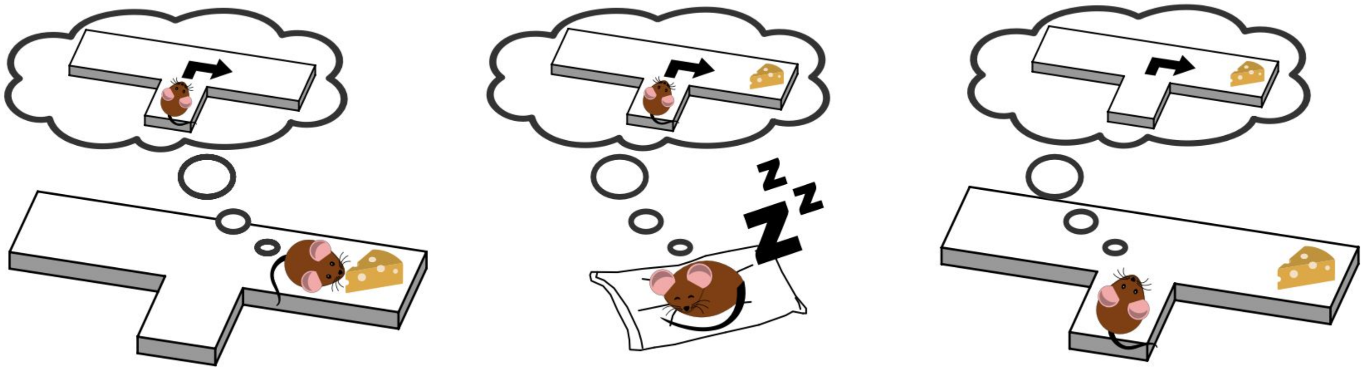
Learning and Planning in Reinforcement Learning



Umesh Singla

Oct 30, 2023

Fig. Mattar and Daw, *Nature Neuroscience*, *2018.*

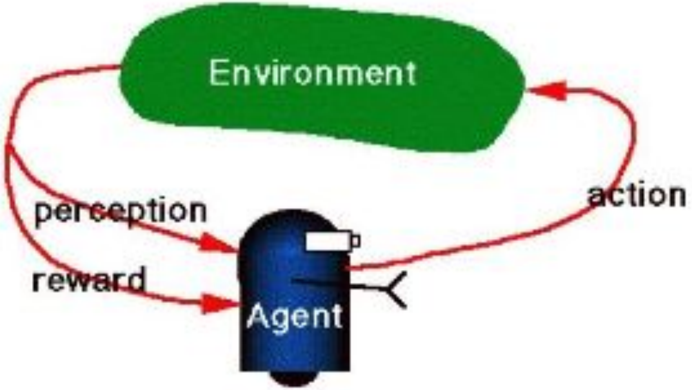
Outline

- Introduction
  - Problem of RL
  - RL Framework: MDP
  - Value Functions
- Model-free RL
  - Temporal Difference learning
  - Q-learning
- Model-based RL
  - Learning and Planning: Dyna-Q

Reinforcement Learning

How can autonomous decision-making agents learn from experience in the world?

