Reinforecement Learning

Linear value function approximation

Aleksey Grinchuk

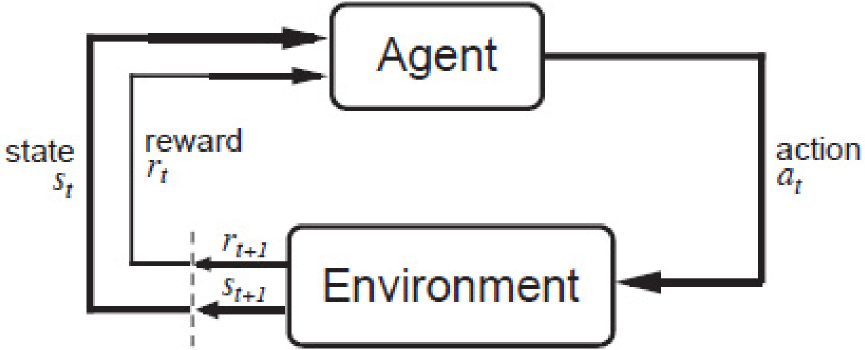
[1]Department of Control and Applied Mathematics
Moscow Institute of Physics and Technology

[2]Department of Information and Technology
Skolkovo Insititute of Science and Technology

[3]Department of Computer Science
Duke University

November 6, 2016

The MDP formalism



Notation

$< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma >$ – Markov decision process

- $\mathcal{S} = \{s_1, \dots, s_n\}$ is a finite set of states
- $\mathcal{A} = \{a_1, \dots, a_m\}$ is a finite set of actions
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbf{E}[R_{t+1} | S_t = s, A_t = a]$
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'}^a = \mathbf{P}\left[s_{t+1} = s' | S_t = s, A_t = a\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

$\pi(a|s) = \mathbf{P}\left[A_t = a | S_t = s\right]$ is a policy function
$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$ is a total reward

Definition

The action-value function $Q^\pi(s,a)$ is the expected return starting from state s, taking action a and then following policy π:

$$Q^\pi(s,a) = \mathbf{E}_\pi[G_t | s_t = s, a_t = a] = \mathbf{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

Bellman expectation equation

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right] =$$
$$= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^\pi(s', a')$$

Bellman optimality equation

$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} Q^*(s', a')$$

Optimal policy

Theorem

For any Markov Decision Process

- There exists an optimal policy π^* that is better than or equal to all other policies, $\pi^* \geq \pi, \forall \pi$
- All optimal policies achieve the optimal action-value function, $Q^{\pi^*}(s, a) = Q^*(s, a)$

An optimal policy can be found by maximising over $Q^*(s, a)$:

$$\pi^*(a|s) = \begin{cases} 1, \text{if } a = \arg\max_{a \in \mathcal{A}} Q^*(s, a) \\ 0, \text{otherwise} \end{cases}$$

If we know $Q^*(s, a)$, we immediately have the optimal policy

Solving the Bellman Optimality Equation

Bellman optimality equation

$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} Q^*(s',a')$$

Bellman Optimality Equation in the form written above is hard to solve:

- non-linear
- usually we do not know $\mathcal{R}_s^a$ and $\mathcal{P}_{ss'}^a$
- closed form solution doesn't exist (in general)

Iterative solution methods

- Value iteration
- Policy iteration
- Q-learning
- Sarsa

Reinforcement learning algorithms classification

Model-based algorithms

Build a model of system behavior from samples, and the model is used to compute a value function or policy.

Model-free algorithms

Use samples to learn a value function, from which the policy is implicitly derived. Can be decomposed into two subproblems.

- **Model-free prediction** aims to estimate the value function of an unknown MDP.
- **Model-free control** aims to optimise the value function of an unknown MDP.

