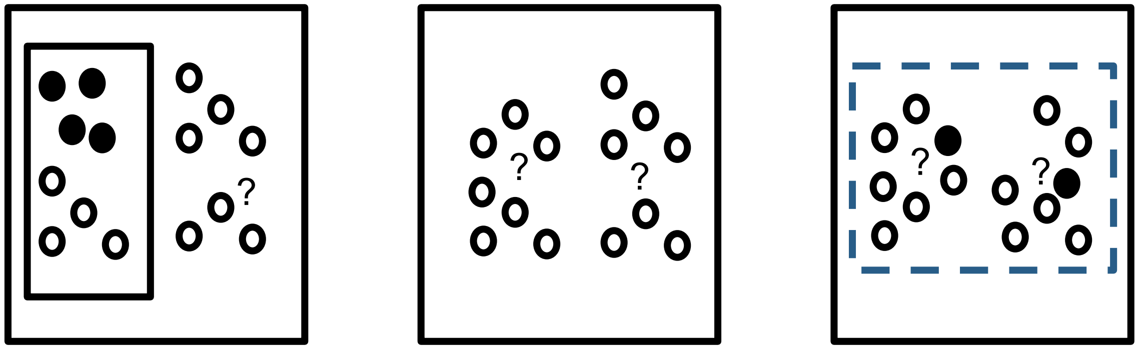
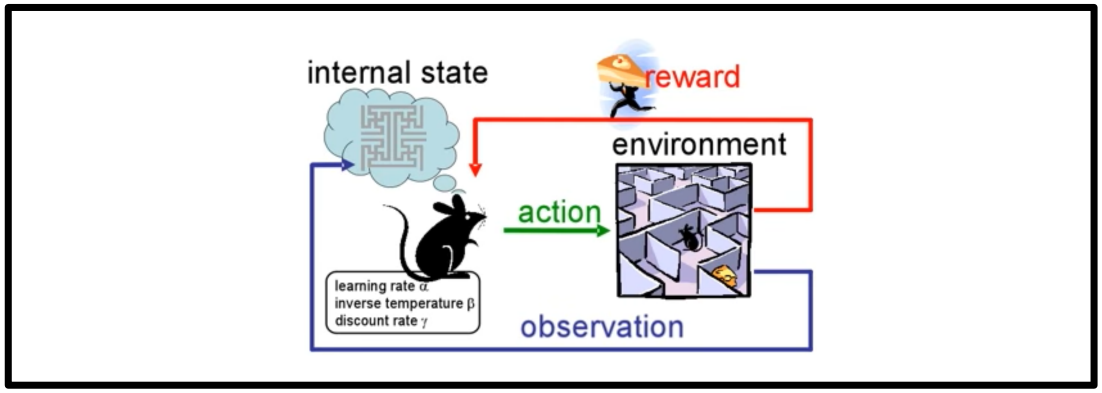
Machine Learning
CSCE 4205 / 5215

Reinforcement Learning


?
?
?
?
?


internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation

Definition

Reinforcement learning: an area of machine learning, inspired by behaviorist psychology, concerned with how software **agents** ought to take ***actions*** in an *environment* so as to maximize some notion of cumulative ***reward.***

- No explicit teacher/trainer, only a reward signal
- Focus on on-line performance
- Exploration vs exploitation tradeoff

Basic idea of RL

Capture the most aspects of the real problem facing the learning agent interacting over time with its environment to achieve a goal.

A learning agent must be able to

- sense the state of its environment to some extent
- take actions that affect the state
- have goal(s) relating to state of the environment

Unchartered territory

Learn from its own experience

Maximize reward signal instead of trying to find a hidden structure

Challenge in RL

- Tradeoff between Exploration and Exploitation

- Prefer actions tried in the past which produced effective reward

- To do this, try actions not selected before

- Agent must exploit what it already experienced to obtain reward, but also explore to make better action selections in future

Model-based reinforcement learning

- Model-based: adjust **value functions** immediately upon new information about the environment or the animal’s internal state
    - e.g. playing chess well requires simulating future moves and anticipating opponent’s plans
    - Many competitive/cooperative games are more quickly optimized similarly using a theory of mind
    - Also, rewards may be a function of the animal’s motivational state, which may require a model
        - e.g. twice as much food may not be twice as good!

