Reinforcement Learning

Lecture 5: Control

Lecturer: Haim Permuter *Scribe: Ron Shoham*

I. Model-free control

In the last lecture, we learned about model-free prediction. We used different types of model-free methods such as Monte Carlo and Temporal-Difference to evaluate a policy performance. In this lecture we talk about different model-free methods to improve a non-random greedy policy, i.e., we address the control problem.

II. Generalized policy iteration

In this section we discuss how to iteratively improve our greedy policy. This process is done by repetition of two steps - policy evaluation and policy improvement. In the last lecture we have learned to evaluate value function of a given policy, i.e. $V = V_\pi$. In using this approach for policy improvement we get

$$\pi'(s) = \underset{a\in \mathrm{A}}{\operatorname{argmax}}\{r(s,a) + \gamma \sum_{s'\in \mathrm{S}} p(s'|s,a)V(s')\}. \tag{1}$$

To implement a greedy policy improvement over equation (1), we must use an MDP model. Therefore, we use a Generalized Policy Iteration over the action-value function $Q(s,a)$

$$\pi'(s) = \underset{a\in \mathrm{A}}{\operatorname{argmax}}\{Q(s,a)\}. \tag{2}$$

In using Q function for greedy policy improvement (2), we obtain a model-free method.

III. ϵ-greedy policy and GLIE

Using greedy policy improvement can result in some of the options not being explored, and as such, the process may fail to learn the optimal policy. For example, at 'T' junctions, and you must choose whether to turn right or left. The first time you choose left and get a reward of 0. The second time you choose right and get a reward of +1. Later you

again choose to turn right and again receive a reward of +1, and so on. But for each of these choices, how do you know whether you have selected the better option? Indeed, under these conditions, you cannot know, and herein lies the need for exploration.

A. ϵ-greedy policy

As we discussed above, to learn an optimal policy, exploration is required. To do so, we need to add randomality to our policy. The -greedy policy is defined as follows

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + (1-\epsilon) & , a^* = \text{argmax}\{Q_\pi(s,a)\} \\ \frac{\epsilon}{m} & , \text{otherwise}, \end{cases} \tag{3}$$

where $m = |\text{A}|$. This policy ensures that with probability, we choose a random action.

Theorem 1 (ϵ-greedy policy iteration) For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$.

Proof:

$$\begin{aligned} Q(s, \pi'(s)) &= \sum_{a \in \text{A}} \pi'(a|s) Q_\pi(s,a) \\ &= \frac{\epsilon}{m} \sum_{a \in \text{A}} Q_\pi(s,a) + (1-\epsilon) \max_a Q_\pi(s,a) \\ &\geq \frac{\epsilon}{m} \sum_{a \in |\text{A}|} Q_\pi(s,a) + (1-\epsilon) \sum_{a \in \text{A}} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon} Q_\pi(s,a) \\ &= \sum_{a \in \text{A}} \pi(a|s) Q_\pi(s,a) \\ &= V_\pi(s). \end{aligned}$$

B. GLIE - greedy in the limits with infinite exploration

Although the learning process requires some exploration, we want our policy to converge after enough visits in all of the states. The following conditions guaranty this kind of exploration

1) Number of visits of state-action pair goes to infinity

$$\lim_{k \to \infty} N_k(s,a) = \infty.$$

2) The policy converges on a greedy policy

$$\lim_{k\to\infty} \pi_k(a|s) = 1_{[a=\underset{a'\in \mathrm{A}}{\operatorname{argmax}}\{Q_k(s,a')\}]}.$$

Where k is the number of episodes. For example, a GLIE Monte Carlo algorithm can be defined as follows

1) Use π to run the kth episode $\{S_1, A_1, R_2, ..., S_{terminate}\}$
2) For each state-action pair in the episode

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t)).$$

3) Policy improvement based on new Q function

$$\epsilon \leftarrow \frac{1}{k}$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q).$$

IV. Temporal-Difference Control - SARSA

In Lecture 4, we reviewed a method for model prediction called Temporal-Difference (TD) Learning where we discussed its advantages and disadvantages relative to Monte Carlo. Now we apply a similar idea for the control challenge.

A. One-Step SARSA

Similar to the case of Temporal-Difference, here we first show how to apply a single-step SARSA (State-Action-Reward-State-Action)

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A)). \tag{4}$$

SARSA(0) looks a single step forward and improves the policy with respect to what it learns by looking ahead.

Algorithm 1 SARSA

Initialize: $Q(s,a)$ arbitrarily, and $Q(\text{terminal state},:) = 0$.

repeat

Initialize S

Choose A using policy derived from Q (e.g., ϵ-greedy)

For each step of episode:

Take action A, observe R, S'

Choose A' using policy derived from Q (e.g., ϵ-greedy)

$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$

$S \leftarrow S'$; $A \leftarrow A'$

until S is terminal

Now comes the question of how we can guarantee the convergence of SARSA.

