

Maximum Entropy Inverse Reinforcement Learning

Ziebart, Maas, Bagnell, Dey

Presenter: Naireen Hussain

Overview

- What is Inverse Reinforcement Learning (IRL)?
- What are the difficulties with IRL?
- Researchers' Contributions
- Motivation of Max Entropy Set Up
 - Problem Set-Up
- Algorithm
- Experimental Set-Up
- Discussion
- Critiques and Limitations
- Recap

What is inverse reinforcement learning?

Given access to trajectories generated from an expert, can a reward function be learned that induces the same behaviour as the expert?

- a form of imitation learning

How is this different than the previous forms of RL we've seen before?

What is inverse reinforcement learning?

“forward” reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

reward function $r(\mathbf{s}, \mathbf{a})$

learn $\pi^*(\mathbf{a}|\mathbf{s})$

inverse reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

samples $\{\tau_i\}$ sampled from $\pi^*(\tau)$

learn $r_\psi(\mathbf{s}, \mathbf{a})$

 reward parameters

...and then use it to learn $\pi^*(\mathbf{a}|\mathbf{s})$

Difficulties with IRL

Ill posed problem – no unique set of weights describing the optimal behaviour

- Different policies may be optimal for different reward weights (even when it's all zeros!)
- Which policy is preferable?
- Match feature expectations [Abbeel & Ng, 2004]
 - No clear way to handle multiple policies
- Use maximum margin planning [Ratliff, Bagnell, Zinkevich 2006]
- Maximize margin between reward of expert to the reward of the best agent policy plus some similarity measure
- Suffers in the presence of an **sub - optimal** expert, as no reward function makes the agent optimal and significantly better than any observed behaviour

Researcher's Contribution

- Created the Maximum Entropy IRL (MaxEnt) framework
- Provided an algorithmic approach to handle uncertainties in actions
- Efficient Dynamic Programming algorithm
 - case study of predicting driver's behaviour
 - prior work in this application was inefficient [Liao et al, 2007]
 - **largest** IRL experiment in terms of data set size at the time (2008)

Why use Max Entropy?

- Principle of Max Entropy [Jaynes 1957] – demonstrates that the best distribution over current information is one with the largest entropy

$$H(\zeta|\theta) = - \sum_{\zeta} P(\zeta|\theta) \log(P(\zeta|\theta))$$

- Prevents issues with label bias
 - Portions of state space with many branches will each be biased to being less likely, and while areas with fewer branches will have higher probabilities (locally greedy)
 - The consequences of label bias is:
 - 1) the most rewarding path being not the most likely
 - 2) two different but equally rewarded paths with different probability

Problem Set-Up

- Agent is optimizing a reward function that linearly maps the features of each state \mathbf{f}_s in the path ζ to a state reward value.
- Reward is parameterized by the weights θ :

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

- Expected empirical feature counts based on m demonstrations :

$$\tilde{\mathbf{f}} = \frac{1}{m} \sum_i \mathbf{f}_{\zeta_i}$$

Algorithm Set-Up

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

- Reward function uses a Boltzmann distribution
- Above formulation assumes **deterministic** MDP's

ζ - path (must be finite for $Z(\theta)$ to converge, or use discounted rewards for infinite paths)

θ - reward weights

$Z(\theta)$ - partition function, normalization value

Algorithm Set-Up

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

Observations here are introduced to make the stochastic MDP deterministic given previous state distributions

- Two further simplifications are made:
- The partition function is constant for all outcome samples
 - Transition randomness doesn't affect behaviour

o – outcome sample

T – Transition distribution

Maximum Likelihood Estimation

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

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

- Use the maximize likelihood of observing expert data for θ as the cost function for θ
- **convex** for deterministic MDPs
- intuitively can be understood as difference in agent's empirical feature counts, and the expert's expected feature counts
 - Used sample based approach to compute expert's feature counts

