Reinforcement Learning

M. Plüss, E. Arnold

University of Basel

Mai 13, 2019

Section 1

Reinforcement Learning

Reinforcement Learning - Idea

- Agent takes actions in an interactive environment (reward)
- Learning how good/bad actions are
- Goal: Maximize positive rewards
- Close relationship with biology, psychology and neuroscience

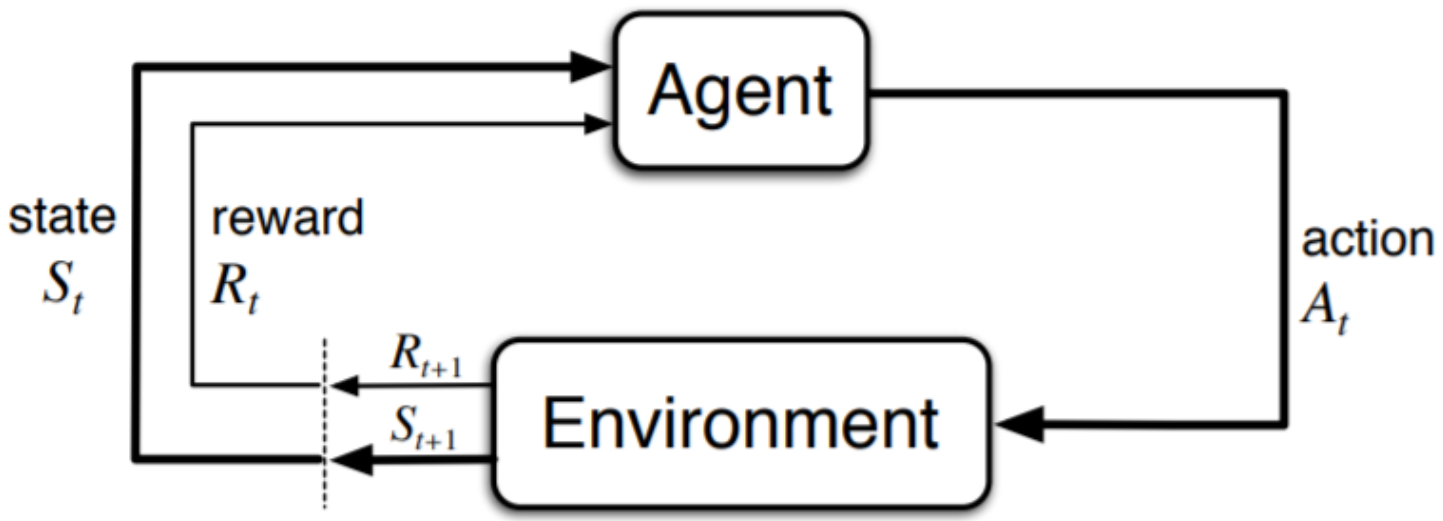


Image source: Sutton, Barto (1998)

