Structured Apprenticeship Learning

Abdeslam Boularias[1], Oliver Krömer[2], and Jan Peters[1,2]

[1] Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany
[2] Darmstadt University of Technology, 64289 Darmstadt, Germany

Abstract. We propose a graph-based algorithm for apprenticeship learning when the reward features are noisy. Previous apprenticeship learning techniques learn a reward function by using only local state features. This can be a limitation in practice, as often some features are misspecified or subject to measurement noise. Our graphical framework, inspired from the work on Markov Random Fields, allows to alleviate this problem by propagating information between states, and rewarding policies that choose similar actions in adjacent states. We demonstrate the advantage of the proposed approach on grid-world navigation problems, and on the problem of teaching a robot to grasp novel objects in simulation.

1 Introduction

Programming robots to perform complicated tasks, such as grasping and manipulating objects, is a laborious and time-intensive engineering process. Markov Decision Processes (MDPs) provide an efficient mathematical tool to handle such tasks with minimum human effort. In this framework, the task is simply defined by a reward function. However, in many problems, even the specification of a reward function is not always straightforward. An alternative approach consists of demonstrating examples of a desired behavior and learning a policy that leads to a similar behavior. This type of learning is known as imitation learning and has been widely explored in robotics [1].

Abbeel and Ng [2] introduced a new paradigm of imitation learning known as apprenticeship learning. Rather than directly mimicking the actions of the human, the aim of apprenticeship learning is to recover a reward function under which the human policy is optimal. The learned reward function is then used to find an optimal policy. The process of recovering a reward function is known as Inverse Reinforcement Learning (IRL).

Prior work on apprenticeship learning is based on representing the rewards as a function of state-action features [3–8]. However, this can be a problem in practice when the reward features are noisy or misspecified. Therefore, the features specified by a user are not always sufficient for describing a reward function and for choosing actions accordingly.

An example problem would be planning to grasp an unknown object using visual information. The calculated features are often subject to noise due to measurement errors and self-occlusions. It is also difficult to encode the preference for grasping an object from a specific part, such as a handle, given that these parts come in different shapes.

A similar problem in computer vision, known as segmentation, has been efficiently solved using a family of graphical models known as Markov Random Fields (MRFs) [9–12]. The key insight behind the performance of Markov fields is that neighbor points on an image tend to have similar labels. Therefore, even a small set of noisy features can be sufficient for classifying a point when considered together with its neighbors. However, Markov fields classify the points by using only immediate costs (or rewards) and cannot be used for learning complex goal-directed behaviors, such as manipulating objects.

In this paper, we build on this insight and introduce a new apprenticeship learning technique that extends Markov Random Fields to sequential decision-making problems. We start by specifying a graph that loosely indicates which pairs of states are supposed to have similar optimal actions. Subsequently, we derive a distribution on policies, wherein the probability of a policy is proportional to its value, and inversely proportional to the number of pairs of adjacent states that have different actions. Consequently, policies are penalized for selecting actions that are inconsistent with the graph. We show that this distribution is an MRF, and describe a dynamic programming procedure that reduces planning in MDPs with MRFs to a sequence of inference problems in MRFs.

The experimental analysis, presented at the end of this paper, shows that this approach can improve the performance of an apprenticeship learning algorithm when the reward features are noisy or misspecified. Specifically, we compare the proposed algorithm to the MaxEnt IRL algorithm [7] on grid-worlds with long planning horizons. We also compare to our previous work on learning to grasp new objects [13]. In [13], the grasping points on an object are classified using an MRF, while the preshaping and the approach direction of the robot hand are given by a heuristic. In this paper, we show how to learn complete grasping policies by using structured apprenticeship learning.

2 Background

In this section, we provide the theoretical background that is necessary for understanding the remainder of this paper.

2.1 Markov Decision Processes

Formally, a Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \mu_0, \gamma)$, where $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions. T is a transition function with $T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ for $s, s' \in \mathcal{S}, a \in A$, and R is a reward function where $R(s, a)$ is the reward given for executing action a in state s. The initial state distribution is denoted by μ_0, and $\gamma \in [0, 1]$ is a discount factor. A Markov Decision Process without a reward function is denoted by MDP\R. We assume that the reward function is a linear combination of K feature vectors ϕ_k

with weights θ_k,

$$\forall(s,a) \in \mathcal{S} \times \mathcal{A} : R(s,a) = \sum_{k=1}^{K} \theta_k \phi_k(s,a).$$

A deterministic policy π is a function that returns an action $a = \pi(s)$ for each state s. The expected return $J(\pi)$ of a policy π is the expected sum of rewards that will be received when following policy π, i.e.

$$J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | \mu_0, \pi, T].$$

An optimal policy π^* is one satisfying $\pi^* \in \arg\max_\pi J(\pi)$. The expectation of a feature ϕ_k for a policy π is defined as

$$\phi_k^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi_k(s_t, a_t) | \mu_0, \pi, T].$$

Using this definition, the expected return of a policy π can be written as a linear function of the feature expectations

$$J(\pi) = \sum_{k=1}^{K} \theta_k \phi_k^\pi.$$

2.2 Apprenticeship learning

The aim of apprenticeship learning is to find a policy π that is nearly as good as a policy π^E demonstrated by a human expert, i.e., $J(\pi) \geq J(\pi^E) - \epsilon$. However, the expected returns of π and π^E cannot be directly compared, unless a reward function is provided. As a solution to this problem, Ng and Russell [14] proposed to first learn a reward function, assuming that the expert is optimal, and then use it to recover the expert's generalized policy.

