Lecture 18 Online Learning (Part II) and Intro to Reinforcement Learning

Lei Li, **Yu-Xiang Wang**

Recap: Online Learning

- Learning with expert advice
  - A summary of regret bound: # mistakes - Oracle # of mistakes

	Consistency	**Halfing**	**Weighted Majority**	**Randomized WM**
Realizable setting	$\min(T, \lvert\mathcal{H}\rvert)$	$\min(T, \log\lvert\mathcal{H}\rvert)$	$\min(T, \log\lvert\mathcal{H}\rvert)$	$\min(T, \log\lvert\mathcal{H}\rvert)$
Agnostic setting	n.a.	n.a.	$(1+\epsilon)m + \log\lvert\mathcal{H}\rvert/\epsilon$	$\sqrt{m\log\lvert\mathcal{H}\rvert} = O(\sqrt{T\log\lvert\mathcal{H}\rvert})$

Recap: Hedge (aka Exponential weighted average) algorithm

EXPONENTIAL-WEIGHTED-AVERAGE (N)

for $i \leftarrow 1$ **to** N **do**
  $w_{1,i} \leftarrow 1$
for $t \leftarrow 1$ **to** T **do**
  RECEIVE(x_t)
  $\widehat{y}_t \leftarrow \frac{\sum_{i=1}^{N} w_{t,i} y_{t,i}}{\sum_{i=1}^{N} w_{t,i}}$
  RECEIVE(y_t)
  for $i \leftarrow 1$ **to** N **do**
    $w_{t+1,i} \leftarrow w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)}$
return $\mathbf{w}_{T+1}$

- Works for linear loss function in its first argument
  - That is bounded
- Also works for any convex loss function in its first argument
  - Why?

This lecture

- Online Learning (Part II)
  - Online Gradient Descent

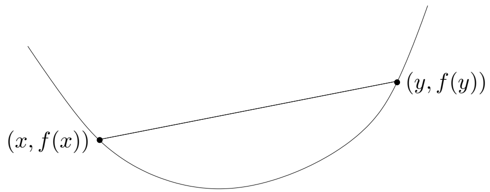
- Reinforcement Learning
  - Problem setup
  - Markov Decision Processes

Recap: Convex functions / sets and subgradient

Convex function: $f : \mathbb{R}^n \to \mathbb{R}$ such that $\mathrm{dom}(f) \subseteq \mathbb{R}^n$ convex, and

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y) \quad \text{for all } 0 \le t \le 1$$

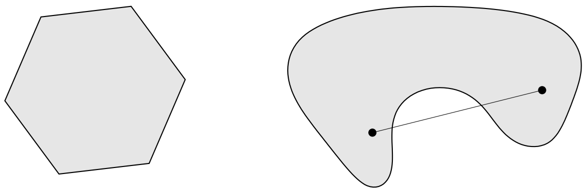
and all $x, y \in \mathrm{dom}(f)$



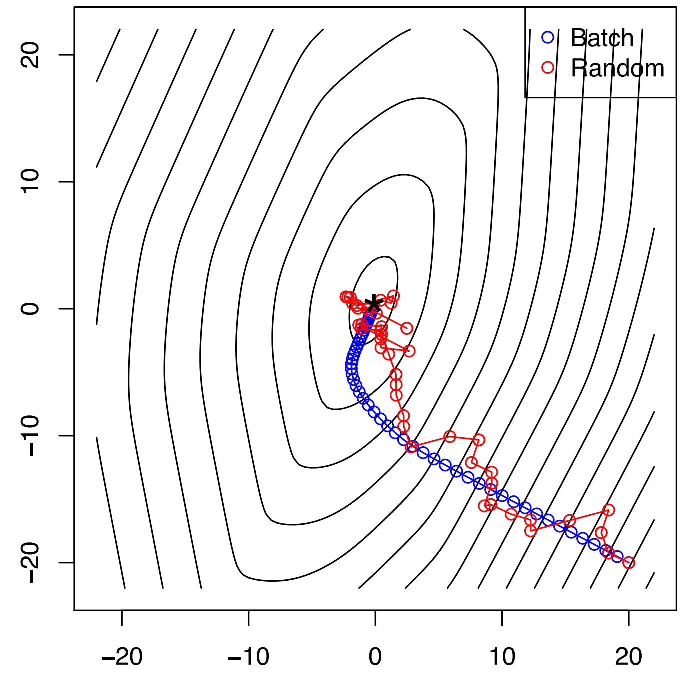
- First order definition / subgradient

Convex set: $C \subseteq \mathbb{R}^n$ such that

$$x, y \in C \implies tx + (1-t)y \in C \quad \text{for all } 0 \le t \le 1$$



Recap: Gradient Descent and SGD (from Lecture 4)



- Problem: $\min_{\theta} f(\theta)$
- GD alg.: $\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$
- SGD alg.: $\theta_{t+1} = \theta_t - \eta_t \hat{\nabla} f(\theta_t)$

- Example when solving ERM:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i))$$

  - Pick a **single** data point i uniformly at random

$$\nabla_{\theta} \ell(\theta, (x_i, y_i))$$

(Projected) Subgradient "Descent"

- When we have constraints and non-differentiable convex functions, we can use
  - Projected Subgradient method
  - Projected Stochastic subgradient method

The problem of Online Convex Optimization

- Problem setup:

- Performance metric --- regret

Examples of OCO problems

- Example 1: Prediction with Expert Advice

- Example 2: Online Linear models

- Example 3: Portfolio Selection

Algorithm: OGD, i.e., Online (projected) (sub)Gradient Descent

- Standard projected subgradient updates

- Assumptions needed:
  - Bounded domain
  - Lipschitz loss functions

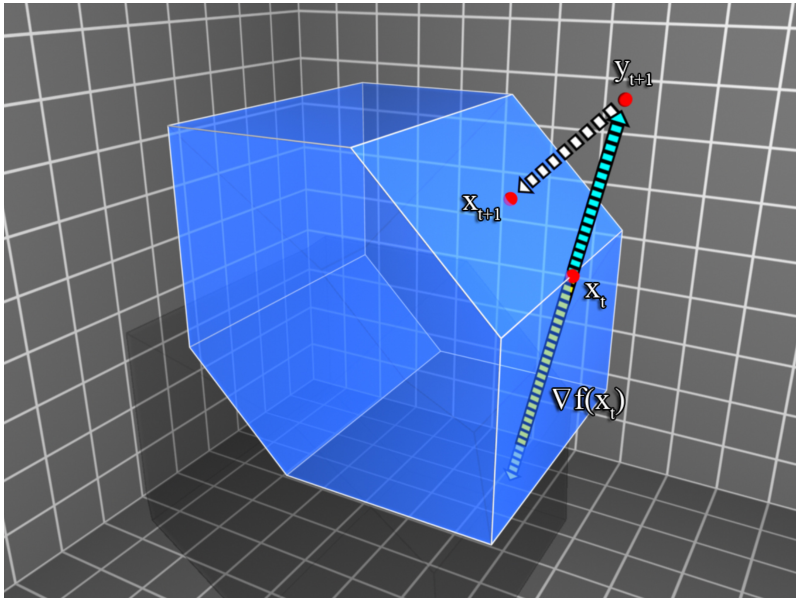


Figure 3.1: Online gradient descent: the iterate $\mathbf{x}_{t+1}$ is derived by advancing $\mathbf{x}_t$ in the direction of the current gradient ∇_t, and projecting back into $\mathcal{K}$.

Analysis of OGD

- By convex functions

- By the update rule (and property of projection)

- Put them together!

Analysis of OGD (continues)

- Telescoping

Regret bound for OGD

Theorem 3.1. Online gradient descent with step sizes $\{\eta_t = \frac{D}{G\sqrt{t}},\ t \in [T]\}$ guarantees the following for all $T \geq 1$:

$$\text{regret}_T = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}^\star \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x}^\star) \ \leq \frac{3}{2} GD\sqrt{T}$$

- “Any-time” algorithm with a decreasing learning rate schedule
- Learning rate depends on t. (exercise to prove that this works.)

Online to Batch conversion: How do I use OGD to solve ERM?

