15-466
Computer Game Programming

Learning II:
Learning to React

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Example

• Suppose we want to learn when and what action (countermove) to take
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

First, define possible actions:

Punch, Block
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

First, define possible actions:
  Punch, Block

Second, define possible states:
  Health=Low/High, Enemy Health=Low/High, Enemy Block=Yes/No
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

First, define possible actions:
   • Punch, Block

Second, define possible states:
   • Health=Low/High, Enemy Health=Low/High, Enemy Block=Yes/No

Third, define scores (rewards):
   • Punch Enemy = if blocked: 0, if not: 1 (High Health), 5 (Low Health)
   • Got Punch = if blocked: 0, if not: -1 (High Health), -5 (Low Health)
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

  First, define possible actions:
  Punch, Block

  Second, define possible states:
  Health=Low/High, Enemy Health=Low/High, Enemy Block=Yes/No

  Third, define scores (rewards):
  Punch Enemy= if blocked: 0, if not: 1 (High Health), 5 (Low Health)
  Got Punch= if blocked: 0, if not: -1 (High Health), -5 (Low Health)

What do we do next? How do we model this mathematically?
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

*Construct MDP:*

\[ S_1 = \{ \text{Health}=L, \text{EnHealth}=H, \text{EnBlock}=0 \} \]

shortened as \{H=L, EH=H, EB=0\}

\[ A_1 = \text{Punch shortened as } A=P \]

rewards = 1 if enemy doesn't punch back

rewards = -4 otherwise

\[ A_1 = \text{Block shortened as } A=B \]

rewards = 0

*What else are we missing to construct full MDP?*
Learning with a Model

- Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

\[
\begin{align*}
S_1 &: \{H=L, \ EH=H, \ EB=0\} \\
S_2 &: \{H=L, \ EH=H, \ EB=0\} \\
S_3 &: \{H=L, \ EH=H, \ r=1, \ Prob=0.3, \ EB=1\} \\
S_4 &: \{H=L, \ EH=L, \ EB=0\} \\
S_5 &: \{H=L, \ EH=L, \ EB=1\}
\end{align*}
\]

Assuming:

\[
\begin{align*}
Prob(\text{Enemy puts Block for Future}) &= 0.6 \\
Prob(\text{Enemy punches back}) &= 0.1 \\
Prob(\text{Enemy does nothing}) &= 0.3 \\
Prob(\text{EH goes to L at each punch}) &= 0.5
\end{align*}
\]

Can we reduce the # of transitions?
• Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

- **S1**: \{H=L, EH=H, EB=0\}
  - Expected r = -0.2
  - Prob = 0.2

- **S2**: \{H=L, EH=H, EB=0\}
  - Prob(Enemy puts Block for Future) = 0.6
  - Prob(Enemy punches back) = 0.1
  - Prob(Enemy does nothing) = 0.3
  - Prob(EH goes to L at each punch) = 0.5

- **S3**: \{H=L, EH=H, r=1, Prob=0.3, EB=1\}
  - Expected r = -0.2, Prob = 0.2

- **S4**: \{H=L, EH=L, EB=0\}
  - r = 1, Prob = 0.3

- **S5**: \{H=L, EH=L, EB=1\}

**How many states we’ll have in this MDP?**

$$E_{\text{S2}} = -0.2 = \frac{-4 \times 0.3 + 1 \times 0.1}{0.2}$$

$$E_{\text{S3}} = -0.2$$

$$E_{\text{S4}} = 0$$

$$E_{\text{S5}} = 0$$
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Assuming:

Prob(Enemy puts Block for Future) = 0.6
Prob(Enemy punches back) = 0.1
Prob(Enemy does nothing) = 0.3
Prob(EH goes to L at each punch) = 0.5

We want to find a policy that maximizes our expected score (reward)

No goal state. So, we will want to the policy that favors more immediate rewards (discounts future rewards)

We need to compute the expected reward value for every state:

\[ v^*(s) = \max_a E\{ r + \gamma v^*(\text{succ}(s,a)) \}, \]

where \( \gamma = [0,1) \) is the discount factor.
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

