1. Model-based

\[ \langle s, a, r \rangle^* \rightarrow \text{model learner} \rightarrow T/R \rightarrow \text{MDPsolve} \rightarrow Q^+ \rightarrow \text{argmax} \rightarrow \pi \]
1. Model-based

\[ \langle s, a, r \rangle^* \rightarrow \text{model learner} \rightarrow T/R \rightarrow \text{MDPsolve} \rightarrow Q^* \rightarrow \arg\max \rightarrow \pi \]

2. Value-function-based model-free

\[ \langle s, a, r \rangle^* \rightarrow \text{value update} \rightarrow Q \rightarrow \arg\max \rightarrow \pi \]
\[ V(S_t) \leftarrow E_\pi \left[ R_{t+1} + \gamma V(S_{t+1}) \right] \]
$V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)]$
$$V(S_t) \leftarrow V(S_t) + \alpha \left[ R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$$
TD Prediction

Policy Evaluation (the prediction problem):
for a given policy $\pi$, compute the state-value function $v_\pi$

Recall: Simple every-visit Monte Carlo method:
$$V(S_t) \leftarrow V(S_t) + \alpha \left[ G_t - V(S_t) \right]$$

**target:** the actual return after time $t$

The simplest temporal-difference method, TD(0):
$$V(S_t) \leftarrow V(S_t) + \alpha \left[ R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$$

**target:** an estimate of the return
Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps

Sampling: update does not involve an expected value
  - MC samples
  - DP does not sample
  - TD samples
Initialize $V(s)$ arbitrarily, $\pi$ to the policy to be evaluated
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        $a \leftarrow$ action given by $\pi$ for $s$
        Take action $a$; observe reward, $r$, and next state, $s'$
        $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
        $s \leftarrow s'$
    until $s$ is terminal
TD(0)

Normal learning rates: 0.01 – 0.2
.5 – very noisy
0.00001 – Matt’s not patient enough

V(S1) = 10  V(S2) = 5  r=+2

Initialize $V(s)$ arbitrarily, $\pi$ to the policy to be evaluated
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        $a \leftarrow$ action given by $\pi$ for $s$
        Take action $a$; observe reward, $r$, and next state, $s'$
        $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s') - V(s)]$
        $s \leftarrow s'$
    until $s$ is terminal
• TD methods do not require a model of the environment, only experience

• TD, but not MC, methods can be fully incremental
  You can learn **before** knowing the final outcome
  Less memory
  Less peak computation
  You can learn **without** the final outcome
  From incomplete sequences

• Both MC and TD converge (under certain assumptions to be detailed later), but which is faster?
• DP, MC, TD
• Model?
• Bootstrapping
• On-policy/off-policy
• Batch updating vs. online updating
Ex. 6.4

- \( V(B) = \frac{3}{4} \)

<table>
<thead>
<tr>
<th></th>
<th>A, 0</th>
<th>B, 0</th>
<th>B, 1</th>
<th>B, 1</th>
<th>B, 1</th>
<th>B, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ex. 6.4

- $V(B) = \frac{3}{4}$
- $V(A) = 0$? Or $\frac{3}{4}$?
The prediction that best matches the training data is \( V(A)=0 \)

- This **minimizes the mean-square-error** on the training set
- This is what a batch Monte Carlo method gets

If we consider the sequentiality of the problem, then we would set \( V(A)=.75 \)

- This is correct for the **maximum likelihood** estimate of a Markov model generating the data
- i.e., if we do a best fit Markov model, and assume it is exactly correct, and then compute what it predicts (how?)

- This is called the **certainty-equivalence estimate**
- This is what TD(0) gets
\( V(1) = 0.5 \)
\( V(2) = 0.5 \)

- 4 is terminal state
- \( V(3) = 0.5 \)
- TD(0) here is better than MC. Why?
• 4 is terminal state
• $V(3) = 0.5$
• TD(0) here is better than MC. Why?

• Visit 2 on the kth time, state 3 visited 10k times
• Variance for MC will be much higher than TD(0) because of bootstrapping
• 4 is terminal state
• $V(3) = 0.5$

• Change so that $R(3,a,4)$ was deterministic...
• Now, MC would be faster
Properties of Learning Rates

\[ V_T(s) = V_{T-1}(s) + \alpha_T (R_T(s) - V_{T-1}(s)) \]

\[ \lim_{T \to \infty} V_T(s) = V(s) \]

1. \[ \sum_T \alpha_T = \infty \]
2. \[ \sum_T \alpha_T^2 < \infty \]

Fixed learning rate \(\rightarrow\) doesn't satisfy these
Q-Learning

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \]

Initialize \( Q(s, a), \forall s \in S, a \in A(s) \), arbitrarily, and \( Q(terminal-state, \cdot) = 0 \)
Repeat (for each episode):
  Initialize \( S \)
    Repeat (for each step of episode):
      Choose \( A \) from \( S \) using policy derived from \( Q \) (e.g., \( \varepsilon \)-greedy)
      Take action \( A \), 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

S A R  \( S' \) \{a1, a2, a3\} -> a2  Q(S', a2)  ->  Q(S, A)