Checkpoint

- Online learning
  - Operates in an adversarial environment
  - Almost no assumptions. Do not even use probability theory

- The idea of regret and no-regret learning algorithms

- Useful algorithmic ideas:
  - Hedge / Exponential Weighted Averages
  - Online gradient descent

What we did not cover

- The strongly convex case
- Adapting to the geometry
  - AdaGrad / ADAM
- Adaptive regret / dynamic regret

- Modern applications to Ensemble learning, AutoML

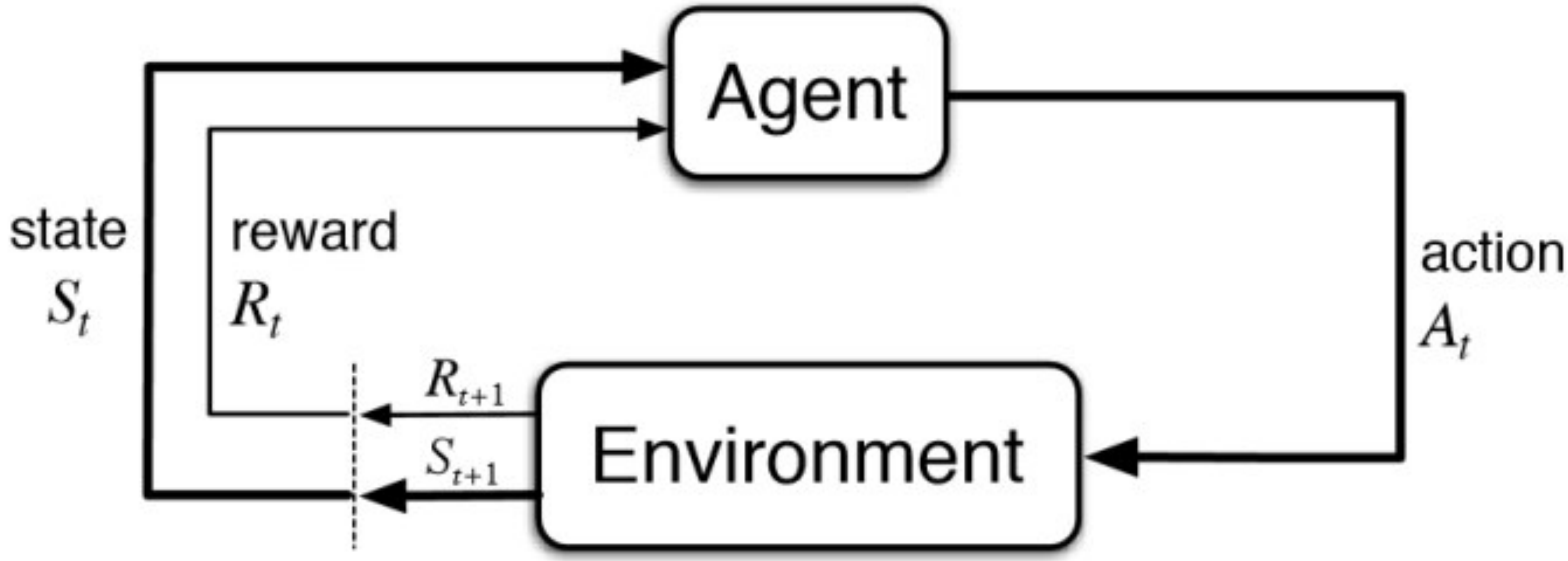
[PDF] Hyperband: A novel bandit-based approach to hyperparameter optimization
L Li, K Jamieson, G DeSalvo, A Rostamizadeh… - The Journal of Machine …, 2017 - jmlr.org
Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian optimization to adaptively select …
☆ Save 99 Cite Cited by 1612 Related articles Import into BibTeX

This lecture

- Online Learning (Part II)
  - Online Gradient Descent

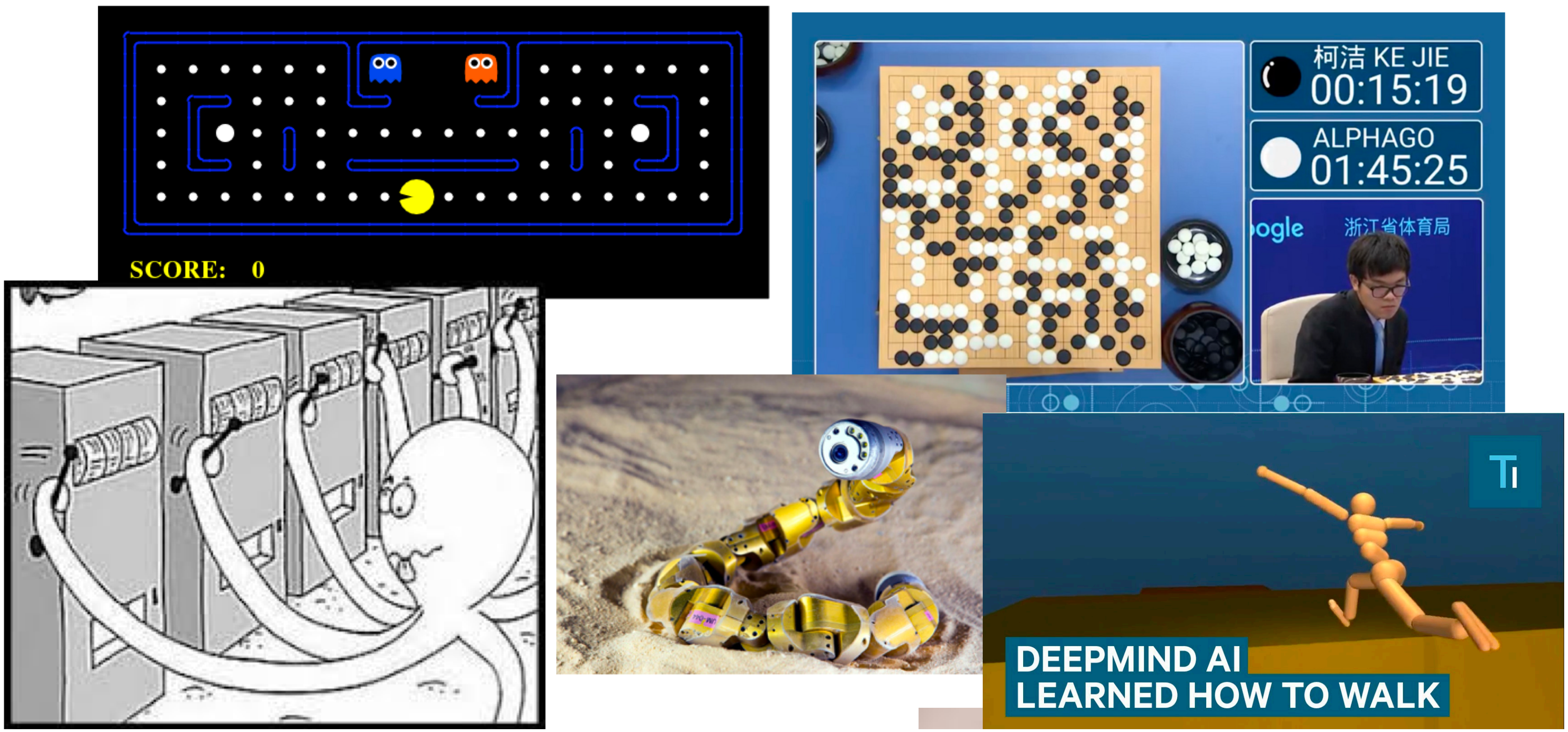
- Reinforcement Learning
  - Problem setup
  - Markov Decision Processes

An RL agent learns interactively through the feedbacks of an environment.



- Learning how the world works (dynamics) and how to maximize the long-term reward (control) at the same time.

Reinforcement learning is among the hottest area of research in ML!



"RL" is Top 1 Keyword at NeurIPS'2021, appearing 199 times
"Deep Learning" only 129 times [source]

Applications of RL in the real life

- RL for robotics.
- RL for dialogue systems.
- RL for personalized medicine.
- RL for self-driving cars.
- RL for new material discovery.
- RL for sustainable energy.
- RL for feature-based dynamic pricing.
- RL for maximizing user satisfaction.
- RL for QoE optimization in networking
- …

Reinforcement learning problem setup

- State, Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O}$$

- Policy: $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - When the state is observable:
  - Or when the state is not observable

$$\pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A}$$

- Learn the best policy that maximizes the expected reward
  - Finite horizon (episodic) RL: $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{T} R_t]$
  - Infinite horizon RL:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} R_t]$$

T: horizon

γ: discount factor

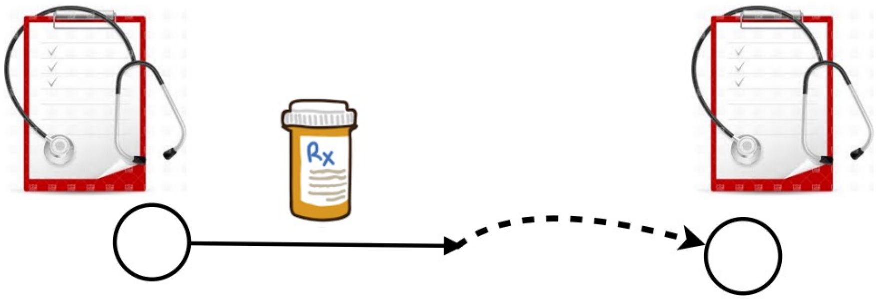
RL for robot control

- States: The physical world, e.g., location/speed/acceleration and so on.
- Observations: camera images, joint angles
- Actions: joint torques
- Rewards: stay balanced, navigate to target locations, serve and protect humans, etc.