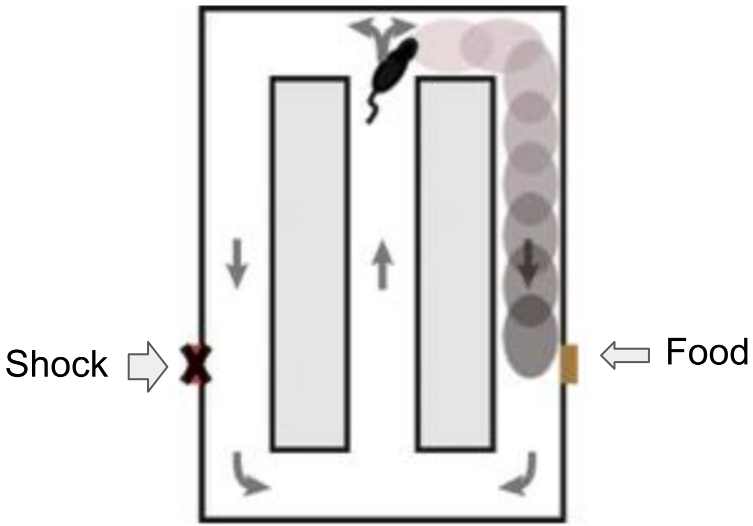
Reward: Food or shock



Reward: Numbers

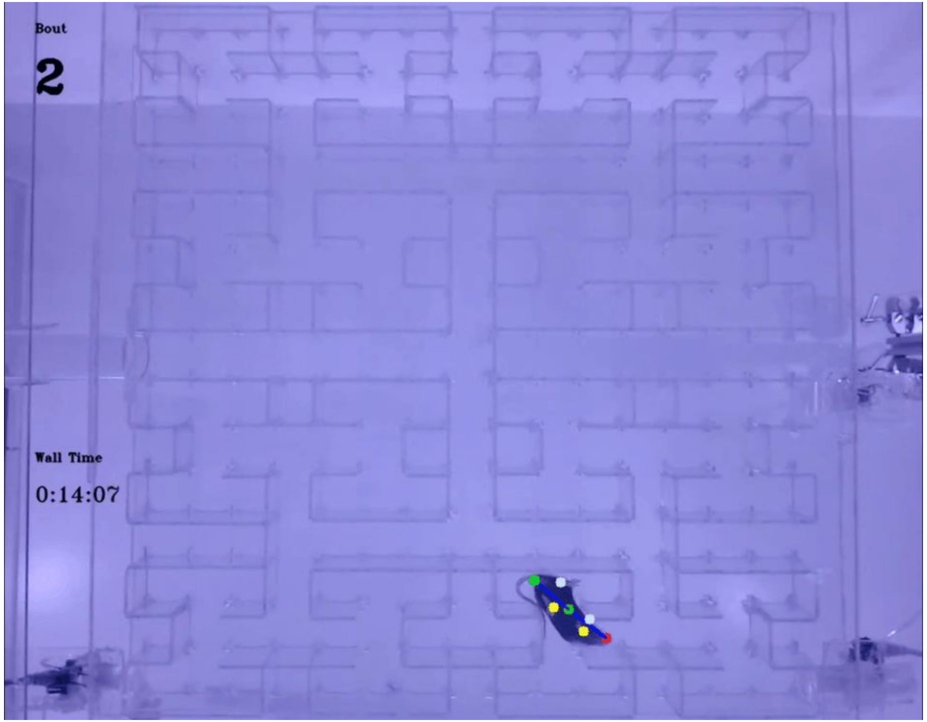
Fig. Precup, D. *Neuromatch Academy.*

Reinforcement Learning



Gupta et al (2010); Wikenheiser & Redish (2014)

Reinforcement Learning



Value Learning

World Learning

Decision-Making

Exploring

Planning

…

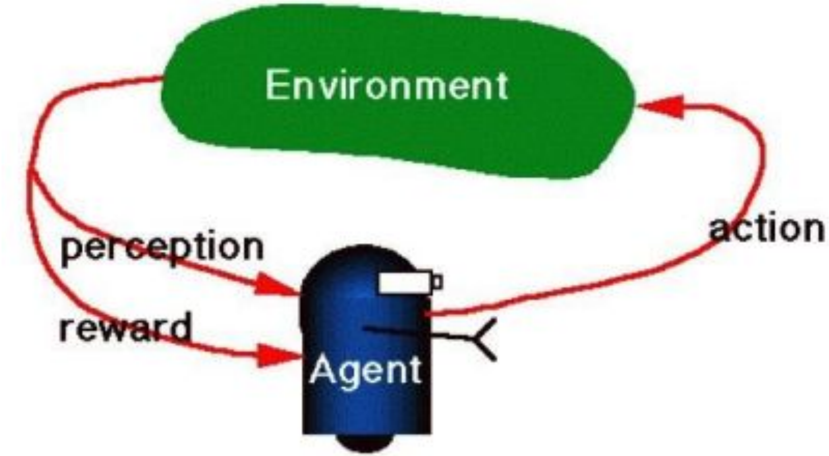
Rosenberg et al., 2022

Challenges of RL

- How to learn in noisy environments?
- When to explore, versus when to exploit?
- How to learn from delayed and immediate rewards?
- How to navigate complex worlds with tractable manageable models?
- How to prove computational guarantees (e.g., convergence, optimality)?