Reinforcement Learning - Example

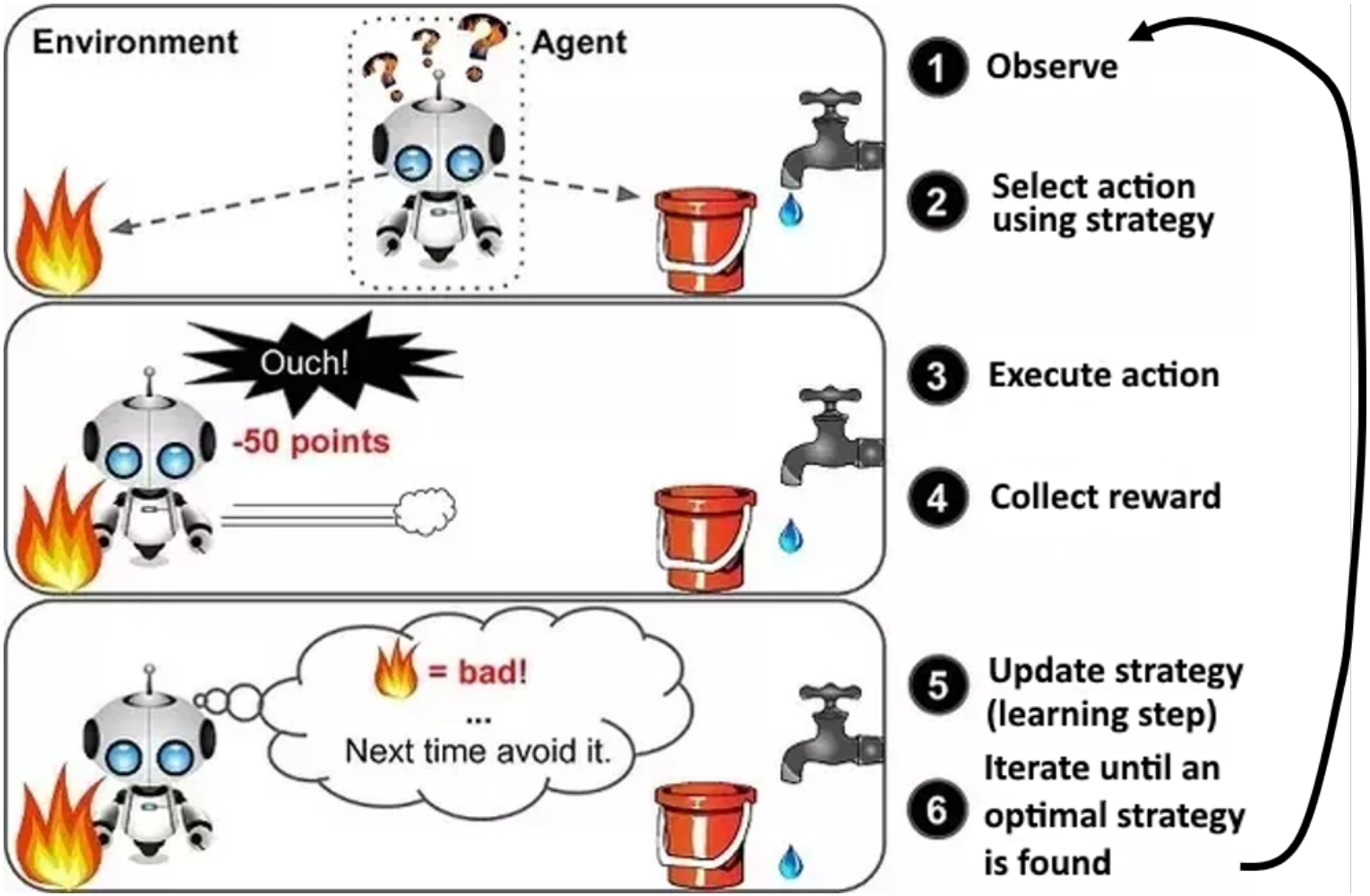


Image source: A. Géron for Safari Books Online

Difference to [Un-] Supervised-Learning

- Absence of an expert/teacher which labels input
- Agent has to distinguish good/bad behaviour by itself
- For that, the agent needs feedback from each state/input $\rightarrow$ **Reward** or **Reinforcement**

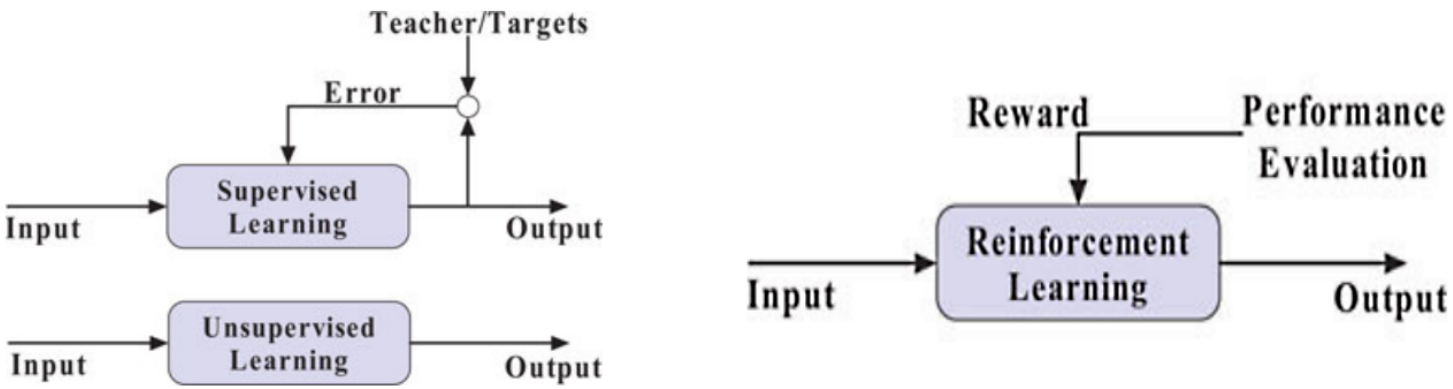
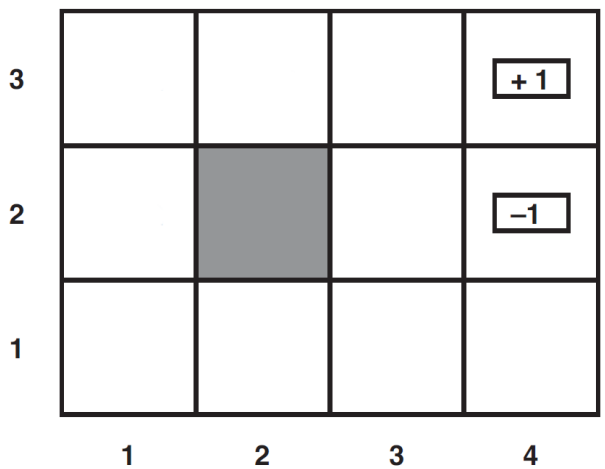


