JMLR: Workshop and Conference Proceedings 24:131–141, 2012 10th European Workshop on Reinforcement Learning

Semi-Supervised Apprenticeship Learning

Michal Valko MICHAL.VALKO@INRIA.FR
Mohammad Ghavamzadeh MOHAMMAD.GHAVAMZADEH@INRIA.FR
Alessandro Lazaric ALESSANDRO.LAZARIC@INRIA.FR
INRIA Lille - Nord Europe, team SequeL, France

Editor: Marc Peter Deisenroth, Csaba Szepesvári, Jan Peters

Abstract

In apprenticeship learning we aim to learn a good policy by observing the behavior of an expert or a set of experts. In particular, we consider the case where the expert acts so as to maximize an unknown reward function defined as a linear combination of a set of state features. In this paper, we consider the setting where we observe many sample trajectories (i.e., sequences of states) but only one or a few of them are *labeled* as experts' trajectories. We investigate the conditions under which the remaining *unlabeled* trajectories can help in learning a policy with a good performance. In particular, we define an extension to the max-margin inverse reinforcement learning proposed by Abbeel and Ng [2004] where, at each iteration, the max-margin optimization step is replaced by a semi-supervised optimization problem which favors classifiers separating clusters of trajectories. Finally, we report empirical results on two grid-world domains showing that the semi-supervised algorithm is able to output a better policy in fewer iterations than the related algorithm that does not take the unlabeled trajectories into account.

1. Introduction

In traditional reinforcement learning, agents are rewarded for achieving certain states and try to learn a policy that maximized the rewards cumulated over time. Constructing such a reward can be not only tedious but a reward could also be difficult to encode explicitly. The goal of apprenticeship learning [Ng and Russell, 2000] is to learn a good behavior by observing the behavior of an expert (usually a human). In fact, in many applications it is easier to collect observations of experts' behaviors rather than encoding a suitable reward function to obtain the desired behavior. The effectiveness of this approach to learn non-trivial behaviors has already been demonstrated on complex tasks such as inverted helicopter navigation [Ng et al., 2004], ball-in-a-cup game [Boularias et al., 2011], or learning highway driving behavior [Abbeel and Ng, 2004; Levine et al., 2011].

A typical scenario for apprenticeship learning is to have an expert performing a few optimal (or close to the optimal) trajectories and use these to learn a good policy. In this paper, we consider a different setting where we also have access to many other trajectories, for which we do not know whether they are actually generated by an expert or not. For instance, in learning a driving behavior we might have recorded many driving patterns but only a few has been thoroughly analyzed and labeled as *expert* trajectories. We refer to this setting as *semi-supervised apprenticeship learning*.

Since the beginning of apprenticeship learning [Ng and Russell, 2000], several different approaches have been proposed, including inverse reinforcement learning [Abbeel and Ng, 2004], max-margin planning [Ratliff et al., 2006], or maximum entropy inverse reinforcement learning [Ziebart et al., 2008]. In this paper, we address the semi-supervised setting by an extension of the (max-margin) inverse reinforcement learning proposed by Abbeel and Ng [2004]. We assume that the *"good"* and *"bad"* trajectories are well enough separated in some feature space defined over the state space. Therefore, a decision boundary that separates good and bad trajectories, should not cross the dense regions of the data. This distributional assumption is typical in semi-supervised learning [Zhu, 2008] and it is often referred to as *cluster assumption*. Such an assumption enables us to use semi-supervised support vector machines [Bennett and Demiriz, 1999] and to take advantage of the unlabeled trajectories when learning a policy. This assumption can be verified if some policies (and therefore trajectories) are more natural given the semantics of the problem.

The IRL method of Abbeel and Ng [2004] is an iterative algorithm, where in each step a new reward is generated. In each step we use a MDP solver to compute the optimal policy given that reward. The goal of IRL is to find a policy (or a mixture thereof) that would match the expected feature counts of the expert (i.e., the probability with which the agent achieves different states). The new reward is generated as a solution to a max-margin classification, where the expert's feature expectations are considered as positives and the feature expectation of the generated policies as negatives. In the method we propose, we aim to reduce the number of iterations of this procedure and therefore reduce the number of calls to the MDP solver. We do it by augmenting the feature expectations of the expert's and generated policies with the feature expectations of unlabeled trajectories. The aim is to improve the max-margin classification learning and discover better policies in fewer iterations. Furthermore, if the expert performer does not achieve the maximum reward already, the unlabeled trajectories may help us discover a better performing policy.