Computational framework for RL

How do we formalize this process? How do we handle uncertainty?



We define a **Markov decision process**.

Adapted from: Precup, D., DeWitt, E. and Mattar, M.G, *Neuromatch Academy*

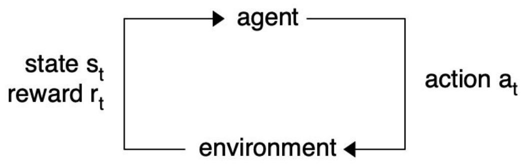
Computational framework for RL

- At every time step *t*, the agent perceives the **state** of the environment.
- Based on this perception, it chooses an **action**.
- The action causes the agent to receive a numerical **reward**, and the agent uses this information to *improve* its future actions.
- Objective: Find a way of choosing actions, called a policy which **maximizes the agent's long-term expected return**.

Adapted from: DeWitt, E. and Mattar, M.G, *Neuromatch Academy*

MDP

A Markov decision process (**MDP**) is defined by the following:



- A **state space** *S*
- An **action space** *A*
- **Transition probabilities**

  $$P(s' \mid s, a) = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

  that indicate, at any time *t*, how frequently an agent moves from state *s* to state *s'* after taking action *a*.
- A **reward function** *R(s, s', a)*, providing immediate feedback when the agent takes action *a* in state *s* and moves to state *s'*.
  Rewards are **scalar**: the higher, the better.

$$\text{MDP} = \{\mathcal{S}, \mathcal{A}, P(s'|s,a), R(s,s',a)\}$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Example

s* ∈ *S board position and results of roll of dice

a* ∈ *A one of any allowed moves

R(s) =
- +1 if agent wins the game
- -1 if agent loses the game
- 0 for all previous positions

***P(s' | s, a)* ~**
- agent moves
- opponent rolls dice
- opponent moves
- agent rolls dice

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Policy and their expected returns

A **policy** $\pi : S \rightarrow A$ is a mapping of states to actions.

Experience under policy π

$$\begin{array}{lcccccc} \text{state} & s_0 & \xrightarrow{\pi(s_0)} & s_1 & \xrightarrow{\pi(s_1)} & s_2 & \cdots \\ \text{reward} & r_0 & & r_1 & & r_2 & \cdots \end{array}$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Expected Return i.e. Value

$$\mathrm{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R(s_t)\,\middle|\,s_0=s\right] =$$

the expected value of the

discounted ***infinite-horizon*** *return,*

starting in state s *at time t=0,*

and following ***policy π****.*

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Value Functions

$$V^{\pi}(s) \;=\; \mathrm{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \,\middle|\, s_0 = s\right]$$

expected return,

starting in state s,

following policy $\boldsymbol{\pi}$

$$Q^{\pi}(s, a) \;=\; \mathrm{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \,\middle|\, s_0 = s, a_0 = a\right]$$

expected return,

starting from state s,

taking action a,

then following policy $\boldsymbol{\pi}$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Bellman equation for State Value function

$$
\begin{aligned}
V^{\pi}(s) &= \mathrm{E}^{\pi}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s \right] \\
&= R(s) \;+\; \gamma \sum_{s'} P(s'|s,\pi(s))\, \mathrm{E}^{\pi}\left[R(s_1) + \gamma R(s_2) + \cdots \,\middle|\, s_1 = s' \right] \\
&= R(s) \;+\; \gamma\, \mathrm{E}^{\pi}\left[R(s_1) + \gamma R(s_2) + \cdots \,\middle|\, s_0 = s \right] \\
&= R(s) \;+\; \gamma \sum_{s'} P(s'|s,\pi(s))\, V^{\pi}(s')
\end{aligned}
$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Bellman equations for V and Q

$$V^{\pi}(s) \;=\; R(s) \;+\; \gamma \sum_{s'} P(s'|s,\pi(s))\, V^{\pi}(s')$$

$$Q^{\pi}(s,a) \;=\; R(s) \;+\; \gamma \sum_{s'} P(s'|s,a)\, V^{\pi}(s')$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Model-free approach

Consider the **model** *{S, A, P(s'|s,a), R(s)}* defined by an **MDP**.