Defining the Value of a state

A policy π is a mapping from each state **s** and action **a** to the probability $\pi(s, a)$ of taking action **a** when in state **s**.

Informally, the value of a state **s** under a policy π, denoted $V\pi_s$, is the expected return when starting in s and following π thereafter.

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

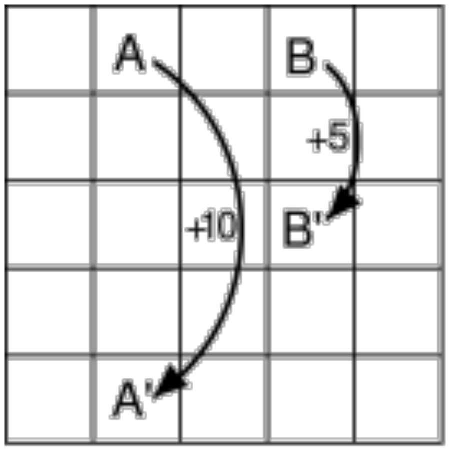
Value functions

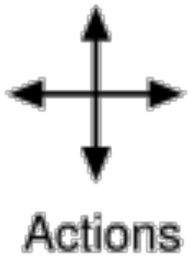
Functions of states (or of state-action pairs) that estimate *how good* it is for the agent to be in a given state (or how good it is to perform a given action in a given state).

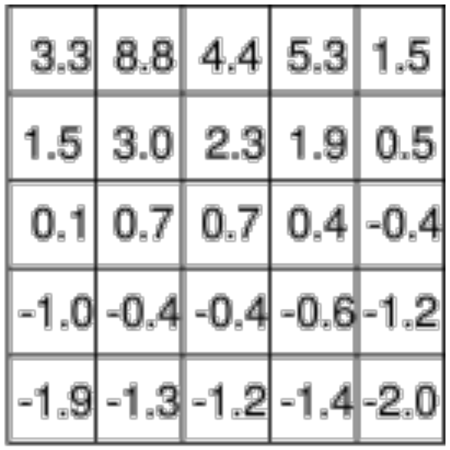
The notion of "how good" is defined in terms of **future rewards** that can be expected, or, to be precise, in terms of **expected return**.

The rewards that the agent can expect to receive in the future depend on what actions it will take.

Example: Gridworld







3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(a)

(a) exceptional reward dynamic

(b)

(b) state-value function for the equiprobable random policy

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

Model-free Reinforcement learning

What we will primarily discuss today

Features of model-free RL:

- Update directly from reward prediction error
- No simulating or estimating futures
- Requires lots of experience
- Models habit learning
- Strong neurophysiological correlates

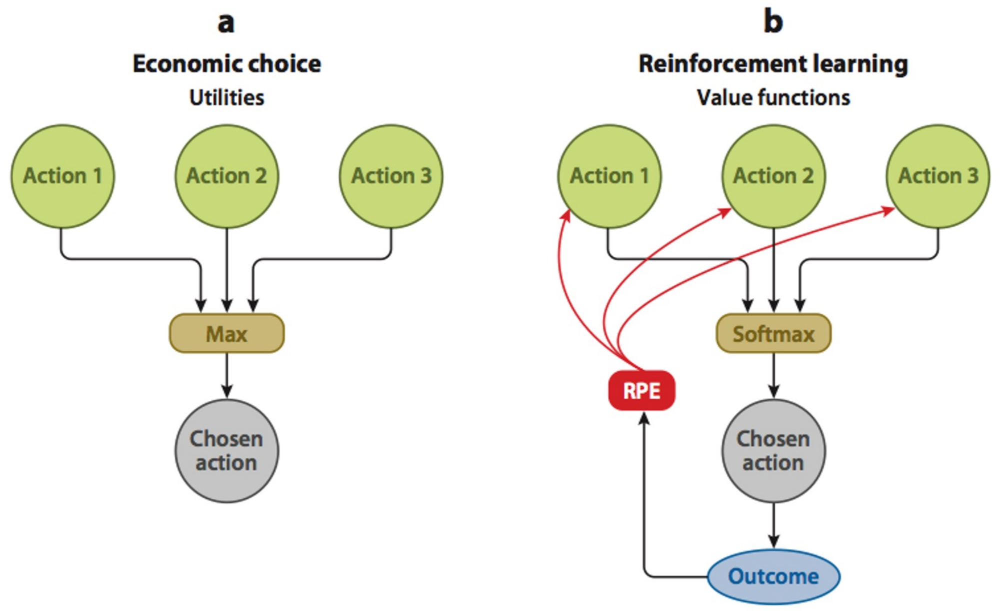


Figure 1

Economic and reinforcement learning theories of decision making. (*a*) In economic theories, decision making corresponds to the selection of an action with maximum utility. (*b*) In reinforcement learning, actions are chosen probabilistically (i.e., softmax) on the basis of their value functions. In addition, value functions are updated on the basis of the outcome (reward or penalty) resulting from the action chosen by the animal. Abbreviation: RPE, reward prediction error.

