Reinforcement Learning

Searching for optimal policies I:

Bellman equations and optimal policies

Mario Martin
Universitat politècnica de Catalunya
Dept. LSI

How to find optimal policies

- Bellman equations for value functions
- Evaluation of policies
- Properties of the optimal policy
- Methods:
  - Dynamic Programming
    - Policy Iteration
    - Value Iteration
    - +[Asynchronous Versions]
  - RL algorithms
    - Q-learning
    - Sarsa
    - TD-learning

Value Functions

- The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State-value function for policy π :

$$V^{\pi}(s) = E_{\pi}\left\{R_t \mid s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

- The value of taking an action in a state under policy π is the expected return starting from that state, taking that action, and thereafter following π :

Action - value function for policy π :

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \mid s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Bellman Equation for a Policy π

The basic idea:

$$
\begin{aligned}
R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \cdots \\
&= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \cdots \right) \\
&= r_{t+1} + \gamma R_{t+1}
\end{aligned}
$$

So:

$$
\boxed{
\begin{aligned}
V^\pi(s) &= E_\pi \left\{ R_t \middle| s_t = s \right\} \\
&= E_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \middle| s_t = s \right\}
\end{aligned}
}
$$

Bellman Equation for a Policy π

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}$$
$$= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\}$$

Or, without the expectation operator:

$$V^{\pi}(s) = \sum_{s'} P_{ss'}^{\pi(s)}\left[R_{ss'}^{\pi(s)} + \gamma V^{\pi}(s')\right] \quad \text{(generic)}$$

$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')\left[r(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s') r(s,\pi(s),s') + \sum_{s'} T(s,\pi(s),s')\left[\gamma V^{\pi}(s')\right]$$

$$V^{\pi}(s) = r(s,\pi(s),s') + \gamma V^{\pi}(s') \quad \text{(deterministic environment)}$$

Bellman Equation for a Policy π

- When we are using estimations of the values, we call TD error to

$$TDerror(s) = V^{\pi}(s) - \sum_{s'} T(s,\pi(s),s')\left[r(s,\pi(s),s') + \gamma V^{\pi}(s') \right]$$

Value Functions

- The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State-value function for policy π :

$$V^{\pi}(s) = E_{\pi}\left\{R_t \mid s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

- The value of taking an action in a state under policy π is the expected return starting from that state, taking that action, and thereafter following π :

Action-value function for policy π :

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \mid s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Q-value Bellman Equation

The basic idea:

$$R_t = \underbrace{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \cdots}$$

$$= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \cdots \right)$$

$$= r_{t+1} + \gamma R_{t+1}$$

Follow policy

Action a

So:

$$\boxed{\begin{aligned} Q^\pi(s,a) &= E_\pi\left\{R_t \middle| s_t = s, a_t = a\right\} \\ &= E_\pi\left\{r_{t+1} + \gamma V^\pi(s_{t+1}) \middle| s_t = s, a_t = a\right\} \end{aligned}}$$

Q-value Bellman Equation

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\}$$
$$= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\}$$

Or, without the expectation operator:

$$Q^\pi(s,a) = \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^\pi(s') \right] \text{ (generic)}$$

$$Q^\pi(s,a) = \sum_{s'} T(s,a,s') \left[r(s,a,s') + \gamma V^\pi(s') \right]$$

$$Q^\pi(s,a) = \sum_{s'} T(s,a,s') r(s,a,s') + \sum_{s'} T(s,a,s') \left[\gamma V^\pi(s') \right]$$

$$Q^\pi(s,a) = r(s,a,s') + \gamma V^\pi(s') \qquad \text{(deterministic environment)}$$

Q-value Bellman Equation

- When we are using estimations of the values, we call TD error to

$$TDerror(s,a) = Q^{\pi}(s,a) - \sum_{s'} T(s,a,s')\left[r(s,a,s') + \gamma V^{\pi}(s') \right]$$

Calculation of value functions for a given policy (policy evaluation)

Policy Evaluation: for a given policy π, compute the state-value function V^π

Recall: **State - value function for policy π :**

$$V^\pi(s) = E_\pi\left\{R_t \mid s_t = s\right\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\}$$

First way: Solve a set of linear equations

Bellman equation for V^π :

$$V^\pi(s) = \sum_a \pi(s,a)\sum_{s'} P_{ss'}^a\left[R_{ss'}^a + \gamma V^\pi(s')\right]$$

— a system of $|S|$ simultaneous linear equations

Iterative Method for policy evaluation

Second way: iterative method (convergence proved)

$$V_0 \to V_1 \to \cdots \to V_k \to V_{k+1} \to \cdots \to V^{\pi}$$

a "sweep"

A sweep consists of applying a **backup operation** to each state.