If we know the model, we can learn and plan using policy or value iteration.

But what if we don’t know the model i.e. *P(s'|s,a)* and *R(s)*?

Can we learn the value function or optimal policy *directly from experience*?

Temporal Difference error

How to estimate the mean of a random variable X from IID samples?

$x_{1},\ x_{2},\ x_{3},\ x_{4},\ x_{5},\ x_{6},\ x_{7},\ x_{8},\ x_{9},\dots$

We do **incremental update**:

Initialize: $\mu_0 = 0$

Update: $\mu_t = (1 - \alpha)\,\mu_{t-1} + \alpha x_t \quad \text{for } \alpha \in (0, 1)$

The update is a convex sum of the old estimate and latest sample. It can also be written as:

$$\mu_t = \mu_{t-1} + \alpha\,(x_t - \mu_{t-1})$$

The corrective term $(x_t - \mu_{t-1})$ is known as **temporal difference**.

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Temporal Difference (TD) learning

How to estimate $V^{\pi}(s)$ from experience without knowing $P(s' \mid s, \boldsymbol{\pi}(s))$?

Bellman Equation (with model):

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi(s)) V^{\pi}(s')$$

Temporal difference estimation (without model):

$$\textit{Initialize:} \quad V_0(s) = 0 \quad \text{for all} \quad s \in \mathcal{S}$$

$$\textit{Update:} \quad V_{t+1}(s_t) = \underbrace{V_t(s_t)}_{\text{previous estimate}} + \alpha \Big[\underbrace{r_t + \gamma V_t(s_{t+1})}_{\text{TD target}} - V_t(s_t) \Big]$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

TD & Q-learning

TD learning:

$$\textit{Initialize:} \quad V_0(s) = 0 \quad \text{for all} \quad s \in \mathcal{S}$$

$$\textit{Update:} \quad V_{t+1}(s_t) = \underbrace{V_t(s_t)}_{\text{previous estimate}} + \alpha\Big[\underbrace{r_t + \gamma V_t(s_{t+1})}_{\text{TD target}} - V_t(s_t)\Big]$$

Q-learning:

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{previous estimate}} + \alpha\Big[\underbrace{r_t + \gamma \max_{a'} Q_t(s_{t+1}, a')}_{\text{TD target}} - Q_t(s_t, a_t)\Big]$$

Adapted from: Lawrence Saul, *Probabilistic Reasoning and Decision-Making*. UC San Diego 2020

Q-learning

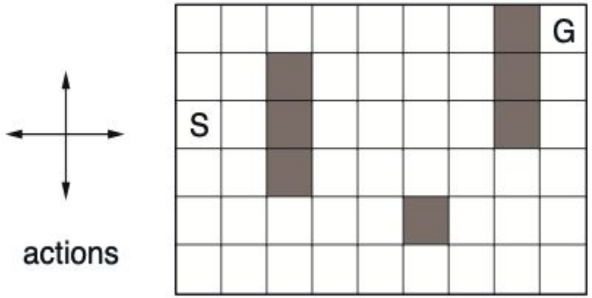
Initialize $Q(s, a)$ for all $s \in S$ and $a \in A$.