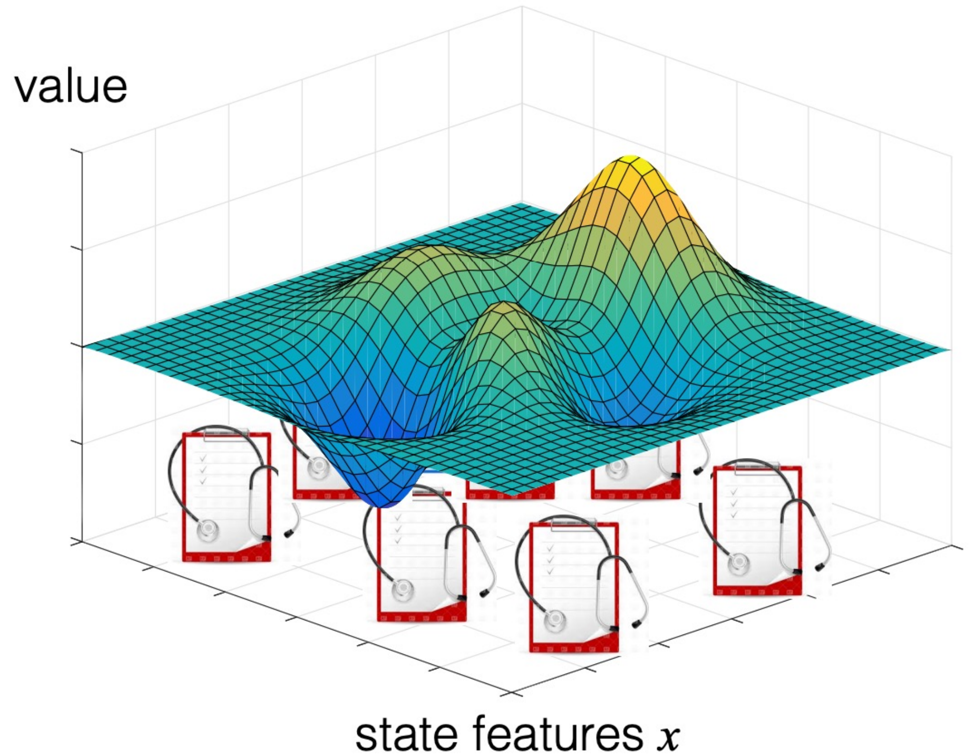
RL for Inventory Management

- State: Inventory level, customer demand, competitor's inventory
- Observations: current inventory levels and sales history
- Actions: amount of each item to purchase
- Rewards: profit

RL for Adaptive medical treatment



- State: diagnosis
- Action: treatment
- Reward: progress in recovery



(example / illustration due to Nan Jiang)

Example: Supervised learning vs RL in movie recommendation

- Bob is described by a feature vector
  - s =[Previous movies watched / Rating / Written reviews]

- Supervised learning predicts how likely Bob will click on "aliens vs predators"

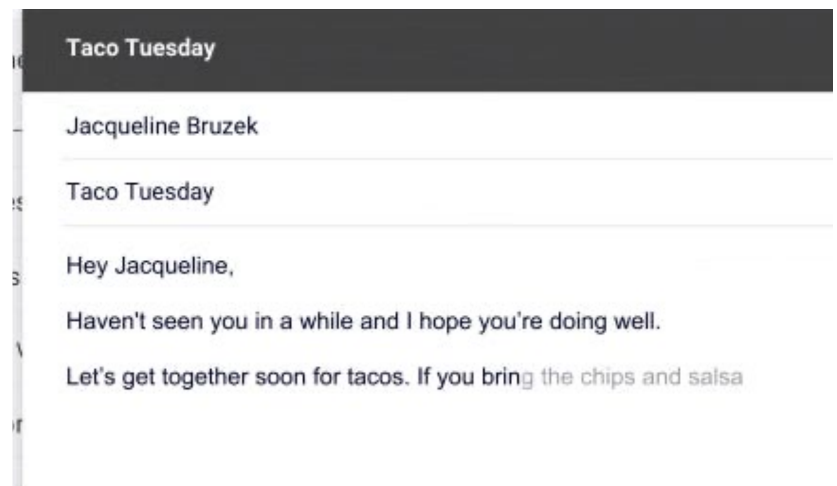
- Reinforcement learning aims at controlling Bob
  - So in the future, Bob will develop a taste for "aliens vs predators" (e.g., from having watched "aliens" and "predators" both).

A broader view: Let’s consider a few other machine learning tasks

- Hospitals need to decide **who to test** based on symptoms and other patient attributes

- Train a classifier on historic records to predict the test outcome.
- The accuracy is high on a holdout set!

- Large tech wants to improve user experience on their popular email service



- Train a **large language model** with user data to **complete sentences**
- It seems to work great!

What could go wrong?

Every machine learning problem is secretly a control (or RL) problem

- If I test patients using the new rule, the distribution of patients receiving the test will be different!
- Should I still trust my classifier?

- If I deploy the new "Guess what you will write" prompt, what users will enter may change!
- Is the model fulfilling its own prophecy?

The ultimate goal is NOT prediction, but to:
minimize disease transmission / maximize user experience!

Reinforcement learning is very challenging

- The agent needs to:
  - Learn the state-transitions ----- How the world works
  - Learning the costs / rewards ----- Cost of actions
  - Learning how to search ----- Come up with a good strategy

- All at the same time

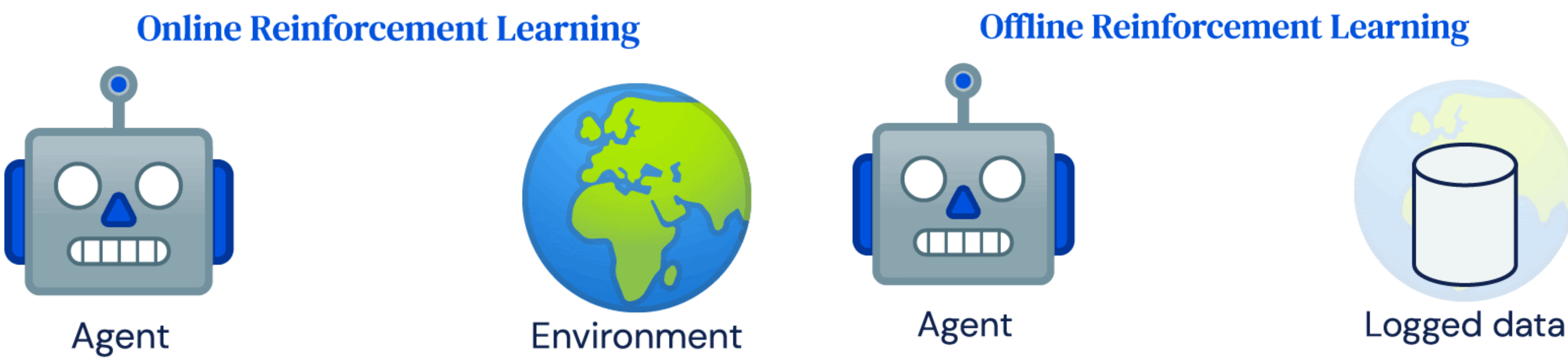
Let us tackle different aspects of the RL problem one at a time

- **Markov Decision Processes: (remainder of this lecture)**
  - Dynamics are given no need to learn. planning only.

- RL algorithms (next lecture)
  - Model-based RL vs Model-free RL
  - Temporal difference learning
  - Function approximation

- Exploration (final lecture)
  - Bandits: Explore-Exploit in simple settings
  - RL: Explore-Exploit in Learning MDPs

Online RL vs Offline RL



Exploration is often **expensive**, **unsafe**, **unethical** or **illegal** in practice, e.g., in self-driving cars, or in medical applications.

Can we learn a policy from already **logged interaction data**?

***Offline RL won't be covered, but it's an important problem**

Let's start by formulating Markov Decision processes (MDP).

- Infinite horizon / discounted setting

$$\mathcal{M}(\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu)$$

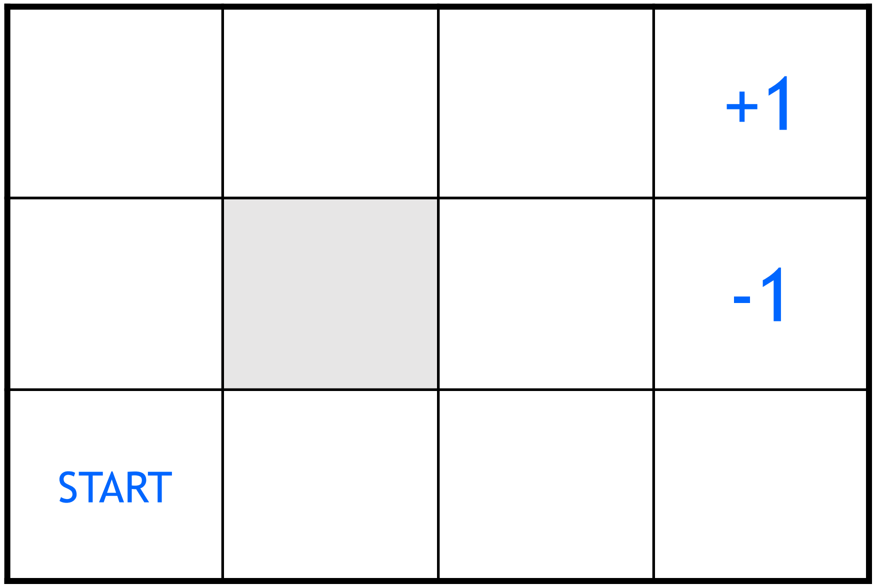
Transition kernel: $P: \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ i.e. $P(s'|s,a)$

(Expected) reward function: $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R} / [0, R_{max}]$ $\mathbb{E}[R_t \mid S_t = s, A_t = a] =: r(s,a)$

