Wrap-up of Bandits
Sequential decision making
Markov Decision Processes

Admin Issues

- Homework 1 posted, due Jan 27 by midnight
- TA list finalized - please monitor MyCourses for messages from TAs and discussion boards
- Return to in-person classes form Jan 24 - lectures will continue to be recorded and open for zoom attendance for the whole term
- Office hours for Doina will be online (same zoom link as before)
- Up to 20% of lectures can be offered online - we will use this as needed
- Please stay safe and follow university guidelines in terms of masking and self-isolating/caring for others who are sick

Softmax (Boltzmann) Exploration

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Consider $H_t(a) = Q_t(a)/T$

This is Boltzmann or softmax exploration!

If the temperature T is very large (towards infinity) - same as uniform

If temperature T goes to 0, same as greedy

A bit of recap

- Problems vs Solution Methods
- Evaluative vs Instructive feedback
- Associative vs Non-associative learning

Problem space

	Single State	Associative
Instructive feedback		
Evaluative feedback		

Problem space

	Single State	Associative
Instructive feedback		
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback		Supervised learning
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback	Averaging (Imitiation)	Supervised learning
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback	Averaging (Imitiation)	Supervised learning
Evaluative feedback	Bandits (Function optimization)	Contextual bandits

More real motivations...

Clinical trials:





$\mathcal{B}(\mu_1)$ $\mathcal{B}(\mu_2)$ $\mathcal{B}(\mu_3)$ $\mathcal{B}(\mu_4)$ $\mathcal{B}(\mu_5)$

- choose a treatment A_t for patient t
- observe a response $X_t \in \{0, 1\} : \mathbb{P}(X_t = 1) = \mu_{A_t}$
- Goal: maximize the number of patient healed

Recommendation tasks:

ν_1 ν_2 ν_3 ν_4 ν_5

- recommend a movie A_t for visitor t
- observe a rating $X_t \sim \nu_{A_t}$ (e.g. $X_t \in \{1, \ldots, 5\}$)

Linear bandits

- An additive effects model.
- Suppose each round we take a decision $x \in \mathcal{D} \subset \mathcal{R}^d$.
  - x is paths on a graph.
  - x is a feature vector of properties of an ad
  - x is a which drugs are being taken
- Upon taking action x, we get reward r, with expectation:

$$\mathbb{E}[r|x] = \mu^\top x$$

- only d unknown parameters (and "effectively" 2^d actions)
- W desire an algorithm $\mathcal{A}$ (mapping histories to decisions), which has low regret.

$$T\mu^\top x_* - \sum_{t=1}^{T} \mathbb{E}[\mu^\top x_t|\mathcal{A}] \leq ??$$

(where x_* is the best decision)

Regression!

- Define:

$$A_t := \sum_{\tau<t} x_\tau x_\tau^\top + \lambda I, \;\; b_t := \sum_{\tau<t} x_\tau r_\tau$$

- Our estimate of μ

$$\hat{\mu}_t = A_t^{-1} b_t$$

- Confidence of our estimate:

$$\|\mu - \hat{\mu}_t\|_{A_t}^2 \leq \mathcal{O}(d \log t)$$

LinUCB

- Again, optimism in the face of uncertainty.
- Define:

$$B_t := \{\nu | \|\nu - \hat{\mu}_t\|_{A_t}^2 \leq \mathcal{O} d \log t\}$$

- (Lin UCB) take action:

$$x_t = \text{argmax}_{x \in \mathcal{D}} \max_{\nu \in B_t} \nu^\top x$$

  then update A_t, B_t, b_t, and $\hat{\mu}_t$.
- Equivalently, take action:

$$x_t = \text{argmax}_{x \in \mathcal{D}} \ \hat{\mu}_t^\top x + (d \log t) \sqrt{x A_t^{-1} x}$$

What about context?