Figure: Reinforcement Learning compared to other learning strategies.

Image source: Wang, Babuska (2012)

Reminder - Grid World

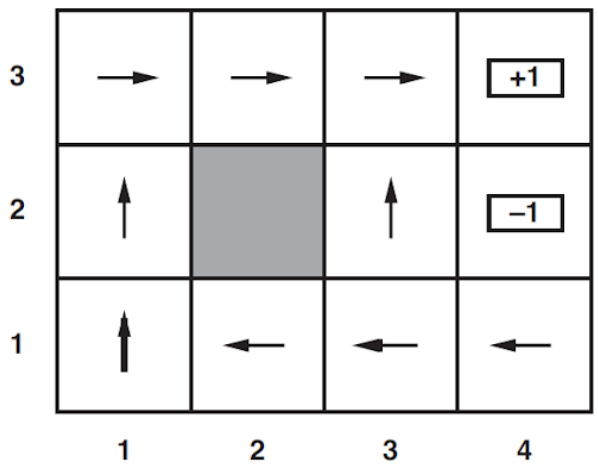
- Agent starts from (1,1) (start state)
- Every state s gives a reward R of $R(s) = -0.04$ except the goal states $R((4,3)) = +1$ and $R((4,2)) = -1$
- Transition probabilities:
  - 80% chance of desired direction
  - 10% chance of going left
  - 10% chance of going right



Reminder - Terminology

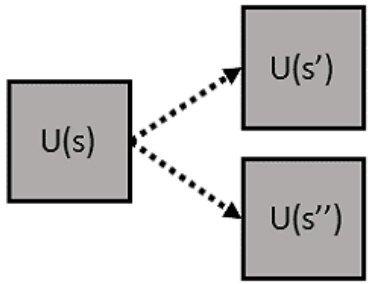
- **Trial**: One action sequence from the start state to a terminal state
- **Policy**: Instructions on which actions the agent should take in a given state (arrows)

Policy: $\pi(s) \rightarrow a$



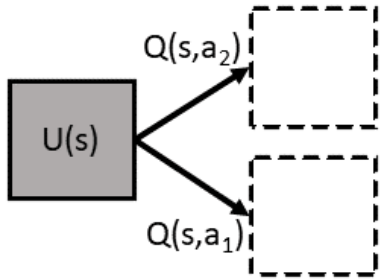
Agent design

Utility-based Agent



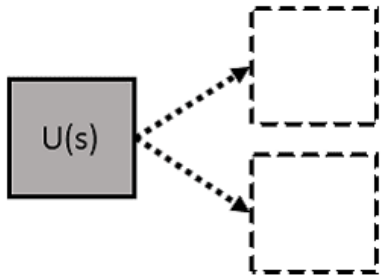
- Calculations based on utility
- Needs state dependencies

Q-learning Agent



- Needs a list of applicable actions for each state

Reflex Agent



- Reacts directly on percepts
- Bijection of states and actions
- E.g.: AI in a PONG-game

Section 2

Passive Learning

Passive Learning - Idea

- Simplest method to calculate a state's utility
- Fixed policy given
- Goal: Find utilities satisfying the **Bellman Equation**
- Similar to policy evaluation
- Difference: $P(s'|s,a)$ unknown $\rightarrow$ has to be learned