Initial state distribution $\mu \in \Delta(\mathcal{S})$

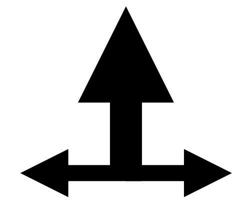
Discounting factor: γ

Example: Frozen lake.



actions: UP, DOWN, LEFT, RIGHT

UP e.g.,



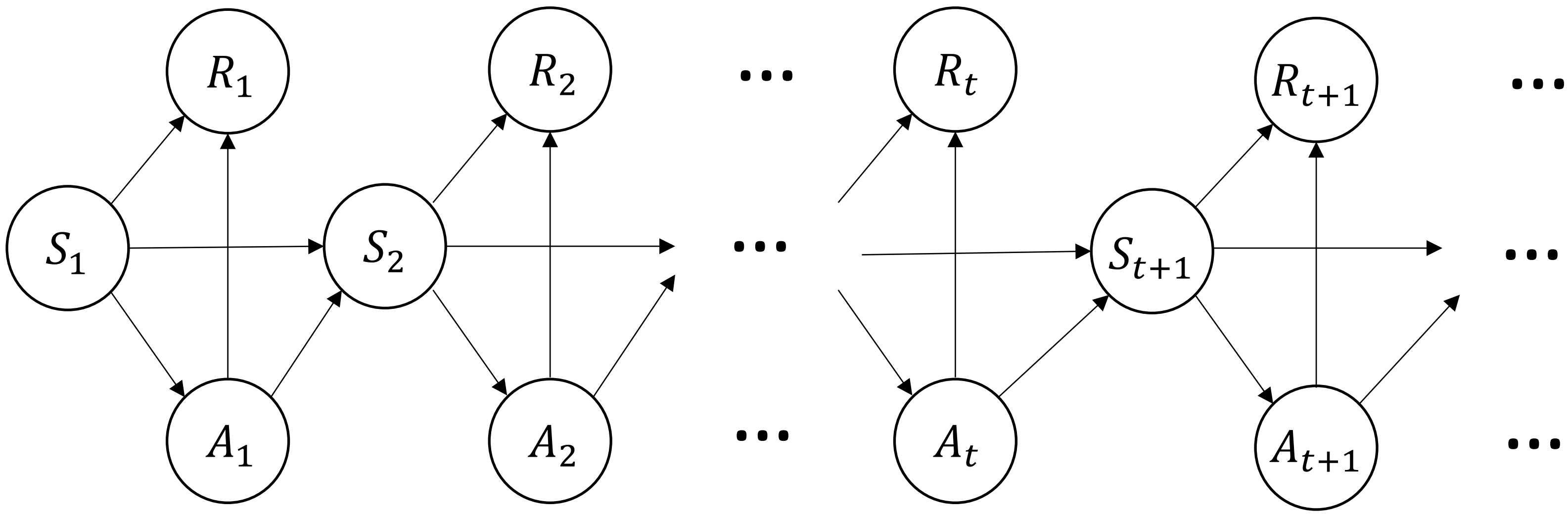
State-transitions with action **UP**:

80% move up
10% move left
10% move right

*If you bump into a wall, you stay where you are.

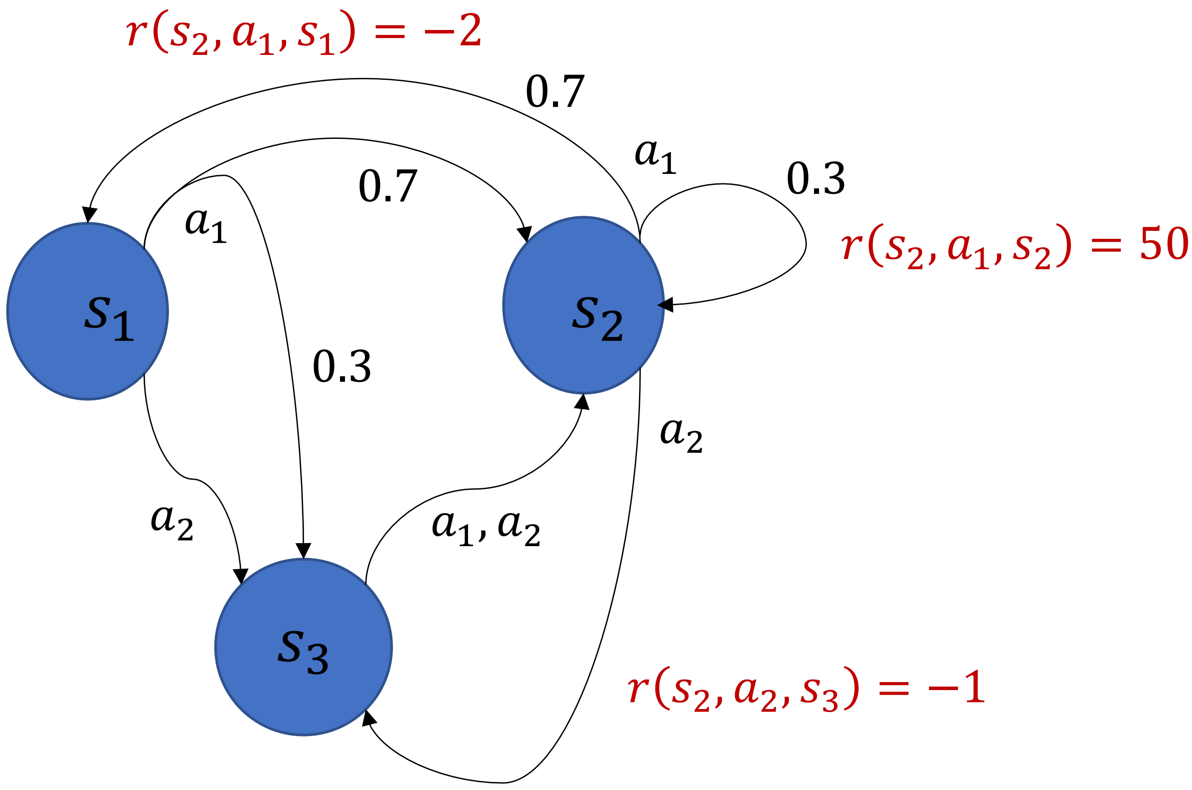
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- Finite horizon or infinite horizon?
- What is a good policy?

Parameters of an MDP are factorizations of the joint distribution



- Initial state distribution
- Transition dynamics
- Reward distribution

State-space diagram representation of an MDP: An example with 3 states and 2 actions.



* The reward can be associated with only the state s' you transition into.
* Or the state that you transition from s and the action a you take.
* Or all three at the same time.

Reward function and Value functions

- Immediate reward function r(s,a)
  - expected **immediate** reward

    $$r(s,a) = \mathbb{E}[R_1 | S_1 = s, A_1 = a]$$

    $$r^{\pi}(s) = \mathbb{E}_{a \sim \pi(a|s)}[R_1 | S_1 = s]$$

- state value function: $V^{\pi}(s)$
  - expected **long-term** return when starting in *s* and following π

    $$V^{\pi}(s) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s]$$

- state-action value function: $Q^{\pi}(s,a)$
  - expected **long-term** return when starting in *s*, performing *a,* and following π

    $$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s, A_1 = a]$$

Optimal value function and the MDP planning problem

$$V^\star(s) := \sup_{\pi \in \Pi} V^\pi(s)$$

$$Q^\star(s,a) := \sup_{\pi \in \Pi} Q^\pi(s,a).$$

Goal of MDP planning:

$$\text{Find } \pi^* \text{ such that } V^\pi(s) = V^*(s) \quad \forall s$$

Approximate solution:

$$\pi \text{ is } \epsilon\text{-optimal if } V^\pi \geq V^*(s) - \epsilon \mathbf{1}$$

General policy, Stationary policy, Deterministic policy

- General policy could depend on the entire history

$$\pi : (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^* \times \mathcal{S} \to \Delta(\mathcal{A})$$

- Stationary policy

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})$$

- Stationary, Deterministic policy

$$\pi : \mathcal{S} \to \mathcal{A}$$

Two surprising facts about MDPs

1. It suffices to consider stationary / deterministic policies.

2. There exists a stationary / deterministic policy that is optimal simultaneously for all initial state distributions.

Bellman equations – the fundamental equations of MDP and RL

- An alternative, recursive and more useful way of defining the V-function and Q function

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

- Exercise:
  - Prove Bellman equation from the definition.
  - Write down the Bellman equation using Q function alone.

    $Q^\pi(s,a) = ?$

Bellman optimality equations characterizes the optimal policy

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

- system of n non-linear equations
- solve for V*(s)
- easy to extract the optimal policy

- having Q*(s,a) makes it even simpler

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

Bellman equations in matrix forms

- Lemma 1.4 (Bellman consistency): For stationary policies, we have

$$V^\pi(s) = Q^\pi(s, \pi(s)).$$

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^\pi(s') \right].$$

- In matrix forms:

$$V^\pi = r^\pi + \gamma P^\pi V^\pi$$

$$Q^\pi = r + \gamma P V^\pi$$

$$Q^\pi = r + \gamma P^\pi Q^\pi .$$

Value iterations for MDP planning

- Recall: Bellman optimality equations

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

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s',a') \right].$$

$$\mathcal{T}Q = r + PV_Q$$ where $$V_Q(s) := \max_{a \in \mathcal{A}} Q(s,a).$$

Theorem: Q = Q* if and only if Q satisfies the Bellman optimality equations.