- $S_2$: Expected $r=-0.2$ ($=-4*0.3+1*0.1$), $\text{Prob}=0.2$
- $S_1$: $\{H=L, EH=H, EB=0\}$
- $S_3$: Expected $r=-0.2, \text{Prob}=0.2$
- $S_4$: $\{H=L, EH=L, EB=0\}$
- $S_5$: $\{H=L, EH=L, EB=1\}$

**Assuming:**

- $\text{Prob(Enemy puts Block for Future)} = 0.6$
- $\text{Prob(Enemy punches back)} = 0.1$
- $\text{Prob(Enemy does nothing)} = 0.3$
- $\text{Prob(EH goes to L at each punch)} = 0.5$

**Optimal value function satisfies:**

$$v^*(s) = \max_a E\{r + \gamma v^*(\text{succ}(s,a))\}$$

$$E\{r + \gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\}$$
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

\[
\begin{align*}
&\text{At any state } s, \text{ always take action } \pi^*(s) \text{ that maximizes...} \\
&\text{Expected } r=-0.2, \text{ Prob}=0.2 \\
&\text{Expected } r=-0.2, \text{ Prob}=0.2 \\
&\text{Optimal value function satisfies: } v^*(s) = \max_a E\{r + \gamma v^*(\text{succ}(s,a))\}
\end{align*}
\]

Assuming:

\[
\begin{align*}
\text{Prob(Enemy puts Block for Future)} &= 0.6 \\
\text{Prob(Enemy punches back)} &= 0.1 \\
\text{Prob(Enemy does nothing)} &= 0.3 \\
\text{Prob(EH goes to L at each punch)} &= 0.5
\end{align*}
\]

If you know \( v^*(s) \), what is optimal policy?

\[
E\{r + \gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a))\{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\}
\]
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

**Assuming:**

- \( \text{Prob(Enemy puts Block for Future)} = 0.6 \)
- \( \text{Prob(Enemy punches back)} = 0.1 \)
- \( \text{Prob(Enemy does nothing)} = 0.3 \)
- \( \text{Prob(EH goes to L at each punch)} = 0.5 \)

\[
E\{r + \gamma v^*(s')\} = \sum_{s' \in \text{succ}(s,a)} P(succ(s,a)) \{r(s,a,succ(s,a)) + \gamma v^*(succ(s,a))\}
\]
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

\[ S_1 \]
\[ \{H=L, \]
\[ \text{EH}=H, \]
\[ \text{EB}=0\} \]

\[ S_2 \]
\[ \{H=L, \]
\[ \text{EH}=H, \]
\[ \text{EB}=0\} \]

\[ \text{Expected } r= -0.2 \]
\[ (= -4 \times 0.3 + 1 \times 0.1) \]
\[ \text{Prob}=0.2 \]

\[ S_3 \]
\[ \{H=L, \]
\[ \text{EH}=H, \]
\[ r=1, \text{Prob}=0.3 \]
\[ \text{EB}=1\} \]

\[ S_4 \]
\[ \{H=L, \]
\[ \text{EH}=L, \]
\[ \text{EB}=0\} \]

\[ \text{A=P} \]

\[ S_5 \]
\[ \{H=L, \]
\[ \text{EH}=L, \]
\[ \text{EB}=1\} \]

\[ r=1, \text{Prob}=0.3 \]

Expected \[ r= -0.2, \text{Prob}=0.2 \]

Assuming:

\[ \text{Prob(Enemy puts Block for Future)} = 0.6 \]
\[ \text{Prob(Enemy punches back)} = 0.1 \]
\[ \text{Prob(Enemy does nothing)} = 0.3 \]
\[ \text{Prob(EH goes to L at each punch)} = 0.5 \]

How do we compute \( v^*(s) \)?

