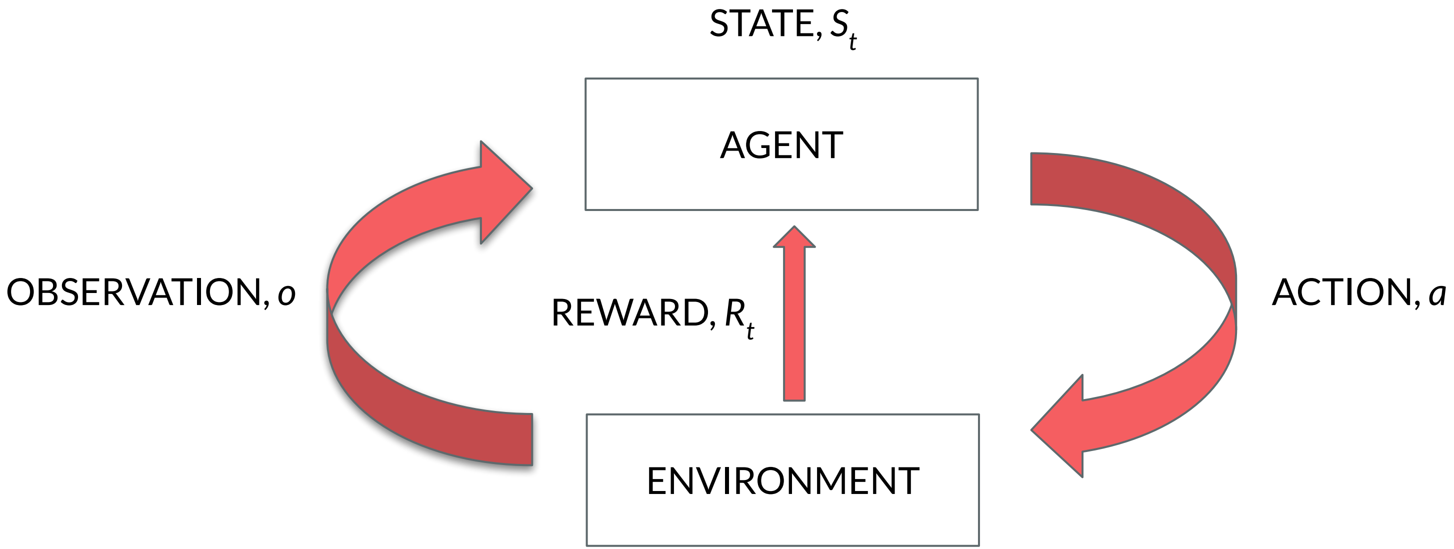
Reinforcement Learning: An Introduction

Benjamin Chew
Insight Data Science, Toronto, Canada

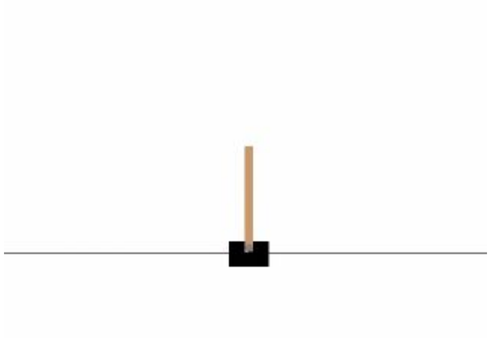
What is Reinforcement Learning?

Agent interacts with its environment to **maximize cumulative reward** over time

Agent **learns** to make **sequences** of good decisions



Reinforcement Learning Examples



CartPole

Breakout

Basketball

Go

StarCraft II

Reinforcement Learning

Optimization

Find an optimal way to make decisions (i.e. yield best rewards)

Delayed Consequences

Introduces problems involving planning and learning

Exploration

“Trial and error”

Generalization*

Delayed Consequences

Planning

Long-term outlook: *Sacrifice immediate reward for future benefits*

Learning

Credit Assignment: *How do we assign credit to the action that lead to a future gain or loss?*

Reinforcement Learning vs Supervised Learning

No supervisors or explicit labels

- Reward signal

Delayed consequences

Time is important

- Sequential, non i.i.d data
- Agent's action influences subsequent observations

Markov Decision Process (MDP)

Reinforcement Learning problems can be formalized as *Markov Decision Processes*

Markov property: "*The future is independent of the past given the present*"