Reinforcement Learning

(Sutton & Barto 1998)

- Goal: Maximize future rewards through...
- Valuation of actions
- Probabilistic action selection
- Update on value from prediction error

Basic Reinforcement model

1. a set of environment states, S
2. a set of actions, A
3. rules* of transitions between states
4. rules* that determine a direct scalar reward
5. rules* that govern observations

Reward policy

* Rules may be stochastic (randomly determined)

Note, much of human behavior, animal behavior, robot behavior, and even abstract decision making can be formulated this way

Example: Autonomous driving

- In usual circumstances we would require an autonomous vehicle to

$$\left.\begin{array}{l}\text{put safety first}\\ \text{minimize ride time}\\ \text{reduce pollution}\\ \text{offer passengers comfort}\\ \text{obey the road rules}\end{array}\right\}\ \textbf{Goals}$$

- With an autonomous **race** car, we would emphasize speed much more than the driver's comfort.
- The programmer cannot predict everything that could happen on the road. Instead of building lengthy "if-then" instructions, the programmer
-

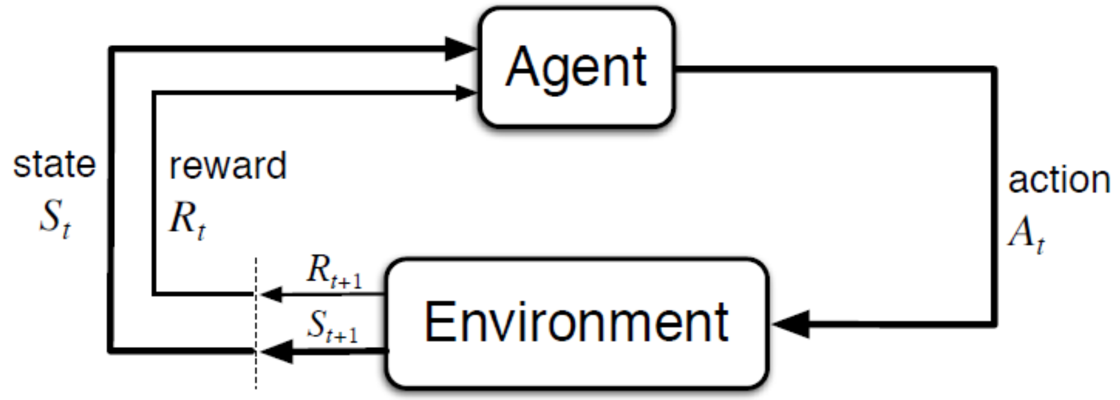
Prepares the reinforcement learning agent to be capable of learning from the system of rewards and penalties.

The agent (another name for reinforcement learning algorithms performing the task) gets rewards for reaching specific goals.

Deep RL for motion planning

The Agent-Environment Interface

- Agent = the learner and decision-maker
- Environment = everything outside the agent that the agent interacts with the agent selects actions and the environment responds to these actions and presents new situations to the agent



Interaction happens continually

Alternately..

- Actions can be any **decisions we want to learn how to make**
- States can be **anything we can know** that might be useful in making decisions
- The general rule we follow is: Anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment.
- In practice, the agent-environment boundary is determined once one has selected particular states, actions, and rewards, and thus has identified a specific decision-making task of interest.

The Markov Property

- In the reinforcement learning framework, the agent makes its decisions as a function of a signal from the environment called the environment's state.
- What is required of the state signal?
- What kind of information should we expect and not expect it to provide?
- In particular, we define a property of environments and their state signals called the **Markov property**.

What is required of the state signal?

