Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Learn from observing humans or other agents
- Learn from past policies, re-use experience from old policies
- Learn the *optimal* policy while following an *exploration* policy
- Learn multiple policies while following one policy

Chapter 6, Sutton Barto
Section 6.4, 6.5, 6.6

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Importance sampling
- Q-learning

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Importance sampling

Monte-Carlo Off-policy with importance sampling

- Importance along the whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)\pi(A_{t+1}|S_{t+1}) \ldots \pi(A_T|S_T)}{\mu(A_t|S_t)\mu(A_{t+1}|S_{t+1}) \ldots \mu(A_T|S_T)} G_t$$

- Update towards the correct return

$$V(S_t) \to V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

- Not practical, too high variance

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$ while following another policy $\mu(a \mid s)$

- Importance sampling

TD Off-policy with importance sampling

- Importance sampling correction at each step

$$V(S_t) \rightarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(S_t)\right)$$

- Lower variance than MC importance sampling
- Policies need to be similar over a single step

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- **Q-Learning approach** [Watkins, 1989]
  - Suited for TD(0)
  - No importance sampling
  - Next action using the behavior policy μ, i.e., $A_{t+1} \sim \mu(\cdot \mid S_t)$
  - Assess alternative successor action with policy π, i.e., $A' \sim \pi(\cdot|S_t)$
  - Update $Q(S_t, A_t)$ considering the alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$ while following another policy $\mu(a \mid s)$

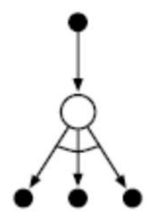
- The target policy π is greedy with respect to $Q(s,a)$

$$\pi(S_{t+1}) = argmax_{a'} Q(S_{t+1}, a')$$

- The behavior policy μ is ϵ –greedy with respect to $Q(s,a)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, argmax_{a'} Q(S_{t+1}, a')) - Q(S_t, A_t))$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') - Q(S_t, A_t))$$


Q-learning

Theorem

The Q-Learning converges towards the optimal action-value function with GLIE and

$$\lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t = \infty \quad \text{and} \quad \lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t^2 < \infty$$

Q-Learning

1. Start with initial Q-function (e.g., all zeros)
2. Take an action according to an explore/exploit policy (should converge to greedy policy, i.e. GLIE)
3. Perform TD update

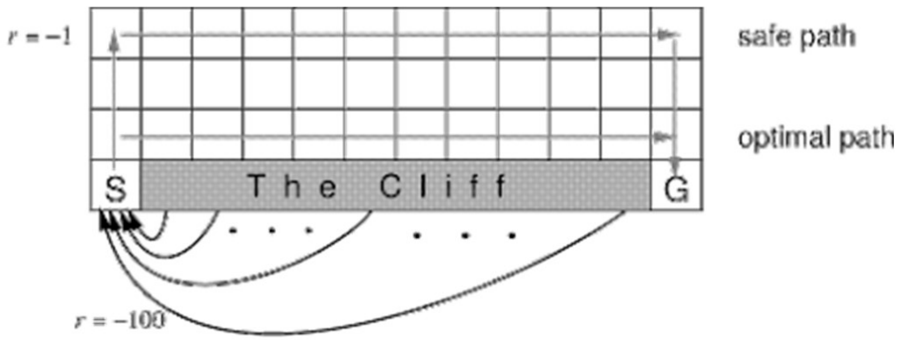
   $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') - Q(S_t, A_t))$$

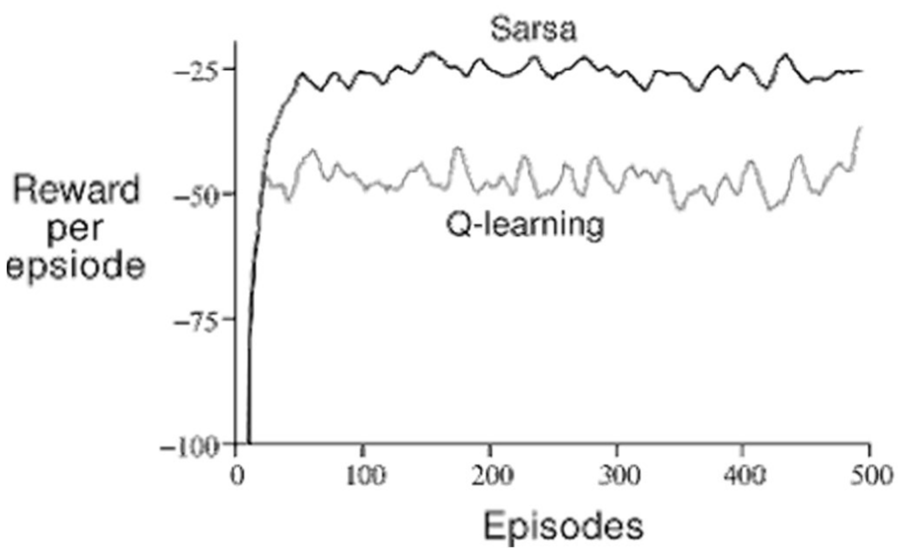
   Q(s,a) is current estimate of optimal Q-function.
4. Goto 2

