Statistical Techniques in Robotics (16-831, S22) Lecture #25 (Wednesday, April 20)

IRL as Entropy Maximization (Extra Credit)

Lecturer: Kris Kitani *Scribes: Xuanbai Chen, Jia Shi*

1 Review

We are covering the inverse reinforcement learning algorithms in the last several lectures. In the last lecture, we mainly focused on the Max Margin IRL. We will firstly review it.

1.1 Max Margin IRL

In this section, we mainly introduce and review the Max-Margin IRL algorithm [1]as a quadratic programming problem for IRL. In this algorithm, it is not guaranteed to recover a true reward function, however, it will find an expert optimal policy. The agent can observe expert behavior from which will attempt to recover the unknown reward. This setting assumes that such reward can be expressed as linear combination of known "features". Initially speaking, we can judge the value function of experts optimal one is larger than the estimated one. So it can be concluded as:

$$V(\pi) \leq V(\pi^*)$$

Then when assuming a linear reward function, we can express the optimization problem for Max-Margin IRL as:

$$\begin{aligned} &\max_{\theta} \ t \quad \text{where } t \text{ is a margin variable} \\ &\quad \text{s.t. } \boldsymbol{\theta}^T \boldsymbol{\mu}(\pi^*) \geq \boldsymbol{\theta}^T \boldsymbol{\mu}(\pi) + t \quad \forall \pi \end{aligned} \tag{1}$$

Though it is formed well, in this point, there are mainly two problems need to be solved.

1. No bound on the reward parameters !
2. Can't compare against all policies !

While for the former problem, we can leverage the regularization method to solve. For the latter, we can leverage reinforcement learning to find most competitive policies.

By using this approach, the regularized objective thus becomes:

$$\begin{aligned} &\max_{\theta} \ t \\ &\quad \text{s.t. } \boldsymbol{\theta}^T \boldsymbol{\mu}(\pi^*) \geq \boldsymbol{\theta}^T \boldsymbol{\mu}(\pi) + t \quad \forall \pi \\ &\quad\quad ||\boldsymbol{\theta}||_2 \leq 1 \end{aligned} \tag{2}$$

The last line is a L2 Norm constraint and we can further re-write the objective as a minimization as follows:

$$\begin{aligned} \min_{\theta,t} \quad & \lambda||\boldsymbol{\theta}||_2 - t \\ \text{s.t.} \quad & \boldsymbol{\theta}^T(\boldsymbol{\mu}(\pi^*) - \boldsymbol{\mu}(\pi)) \geq t \quad \forall \pi_n \in \Pi \end{aligned} \tag{3}$$

The last line means that it owns finite set of competitive policies. We can observe that it is similar with the objective function of SVM max-margin classifier:

$$\begin{aligned} \min_{w,\xi} & ||\boldsymbol{w}||_2 + C\sum_i \xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \end{aligned} \tag{4}$$

After observing the Equation 3 and 4, we can find the IRL can be framed as a max-margin classifier. So similar to SVM, the issue can also be solved by online gradient descent with quadratic programming. We show the Algorithm 1.1 below:

Algorithm 1 Max-Margin IRL ($\mu(\pi^*)$)

```
0: for $i = 1, \dots n$ do
0:     $\hat{\pi} = \text{argmax}_\pi \boldsymbol{\theta}^T\boldsymbol{\mu}(\pi)$        Competitive policy: Solve RL with linear reward function
0:     $\boldsymbol{\mu}(\hat{\pi}) = \mathbb{E}_{p_0,\mathcal{T},\hat{\pi}}\big[\sum_{t=0}^{\infty}\gamma^t\phi(s_t)\big]$        Compute expected feature counts of competitive policy (MC
   roll outs)
0:     $\boldsymbol{\theta} = \text{QuadProg}(\lambda, \boldsymbol{\mu}(\pi^*), \{\boldsymbol{\mu}(\hat{\pi})\}_{n=1}^{i})$
0: end for
   return $\boldsymbol{\theta}$ =0
```

1.2 Structured Output Max Margin IRL

In this section, we will mainly review the Structured Output Max-Margin IRL [2]. Instead of leveraging a margin of 1 for minimization objective derived like Eq. 4 in the Max-Margin IRL algorithm, this algorithm uses the notion of a variable margin, which depends on a distance between policies and formed it as a loss:

$$\begin{aligned} \min_{\theta,\xi} \quad & \lambda||\boldsymbol{\theta}||_2 + \sum_i \xi_i \\ \text{s.t.} \quad & \boldsymbol{\theta}^T(\boldsymbol{\mu}(\pi^*) - \boldsymbol{\mu}(\pi)) \geq l(\pi^*, \pi_i) - \xi_i \quad \forall i \end{aligned} \tag{5}$$

$l(\pi^*, \pi)$ is the variable margin. The intuition is that if the polices are very different, policy values should be very different. Examples of these variable loss are: 1.number of disagreements between policies; 2.physical distance between trajectories resulting from policies.

