Temporal-Difference Learning and Q-Learning

Dominik Ziegler
12.11.2019
Recap: Simple Monte Carlo

\[ V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)] \]

- \( V(S_t) \) ... Estimate of state-value function \( \nu_\pi \)
- \( \alpha \) ... Constant step size parameter
- \( G_t \) ... Actual return following time \( t \)
Limitations

• Monte Carlo methods wait until return is known

• Wait until the end of episode to determine the increment to $V(S_t)$

• Our example: Wait until arrived at home, to update $V(S_t)$
Temporal-Difference Learning
One-step Temporal-Difference Method - TD(0)

\[ V(S_t) \leftarrow V(S_t) \]

\( V(S_t) \) ... Estimate of state-value function \( v_\pi \)
One-step Temporal-Difference Method - TD(0)

\[ V(S_t) \leftarrow V(S_t) + \alpha \]

\( V(S_t) \) ... Estimate of state-value function \( \nu_\pi \)

\( \alpha \) ... Constant step size parameter
One-step Temporal-Difference Method - TD(0)

\[ V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} ] \]

\( V(S_t) \)... Estimate of state-value function \( v_\pi \)
\( \alpha \)... Constant step size parameter
\( R_t \)... Reward at time \( t \)
One-step Temporal-Difference Method - TD(0)

\[ V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \]

\( V(S_t) \) ... Estimate of state-value function \( \nu_{\pi} \)
\( \alpha \) ... Constant step size parameter
\( R_t \) ... Reward at time \( t \)
\( \gamma \) ... Discount-rate parameter
TD(0) for estimating $\nu_\pi$

Input: the policy $\pi$ to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in S^+$, arbitrarily except that $V(terminal) = 0$
Loop for each episode:
   Initialize $S$
   Loop for each step of episode:
      $A \leftarrow$ action given by $\pi$ for $S$
      Take action $A$, observe $R, S'$
      $V(S_t) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
      $S \leftarrow S'$
   until $S$ is terminal
Temporal-Difference Prediction

Monte-Carlo:

\[ V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)] \]

**Target:** Actual return after time \( t \)

Simplest TD-Method:

\[ V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \]

**Target:** Estimate of the return
Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

Dominik Ziegler
12.11.2019
## Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5</td>
<td>35</td>
<td>40</td>
</tr>
</tbody>
</table>
Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>exiting highway</td>
<td>20</td>
<td>15</td>
<td>35</td>
</tr>
</tbody>
</table>
Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>exiting highway</td>
<td>20</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>2ndary road, behind truck</td>
<td>30</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Arrange the states on the route:
- Leaving office, Friday at 6
- Reach car, raining
- Exiting highway
- Entering home street
- Arrive home

Driving home on October 12, 2019
# Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>exiting highway</td>
<td>20</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>2ndary road, behind truck</td>
<td>30</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>entering home street</td>
<td>40</td>
<td>3</td>
<td>43</td>
</tr>
</tbody>
</table>
## Example Driving Home

<table>
<thead>
<tr>
<th>State</th>
<th>Elapsed Time (minutes)</th>
<th>Predicted Time to Go</th>
<th>Predicted Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaving office, friday at 6</td>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>reach car, raining</td>
<td>5</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>exiting highway</td>
<td>20</td>
<td>15</td>
<td>35</td>
</tr>
<tr>
<td>2ndary road, behind truck</td>
<td>30</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>entering home street</td>
<td>40</td>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>arrive home</td>
<td>43</td>
<td>0</td>
<td>43</td>
</tr>
</tbody>
</table>
Changes Recommended by Monte Carlo and TD

![Graph showing changes in predicted total travel time across different situations.](image)


Dominik Ziegler
12.11.2019
Advantages of TD Prediction

• No model of the environment, reward and next-step probability distribution
• Only experience
• Online, fully incremental (learn before knowing the outcome)
• Learn each time step vs. end of episode
• Less memory
• Less peak computation
• Converges to $v_\pi$ for any fixed policy $\pi$
Which method (MD or TD) learns faster?
Example Random Walk – Markov reward process
Example Random Walk – Markov reward process

start
Example Random Walk – Markov reward process
Example Random Walk – Markov reward process
Values learned with TD(0) ($\alpha = 0.1$)

Learning Curves for MC and TC

RMS error, averaged over states

What if only a finite amount of experience is available?
Optimality of TD(0) – Batch Updating