- State signal should include immediate sensations such as sensory measurements
- State representations can be **highly processed versions** of original sensations, or they can be complex structures built up over time from the sequence of sensations

For e.g.:
1. We can move our eyes over a scene, with only a tiny spot corresponding to the fovea visible in detail at any one time, yet build up a rich and detailed representation of a scene

2. We can look at an object, then look away, and know that it is still there

3.We can hear the word “yes" and consider ourselves to be in totally different states depending on the question that came before and which is no longer audible

What should the state signal provide?

- The state signal should not be expected to inform the agent of everything about the environment, or even everything that would be useful to it in making decisions
- We need a state signal that **summarizes** past sensations compactly in such a way that all relevant information is retained. This normally requires more than the immediate sensations, but never more than the complete history of all past sensations. (This is the ideal)

A state signal that succeeds in retaining all relevant information is said to be Markov, or to have the Markov property.

The Markov property is important in reinforcement learning because decisions and values are assumed to be a function only of the current state

A reinforcement learning task that satisfies the Markov property is called a Markov decision process, or MDP

MDP - markov decision process

An agent interacts with the environment in discrete time steps

1. at time t
2. in state s_t
3. agent receives observation o_t
4. with a reward r_t
5. chooses an action a_t
6. environment moves to a new state s_{t+1} ...

“Markov” of MDPs

- The probability of outcomes and observations depends only on the current state (and not prior states)
    - “History doesn’t matter”? Not exactly
    - Modeling a soda dispenser (which coins irrelevant, just the $)
        - X = $ inserted already
    - Modeling ball trajectory? (much physics is assumed an MDP)
        - $X = p, v_t, v_r$
    - Modeling building of trust over time? (include personality stats)
        - X = trustworthiness
- Markov problems are MUCH easier to compute over since history is factored out, and there are many algorithms to help do this optimally.

Picking the best action

The best action **maximizes future reward**.

But we get a delayed reward, quite often

One solution: estimate the value of states (and actions in those states)

- Exploiting your knowledge:
  - Pick the action that is currently the most valued
- Exploring (given uncertainty in your knowledge)
  - Pick a reasonably good action, with some randomness mixed in (e.g. softmax)

Value functions

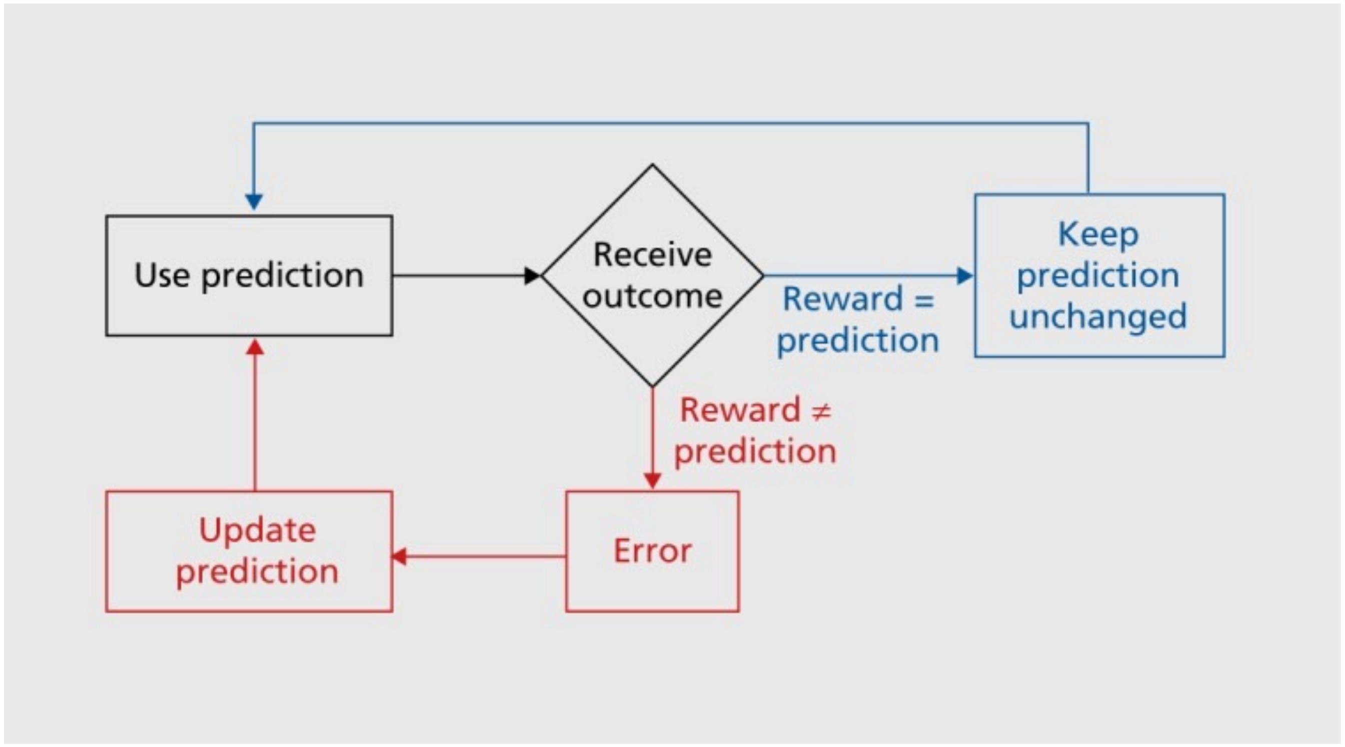
Value function - the expected sum of future rewards.