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, ..., S_t]$$

State carries relevant information from history
State is a sufficient statistic of the future

$$P_{ss'} = P[S_{t+1} = s' \mid s_t = s, a_t = a]$$

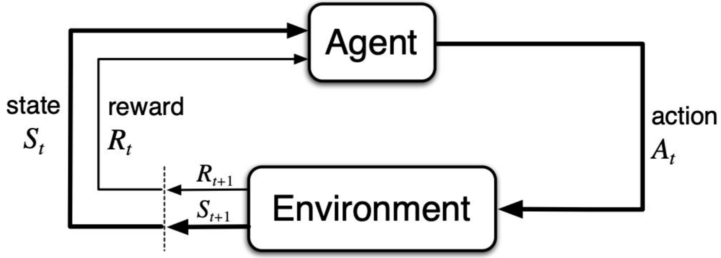
Sequence of random states S_1, S_2, ... S_t with Markov property

Markov Decision Process (MDP)

Abstraction of goal-directed learning from interaction

Three main signals:

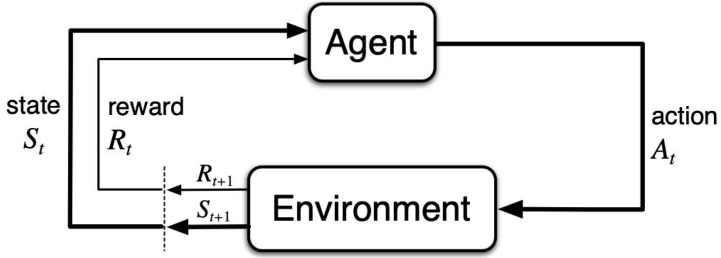
> ***A****: choices made by the agent*
> ***S****: basis on which choices are made*
> ***R****: goal of the agent*



Agent-Environment Interface

‘Four Argument’ Function

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$



Probability Distribution of all States and Rewards

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

State Transition Probabilities

$$\sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

Expected Reward for State-Action Pairs

$$\sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

Expected Reward for State-Action Next-State

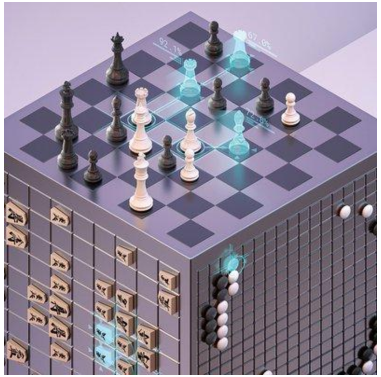
$$\sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

Goals and Rewards

Reward Hypothesis

“That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)”

Reward signal communicates **<u>what</u>** you want achieved, not **<u>how</u>**



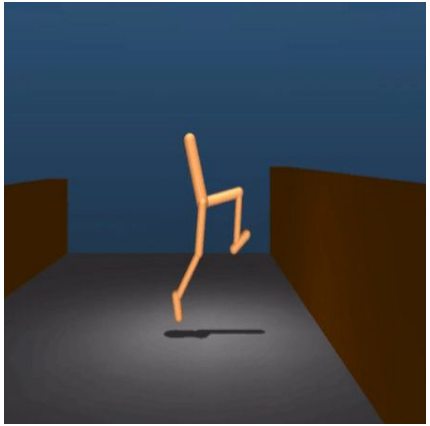
Returns and Episodes

Finite Episodes: *Natural notion of final time step*

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

Continuing Tasks: *State space with no terminal states*

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$



Policy and Value Functions

Policy **π** is a mapping from states to probabilities of selecting each possible action

π(a|s): *probability that $A_t = a$ if $S_t = s$*

State Value Function (under policy π)

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Action Value Function (under policy π)

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

Bellman Equation for v_π

Expresses a relationship between the value of a state and values of its successor states

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[\underline{r + \gamma v_\pi(s')} \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

Bellman Equation for v_π

Expresses a relationship between the value of a state and values of its successor states

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} \underline{p(s', r|s, a)} \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

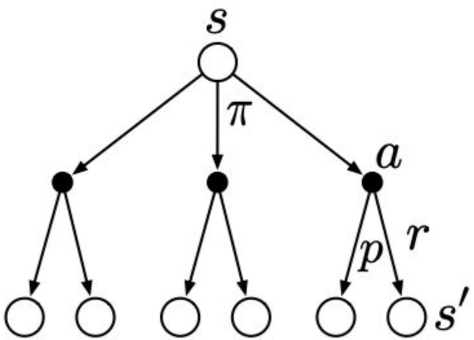
Bellman Equation for v_π

Expresses a relationship between the value of a state and values of its successor states