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

- 1) Start from a terminal state
- 2) Compute the partition function at each state and action to obtain local action probabilities
- 3) Compute state frequencies at each time step
- 4) Sum over agent's state frequency all time steps
- 5) **This is similar to value iteration!**

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

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_{s_i} D_{s_i} \mathbf{f}_{s_i}$$

Experimental Set-Up

- The researchers were trying to investigate if a reward function for predicting driving behaviour could be recovered.
- Modelled road network as an MDP
- Due to different start and end positions, each trip's MDP is slightly different
 - Because of differing MDP's reward weight are treated as independent of the goal, so a single set of weights θ can be learned from many different MDP's

Dataset Details

- Collected driving data of 100,000 miles spanning 3,000 driving hours for Pittsburgh
- Fitted GPS data to the road network, to generate ~13,000 road trips
- Discarded noisy trips, or trips that were too short (less than 10 road segments)
 - This was done to speed up computation time

Path Features

Four different road aspects considered:

- **Road type:** interstate to local road
- **Speed:** high speed to low speed,
- **Lanes:** multi-lane or single lane
- **Transitions:** straight, left, right, hard left, hard right

There was a total of 22 features used to represent this state

Results

Model	% Matching	% >90% Match	Log Prob	Reference
Time- Based	72.38	43.12	N/A	n/a
Max Margin	75.29	46.56	N/A	[Ratliff, Bagnell, & Zinkevich, 2006]
Action	77.30	50.37	-7.91	[Ramchandran & Amir 2007]
Action (Cost)	77.74	50.75	N/A	[Ramchandran & Amir 2007]
MaxEnt	78.79	52.98	-6.85	[Zeibart et al. 2008]

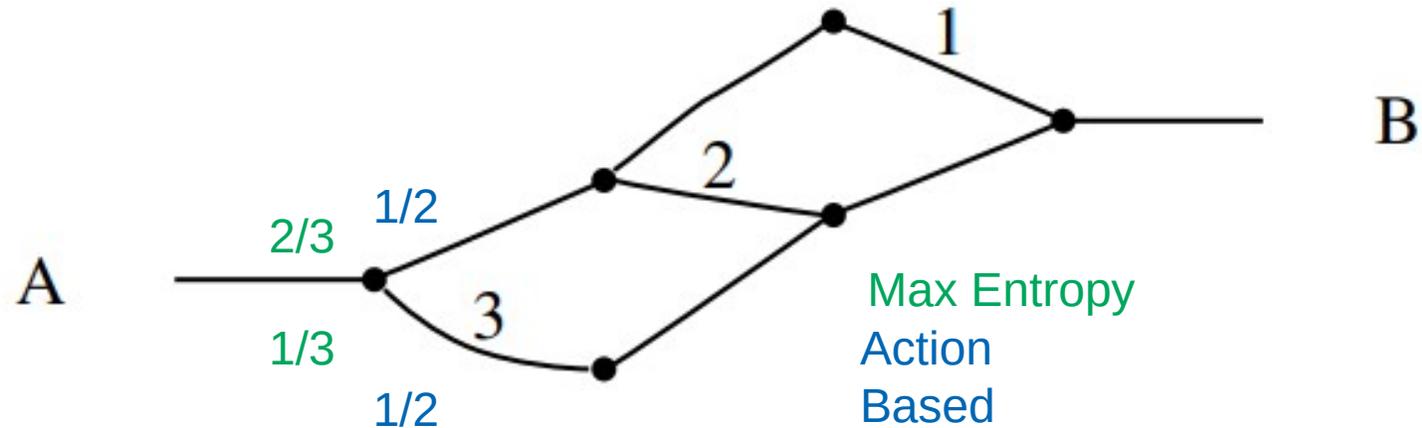
Time Based: Based on expected travel time, weights the cost of a unit distance of road to be inversely proportional to the speed of the road

Max Margin: maximize margin between reward of expert to the reward of the best agent policy plus some similarity measure