Optimal value function satisfies:

\[ v^*(s) = \max_a E\{r + \gamma v^*(\text{succ}(s,a))\} \]
\[ \pi^*(s) = \arg\max_a E\{r + \gamma v^*(\text{succ}(s,a))\} \]

\[ E\{r + \gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\} \]
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

\[ S_2 \]
\[ \{H=L, \text{EH}=H, \text{EB}=0\} \]

Expected \( r=-0.2 \)
\[ (= -4*0.3 + 1*0.1) \]
\[ \text{Prob}=0.2 \]

\[ S_3 \]
\[ \{H=L, \text{EH}=H, r=1, \text{Prob}=0.3 \text{ EB}=1\} \]

\[ S_1 \]
\[ \{H=L, \text{EH}=H, \text{EB}=0\} \]

\[ A=P \]

Expected \( r=-0.2, \text{Prob}=0.2 \)

\[ S_4 \]
\[ \{H=L, \text{EH}=L, \text{EB}=0\} \]

\[ S_5 \]
\[ \{H=L, \text{EH}=L, \text{EB}=1\} \]

\( r=1, \text{Prob}=0.3 \)

Assuming:
\[ \text{Prob(Enemy puts Block for Future)} = 0.6 \]
\[ \text{Prob(Enemy punches back)} = 0.1 \]
\[ \text{Prob(Enemy does nothing)} = 0.3 \]
\[ \text{Prob(EH goes to L at each punch)} = 0.5 \]

How do we compute \( v^*(s) \)?

Value iteration (VI) is the easiest way

Optimal value function satisfies:
\[ v^*(s) = \max_a E\{r + \gamma v^*(\text{succ}(s,a))\} \]
\[ \pi^*(s) = \arg\max_a E\{r + \gamma v^*(\text{succ}(s,a))\} \]

\[ E\{r + \gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a))\{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\} \]
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Value Iteration:

For every $s$, set $v(s) = 0$ (or any other admissible value)
Iterate over all states $s$ and update until convergence:
$$v(s) = \max_a E\{r + \gamma v(succ(s,a))\}$$

For $N$ states, how many updates it takes to converge for $\gamma=0$?

Optimal value function satisfies:
$$v^*(s) = \max_a E\{r + \gamma v^*(succ(s,a))\}$$
$$\pi^*(s) = \arg\max_a E\{r + \gamma v^*(succ(s,a))\}$$

$$E\{r + \gamma v^*(succ(s,a))\} = \sum_{succ(s,a)} P(succ(s,a)) \{r(s,a, succ(s,a)) + \gamma v^*(succ(s, a))\}$$
Learning with a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Value Iteration:

For every s, set \( v(s) = 0 \) (or any other admissible value)
Iterate over all states \( s \) and update until convergence:

\[
v(s) = \max_a E\{r + \gamma v(succ(s,a))\}
\]

What did we assume in our approach to learning?

Optimal value function satisfies:

\[
v^*(s) = \max_a E\{r + \gamma v^*(succ(s,a))\}
\]

\[
\pi^*(s) = \arg\max_a E\{r + \gamma v^*(succ(s,a))\}
\]

\[
E\{r + \gamma v^*(succ(s,a))\} = \sum_{succ(s,a)} P(succ(s,a))\{r(s,a,succ(s,a)) + \gamma v^*(succ(s,a))\}
\]
Learning without a Model

• Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

\[ \begin{align*}
S_1 &\xrightarrow{A=P} S_2 \\
S_2 &\xrightarrow{a} S_3 \\
S_3 &\xrightarrow{a} S_4 \\
S_4 &\xrightarrow{a} S_5
\end{align*} \]

- Expected \( r = ? \), \( \text{Prob} = ? \)
- \( \{H=L, EH=H, EB=0\} \)
- \( \{H=L, EH=H, EB=1\} \)
- \( \{H=L, EH=L, EB=0\} \)
- \( \{H=L, EH=L, EB=1\} \)

**Expected r=?**

**Optimal value function satisfies:**

\[ v^*(s) = \max_a E \{ r + \gamma v^*(\text{succ}(s,a)) \} \]
\[ \pi^*(s) = \arg\max_a E \{ r + \gamma v^*(\text{succ}(s,a)) \} \]

**Assuming:**

- \( \text{Prob(Enemy puts Block for Future)} = ? \)
- \( \text{Prob(Enemy punches back)} = ? \)
- \( \text{Prob(Enemy does nothing)} = ? \)
- \( \text{Prob(EH goes to L at each punch)} = ? \)
Learning without a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Expected $r=?$, Prob=?