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \underline{\pi(a|s)} \sum_{s', r} p(s', r | s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

Bellman Equation for v_π

Expresses a relationship between the value of a state and values of its successor states



Dynamic Programming

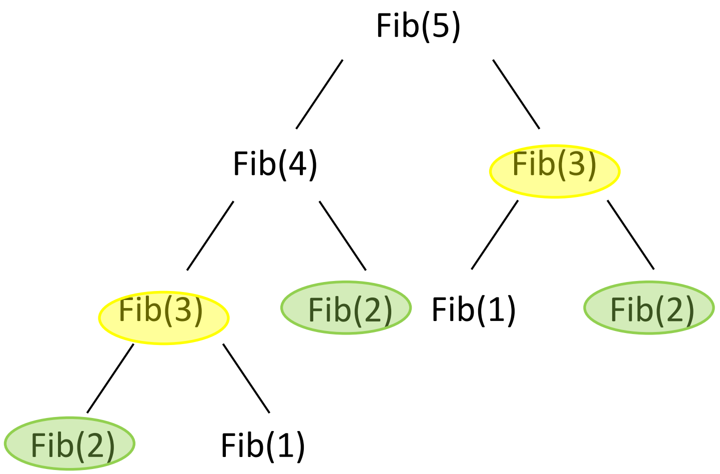
Dynamic	sequential or temporal component to the problem
Programming	optimising a "program", i.e. a policy

Dynamic Programming is a very general solution method for problems which have two properties:

1) Optimal sub-structure
2) Overlapping sub-problems

MDPs satisfy both properties

- Bellman equation gives recursive decomposition
- Value function stores and reuses solutions



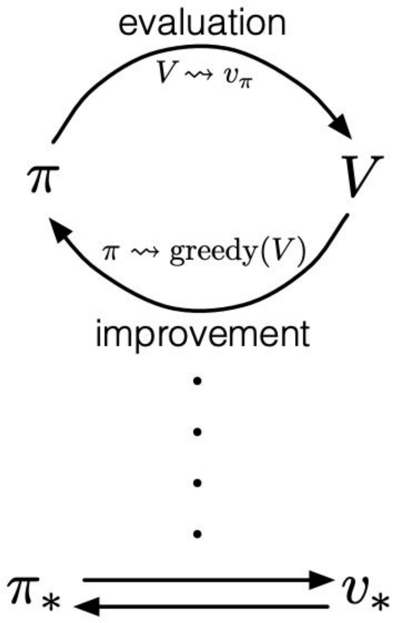
Policy Iteration

Policy Evaluation (Prediction)

Compute state-value function $\boldsymbol{v}_{\pi}$ for policy $\boldsymbol{\pi}$

Policy Improvement

Find policy $\boldsymbol{\pi}'$ that produces greater return $\boldsymbol{v}_{\pi'}$