Theorem 2 (SARSA convergence) SARSA convergence to the optimal action-value function is guaranteed under the following conditions:

1) The sequence of policies is GLIE
2) Robbins-Monro sequence of step sizes α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

B. n-Step SARSA

As we learned in Lecture 4, there exists a generalization of TD learning for more than a single step:

$$\begin{aligned}
&n = 1 \quad (\text{SARSA}) && G_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \\
&n = 2 && G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}) \\
& && \vdots \\
&n = \infty \quad (\text{MC}) && G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T.
\end{aligned}$$

Update is done by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t^{(n)} - Q(S_t, A_t)].$$

C. SARSA(λ)

Similar to what we learned in Lecture 4, here, too, we can use the TD(λ) variation. In so doing, we use a moving average to combine information over all the time-steps:

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)},$$

Algorithm 2 SARSA(λ)

Initialize: $Q(s, a)$ arbitrarily.

repeat(for each episode):

$E(s, a) = 0$, for all $s \in \mathrm{S}, a \in \mathrm{A}$

Initialize S, A

For each step of episode:

Take action A, observe R, S'

Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)

$\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$

$E(S, A) \leftarrow E(S, A) + 1$

for all $s \in \mathrm{S}, a \in \mathrm{A}$:

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t E(s, a)$

$E(s, a) \leftarrow \gamma \lambda E(s, a)$

$S \leftarrow S', A \leftarrow A'$

until S is terminal

V. Off-Policy Learning

In this section, we talk about training the target policy $\pi(a|s)$ while following behavior policy $\mu(a|s)$. This methodology of learning is important for the following cases:

- Learn from observing humans or other agents

- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$
- Learn about optimal policy while following exploratory policy
- Learn about multiple policies while following one policy

A. Importance Sampling

Estimate the expectation of a different distribution

$$\begin{aligned} \mathrm{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\ &= \mathrm{E}_{X\sim Q}[\frac{P(X)}{Q(X)}f(X)]. \end{aligned}$$

B. Importance Sampling for Off-Policy Monte Carlo

We first show how off-policy learning with importance sampling is applied in using a Monte-Carlo approach:

- Use the returns that were generated by using μ to evaluate π
- Weight return G_t according to similarity between policies
- Calculate

$$\begin{aligned} G_t^{\pi/\mu} &= \frac{\pi(a_t|s_t)p(s_{t+1}|s_t,a_t)\pi(a_{t+1}|s_{t+1})p(s_{t+2}|s_{t+1},a_{t+1})\dots}{\mu(a_t|s_t)p(s_{t+1}|s_t,a_t)\mu(a_{t+1}|s_{t+1})p(s_{t+2}|s_{t+1},a_{t+1})\dots}G_t \\ &= \frac{\pi(a_t|s_t)\pi(a_{t+1}|s_{t+1})\dots}{\mu(a_t|s_t)\mu(a_{t+1}|s_{t+1})\dots}G_t. \end{aligned}$$

- Update value

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t).$$

The main problem with this kind of learning is that the more steps we make, the higher the variance. Also, this approach cannot be used when μ is zero but π is not zero.

C. Importance Sampling for Off-Policy TD

Applying off-policy learning with importance sampling in temporal-difference fashion addresses some of the problems occur via the Monte-Carlo method.

- Use TD targets generated from μ to evaluate π
- TD target is $R + \gamma V(S')$
- $V(S_t) \leftarrow V(S_t) + \alpha[\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(S_{t+1})]$

This results in lower variance than that obtained through MC importance sampling.

VI. Q-Learning

We now consider the off-policy learning algorithm of the state-action function $Q(s, a)$, where the importance sampling is not needed. The next action is chosen using a behavior policy $A \sim \mu(\cdot|S)$, whereas the target is calculated using alternative policy $A' \sim \pi(\cdot|S)$,i.e.,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A'_t) - Q(S_t, A_t)).$$

The target policy π is greedy w.r.t. $Q(s, a)$

$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\{Q(S_{t+1}, a')\},$$

and the behavior policy μ is ϵ-greedy w.r.t. $Q(s, a)$. The Q-learning target can be simplified

$$\begin{aligned} R_{t+1} + \gamma Q(S_{t+1}, A') &= R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\text{argmax}}\{Q(S_{t+1}, a')\}) \\ &= R_{t+1} + \max_{a'}\{\gamma Q(S_{t+1}, a')\}. \end{aligned}$$

Algorithm 3 Q-learning algorithm for off-policy control

Initialize: $Q(s, a)$ arbitrarily, and $Q(\text{terminal state}, :) = 0$.

repeat(for each episode):

Initialize S

For each step of episode:

Take action A using ϵ-greedy policy derived from Q, observe R, S'

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

until S is terminal

REFERENCES

[1] UCL course on RL by David Silver, Lecture 5, Model-Free Prediction http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/control.pdf.