A **full policy-evaluation backup**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Iterative Policy Evaluation

Input π, the policy to be evaluated

Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\quad \Delta \leftarrow 0$

$\quad$ For each $s \in \mathcal{S}$:

$\qquad v \leftarrow V(s)$

$\qquad V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

$\qquad \Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output $V \approx V^\pi$

Policy space: Ordering and properties of the optimal policy

- We define a partial ordering of policies "¸" in the following way:

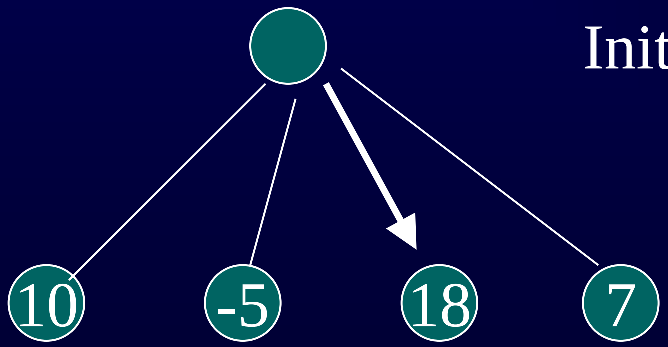
π' ¸ π **iff** $V^{\pi'}(s)$ ¸ $V^{\pi}(s)$ 8 s

- The optimal policy (π^*)
  - Could be not unique [but all share same value function $V^*=V^{\pi^*}$]
  - Some are deterministic

    [in no deterministic policies $\pi(s,a)$ *means prob. of taking action a in state s*]
  - All share the same value function
  - Optimal policies are the greedy policies with rspect to V^* or Q^*

Greedy policies

- A policy is greedy with respect to a value function it is optimal according to that value function for a one-step problem



Initial state

Different actions

Possible next states with their V(s) value

Obtaining Greedy Policies from Values

- Policy derived from values

$$\pi(s_i) = \arg\max_{a \in A} \left(\sum_{\forall j} T(s_i, a, s_j)\left(r(s_j) + \gamma V(s_j)\right) \right)$$

$$\pi(s_i) = \arg\max_{a \in A} \quad Q(s_i, a)$$

- Relation between V and Q values in Greedy policies

$$\boxed{V^{\pi}(s_t) = \max_{a \in A} Q^{\pi}(s_t, a)}$$

Reinforcement Learning

Searching for optimal policies II: Dynamic Programming

Mario Martin
Universitat politècnica de Catalunya
Dept. LSI

Two Methods for Finding Optimal Policies

- Bellman equations to organize the search for the policies in a Markovian world
- Dynamic Programming
  - Policy iteration
  - Value iteration

Policy Improvement

Suppose we have computed V^{π} for a deterministic policy π.

For a given state s,
would it be better to do an action $a \neq \pi(s)$?

The value of doing a in state s is:

$$Q^{\pi}(s,a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \,\middle|\, s_t = s, a_t = a\right\}$$
$$= \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

It is better to switch to action a for state s if and only if

$$Q^{\pi}(s,a) > V^{\pi}(s)$$

Policy Improvement Cont.

Do this for all states to get a new policy π' that is **greedy** with respect to V^π :

$$\begin{aligned}\pi'(s) &= \arg\max_a Q^\pi(s,a) \\ &= \arg\max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^\pi(s')\right]\end{aligned}$$

Then $V^{\pi'} \geq V^\pi$

Policy Iteration

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \cdots \pi^* \rightarrow V^* \rightarrow \pi^*$$

policy evaluation

policy improvement
"greedification"

Policy Iteration

Choose an arbitrary policy π

repeat

For each state (compute the value function)

$$V^{\pi}(s) := \sum_{s' \in S} \left(r(s') + \gamma V^{\pi}(s') \right) T(s, \pi(s), s')$$

For each state (improve the policy at each state)

$$\pi'(s) := \arg\max_{a \in A} \left(\sum_{s' \in S} \left(r(s') + \gamma V^{\pi}(s') \right) T(s, a, s') \right)$$

π:=π’

until no improvement is obtained

Policy Iteration

- Guaranteed to improve in less iterations than the number of states [Hooward 1960]
- Relaxation can be done in parallel and asynchronously (not complete sweeps at each iteration)

Value Iteration

Recall the **full policy-evaluation backup**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Here is the **full value-iteration backup**:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Value Iteration Cont.