$A=P$,

$S_2$:

{H=L, EH=H, EB=0}

Expected $r=?$, Prob=?

$S_3$:

{H=L, EH=H, $r=1$, Prob=?, EB=1}

Expected $r=?$, Prob=?

$S_4$:

{H=L, EH=H, EB=0}

$r=1$, Prob=?

$S_5$:

{H=L, EH=L, EB=1}

Assuming:

$\text{Prob(Enemy puts Block for Future)} = {?}$

$\text{Prob(Enemy punches back)} = {?}$

$\text{Prob(Enemy does nothing)} = {?}$

$\text{Prob(EH goes to L at each punch)} = {?}$

But we can take actions and observe the outcomes and rewards!

Can we use our observations to learn optimal policy?

Optimal value function satisfies:

$v^*(s) = \max_a E\{r + \gamma v^*(\text{succ}(s,a))\}$

$\pi^*(s) = \arg\max_a E\{r + \gamma v^*(\text{succ}(s,a))\}$

$E\{r + \gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\}$
Learning without a Model

• Suppose we want to learn when and what action (countermove) to take

Construct MDP:

Expected $r=\$? $\text{Prob}=\$?

$S_2$

Expected $r=?$ $\text{Prob}=\$?

$S_3$

Expected $r=?$ $\text{Prob}=\$?

$S_4$

$S_5$

$A=P$

But we can take actions and observe the outcomes and rewards!

Can we use our observations to learn optimal policy?

Yes, but first let us transition from $v(s)$ to $Q(s,a)$ (from values associated with states to values associated with state-action pairs)

Assuming:

$\text{Prob(Enemy puts Block for Future)} = \$?$

$\text{Prob(Enemy punches back)} = \$?$

$\text{Prob(Enemy does nothing)} = \$?$

$\text{Prob(EH goes to L at each punch)} = \$?$

$E\{r+\gamma v^*(\text{succ}(s,a))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma v^*(\text{succ}(s,a))\}$
Learning without a Model: Q-learning

- Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

Expected $r=$?
Prob=?

$Q(s_1, A=P)$

$r=1$, Prob=?

Expected $r=}$, Prob=?

Optimal Q-values satisfy:

$$Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(\text{succ}(s,a'),a')\}$$

$$E\{r + \gamma \max_{a'} Q^*(\text{succ}(s,a'))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma \max_{a'} Q^*(\text{succ}(s,a'),a')\}$$
Learning without a Model: Q-learning

- Suppose we want to learn when and what action (countermove) to take

**Construct MDP:**

\[
\begin{align*}
&\text{S}_1: \{H=L, \quad EH=H, \quad EB=0\} \\
&\text{S}_2: \{H=L, \quad EH=H, \quad EB=0\} \\
&\text{S}_3: \{H=L, \quad EH=H, \quad r=1, \quad EB=1\} \\
&\text{S}_4: \{H=L, \quad EH=L, \quad EB=0\} \\
&\text{S}_5: \{H=L, \quad EH=L, \quad EB=1\}
\end{align*}
\]

\[
\text{Q}(s_1, A=P) = \text{E}[r + \gamma \max_{a'} Q^*(\text{succ}(s,a'), a')]
\]

\[
E[r + \gamma \max_{a'} Q^*(\text{succ}(s,a'))] = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma \max_{a'} Q^*(\text{succ}(s,a'), a')\}
\]

If you know \(Q^*(s,a)\), what is optimal policy?

At any state \(s\), always take action \(\pi^*(s)\) that maximizes...

Optimal Q-values satisfy:

\[
Q^*(s,a) = \text{E}[r + \gamma \max_{a'} Q^*(\text{succ}(s,a'), a')]
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take.