- Does not require model since we learn Q directly
- Uses explicit |S|x|A| table to represent Q
- Explore/exploit policy directly uses Q-values

SARSA vs Q-Learning

Cliff Walking (undiscounted, episodic task)
- ϵ-greedy policy with $\epsilon = 0.1$
- Q-learning off-policy, more risky policy (because of ϵ-gready)





Optimal policy, but lower Reward (off-policy). If ϵ is gradually reduced both policies converge to the optimal one

Explore/Exploit Policies

- Boltzmann Exploration policy

  – Select action a with probability,

$$\Pr(a \mid s) = \frac{\exp\left(Q(s,a)/T\right)}{\sum_{a' \in A} \exp\left(Q(s,a')/T\right)}$$

  – T is the temperature. Large T means that each action has about the same probability. Small T leads to more greedy behavior.

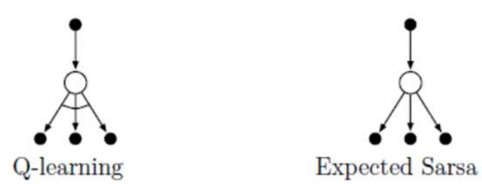
  – Typically start with large T and decrease with time

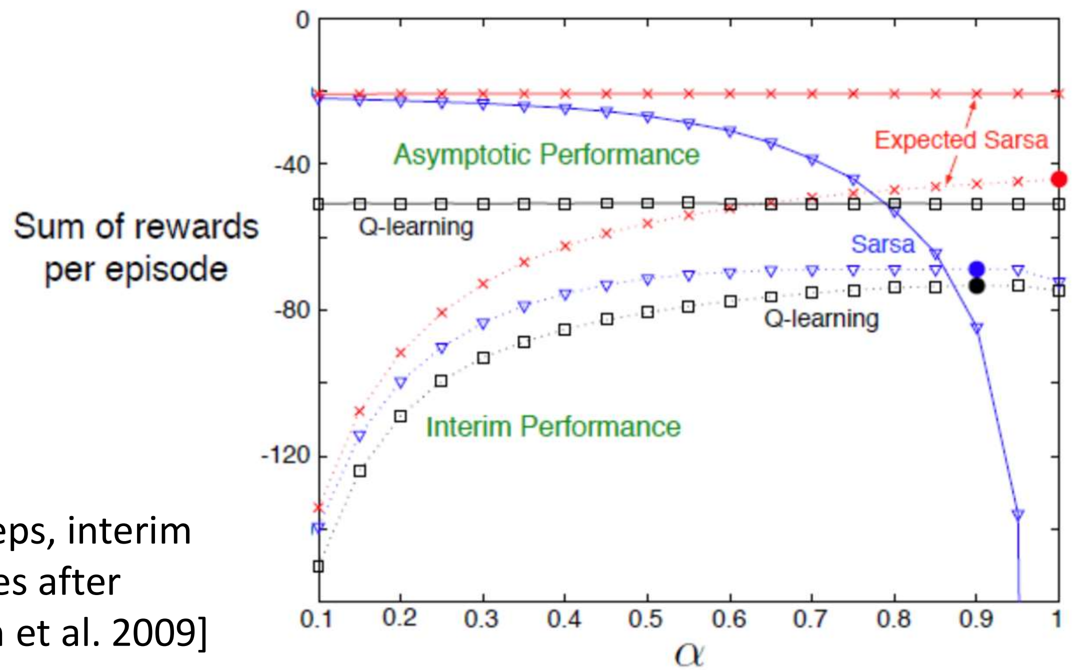
Expected Sarsa

Analogous to Q-Learning, but update with respect to the expected value instead of the maximum over next state actions

