III. Model-Free Learning

- $TD(\lambda)$ used for

  `ValueDetermination`

  Given $\pi_t(s)$, compute $U_t(s)$

  within `PolicyIteration`

- Next step of `PolicyIteration`:

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

$$\pi_{t+1}(s) = \operatorname{argmax}_a \sum_{s'} \boxed{P(s' \mid s, a)} U_t(s')$$

$\Rightarrow$ Need model: $\boxed{P(s' \mid s, a)}$

- Ok for Backgammon

  What about Factory??

Curse of Modeling

- So far: "Known" environment . . .

  Agent knows

  M_{ij}^{a}: Dist over $S \times A \times S$

  $P(\, s' \,|\, s, a\,)$

  R: $S \times A \times S \to \Re$

  $R(\, s_t, a_t, s_{t+1}\,) = v$

- Typically, M_{ij}^{a}, $R(\,\cdots)$ unknown!

  . . . so agent can't choose actions . . .

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

Option#2: . . .

Q Function

Define $Q_\pi(s,a) \equiv$ cumulative reward of
performing a in s
then following π from then on

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

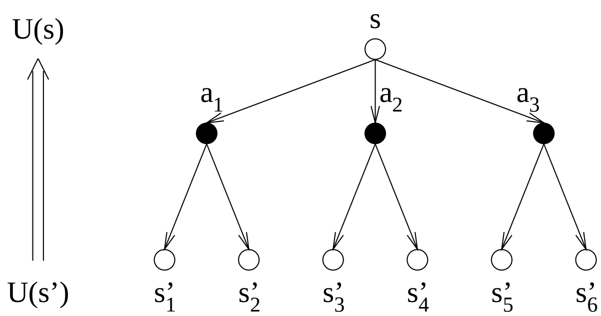
- If we knew $Q(\cdot,\cdot)$,
  can choose optimal action $\pi(s)$
  even without knowing $P(s' \mid s,a)$!

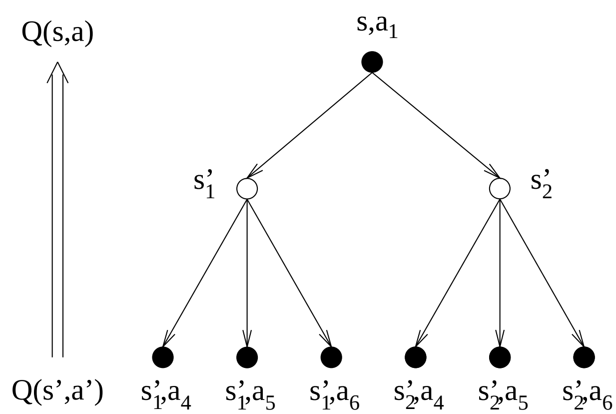
$$\pi_Q(s) \;=\; \operatorname{argmax}_a \{\; Q(s,a) \;\}$$

$\Rightarrow$ 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^a_{ij})

Difference between U and Q

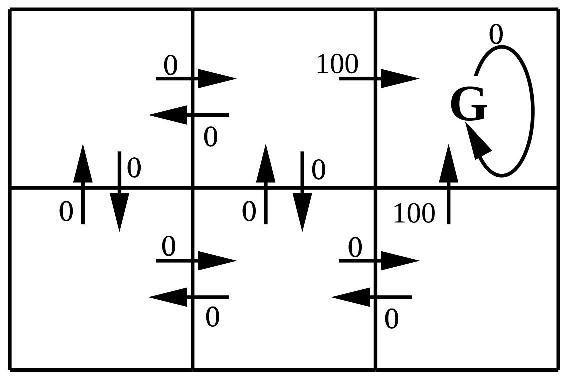




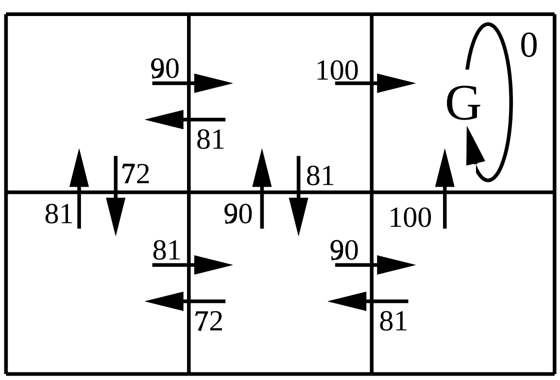
$$U(s) \;=\; R(\,s\,) + \max_a \sum_{s'} M^{a}_{s,s'}\, U(s')$$