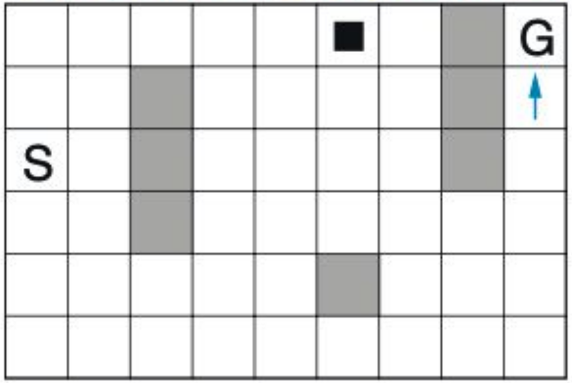
Loop for each episode:

A. $S \leftarrow$ current (non-terminal) state
B. $A \leftarrow$ greedy(S, Q)
C. Take action A; observe resultant reward, R, and state, S'
D. $Q(S, A) \leftarrow Q(S, A) + \alpha \, [R + \gamma \max_a Q(S', a) - Q(S, A)]$

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Example: Q-learning





One-step update per episode

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Learning vs Planning

Jensen et al., 2023. *bioRxiv*.

Learning vs Planning

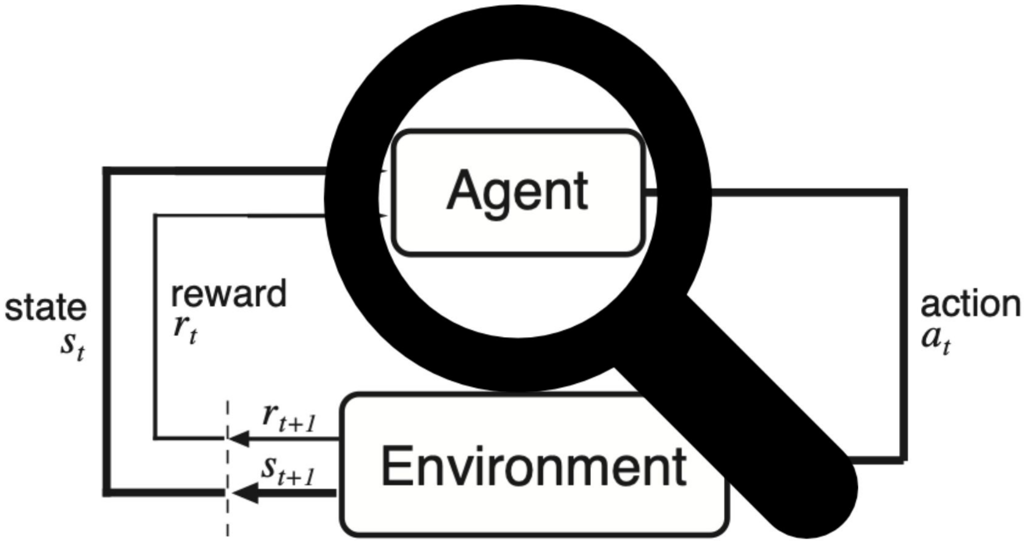
Instead of learning values from experience, ***planning*** is the process of computing action values from a model.

Model-free

Model-based

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

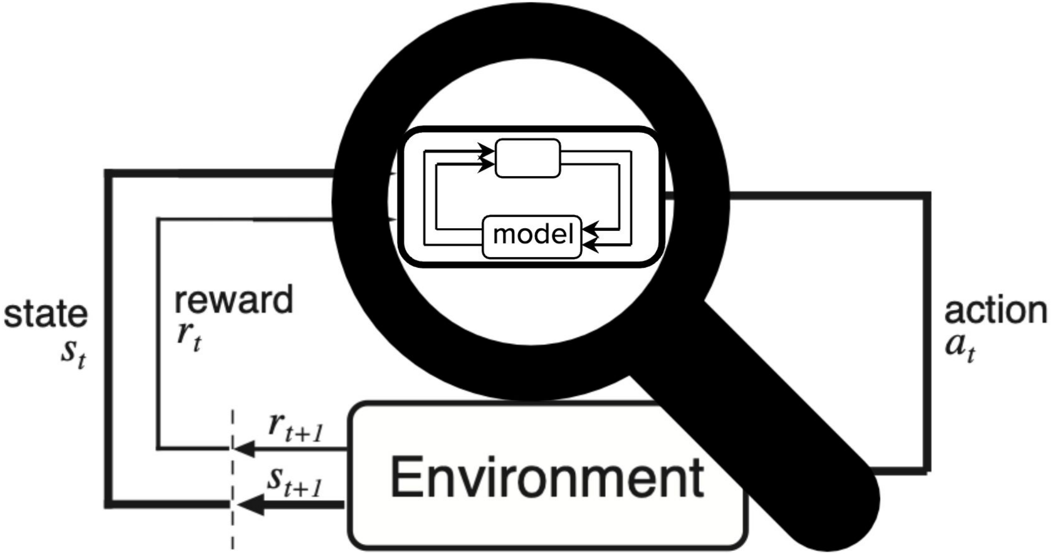
What is a model?