And after using Max-Margin Planning, the Equation can be changed into below:

$$\min_\theta \frac{1}{2}||\boldsymbol{\theta}||_2 + \lambda\sum_d \big\{(\boldsymbol{\theta}^T F_d + l_d^T)\eta - \boldsymbol{\theta}^T F_d \eta_d\big\} \tag{6}$$

where $l(\pi^*, \pi_d) = l_d^T \eta^*$ is the variable margin for MMP (structural loss) and we assume a linear relationship between occupancy (eta) and structural loss vector. Therefore, the final objective for MMP is given by:

$$\min_{\theta} \frac{1}{2}||\boldsymbol{\theta}||^2 + \lambda \sum_{d} \left\{ (\boldsymbol{\theta}^T F_d + l_d^T)\eta_d^* - \boldsymbol{\theta}^T F_d \eta_d \right\} \tag{7}$$

The final two versions of the Max-Margin Planning: MMP-batch and MMP-online are shown below:

Algorithm 2 Max-Margin Planning-Batch ($\{$ F$_d, \eta_d, l_d\}_{d=1}^{D}, \lambda, \alpha, T$)

```
0: $\boldsymbol{\theta} \leftarrow 0$
0: **for** $i = 1, \ldots, T$ **do**
0:   $\pi_d = \mathrm{RL}(\boldsymbol{\theta}^T F_d + l_d^T) \quad \forall d$
0:   $\eta_d = \textsc{CountVisitation}((\pi_d)) \quad \forall d$
0:   $g = \textsc{ComputeSubgrad}((\boldsymbol{\theta}, F_d, l_d, \eta_d)) \quad \forall d$
0:   $\boldsymbol{\theta}_n = \boldsymbol{\theta}_n - \alpha_t \cdot g_n \quad \forall n$
0: **end for**
   **return** $\boldsymbol{\theta}$ =0
```

Algorithm 3 Max-Margin Planning-Online ($\{$ F$_d, \eta_d, l_d\}_{d=1}^{D}, \lambda, \alpha, T$)

```
0: $\boldsymbol{\theta} \leftarrow 0$
0: **for** $i = 1, \ldots, T$ **do**
0:   $\pi_d = \mathrm{RL}(\boldsymbol{\theta}^T F_d + l_d^T)$
0:   $\eta_d = \textsc{CountVisitation}((\pi_d))$
0:   $g = \textsc{ComputeSubgrad}((\boldsymbol{\theta}, F_d, l_d, \eta_d))$
0:   $\boldsymbol{\theta}_n = \boldsymbol{\theta}_n - \alpha_t \cdot g_n \quad \forall n$
0: **end for**
   **return** $\boldsymbol{\theta}$ =0
```

2 Summary

2.1 Entropy Maximization IRL

2.1.1 Information Entropy

The Shanon Entropy [4]is defined as following, which is use to measure the randomness of a random variable.

$$H(x) = -\sum_{x} p(x) log p(x)$$

If we are trying to estimate a distribution, without any constrain or information, uniform distribution would be a reasonable assumption.

According to the principle of maximum entropy, given a set of distributions that explain a set of data equally well, the best choice is the one with highest entropy according to Occam's razor. For uniform distribution, all value have the same probability, thus it have high entropy.

2.1.2 Problem setup

We are given a sample of trajectory sampling from prior expert demonstrations with distribution of $\tilde{p}(\zeta)$, and are trying to estimate a distribution $p(\zeta)$ which have close distribution to $\tilde{p}(\zeta)$.

Since it's infeasible to match all value in those two distributions, we choose to minimize the distance between the expected feature counts, which is a close approximation of the distribution. Thus we can formulate the following constrain

$$\sum_{\zeta} p(\zeta)\mu(\zeta) = \sum_{d} \tilde{p}(\zeta_d)\mu(\zeta_d)$$

The estimated distribution should also align with the maximum entropy theory. Thus we can maximizing the entropy by minimize the negative entropy(by removing the negative sign in H(x)). The second constrain are

$$min_p \sum_{\zeta} p(\zeta) log p(\zeta)$$

Finally, our distribution should under probability constrain that the sum of all possible trajectory sum up to 1. Thus we have

$$\sum_{\zeta} p(\zeta) = 1$$

Formally, we can formula those three constrain in objective function in a Lagrangian form:

$$L(p,\theta,V) = \underbrace{\sum_{\zeta} p(\zeta) log p(\zeta)}_{\text{Shannon Entropy}} + \underbrace{\theta^T \Big\{ \sum_{d} p_u(\zeta_d)\mu(\zeta_d) - \sum_{\zeta} p(\zeta)\mu(\zeta) \Big\}}_{\text{Feature matching}} + \underbrace{V \Big\{ \sum_{\zeta} p(\zeta - 1) \Big\}}_{\text{Probability constraint}}$$

This is a constrain optimize problem for which V and θ are Lagrangian multiplier.

