Inverse Reinforcement Learning

Deep Maximum Entropy Model

Pulkit Khandelwal

SOCS, McGill University

What is Inverse Reinforcement Learning?

As introduced by Russell ’98:
Given

- Measurements of an agent’s behavior over time, in a variety of circumstances
- if needed, measurements of the sensory inputs to that agent
- if available, a model of the environment

Determine
the reward function being optimized
Sources of motivation?

- Animal and human behavior: example- bee foraging
- Construct an intelligent agent which performs well in a particular domain

Inverse Reinforcement Learning: Linear Programming Problem Formulation

So, given an observed behavior, we are tasked to determine the reward function which best explains the behavior.
Any Issues?
There might be several functions which satisfy this. This is thus an ambiguity! We can get a set of reward functions.

Ng and Russell ’00 proposed a simple heuristic- to solve an LP formulation:

Inverse Reinforcement Learning: Linear Programming Problem Formulation [Contd.]

Let $\pi(s) = a_1$

maximize $\sum_{i=1:N} min_{\in\{a_2,.....,a_k\}}\{(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1}\} - \lambda||R||_1$

such that:

$(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1} \succeq 0$

$\forall a \in A \setminus a_1$

with the condition that,

$|R_i| \leq R_{max}$, i=1,2,.....,N

A simple Proof?

Inverse Reinforcement Learning: Linear Programming Problem Formulation [Contd.]

- The previous formulation was for Small state spaces
- Large state spaces?

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + ... + \alpha_d\phi_d(s) \tag{1}$$

Also,

$$V^{\pi} = \alpha_1 V_1^{\pi} + \alpha_2 V_2^{\pi}(s) + ... + \alpha_d V_d^{\pi} \tag{2}$$

- $\therefore$,
  maximize
  $\sum_{s \in S_o} min_{\in \{a_1, a_2, ..., a_k\}} f(E_{s' \sim P_{sa_1}}(V^{\pi}(s')) - E_{s' \sim P_{sa}}(V^{\pi}(s')))$
  such that:
  $|\alpha_i| \leq 1$, i=1,2,.....,d

Apprenticeship Learning via IRL [Abbeel and Ng '04]

- Learning from demonstration by an expert. Given some expert trajectories, find the reward function!
- ***Key idea*** Try to match feature expectations and thereby recover a reward function (which actually might not be the true reward function)
-

$$E_{s_o \sim D}[V^{\pi}(s^{'})] = E[\sum_{t=0}^{\infty} \gamma^t R(S_t)|\pi] \tag{3}$$

$$= E[\sum_{t=0}^{\infty} \gamma^t w.\phi(s_t)|\pi] \tag{4}$$

$$= w.E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \tag{5}$$

- To find the feature expectation:

$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]$
Then find the reward function!

Maximum Entropy Model [Reibart et. al '08]

- Feature counts for each path: $f_\zeta = \sum_{s_j \in \zeta} \mathbf{f}_{s_j}$
- Hence, reward can be expressed as:
  $\text{reward}(f_\zeta) = \theta^T f_\zeta$
- Now, expected feature counts over all the trajectories $\bar{f} = (1/m) \sum_i f_\zeta$
- *Proposed:* Matching feature expectations:
  $\bar{f} = \sum_{\zeta_i} P(\zeta_i) \mathbf{f}_{\zeta_i}$
- Though, an **ambiguity**:
  - Each policy can be optimal for many reward functions
  - Many policies can lead to same feature counts
- Therefore, proposes and alternate distribution, based on an entropy formulation:
  $P(\zeta_i) = (1/Z(\theta)) e^{\theta^T \mathbf{f}_{\zeta_i}}$

Maximum Entropy Model: Algorithm

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

Deep Maximum Entropy Model

Now represent the reward function as a non-linear function instead of linear features!

$$R(s) = \alpha\phi(s) \tag{6}$$

$$R(s) = \alpha\sigma(W.\phi(s)) \tag{7}$$

This can be done for more hidden units and more hidden layers!