2. Background

Let MDP\R denote a finite state Markov decision process without a reward function: (S, A, T, γ, D), where S is a finite set of states, A is a set of actions, T is a set of transition probabilities, γ is a discount factor, and D is the initial state distribution from which a starting state s_0 is sampled. We assume that we are given a vector of state features $\phi : S \to [0,1]^k$. We also assume the existence of a *true* reward function $R^\star$ defined as a *linear* combination of the features ϕ, i.e., $R^\star(s) = \boldsymbol{w}^\star \cdot \phi(s)$, where $\|\boldsymbol{w}^\star\|_1 \leq 1$ is the true weight reward vector. The value function of a policy π is defined as the expected sum of discounted rewards, i.e., $V^\pi(s) = \mathbb{E}\big[\sum_t \gamma^t R(s_t)\big]$, thus the expected value of a policy π w.r.t. to the initial distribution D becomes

$$\mathbb{E}_{s_0 \sim D}[V^\pi(s_0)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right] = \boldsymbol{w} \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\right] = \boldsymbol{w} \cdot \boldsymbol{\mu}(\pi),$$

where $\boldsymbol{\mu}(\pi) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\big]$ is the vector of *feature expectations* given policy π. We also assume that we can access an MDP solver which, given a reward function, outputs a policy that maximizes it. The main input of an IRL method is a set of m expert trajectories. Let a trajectory $\tau_i = (s^{(i)}_{E,1}, \ldots, s^{(i)}_{E,T_i})$ be a sequence of T_i states generated by the expert

Algorithm 1: IRL: Inverse reinforcement learning [Abbeel and Ng, 2004]

Input: ε, expert trajectories $\{\tau_i\}_{i=1}^m$
Estimate $\hat{\boldsymbol{\mu}}_E \leftarrow \frac{1}{m}\sum_{i=1}^m \sum_{t=0}^{T_i} \gamma^t \phi(s_{E,t}^{(i)})$
Randomly pick $\pi^{(0)}$ and set $i = 1$
repeat
  $t^{(i)} \leftarrow \max_{\|\boldsymbol{w}\|_2 \leq 1} \min_{j<i} \ \boldsymbol{w} \cdot (\hat{\boldsymbol{\mu}}_E - \hat{\boldsymbol{\mu}}^{(j)})$
  Let $\boldsymbol{w}^{(i)}$ be the one attaining this maximum
  $\pi^{(i)} \leftarrow \text{MDPSolver}(R = \boldsymbol{w}^{(i)} \cdot \boldsymbol{\phi})$
  Estimate $\hat{\boldsymbol{\mu}}^{(i)} \leftarrow \boldsymbol{\mu}(\pi^{(i)})$
  $i \leftarrow i + 1$
until $t^{(i)} \leq \varepsilon$

policy π_E starting from state $s_{E,1}^{(i)}$. Consequently, the input of an IRL algorithm is a set $\{\tau_i\}_{i=1}^m$. Given the set of trajectories we can compute an empirical estimate of their feature expectations as

$$\widehat{\boldsymbol{\mu}}_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T_i}\gamma^t \phi(s_{E,t}^{(i)}).$$

The goal of IRL algorithm of Abbeel and Ng [2004] is to find a policy $\tilde{\pi}$, such that $\|\boldsymbol{\mu}(\tilde{\pi}) - \boldsymbol{\mu}_E\|_2 \leq \varepsilon$. In such case, for any $\|\boldsymbol{w}\|_1 \leq 1$:

$$\left|\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t)|\pi_E\right] - \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t)|\tilde{\pi}\right]\right| = |\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}(\tilde{\pi}) - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}_E| \leq \|\boldsymbol{w}\|_2 \, \|\boldsymbol{\mu}(\tilde{\pi}) - \boldsymbol{\mu}_E\|_2 = \varepsilon,$$

since $\|\boldsymbol{w}\|_2 \leq \|\boldsymbol{w}\|_1 \leq 1$. This implies that if we find a policy $\tilde{\pi}$ whose feature expectations are ε-close to the ones of the expert, then the performance of $\tilde{\pi}$ would also be ε-close to the performance of the expert, no matter the actual value of the true reward vector $\boldsymbol{w}^*$. Algorithm 1 of Abbeel and Ng [2004] is therefore trying to find such $\tilde{\pi}$. Notice that the algorithm terminates whenever the margin $t^{(i+1)}$ falls below a given threshold ε given as input to the algorithm. As a result,

$$\forall \boldsymbol{w}, \text{ where } \|\boldsymbol{w}\|_2 \leq 1, \quad \exists i \ \text{ s.t. } \ \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}^{(i)} \leq \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\mu}_E - \varepsilon. \tag{1}$$