However, the problem of learning a reward function given an optimal policy is ill-posed [2]. In fact, a large class of reward functions may lead to the same optimal policy. Most of the apprenticeship learning literature has focused on solving this particular problem. Examples of the proposed solutions include incorporating prior information on the reward function, minimizing the margin $\|J(\pi) - J(\pi^E)\|$, or maximizing the entropy of the distribution on state-actions under a learned stochastic policy [7]. In this work, we will use the maximum entropy regularization.

The principle of maximum entropy states that the simplest policy that best represents the provided examples is the one with the highest entropy, subject to the constraint of matching the expected return of the demonstrated actions. This latter constraint can be satisfied by ensuring that the feature counts of the learned policy match with those of the demonstration,

$$\forall k \in \{1, \ldots, K\} : \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi_k(s_t, a_t) | \mu_0, \pi, T] = \hat{\phi}_k \tag{1}$$

where $\hat{\phi}_k$ denotes the empirical expectation of feature k calculated from the demonstration. The MaxEnt IRL approach [7] consists of finding the parameters θ of a policy π that maximizes the entropy of the distribution on the state-action trajectories subject to constraint (1). Solving this problem leads to maximizing the likelihood of the demonstrated trajectories under an exponential distribution of the policies.

2.3 Markov Random Fields

A Markov Random Field (MRF) is a graphical model used for representing joint probability distributions. The MRF defines a probability distribution over N discrete variables $Y = \{y_1, \dots, y_n\}$. Each variable corresponds to the label of a node in a graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Each node x_i is assigned to a label y_i from a set $\mathcal{L}$ of possible labels. Therefore, the MRF defines a probability distribution over $\mathcal{L}^N$.

We focus on a particular tractable class of MRFs known as Associative Markov Network (AMN) [15], where potentials $\rho(x_i, y_i)$ and $\rho(x_i, x_j)$ are associated with each node $x_i \in \mathcal{V}$ labeled by y_i, and each edge $(x_i, x_j) \in \mathcal{E}$ such that x_i and x_j have the same label. The AMN model uses the log-linear function for representing a potential as a function of the features, i.e. $\log \rho(x_i, y_i) = \sum_k \theta_k \phi(x_i, y_i)$ and $\log \rho(x_i, x_j) = \sum_k \lambda_k \psi_k(x_i, x_j)$, where $\theta_k \in \mathbb{R}$ are node weights, $\phi_k(x_i, y_i) \in \mathbb{R}$ are features of node x_i labeled by y_i, $\lambda_k \in \mathbb{R}$ are edge weights, and $\psi_k(x_i, x_j) \in \mathbb{R}$ are features of edge (x_i, x_j). The joint probability distribution on the labels $(y_1, \dots, y_n)$ is given by

$$P(y_1, \dots, y_n | x_1, \dots, x_n) \propto \exp\Big(\sum_{x_i \in \mathcal{V}} \sum_k \theta_k \phi(x_i, y_i) + \sum_{\substack{(x_i, x_j) \in \mathcal{E} \\ \text{s.t. } y_i = y_j}} \sum_k \lambda_k \psi_k(x_i, x_j) \Big).$$

3 Structured Apprenticeship Learning

In this section, we present the structured apprenticeship learning problem and show how it can be solved efficiently.

3.1 Key insight

The classical framework of apprenticeship learning is based on hand-coding the features ϕ of the reward and learning the weights θ. In practice, it is often difficult to find an appropriate set of features that contains necessary and sufficient information about the reward. Moreover, these features are usually obtained from empirical data and are subject to measurement errors. On the other hand, most real-world problems exhibit a certain structure wherein states that are close together tend to have the same optimal action. This implicit information about the optimal policy, given by the structure, can be used to partially overcome the problem of finding an appropriate set of reward features. Intuitively, a distance can be interpreted as a kernel that measures the similarity between the approximate values of two states under an optimal policy. In many robotic applications, such as navigation, this distance is simply the Euclidean or the geodesic distance.