Prediction

- **Monte-Carlo Learning**
  updates value $Q(S_t, A_t)$ toward actual return G_t:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

- **Temporal-Difference Learning**
  updates value toward estimated return $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

Control

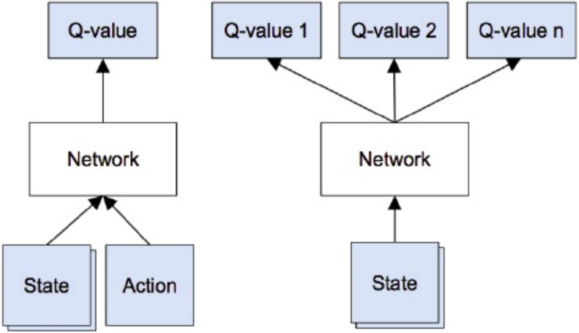
- **On-policy methods** attempt to evaluate or improve the policy that is used to make decisions
- **Off-policy methods** attempt to evaluate or improve the behavior policy while following another.

Exploration vs. exploitation

One of the key challenges of RL is to balance between using what you learned and trying to find something even better.

- **Greedy policy** works well only if we (somehow) have already learned an optimal value function and fails otherwise, especially in cases of huge MDPs.
- ϵ**-greedy policy** is a simple heuristic which is better than greedy policy, but it is unclear how to control it.
- **Bayesian neural networks** is a state-of-the-art approach which is currently in the stage of development.
  Houthooft, Rein, et al. "Variational Information Maximizing Exploration." arXiv preprint arXiv:1605.09674 (2016).

Deep learning approach: DQN



$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

Linear value-function approximation

- $< S, R, P, \gamma >$ – Markov Reward Process (MRP),
  A – raw features,
  $\Phi = [\phi_1 \dots \phi_n]$ – encoded features
- $\hat{R}$ – reward function approximation,
  $\Delta R = R - \hat{R}$ – reward function approximation error
- $\hat{P}(\phi) \approx \mathbf{E}(\phi')$ – single feature approximation,
  $\hat{P}(\Phi) = [\hat{P}(\phi_1) \dots \hat{P}(\phi_n)]$ – model approximation,
  $\Delta\Phi = P\Phi - \hat{P}(\Phi)$ – model approximation error
- $\hat{V}(\Phi) = \Phi w$ – linear value function approximation.

Inference

Let us write down the Bellman error:

$$BE(\hat{V}) = R + \gamma P \hat{V}(\Phi) - \hat{V}(\Phi)$$

With linear value function approximation this expression takes the form:

$$BE(\hat{V}) = R + \gamma P \Phi w - \Phi w = \hat{R} + \Delta R + \gamma [\Delta \Phi + \hat{P}(\Phi)] w - \Phi w =$$

$$= \underbrace{(\Delta R + \gamma \Delta \Phi w)}_{\text{learn a good model}} + \underbrace{(\hat{R} + \gamma \hat{P}(\Phi) w - \Phi w)}_{\text{find a good value function approximation}}$$

Encoder and decoder framework

Definition

The encoder $\mathcal{E} : A \to \Phi$ is a feature space transformation $\mathcal{E}(A) = \Phi$.

Definition

The decoder $\mathcal{D}$ is a matrix predicting $[PA, R]$ from $\mathcal{E}(A)$.

Definition

$\Phi = \mathcal{E}(A)$ is predictively optimal with respect to A if there are exists a $\mathcal{D}$ such that

$$\mathcal{E}(A)\mathcal{D} = [PA, R]$$

Theorem

For any MDP with predictively optimal $\Phi = \mathcal{E}(A)$ for policy π there exists $\hat{V}^\pi$, such that $\mathrm{BE}(\hat{V}^\pi) = 0$

References

Literature

- **Reinforcement Learning An Introduction** Richard S. Sutton and Andrew G. Barto, MIT press (1998)
- David Silver lectures on YouTube

Frameworks

- http://www.arcadelearningenvironment.org/
- https://gym.openai.com/
- Deepmind framework for "Starcraft 2"