Clinical trials:





$\mathcal{B}(\mu_1)$ $\mathcal{B}(\mu_2)$ $\mathcal{B}(\mu_3)$ $\mathcal{B}(\mu_4)$ $\mathcal{B}(\mu_5)$

- choose a treatment A_t for patient t
- observe a response $X_t \in \{0, 1\} : \mathbb{P}(X_t = 1) = \mu_{A_t}$
- Goal: maximize the number of patient healed

Recommendation tasks:

ν_1 ν_2 ν_3 ν_4 ν_5

- recommend a movie A_t for visitor t
- observe a rating $X_t \sim \nu_{A_t}$ (e.g. $X_t \in \{1, \ldots, 5\}$)

The Contextual Bandit Game

- Game: for $t = 1, 2, \ldots$
  - At each time t, we obtain context (e.g. side information, user information) c_t
  - Our feasible action set is A_t.
  - We choose arm $a_t \in A_t$ and receive reward r_{t,a_t}.
    (what assumptions on the reward process?)
- Goal: Algorithm $\mathcal{A}$ to have low regret:

$$\mathbb{E}[\sum_t (r_{t,a_t^*} - r_t)|\mathcal{A}] \leq ??$$

where $\mathbb{E}[r_{t,a_t^*}]$ is the optimal expected reward at time t.

How should we model outcomes?

- Example: ad (or movie, song, etc) prediction.
  What is prob. that a user u clicks on an ad a.
- How should we model the click probability of a for user u?
- Featurizations: suppose we have $\phi_{\text{ad}}(a) \in \mathcal{R}^{d_{\text{ad}}}$ and $\phi_{\text{user}}(u) \in \mathcal{R}^{d_{\text{user}}}$.
- We could make an "outer product" feature vector x as:

$$x(a, u) = \text{Vector}(\phi_{\text{ad}}(a)\phi_{\text{user}}(u)^\top) \in \mathcal{R}^{d_{\text{ad}} d_{\text{user}}}$$

- We could model the probabilities as:

$$\mathbb{E}[\textit{click} = 1 | a, u] = \mu^\top x(a, u)$$

  (or log linear)
- How do we estimate μ?

Contextual Linear bandits

- Suppose each round t, we take a decision $x \in \mathcal{D}_t \subset \mathcal{R}^d$ (D_t may be time varying).
  - map each ad/user a to $x(a, u)$.
  - $D_t = \{x(a, u_t) | a \text{ is a feasible ad at time } t\}$
  - Our decision is a feature vector in $x \in D_t$.
- Upon taking action $x_t \in D_t$, we get reward r_t, with expectation:

$$\mathbb{E}[r_t | x_t \in D_t] = \mu^\top x_t$$

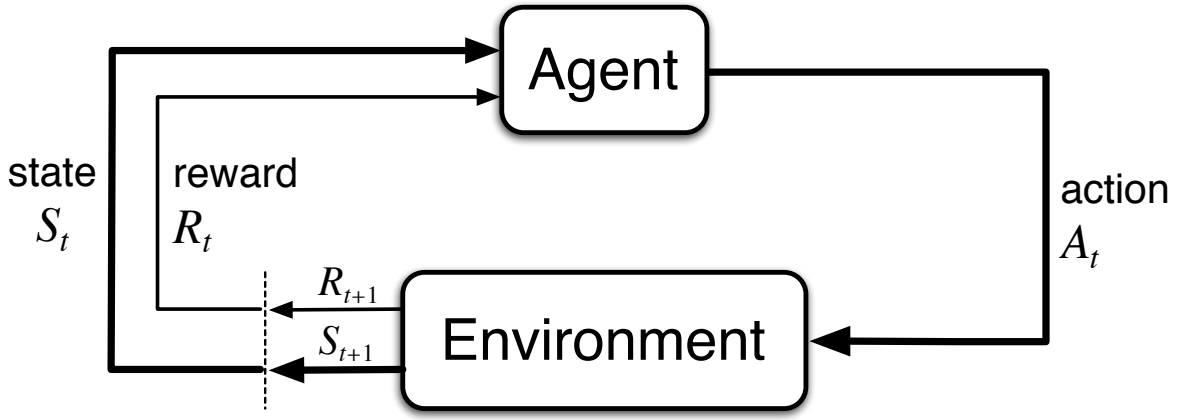
(here μ is assumed constant over time).

