Algorithms for Learning Markov Field Policies

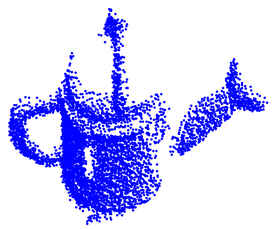
Abdeslam Boularias[1] Oliver Krömer[2] Jan Peters[1,2]

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

1 Overview

Aim: Learning to **grasp** and manipulate **unknown objects**

Barrett Robot Hand

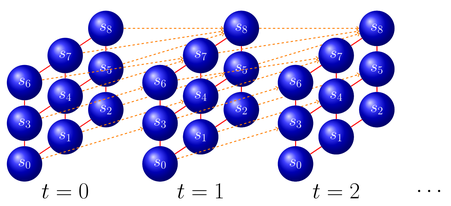


Point Cloud of an Unknown Object

Problems: **Noise** in the features of the value function.
Shape-related features are difficult to define.

Solution: Markov Random Field Policies, A Structured Output Prediction Technique that Combines Control and Vision

1. Define a **similarity measure** in the state space (domain knowledge);
2. Construct a k-nearest neighbors graph of states;
3. Learn a distribution on policies such that the probability of a policy is proportional to its value and to the number of **adjacent states** that have **the same action**.



Markov Decision Process with State Similarity Graph (in red)

2 Background

2.1 Markov Decision Process (MDP)

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, T, R)$, where

- $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions,
- T are transition probabilities, with $T(s,a,s') = P(s'|s,a)$ for $s, s' \in \mathcal{S}, a \in A$,
- R is a reward function where $R(s) \in \mathbb{R}$ is the reward of state s.

2.2 Policies and Value Functions

- A policy π is a function that maps every state into an action.
- The value of policy π is defined as $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(s_t) | s_0 = s, a_t = \pi(s_t)]$, where H is a horizon. It is also given by *Bellman equation*

$$V^\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') V^\pi(s').$$

3 Markov Random Field Policies for Reinforcement Learning

3.1 Structure penalty

- **Optimal policies are smooth**: close states tend to have the same optimal action. How can we exploit this property in reinforcement learning?
- The similarity measure is given as a **Gram matrix** K of a kernel that approximates the optimal value function, where $K(\langle \text{state}_i, \text{action}_i \rangle, \langle \text{state}_j, \text{action}_j \rangle) \in \mathbb{R}$
- The approximation error is given by the minimum **Bellman error**, defined as

$$BE(K, \pi_t) = \min_{w_t, w_{t+1} \in \mathbb{R}^{|\mathcal{S}|}} \| K_{\pi_t} w_t - \big(R + \gamma T_{\pi_t} K_{\pi_{t+1}} w_{t+1}\big) \|_1.$$

where $K_\pi(s_i, s_j) = K(\langle s_i, \pi(a_i) \rangle, \langle s_j, \pi(a_j) \rangle)$ and $T_\pi(s, s') = T(s, \pi(s), s')$

- The value function of an optimal policy has a low Bellman error with Gram matrix K (domain knowledge) $\Rightarrow$ use the Bellman error as a surrogate function for measuring how close a policy is to an optimal one

3.2 Optimization Problem

From $t = H$ to 0, find a probability distribution P on deterministic policies π_t

$$\max_P \sum_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t | \pi_{t+1:H}) \sum_{s \in \mathcal{S}} V^{\pi_{t:H}}(s), \tag{1}$$

subject to the following constraints

$$\sum_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t | \pi_{t+1:H}) = 1,$$

$$-\sum_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t | \pi_{t+1:H}) \log P(\pi_t | \pi_{t+1:H}) \geq \rho, \quad \text{(entropy bound)}$$

$$\| \sum_{\pi_t \in \mathcal{A}^{|\mathcal{S}|}} P(\pi_t | \pi_{t+1:H}) [K_{\pi_t} w_t - \gamma T_{\pi_t} K_{\pi_{t+1}} w_{t+1}] - R \|_1 \leq \epsilon. \quad \text{(Bellman error)}$$

ρ is a lower bound on the entropy of P, it is used for controlling the exploration.
ϵ is an upper bound on the Bellman error, it is used for controlling the smoothness.

3.3 Solution

$$P(\pi_t | \pi_{t+1:H}) \propto \exp\Big(\underbrace{\frac{1}{\tau}}_{\text{exploration factor}} \big(\overbrace{\sum_{s \in \mathcal{S}} V^{\pi_{t:H}}(s)}^{\text{expected sum of rewards}} + \underbrace{\lambda^T [K_{\pi_t} \mathbf{w}_t - \gamma T_{\pi_t} K_{\pi_{t+1}} \mathbf{w}_{t+1}]}_{\text{smoothness term}} \big) \Big).$$

Parameter τ is initialized to a large value, and is gradually decreased as more samples are collected. The other parameters are learned by a gradient descent on the Lagrangian dual. We use *Metropolis-Hastings* for finding $\pi^* = \arg\max_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi)$.