We can then construct a *mixture* policy $\tilde{\pi}$, such that for any $\boldsymbol{w}$, the feature expectations $\boldsymbol{\mu}(\tilde{\pi})$ are at most ε away from the feature expectations $\boldsymbol{\mu}_E$ of the expert,

$$\boldsymbol{\mu}(\tilde{\pi}) = \min_{\boldsymbol{\mu}} \|\boldsymbol{\mu}_E - \boldsymbol{\mu}\|_2 \ \text{ s.t. } \ \boldsymbol{\mu} = \sum_i \lambda_i \boldsymbol{\mu}^{(i)} \ \text{ and } \ \sum_i \lambda_i = 1. \tag{2}$$

This corresponds to a mix of the policies computed during the execution of Algorithm 1 with the mixing weights $\boldsymbol{\lambda}$. The resulting mixture policy can be executed by sampling from $\{\pi^{(i)}\}_i$ at the beginning and then acting according to $\pi^{(i)}$. For such a mixture, it is guaranteed that $\|\boldsymbol{\mu}_E - \boldsymbol{\mu}(\tilde{\pi})\|_2 \leq \varepsilon$. However, having a resulting policy being a mixture of several policies instead of a single policy is often seen as a drawback of the IRL method [Ratliff et al., 2006; Ziebart et al., 2008].

3. Semi-supervised inverse reinforcement learning

The optimization step for $\boldsymbol{w}$ of Algorithm 1 described in Section 2 is equivalent to support vector machines (SVM) by Vapnik [1995], if we associate a label $+1$ with the expert's feature expectations $\boldsymbol{\mu}_E$, and a label -1 with the feature expectations of the policies computed so far: $\{\boldsymbol{\mu}(\pi^{(j)}) : j = 0 \dots (i-1)\}$. As Abbeel and Ng [2004] point out, the optimization can be solved by any quadratic program solver or a specialized SVM solver.

In the following, we describe how to improve the previous IRL algorithm when we have access to many other trajectories but without knowing whether they are generated by an expert or not. Specifically, unlabeled trajectories can help us to find $\boldsymbol{w}$ that separates the expert feature counts from the non-expert ones faster. We describe how to use semi-supervised SVMs for this purpose.

Besides the standard *hinge loss* $\mathcal{L}(f, \boldsymbol{x}, y) = \max\{1 - y\,|f(\boldsymbol{x})|\,, 0\}$ where $(\boldsymbol{x}, y)$ is a labeled sample and f is a generic classifier, semi-supervised support vector machines (SVMs) also uses the *hat loss* $\widehat{\mathcal{L}}(f, \boldsymbol{x}) = \max\{1 - |f(\boldsymbol{x})|\,, 0\}$ on unlabeled data [Bennett and Demiriz, 1999] and define the optimization problem

$$\hat{f} = \min_f \left[\sum_{i \in L} \mathcal{L}(f, \boldsymbol{x}_i, y_i) + \gamma_l \|f\|^2 + \gamma_u \sum_{i \in U} \widehat{\mathcal{L}}(f, \boldsymbol{x}_i) \right], \tag{3}$$

to compute the max-margin decision boundary $\hat{f}$ that avoids dense regions of data, where L and U are the sets of labeled and unlabeled samples respectively, γ_l is the cost penalty as in standard SVMs, and γ_u weighs the influence of unlabeled examples in U. No matter what the label of an unlabeled example would be, the hat loss $\widehat{\mathcal{L}}$ penalizes functions f with a small distance to unlabeled examples. The hat loss, however, makes the optimization problem (3) non-convex. As a result, it is hard to solve the problem optimally and most of the work in this field has focused on approximations. A comprehensive review of these methods was done by Zhu [2008]. In our experiments, we make use of the transductive SVMs by Joachims [1999a].