Adapted from: DeWitt, E. and Mattar, M.G, *Neuromatch Academy*

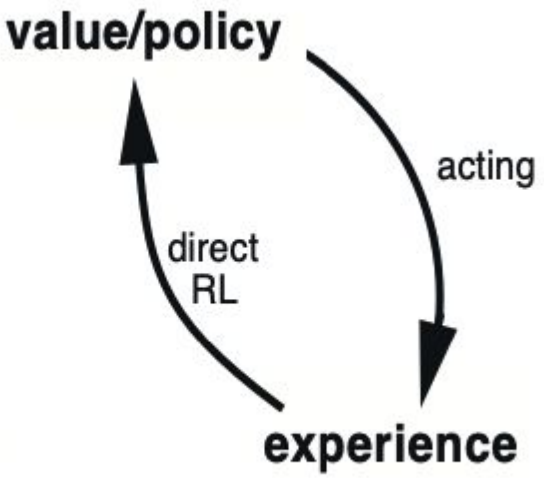
What is a model?

A model is a representation of how the world will respond to the agent's actions.



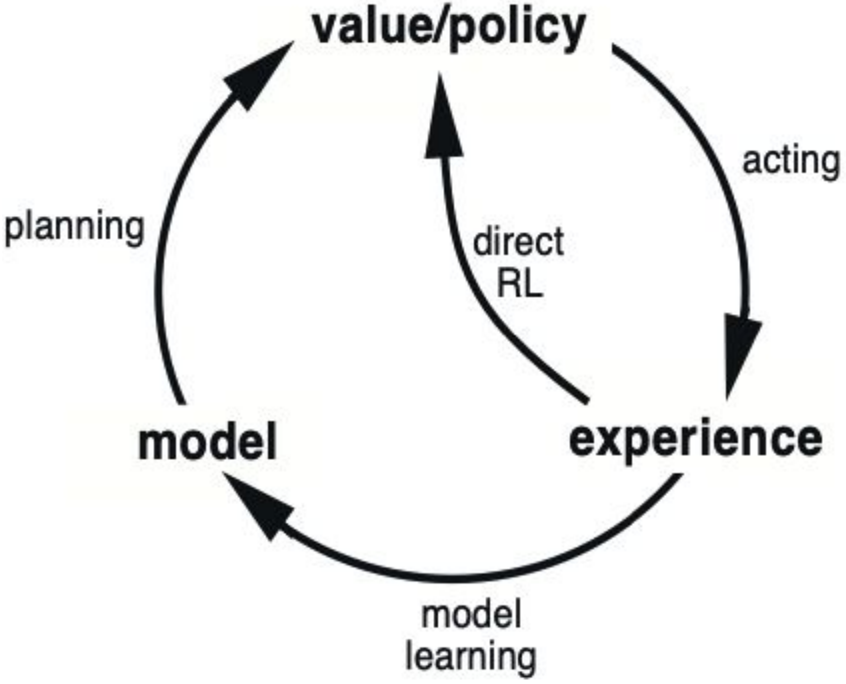
Adapted from: DeWitt, E. and Mattar, M.G, *Neuromatch Academy*

Dyna-Q: Adding *planning* to Q-learning agent



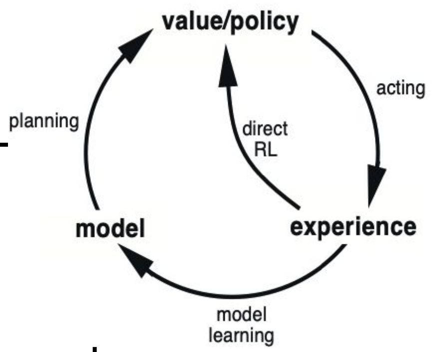
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Dyna-Q: Adding *planning* to Q-learning agent



Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Dyna-Q



Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in S$ and $a \in A$.