4 Markov Random Field Policies for Apprenticeship Learning

- Apprenticeship Learning via Inverse Reinforcement Learning (IRL) [1] consists in learning a reward function that maximizes the value of an expert's policy $\hat{\pi}$.
- The reward function is usually assumed to be linear, $R(s) = \sum_i \theta_i \phi_i(s)$.
- The learned reward is used to find a policy that generalizes the observed behavior.

4.1 Structure Matchings

- Given an expert's policy $\hat{\pi}_{0:H}$ and a Gram matrix K, we are interested in finding a distribution P on policies $\pi_{0:H}$ that has a Bellman error similar to that of $\hat{\pi}_{0:H}$.
- Sufficient condition: $\mathbb{E}_{\pi_t \sim P}[K_{\pi_t}] = K_{\hat{\pi}_t}$ and $\mathbb{E}_{\pi_{t:t+1} \sim P}[T_{\pi_t} K_{\pi_{t+1}} T_{\pi_t}^T] = T_{\hat{\pi}_t} K_{\hat{\pi}_{t+1}} T_{\hat{\pi}_t}^T$.

4.2 Solution

Enforcing the structure matching constraints, in addition to the value matching constraints [2], and maximizing the entropy of P leads to the solution

$$P(\pi_t | \pi_{t+1:H}) \propto \exp\Big(\underbrace{\sum_{s \in \mathcal{S}} V_\theta^{\pi_{t:H}}(s)}_{\text{expected sum of rewards}} + \overbrace{\sum_{s_i, s_j \in \mathcal{S}} \lambda_{i,j} K\big(\langle s_i, \pi_t(s_i) \rangle, \langle s_j, \pi_t(s_j) \rangle\big)}^{\text{smoothness term}} \Big)$$

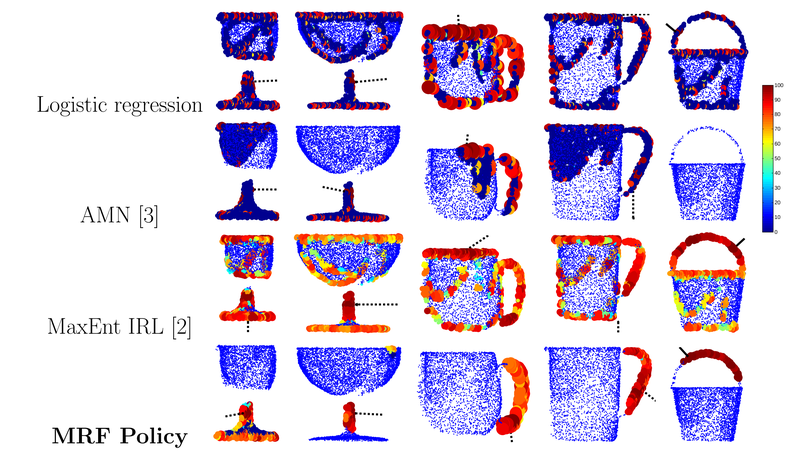
where $V_\theta^{\pi_{t:H}}(s) = \sum_i \theta_i \phi_i(s) + \gamma \sum_{s' \in \mathcal{S}} T_{\pi_t}(s, s') V_\theta^{\pi_{t+1:H}}(s')$.

- Parameters θ and λ are learned by maximizing the likelihood of the demonstration.
- An optimal policy $\pi^* \in \arg\max_{\pi \in \mathcal{A}^{|\mathcal{S}|}} P(\pi)$ is found by dynamic programing.

	$\lambda = 0$	$\lambda \in \mathbb{R}$
$\gamma = 0$	Logistic Regression	Associative Markov Networks (AMN) [3]
$\gamma \in \mathbb{R}$	Maximum Entropy IRL [2]	Markov Random Field Policies

5 Experiments

From a high-level point of view, grasping an object can be seen as an MDP with three time-steps: (1) **reaching** the object, (2) **preshaping** the hand, and (3) **grasping**. The following results show the learned values of the first time-step (reaching the object). Each point on an object corresponds to a reaching 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.



References

[1] Pieter Abbeel and Andrew Ng. Apprenticeship Learning via Inverse Reinforcement Learning. ICML 2004.

[2] Ziebart, B., Maas, A., Bagnell, A., and Dey, A. Maximum Entropy Inverse Reinforcement Learning. AAAI 2008.

[3] Taskar, B.: Learning Structured Prediction Models: A Large Margin Approach. PhD thesis, Stanford University, 2004.

NIPS 2012, Lake Tahoe, December 2012