- Our regret:

$$\mathbb{E}[\sum_t (\mu^\top x_{t,a_t^*} - \mu^\top x_t) | \mathcal{A}] \leq ??$$

(where x_{t,a_t^*} is the best decision at time t)

The Agent-Environment Interface



Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

Agent observes state at step t: $S_t \in \mathcal{S}$

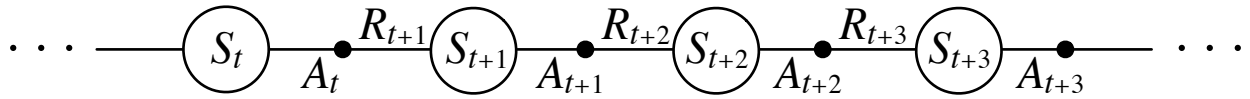
produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$

Trajectory and History

- A sequence of states, actions and rewards



- We will sometimes use the notation $\tau_{ij} = S_i A_i R_{i+1} \ldots S_j$
- We will use the term *history* to refer to the trajectory prior to the current time step
- In general, next states and rewards can depend on the history since the beginning of time

Markov Property

- An assumption about the environment
- Next state and reward depend only on the previous state and action, and noting else that happened in the past

$$p(S_{t+1} = s', R_{t+1} = r \,|\, S_t = s, A_t = a) = p(S_{t+1} = s', R_{t+1} = r \,|\, S_t = s, A_t = a, \tau_t), \forall \tau_t$$

- The assumption is useful to develop, analyze and understand algorithms
- It does NOT mean it has to always hold

Markov Decision Processes

- If a reinforcement learning task has the Markov Property, it is basically a **Markov Decision Process (MDP)**.
- If state and action sets are finite, it is a **finite MDP**.
- To define a finite MDP, you need to give:
  - **state and action sets**
  - one-step "dynamics"

$$p(s', r|s, a) = \mathbf{Pr}\{S_{t+1}=s', R_{t+1}=r \mid S_t=s, A_t=a\}$$

$$p(s'|s, a) \doteq \Pr\{S_{t+1}=s' \mid S_t=s, A_t=a\} = \sum_{r\in\mathcal{R}} p(s', r|s, a)$$

$$r(s, a) \doteq \mathbb{E}[R_{t+1} \mid S_t=s, A_t=a] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s', r|s, a)$$

The Agent Learns a Policy

Policy at step t = π_t =

a mapping from states to action probabilities

$\pi_t(a \mid s)$ = probability that $A_t = a$ when $S_t = s$

Special case - *deterministic policies*:

$\pi_t(s)$ = the action taken with prob=1 when $S_t = s$

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

An Example Finite MDP

Recycling Robot

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: `high`, `low`.
- Reward = number of cans collected

