Fundamentals of Reinforcement Learning

LESSON 01/ THE K-ARMED BANDIT PROBLEM

* understanding the temporal nature of the bandit problem.
* define K-armed bandit.
* define action-values.
* define reward.

LESSON 02/ WHAT TO LEARN? ESTIMATING ACTION VALUES

* define action-value estimation methods.
* define exploration and exploitation.
* select actions greedily using an action-value function.
* define online learning.
* understand a simple online sample-avg action-value estim. method.
* define the general online equation.
* understand const. step size in case of non stationarity.

LESSON 03/ EXPLORATION VS EXPLOITATION TRADEOFF

* cmp short term benefits of exploit vs long term of explore.
* understand optimistic initial values.
* describe the benefits of opt. init. vals for early exploration.
* explain criticism of optimistic initial values.
* desc. the upper confidence bound action selection method.
* define optimism in the face of uncertainty.

WEEK 01 - PAGES 25 - 36 OF RL-BOOK. → LECTURES
PAGES 42 - 43 OF RL-BOOK. → BEFORE ASSIGNMENTS

RL - explore alternatives, find which one probably maxes rewards.

SL - from historical data, try to generalize with least error & no bias.

---

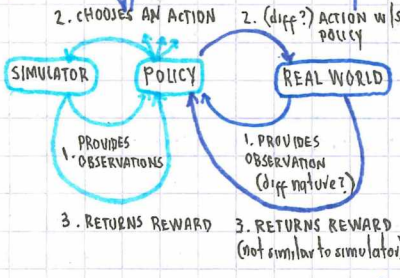
q_* is not known to the agent, just like the doctor doesn't know the effectiveness of each treatment. So, estimate it.

100% = 6 to 12 | 78% = 10, 22% = 12 | 92% = 10, 8% = 144

EXPECTED REWARDS: 9 | 11 | 20.72

REPEATEDLY TRY ACTIONS → FIND MEAN OF REWARDS

John Langford, CTX Bandits for Real World RL/



MIND THE GAP:

LARGE SIMULATOR AND REALITY DIFFERENCE

APPLICABILITY OF SIMULATOR ACQUIRED KNOWLEDGE MAY NOT BE SO FEASIBLE IN REAL WORLD CASES.

How do you do RWRL?

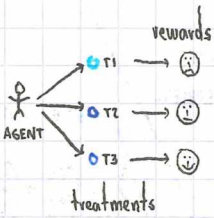
↓ TEMPORAL CREDIT ASSIGNMENT (WHICH MOVE LOST THE GAME)
↓ CONTROL ENVIRONMENT
↓ COMPUTATIONAL EFFICIENCY (GOOD IN SIMUL. - IF LOTS OF SAMPLES)
↓ STATE (FUNDAMTL FOR DECISIONS)
↓ LEARNING
↓ CARE ABOUT LAST POLICY

Shift your priorities!

↑ GENERALIZATION (ACROSS DIFF OBSERVATIONS)
↑ BELIEVE THE ENVIRONMENT CONTROLS YOU
↑ STATISTICAL EFFICIENCY
↑ FEATURES (STATE MAY BE TOO COMPLEX)
↑ OFF-POLICY EVALUATION (SUPPORTING ALGORITHMS)
↑ CARE ABOUT EVERY POLICY

The K-Armed Bandit/

In the **k-armed bandit** problem, we have an **agent** that chooses between "k" **actions** and receives a **reward** based on the action it chooses.



Action Values/

the value is the expected reward : (after action has been taken)

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a] \quad \forall_a \in \{1 \dots k\}$$

- $q_*(a)$: VALUE OF TAKING AN ACTION
- $\doteq$: IS DEFINED AS
- $\mathbb{E}$: EXPECTATION OF R_t
- $|A_t = a$: GIVEN WE SELECT ACTION "a"
- $\forall_a \in \{1 \dots k\}$: FOR ALL/EACH POSSIBLE ACTION 1 THROUGH K

THIS CONDITIONAL EXPECTATION IS DEFINED AS A SUM OF ALL THE POSSIBLE REWARDS

the goal of the agent is to maximize = the expected reward

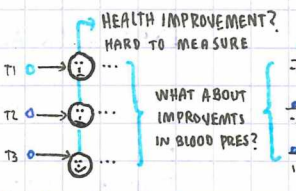
$$= \sum_r p(r|a) * r$$

INSIDE SUM, WE MULTIPLY REWARD WITH PROBABILITY OF OBSERVING THAT REWARD GIVEN ACTION "a".