The objective of IRL is to find such mixture policy that would be close to the expert's one in the feature space. IRL builds this mixture iteratively with every new policy getting closer. In the early rounds however only few policies are available. This is where we can take advantage of the unlabeled trajectories. Assuming that the expert and non-expert policies are separated in the feature space, optimizing the objective (3) can result in $\boldsymbol{w}$ that is very close to $\boldsymbol{w}^\star$ and in turn can generate better performing policy sooner. In other words, our algorithm in the beginning looks for a policy that would be optimal for $\boldsymbol{w}$ suggested by the unlabeled data. As the more new policies are generated, they outweigh this bias and the final mixture policy would be close to the expert's in the feature space for any possible $\boldsymbol{w}$. Figure 3.1 illustrates how can unlabeled trajectories help the learning when the cluster assumption is satisfied. In the figure, the true reward is the vertical line, the feature expectations of the expert is the big red dot on the bottom left and the feature expectation of the generated trajectory (after the first iteration) is the big blue dot on the top right. The remaining dots correspond to the feature expectations of the unlabeled trajectories. Semi-supervised learning objective gets penalized when the learned reward (decision boundary) crosses the dense regions of the feature space and therefore can recover a reward close to the true one.

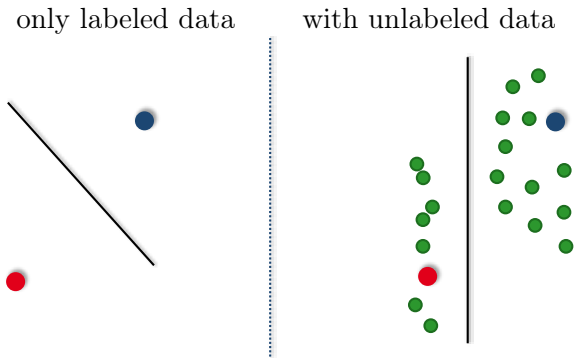

Figure 3.1: Cluster assumption of *semi-supervised learning*. **Left:** supervised learning. **Right:** semi-supervised learning (with unlabeled trajectories)

Specifically, the optimization problem (3) for linear functions f, where the experts feature expectation are labeled as $+1$ and all the feature expectations from the generated policies labeled as -1, simplifies to:

$$\min_{\boldsymbol{w}}\Big(\max\{1 - \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}}_E, 0\} + \gamma_l \left\|\boldsymbol{w}\right\|_2 + \sum_{j<i} \max\{1 + \boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}}^{(j)}, 0\} + \gamma_u \sum_{u\in U} \max\{1 - \left|\boldsymbol{w}^\mathsf{T}\hat{\boldsymbol{\mu}}_u\right|, 0\}\Big)$$

Algorithm 2 shows the pseudocode of our SSIRL algorithm. At the beginning, we compute the empirical estimates $\hat{\boldsymbol{\mu}}_u$ for each unlabeled trajectory (or a set of m_u trajectories, if several trajectories belong to the same performer). After the optimization step for $\boldsymbol{w}$, which corresponds to (3), we normalize $\boldsymbol{w}$ such that $\left\|\boldsymbol{w}\right\|_2 = 1$, in order to preserve the approximation properties of the IRL algorithm. Note that since both SSIRL and IRL algorithms iteratively create new policies, eventually the unlabeled trajectories will be outweighed and, in the limit, the performance of the generated mixture policies will be similar. However, with the informative set of unlabeled trajectories present during the learning, SSIRL is able to recover $\boldsymbol{w}^\star$ faster. Therefore, it may need less iterations than IRL and consequently smaller number of calls to the MDP solver.

After the execution of SSIRL algorithm, we need to perform the same optimization procedure (Equation 2) as for the IRL algorithm to get the mixing weights $\boldsymbol{\lambda}$. However, we would like to stress an important point about policy mixing with SSIRL: Even though the feature expectations from the unlabeled trajectories can speed up the learning we do not know the policies that generated them. This has two consequences. First, these feature expectations cannot be used for finding the final mixture after the algorithm has converged. Consequently, the final mixture policy still consists of policies related to labeled trajectories. Second, the distance between the unlabeled feature counts (i.e. the margin t) cannot be taken into account for the stopping criterion. Otherwise, it could for example happen that we have an unlabeled feature expectations which are ε close to $\boldsymbol{\mu}_E$. The algorithm would terminate immediately but no policy could be actually generated. Therefore, the benefit of SSIRL is that a better $\boldsymbol{w}$ is likely to be found earlier and consequently a better policy is *generated* for that $\boldsymbol{w}$.