After solving the above function by finding extrema, we can get

$$p(\zeta) = \frac{exp(\theta^T \mu(\zeta))}{V+1}$$

This is a Boltzmann distribution for which $p(\zeta) \propto exp(\theta^T \mu(\zeta))$, which is the optimal parametric form of the trajectory likelihood to solve the problem.

If we are taking the gradient of the above likelihood respect to the Lagrangian multiplier θ, we have

$$\nabla L(\theta) = \sum_{d} p_u(\zeta_d)\mu(\zeta_d) - \sum_{\zeta} p_\theta(\zeta)\mu(\zeta) = \bar{\mu}_d - \bar{\mu}$$

$\bar{\mu}_d$ and $\bar{\mu}$ are the expected feature counts of the experts and the learned policy. Then we can use gradient descent to make update

$$\theta = \theta - \lambda(\bar{\mu}_d - \bar{\mu})$$

In this cause, $\bar{\mu}$ is an unknown value and we need to compute it by solving an reinforcement learning problem. The pipeline of the algorithm is shown below.

Algorithm 4 MaxEnt IRL (λ, π_*)

```
0: $\bar{\mu}_D = CountFeatures(\mathcal{E}, \pi^*)$
1: while $\nabla L(\theta) \geq \epsilon$ do
1:     $\pi$ = SoftValueIteration$(\mathcal{E}, \theta)$
1:     $\bar{\mu}$ = CountFeatures$(\mathcal{E}, \theta)$
1:     $\theta = \theta - \lambda(\bar{\mu}_d - \bar{\mu})$
2: end while
   =0
```

In line 2, we precompute the average faeture count of the expert policy. In line 4, we are estentially solving a maximum entropy RL problem use value iteration to compute the policy. In line 5, we can use the computed policy to interact with environment $\mathcal{E}$ to count the average feature of learned policy. With the learned feature, we can use gradient descent to update the parameter θ

2.1.3 Activity Forecasting

One application of the IRL is the Activity Forecasting. This problem is trying to predict the distribution of pedestrian walking path from one place to another. The algorithm is essential take the pedestrian trajectory as demonstrated activity, and try to learn the reward function of pedestrian.

Formally, the activity sequence of pedestrian is generated by a MDP, we have

$$\zeta = x_0, a_0, r(x_0), x_1, a_1, r(x_1)..x_g, a_g, r(x_g)$$

The policy is model by the exponential function depends on the immediate reward and the expected future payoff. The policy determine the sequence above.

$$p(a|x) \propto exp\{ \underbrace{r(x)}_{\text{Reward}} + \underbrace{\sum_{x}^{'} p(x'|x,a)V(x')\}}_{\text{Expected future payoff}}$$

The V and Q function determine the policy above.

$$V(x) = softmax_a Q(x,a)$$

$$Q(x,a) = r(x) + \sum_{x}^{'} p(x'|x,a)V(x')$$

In the problem setup, the reward function is linear combination of enumerated perception feature weighted by Lagrangian parameter θ. $\phi(s)$ is step-wise state features.

$$r(s) = \theta^T \phi(s)$$

With reward function, we can apply it to a new environment to get the value function. With value function, we can obtain a policy the pedestrian is trying to use, and finally the path that the pedestrian is doing to walk can be sample from the policy, which essentially achieve the Activity Forecasting.

We can try to solve for θ with maximum entropy IRL method method in the last section.