Action: Locally probabilistic Bayesian IRL model

Action (cost): – lowest cost path from the weights predicted from the action model

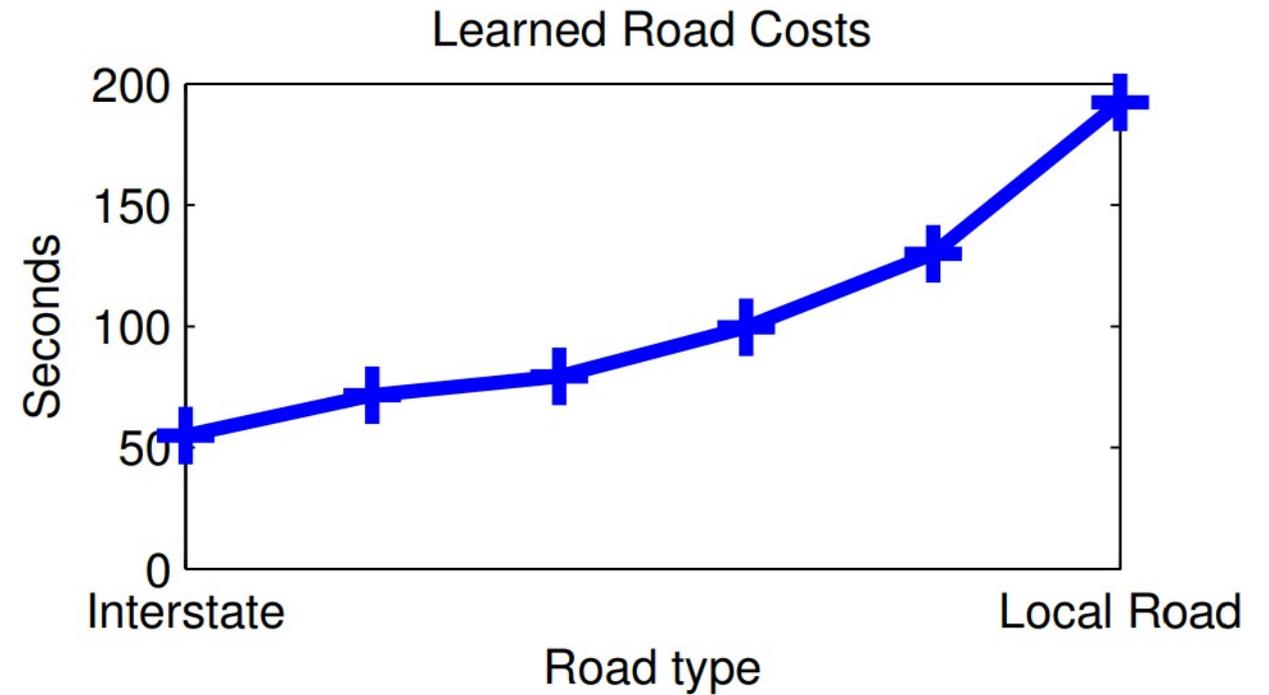
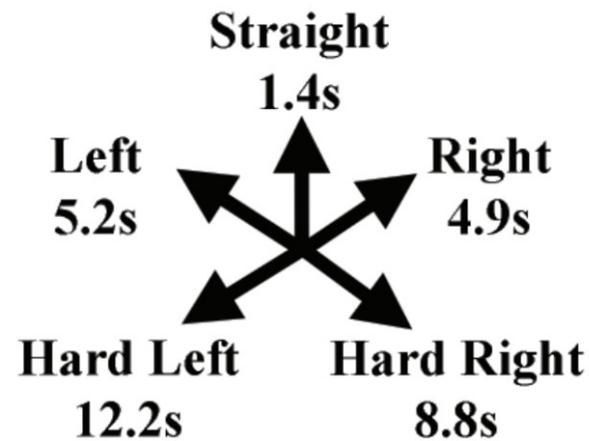
Discussion