Loop forever:

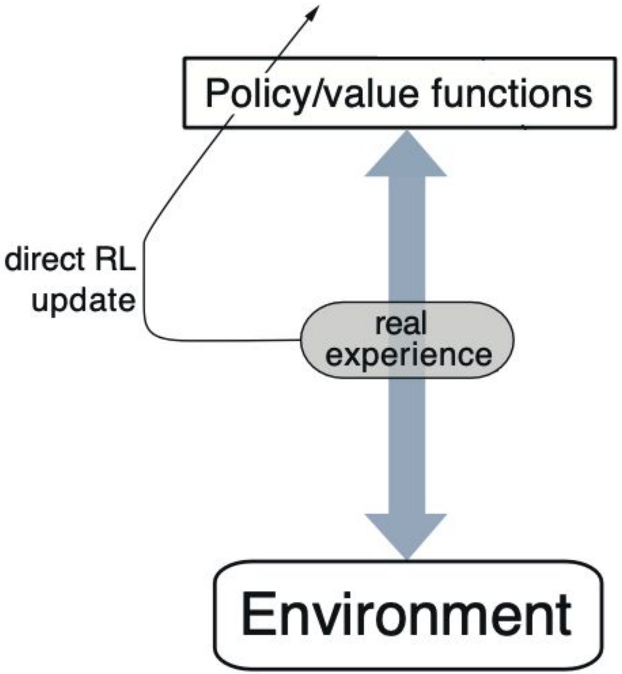
A. $S \leftarrow$ current (nonterminal) state
B. $A \leftarrow$ greedy(S, Q)
C. Take action A; observe resultant reward, R, and state, S'
D. $Q(S, A) \leftarrow Q(S, A) + \alpha\, [R + \gamma \max_a Q(S', a) - Q(S, A)]$
E. $Model(S, A) \leftarrow R, S'$
F. Loop repeat ***n*** times:
   a. $S \leftarrow$ random previously observed state
   b. $A \leftarrow$ random action previously taken in S
   c. $R, S' \leftarrow Model(S, A)$
   d. $Q(S, A) \leftarrow Q(S, A) + \alpha\, [R + \gamma \max_a Q(S', a) - Q(S, A)]$

} **model learning** (steps D–E)

} **planning** (step F)

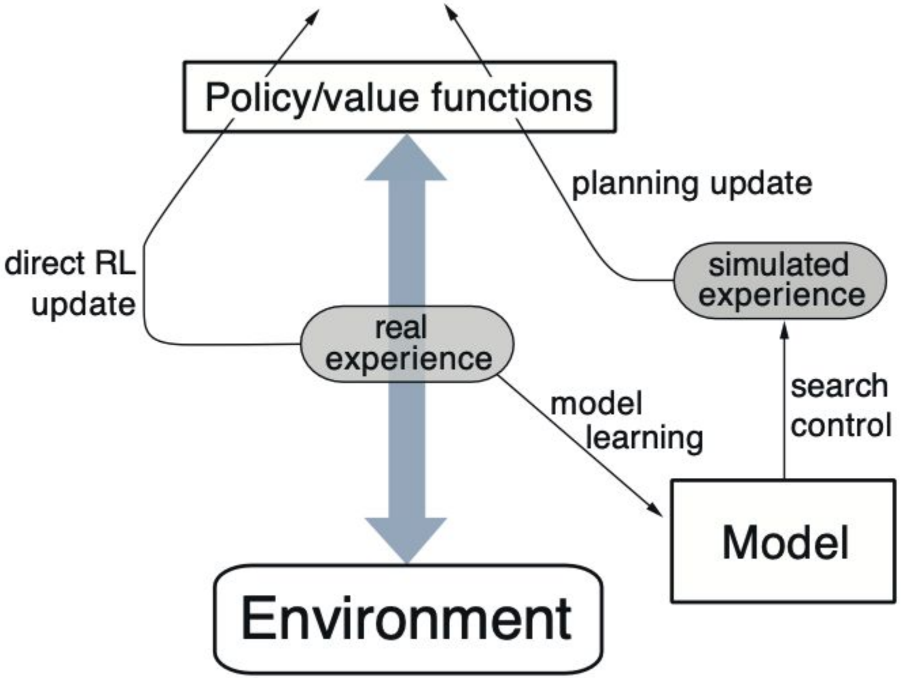
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Dyna-Q