3.2 Problem statement

Given an appropriate definition of a measure between states, we construct a k-nearest neighbors graph where the nodes correspond to states and the set of edges is denoted by $\mathcal{E}$. Structured apprenticeship learning can be formulated as the problem of finding a distribution P on deterministic policies, denoted by π, that has the highest possible entropy while matching the expected value of each state-action feature ϕ_k with the one calculated from the demonstration, $\hat{\phi}_k$. Moreover, a set of edge features ψ_k is defined over edges in $\mathcal{E}$. Edge features $\psi_k(s_i, s_j)$ take a zero value when $\pi(s_i) \neq \pi(s_j)$. The distribution P should also match the expected value of each edge feature ψ_k with the one calculated from the demonstration, $\hat{\psi}_k$. Thus, this problem can be defined as

$$\max_{P,\mu^\pi} \Big(- \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \log P(\pi) \Big), \tag{2}$$

subject to

$$\begin{aligned} \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) &= 1, \\ \forall \phi_k : \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \sum_{s \in \mathcal{S}} \mu^\pi(s) \phi_k(s, \pi(s)) &= \hat{\phi}_k, \\ \forall \psi_k : \sum_{(s_i,s_j) \in \mathcal{E}} \psi_k(s_i, s_j) \sum_{\pi, \pi(s_i) = \pi(s_j)} P(\pi) &= \hat{\psi}_k, \end{aligned}$$

where $\mu^\pi(s)$ is the expected discounted number of visits of state s,

$$\forall \pi, s : \mu^\pi(s) = \mu_0(s) + \gamma \sum_{s'} \mu^\pi(s') T(s', \pi(s'), s).$$

The state-action features ϕ_k correspond to the basis functions of the reward function. The edge features ψ_k can be interpreted as the basis functions of a reward given for choosing the same actions for adjacent states. For instance, $\psi_k(s_i, s_j)$ can be the inverted distance between states s_i and s_j.

Another interesting example is when $\psi_k(s_i, s_j) = 1, \forall (s_i, s_j) \in \mathcal{E}$, in which case the last constraint in Problem 2 corresponds to forcing the probability of the policies that choose the same action in adjacent states to be equal to the empirical probability $\hat{\psi}_k$. In this manner, the robot learns a policy that also imitates the structure of the human policy. If the provided structure is not relevant to the task, then $\hat{\psi}_k$ will be low. Consequently, the learned policy will not be forced to have a similar structure.

3.3 Proposed solution

The Lagrangian of this problem is given by

$$\begin{aligned} L(P,\eta,\theta,\lambda) = &- \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi)\log P(\pi) + \eta\Big(\sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) - 1\Big) \\ &+ \sum_k \theta_k \Big(\sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi) \sum_{s \in \mathcal{S}} \mu^\pi(s)\phi_k(s,\pi(s)) - \hat{\phi}_k \Big) \\ &+ \sum_k \lambda_k \Big(\sum_{\pi, \pi(s_i)=\pi(s_j)} P(\pi) \sum_{(s_i,s_j)\in\mathcal{E}} \psi_k(s_i,s_j) - \hat{\psi}_k \Big). \end{aligned}$$

Therefore

$$\begin{aligned} \partial_{P(\pi)} L(P,\eta,\theta,\lambda) = &\sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) \\ &- \log P(\pi) + \eta - 1. \end{aligned}$$

The optimal solution satisfies $\partial_{P(\pi)} L(P,\eta,\theta,\lambda) = 0$, then

$$\log P(\pi) = \sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) - \log Z(\theta,\lambda), \tag{3}$$

where $Z(\theta,\lambda)$ is a partition function. Therefore

$$P(\pi) \propto \exp\big(\sum_s \mu^\pi(s) \sum_k \theta_k \phi_k(s,\pi(s)) + \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \text{s.t. } \pi(s_i)=\pi(s_j)}} \sum_k \lambda_k \psi_k(s_i,s_j) \big). \tag{4}$$

3.4 Relation to other methods

The probability distribution given by Equation 4 is a Markov Random Field, as illustrated in Figure 1. The probability of choosing action a in a given state s depends on the expected return of (s,a) and the actions chosen in neighboring states. There is a clear similarity between the distribution of policies in structured apprenticeship learning and the distribution of joint labels in Associative Markov Networks (AMN) [16] in particular. In fact, the proposed framework generalizes AMN to sequential decision making problems. States are the input points and actions are the labels, the labeling cost is given by the reward. The label distribution in AMN can be derived from Equation 4 by setting $\gamma = 0$, then μ^π becomes equal to μ_0, the initial state distribution, which is a constant factor that can be included in the features ϕ_k. Also, the MaxEnt method [7] can be derived from Equation 4 by setting $\lambda = 0$. Note that Ziebart et al. [7] use a

	$\lvert\mathcal{E}\rvert = 0$	$\lvert\mathcal{E}\rvert \neq 0$
$\gamma = 0$	Logistic regression	AMN [16]
$\gamma \neq 0$	MaxEnt IRL [7]	Structured Apprenticeship Learning

Table 1. Relation between Structured Apprenticeship Learning and other methods.

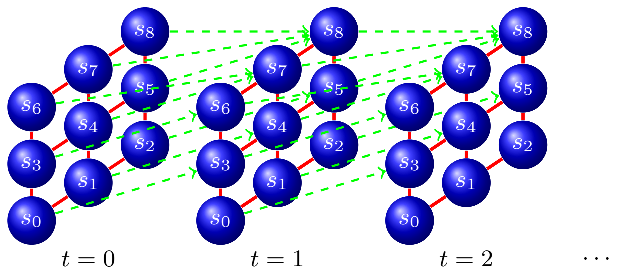


Fig. 1. Structured Markov Decision Process

distribution on paths instead of policies. The following table shows the relation between structured apprenticeship learning and other methods.