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

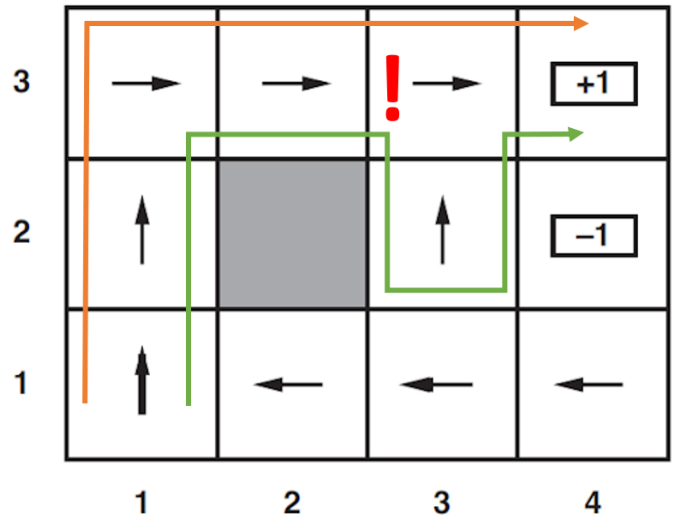
All learning methods converge to the Bellman Equation

Passive Learning - Direct Utility Estimation (DUE)

- Let the Agent run a few trials
- $U^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t R(S_t)$
- $S_0 = s$
- Discount factor γ
- Reward-to-go
- Instance of supervised learning
- Keep running average

Passive Learning - Direct Utility Estimation (DUE)

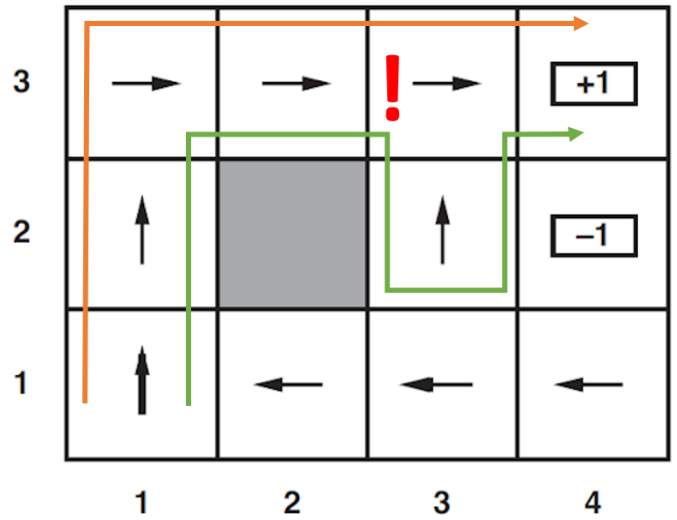
- Let the Agent run a few trials
- $U^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t R(S_t)$
- $S_0 = s$
- Discount factor γ
- Reward-to-go
- Instance of supervised learning
- Keep running average



$$U^{\pi}((1,1)) = \frac{(1-0.2)+(1-0.28)}{2}$$
$$= 0.76 \text{ (with } \gamma = 1)$$

Passive Learning - Direct Utility Estimation (DUE)

- Let the Agent run a few trials
- $U^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t R(S_t)$
- $S_0 = s$
- Discount factor γ
- Reward-to-go
- Instance of supervised learning
- Keep running average



$$U^{\pi}((1,1)) = \frac{(1-0.2)+(1-0.28)}{2}$$
$$= 0.76 \text{ (with } \gamma = 1)$$

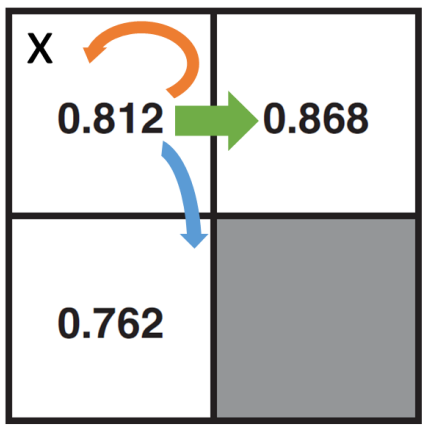
Utilities are not independent!