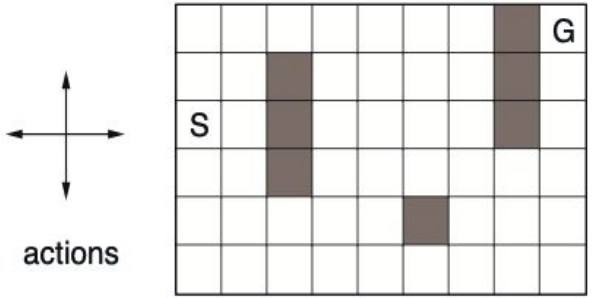
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

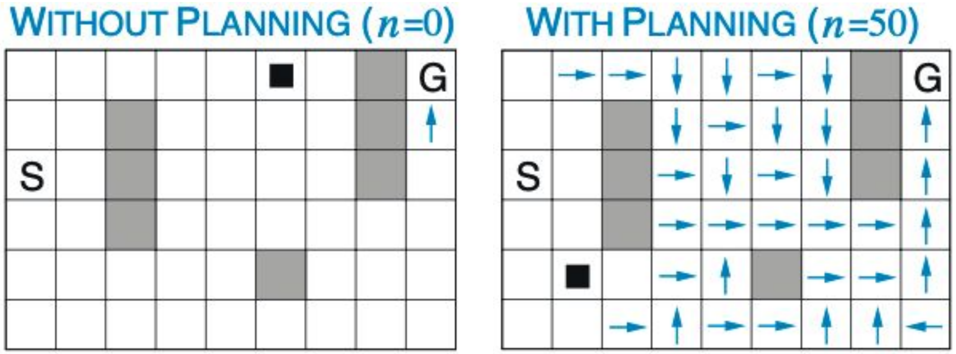
Dyna-Q

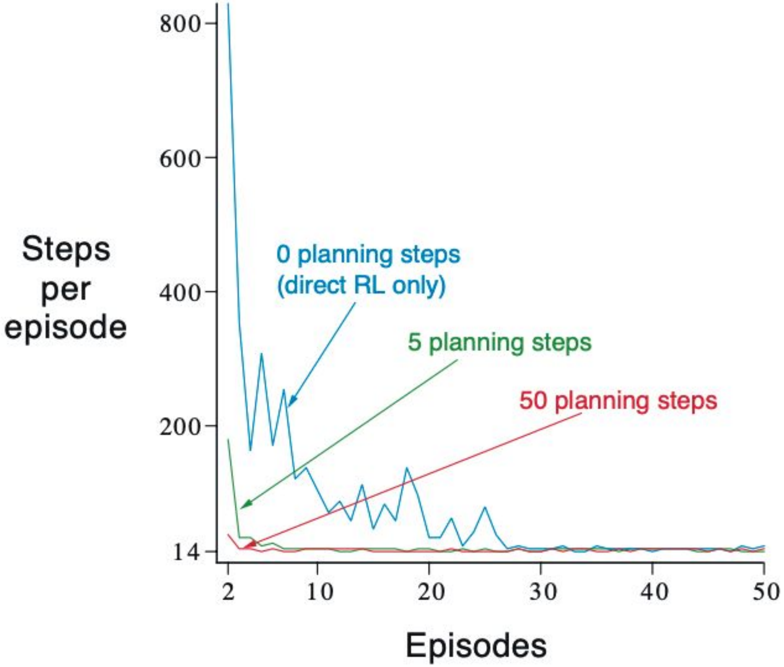


Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*

Example







Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.*