Finally, we should mention that the concept of using structured output prediction for learning by demonstration was also considered in [17]. Although, the approach of [17] consists in classifying robot trajectories using conditional random fields in a supervised learning fashion, without considering a reward function and planning the trajectories.

3.5 Learning procedure

We provide a solution for finding the reward parameters θ and the structure parameters λ that maximize the likelihood of an expert policy π^E demonstrated in a domain that can be different from the testing domain. The demonstrations are given by state-action trajectories, and the empirical feature averages $\hat{\phi}_k$ and $\hat{\psi}_k$ are calculated from these trajectories. We denote the set of states that appear in the demonstrations by $\mathcal{S}^E$, and the corresponding set of edges by $\mathcal{E}^E$.

From the Representer Theorem [18], we know that the parameters θ and λ that maximize $\log P(\pi)$ (Equation 3) are given by

$$\theta_k = \sum_{s \in \mathcal{S}^E} \alpha_s \phi_k(s, \pi^E(s)),$$

$$\lambda_k = \sum_{\substack{(s_i, s_j) \in \mathcal{E}^E \\ \text{s.t. } \pi^E(s_i) = \pi^E(s_j)}} \beta_{s_i, s_j} \psi_k(s_i, s_j),$$

where $\alpha_s, \beta_{s_i,s_j} \in \mathbb{R}$. Therefore, the log-probability of a deterministic policy π defined on an arbitrary structured MDP $(\mathcal{S}, \mathcal{E}, \mathcal{A}, T, \mu_0, \gamma)$ is given by

$$\log P(\pi) = \underbrace{R_\beta(\mathcal{E}, \pi)}_{\text{structure reward}} + \underbrace{\sum_s \mu^\pi(s) R_\alpha(s, \pi(s))}_{\text{expected return}} - \log Z(\alpha, \beta), \tag{5}$$

$$R_\alpha(s, a) \overset{def}{=} \sum_{s' \in \mathcal{S}^E} \alpha_{s'} k(\langle s, a\rangle, \langle s', \pi^E(s')\rangle),$$

$$R_\beta(\mathcal{E}, \pi) \overset{def}{=} \sum_{\substack{(s_i,s_j)\in\mathcal{E} \\ \pi(s_i)=\pi(s_j)}} \sum_{\substack{(s'_i,s'_j)\in\mathcal{E}^E \\ \pi^E(s'_i)=\pi^E(s'_j)}} \beta_{s'_i,s'_j} k_e(\langle s_i, s_j\rangle, \langle s'_i, s'_j\rangle),$$

where k and k_e are kernel functions used for measuring similarities between state-action couples and between edges respectively, they are defined as

$$k(\langle s, a\rangle, \langle s', a'\rangle) = \sum_k \phi_k(s, a)\phi_k(s', a'),$$

$$k_e(\langle s_i, s_j\rangle, \langle s'_i, s'_j\rangle) = \sum_k \psi_k(s_i, s_j)\psi_k(s'_i, s'_j).$$

The partition function $Z(\alpha, \beta)$ is given by

$$Z(\alpha, \beta) = \sum_{\pi \in \mathcal{A}^{|\mathcal{S}|}} \exp\Big(R_\beta(\mathcal{E}, \pi) + \sum_s \mu^\pi(s) R_\alpha(s, \pi(s))\Big). \tag{6}$$

In the learning phase, Equation 5 is used for finding parameters α and β that maximize the likelihood of the expert's policy π^E. Since this likelihood function is concave, the optimal parameters α and β can be found by using standard optimization methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. A key drawback of this approach is the computational cost of calculating the partition function $Z(\alpha, \beta)$ at each step of the optimization algorithm, which is $\mathcal{O}(|\mathcal{A}|^{|\mathcal{S}|}|\mathcal{S}|^3)$, this corresponds to the cost of calculating the values of all the deterministic policies using value iteration for instance.

In practice, this problem can be addressed by using several possible tricks. First, we reuse the values calculated for a given policy π as the initial values of all the policies that differ from π in one state only, the values of these policies are found after a few iterations. Second, we may be interested in finding a probability distribution on a restricted set of eligible policies, instead of all possible policies. The learned reward function will then discriminate the expert's policy π^E against other alternative candidates. For example, we can consider all the policies that differ from π^E by only one action, or the optimal policies given a sequence of hypothesized reward functions, as in Maximum Margin Markov Networks [16]. Finally, we can decompose the state space into a set of weakly connected components $\mathcal{C} = \{S_i \subset \mathcal{S}\}$, and separately calculate the partition function of each component, which reduces the computational complexity to $\mathcal{O}(\sum_{\mathcal{S}_i \in \mathcal{C}} |\mathcal{A}|^{|\mathcal{S}_i|}|\mathcal{S}_i|^3)$. This procedure is given in Algorithm 1.

```
Input: A structured MDP\R $(\mathcal{S}, \mathcal{E}, \mathcal{A}, T, \mu_0, \gamma)$ ;
Let $\mathcal{C}_0$ be the set of weakly connected components in the graph defined by the
states and the edges in $\mathcal{E}$ ;
$t \leftarrow 0$;
repeat
    $t \leftarrow t + 1$ ; $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
    foreach $\mathcal{S}_i \in \mathcal{C}_t, s \in \mathcal{S}_i, a \in \mathcal{A}, \mathcal{S}_j \in \mathcal{C}_t, s' \in \mathcal{S}_j$ do
        if $T(s, a, s') \neq 0$ then
            $\mathcal{S}_k = \mathcal{S}_i \cup \mathcal{S}_j$ ; $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{\mathcal{S}_k\}$ ;
        end
    end
until $\mathcal{C}_t = \mathcal{C}_{t-1}$;
Output: A set of weakly connected components $\mathcal{C}$;
```