Algorithm 2: SSIRL: Semi-supervised inverse reinforcement learning

Input: ε, γ_l, γ_u

expert trajectories $\{s_{E,t}^{(i)}\}$

unlabeled trajectories from U performers $\{s_{u,t}^{(i)}\}$

estimate $\hat{\boldsymbol{\mu}}_E \leftarrow \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T_i}\gamma_l^t\phi(s_{E,t}^{(i)})$

for $u = 1$ **to** U **do**

estimate $\hat{\boldsymbol{\mu}}_u \leftarrow \frac{1}{m_u}\sum_{i=1}^{m_u}\sum_{t=0}^{T_{u,i}}\gamma^t\phi(s_{u,t}^{(i)})$

end for

randomly pick $\pi^{(0)}$ and set $i \leftarrow 1$

repeat

$$\boldsymbol{w}^{(i)} \leftarrow \min_{\boldsymbol{w}}\Big(\max\{1-\boldsymbol{w}^\intercal\hat{\boldsymbol{\mu}}_E,0\}+\gamma_l\left\|\boldsymbol{w}\right\|_2 + \sum_{j<i}\max\{1+\boldsymbol{w}^\intercal\hat{\boldsymbol{\mu}}^{(j)},0\}+\gamma_u\sum_{u\in U}\max\{1-\left|\boldsymbol{w}^\intercal\hat{\boldsymbol{\mu}}_u\right|,0\}\Big)$$

$\boldsymbol{w}^{(i)} \leftarrow \boldsymbol{w}^{(i)}/\left\|\boldsymbol{w}^{(i)}\right\|_2$

$\pi^{(i)} \leftarrow \mathrm{MDP}(R=(\boldsymbol{w}^{(i)})^\intercal\boldsymbol{\phi})$

estimate $\hat{\boldsymbol{\mu}}^{(i)} \leftarrow \boldsymbol{\mu}(\pi^{(i)})$

$t^{(i)} \leftarrow \min_i \boldsymbol{w}^\intercal(\hat{\boldsymbol{\mu}}_E - \hat{\boldsymbol{\mu}}^{(i)})$

$i \leftarrow i+1$

until $t^{(i)} \le \varepsilon$

4. Experiments

The purpose of this section is to show that with unlabeled trajectories, SSIRL can find a good policy in less iterations than IRL, thus requiring fewer calls to the MDP solver. For the comparison we use the variations of the gridworld domain of Abbeel and Ng [2004].

4.1. Gridworld

In the first set of experiments, we use 64×64 gridworlds. The agent has 4 actions (*north*, *west*, *south*, *east*) with 70% chance of success and 30% chance of taking a random different action. The grid is divided into 64 macrocells of size 8×8. These macrocells define the features: for each macrocell i and state s there is a feature ϕ_i such that $\phi_i(s) = 1$ if the state s belongs to the macrocell i and 0 otherwise. The rewards are randomly generated such that the $\boldsymbol{w}^\star$ is sparse, which gives rise to interesting policies [Abbeel and Ng, 2004]. The discount factor γ is set to 0.99.

To simulate the clustering assumption in this domain, we sample the trajectories as follows. We sample $\boldsymbol{w}^\star$ for the expert and a different $\boldsymbol{w}'$ for the non-expert performer. The estimated feature expectations $\hat{\boldsymbol{\mu}}_E$ for the expert were computed by simulating the policy optimizing $\boldsymbol{w}^\star$. The first half of the unlabeled trajectories corresponds to different simulations of the same expert policy. The other half of trajectories was simulated from a policy optimizing $\boldsymbol{w}'$. Therefore, this setup also shows how SSIRL can address *sampling errors*, when the expert trajectory is a noisy sample from the optimal policy.

