III. Model-Free Learning

• $TD(\lambda)$ used for
  
  Value Determination
  
  Given $\pi_t(s)$, compute $U_t(s)$
  within PolicyIteration

• Next step of PolicyIteration:

  Given $U_t(s)$, compute $\pi_{t+1}(s)$

  \[
  \pi_{t+1}(s) = \arg\max_a \sum_{s'} P(s' | s, a) U_t(s')
  \]

  \[\Rightarrow \text{Need model: } P(s' | s, a)\]

• Ok for Backgammon
  
  What about Factory??
Curse of Modeling

• So far: “Known” environment . . .

Agent knows
\[ M_{i,j}^a : \text{Dist over } S \times A \times S \]
\[ P(s'|s,a) \]
\[ R : S \times A \times S \to \mathbb{R} \]
\[ R(s_t,a_t,s_{t+1}) = v \]

• Typically, \( M_{i,j}^a, R(\cdots) \) unknown!

   . . . so agent can’t choose actions . . .

Option #1: First estimate \( \hat{M}(\cdots), \hat{R}(\cdots) . . . \)
then find best policy, based on \( \hat{M}, \hat{R} \)

Option #2: . . .
Define $Q_\pi(s, a) \equiv$ cumulative reward of 
performing $a$ in $s$
then following $\pi$ from then on

$$Q(s, a) \equiv R(s) + \sum_{s'} P(s' | s, a) \max_{a'} Q(s', a')$$

• If we knew $Q(\cdot, \cdot)$,
can choose optimal action $\pi(s)$
even without knowing $P(s' | s, a)$ !

$$\pi_Q(s) = \arg\max_a \{ Q(s, a) \}$$

⇒ Just need to learn this
$Q(\cdot, \cdot)$ evaluation function

• Need to know set of actions $\{a\}$ for each state $s$
but NOT where each action goes ($M_{ij}^a$)
Difference between $U$ and $Q$

\[ U(s) = R(s) + \max_a \sum_{s'} M_{s,s'}^a U(s') \]

\[ Q(s, a_1) = R(s) + \sum_{s'} M_{s,s'}^{a_1} \max_{a'} Q(s', a') \]
Example: Simple Deterministic World

\[ R(s, a) \] (immediate reward values)

\[ Q(s, a) \text{ values } (\gamma = 0.9) \quad U^*(s) \text{ values} \]

An optimal policy
Training Rule to Learn $Q$

- $Q\pi$ and $U\pi$ closely related:

$$U\pi(s) = \max_{a'} \{ Q\pi(s, a') \}$$

- Consider deterministic case:

$s' = \delta(s, a)$ is state resulting from applying action $a$ in state $s$

$$\Rightarrow Q(s_t, a_t) = R(s_t) + \gamma U(\delta(s_t, a_t))$$

$$= R(s_t) + \gamma \max_{a'} \{ Q(s_{t+1}, a') \}$$

Let: $\hat{Q} \equiv$ approx to $Q$

- Training rule: (Bellman backup-ish)

$$\hat{Q}(s, a) \leftarrow R(s) + \gamma \max_{a'} \{ \hat{Q}(s', a') \}$$
\textbf{Q-Learning for Deterministic Worlds}

For each $s, a$

initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state $s$

Do forever:

- Select an action $a$ and execute it
- Receive immediate reward $r = R(s)$
- Observe new state $s' = \delta(s, a)$
- Update table entry for $\hat{Q}(s, a)$:
  \[ \hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \{ \hat{Q}(s', a') \} \]
- $s \leftarrow s'$
Updating $\hat{Q}$

Initial state: $s_I$

Next state: $s_2$

$$\hat{Q}(s_1, a_r) \leftarrow R(s_1) + \gamma \max_{a'} \hat{Q}(\delta(s_1, a_r), a')$$

$$= 0 + 0.9 \max\{63, 81, 100\}$$

$$= 90$$

Thrm: If rewards $\geq 0$, then

$$(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$
\( \hat{Q} \) converges to \( Q \) ...

