Maximum Entropy Inverse Reinforcement Learning

Presenter: Aditya Arjun

10-26-2021

Motivation: Reward Specification is Difficult

- Problem: Robotics tasks often complicated and difficult to hand-engineer rewards that provide learning signal
- **Inverse-RL** seeks to predict reward function using expert demonstrations
- Given reward function, can use standard RL techniques to solve

Inverse RL is also Difficult: Reward Ambiguity

- Expert policy could be optimal under variety of reward functions
- Given expert behavior might not be optimal under its reward function
- Hard to deal with case where no reward function makes demonstrated behavior both **optimal** and **better than alternate policies**

Key Idea: Principle of Maximum Entropy

- Handle ambiguity by using a probabilistic model using principle of maximum entropy to resolve ambiguities in choosing distributions
- Plans with equivalent rewards have equal probabilities, higher rewards exponentially preferred
- Other approaches may not make highest reward policy the most probable policy and may differently weight policies with same expected reward

Problem Setting

- Assume agent attempting to optimize reward function that is a **weighted linear combination** of features of each state (θ)
- Given trajectories (ζ) of states s_i and actions a_i taken by agent and MDP\R, learner must **find reward** that make demonstrations near-optimal
- Rediscovering original reward weights ambiguous problem

$$\mathbf{f}_\zeta = \sum_{s_j \in \zeta} \mathbf{f}_{s_j}$$

Feature Counts are Sum of Features over Trajectory

$$\text{reward}(\mathbf{f}_\zeta) = \theta^\top \mathbf{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \mathbf{f}_{s_j}$$

Reward for Trajectory is Weighted Linear Combination of Feature Counts

Related Work and Limitations

- Apprenticeship Learning via Inverse Reinforcement Learning (Abbeel and Ng 2004) – Matching feature counts is ambiguous because of issue where policy can be optimal under many different rewards
- Maximum Margin Planning (Ratliff, Bagnell and Zinkevich 2006) – Fails when no single reward function makes demonstrated behavior optimal and better than other policies (Ex. Imperfect agent)
- Bayesian Inverse Reinforcement Learning (Ramachandran and Amir 2007) – Action based approach which suffers from only comparing with paths locally at action level rather than all paths which branched earlier (label bias), which gives higher probability mass to paths with smaller branching factor (not necessarily optimal)

Approach: Use Maximum Entropy Distribution

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^\top \mathbf{f}_{s_j}}$$

Probability of Trajectory is Exponentiated Reward over Partition Function

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

Probabilities give Stochastic Policy for MDP

$$P(\zeta|\theta, T) = \sum_{o \in \mathcal{T}} P_T(o) \frac{e^{\theta^\top \mathbf{f}_\zeta}}{Z(\theta, o)} I_{\zeta \in o}$$

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

Tractable Approximation of Distribution over Paths using Transition Distribution

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

Goal: Optimize Log Likelihood of Demonstrated Agent Behaviors

Optimization Objective and Gradient

- Can use gradient-based optimization based on empirical and expected feature counts
- D_s (expected state visitation frequencies) difficult to calculate straightforwardly because of exponential growth of # paths with horizon

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

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

Gradient is Difference between Expected Empirical Feature Counts and Learner's Expected Feature Counts

Algorithm for State Frequency

- Approximates state frequencies by recursively backing up from terminal states
- Recursively compute probability mass of each branch by partition
- Calculate local action probabilities
- Determine state frequency counts at each time step
- Sum state frequency over all time steps to calculate desired value

Algorithm 1 Expected Edge Frequency Calculation

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_i,t}$

Experimental Setup

- Experiment on Real-World Data: Driver Route Modeling as **deterministic MDP**
- Road Network around Pittsburgh, Pennsylvania with 300,000 states and 900,000 actions
- 100,000 miles of travel over 3,000 hours of Yellow Cab taxi drivers driving
- Each trip specifies a slightly different MDP (because of different destination/goal state), but paper assumes that reward weight is independent of goal state
- Features: Road Type, Speed, Lanes, Transitions
- **Goal**: Determine reward function for taxi drivers to train policy to find route between location and goal destination
- Difficult Reward Function (Tradeoff between Time, Safety, Stress, Fuel Costs, etc.)