Recycling Robot MDP

$\mathcal{S} = \{\texttt{high}, \texttt{low}\}$

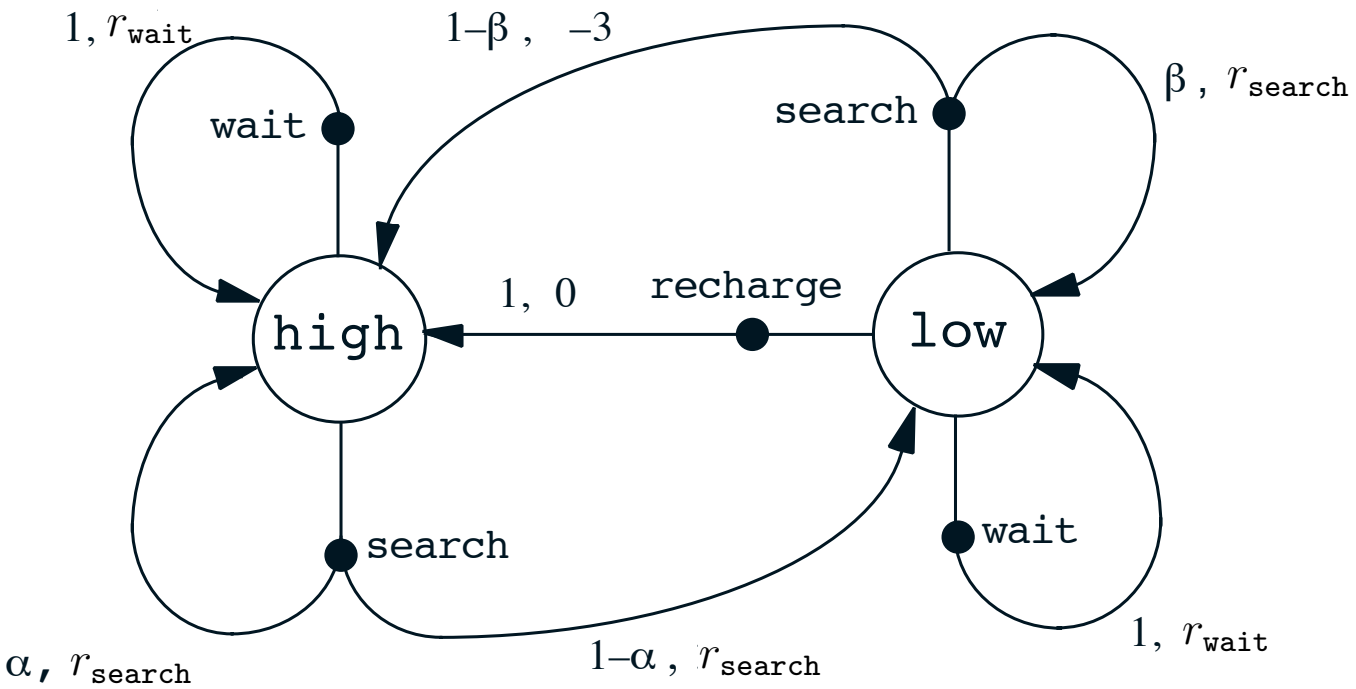
$\mathcal{A}(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$

$\mathcal{A}(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$

$r_{\texttt{search}}$ = expected no. of cans while searching

$r_{\texttt{wait}}$ = expected no. of cans while waiting

$r_{\texttt{search}} > r_{\texttt{wait}}$



The Markov Property

- By "the state" at step t, the book means whatever information is available to the agent at step t about its environment.
- The state can include immediate "sensations," highly processed sensations, and structures built up over time from sequences of sensations.
- Ideally, a state should summarize past sensations so as to retain all "essential" information, i.e., it should have the **Markov Property**:

$$\mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\} =$$
$$p(s', r|s, a) = \mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\}$$

- for all $s' \in \mathcal{S}^+, r \in \mathcal{R}$, and all histories $S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t$.

The Meaning of Life

(goals, rewards, and returns)

Goals and Rewards

- Is a scalar reward signal an adequate notion of a goal?—maybe not, but it is surprisingly flexible.
- A goal should specify **what** we want to achieve, not **how** we want to achieve it.
- A goal must be outside the agent's direct control—thus outside the agent.
- The agent must be able to measure success:
  - explicitly;
  - frequently during its lifespan.