- $Q(s,a)$ = action value function
- $V(s)$ = state value function
- $V(s) = \sum_a p(a)\, Q(s,a)$

But how to estimate this? We need a few more concepts:

- Reward prediction error
- Learning rates
- Temporal discounting

Learning by prediction error aka
We often learn by making mistakes!!!



https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4826767/

Reward prediction error

- RPE = actual - predicted reward value
- An “error signal” used to adjust the value function
- New Value = Old Value + α * RPE
  - α = the learning rate, discussed next
- Predicted reward value (‘pr’ below):
  - if **$Q(s_t, a_t)$** is the sum of expected future rewards at time t ($pr_1 + pr_2 + pr_3 + \ldots$) and **$Q(s_{t+1},a_{t+1})$** is the sum of expected future rewards at time t+1 ($pr_2 + pr_3 + \ldots$) then
  - $Q(s_t,a_t) = pr_t + best_a\ Q(s_{t+1})$
  - Best $Q(s_{t+1}) = max_a(Q(s_{t+1},a))$
  - So...
  - pr_t (predicted reward value) = $Q(s_t,a_t) - max_a(Q(s_{t+1},a))$

Adapting the learning rate, α

You want to learn to minimize the error using an update, but at a learning rate, α, that allows your value function to

- **Learn quickly** (high α) and adapt to changing environments
- But **remain stable enough** (low α) to noise and stochastic rewards to avoid forgetting
  - protip: low α averages stochastic rewards over time!

The right value of α depends upon how stable your environment is

- More unstable or critical → use a higher α
  - Goes for unstable model (naive learner) or unstable environment
  - Life/death learning moments justify high α, but such adjustments can backfire (e.g. PTSD, trauma-induced illnesses)
- Note, unstable ≠ stochastic
  - fixed probabilities are stochastic, but can be highly stable
  - Alternatively, you can have non-stochastic rules that change, leading to an unstable environment

Temporal discounting

If you lived forever, a good strategy is to learn every language, every subject... then exploit that knowledge. But you don't live forever. You don't want your algorithms to act like they operate forever…

RL Models introduce a **discount factor γ** between 0 and 1 to reward value

- $\gamma = 1$ represents no discounting
  - rewards tomorrow are equivalent to rewards today
- $\gamma \rightarrow 0$ leads to "hedonistic behavior".
  - seeking immediate reward at the expense of more long-term reward

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

Q-learning

Put all of that together:

- Markov model
- State-action value function
- Reward prediction error
- Learning rate adaptation
- Temporal discounting

and you get a powerful model-free learning method that powers much of machine-based and neuroscience-related reinforcement learning

- Q learning!

Q-learning

high values = pleasure

high values = pleasant anticipation

$$Q_{t+1}(s_t,a_t) = \underbrace{Q_t(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t,a_t)}_{\text{old value}}\right]$$

low values = pain

low values = fear

This comes from:

Updated value = old value + α (desired - old)

Q_{t+1}(state s, action a) = Q_t + α [R_{t+1} - [Q_t - γ $\max_a$ Q_t(next state, action a)]]

Summary

- The basic reinforcement learning framework
  - Markov decision process
- Value functions
- Model-free Q-learning involving:
  - Reward Prediction Error
  - Temporal Discounting