Baselines + Metrics

- Baselines
  - Maximum Margin Planning (MMP) (Ratliff, Bagnell and Zinkevich 2006)
  - Action-based Distribution Model (Action) used in Bayesian IRL and hybrid IRL
  - Time-based Model
- Metrics
  - Percentage Route shared between Demonstrated Data and Most Likely MDP Path
  - Percentage of Demonstrated Data with >90% Match with Model's Predicted Path
  - Average Log Probability of Demonstrated Trajectories (Not Relevant for all Baselines)

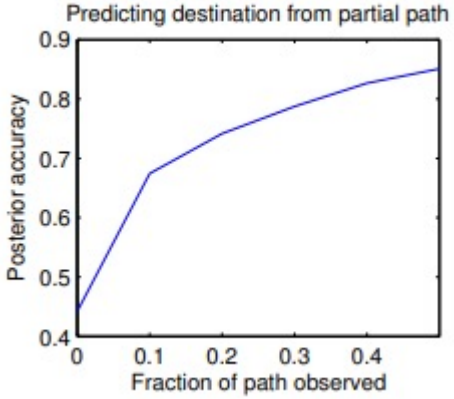
Experimental Results

- MaxEnt Model Performs better than all proposed baselines on all given metrics
- Log Prob only relevant to Action Model/Max Ent because other approaches do not calculate trajectory probabilities

	Matching	90% Match	Log Prob
Time-based	72.38%	43.12%	N/A
Max Margin	75.29%	46.56%	N/A
Action	77.30%	50.37%	-7.91
Action (costs)	77.74%	50.75%	N/A
MaxEnt paths	**78.79%**	**52.98%**	**-6.85**

Further Applications from Experiment

Figure 4: Destination distribution (from 5 destinations) and remaining path distribution given partially traveled path. The partially traveled path is heading westward, which is a very inefficient (i.e., improbable) partial route to any of the eastern destinations (3, 4, 5). The posterior destination probability is split between destinations 1 and 2 primarily based on the prior distribution on destinations.



Discussion of Results

- Algorithm performs excellently given use case with complex/noisy rewards
- Exceeds performance of other IRL approaches on all metrics
- Potential Applications of System beyond Route Recommendations
    - Predict user action using inference algorithm
    - Easy to infer destination given portion of path taken so far and offer advice/route adjustments if needed

Limitations

- Problem statement assumes reward function linear in features
  - Ex. Robotics tasks might use non-linear Euclidean distance between objects as reward
  - Could fix with complex designed features that capture needed information, but this requires work
- General IRL Issue: Requires known dynamics for MDP

Future Work

- Use deep neural representation of reward function rather than linear version to allow for more complex nonlinear rewards
- Extension of ideas to more domains with not necessarily optimal expert behavior, as learning from video demonstration to leverage new sources of data

Extended Readings

- A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress (2020) (https://arxiv.org/pdf/1806.06877.pdf)
- Maximum Entropy Deep Inverse Reinforcement Learning (2016) (https://arxiv.org/pdf/1507.04888.pdf)
- Efficient Sampling-Based Maximum Entropy Inverse Reinforcement Learning with Application to Autonomous Driving (2020) (https://arxiv.org/pdf/2006.13704.pdf)

Summary

- **IRL Problem**: Extract reward function that maximizes likelihood of demonstrated expert data such that other policies have low-probability trajectories
- Difficult because may not be a single deterministic reward that makes expert probable and other policies improbable (especially if expert data is noisy/suboptimal)
- By incorporating maximum entropy, proposed approach can learn to deal with cases where policy is optimal under many different rewards
- Authors able to apply maximum entropy principle to a noisy expert dataset of Taxi Drivers route preferences and achieve better performance compared to standard IRL approaches