The reward hypothesis

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

- A sort of *null hypothesis*.
  - Probably ultimately wrong, but so simple we have to disprove it before considering anything more complicated

Rewards and returns

- The objective in RL is to maximize long-term future reward
- That is, to choose A_t so as to maximize $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$
- But what exactly should be maximized?
- The discounted *return* at time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

the *discount rate*

$$\gamma \in [0, 1)$$

γ	Reward sequence	Return
0.5(or any)	1 0 0 0…	
0.5	0 0 2 0 0 0…	
0.9	0 0 2 0 0 0…	
0.5	-1 2 6 3 2 0 0 0…	

4 value functions

	state values	action values
prediction	v_π	q_π
control	v_*	q_*

- All theoretical objects, mathematical ideals (expected values)

- Distinct from their estimates:

$$V_t(s) \qquad Q_t(s,a)$$

Values are *expected* returns

- The value of a state, given a policy:

$$v_\pi(s) = \mathbb{E}\{G_t \mid S_t = s, A_{t:\infty} \sim \pi\} \qquad v_\pi : \mathcal{S} \to \Re$$

- The value of a state-action pair, given a policy:

$$q_\pi(s,a) = \mathbb{E}\{G_t \mid S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\} \qquad q_\pi : \mathcal{S} \times \mathcal{A} \to \Re$$

- The optimal value of a state:

$$v_*(s) = \max_\pi v_\pi(s) \qquad v_* : \mathcal{S} \to \Re$$

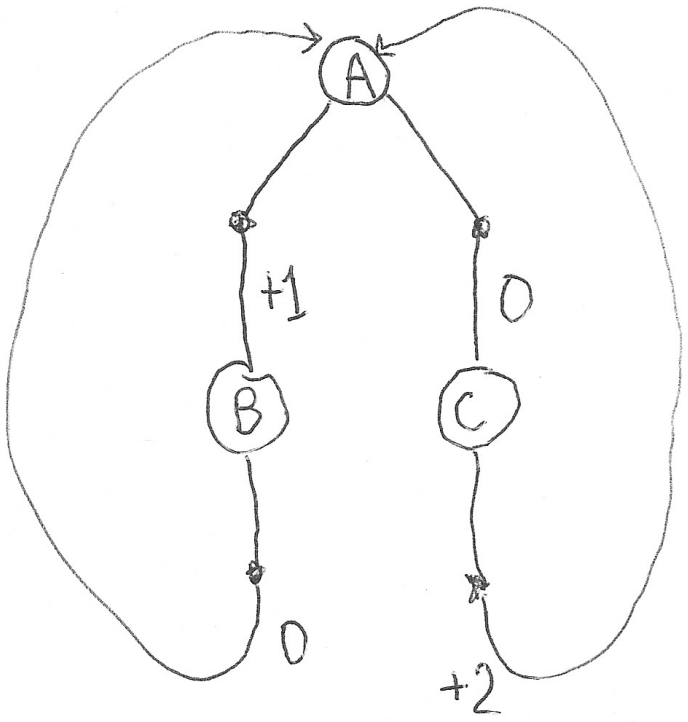
- The optimal value of a state-action pair:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \qquad q_* : \mathcal{S} \times \mathcal{A} \to \Re$$

- Optimal policy: π_* is an optimal policy if and only if

$$\pi_*(a|s) > 0 \text{ only where } q_*(s,a) = \max_b q_*(s,b) \qquad \forall s \in \mathcal{S}$$

  - in other words, π_* is optimal iff it is *greedy* wrt q_*



What policy is optimal?

A: left

B: Right C: Other

If $\gamma = 0$?

If $\gamma = .99$

If $\gamma = \frac{1}{2}$?

Return

Suppose the sequence of rewards after step t is:

$$R_{t+1}, R_{t+2}, R_{t+3}, \ldots$$

What do we want to maximize?

At least three cases, but in all of them,

we seek to maximize the **expected return**, $E\{G_t\}$, on each step t.

- Total reward, G_t = sum of all future reward in the episode
- Discounted reward, G_t = sum of all future *discounted* reward
- Average reward, G_t = average reward per time step

Episodic Tasks

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze

In episodic tasks, we almost always use simple *total reward*:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T ,$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

Continuing Tasks

Continuing tasks: interaction does not have natural episodes, but just goes on and on...