Construct MDP:

\[
Q(s_1, A=P) \
A=P
\]

\[
\{H=L, \quad EH=H, \quad EB=0\}
\]

Expected \( r=\) ?, \( Prob=\) ?

\[
S_2
\]

\[
\{H=L, \quad EH=H, \quad r=1, \quad Prob=\? \}
\]

Expected \( r=\) ?, \( Prob=\) ?

\[
S_3
\]

\[
\{H=L, \quad EH=L, \quad EB=0\}
\]

\[
S_4
\]

\[
\{H=L, \quad EH=L, \quad EB=1\}
\]

\[
S_5
\]

\[
r=1, \quad Prob=\?
\]

Optimal Q-values satisfy:

\[
Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(succ(s,a'),a')\}
\]

\[
\pi^*(s) = \arg\max_a Q^*(s,a)
\]

\[
E\{r + \gamma \max_{a'} Q^*(succ(s,a'))\} = \sum_{succ(s,a)} P(succ(s,a)) \{r(s,a,succ(s,a)) + \gamma \max_{a'} Q^*(succ(s,a'),a')\}
\]
Learning without a Model: Q-learning

- Suppose we want to learn when and what action (countermove) to take

Construct MDP:

- Optimal Q-values satisfy:
  \[ Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(\text{succ}(s,a'),a')\} \]
  \[ \pi^*(s) = \arg\max_a Q^*(s,a) \]

\[
E\{r + \gamma \max_a Q^*(\text{succ}(s,a'))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a))\{r(s,a,\text{succ}(s,a)) + \gamma \max_a Q^*(\text{succ}(s,a'),a')\}
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

**Q-learning:**

For every \(\{s,a\}\), set \(Q(s,a) = 0\) (or any other admissible value)

Every time action \(a\) at state \(s\) gets executed

- observe the reward \(r\) and outcome state \(s'\)
- update \(Q(s,a)\) to reflect the observations:

\[
Q(s,a) = (1-\alpha) Q(s,a) + \alpha (r + \gamma \max_{a'} Q(s',a'))
\]

Now, learning without a Model is easier!

**Optimal Q-values satisfy:**

\[
Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(\text{succ}(s,a'),a')\}
\]

\[
\pi^*(s) = \arg\max_a Q^*(s,a)
\]

\[
E\{r + \gamma \max_{a'} Q^*(\text{succ}(s,a'))\} = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma \max_{a'} Q^*(\text{succ}(s,a'),a')\}
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

**Q-learning:**

For every \( \{s,a\} \), set \( Q(s,a) = 0 \) (or any other admissible value)

Every time action \( a \) at state \( s \) gets executed

- observe the reward \( r \) and outcome state \( s' \)
- update \( Q(s,a) \) to reflect the observations:
  \[
  Q(s,a) = (1-\alpha) Q(s,a) + \\
  \alpha (r + \gamma \max_{a'} Q(s',a'))
  \]

What does the above equation really do?

What happens for \( \alpha = 1 \)? For \( \alpha = 0 \)?

**Optimal Q-values satisfy:**

\[
Q^*(s,a) = E\{r + \gamma \max_{a'} Q^*(succ(s,a'),a')\}
\]

\[
\pi^*(s) = \arg\max_a Q^*(s,a)
\]

\[
E\{r + \gamma \max_{a'} Q^*(succ(s,a'))\} = \\
\sum_{succ(s,a)} P(succ(s,a))\{r(s,a,succ(s,a)) + \gamma \max_{a'} Q^*(succ(s,a'),a')\}
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

**Q-learning:**

For every \( \{s, a\} \), set \( Q(s, a) = 0 \) (or any other admissible value)

Every time action \( a \) at state \( s \) gets executed

observe the reward \( r \) and outcome state \( s' \)

update \( Q(s, a) \) to reflect the observations:

\[
Q(s, a) = (1 - \alpha) Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a'))
\]

What happens if the opponent changes his tactics (or the opponent switched altogether)?