Passive Learning - Adaptive Dynamic Programming (ADP)

- Consider only local changes $\rightarrow$ recursive
- $U^{\pi}(s) = R(s) + \gamma \sum_{s'} [P(s'|s, \pi(s)) * U^{\pi}(s')]$
- No independence assumption anymore
- $P(s'|s, \pi(s))$ has to be learned from trials

Passive Learning - Adaptive Dynamic Programming (ADP)

- Consider only local changes $\rightarrow$ recursive
- $U^{\pi}(s) = R(s) + \gamma \sum_{s'} [P(s'|s, \pi(s)) * U^{\pi}(s')]$
- No independence assumption anymore
- $P(s'|s, \pi(s))$ has to be learned from trials



$$U^{\pi}(X) = -0.04$$
$$+0.8 * 0.868$$
$$+0.1 * 0.762$$
$$+0.1 * 0.812$$
$$=0.811$$

Passive Learning - Temporal Difference Learning (TD)

- Computationally cheaper than ADP
- No need for $P(s'|s, \pi(s))$
- ADP works with expected utility while TD only works with 'observed' successor
- $U^{\pi}(s) = U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$
- α: learning rate

Section 3

Active Learning

Passive vs. Active Learning

	Passive Learning	Active Larning
policy	given and fixed	none
updates of $U(s)$	after each trial	after each step
considered actions	given by policy	all possible

Bellmann equations: Passive-Learning $\rightarrow$ Active-Learning

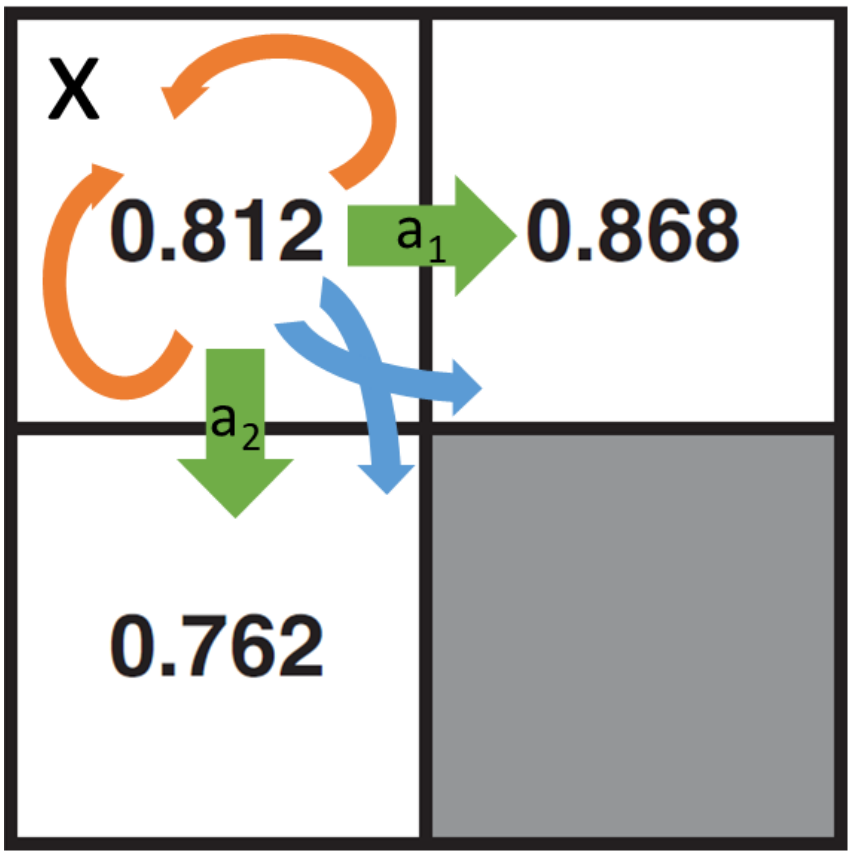
$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U^{\pi}(s')$$

$$\downarrow$$

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s, a) U(s')$$

Active Learning - Adaptive Dynamic Programming (ADP)

$$U(s) = R(s) + \gamma \max_a \sum_{s'} [P(s'|s,a) * U(s')]$$



$$U^{\pi}(X) = -0.04$$
$$\max_a(0.811,$$
$$0.8 * 0.762$$
$$+ 0.1 * 0.868$$
$$+ 0.1 * 0.812)$$
$$= 0.811$$