Algorithm 1: Decomposing the state space into weakly connected components.

3.6 Finding policies for finite horizons

For a finite horizon H, an optimal policy π can be non-stationary, i.e. $\pi_{0:H} = (\pi_0, \pi_1, \ldots, \pi_H)$. Moreover, if the initial state distribution μ_0 is unknown, then Problem 2 should be stated for every initial state and time-step. Assuming that the reward and structure (edges) parameters α and β are stationary (they do not depend on the starting state or time), the solution is given by the conditional probability distribution

$$P(\pi_t | \pi_{t+1:H}) \propto \exp\Big(R_\beta(\mathcal{E}, \pi_t) + \sum_s V_\alpha^{\pi_{t:H}}(s)\Big),$$
$$V_\alpha^{\pi_{t:H}}(s) = R_\alpha(s, \pi_t(s)) + \gamma \sum_{s'} T(s, \pi_t(s), s') V_\alpha^{\pi_{t+1:H}}(s').$$

Algorithm 2 describes a dynamic programming procedure for finding a policy $\pi^*_{0:H} = (\pi^*_0, \pi^*_1, \ldots, \pi^*_H)$ that satisfies

$$\forall t \in [0, H] : \pi^*_t = \operatorname*{argmax}_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t | \pi^*_{t+1:H}).$$

The planning problem is reduced to a sequence of inference problems in Markov fields, i.e. finding joint labels that have the highest probability. The inference problem itself also can be efficiently solved using techniques such as graph min-cut [19], α-expansions and linear programming relaxation [15]. We adopt the α-expansions technique. For more details, we refer the reader to Taskar (2004) [16].

4 Experiments

In this section, we present experiments on learning to navigate in gridworlds with noisy reward features, and learning to grasp unknown objects. We compare

```
Input: A structured MDP $(\mathcal{S},\mathcal{E},\mathcal{A},T,\gamma)$ with reward parameters $\alpha$ and
       structure parameters $\beta$, a planning horizon $H$;
$\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : Q_\alpha^{H+1}(s,a) = 0$;
for $t = H : 0$ do
    $\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : Q_\alpha^t(s,a) = R_\alpha(s,a) + \gamma \sum_{s'} T(s,a,s') Q_\alpha^{t+1}(s', \pi_{t+1}^*(s'))$;
    Use an inference algorithm in the MRF defined on the graph $(\mathcal{S},\mathcal{E})$ to label
    states with actions. The potential of the nodes is given by $Q_\alpha$ and the
    potential of the edges is given by $R_\beta$;
    Denote by $\pi_t^*$ the policy returned by the inference algorithm;
end
Output: A non-stationary policy $\pi^* = (\pi_0^*, \pi_1^*, \dots, \pi_H^*)$;
```
Algorithm 2: Reducing planning in structured MDP to inference in MRFs.

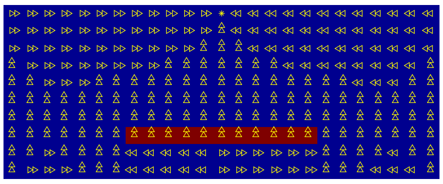
Structured Apprenticeship Learning (SAL) with MaxEnt IRL [7]. Note that Ziebart et al. [7] used a distribution on trajectories while we use distributions on policies.

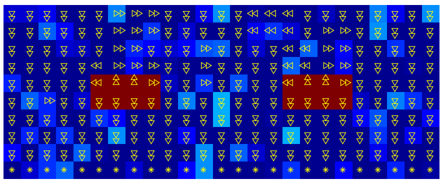
4.1 Gridworlds

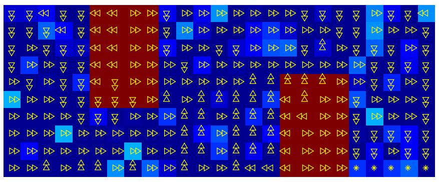
We consider 10×25 gridworlds. A state corresponds to the location of a robot, which has four actions for moving in one of the four directions of the compass. The actions are stochastic, the robot is randomly moved to one of the four adjacent states with probability 5%. There are two reward features, ϕ_1 and ϕ_2. Feature ϕ_1 takes value 1 in the goal states and 0 elsewhere. Feature ϕ_2 indicates bad regions that should be avoided (big obstacles, large holes, slippery floors, etc. . .). The true reward function is given by $R(s,a) = \phi_1(s) - 20\phi_2(s)$.