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma E[Q(S_{t+1}, A_{t+1})|S_{t+1}] - Q(S_t, A_t))$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$



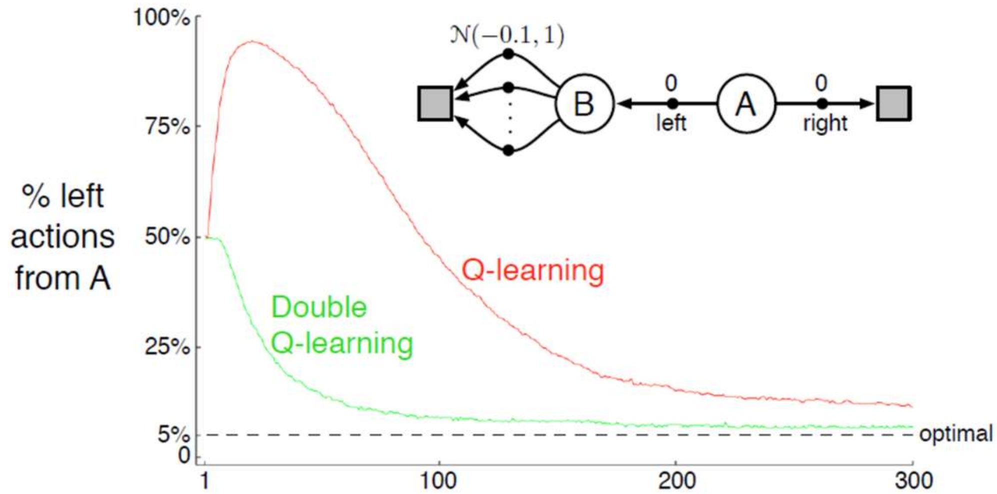


Cliff Walk
ϵ-greedy policy with $\epsilon = 0.1$
asymptotic aver over 100000 eps, interim over first 100, data are averages after 50.000 and 10 runs [van Seijen et al. 2009]

Double Learning

Maximization bias: a maximum over estimated values is used as an estimate of the maximum value, which can lead to a positive bias

Chapter 6, Sutton Barto, Section 6.7



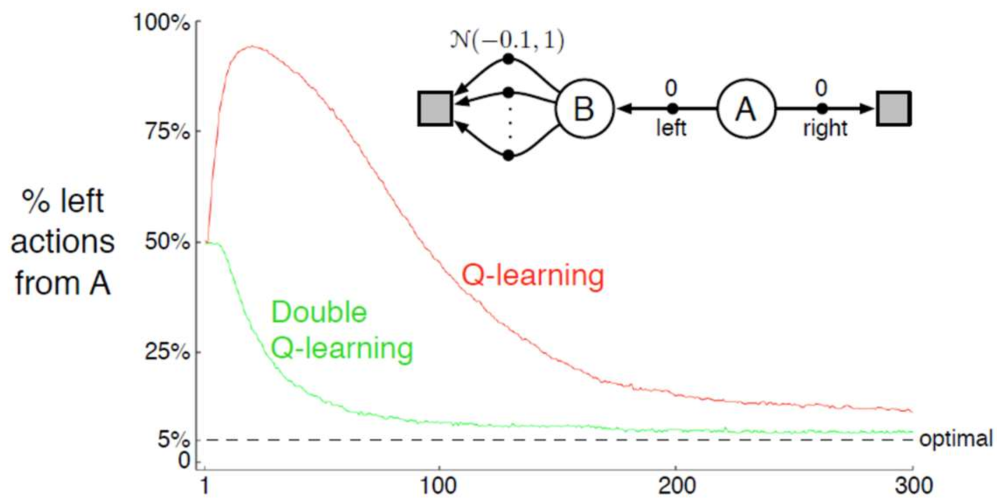
ϵ-greedy policy with $\epsilon = 0.1$, with right reward is 0, with left rewards mean is -0.1, but left may be preferred during the learning process

The same samples are used both to determine the maximizing action and to estimate its value. Divide the plays in two sets and use them to learn two independent estimates

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha(R_{t+1} + \gamma Q_2\big(S_{t+1}, argmax_{a'} Q_1(S_{t+1}, a')\big) - Q_1(S_t, A_t))$$