Value iterations for MDP planning

- The value iteration algorithm iteratively applies the Bellman operator until it converges.
  1. Initialize Q_0 arbitrarily
  2. for i in 1,2,3,…, k, update $Q_i = \mathcal{T} Q_{i-1}$
  3. Return Q_k

- **What is the right question to ask here?**

Convergence of value iteration for solving MDPs

- Lemma 1. The Bellman operator is a γ-contraction.

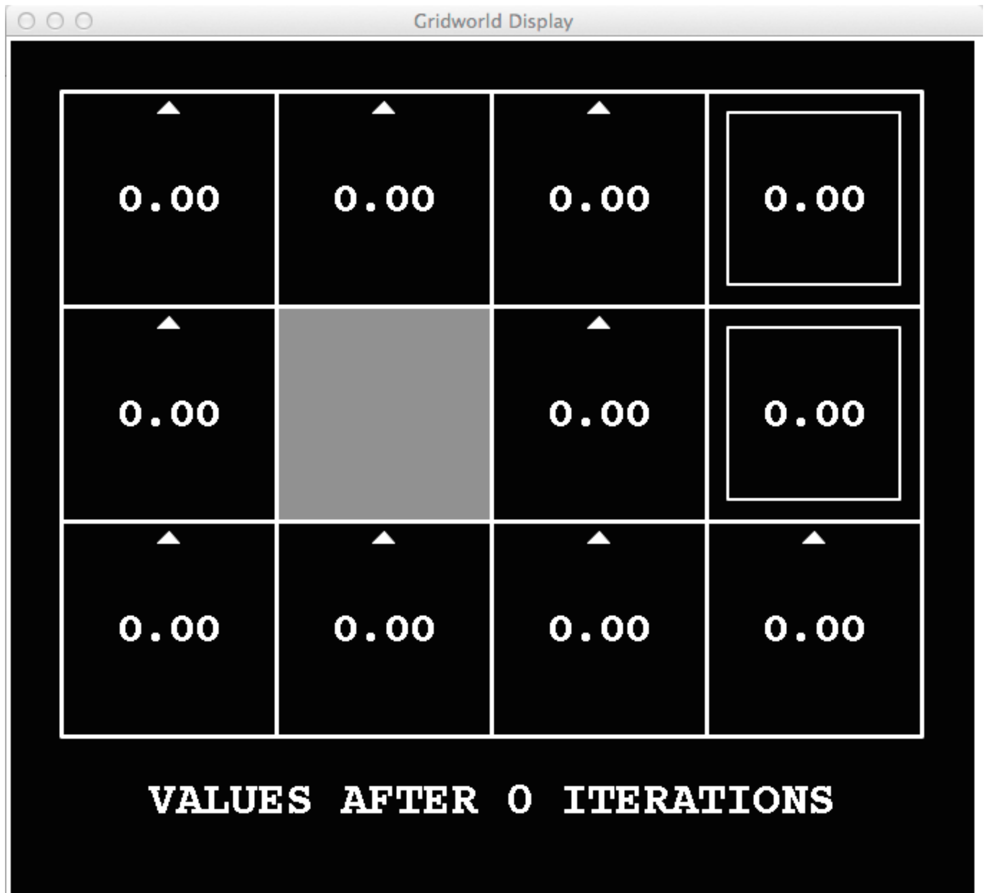
$$\text{For any two vectors } Q, Q' \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|},$$

$$\|\mathcal{T}Q - \mathcal{T}Q'\|_\infty \leq \gamma \|Q - Q'\|_\infty$$

  - Prove this in the optional HW4.

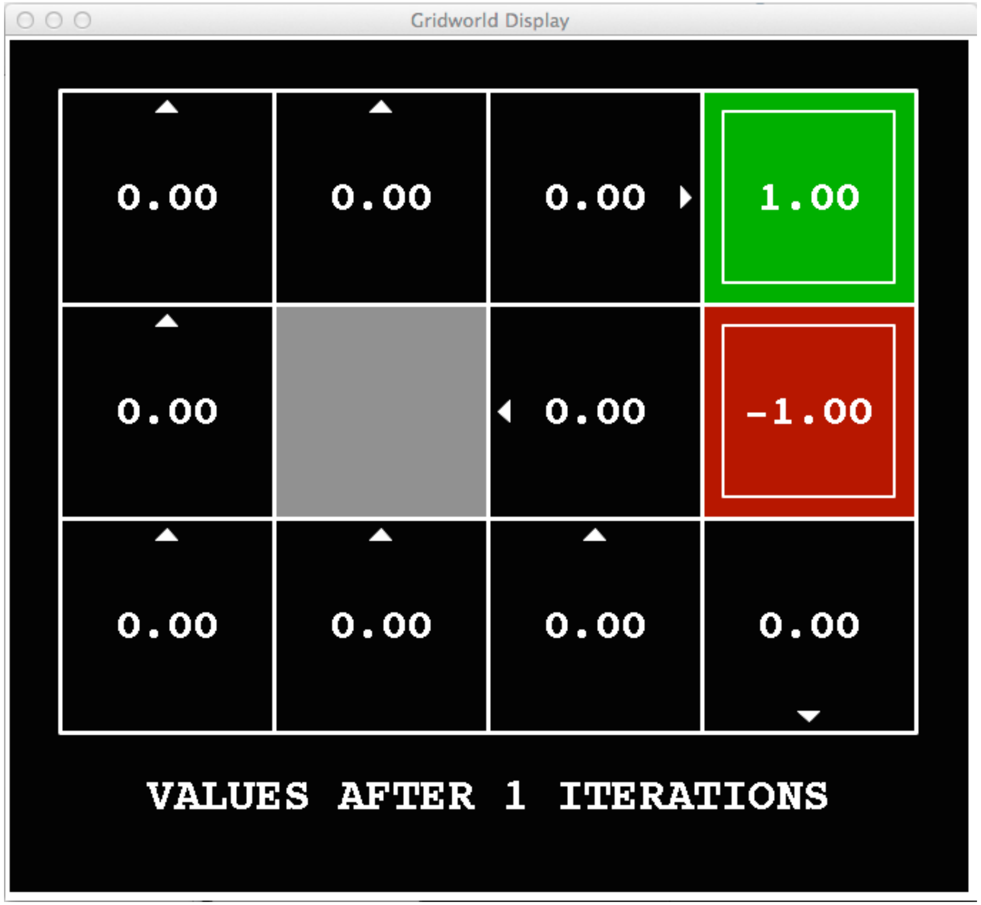
- Fast convergence of value iterations to Q*:

k=0



Noise = 0.2
Discount = 0.9
Living reward = 0

k=1



Noise = 0.2
Discount = 0.9
Living reward = 0

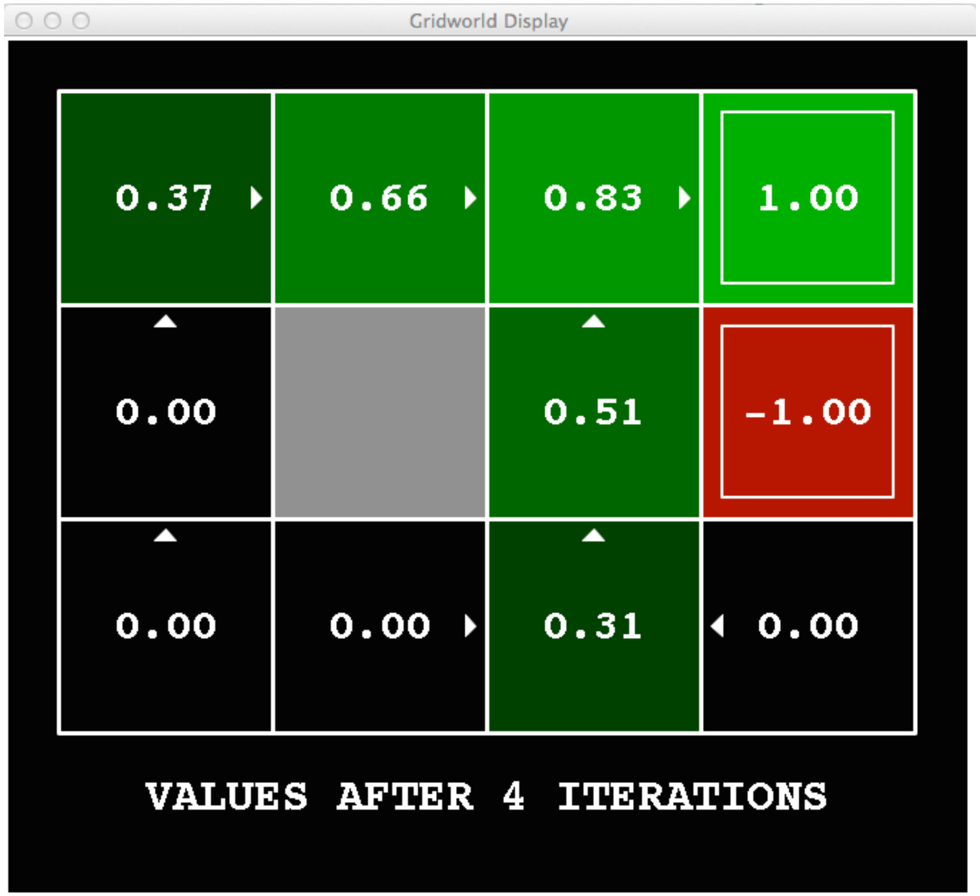
k=2

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

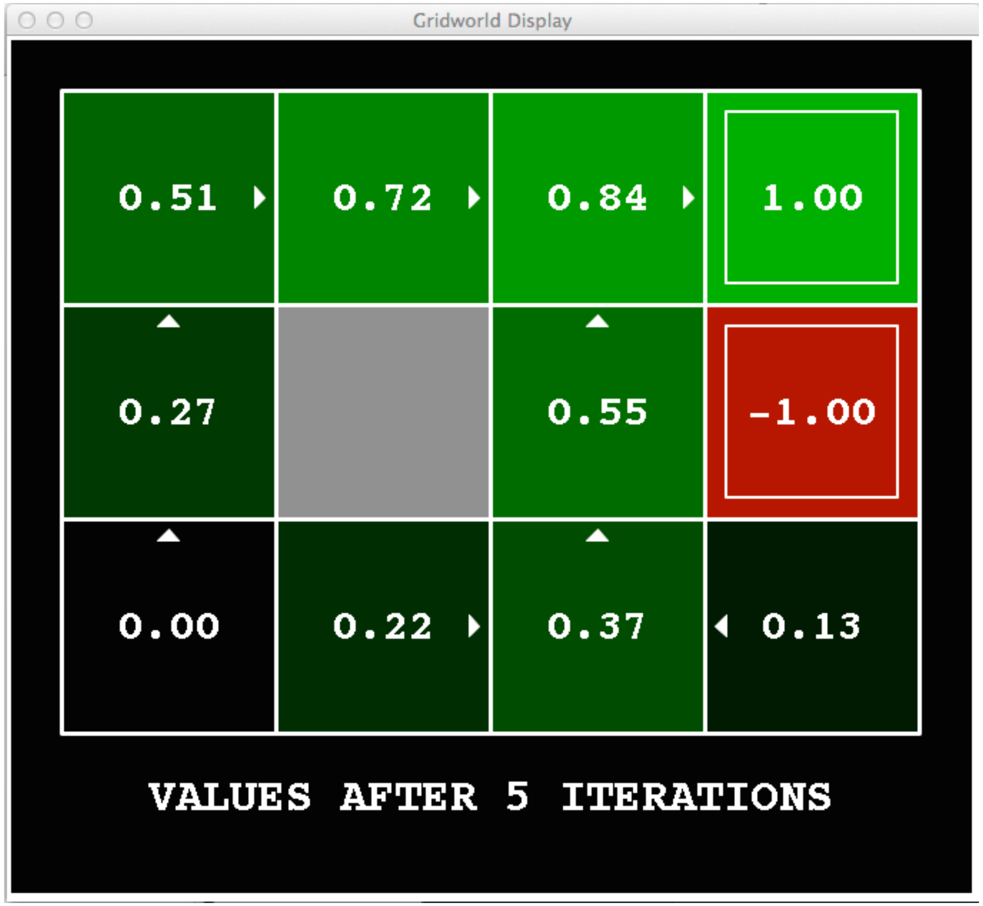
Noise = 0.2
Discount = 0.9
Living reward = 0

k=4



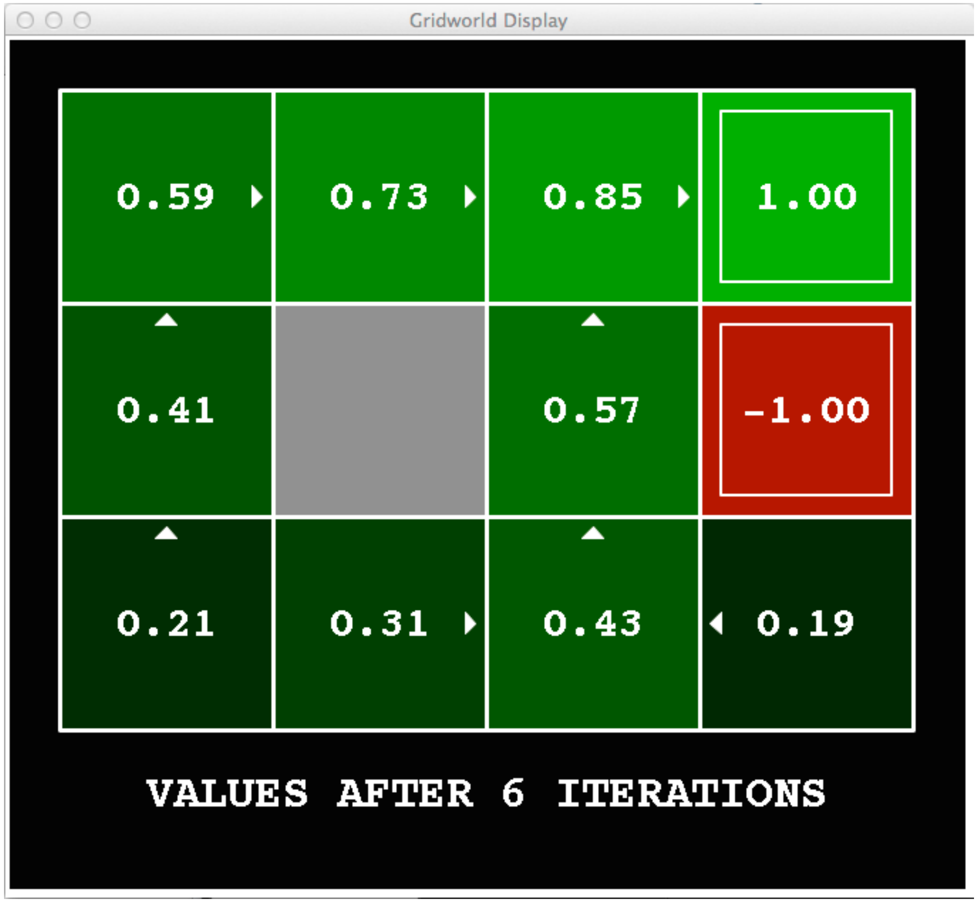
Noise = 0.2
Discount = 0.9
Living reward = 0

k=5



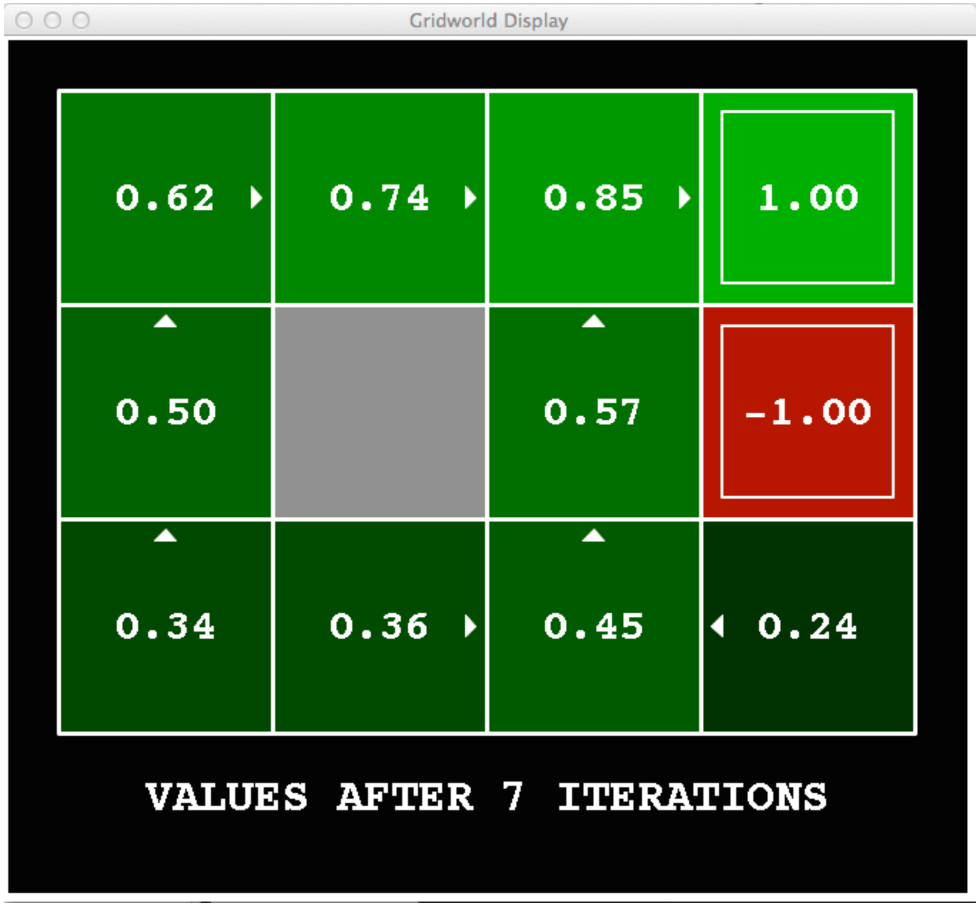
Noise = 0.2
Discount = 0.9
Living reward = 0