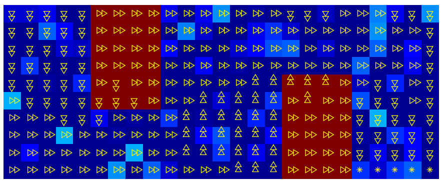
Based on the assumption that bad states tend to be regrouped in large regions, an optimal policy is expected to select the same action for most of the time, and occasionally turn left or right to avoid an obstacle. Therefore, we use the Manhattan distance on the grid as a similarity measure between states, and consider the immediate neighbors as adjacent states in the graph used by SAL. We use a constant edge feature, set to 1. The parameters of MaxEnt IRL and SAL are learned by maximizing the likelihood of the policy shown in Figure 2 (a) using the BFGS method. In the learning process, we restrict the policy distribution to the set of policies that differ from the demonstration in at most one state. This was sufficient for learning fairly accurate reward weights, $(1.2, -18.0)$ for SAL and $(1.2, -20.7)$ for MaxEnt, while reducing the learning time to 173 seconds for SAL and only 4 seconds for MaxEnt. The learned edge weight for SAL is $\lambda = 0.28$.

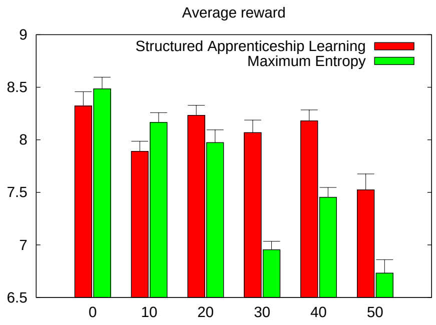
Tests were performed in two gridworlds, shown in Figures 2 (b,c,d). In each gridworld, there are two large regions characterized by ϕ_2 set to 1. The remaining states have ϕ_2 set to 0. Randomly selected states have been noised, i.e. their values of ϕ_2 were altered to values uniformly sampled from the interval $[0, 0.2]$. We used the α-expansions algorithm for inference in SAL. The average planning times were 0.58 and 1.7 seconds for SAL, and 0.4 and 1.05 seconds for MaxEnt.


(a) Training with optimal policy

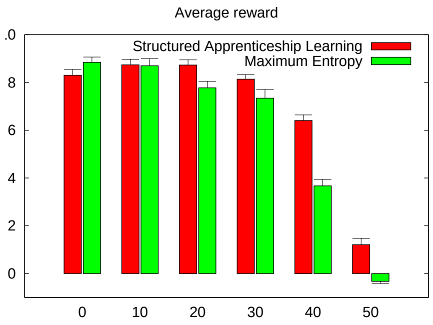

(b) Testing with SAL


(c) Testing with MaxEnt IRL


(d) Testing with SAL



(e) First domain results



(f) Second domain results

Fig. 2. Experiments in gridworlds. Blue indicates a low value of a feature associated with negative reward, and red indicates a higher value of that feature. In the testing domains, a white noise is added to the negative-weighted feature of randomly chosen states. Subfigures (e) and (f) show the average rewards of MaxEnt and SAL as a function of the percentage of noisy states.

Figures 2 (e,f) show the average actual rewards of SAL and MaxEnt policies as a function of the percentage of states that have been noised. The results are averaged over 10^5 trials of length 50 with the discount factor $\gamma = 1$. Notice that with low levels of noise, MaxEnt slightly outperforms SAL. This is due to the fact that the SAL policy prefers to choose a similar action for adjacent states, even when sometimes a different action is optimal. For the same reason, SAL significantly outperforms MaxEnt in high levels of noise, where the robot with MaxEnt policy spends most of its time trying to avoid most of the noisy states, as shown in Figure 2 (c).

This experiment shows that SAL can improve over MaxEnt IRL when the reward features are noisy. However, when the noise is too small or absent, the performance of SAL can be lower than that of MaxEnt IRL. This is due to the bias introduced by SAL, which favors smooth policies. Nevertheless, SAL is intended to be used only when the noise level is significant.

4.2 Grasping unknown objects

From a high-level point of view, grasping an object can be seen as an MDP with three steps: reaching, preshaping, and grasping (Figure 3). At any step, the robot can either proceed to the next step or restart from the beginning and get a reward of 0. The robot always starts at $t = 0$ from the same initial state s_0, the set of actions corresponds to the set of points on the surface of the object, assuming that the approach direction is always given by the surface normal vector. At $t = 1$, the state is given by a surface point and an approach direction, the set of actions correspond to the set of all possible hand orientations. At $t = 2$, the state is given by a surface point, an approach direction and a hand orientation. There are two possible last actions, closing the fingers or restarting.