$$Q(s,a_1) \;=\; R(\,s\,) + \sum_{s'} M^{a_1}_{s,s'} \max_{a'} Q(s',a')$$

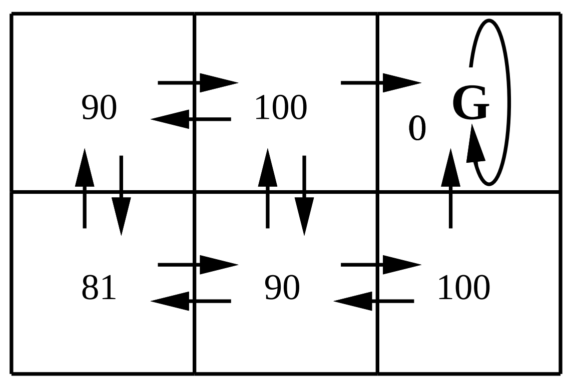
Example: Simple Deterministic World



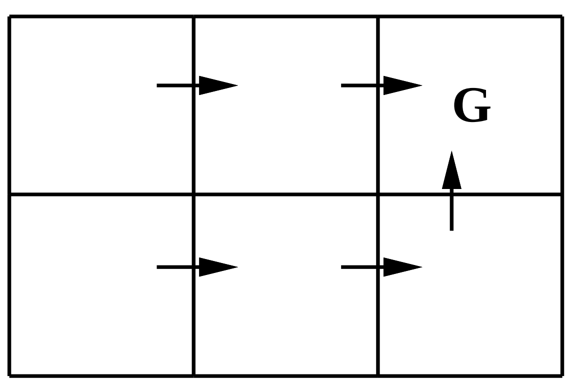
$R(\,s, a\,)$ (immediate reward values)



$Q(s, a)$ values ($\gamma = 0.9$)



$U^{*}(s)$ values



An optimal policy

Training Rule to Learn Q

- Q_π and U_π closely related:

$$U_\pi(s) \quad = \quad \max_{a'} \{Q_\pi(s,a')\}$$

- Consider *deterministic* case:

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

$$\Rightarrow \quad Q(s_t,a_t) = R(s_t) + \gamma\, U(\delta(s_t,a_t))$$
$$= R(s_t) + \gamma \max_{a'}\{Q(s_{t+1},a')\}$$

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

- Training rule: (Bellman backup-ish)

$$\hat{Q}(s,a) \quad \leftarrow \quad R(s) + \gamma \max_{a'}\{\hat{Q}(s',a')\}$$

Q-Learning for Deterministic Worlds

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

Observe current state s

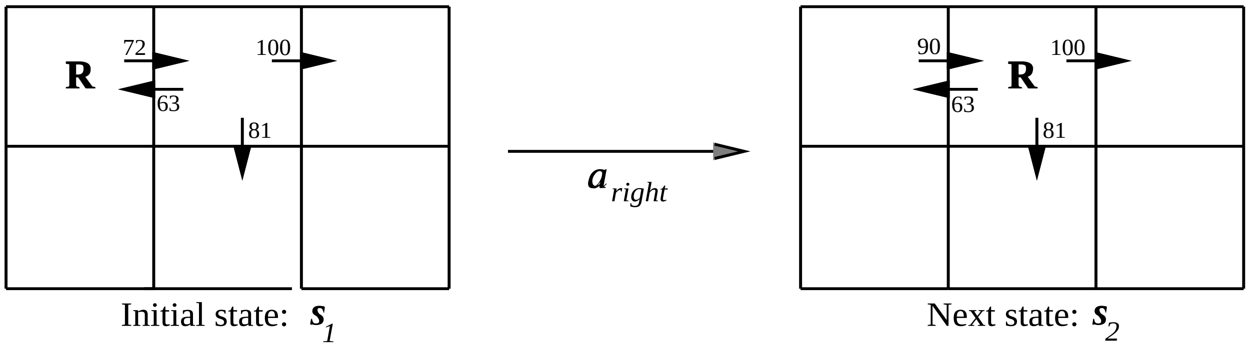
Do forever:

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

$$\hat{Q}(s, a) \quad \leftarrow \quad r + \gamma \max_{a'} \{\hat{Q}(s', a')\}$$