Double Learning

Maximization bias: a maximum over estimated values is used as an estimate of the maximum value, which can lead to a positive bias



Initialize $Q_1(s,a)$ and $Q_2(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily
Initialize $Q_1(\textit{terminal-state}, \cdot) = Q_2(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q_1 and Q_2 (e.g., ε-greedy in $Q_1 + Q_2$)
        Take action A, observe R, S'
        With 0.5 probabilility:

$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S',a)\big) - Q_1(S,A)\Big)$$

        else:

$$Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S',a)\big) - Q_2(S,A)\Big)$$

        $S \leftarrow S'$
    until S is terminal

Learning and Planning

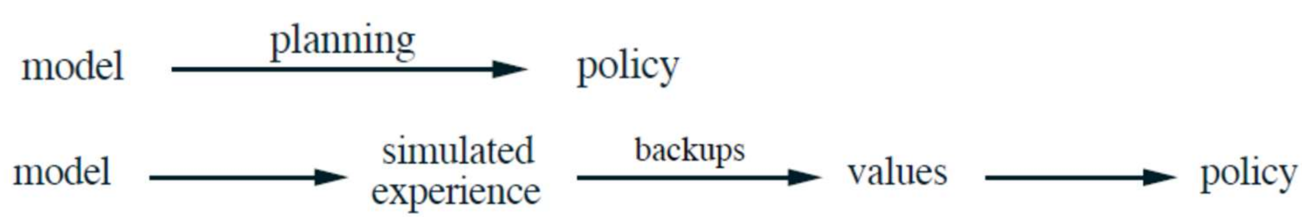
Chapter 8, Sutton Barto
Section 8.1,8.2, 8.5

Combine Model-based and Model-free methods

Model-free
- Learn value function and action-value function via experience

Model-based
- Learn the model via experience
- Use the model to generate the value function/policy (learn from <u>simulated experience</u>)

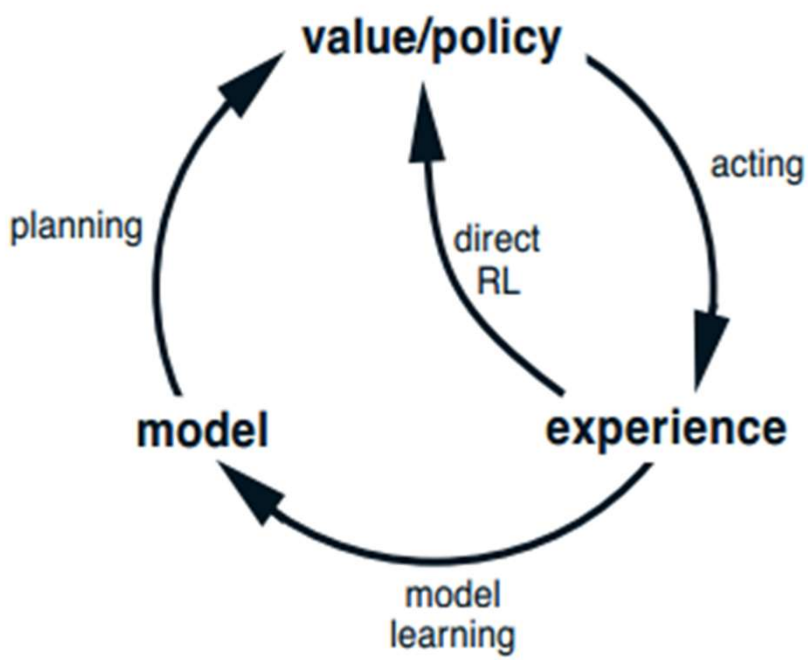
Combined (Dyna)
- Learn the model via experience
- Learn and plan via real and simulated experience

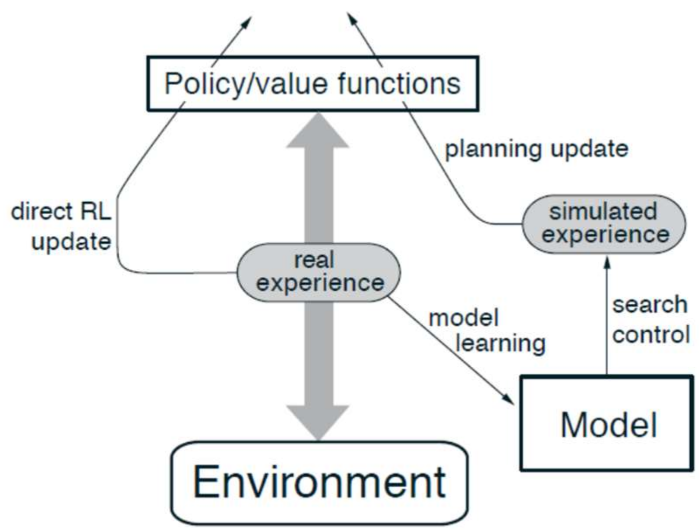


Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)
- Learn the model via experience
- Learn and plan via real and simulated experience





Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)

- Learn the model via experience
- Learn and plan via real and simulated experience

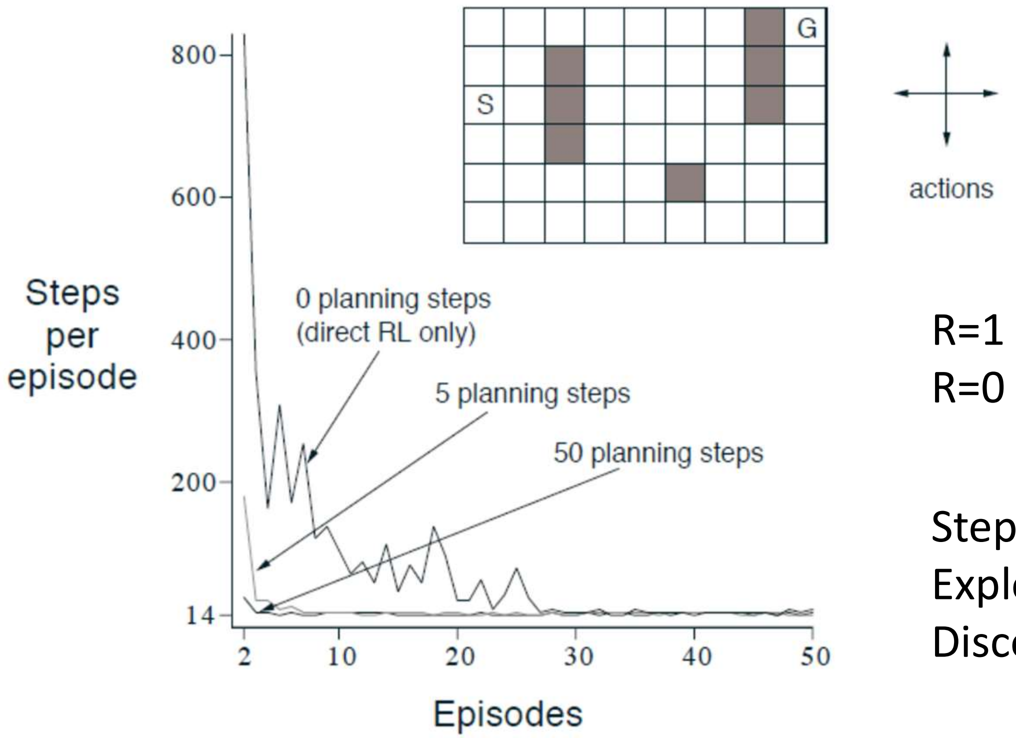
Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
  (a) $S \leftarrow$ current (nonterminal) state
  (b) $A \leftarrow \epsilon\text{-greedy}(S, Q)$
  (c) Execute action A; observe resultant reward, R, and state, S'
  (d) $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
  (e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
  (f) Repeat n times:
    $S \leftarrow$ random previously observed state
    $A \leftarrow$ random action previously taken in S
    $R, S' \leftarrow Model(S, A)$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)

- Learn the model via experience
- Learn and plan via real and simulated experience