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Value Iteration

- *Proved by Singh and Yee*:

$$\left| V^* - \hat{V} \right|_{\sup} \leq \varepsilon \quad \Rightarrow \quad \left| V^* - \hat{V} \right|_{\sup} \leq \gamma\varepsilon$$

- Error is decreased by a factor of γ on every iteration

Notes About Value Iteration

- Relaxation can be done
  - Asynchronously
  - In parallel

Summary

- Bellman eqs. for value functions
- Optimal policies are greedy policies
- How greedy policies can be derived form value functions
- How a policy can be evaluated
- How to iteratively improve the policy (policy iteration)
- How to calculate the value function for the optimal policy without explicit representation of policy (value iteration)

Method for Leaning Behaviors

I- Learn a world model

II- Find the optimal policy with previous algorithms

III- Execute the policy forever

Problems

- A world model is needed (transitions and reinforcements)
- Large amount of recourses involved before improving the policy
- What happen when the environment is changing?

ARE ALL THESE CONTRAINTS NECESSARY?

Reinforcement Learning

Searching for optimal policies III: RL algorithms

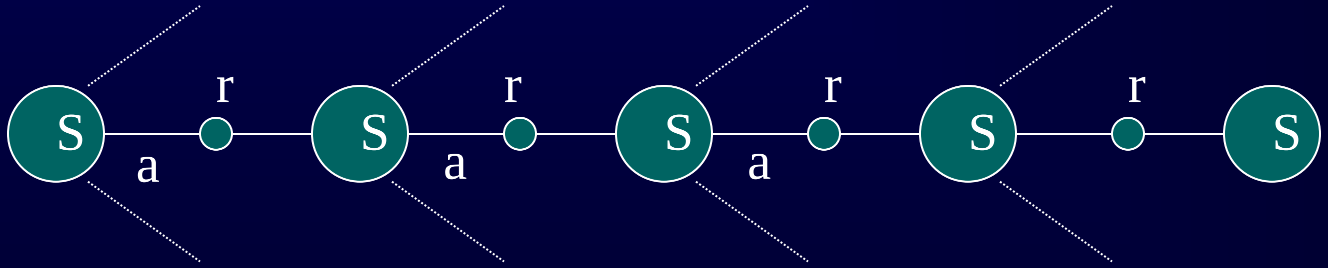
Mario Martin

Universitat politècnica de Catalunya

Dept. LSI

RL algorithms

- Active learning (learning by doing)