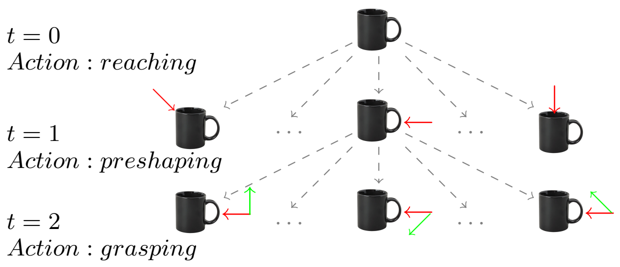


Fig. 3. Grasping as a three-step Markov Decision Process.

In this experiment, we are interested in learning to grasp objects from their handles. The reward of each step depends on the current state. There is no reward at $t = 0$. The reward R_1 defined at $t = 1$ is a function of the first three eigenvalues of the scatter matrix defined by the 3D coordinates of the points inside a small ball centered on the selected point. These features indicate if a point is on a handle. The reward R_2, defined at $t = 2$, is a function of collision features. We simulate the trajectories of 10 equidistant points on each finger of a Barrett robot hand (a three-fingered gripper). The collision features are binary variables indicating whether or not there will be a collision with the object, during the grasping, for each one of the specified finger points.

Based on the assumption that points that are close to each other should have the same action (i.e. same approach direction and hand orientation), the structure of this MDP is given by the k-nearest neighbors graph, using the Euclidean distance and $k = 6$ in the state space of positions (or surface points), and the angular distance, with $k = 2$ in the discretized state space of hand orientations. We use a quadratic kernel for learning R_1, and the Hamming distance between the feature vectors as a kernel for learning R_2. We also use a single constant feature for all the edges.

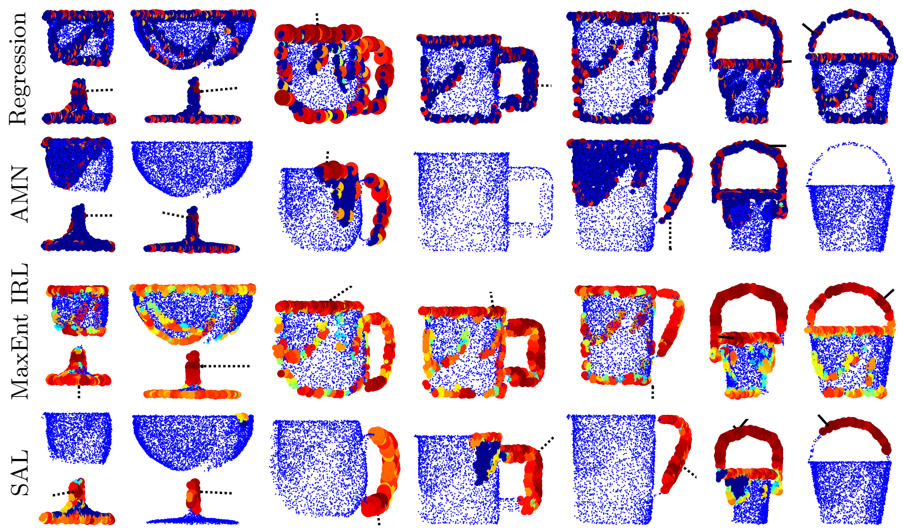


Table 2. Learned Q-values at $t = 0$. Each point on an object corresponds to an action. Blue indicates low values and red indicates high values. The black arrow indicates the approach direction in the optimal policy according to the learned reward function.

We used one object for training and provided six trajectories leading to a successful grasp from its handle. For testing, we compared SAL with MaxEnt IRL, AMN and Logistic Regression, which is equivalent to AMN without the graph structure. For AMN and Logistic Regression, only the reward R_1 at time-step 1 is learned, since these are classification methods and do not consider subsequent rewards.

Table 2 shows the Q-values at $t = 0$ and the approach directions at optimal grasping points given the reward functions learned by different algorithms. Notice how SAL improves over the other methods by generally giving high values to handle points only. The values of the other points are zeros because the optimal action at these points is to restart rather than to grasp. The confusion in the other methods comes from noise features and self-occlusions. Notice also that SAL improves over AMN by considering the reward at $t = 2$ while making a decision at $t = 1$. Figure 4 shows the percentage of successful grasps using the objects in Table 2. A grasp is labeled successful if it is located on a handle and the hand orientation is orthogonal to the handle and the approach direction.

Note that Ratliff [20] solved a similar grasp prediction problem by using a structured output prediction technique. The experimental setup used in [20] is different from ours since it contained complete 3D models of the objects instead of point clouds. We compared SAL only with MaxEnt IRL, which is a special case of SAL, in order to illustrate the advantage of using MRF. The approach of Ratliff [20] for grasping can be extended in a similar way to handle sequential decision-making. In fact, structured apprenticeship learning with MRFs is one

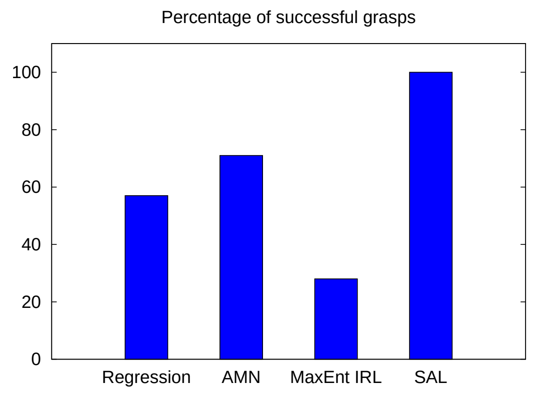


Fig. 4. Percentage of grasps labeled as successful.

way of using structured prediction in MDPs, one can also use other techniques such as Max-Margin Markov Networks [16].