Small Gridworld

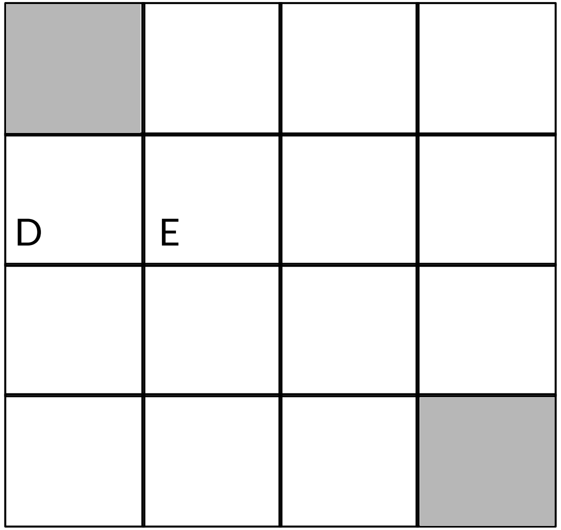
actions 

$R_t = -1$ on all transitions

$p(E, -1 \mid D, \text{right}) = 1$

$p(D, -1 \mid D, \text{left}) = 1$

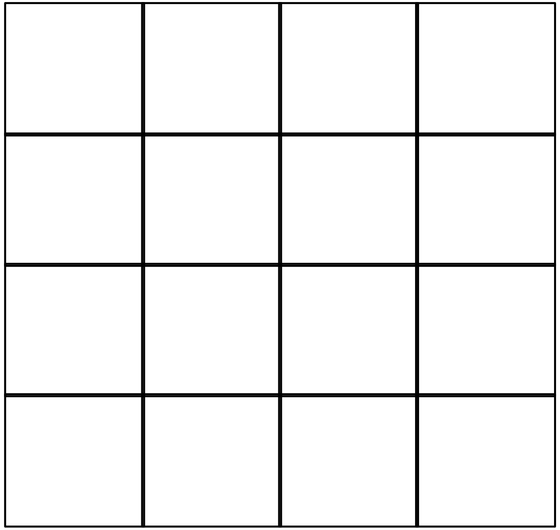
Terminal State



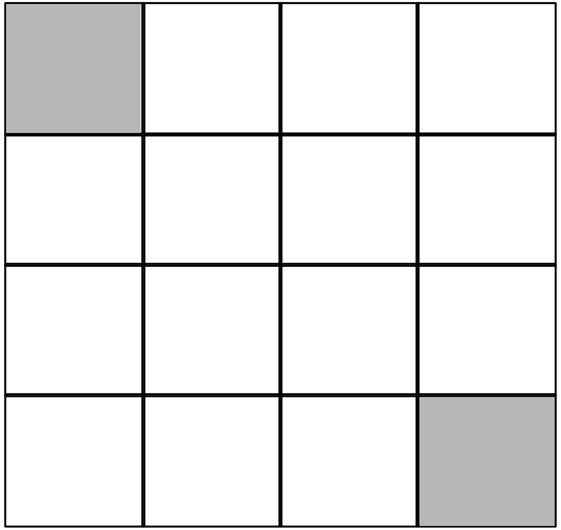
Terminal State

Small Gridworld

V_k for random policy



Greedy policy w.r.t. V_k



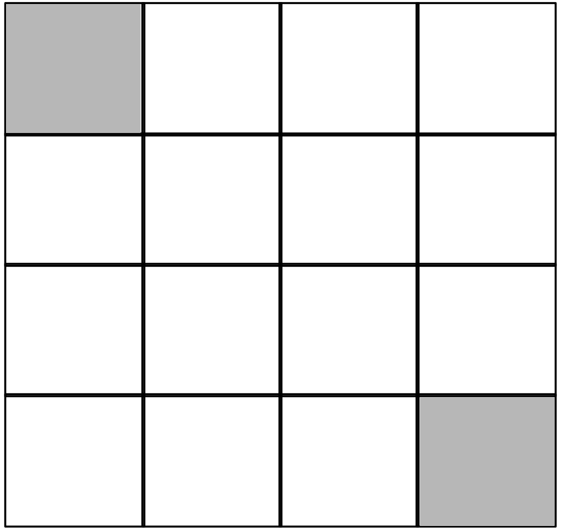
$\boldsymbol{k} = 0$

Random policy

Small Gridworld

V_k for random policy

Greedy policy w.r.t. V_k



$k = 0$

Random policy

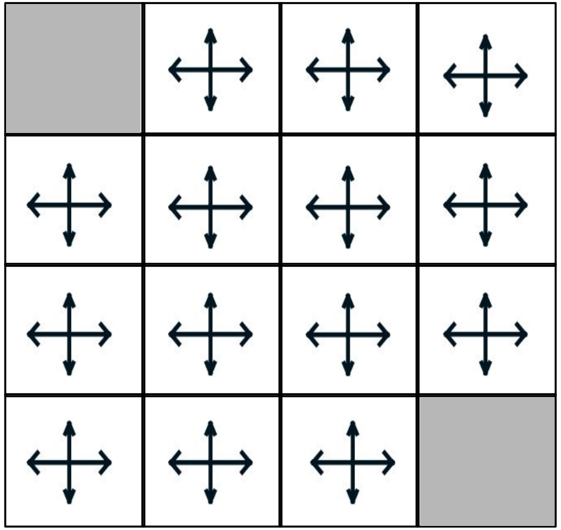
Small Gridworld

V_k for random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Greedy policy w.r.t. V_k