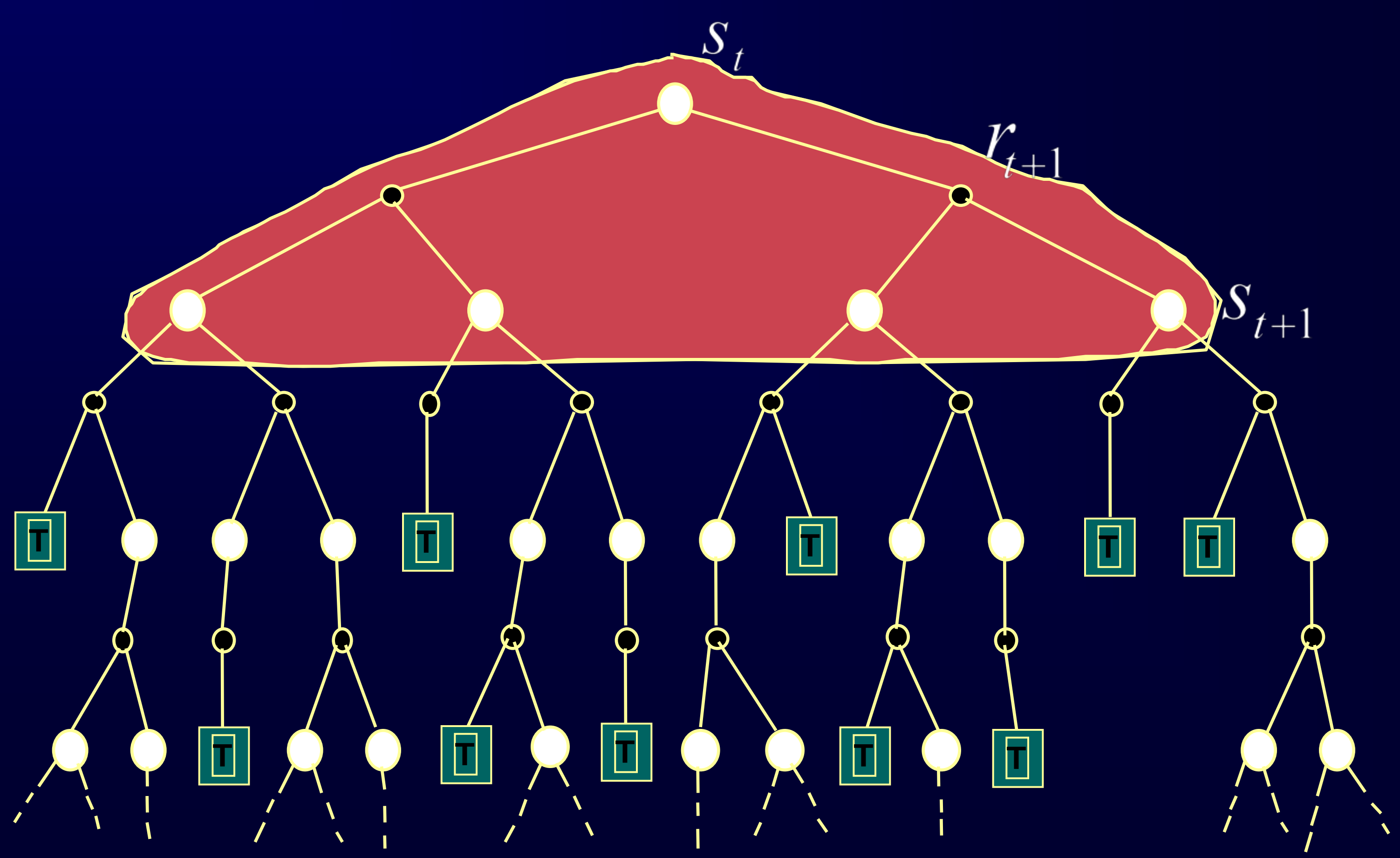


RL algorithms

- Take advantage of asynchronous updates (limit case: update only one state - the current state)
- Experiences allow a sampling of the model (transition probabilities are indirectly estimated while interacting with the environment)
- Advantages
  - No model of the world needed
  - Good policies before learning the optimal policy
  - Reacts to changes in the environment