- $s \leftarrow s'$

Updating $\hat{Q}$



$$\begin{aligned}\hat{Q}(s_1, a_r) &\leftarrow R(s_1) + \gamma \max_{a'} \hat{Q}(\delta(s_1, a_r), a') \\ &= 0 + 0.9 \ \max\{63, 81, 100\} \\ &= 90\end{aligned}$$

Thrm: If rewards ≥ 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 ∘ deterministic world
∘ 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 $\hat{Q}_n$

$$= \max_{s,a} \{ |\hat{Q}_n(s,a) - Q(s,a)| \}$$

Claim: After each full interval,

$$\Delta_{n+fi} \leq \gamma \Delta_n$$

(largest error in $\hat{Q}$ is reduced by γ)

- Error in revised estimate $\hat{Q}_{n+1}(s,a)$
  (after updating $\hat{Q}_n(s,a)$, on iteration $n+1$)

$$
\begin{aligned}
&|\hat{Q}_{n+1}(s,a) - Q(s,a)| \\
&\quad = \; |(R(\,s\,) \; + \; \gamma \, \mathsf{max}_{a'} \hat{Q}_n(s',a')) \\
&\qquad\quad - \; (R(\,s\,) \; + \; \gamma \, \mathsf{max}_{a'} Q(s',a'))| \\
&\quad = \; \gamma \, | \, \mathsf{max}_{a'} \hat{Q}_n(s',a') - \mathsf{max}_{a'} Q(s',a')| \\
&\quad \leq \; \gamma \, \mathsf{max}_{a'} |\hat{Q}_n(s',a') - Q(s',a')| \\
&\quad \leq \; \gamma \, \mathsf{max}_{s'',a'} |\hat{Q}_n(s'',a') - Q(s'',a')| \\
&\quad \leq \; \gamma \, \Delta_n
\end{aligned}
$$

Uses: $|\max_a f_1(a) - \max_a f_2(a)| \quad \leq \quad \max_a |f_1(a) - f_2(a)|$

Nondeterministic Case TD-style Learning

So far: $\left\{ \begin{array}{l} \text{Reward} \\ \text{Next state} \end{array} \right\}$ 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 $\alpha_t = \dfrac{1}{1 + visits_t(s,a)}$

- $\hat{Q}$ 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

$$\pi(s) \;=\; \text{argmax}_a \sum_{s'} \boxed{P(\, s' \,|\, s, a\,)}\, U_t(\, s'\,)$$

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

$$\pi(s) \quad = \quad \underset{a}{\text{argmax}}\{\; Q(s, a) \;\}$$

Issue: Where to "Drive", during Learning

- Given the $Q(\cdot,\cdot)$ value, optimal action is...

  $$\pi(s) \quad = \quad \underset{a}{\text{argmax}}\{\ Q(s,a)\ \}$$

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

- Why not just use "optimal action"?

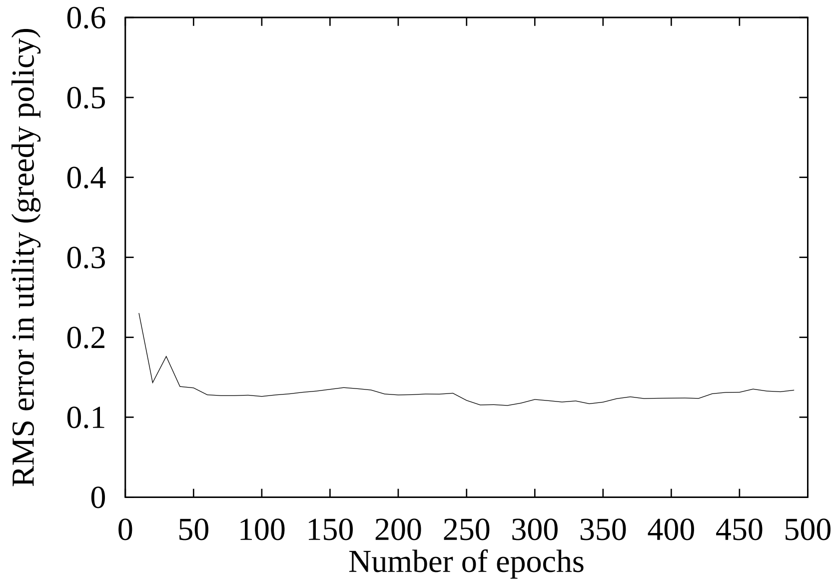
  When learner reaches state s, perform action