In this class, for continuing tasks we will always use *discounted return*:

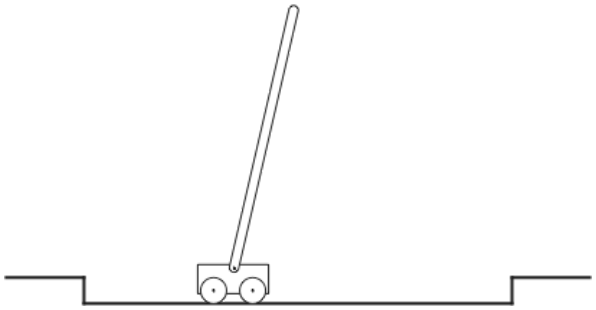
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where γ, $0 \le \gamma \le 1$, is the **discount rate**.

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

Typically, $\gamma = 0.9$

An Example: Pole Balancing



Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As an **episodic task** where episode ends upon failure:

reward = +1 for each step before failure

$\Rightarrow$ return = number of steps before failure

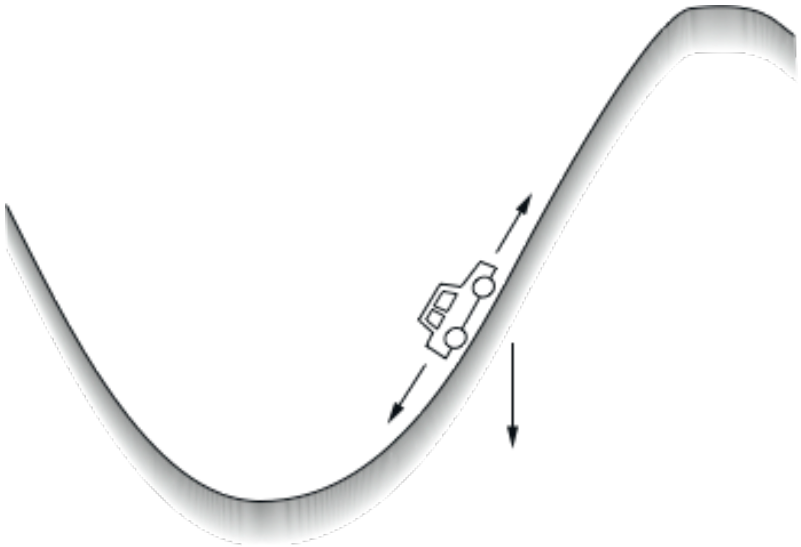
As a **continuing task** with discounted return:

reward = −1 upon failure; 0 otherwise

$\Rightarrow$ return = $-\gamma^k$, for k steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

Another Example: Mountain Car



Get to the top of the hill
as quickly as possible.

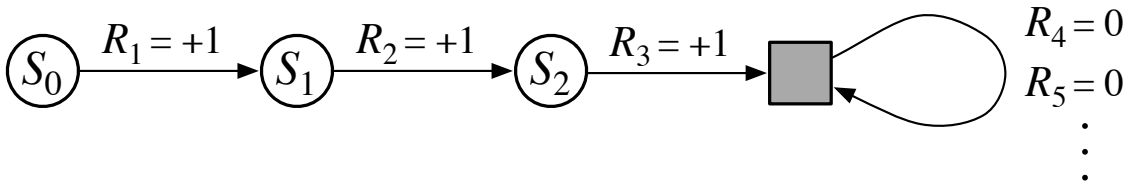
reward = –1 for each step where **not** at top of hill

$\Rightarrow$ return = – number of steps before reaching top of hill

Return is maximized by minimizing
number of steps to reach the top of the hill.