k=6



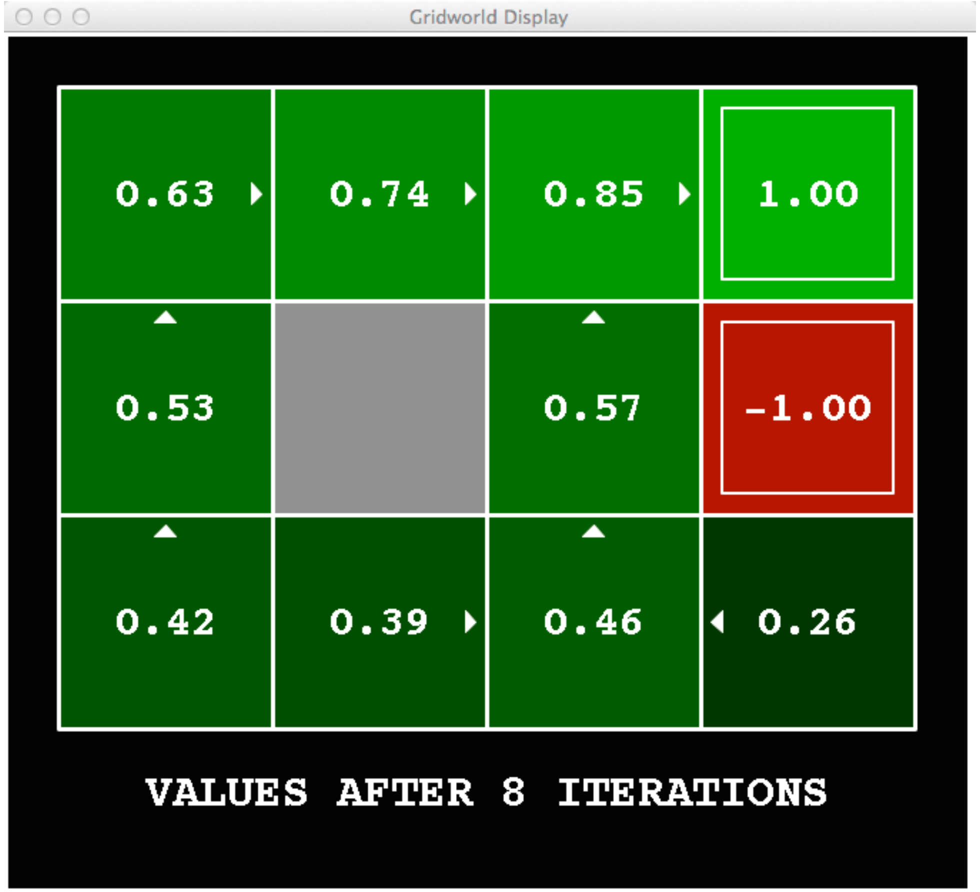
Noise = 0.2
Discount = 0.9
Living reward = 0

k=7



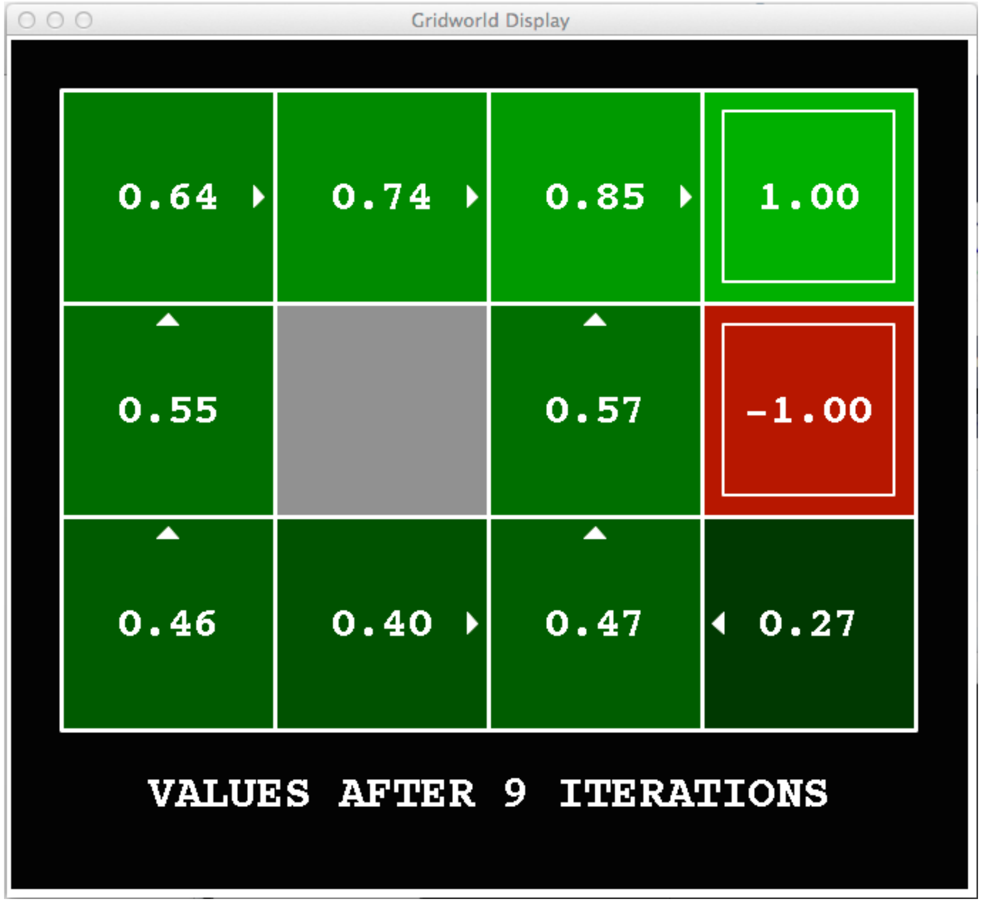
Noise = 0.2
Discount = 0.9
Living reward = 0

k=8



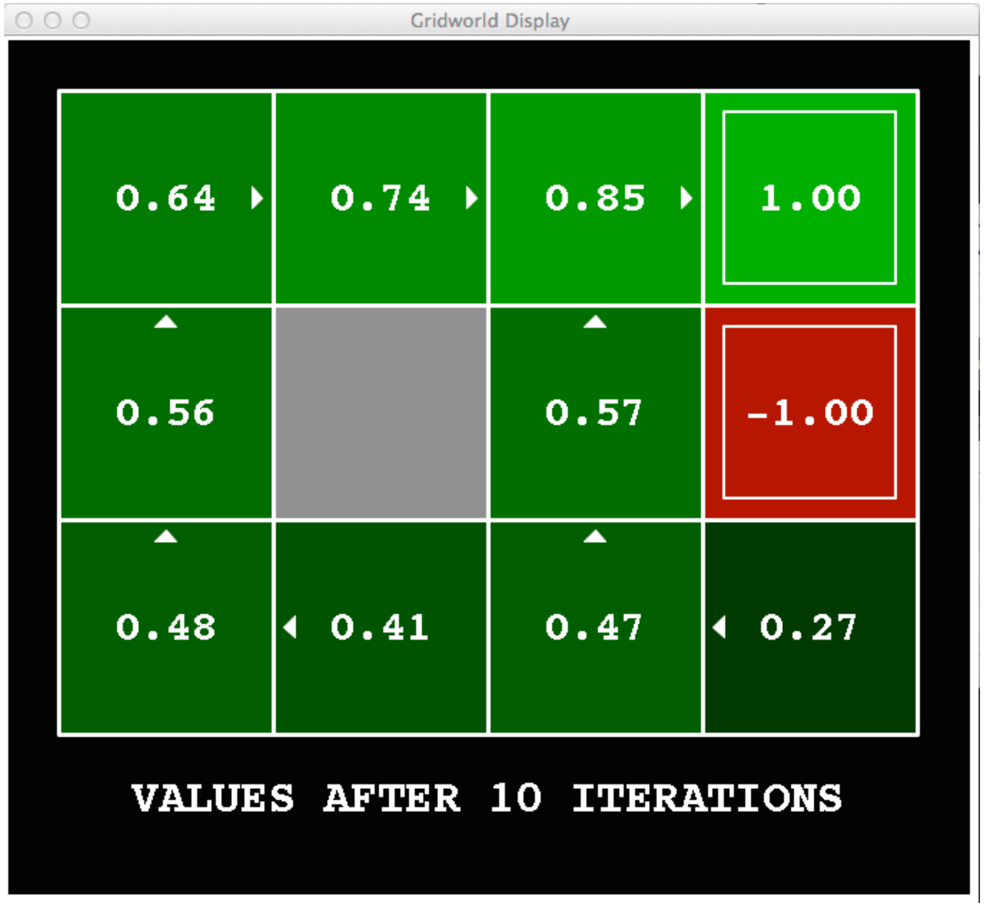
Noise = 0.2
Discount = 0.9
Living reward = 0