Active Learning - Q-Learning

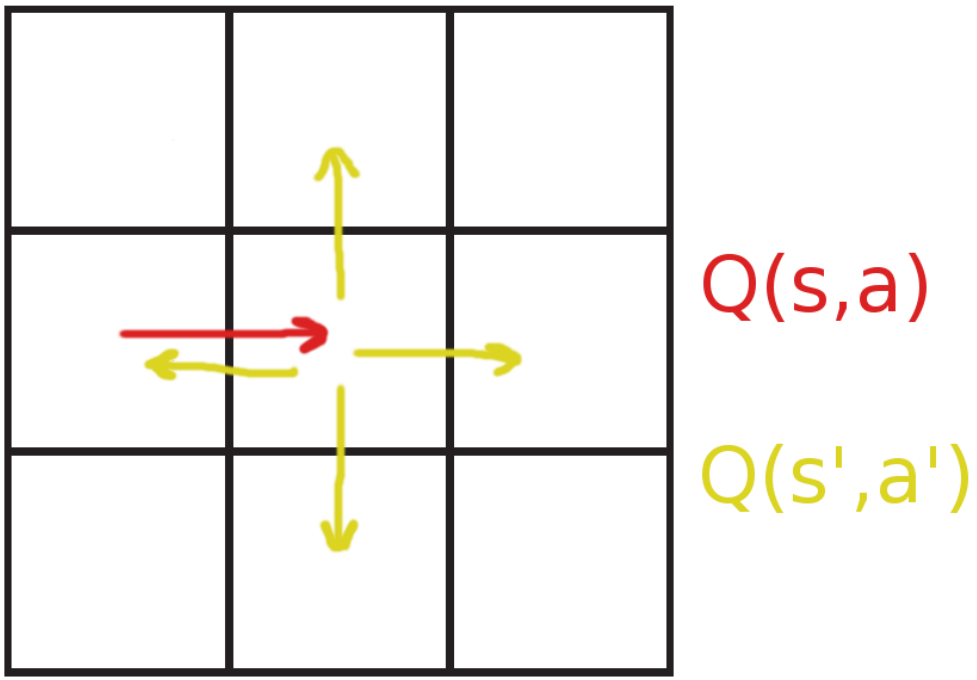
- Learning state-action pair utilities
- $U(s) = max_a Q(s, a)$ where $Q(s, a)$ is state-action pair utility

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s) + \gamma * max_{a'} Q(s', a') - Q(s, a))$$

- α: learning factor
- γ: discount factor
- s', a': states and actions that can be applied next
- Rating utility of state-action pair based on following state-action pair

Active Learning - Q-Learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma * max_{a'} Q(s',a') - Q(s,a))$$

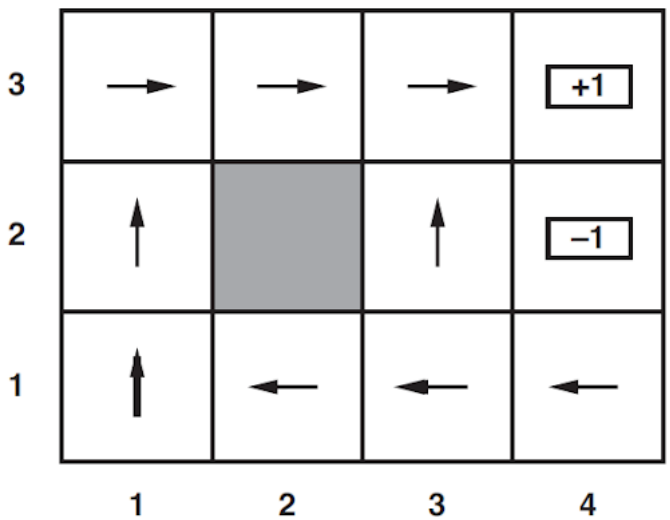


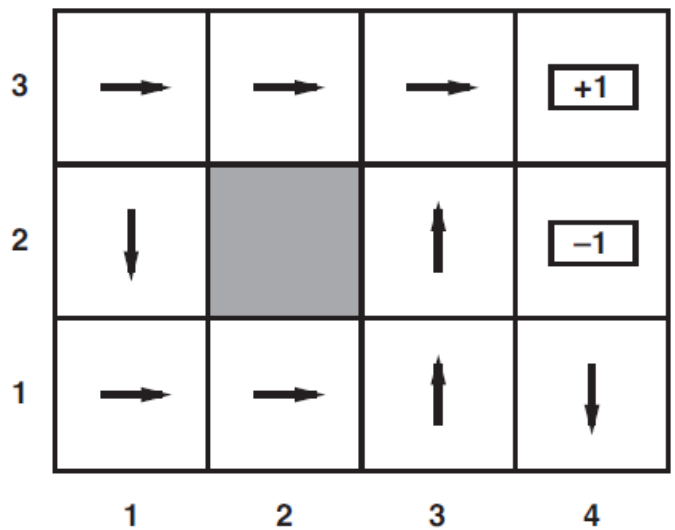
Section 4

Exploration vs. Exploitation

Exploration vs. Exploitation

- Always choose the best looking action (one-step look-ahead)
- Initial policy = optimal policy
- Nevertheless, a sub-optimal policy is learned