• Train completely on a finite amount of data
  • e.g. train repeatedly on 10 episodes until convergence
• Compute updates according to TD or MC, but only update estimates after each complete pass through the data
• TD(0) converges for sufficiently small $\alpha$
• Constant-$\alpha$ MC also converges under these conditions, but to a difference answer!
Random Walk under Batch Updating

Example: You are the Predictor

A, 0, B, 0  B, 1
B, 1        B, 1
B, 1        B, 1
B, 1        B, 0
What are the best values for the estimates $V(A)$ and $V(B)$?
$V(B) = ?$

<table>
<thead>
<tr>
<th>A, 0, B, 0</th>
<th>B, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, 1</td>
<td>B, 1</td>
</tr>
<tr>
<td>B, 1</td>
<td>B, 1</td>
</tr>
<tr>
<td>B, 1</td>
<td>B, 0</td>
</tr>
</tbody>
</table>
\[ V(B) = \frac{6}{8} = \frac{3}{4} \]
$V(A) = ?$

A, 0, B, 0  B, 1
B, 1        B, 1
B, 1        B, 1
B, 1        B, 0
Model as markov process

A \quad r = 0 \quad 100\% \quad B
Model as markov process
Model as Markov process

Diagram:

- From A to B with probability 100%:
  - $r = 0$
- From B to the other states with probabilities:
  - $r = 1$, 75%
  - $r = 0$, 25%
$$V(A) = \frac{3}{4}$$
You are the Predictor

- Batch MC gives $V(A) = 0$
  - Gives minimum squared error on training data

- Batch TD(0) gives $V(A) = \frac{3}{4}$
  - Correct for the maximum likelihood estimate of a Markov model generating the data
  - This is called the certainty-equivalence estimate
Sarsa
Learning An Action-Value Function

Estimate $q_{\pi}$ for the current policy $\pi$
One-step Sarsa

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) \]

If \( S_{t+1} \) is terminal, then \( Q(S_{t+1}, A_{t+1}) = 0 \)

\( Q(S_t, A_t) \) … Action-value function
One-step Sarsa

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [ \quad ] \]

If \( S_{t+1} \) is terminal, then \( Q(S_{t+1}, A_{t+1}) = 0 \)

\( Q(S_t, A_t) \) ... Action-value function

\( \alpha \) ... Constant step size parameter
One-step Sarsa

\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \] 

If \( S_{t+1} \) is terminal, then \( Q(S_{t+1}, A_{t+1}) = 0 \)

\( Q(S_t, A_t) \) … Action-value function
\( \alpha \) … Constant step size parameter
\( R_t \) … Reward at time \( t \)
One-step Sarsa

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

If \( S_{t+1} \) is terminal, then \( Q(S_{t+1}, A_{t+1}) = 0 \)

\( Q(S_t, A_t) \) … Action-value function

\( \alpha \) … Constant step size parameter

\( R_t \) … Reward at time \( t \)

\( \gamma \) … Discount-rate parameter
\[ Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \]

\[ S_t A_t \ R_{t+1} \ S_{t+1} A_{t+1} = Sarsa \]
Example: Windy Gridworld

Wind


undiscounted, episodic, reward = $-1$ until goal
ε-greedy Sarsa

\( \epsilon = 0.1 \)
\( \alpha = 0.5 \)
Initial values \( Q(s, a) = 0 \)
Q-Learning
Q-Learning: Off-Policy TD Control

- Off-Policy TD control algorithm
- Learned action-value function, $Q$, directly approximates $q_*$
- Independent of policy being followed
- Policy only determines which state-action pairs are visited and updated
- Only requirement: All state-action pairs continue to be updated
- $Q$ has been shown to converge with probability 1 to $q_*$

One-step Q-Learning

\[
Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]
\]

\(Q(S_t, A_t)\) … Action-value function
\(\alpha\) … Constant step size parameter
\(R_t\) … Reward at time \(t\)
\(\gamma\) … Discount-rate parameter
Example: Cliff Walking

Performance: Sarsa and Q-Learning

\[ \epsilon = 0.1 \]

Summary

- Introduced new learning method: One-step, tabular, model-free *temporal-difference* (TD) learning
- Bootstrap and sample, combining aspects of DP and MC methods
- If the world is truly Markov, then TD will learn faster than MC
- Can be used for predicting:
  - financial data, life spans, elections, weather, animal behavior, …
- MC lower error on past data, but higher error on future data
- Extend prediction to control by employing some form of *generalized policy iteration* (GPI):
  - On-policy control: Sarsa,
  - Off-policy control: Q-learning