k=9



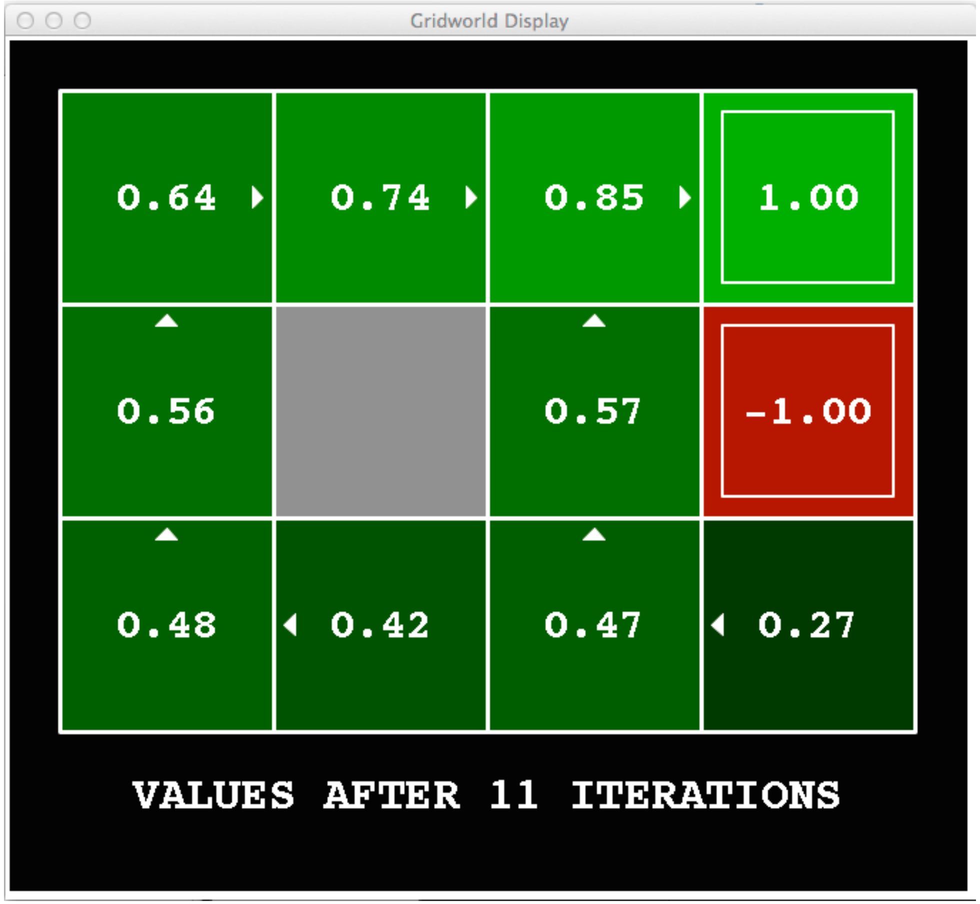
Noise = 0.2
Discount = 0.9
Living reward = 0

k=10



Noise = 0.2
Discount = 0.9
Living reward = 0

k=11

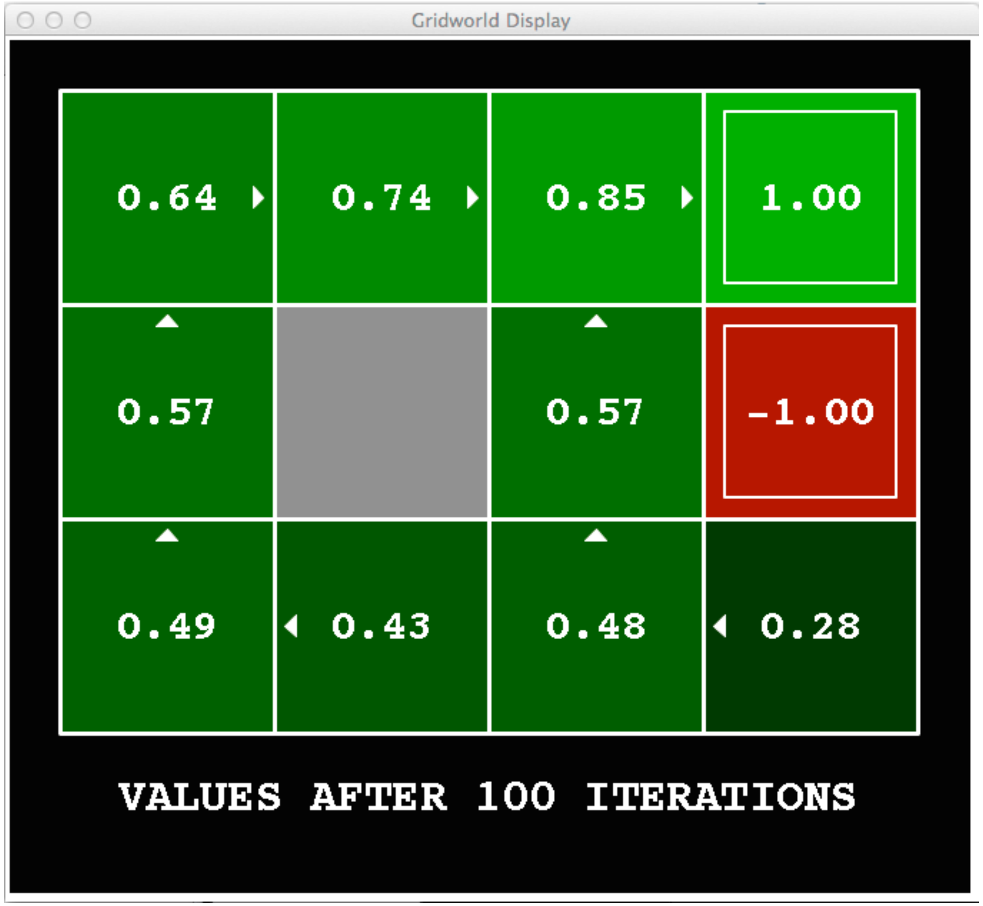


Noise = 0.2
Discount = 0.9
Living reward = 0

k=12

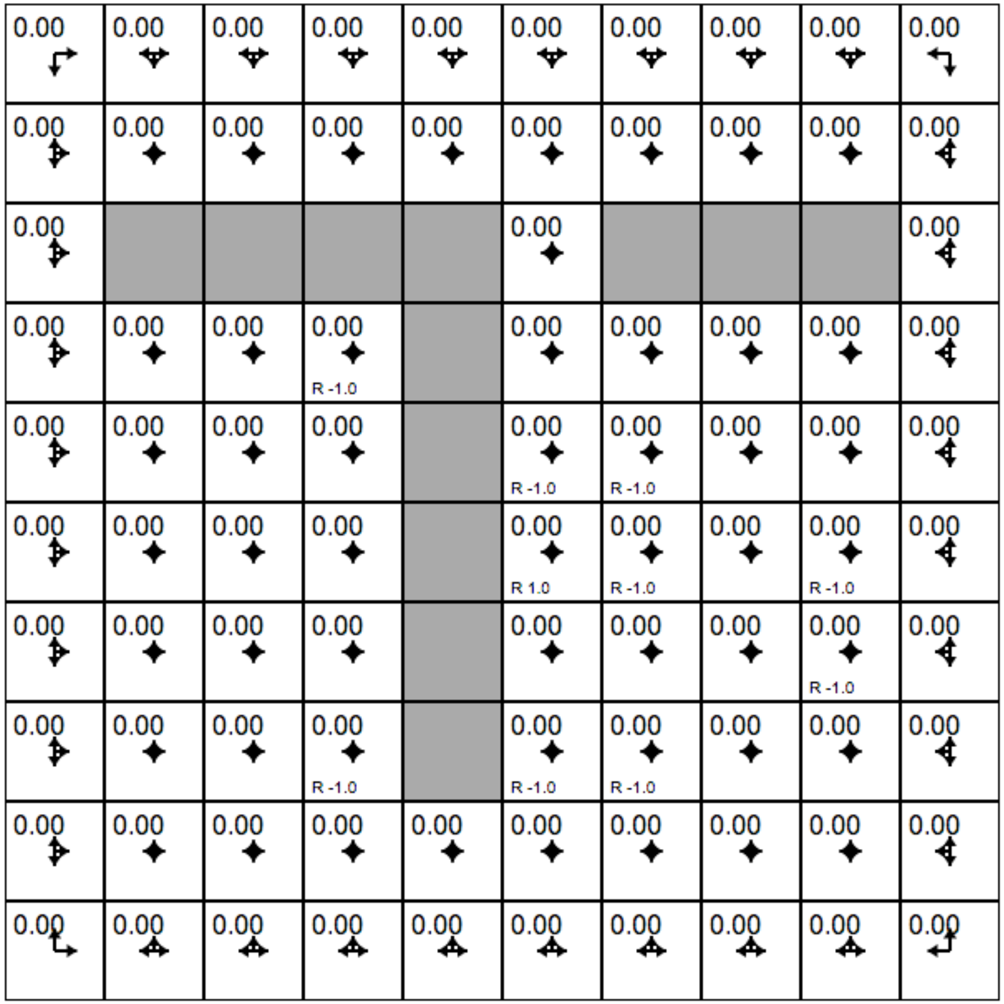
Noise = 0.2
Discount = 0.9
Living reward = 0

k=100



Noise = 0.2
Discount = 0.9
Living reward = 0

Demo: grid worlds



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Checkpoint

- What is RL? What are its motivating applications?

- A model of RL --- Markov Decision Processes
  - Value functions: Q functions and V functions
  - Bellman equations

- MDP planning / inference problem
  - Value iterations

Next lecture

- RL algorithms:
  - What happens if we don’t know the MDP?