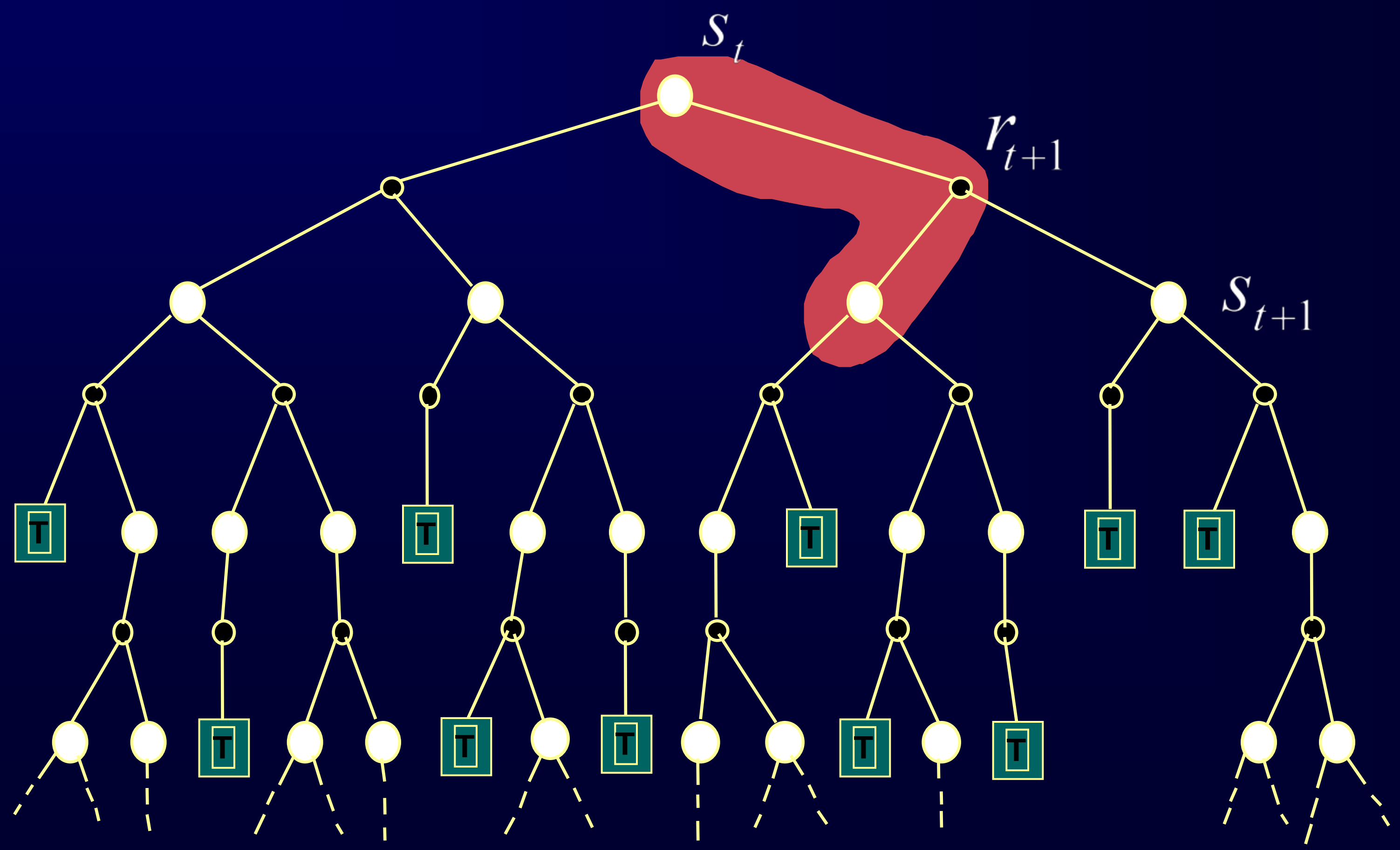
Dynamic Programming backup

$$V(s_t) \leftarrow E_\pi\{r_{t+1} + \gamma V(s_t) \mid s_t = s, \pi\}$$



Temporal Difference backup

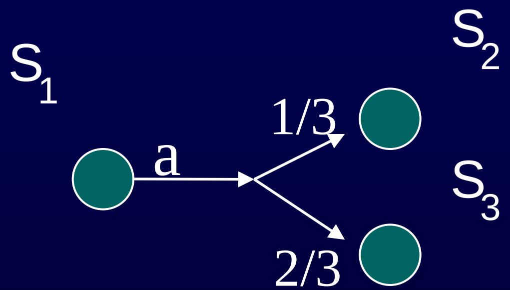
$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$



Temporal Difference backup

- Assume ®(s) = $\frac{1}{\#\text{ times visited state} + 1}$

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$



First experiment s_1 ! s_3

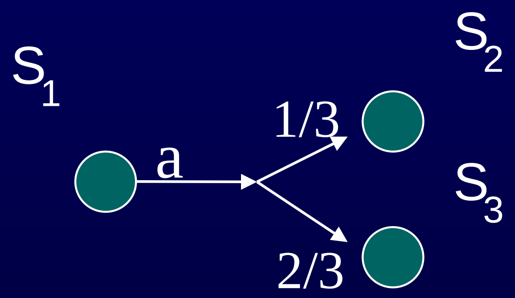
$$V(s_1) \leftarrow 0 + \alpha\left[r_3 + \gamma V(s_3) - 0\right]$$

$$V(s_1) \leftarrow 0 + 1\left[r_3 + \gamma V(s_3) - 0\right] = r_3 + \gamma V(s_3)$$

Temporal Difference backup

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

®(s) = $\dfrac{1}{\#\ \text{times visited state} + 1}$



First experiment s_1 ! s_3
Second experiment s_1 ! s_2

$$V(s_1) \leftarrow r_3 + \gamma V(s_3) + \alpha\left[r_2 + \gamma V(s_2) - r_3 + \gamma V(s_3)\right]$$

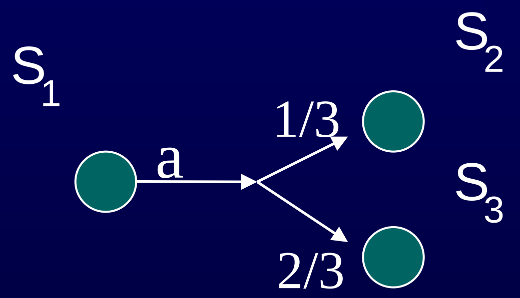
$$V(s_1) \leftarrow r_3 + \gamma V(s_3) + \frac{1}{2}\left[r_2 + \gamma V(s_2) - r_3 + \gamma V(s_3)\right]$$

$$V(s_1) \leftarrow \frac{1}{2}\left[r_2 + \gamma V(s_2)\right] + \frac{1}{2}\left[r_3 + \gamma V(s_3)\right]$$

