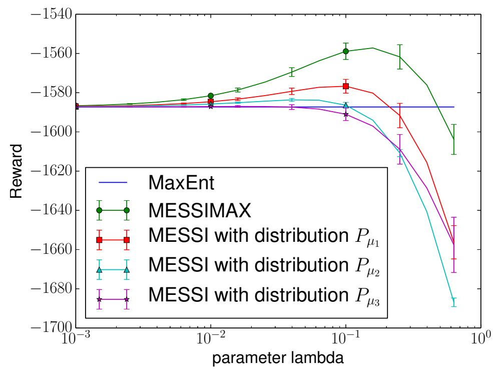

Figure 6: Results on the grid-world problem as a function of parameter $\lambda$.

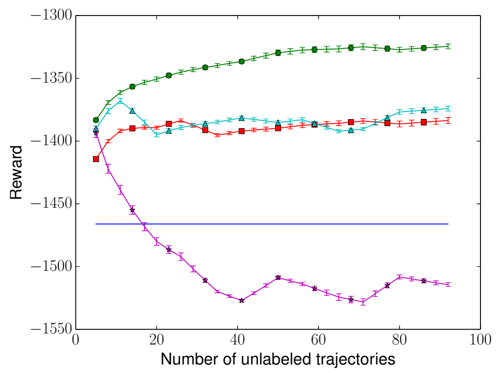

Figure 4: Results on the grid-world problem as a function of number of unsupervised trajectories.

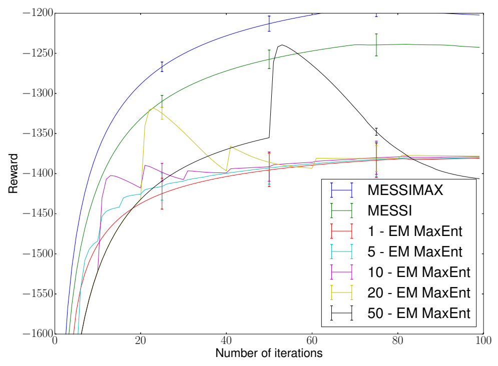

Figure 7: Comparison between the performance of MESSI and MESSIMAX with the iterative $\eta$-EM-MaxEnt with $\eta = 1, 5, 10, 20, 50$ on the grid-world problem.

## B The Pit Problem

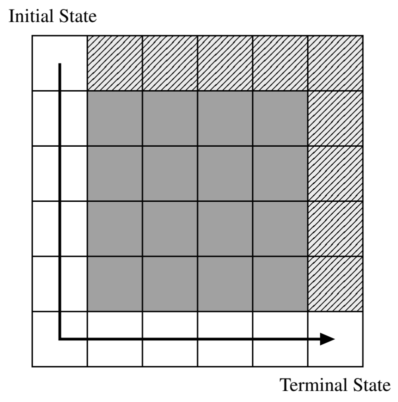

Figure 8: The pit domain.

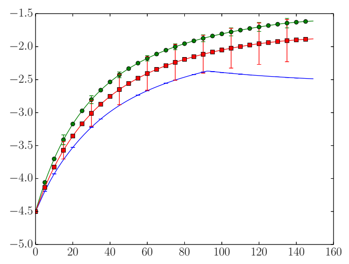
Figure 9: Results on the pit problem as a function of number of iterations of the algorithms.

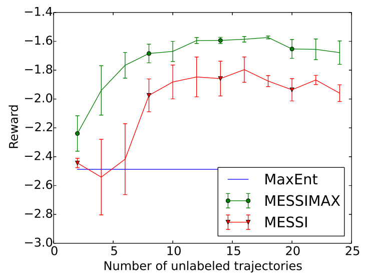

Figure 10: Results on the pit problem as a function of number of unsupervised trajectories.

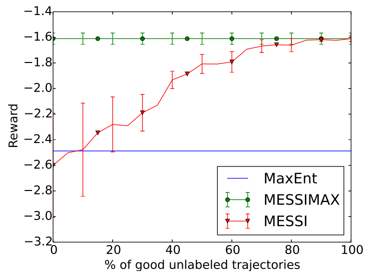

Figure 11: Results on the pit problem as a function of $\nu$.

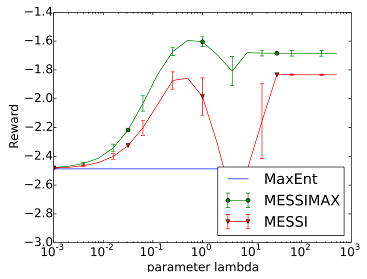

Figure 12: Results on the pit problem as a function of $\lambda$.

In this set of experiments, we use a $6 \times 6$ grid world representing an edge surrounding a pit (see Fig. 8). The agent has $4$ possible actions (*up*, *down*, *left*, *right*) with $85\%$ chance of success and $15\%$ chance of taking a random different action. The initial state is the square with coordinate $(1, 1)$ and the terminal state is the square with coordinate $(6, 6)$. The grid is divided in $3$ parts that correspond to the $3$ different features, the left edge (represented by the white squares in Fig. 8), the right edge (shaded squares), and the pit (the gray squares in the middle).

The objective of the expert is to move from the initial square to the terminal state by avoiding the pit in the middle. The (single) expert trajectory (the black arrow) provided to the learner goes around the pit counter-clockwise. In this case, the unsupervised trajectories are generating as a mixture of trajectories generated from a deterministic policy that goes around the pit either clockwise or counter-clockwise (basically $P_{u^*}$) and trajectories are generated from a policy that crosses the pit.

The similarity function used in this experiment is hand-crafted for this domain and is defined as $s(\zeta_j, \zeta_k) = \exp(-\|n_j - n_k\|)$, where $n_j$ denotes the number of change of direction in the trajectory $\zeta_j$ (for instance going left then down). This is an example of a hand-crafted similarity function that fits nicely to the problem, since a trajectory is good if and only if it goes around the pit, and thus, turns only once.

Since a good trajectory should carefully avoid the pit, for this experiment, we evaluate the performance of a policy by how frequently it crosses the pit, that corresponds to evaluating the expected feature count $\mathbf{f}_T$ (obtained after $T$ iterations of gradient descent) corresponding to the pit. We denote by $\mathbf{f}_T^{\text{pit}}$ this value, and report $-\mathbf{f}_T^{\text{pit}}$ as a measure of performance in the plots of Figs. 9, 10, 11, and 12. The results basically confirm the discussion in other experiments and show an even stronger advantage of MESSI w.r.t. MaxEnt-IRL, i.e., whenever prior knowledge about the problem is available and a very informative similarity function is chosen, MESSI is very effective in taking advantage of it and significantly outperforms MaxEnt-IRL.