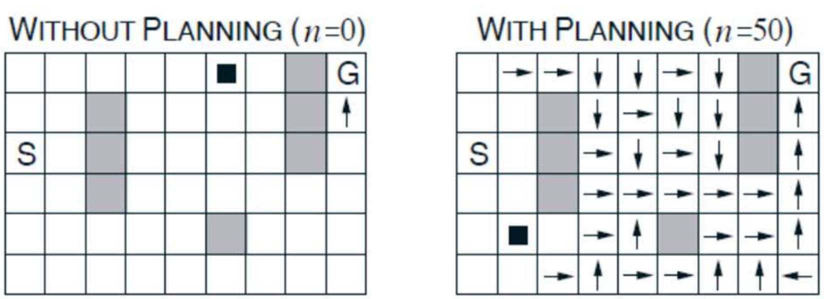
R=1 if goal,
R=0 otherwise

Step-size $\alpha = 0.1$
Exploration $\epsilon = 0.1$
Discount $\gamma = 0.95$

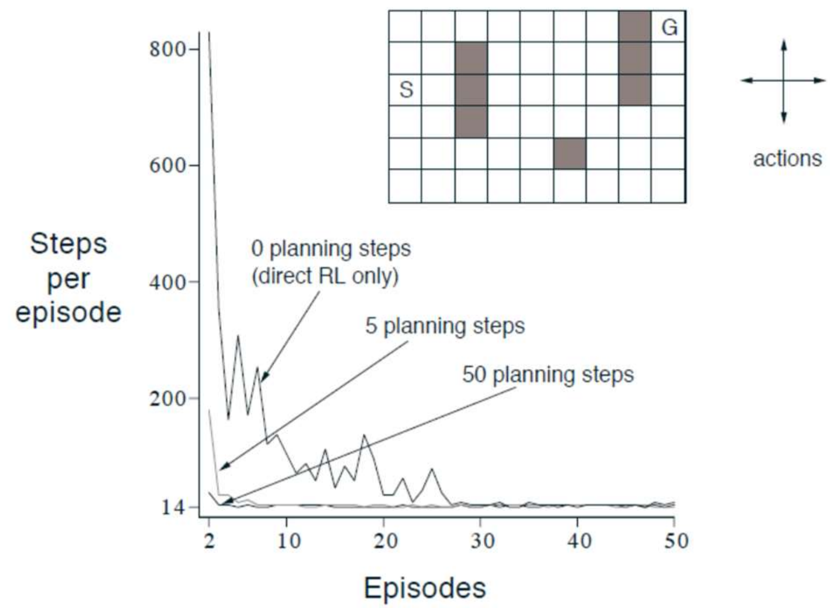
Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)
- Learn the model via experience
- Learn and plan via real and simulated experience



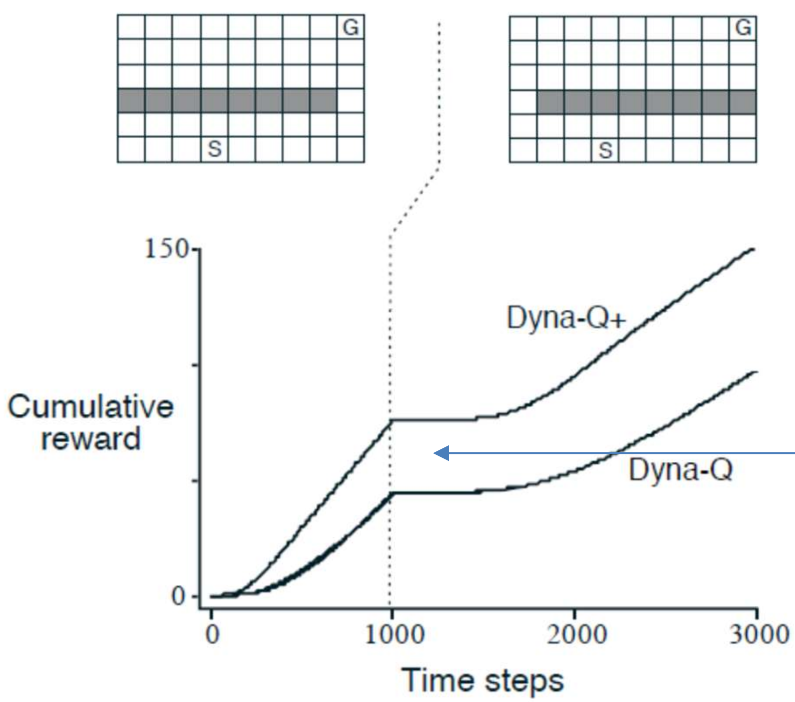
Policies found by planning and nonplanning DynaQ halfway through the second episode. Arrows are for greedy actions in each state



Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)
- When the model is wrong …
- DynaQ+ has a bonus on the explorative behavior