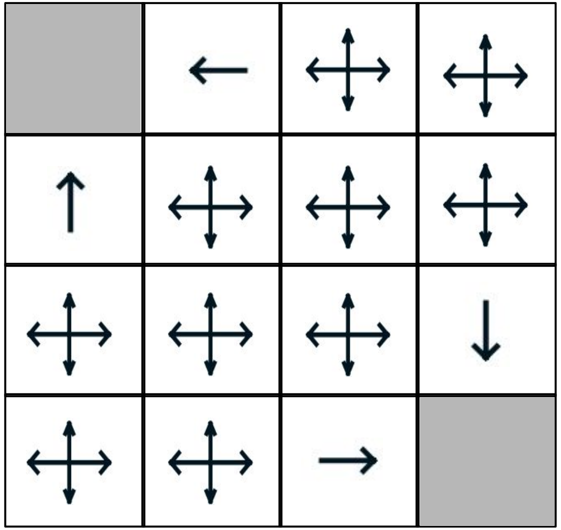
Random policy

Small Gridworld

V_k for random policy

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

Greedy policy w.r.t. V_k



$k = 1$

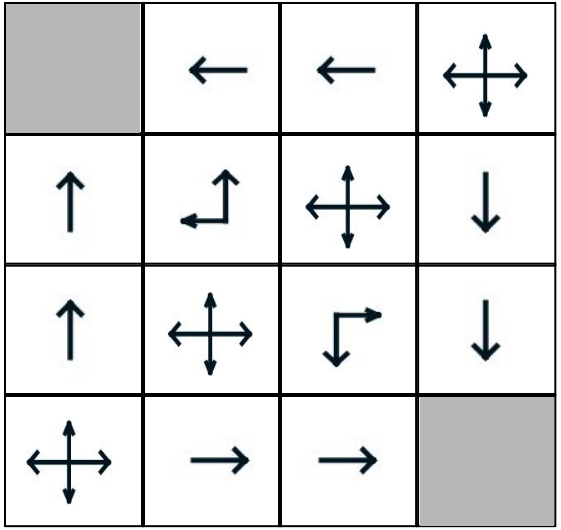
Small Gridworld

V_k for random policy

Greedy policy w.r.t. V_k

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



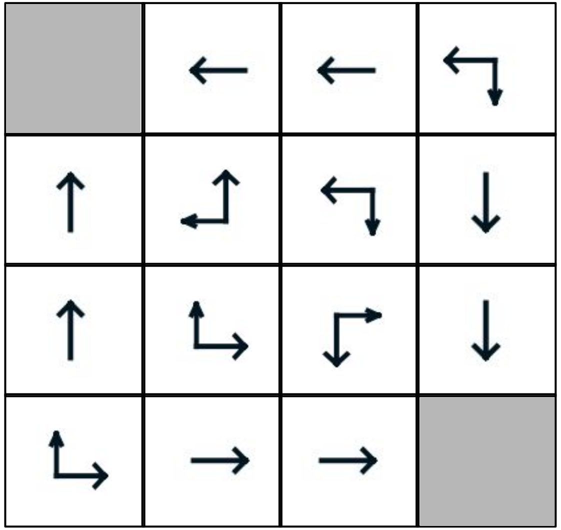
Small Gridworld

V_k for random policy

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

Greedy policy w.r.t. V_k



Optimal policy

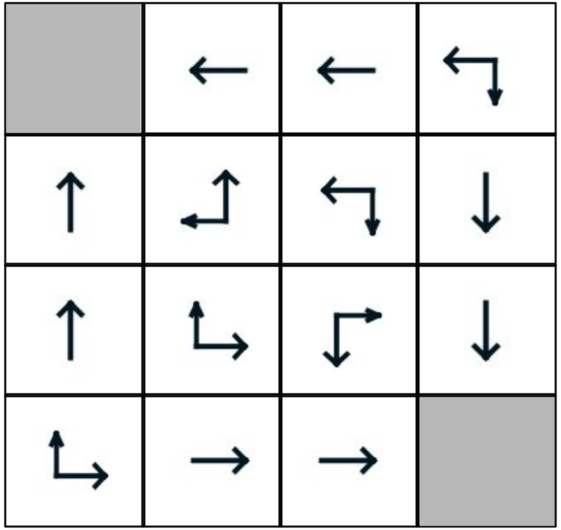
Small Gridworld

V_k for random policy

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

Greedy policy w.r.t. V_k



Optimal policy

Temporal-Difference (TD) Learning

TD methods can learn from direct experience without a model of the environment

TD methods update value estimates without waiting for the final outcome

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$

Reward Hacking

Useful Resources

Reinforcement Learning: An Introduction (aka RL Bible)

http://incompleteideas.net/book/RLbook2018.pdf

David Silver (UCL, Google DeepMind)

https://youtu.be/2pWv7GOvuf0