... if \( \circ \) deterministic world
\( \circ \) visit each \( \langle s, a \rangle \) infinitely often

**Proof**: Let “full interval” \( \equiv \) interval during which each \( \langle s, a \rangle \) is visited.

Let \( \hat{Q}_n \equiv \) table after \( n \) updates;
\( \Delta_n \equiv \) maximum error in \( \hat{Q}_n \)
\[ \Delta_n = \max_{s,a} \{ |\hat{Q}_n(s, a) - Q(s, a)| \} \]

**Claim**: After each full interval,
\[ \Delta_{n+fi} \leq \gamma \Delta_n \]
(largest error in \( \hat{Q} \) is reduced by \( \gamma \))

- Error in revised estimate \( \hat{Q}_{n+1}(s, a) \)
  (after updating \( Q_n(s, a) \), on iteration \( n + 1 \))
\begin{align*}
|\hat{Q}_{n+1}(s, a) - Q(s, a)| &= |(R(s) + \gamma \max_{a'} \hat{Q}_n(s', a')) - (R(s) + \gamma \max_{a'} Q(s', a'))| \\
&= \gamma |\max_{a'} \hat{Q}_n(s', a') - \max_{a'} Q(s', a')| \\
&\leq \gamma \max_{a'} |\hat{Q}_n(s', a') - Q(s', a')| \\
&\leq \gamma \max_{s''} \max_{a'} |\hat{Q}_n(s'', a') - Q(s'', a')| \\
&\leq \gamma \Delta_n
\end{align*}

Uses: \[ \max_a f_1(a) - \max_a f_2(a) \leq \max_a |f_1(a) - f_2(a)| \]
Nondeterministic Case
TD-style Learning

So far: \( \{ \text{Reward, Next state} \} \) are deterministic

What if non-deterministic?

- Redefine \( U, Q \) by taking expected values
  \[
  U^\pi(s) \equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]
  \equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]
  \]
  \[
  Q(s, a) \equiv E[R(s) + \gamma U^*(\delta(s, a))]
  \]

- New training rule:
  (Generalize \( Q \)-learning to nondeterministic worlds)

  \[
  \hat{Q}_t(s, a) \leftarrow (1 - \alpha_t)\hat{Q}_{n-1}(s, a) + \alpha_t[r + \max_{a'}\hat{Q}_{n-1}(s', a')]
  \]

  where \( \alpha_t = \frac{1}{1 + \text{visits}_t(s, a)} \)

- \( \hat{Q} \) converges to \( Q \)
  [Watkins and Dayan, 1992]
Comments on
Q-Learning Update Rule

• Like $TD(0)$
  on-line sampling of transition probabilities

  ⊕ on-line sampling of actions

• After sampling from actions $a \in A$
  approximates full Bellman backup

  [Sample $s'$ in proportion to $P(s'|s,a)$]

Note: With $U$, need $P(s'|s,a)$ to compute action

$$\pi(s) = \arg\max_a \sum_{s'} P(s'|s,a) U_t(s')$$

With $Q$, do NOT need $P(s'|s,a)$

$$\pi(s) = \arg\max_a \{ Q(s,a) \}$$
Issue:
Where to “Drive”, during Learning

• Given the $Q(\cdot, \cdot)$ value, optimal action is . . .
  $$\pi(s) = \arg\max_a \{ Q(s, a) \}$$

• How to learn these $Q(\cdot, \cdot)$ values?

• Why not just use “optimal action”?  
  When learner reaches state $s$, perform action  
  $$\arg\max_a \{ \hat{Q}_t(s, a) \}$$

• Can fall in a rut . . .
  A strategy might SEEM best (at time $t$)  
  as other regions are NOT explored.
Just Exploring “Best” Action

RMS error in utility (greedy policy)

Policy loss (greedy policy)
Should learner just take apparently-best action?

- At time $t = 3$, may think best action is
  Everyone go RIGHT... $\pi^*, T([i, j]) = \text{Right}$

  Does ok... never consider
  $\pi([1, 1]) = \text{Up}$ !

- Issue:
  - In general, need to observe all possible
    $\langle \text{state, action} \rangle$ pairs...
  - In practice, where to go each visit?