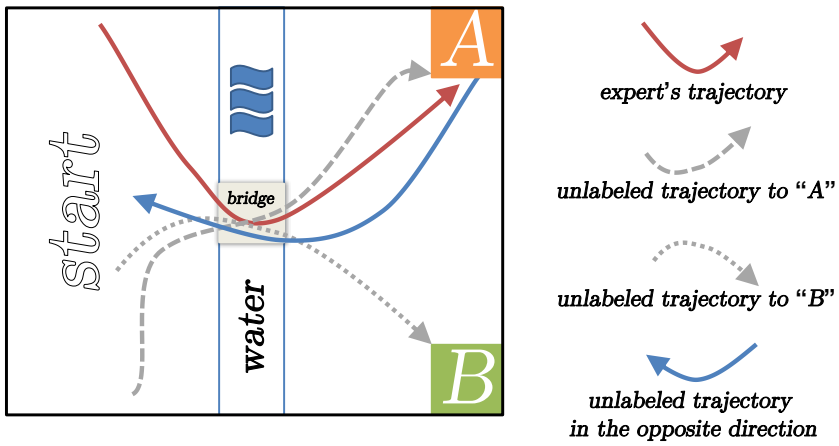

Figure 4.1: Two goals domain

4.2. Two goals navigation

The domain illustrated in Figure 4.1 defines a more natural scenario where clustered feature expectations arise in practice. It consists of a starting section on the left, two places of interest ("A" and "B") in the corners on the right, and a river with a bridge in the middle. The task is to navigate from the start to the one of the two places of interest while avoiding stepping into the river.

The state space, actions, and features are defined as in the previous domain of gridworlds. However, the true reward (unknown to a learner) is set to 0.01 for the feature corresponding to the place "A" and to -0.005 for the features corresponding to the river except for the bridge where it is set to zero (as well for the other regions). The expert demonstrates a trajectory from the start to the place "A". The first half of the unlabeled trajectories consists also from demonstrations navigating from the start to "A" which can be thought of as additional well performing demonstrations.

We perform two experiments with this domain in which we provide different set of trajectories as the other half of unlabeled trajectories. In the first one, the other half of the trajectories are demonstrations from the start to "B" instead. In the second experiment, the other half of demonstrations are trajectories in the reversed direction, i.e., from "A" to the start. Due to the design, one can expect that the both sets of trajectories form two clusters in the feature space of the macrocells. We choose these trajectories such that some features can be helpful (as passing through the bridge) and some not (as going to a different goal or in an opposite direction).

4.3. Implementation details

As an MDP solver, we use a standard policy iteration method. For the optimization of $\boldsymbol{w}$ both in IRL and SSIRL algorithm we use SVMlight implementation of Joachims [1999a],

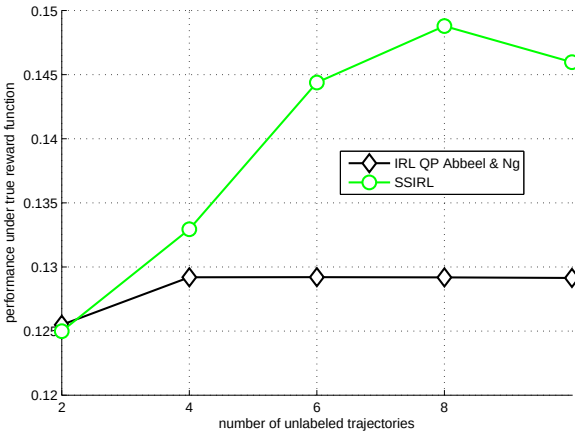

Figure 4.2: Performance of the mixture policies under the true reward after 10 iterations of both algorithms as a function of the number of extra unlabeled trajectories.

which is capable of both supervised and semi-supervised (transductive) SVM learning. Learning parameters γ_l, γ_u were set to the default values. To calculate the mixture coefficients $\boldsymbol{\lambda}$ of the mixture policy (Eq. 2) we use the quadratic program solver CVX by Grant et al. [2006].

4.4. Results

Figures 4.2 and 4.3 show the results of the *gridworld* experiments. Both the true reward function and the reward function for the non-experts are sampled randomly at the beginning of each trial. In Figure 4.2 we varied the number of unlabeled trajectories for our semi-supervised SSIRL method and compared it to IRL. Figure 4.2 shows the performance of the final mixture policy under the true reward (unknown to both algorithms) after 10 iterations[1] of both IRL and SSIRL. The results are averaged over 10 simulations[2]. The improvement levels off after 6 iterations. In Figure 4.3 we fixed the number of unlabeled trajectories to 10 and observe the performance under the true reward function in a single run as we iterate IRL and SSIRL. In this case, IRL converged to the expert's feature counts in few iterations. On the other hand, notice that SSIRL what was able in the begging discover even better mixture policy than the expert. We also show the performance of the optimal policy given the true reward (Figure 4.3, red squares). After the fifth iteration, performance SSIRL drops as the unlabeled trajectories are outweighed by the newly generated policies. As mentioned before, as we increase the number of iterations SSIRL converges to IRL. However, as Figure 4.3 shows, a better policy may be discovered much sooner.