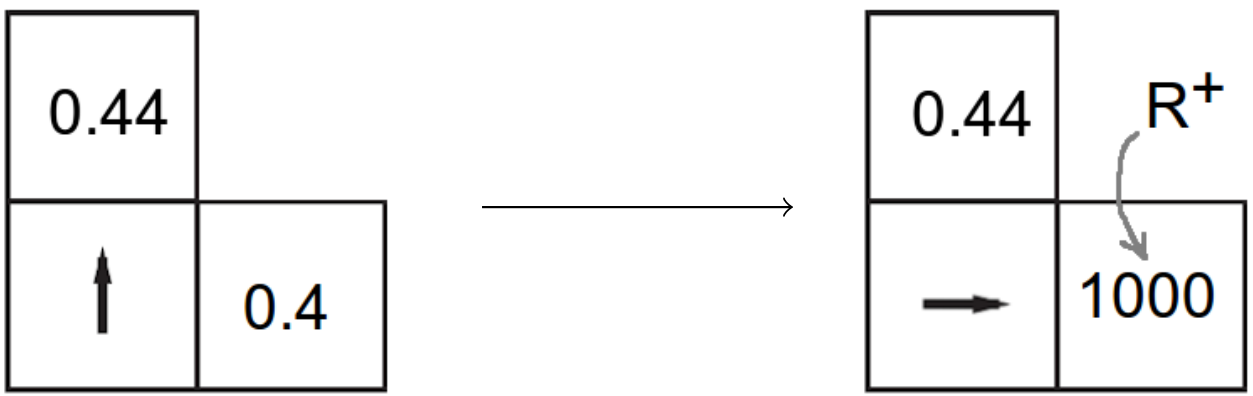




- one-step look-ahead is not enough
- Need the ability to explore possible shortcuts

Exploration vs. Exploitation - GLIE

- Solution: greedy in the limit of infinite exploration
  - Start curious (much exploration)
  - Chance to take random action instead of most attractive
  - Achieved by giving those actions very high rewards R^+
  - Decrease chance over time
- Issue: when to switch from exploring to exploiting?



Section 5

Generalization

Generalization - Idea

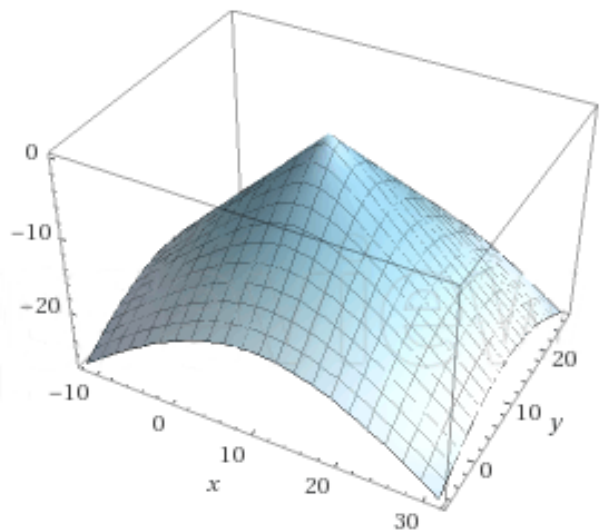
- State-spaces often large
- Calculate utilities with parameters θ
- Function approximation
- Learn parameters instead of utilities

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s)$$

Generalization - Example

- Grid world of size 20x20.
- Goal state +1 in position (10,10)

$$\hat{U}_\theta(s) = \theta_1 f_1(x, y) = \theta_1 \sqrt{(x - 10)^2 + (y - 10)^2}$$



Generalization - Update rule

- Widrow-Hoff rule or Delta rule
- Adjust parameters after each trial

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial E_j(s)}{\partial \theta_i}$$

$$E_j(s) = \frac{(\hat{U}_\theta(s) - u_j(s))^2}{2}$$

- $u_j(s)$: observed utility of state s in j-th trial
- α: learning factor
- Leads to hill-climbing

