Maximum Entropy Inverse Reinforcement Learning

Reza Ahmadzadeh

Comp5500 – Robot Learning

Notation

state s_i
action a_i
trajectory (state-action pairs) ζ
weight vector θ
state features $\boldsymbol{f}_{s_j}$

path feature $\boldsymbol{f}_\zeta = \sum_{s_j \in \zeta} \boldsymbol{f}_{s_j}$

Reward function structure

- A linear combination of weighted features

$$R(\boldsymbol{f}_\zeta) = \theta^\top \boldsymbol{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \boldsymbol{f}_{s_j}$$

Human Demonstrations (policies)

- Single trajectories $\tilde{\zeta}_i$

- Expected empirical feature count for **m** demonstrations

$$\tilde{\boldsymbol{f}} = \frac{1}{m} \sum_i \boldsymbol{f}_{\tilde{\zeta}_i}$$

Matching Feature Expectations

- Similar to previous paper (Abbeel & Ng, 2004)

$$\text{agent} \leftarrow \sum_{path\zeta_i} P(\zeta_i)\boldsymbol{f}_{\zeta_i} = \tilde{\boldsymbol{f}} \rightarrow \text{expert demo}$$

Issues of Previous Work

1. Each policy can be optimal for many reward functions
2. Many policies lead to the same feature counts

Contributions of this paper

- A probabilistic approach
- Uses principle of Maximum Entropy to prevent ambiguity in choosing a distribution over decisions
- Maximum Entropy can also deal with noise in the demonstrations

(the optimal policy from which a reward is learned)

- Matching feature counts using Maximum Entropy principle
- Methods for
  - Deterministic, non –deterministic, and stochastic path distribution
  - Learning from Demonstrated Behavior
  - Efficient state frequency calculations

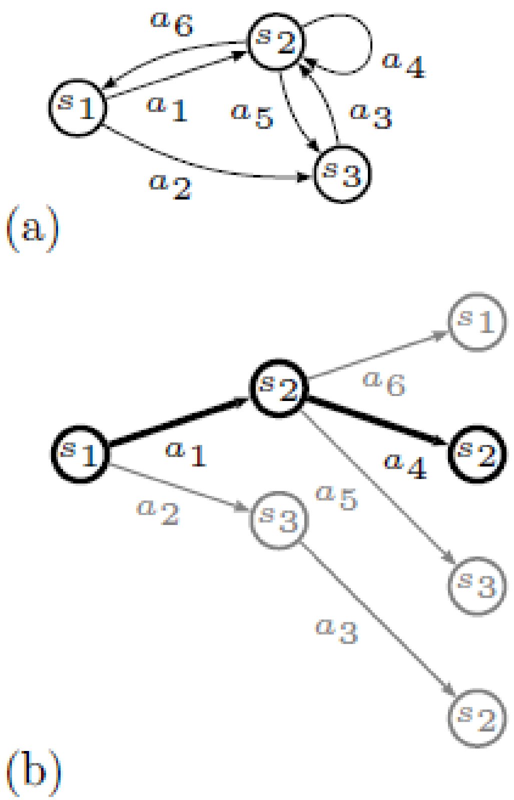
Deterministic Path Distributions

$$\text{agent} \leftarrow \sum_{path\,\zeta_i} P(\zeta_i)\boldsymbol{f}_{\zeta_i} = \tilde{\boldsymbol{f}} \rightarrow \text{expert demo}$$

$$P(\zeta_i|\theta) = e^{R(\theta)} = \frac{1}{Z(\theta)} e^{\theta^\top \boldsymbol{f}_{\zeta_i}}$$

Plans with higher rewards are exceptionally more preferred.