1. Note again, that since number of unlabeled trajectories is finite, after many iterations both IRL and SSL converge to an approximately same policy.
2. The performance of IRL is obviously independent of the number of unlabeled trajectories. The small fluctuations in this and the following plots are due to averaging over different 10 simulations each time. This is to assure that the both algorithms are given the same (noisy) sample of the expert's feature counts.

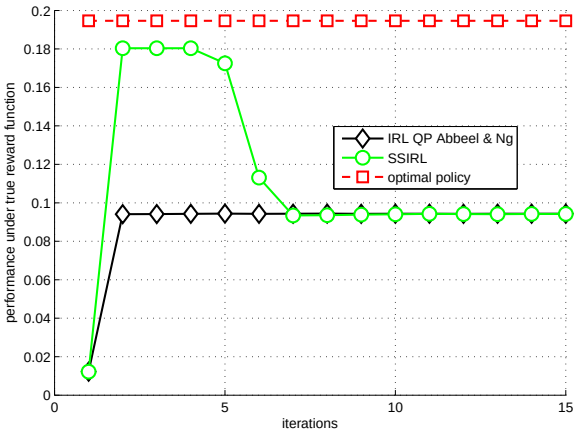

Figure 4.3: **Gridworld** - Performance of the mixture policies under the true reward (unknown to both algorithms). The true reward function and the reward function for unlabeled trajectories are chosen randomly and the graph shows the performance of the mixture policy. The performance of the optimal policy is shown by the dashed line with red squares.

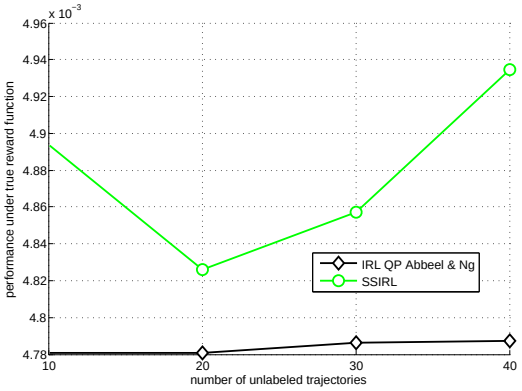

Figure 4.4: **Two goals navigation** - Performance after 5 iterations of both algorithms as a function of the number of extra unlabeled trajectories. First half of the unlabeled trajectories goes from start to "A", the other half goes to "B".

The results for the both setups in *two goals navigation* domain are shown in Figures 4.4 and 4.5. In both cases we vary the number of the unlabeled trajectories given to SSIRL and observed how it affects the performance under the true reward. The figures display the performance of the mixture policies under the true reward after 5 iterations of both IRL and SSIRL. We can observe the improvement of the learned mixture policies in the case of SSIRL, which is typically greater when more unlabeled trajectories are provided.

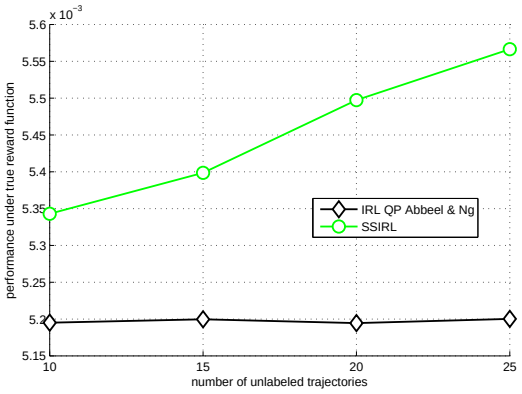

Figure 4.5: **Two goals navigation** - Performance after 5 iterations of both algorithms as a function of the number of extra unlabeled trajectories. First half of the unlabeled trajectories goes from start to "A", the other half goes the opposite direction.

5. Conclusion

We presented a semi-supervised IRL algorithm for apprenticeship learning. The algorithm combines the max-margin IRL algorithm of Abbeel and Ng [2004] and transductive SVMs [Joachims, 1999b]. Our SSIRL algorithm leverages the cluster assumption in the feature expectations space of the provided trajectories. To our best knowledge this is the first work that makes use of the unlabeled trajectories in the apprenticeship learning.

One of the limitation of the SSIRL is that the expected feature counts of provided trajectories should comply with the clustering assumption. Otherwise, including them in the optimization maybe not helpful at all, as shown by Singh et al. [2008]. Moreover, our method inherits the well known limitations of IRL [Abbeel and Ng, 2004], such as the assumption of the linearity for the reward in the feature space or only a mixture policy as the output of the algorithm. Finally, since the stopping criterion for IRL is defined as closeness to the expert trajectory in the feature space, other stopping criteria need to be used to recognize that the policy found by SSIRL is performing better.