We also compared with the classification methods used in [13] (AMN and logistic regression) in order to show the advantage of taking future rewards into account in grasping. In fact, AMN and logistic regression use only the reward function at the first time-step for selecting the grasping point, the approach direction is found by using a heuristic. Table 2 clearly shows that structured apprenticeship learning improves over AMN and logistic regression. In fact, the dynamic programming procedure used in SAL (Algorithm 2) backpropagates the cost related to the collisions of the fingers with the object to the first time-step. Therefore, most of the points with a high value are located on handles.

5 Conclusion

Robotic tasks, such as grasping objects, are often difficult to solve by using manual programming. A solution to this problem, known as apprenticeship learning, consists of providing the robot with a demonstration of an optimal policy for solving the task. The robot learns a reward function that explains the demonstration and uses it for generalization.

In this paper, we showed that the reward function alone is not always sufficient for properly explaining a behavior when some features are noisy, or are poorly specified. To solve this problem, we presented a new technique that we called Structured Apprenticeship Learning, and which is inspired by Markov fields. Experiments on navigation and grasping tasks confirmed that structured apprenticeship learning improves over unstructured methods when the features are noisy.

However, our approach suffers from a higher computational complexity compared to standard MDP algorithms. This problem will be investigated in a future work by using well-known approximate inference techniques in graphical models.

References

1. Schaal, S.: Is Imitation Learning the Route to Humanoid Robots? Trends in Cognitive Sciences **3**(6) (1999) 233–242
2. Abbeel, P., Ng, A.Y.: Apprenticeship Learning via Inverse Reinforcement Learning. In: Proceedings of the Twenty-first International Conference on Machine Learning (ICML'04). (2004) 1–8
3. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum Margin Planning. In: Proceedings of the Twenty-third International Conference on Machine Learning (ICML'06). (2006) 729–736
4. Ramachandran, D., Amir, E.: Bayesian Inverse Reinforcement Learning. In: Proceedings of The twentieth International Joint Conference on Artificial Intelligence (IJCAI'07). (2007) 2586–2591
5. Syed, U., Schapire, R.: A Game-Theoretic Approach to Apprenticeship Learning. In: Advances in Neural Information Processing Systems 20 (NIPS'08). (2008) 1449–1456
6. Syed, U., Bowling, M., Schapire, R.E.: Apprenticeship Learning using Linear Programming. In: Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08). (2008) 1032–1039
7. Ziebart, B., Maas, A., Bagnell, A., Dey, A.: Maximum Entropy Inverse Reinforcement Learning. In: Proceedings of The Twenty-third AAAI Conference on Artificial Intelligence (AAAI'08). (2008) 1433–1438
8. Ziebart, B., Bagnell, A., Dey, A.: Modeling Interaction via the Principle of Maximum Causal Entropy. In: Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML'10). (2010) 1255–1262
9. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.: Discriminative learning of Markov random fields for segmentation of 3d scan data. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'05). (2005) 169–176
10. Munoz, D., Vandapel, N., Hebert, M.: Onboard contextual classification of 3-D point clouds with learned high-order Markov random fields. In: Proceedings of the 2009 IEEE international conference on Robotics and Automation (ICRA'09). (2009)
11. Kohli, P., Kumar, P., Torr, P.: P3 and beyond: Solving energies with higher order cliques. In: IEEE International Conference on Computer Vision and Pattern Recognition (ICCVPR'07). (2007)
12. Ratliff, N., Bagnell, D., Zinkevich, M.: Online subgradient methods for structured prediction. In: In Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS'07). (2007)
13. Boularias, A., Kroemer, O., Peters, J.: Learning Robot Grasping from 3-D Images with Markov Random Fields. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11). (2011)
14. Ng, A., Russell, S.: Algorithms for Inverse Reinforcement Learning. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00). (2000) 663–670
15. Taskar, B., Chatalbashev, V., Koller, D.: Learning associative markov networks. In: Proceedings of the Twenty-First International Conference on Machine Learning (ICML'04). (2004)
16. Taskar, B.: Learning Structured Prediction Models: A Large Margin Approach. PhD thesis, Stanford University, CA (2004)

17. Vakanski, A., Janabi-Sharifi, F., Mantegh, I., Irish, A.: Trajectory learning based on conditional random fields for robot programming by demonstration. In: Proceedings of the IASTED International Conference on Robotics and Applications (RA'2010). (2010)
18. Schölkopf, B., Herbrich, R., Smola, A.: A Generalized Representer Theorem . Computational Learning Theory **2111** (2001) 416–426
19. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (1999) 2001
20. Ratliff, N.: Learning to Search: Structured Prediction Techniques for Imitation Learning. PhD thesis, Carnegie Mellon University (2009)