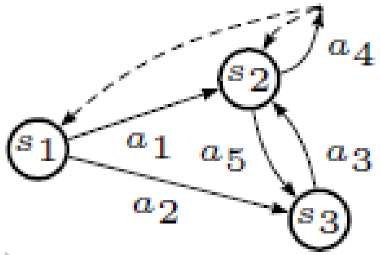
$$\text{normalization factor } Z(\theta) = \sum_{\zeta} e^{\theta^\top \boldsymbol{f}_{\zeta_i}}$$



Non-Deterministic Path Distributions

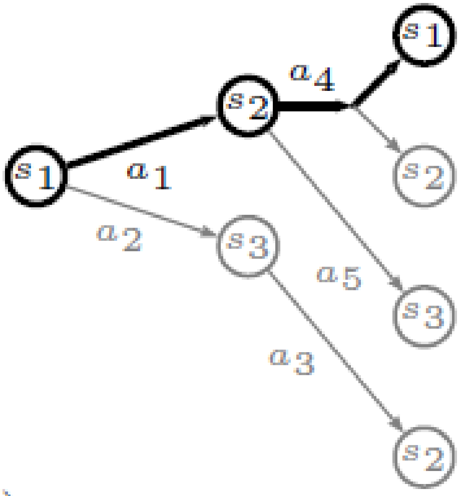
$$\text{agent} \leftarrow \sum_{path\,\zeta_i} P(\zeta_i)\boldsymbol{f}_{\zeta_i} = \tilde{\boldsymbol{f}} \rightarrow \text{expert demo}$$

$$P(\zeta_i|\theta, T) \approx \frac{e^{\theta^\top \boldsymbol{f}_\zeta}}{Z(\theta,T)} \prod_{s_{t+1},a_t,s_t \in \zeta} P_T(s_{t+1}|a_t, s_t)$$



(c)

The action outcome (selected by the agent) can randomly change,
Given the action the MDP is deterministic



(d)

Stochastic Policies

- A distribution over the available actions of each state
- Can be solved using gradient methods

$$P(a|\theta, T) \propto \sum_{\zeta : a \in \zeta_{t=0}} P(\zeta|\theta, T)$$

Learning from Demonstrated Behavior

You will be implementing this algorithm

Learning from Demonstrated Behavior

Algorithm

Input: Demonstrations, arbitrary θ

Output: optimal reward weights θ^* (make the reward $R(\theta) = \theta^\top \boldsymbol{f}_\zeta$)

1. Use the initial θ to build a reward
2. Find the optimal policy $\pi(a|s)$ w.r.t this reward using Value-Iteration
3. Find state visitation frequencies
4. Improve θ through maximizing the likelihood (using gradient ascent)
5. Jump to 2

Algorithm

Input: Demonstrations, arbitrary θ

Output: optimal reward weights θ^* (make the reward $R(\theta) = \theta^\top \boldsymbol{f}_\zeta$)

1. Use the initial θ to build a reward
2. Find the optimal policy $\pi(a|s)$ w.r.t this reward using Policy-Iteration
3. Find state visitation frequencies
4. Improve θ through maximizing the likelihood (using gradient ascent)
5. Jump to 2

Dynamic Programming (Background)

- Dynamic Programming refers to a set of algorithms for computing optimal policies given a perfect model of the environment represented by an MDP.
- One way to find an optimal policy is to calculate the optimal value function for the problem
- It is computationally expensive, but works very well for finite MDPs

For step 2 of the algorithm

Policy Evaluation

- Given a policy, find the value-function

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r + \gamma V_\pi(s')]$$

policy

Transition Probability

How do you implement this?

For step 2 of the algorithm

Iterative Policy Evaluation

Input: a policy, a small threshold for termination, initialize V(s) (must V(terminal states)=0)

Loop:

$\quad \Delta = 0$

$\quad$ Loop for each $s \in S$:

$\qquad v = V(s)$

$\qquad V(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r + \gamma V(s')]$

$\qquad \Delta = \max(\Delta, |v - V(s)|)$

until $\Delta < \epsilon$

For step 2 of the algorithm

Find a better Policy

- Now that you have an optimal value-function, you can find a better policy using

$$\pi(s) = argmax_a \sum_{s'} p(s'|s,a)[r + \gamma V_\pi(s')]$$

Implementation note

for $s \in S$

$$\pi(s) = argmax_a \sum_{s'} p(s'|a,s)[r + \gamma V(s')]$$

For step 2 of the algorithm

Algorithm

Input: Demonstrations, arbitrary θ

Output: optimal reward weights θ^* (make the reward $R(\theta) = \theta^\top \boldsymbol{f}_\zeta$)