Temporal Difference backup

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

®(s) = $\dfrac{1}{\text{\# times visited state + 1}}$



First experiment s_1 ! s_3
Second experiment s_1 ! s_2
Thrid experiment s_1 ! s_3

$$V(s_1) \leftarrow \frac{1}{2}[r_2 + \gamma V(s_2)] + \frac{1}{2}[r_3 + \gamma V(s_3)] + \alpha\left[r_3 + \gamma V(s_3) - \left(\frac{1}{2}[r_2 + \gamma V(s_2)] + \frac{1}{2}[r_3 + \gamma V(s_3)]\right)\right]$$

$$V(s_1) \leftarrow \frac{1}{2}[r_2 + \gamma V(s_2)] + \frac{1}{2}[r_3 + \gamma V(s_3)] + \frac{1}{3}\left[r_3 + \gamma V(s_3) - \left(\frac{1}{2}[r_2 + \gamma V(s_2)] + \frac{1}{2}[r_3 + \gamma V(s_3)]\right)\right]$$

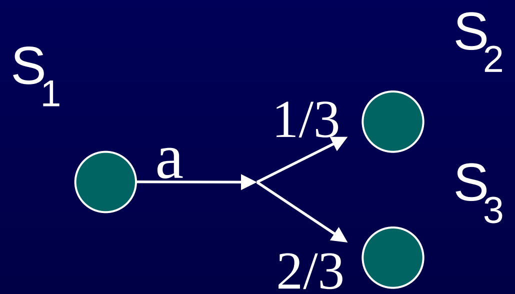
$$V(s_1) \leftarrow \frac{1}{2}[r_2 + \gamma V(s_2)] + \frac{1}{2}[r_3 + \gamma V(s_3)] + \frac{1}{3}[r_3 + \gamma V(s_3)] - \frac{1}{6}[r_2 + \gamma V(s_2)] - \frac{1}{6}[r_3 + \gamma V(s_3)]$$

$$V(s_1) \leftarrow \left(\frac{1}{2} - \frac{1}{6}\right)[r_2 + \gamma V(s_2)] + \left(\frac{1}{2} + \frac{1}{3} - \frac{1}{6}\right)[r_3 + \gamma V(s_3)] = \left(\frac{1}{3}\right)[r_2 + \gamma V(s_2)] + \left(\frac{2}{3}\right)[r_3 + \gamma V(s_3)]$$

Temporal Difference backup

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

®(s) = $\frac{1}{\text{\# times visited state + 1}}$



First experiment s_1 ! s_3
Second experiment s_1 ! s_2
Thrid experiment s_1 ! s_3
: : :

After infinite experiments,

$$V(s_1) \leftarrow T(s_1,a,s_2)\left[r_2 + \gamma V(s_2)\right] + T(s_1,a,s_3)\left[r_3 + \gamma V(s_3)\right]$$

That is,

$$V(s_t) \leftarrow E_\pi\left\{r_{t+1} + \gamma V(s_t) \mid s_t = s, a\right\}$$

The same that DP algorithms calculated but now without knowing transition probabilities!

Q-function backup

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

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

Q-function backup

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

Relation between V and Q values in Greedy policies:

$$V^{\pi}(s_t) = \max_{a \in A} Q^{\pi}(s_t, a)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \underbrace{\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]}_{\text{By the way… This is called TDerror}}$$

Q-function backup

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

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

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

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

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

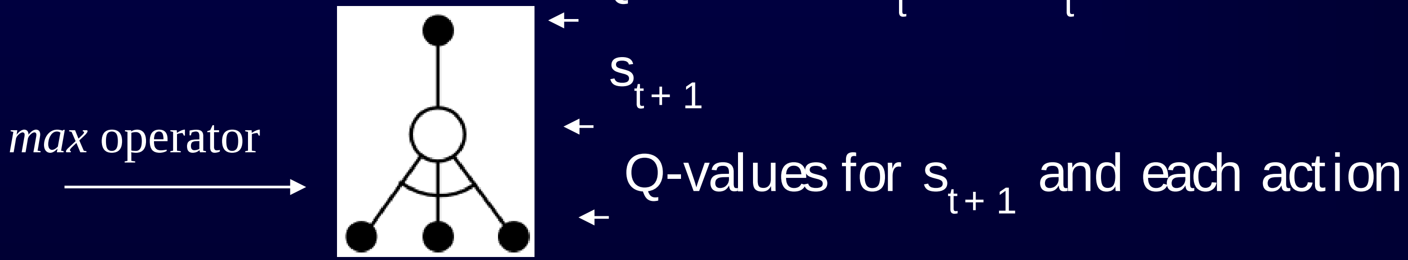
RL algorithms

- TD(0) algorithms
  - Q-learning
  - Sarsa
- TD(1) algorithms
  - Monte Carlo
- General TD-learning
  - n-steps TD estimators
  - TD(λ)