In the future, we plan to enhance other methods with unlabeled trajectories, such as max-margin planning or maximum entropy inverse reinforcement learning. Moreover, we plan to investigate leveraging manifold assumption, which is also well studied in the semi-supervised learning [Zhu et al., 2003].

Acknowledgments

This research work was supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council and FEDER through the "contrat de projets état region 2007–2013", and by PASCAL2 European Network of Excellence. The research leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 270327 (project CompLACS).

References

Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on machine learning*, 2004.

Kristin Bennett and Ayhan Demiriz. Semi-Supervised Support Vector Machines. In *Advances in Neural Information Processing Systems 11*, pages 368–374, 1999.

Abdeslam Boularias, Jens Kober, and Jan Peters. Model-free inverse reinforcement learning. In *Proceedings of Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

Michael Grant, Stephen Boyd, and Yinyu Ye. Disciplined Convex Programming and CVX. *Review Literature And Arts Of The Americas*, C(3):1–26, 2006.

Thorsten Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods Support Vector Learning*, pages 169–184, 1999a.

Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999b.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear Inverse Reinforcement Learning with Gaussian Processes. In *NIPS*, pages 1–9, 2011.

Andrew Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Inverted Autonomous Helicopter Flight via Reinforcement Learning. In *International Symposium on Experimental Robotics*, 2004.

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In Jorge Pinho De Sousa, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. *Proceedings of the 23rd ICML*, 3(10), 2006.

Aarti Singh, Robert D Nowak, and Xiaojin Zhu. Unlabeled data: Now it helps, now it doesn't. In *Advances in Neural Information Processing Systems 21*, 2008.

Vladimir N Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, University of Wisconsin-Madison, 2008.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003.

Brian Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. *Proc AAAI*, 2008.

JMLR: Workshop and Conference Proceedings 24:143–153, 2012 10th European Workshop on Reinforcement Learning

An investigation of imitation learning algorithms for structured prediction

Andreas Vlachos ANDREAS.VLACHOS@CL.CAM.AC.UK
Computer Laboratory, University of Cambridge, UK.

Editor: Marc Peter Deisenroth, Csaba Szepesvári, Jan Peters

Abstract

In the imitation learning paradigm algorithms learn from expert demonstrations in order to become able to accomplish a particular task. Daumé III et al. [2009] framed structured prediction in this paradigm and developed the search-based structured prediction algorithm (SEARN) which has been applied successfully to various natural language processing tasks with state-of-the-art performance. Recently, Ross et al. [2011] proposed the dataset aggregation algorithm (DAGGER) and compared it with SEARN in sequential prediction tasks. In this paper, we compare these two algorithms in the context of a more complex structured prediction task, namely biomedical event extraction. We demonstrate that DAGGER has more stable performance and faster learning than SEARN, and that these advantages are more pronounced in the parameter-free versions of the algorithms.

Keywords: Real-world Applications, Imitation Learning, Natural Language Processing, Structured Prediction.

1. Introduction

Imitation learning algorithms aim at learning controllers from demonstrations by human experts [Schaal, 1999; Abbeel, 2008; Syed, 2010]. Unlike standard reinforcement learning algorithms [Sutton and Barto, 1996], they do not require the specification of a reward function by the practitioner. Instead, the algorithm observes a human expert perform a series of actions to accomplish the task in question and learns a policy that "imitates" the expert with the purpose of generalizing to unseen data. These actions have dependencies between them, since earlier ones affect the input to the following ones and the algorithm needs to handle the discrepancy between the actions of the expert in the demonstration during training and the actions predicted by the learned controller during testing. Imitation learning algorithms have been applied successfully to a variety of domains and tasks including autonomous helicopter flight [Coates et al., 2008] and statistical dialog management [Syed and Schapire, 2007].

For structured prediction tasks in natural language processing (NLP) the output space of an instance is a structured group of labels [Smith, 2011]. For example, in part-of-speech tagging the output for a sentence is a sequence of part-of-speech tags, or in handwriting recognition the output is a sequence of characters. Structures can often be more complex than sequences, for example in syntactic dependency parsing the output space for a sentence is a set of labeled edges spanning the words. While learning methods such as Conditional Random Fields [Lafferty et al., 2001] and Markov Logic Networks [Domingos and Lowd,