A Trick to Unify Notation for Returns

- In episodic tasks, we number the time steps of each episode starting from zero.
- We usually do not have to distinguish between episodes, so instead of writing $S_{t,j}$ for states in episode j, we write just S_t
- Think of each episode as ending in an absorbing state that always produces reward of zero:



- We can cover <u>all</u> cases by writing $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$,

where γ can be 1 only if a zero reward absorbing state is always reached.

Value Functions

- The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State - value function for policy π :

$$v_\pi(s) = E_\pi\left\{G_t \mid S_t = s\right\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right\}$$

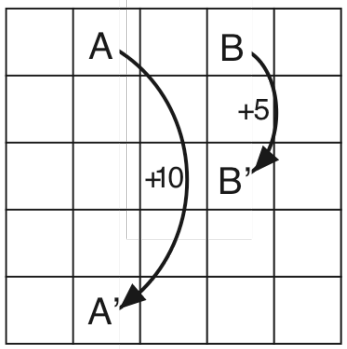
- The **value of an action (in a state)** is the expected return starting after taking that action from that state; depends on the agent's policy:

Action - value function for policy π :

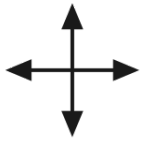
$$q_\pi(s,a) = E_\pi\left\{G_t \mid S_t = s, A_t = a\right\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right\}$$

Gridworld

- Actions: `north`, `south`, `east`, `west`; deterministic.
- If would take agent off the grid: no move but reward = –1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.



(a)



Actions

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

(b)

State-value function for equiprobable random policy; $\gamma = 0.9$

Bellman Equation for a Policy π

The basic idea:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

So:

$$\begin{aligned} v_\pi(s) &= E_\pi \left\{ G_t \,\middle|\, S_t = s \right\} \\ &= E_\pi \left\{ R_{t+1} + \gamma v_\pi \left(S_{t+1} \right) \middle|\, S_t = s \right\} \end{aligned}$$

Or, without the expectation operator:

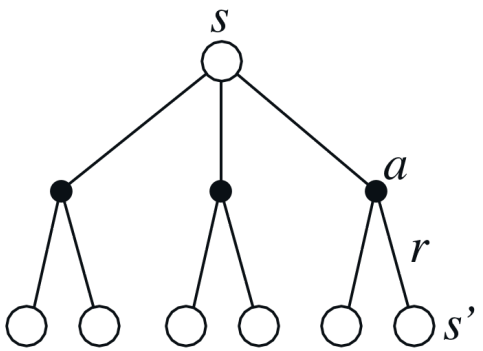
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]$$

More on the Bellman Equation

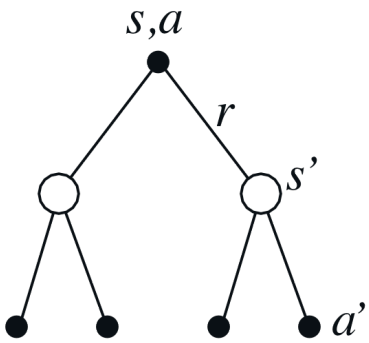
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big]$$

This is a set of equations (in fact, linear), one for each state.
The value function for π is its unique solution.

Backup diagrams:



for v_π



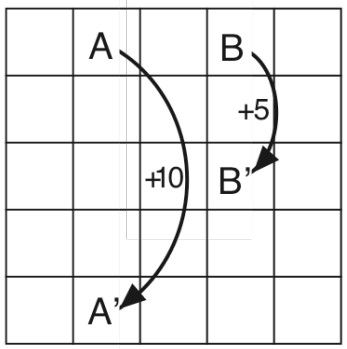
for q_π

Iterative Policy Evaluation – One array version

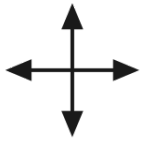
Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Gridworld

- Actions: `north`, `south`, `east`, `west`; deterministic.
- If would take agent off the grid: no move but reward = –1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.



(a)



Actions

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

(b)

State-value function for equiprobable random policy; $\gamma = 0.9$