After 1000 time steps the short path is blocked

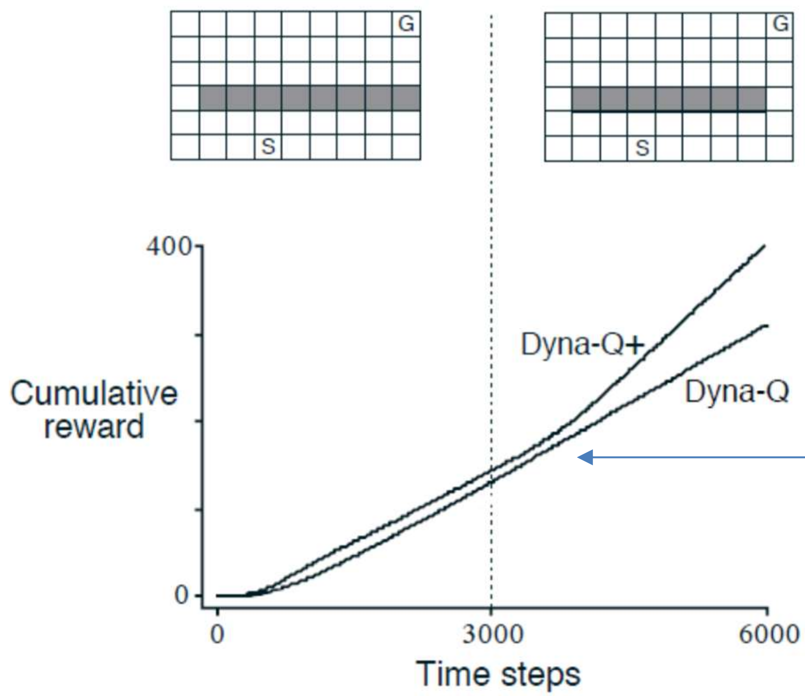
Search for a new path

Learning and Planning

Combine Model-based and Model-free methods

Combined (Dyna)
- When the model is wrong …
- DynaQ+ has a bonus on the explorative behavior

DynaQ+ keeps track of time elapsed since the action-state pair was tried, a bonus reward in simulated experience is provided to test past pair