2.2 Matrix Game IRL

In this section, we would like to explore the Matrix Game IRL [3]. At first, we would like to recall the value function defined as definition.

$$V^\pi(s) = \mathbb{E}_{p,\pi}\left[\sum_{k=0}^{\infty} \gamma^t r(s_t)\right]$$

After reviewing that, we will try to prove that $\mathbb{E}[V^\pi(s)] \to V(\pi)$. In order to do that, we can represent the reward function parameterized by θ.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{p,\pi}[\sum_{t=0}^{\infty} \gamma^t \theta \phi(s_t)] \\ &= \boldsymbol{\theta} \cdot \mathbb{E}_{p,\pi}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)] \\ &= \boldsymbol{\theta} \cdot \boldsymbol{\mu}^{\boldsymbol{\pi}}(\boldsymbol{s}). \end{aligned} \tag{8}$$

Now we want to get the an estimate of V^{π} from a trajectory, hence the expectation over the trajectory. The expectation also include initial state distribution

$$\begin{aligned}\mathbb{E}_{V^{\pi}(s)} &= \mathbb{E}[\sum_{t=0}^{\infty}\gamma^{t}\theta\phi(s_t)] \\ &= \boldsymbol{\theta}\cdot\mathbb{E}[\sum_{t=0}^{\infty}\gamma^{t}\phi(s_t)] \\ &= \boldsymbol{\theta}\cdot\boldsymbol{\mu}(\boldsymbol{\pi}).\end{aligned} \tag{9}$$

After rewriting the value of our expert policy to be better than any other policy, we can acquire this below:

$$\boldsymbol{\theta}\cdot\boldsymbol{\mu}(\boldsymbol{\pi})^{*} \geq \boldsymbol{\theta}\cdot\boldsymbol{\mu}(\boldsymbol{\pi}). \tag{10}$$

We always want to achieve the largest difference by comparing different policies, which will appear between the expert policy and other policies. In this time, we can write it as a minimum over the whole function with a maximum expected value of the next best policy:

$$\min_{\theta}\{\max_{\pi}\boldsymbol{\theta}\cdot\mu(\pi)-\boldsymbol{\theta}\cdot\mu(\pi_E)\}. \tag{11}$$

We can drag the parameter vector $\boldsymbol{\theta}$ out and make the transpose of it to get the following simplified expression,

$$\min_{\theta}\{\max_{\pi}\boldsymbol{\theta}^{\top}(\mu(\pi)-\mu(\pi_E))\}. \tag{12}$$

And then, we would like to form it by leveraging an alternative Interpretation: Vector of expected cumulative features written as matrix multiplication (Row Player, Game Matrix, and Column Player).

$$\min_{\theta}\{\max_{\pi}\boldsymbol{\theta}^{\top}\boldsymbol{G}\boldsymbol{\psi}\}. \tag{13}$$

This notation reveals that IRL is a two player game where $\boldsymbol{G}$ specifies the game and records the differences between policies, $\boldsymbol{\theta}$ specifies the row player, and $\boldsymbol{\psi}$ specifies a column player. After knowing that Matrix Game IRL can be represented as the two player game and it can be solved by the following two algorithms below:

Algorithm 5 MW-MatrixGame (β)

```
0: $\boldsymbol{w}^{(0)} \leftarrow \{w_r^{(0)} = 1\}$
0: for $t = 1, \ldots, T$ do
0:     $R^{(t)} = \frac{\boldsymbol{w}^{(t-1)}}{\sum_r w_r^{(t-1)}}$                row player strategy
0:     RECEIVE($\boldsymbol{l}^{(t)} = \mathrm{G}(\cdot, \boldsymbol{c}^{(t)})$)                Expected loss of game
0:     $w_r^{(t)} = w_r^{(t-1)}\beta^{l_r^{(t)}}$   $\forall r$                update row player strategy
0: end for
   return $\boldsymbol{\theta}$ =0
```

Observed from this algorithm, the third line describe the row player strategy; the fourth line shows the expected loss of game and game matrix and the column player strategy; the fifth line updates the row player strategy. When observing this algorithm, we can find that there is an exponential gradient descent in the fifth line. In this way, we can convert it to the Multiplicative Weights IRL Algorithm below.

Algorithm 6 Multiplicative Weights IRL ($\hat{\pi}_E, \hat{\mu}_E, \beta$)

```
0: for $t = 1, \dots, T$ do
0:     $\theta_i = \frac{\theta_i}{\sum_{i'} \theta_{i'}} \quad \forall i$
0:     $\hat{\pi}_\theta = \text{RL}(r(\cdot; \boldsymbol{\theta}))$
0:     $\hat{\mu} = \text{Estimate}(\pi_\theta)$
0:     $\theta_i = \theta_i \cdot \exp\{\ln \beta \cdot G_i(\hat{\mu}_i)\} \quad \forall i$
0: end for
   =0
```

We update our weights using line 2, use RL over our reward parameter to calculate some optimal policy $\hat{\pi}$ in line 3, use the optimal policy to calculate the expected feature count $\hat{\mu}$ in line 4, before updating with exponentiated gradient descent. Lines 3 and 4 can be understood as a 'prediction step' where the actual game is played.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[3] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

[4] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.