1. Use the initial θ to build a reward
2. Find the optimal policy $\pi(a|s)$ w.r.t this reward using Policy-Iteration
3. Find state visitation frequencies
4. Improve θ through maximizing the likelihood (using gradient ascent)
5. Jump to 2

State Visitation Frequencies (SVF)

- **Algorithm 1** in the paper does not use Policy iteration
- The policy is calculated in the backward pass of **algorithm 1**
- We, instead, have used policy iteration, so we have the policy
- Therefore, we need only the forward pass of **algorithm 1**

For step 3 of the algorithm

State Visitation Frequencies (SVF)

- $D_t(s)$ is the probability of visiting state s at time t

$$
\begin{aligned}
&\text{for } s \in S \\
&\quad D_1(s) = p(s = s_1) \\
&\quad \text{for } t \in T \\
&\quad\quad D_{t+1}(s) = \textstyle\sum_a \sum_s D_t(s')\pi(a|s')p(s|s', a) \\
&\quad D(s) = \textstyle\sum_t D_t(s)
\end{aligned}
$$

Note: Unlike other algorithms,
In this algorithm s' is the previous state

For step 3 of the algorithm

Algorithm

Input: Demonstrations, arbitrary θ

Output: optimal reward weights θ^* (make the reward $R(\theta) = \theta^\top \boldsymbol{f}_\zeta$)

1. Use the initial θ to build a reward
2. Find the optimal policy $\pi(a|s)$ w.r.t this reward using Policy-Iteration
3. Find state visitation frequencies
4. Improve θ through maximizing the likelihood (using gradient ascent)
5. Jump to 2

Maximizing the Likelihood

Maximizing the entropy of the distribution over paths subject to the feature constraints from observed data

= Maximize the likelihood of the observed data under the maximum entropy distribution

$$\theta^* = argmax_\theta L(\theta) = argmax_\theta \sum_{examples} \log P(\tilde{\zeta}|\theta, T)$$

$$\nabla L(\theta) = \tilde{\boldsymbol{f}} - \sum_\zeta P(\zeta|\theta, T)\boldsymbol{f}_\zeta = \tilde{\boldsymbol{f}} - \sum_{s_i} D_{s_i}\boldsymbol{f}_{s_i}$$

For step 4 of the algorithm

Learning from Demonstrated Behavior

- This is a convex problem and can be solved using gradient-based methods
- D is the expected state visitation frequencies
- In practice, we measure the empirical sample-based expectation of the feature values not the true value

$$\nabla L(\theta) = \tilde{\boldsymbol{f}} - \sum_{\zeta} P(\zeta|\theta, T)\boldsymbol{f}_{\zeta} = \tilde{\boldsymbol{f}} - \sum_{s_i} D_{s_i}\boldsymbol{f}_{s_i}$$

$$\text{recall that } \tilde{\boldsymbol{f}} = \frac{1}{m}\sum_i \boldsymbol{f}_{\tilde{\zeta}_i}$$

For step 4 of the algorithm

Gradient ascent

- One step of the algorithm

$$\theta = \theta + \alpha \nabla L(\theta)$$

α is the learning rate

For step 4 of the algorithm

Algorithm

Input: Demonstrations, arbitrary θ

Output: optimal reward weights θ^* (make the reward $R(\theta) = \theta^\top \boldsymbol{f}_\zeta$)

1. Use the initial θ to build a reward
2. Find the optimal policy $\pi(a|s)$ w.r.t this reward using Policy-Iteration
3. Find state visitation frequencies
4. Improve θ through maximizing the likelihood (using gradient ascent)
5. Jump to 2

Programming Assignment 2

- You will be implementing MaxEnt (today's IRL algorithm)
- Deadline for submitting report and code: before 11:59pm 10/23/2019
- This assignment has 12.5% of your final grade
- Make sure to start early

Next Time (10/17/2019)

- No Class 10/15/2019
- Policy Search – Submit your review of paper [18]

  P. Kormushev, S. Calinon, and D. G. Caldwell. "**Robot motor skill coordination with EM-based reinforcement learning**". Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on. IEEE, 2010.

- Presentation of PI2 paper by Eric Zabele

  [19] P., Peter, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "**Skill learning and task outcome prediction for manipulation**." Robotics and Automation (ICRA), IEEE International Conference on. IEEE, 2011.