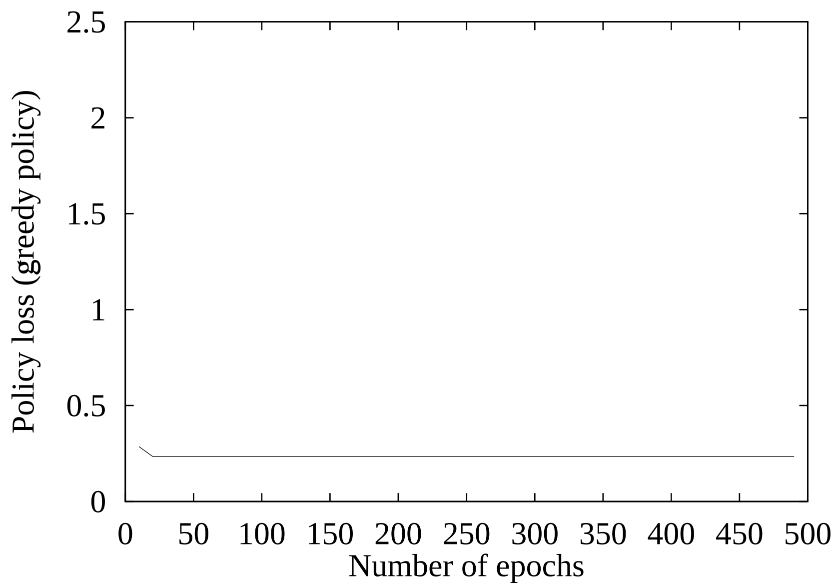
  $$\underset{a}{\text{argmax}}\{\ \hat{Q}_t(\,s,a\,)\ \}$$

- Can fall in a rut...

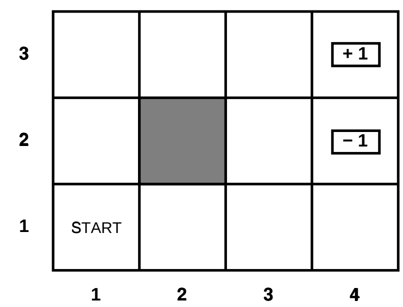
  A strategy might SEEM best (at time t)
  as other regions are NOT explored.

Just Exploring "Best" Action





Should learner just take apparently-best action?



- At time $t = 3$, may think best action is
  Everyone go RIGHT. . . $\pi^{\star,7}([i, j]) = $ Right

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

- Issue:
  - In general, need to observe all possible
    ⟨state, action⟩ pairs. . .
  - In practice, where to go each visit?