Some Results

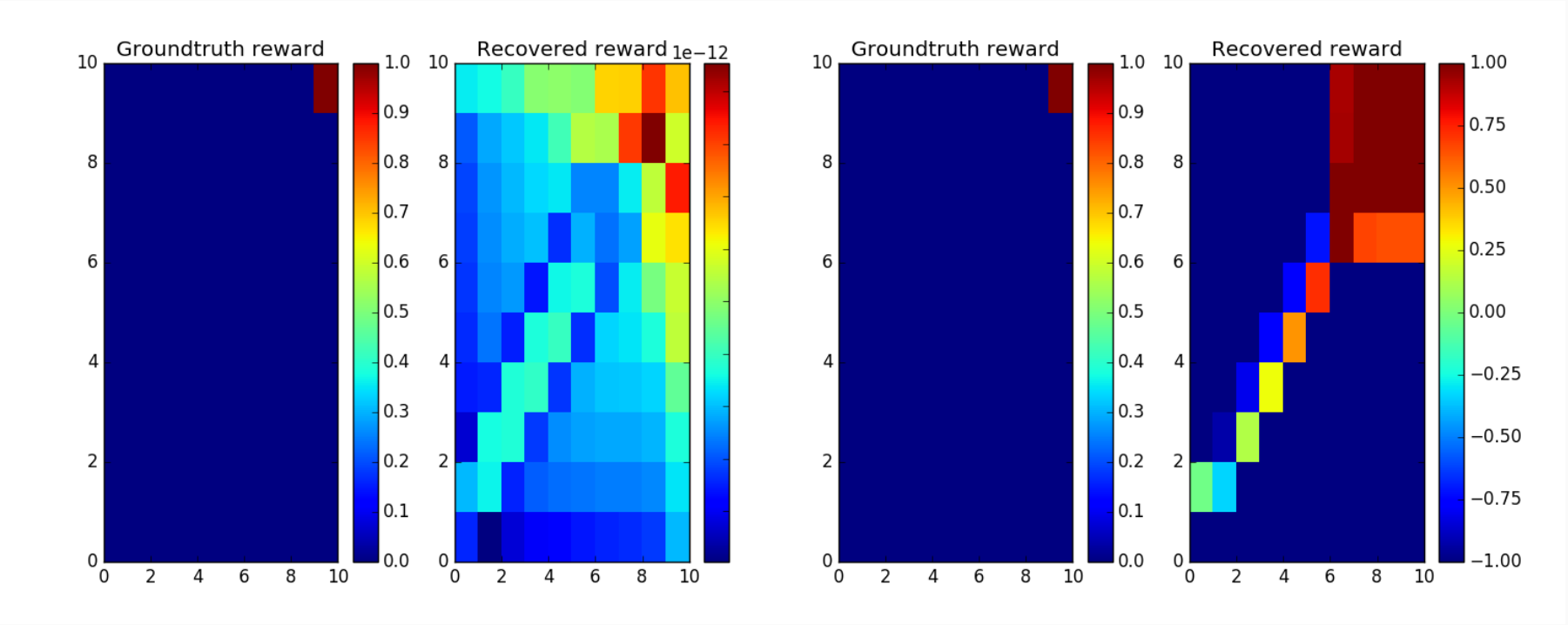
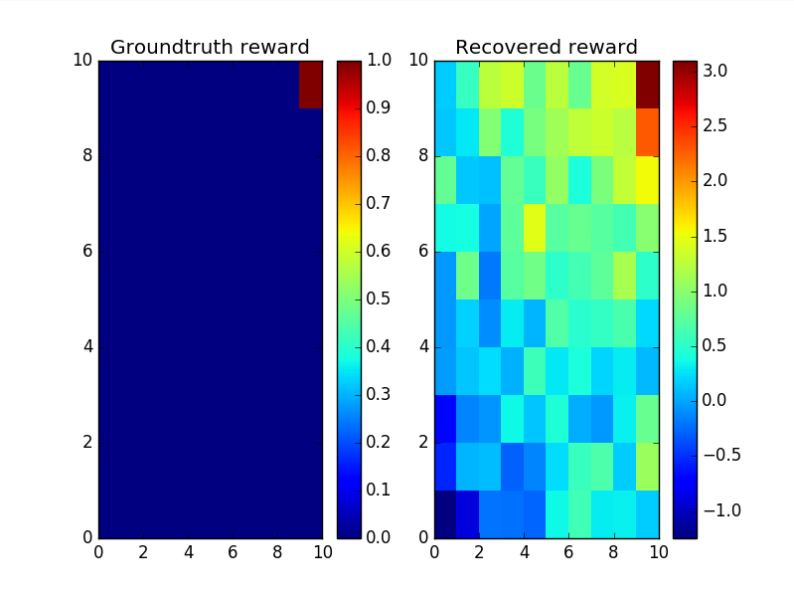
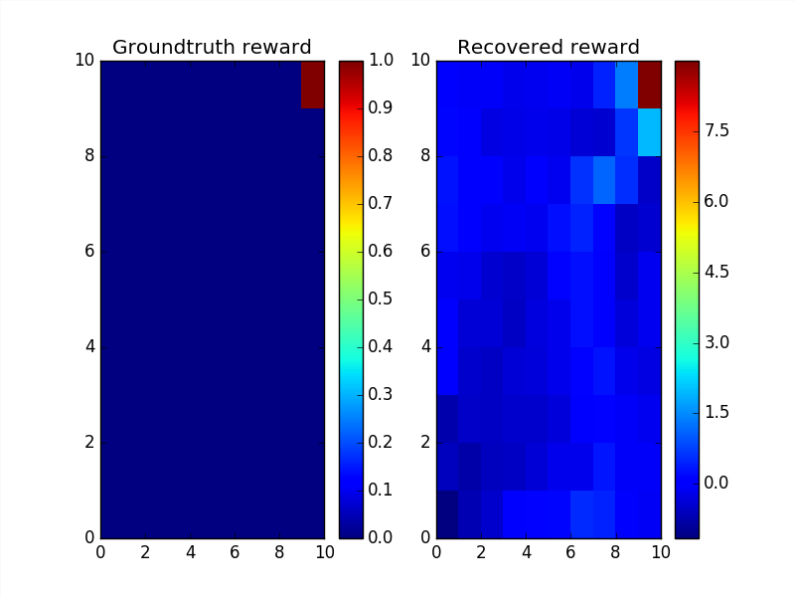


Figure 1: (a) Small state space and (b) Large state space; Linear Programming Recovered Reward

Some Results [Contd.]



(a)



(b)

Figure 2: (a) Maximum Entropy and (b) Deep Maximum Entropy; Recovered Reward; 10X10 grid; Deep NN: 10 hidden layers with 10 units each; γ = 0.9 and number of epochs = 20