- Ability to remove label bias which is present in locally greedy action based distributional models
- MaxEnt gives all paths equal probability due to equal reward
- Action based paths (weighted on future expected rewards) look only locally to determine possible paths
 - $P(A \rightarrow B) \neq P(B \rightarrow A)$

Discussion

The model learns to penalize slow roads and trajectories with many short paths



Discussion

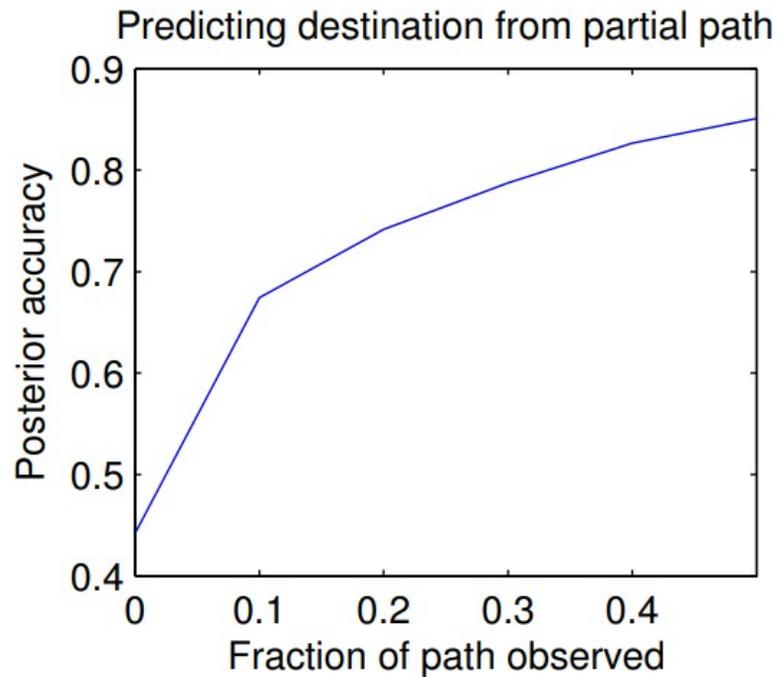
It is possible to infer driving behaviour from partially observable paths with Bayes' Theorem

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)}$$

$$\begin{aligned} P(dest|\tilde{\zeta}_{A \rightarrow B}) &\propto P(\tilde{\zeta}_{A \rightarrow B}|dest)P(dest) \\ &\propto \frac{\sum_{\zeta_{B \rightarrow dest}} e^{\theta^\top \mathbf{f}_\zeta}}{\sum_{\zeta_{A \rightarrow dest}} e^{\theta^\top \mathbf{f}_\zeta}} P(dest) \end{aligned}$$

Discussion

- Possible to infer driving behaviour from partially observable paths



- Destination 2 is far less likely than Destination 1 due to Destination 1 being far more common in the data-set.

Critique / Limitations / Open Issues

- Tests for inferring goal locations were done with only 5 destination locations
 - Easier to correctly predict the goals if they're relatively spread out vs clustered close together
- Relatively small feature space
- Assumes the state transitions are known
- Assumes linear reward function
- Requires hand crafted state features
- Extended to a Deep Maximum Entropy Inverse Learning model [Wulfmeier et al, 2016]

Contributions (Recap)

Problem

How to handle uncertainties in demonstrations due to sub-optimal experts and how to handle ambiguity with multiple reward functions.

Limitations of Prior Work

Max. Marginal prediction is unable to be used for inference (predict probability of path), or handle sub-optimal experts. Previous action based probabilistic models that could handle inferences suffered from label biases.

Key Insights and Contributions

MaxEnt uses a probabilistic approach that maximizes the entropy of the actions, allowing a principled way to handle noise, and it prevents label bias. It also provides an efficient algorithm to compute empirical feature count, leading to state of the art performance at the time.