Q-learning

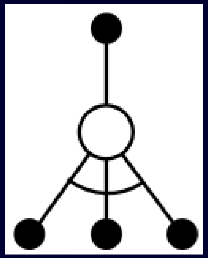
- Based on Q-backups

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



Q-Learning: Off-Policy TD (first version)

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



Initialize *Q(s,a)* and *π(s)* arbitrarily

Set agent in random initial state s

repeat

a:=π(s)

Take action a, get reinforcement r and perceive new state s'

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

$$\pi(s) \leftarrow \arg\max_{a\in A} Q(s,a)$$

s:=s'

until convergence in policy (or repeat forever)

Need for Exploratory actions

- Problems:
  - Asynchronous under the assumption that all states are visited
  - But following always a policy, some states may remain never visited
  - High possibility of being stuck with a non optimal policy in stochastic environments
    - one only state and bad luck in first estimate
    - to maximize the action in one state we must test periodically the values of the neighbor states

Exploration

- It is necessary not to follow always the policy
  - Exploration (taking a non policy action)
- But it is necessary to follow the policy for estimating the values (policy iteration)
  - Exploitation (taking a policy action)
- We must search for a balance between them

Exploration

- ε-greedy action-selection
  - Choose a greedy action with probability (1-ε) and a random action with probability ε
- Softmax action-selection

$$P_s(a) = \frac{e^{Q(s,a)/T}}{\sum_{b \in A} e^{Q(s,b)/T}}$$

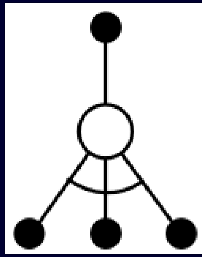
T is a parameter called *Boltzman Temperature* that usually is decreased while the learning life of the agent

Initial Values

- Other ways to avoid exploration:
  - Initializing Q values optimistically, we force an exploration procedure that (for static environments) allow us to eliminate the explicit exploration procedure

Q-Learning: Off-Policy TD (right version)

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



Initialize *Q(s,a)* and *π(s)* arbitrarily

Set agent in random initial state s

repeat

> Select action a depending on the action-selection procedure, the Q values (or the policy), and the current state s
>
> Take action a, get reinforcement r and perceive new state s'
>
> $$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$
>
> $$\pi(s) \leftarrow \arg\max_{a \in A} Q(s,a)$$
>
> s:=s'

until convergence in policy (or repeat forever)

Learning rate parameter: α

- α is used for weighting different experiences
- In stationary environments:

$$\alpha(s) = \frac{1}{\text{number of visits to state } s}$$

In this case, the Q and V values are the exact arithmetic average of the experiences

Learning rate parameter: α

- In non-stationary environments:
  α takes a constant value (usually on the range 0,3..0,5)
- Constant values decay relative influence of past experiences
- As higher the value, higher the learning (more influence of recent experiences in the estimations)

Convergence for Q-learning

- $\lim_{t \to \infty} Q(s,a) = Q^*(s,a)$
- Conditions
  – All states are infinitely visited and each action is executed an infinite number of times
  – $\sum_{i=0}^{\infty} \alpha_s = \infty$ but $\sum_{i=0}^{\infty} \alpha_s^2 < \infty$
- Watkins & Dayan 1992
  – At each "Q-interval" the maximum error is decreased in a γ factor (similar to Value Iteration)

On-line versus Off-line

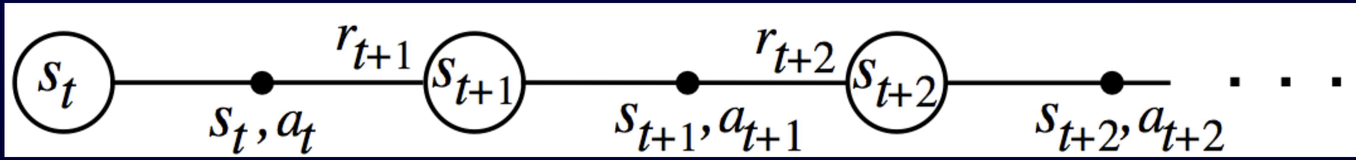
- On-line learning: Values learned are for the current policy used
- Off-line learning: Values learned for one policy while following another one.
- Q-learning is Off-line learning: Values are learned for the greedy policy, not for the ε-greedy policy used while learning
- Sarsa is On-line learning