- How to balance
  $\star$ exploring region  
  $\star$ exploiting “optimal” move
**Approach: Explore/Exploit**

- At time $t$, have estimates  
  $\hat{Q}_t(s, a)$ for each state $s$, action $a$

  Let $f(u, n) = \begin{cases} 
  R^+ & \text{if } n < T \\
  u & \text{otherwise}
  \end{cases}$

  Eg, $R^+ = 2$, $T = 5$

- Maintain count  
  $N(s, a) = \text{#times took action } a \text{ from state } s$

- Select action  
  $\arg\max_a \{ f(\hat{Q}_t(s, a), N(s, a)) \}$

**Effect:** Every action gets (at least) $T = 5$ attempts afterwards, just take best.
Results

Utility estimates over number of iterations:
- (4,3)
- (3,3)
- (2,3)
- (1,1)
- (3,1)
- (4,1)
- (4,2)

RMS error and policy loss over number of epochs:
- RMS error
- Policy loss
Comparison

- $Q$-learning converges in $\approx 26$ trials

- Compare to standard $U$-learning:

(Using same exploration $R^+ = 2$, $T = 5$)

- $Q$-learning is worse
  - 26 vs 18 trials
  - Inferior final error

- Why?
  - $Q$ does not enforce consistency (as no model)

- Clearly: if you have $P(s' | s, a)$ model should use it!
Temporal Difference $Q$-Learning

- Reduce discrepancy between successive $Q$ estimates
  
  $(\hat{Q}_{(n)}$ and $\hat{Q}_{(n-1)})$

Q: When updating $\hat{Q}$, what should “more correct” value be?

- One step time difference:
  
  $Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a \{\hat{Q}(s_{t+1}, a)\}$

- Why not two steps?
  
  $Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a \{\hat{Q}(s_{t+2}, a)\}$

- Or $n$?
  
  $Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \cdots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \{\hat{Q}(s_{t+n}, a)\}$

A: Blend all of these:

$Q^\lambda(s_t, a_t) \equiv (1 - \lambda) [Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots]$
\[ Q^\lambda(s_t, a_t) \equiv (1-\lambda) \left[ Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots \right] \]

• Equivalent expression:

\[
Q^\lambda(s_t, a_t) = r_t + \gamma \left[ (1 - \lambda) \max_a Q(s_t, a_t) + \lambda \ Q^\lambda(s_{t+1}, a_{t+1}) \right]
\]

• TD(\(\lambda\)) algorithm uses above training rule
  – Sometimes converges faster than \(Q\) learning
  – converges for any \(0 \leq \lambda \leq 1\) [Dayan, 1992]
  – Tesauro’s TD-Gammon uses this alg
Dimensions

**Accessibility:** In Accessible env, state ≡ percepts.

**When Rewards:** Are rewards only at TERMINAL states, or any state?

**Prior Knowledge:** Does agent initial know model $M^\alpha_{ij}$, $R(s, a)$
or must it learn this,
as well as utility info?

**Deterministic:** Is $P(s_{t+1} | s_t, a_t) \in \{0, 1\}$?

**Fixed / Changing Policy:**

Given fixed policy:

Agent just “passively” watches world,
trying to learn utility of different states
“Active” agent changes policy.

**Discount:** Relative importance of current reward, vs future reward.

($\gamma = 1$, vs $\gamma < 1$)
Situations

Here: ALWAYS "accessible"
   Doesn’t matter:
   Rewards-only-at-Terminals?
   Discounted?
      \((Q\text{-learning proof needs } \lambda < 1)\)
   Deterministic?

• If ModelKnown, Fixed Policy:
   \(\Rightarrow\) #1A: evaluating fixed policy
   \textit{IMPROVEMENT}: stochastic approx: TD(\(\lambda\))

• If ModelKnown, \textit{Learning} Policy:
   \(\Rightarrow\) computing optimal policy
   Value Iteration, Policy Iteration, \ldots
   \textit{IMPROVEMENT}: scaling, generalization

• If Model \textit{NOT} Known, Learning Policy:
   \(\Rightarrow\) computing optimal policy (unknown)
   \textit{IMPROVEMENT}: Q-Learning
Subtleties and Ongoing Research

- Reinforcement learning for Hierarchical Problem Solvers

- Design optimal exploration strategies Occasionally perform new (non utility optimizing) move

  (see \textit{n-armed bandit} problem \cite{Russell+Norvig, p611})

- \textit{Inaccessible}: State only \textit{partially observable}

- Extend to continuous actions, states