After 1000 time steps the short path is shorter

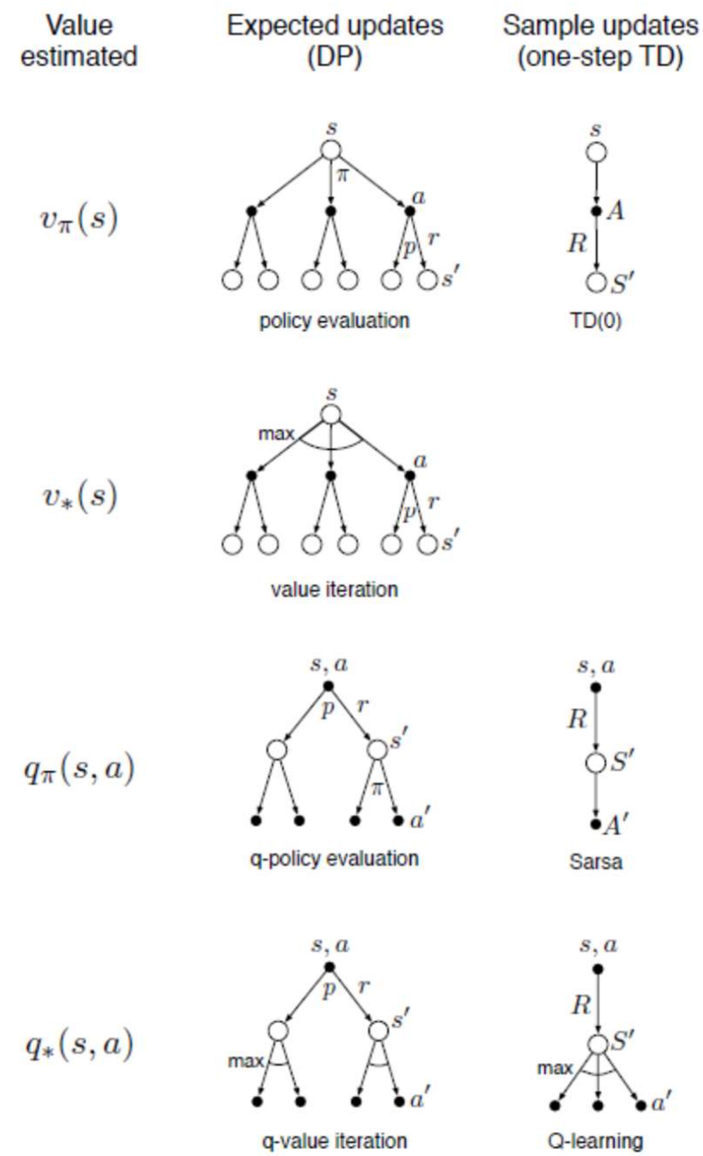
Dyna-Q does not search for a new path

Expected vs Sampled Updates

Diagrams of one-step updates

- Belman Expectation for $v_\pi(s)$
  - Iterative Policy Evaluation (DP)
  - TD Learning (Sampling)
- Belman Expectation for $q_\pi(s,a)$
  - Q-Policy Iteration (DP)
  - Sarsa (Sampling)
- Belman Optimality for $q^*(s,a)$
  - Q-Value Iteration (DP)
  - Q-Learning (Sampling)

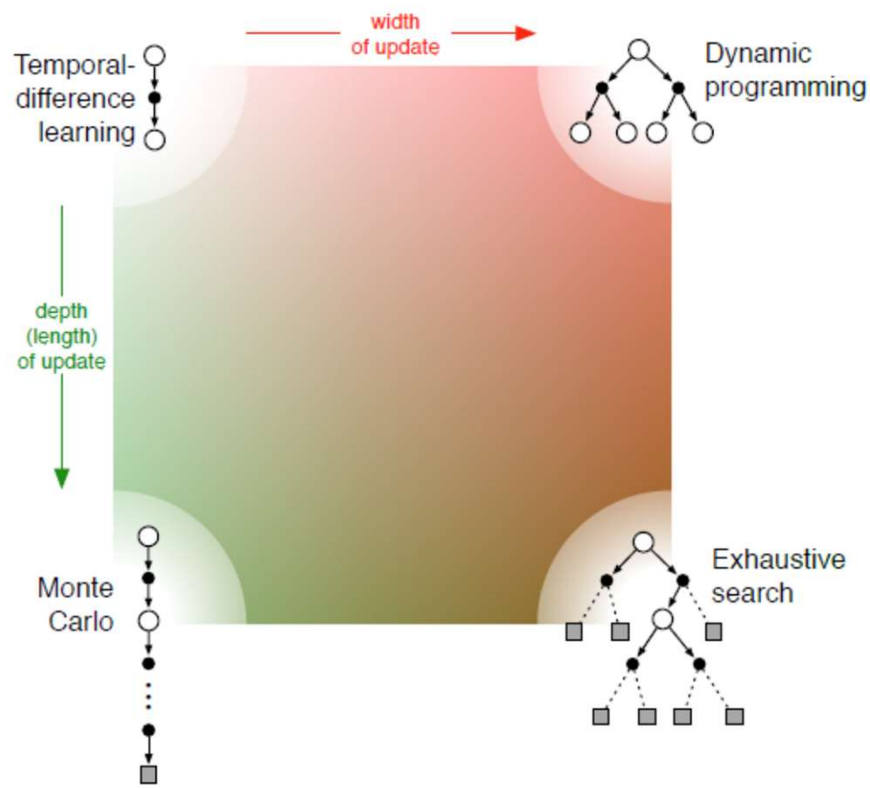
Expected = Model-based, Dynamic Programming
Sampled = Model-free, Learning



DP vs TD

Relationship between DP and TD

- Belman Expectation for $v_\pi(s)$
  - Iterative Policy Evaluation (DP)
  - TD Learning (Sampling)
- Belman Expectation for $q_\pi(s,a)$
  - Q-Policy Iteration (DP)
  - Sarsa (Sampling)
- Belman Optimality for $q^*(s,a)$
  - Q-Value Iteration (DP)
  - Q-Learning (Sampling)



Problems of RL

Curse of Dimensionality
In real world problems ist difficult/impossible to define discrete state-action spaces.

(Temporal) Credit Assignment Problem
RL cannot handle large state action spaces as the reward gets too much dilited along the way.

Partial Observability Problem
In a real-world scenario an RL-agent will often not know exactly in what state it will end up after performing an action. Furthermore states must be history independent.

State-Action Space Tiling
Deciding about the actual state- and action-space tiling is difficult as it is often critical for the convergence of RL-methods. Alternatively one could employ a continuous version of RL, but these methods are equally difficult to handle.

Non-Stationary Environments
As for other learning methods, RL will only work quasi stationary environments.