Section 6

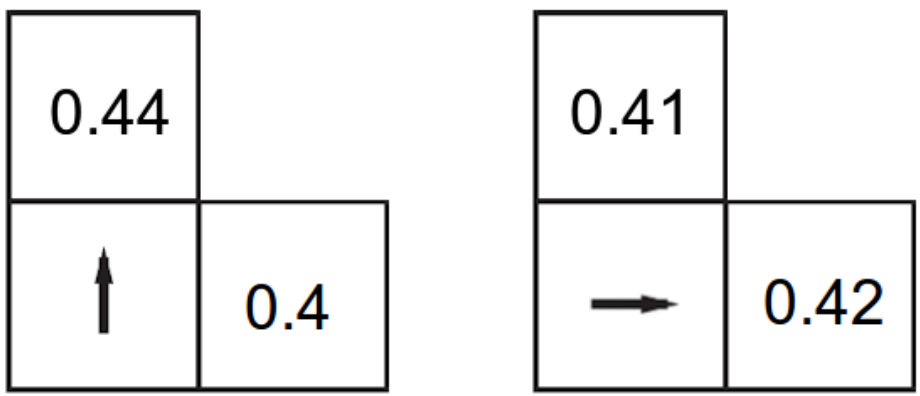
Policy Search

Policy Search - Idea

- Find a policy π_θ for state-space
- Represent policy as parameterized formula
- Like generalization, but parameters for action choice instead of utility calculation
- Obvious policy: choose action which promises biggest reward $\rightarrow$ greedy agent

Policy Search - Problems

- π_θ often discontiuous function
  - Slightly changed parameters $\rightarrow$ choose another action
  - Impossible to take derivative



- Example: greedy agent

Policy Search - Problems

- Solution: stochastic policy representation

Softmax function:

$$\pi_\theta(s,a) = e^{Q_\theta(s,a)} / \sum_{a'} e^{Q_\theta(s,a')}$$

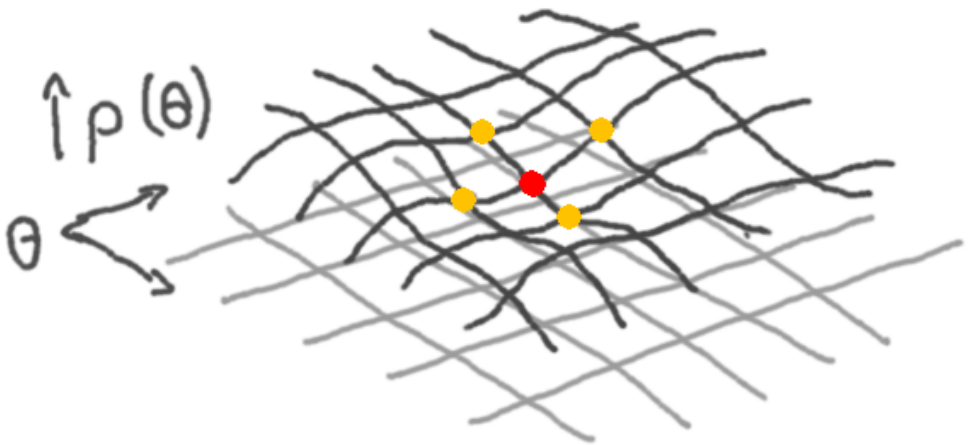
- $\pi_\theta(s,a)$: probability of taking action a in state s
- $Q_\theta(s,a)$: expected utility of taking action a in state s

Policy Search - Policy Improvement

- How to evaluate policy?
- Policy value $\rho(\theta)$
  - Expected reward for executing policy π_θ
- Follow policy gradient $\nabla_\theta \rho(\theta)$
  - Standard optimization problem

Policy Search - Policy Improvement

- $\rho(\theta)$ often not differentiable
  - Execute policy π_θ
  - $\rho(\theta) =$ accumulated reward of trial
  - Observe change of $\rho(\theta)$ for small changes of θ
  - Pick change with biggest improvement of $\rho(\theta)$



Policy Search - Policy Improvement

- Problematic in stochastic environment (same policy, different outcome)
  - Run several trials and take average
- Trials might be expensive or dangerous
  - Obtain estimate of $\nabla_\theta \rho(\theta)$ by looking at $R(a)$
  - $R(a)$: sum of subsequent rewards in trials where we chose action a

It would be dangerous for an agent to go for hundreds of hikes on a ridge of a mountain.

Image source: www.tripadvisor.ch

Section 7

Summary

Summary

- Different ways of designing agents:
  - Learn passively / actively
  - Assign utitlies to states or state-action pairs
- Many different algorithms for learning
  - All converge to Bellmann equation with different speeds
- Difficulty of balancing exploration and exploitation
- Environments can have huge numbers of states:
  - Generalize utility estimation
  - Generalize policy
  - Learn parameters instead of directly learning utilites

Questions

Questions?