Sarsa backup: on-policy learning

- Based on Q-backups

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

- But now we estimate Q values for the current behavior executed:



$$Q(s_{\underline{t}},a_{\underline{t}}) \leftarrow Q(s_t,a_t) + \alpha\left[r_{\underline{t+1}} + \gamma Q(s_{\underline{t+1}},a_{\underline{t+1}}) - Q(s_t,a_t)\right]$$

Sarsa: On-line Q-learning

Initialize $Q(s,a)$ and $\pi(s)$ arbitrarily

Set initial state s

Select action a depending on the action-selection procedure, the Q values (or the policy) and the current state s

repeat

Take action a, get reinforcement r and perceive new state s'

a':= Select action depending on the action-selection procedure, the Q values (or the policy) and the state s'

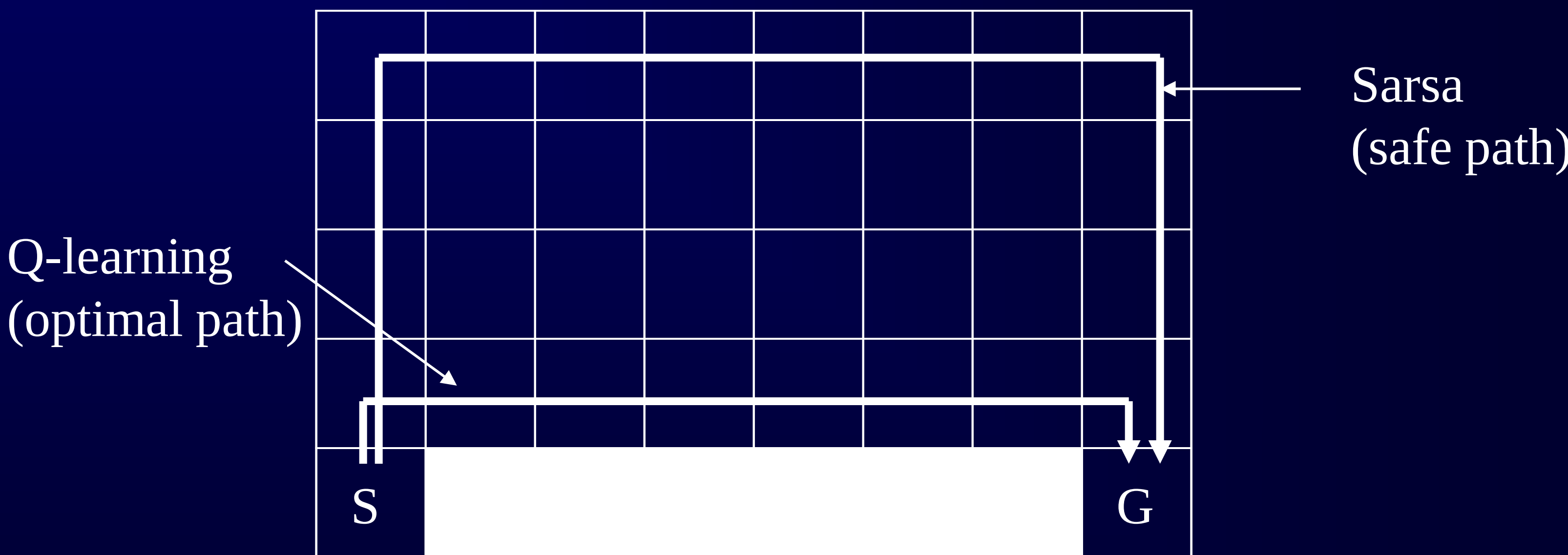
$$Q(s,a) := Q(s,a) + \alpha\left(r + \gamma Q(s',a') - Q(s,a)\right)$$

$$\pi(s) := \arg\max_{a \in A} Q(s,a)$$

r:=r'; s:=s'; a:=a'

until convergence in policy

Differences between Q-learning and Sarsa



r= -1 (after each step)
r= -100 (if she falls in the white area)
deterministic actions but ε-greedy selection procedure

Differences between Q-learning and Sarsa

ε–greedy, ε = 0.1