How can we improve Apprenticeship learning as DNN?

Jeong Gwan Lee

KAIST

(Korea Advanced Institute of Science and Technology)

Before I start,

In MDP, meaning of Reward function?

The reward function is ”the most succinct, robust, and transferable definition of a task.”[1]

[1]Abbeel, Pieter. 2008. Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control. Ph.D. thesis, Stanford University, Stanford, CA, USA.

Apprentice Learning,

is a learning approach, which assumes that the reward function is unknown, but instead, *expert knowledge* is available.

This *expert knowledge* is in the form of sequences of states and actions, trajectories, where the goal state is achieved.

The main idea is to find a optimal policy close to the *expert policy* π^E, to use these trajectories.

Inverse Reinforcement Learning(IRL)

Reward function is parameterized as linear combination of features,

$$R(s) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{s}) \quad s.t. \quad \phi : S \to [0,1]^k,$$

w is the weight vector to be learned,
ϕ is the features vector(basis function),
k is the number of features.

Feature expectations, denoted as μ,

$$\boldsymbol{\mu}(\boldsymbol{\pi}) = E[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(\boldsymbol{s_t})|\pi] \in \mathbb{R}^k.$$

Estimated feature expectations for expert policy,

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{E}} = \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(\boldsymbol{s}_t^{(i)}),$$

Learning process would be repeated until, $||\mu(\pi) - \hat{\mu}_E|| < \epsilon$

Batch, Off-Policy and Model-Free Apprenticeship Learning

IRL set-up

The true reward function belongs to some hypothesis space

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \ |\phi_i(s)| \leq 1, \forall s \in S, 1 \leq i \leq p.$$

$$R^*(s) = \boxed{(\theta^*)^T}\boxed{\phi(s)}$$

parameters, weights.

features; state representation; input.

$R(s) = f_N(\ldots(f_2(f_1(x, \theta_1), \theta_2), \ldots), \theta_N)$

f_i : DNN layer with activation function.

Reward estimator

$\phi(s) = f_N(\ldots(f_2(f_1(x, \theta_1), \theta_2), \ldots), \theta_N)$

f_i : DNN layer with activation function.

Feature expect. estimator

1)depending on input(videos or game display -> CNN, factory cases -> ?, gathering sensor input AMAP(temperature?)), 2)embedding space, 3) why R is related to s regardless of policy

IRL set-up

For any reward function belonging to $\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}$,
Value function V(s) can be expressed,

$$V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \theta^T \phi(s_t) | s_0 = s, \pi] = \theta^T E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi]$$

1.(reward esti.) DNN, Non-linearity, so, might not make feature expectation?

2. (feature extractor) $\phi(s_t)$ might be “the feature output just before softmax” in classification? → Depending on the input.

Feature expectation is,

$$\mu^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi]$$

$$|V^{\pi_E}(s_0) - V^{\tilde{\pi}}(s_0)| = |\theta^T(\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0))| \leq \|\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0)\|_2$$

IRL Algorithm

1. Starts with some initial policy $\pi^{(0)}$ and compute $\mu^{\pi^{(0)}}(s_0)$. Set $j = 1$;
2. Compute $t^{(j)} = \max_{\theta:\|\theta\|_2 \leq 1} \min_{k \in \{0, j-1\}} \theta^T(\mu^{\pi_E}(s_0) - \mu^{\pi^{(k)}}(s_0))$ and let $\theta^{(j)}$ be the value attaining this maximum. At this step, one searches for the reward function which maximizes the distance between the value of the expert at s_0 and the value of *any* policy computed so far (still at s_0). This optimization problem can be solved using a quadratic programming approach or a projection algorithm [1];
3. if $t^{(j)} \leq \epsilon$, terminate. The algorithm outputs a set of policies $\{\pi^{(0)}, \dots, \pi^{(j-1)}\}$ among which the user chooses manually or automatically the closest to the expert (see [1] for details on how to choose this policy). Notice that the last policy is not necessarily the best (as illustrated in Section 4);
4. solve the MDP with the reward function $R^{(j)}(s) = (\theta^{(j)})^T \phi(s)$ and denote $\pi^{(j)}$ the associated optimal policy. Compute $\mu^{\pi^{(j)}}(s_0)$;
5. set $j \leftarrow j + 1$ and go back to step 2.