- How to balance
  - ⋆ exploring region
  - ⋆ exploiting "optimal" move

Approach: Explore/Exploit

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

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

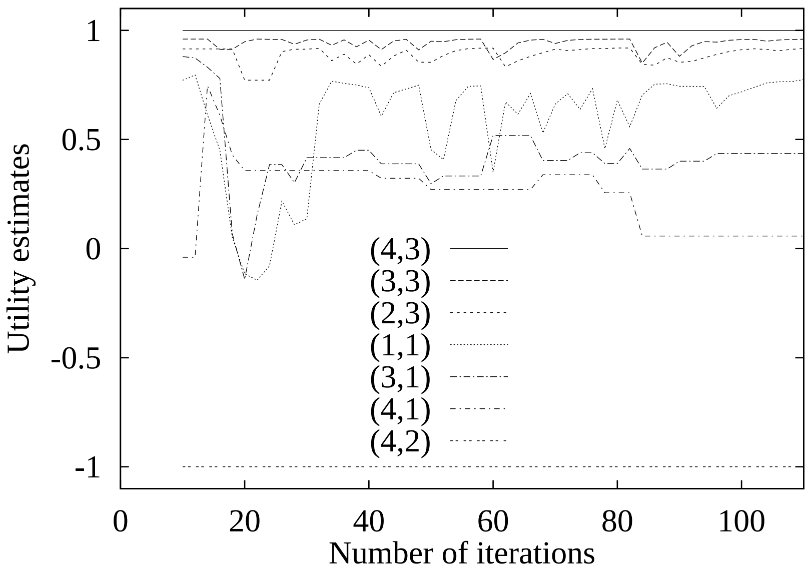
  Eg, $R^+ = 2$, $T = 5$

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

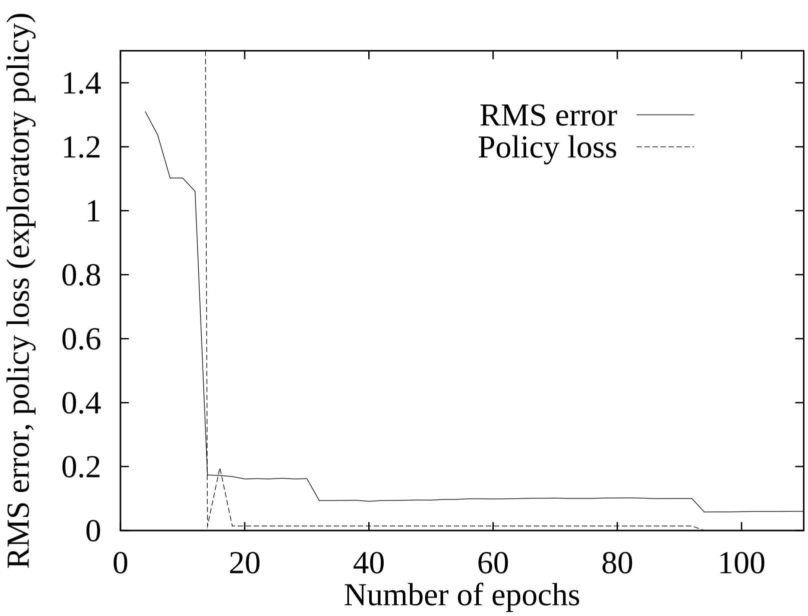
- Select action
  $\operatorname*{argmax}_a \{ \; f(\hat{Q}_t(s, a), N(s, a) \; \}$

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

Results



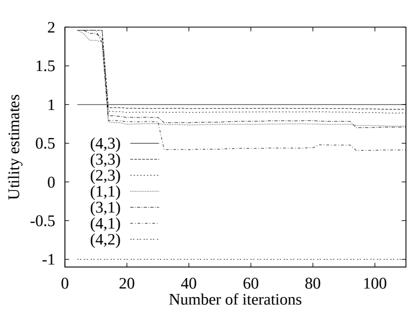
1
0.5
0
-0.5
-1
Utility estimates
(4,3)
(3,3)
(2,3)
(1,1)
(3,1)
(4,1)
(4,2)
0
20
40
60
80
100
Number of iterations

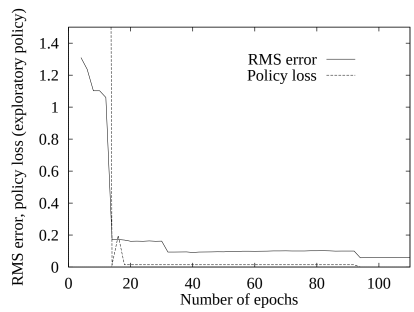


1.4
1.2
1
0.8
0.6
0.4
0.2
0
RMS error, policy loss (exploratory policy)
RMS error
Policy loss
0
20
40
60
80
100
Number of epochs

Comparison

- Q-learning converges in $\approx$ 26 trivials
- 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:

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

TD(λ) Q-Learning

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

- Equivalent expression:

$$\begin{aligned} Q^\lambda(s_t, a_t) &= r_t + \gamma[\;(1-\lambda)\max_a \hat{Q}(s_t, a_t) \\ &\qquad\qquad + \lambda\; Q^\lambda(s_{t+1}, a_{t+1})] \end{aligned}$$

- TD(λ) 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 $\equiv$ percepts.

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

Prior Knowledge: Does agent initial know model

M_{ij}^a, $R(s, a)$

or must it learn this,

as well as utility info?

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

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-learning proof needs $\lambda < 1$)

Deterministic?

- If ModelKnown, Fixed Policy:
  $\Rightarrow$ #1A: evaluating fixed policy
  IMPROVEMENT: stochastic approx: TD(λ)

- If ModelKnown, *Learning* Policy:
  $\Rightarrow$ computing optimal policy
  Value Iteration, Policy Iteration, . . .
  IMPROVEMENT: scaling, generalization

- If Model *NOT* Known, Learning Policy:
  $\Rightarrow$ computing optimal policy (unknown)
  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 n-**armed bandit** problem [Russell+Norvig, p611])

- *Inaccessible:* State only *partially observable*

- Extend to continuous actions, states