**Optimal Q-values satisfy:**

\[
Q^*(s, a) = E\{r + \gamma \max_{a'} Q^*(\text{succ}(s, a'), a')\}
\]

\[
\pi^*(s) = \arg\max_a Q^*(s, a)
\]

\[
E\{r + \gamma \max_{a'} Q^*(\text{succ}(s, a'))\} = \sum_{\text{succ}(s, a)} P(\text{succ}(s, a))\{r(s, a, \text{succ}(s, a)) + \gamma \max_{a'} Q^*(\text{succ}(s, a'), a')\}
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

\[ Q-learning: \]
For every \(\{s,a\}\), set \(Q(s,a) = 0\) (or any other admissible value)
Every time action \(a\) at state \(s\) gets executed
observe the reward \(r\) and outcome state \(s'\)
update \(Q(s,a)\) to reflect the observations:
\[
Q(s,a) = (1-\alpha)Q(s,a) + \alpha (r + \gamma \max_a Q(s',a'))
\]

But we still need to decide how to act before we have optimal \(Q^*\)-values.

Optimal \(Q\)-values satisfy:
\[
Q^*(s,a) = E\{r + \gamma \max_a Q^*(\text{succ}(s,a'),a')\}
\]
\[
\pi^*(s) = \arg\max_a Q^*(s,a)
\]
\[
E\{r + \gamma \max_a Q^*(\text{succ}(s,a'))\} = \\
\sum_{\text{succ}(s,a)} P(\text{succ}(s,a))\{r(s,a,\text{succ}(s,a)) + \gamma \max_a Q^*(\text{succ}(s,a'),a')\}\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

**Q-learning:**
For every \( \{s,a\} \), set \( Q(s,a) = 0 \) (or any other admissible value)
Every time action \( a \) at state \( s \) gets executed
  - observe the reward \( r \) and outcome state \( s' \)
  - update \( Q(s,a) \) to reflect the observations:
    \[
    Q(s,a) = (1-\alpha) Q(s,a) + \alpha (r + \gamma \max_a Q(s',a'))
    \]

**Taking actions while learning:**
with probability \( p \), take a random action,
with probability \( 1-p \), take an action that looks optimal: \( a = \arg\max_a Q(s,a) \)

**Optimal Q-values satisfy:**
\[
Q^*(s,a) = \mathbb{E}(r + \gamma \max_a Q^*(\text{succ}(s,a'),a'))
\]
\[
\pi^*(s) = \arg\max_a Q^*(s,a)
\]
\[
\mathbb{E}(r+\gamma \max_a Q^*(\text{succ}(s,a'))) = \sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{ r(s,a,\text{succ}(s,a)) + \gamma \max_a Q^*(\text{succ}(s,a'),a') \}
\]
Learning without a Model: Q-learning

• Suppose we want to learn when and what action (countermove) to take

\textbf{Q-learning:}

\begin{align*}
\text{For every } \{s,a\}, \text{ set } Q(s,a) &= 0 \text{ (or any other admissible)} \\
\text{Every time action } a \text{ at state } s \text{ gets executed} \\
\text{observe the reward } r \text{ and outcome state } s' \\
\text{update } Q(s,a) \text{ to reflect the observations:} \\
Q(s,a) &= (1-\alpha)Q(s,a) + \\
&\quad \alpha(r+\gamma \max_{a'}Q(s',a')) \\
\end{align*}

\textbf{Taking actions while learning:}

With probability \( p \), take a random action,

With probability \( 1-p \), take an action that looks optimal: \( a = \arg\max_a Q(s,a) \)

\textbf{Optimal Q-values satisfy:}

\begin{align*}
Q^*(s,a) &= E\{r+\gamma \max_{a'}Q^*(\text{succ}(s,a'),a')\} \\
\pi^*(s) &= \arg\max_a Q^*(s,a) \\
E\{r+\gamma \max_{a'}Q^*(\text{succ}(s,a'))\} &= \\
\sum_{\text{succ}(s,a)} P(\text{succ}(s,a)) \{r(s,a,\text{succ}(s,a)) + \gamma \max_{a'}Q^*(\text{succ}(s,a'),a')\} \\
\end{align*}
Keep in Mind

• Reinforcement Learning (RL) techniques such as Q-learning look exciting and powerful but can be infeasible to use

• This is even more true for using to learn during a game with many possible states
  too many updates = too many actions to try

• Whenever possible, one should make use of the model

• Planning can often be much more effective

Nevertheless, using RL in games sounds cool!