LSPI

LSTD-μ

Compute $\mu^{\pi^{(j)}}(s_0)$

LSTD-μ : to estimate feature expectation of intermediate policies

$$\mu_i^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t) | s_0 = s, \pi].$$

seems like Value function

$$\mathcal{H}_\psi = \{\hat{V}_\xi(s) = \sum_{i=1}^{q} \xi_i \psi_i(s) = \xi^T \psi(s), \xi \in \mathbb{R}^q\}$$

LSTD estimate

$$\xi_i^* = \left(\sum_{t=1}^{n} \psi(s_t)(\psi(s_t) - \gamma \psi(s'_t))^T \right)^{-1} \sum_{t=1}^{n} \psi(s_t) \phi_i(s_t)$$

$$(\hat{\mu}^\pi(s_0))^T = \psi(s_0)^T (\Psi^T \Delta \Psi)^{-1} \Psi^T \Phi$$

LSTD-μ

Compute $\mu^{\pi^{(j)}}(s_0)$

LSTD-μ : to estimate feature expectation of intermediate policies

psi ~= Reward function

$$\mu_i^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t) | s_0 = s, \pi].$$

seems like Value function

$$V(s) = f_N(\ldots(f_2(f_1(x,\theta_1),\theta_2),\ldots),\theta_N)?$$

$$\psi(s) = f_N(\ldots(f_2(f_1(x,\theta_1),\theta_2),\ldots),\theta_N)?$$

$$\mathcal{H}_\psi = \{\hat{V}_\xi(s) = \sum_{i=1}^{q} \xi_i \psi_i(s) = \xi^T \psi(s), \xi \in \mathbb{R}^q\}$$

$$\xi_i^* = \left(\sum_{t=1}^{n} \psi(s_t)(\psi(s_t) - \gamma\psi(s_t'))^T\right)^{-1} \sum_{t=1}^{n} \psi(s_t)\phi_i(s_t)$$

LSTD estimate

$$V(s) = f_N(\ldots(f_2(f_1(x,\theta_1),\theta_2),\ldots),\theta_N)?$$

Efficient way to estimate the feature expectation of the expert in s_0

LSTD-μ as off-policy manner

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

Q-learning as off-policy algorithm

State-Action Feature expectation $\mu^\pi(s,a) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, a_0 = a, \pi]$

regards as state-action Q-function

LSTD-Q : learn a policy π by estimating a linear approximation $\hat{Q}^\pi = \Phi\omega^\pi$

$$Aw^\pi = b, \text{ where } A = \Phi^T(\Phi - \gamma \mathbf{P}\Pi_\pi \Phi) \text{ and } b = \Phi^T \mathcal{R}.$$

Additional degree of freedom($a_0 = a$) allows off-policy learning.(LSTD-Q)

LSTD-Q (Q-function) → LSTD-μ (state-action feature expectation)

Deep LSTD-μ

1) Deep feature expectation for Reward hypothesis space

$$R^*(s) = (\theta^*)^T \phi(s) \qquad \phi(s) = f_N(\ldots(f_2(f_1(x,\theta_1),\theta_2),\ldots),\theta_N)$$

f_i : DNN layer with activation function.

$$\mu^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi]$$

2) Deep feature extraction for μ hypothesis space

$$\mathcal{H}_\psi = \{\hat{V}_\xi(s) = \sum_{i=1}^{q} \xi_i \psi_i(s) = \xi^T \psi(s), \xi \in \mathbb{R}^q\} \quad \psi(s) = f_N(\ldots(f_2(f_1(x,\theta_1),\theta_2),\ldots),\theta_N)$$

$$\xi_i^* = \left(\sum_{t=1}^{n} \psi(s_t)(\psi(s_t) - \gamma\psi(s_t'))^T\right)^{-1} \sum_{t=1}^{n} \psi(s_t)\phi_i(s_t)$$

My Conclusion

What is important in IRL is,
1) Expert trajectories, 2) semantic inputs for Reward function

How to get Reward function in IRL are,
1)approximating R well or 2)narrowing the boundary of R

Input(states, features) and Reward function are closely connected in IRL.

Input $\xleftrightarrow[\text{DNN}]{}$ Reward function

DNN might help to infer the reward function.

Lots of complex, meaningful inputs are needed.

[1]Abbeel, Pieter. 2008. Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control. Ph.D. thesis, Stanford University, Stanford, CA, USA.

My Conclusion

Limitation of IRL is,

1) need expert trajectories, so it can't self-study.

Human can do → machine can do!

Human can't do → machine can't do😢

[1]Abbeel, Pieter. 2008. Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control. Ph.D. thesis, Stanford University, Stanford, CA, USA.