CAN BE EXTENDED TO CONTINUOUS REWARD CASE BY SWITCHING THE SUMMATION TO AN INTEGRAL.

If the agent selects the action with the highest value, it achieves that goal. this procedure is called the **argmax**

$\underset{a}{\text{argmax}}\ q_*(a)$: the argument that maximizes the function q_*



q_* IS THE MEAN OF THE DISTRIBUTIONS FOR EACH ACTION

$q_*(T1) = .5 * -11 + .5 * 9 = -1$

$q_*(T2) = 1$

$q_*(T3) = 3$

The Sample-Average Method/

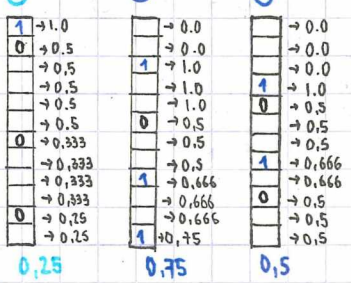
$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

An action is per unit of time so prior actions is $t-1$.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

A REWARD OF 1 IF TREATMENT SUCCEEDS, OTHERWISE 0.

As the doc treats more patients, the ESTIMATES approach the true ACTION VALUES.



But in reality doc doesn't assign random treatments to the patients. they'll give what they currently think it's the best, this is a greedy action.

agent exploits its current knowledge, it tries to get the most reward it can right now.

$$a_g = \underset{a}{\text{argmax}}\ Q_t(a) \Rightarrow 0.75$$

a = T1 T2 T3

a_g: GREEDY ACTION

or the agent can sacrifice immediate reward, seeking to explore to gain more information of other actions.

one of the fundamental problems of RL, is an agent can't do both exploit (maximize current reward short-term) and explore (sacrifice reward right know in order to learn, more beneficial for long-term).

Incremental Action Value estimation

Say you've got millions of requests. It takes a lot of space to store past rewards and be computing again the action value estimates. You really want to only store one single value per treatment, and update it on the go every time an explore is done.

THE INCREMENTAL UPDATE RULE IS AN INSTANCE OF A MORE GENERAL RULE.

THE GENERAL UPD. RULE CAN BE USED TO SOLVE NON-STATIONARY BANDIT PROBLEM.

SAMPLE-AVERAGE METHOD UNTIL N+1

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

make the definition recursive... recursive... recursive... ∅!

EXTRACT CURRENT REWARD OUT OF SUM

$$= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$$

PREVIOUS ESTIMATE, Q_n, SAMPLE-AVERAGE METHOD UNTIL N

$$= \frac{1}{n}\left(R_n + (n-1)\cdot\frac{1}{(n-1)}\sum_{i=1}^{n-1} R_i\right)$$

$$\Downarrow \quad \left(R_n + (n-1)\cdot Q_n\right)\cdot\frac{1}{n}$$

$$\Downarrow \quad \left(R_n + nQ_n - Q_n\right)\cdot\frac{1}{n}$$

$$\Downarrow \quad Q_n + \frac{1}{n}\left(R_n - Q_n\right)$$

INCREMENTAL UPDATE RULE

$$Q_{n+1} = Q_n + \frac{1}{n}\left(R_n - Q_n\right)$$

NEW ESTIMATE · TARGET · GENERAL RULE/FORM · OLD ESTIMATE · STEP SIZE

(this form will show up again many times in future)

the error in the estimate is the difference between the old estimate and the new target. taking a step towards that target will create a new estimate that reduces our error.

the size of the step is determined by step-size parameter and the error of the old estimate.

α_n

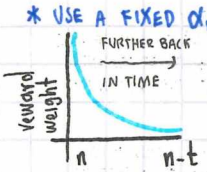
step-size can be an $f(n)$ that produces [0...1]

In the case of sample-average, step-size $f(n) = \frac{1}{n}$

Non-Stationary Bandit Problem

What if one of the treatments suddenly became more effective under certain conditions, like seasonality / winter, which could change the reward distribution.

* USE A FIXED α_n: e.g. "0.1" the most recent reward will weigh more than all previous (which have 0.1 * 0.1..)



weight fades exponentially with time.

Decaying Past Rewards

the most recent contributes most the current estimate.

$$Q_{n+1} = Q_n + \alpha_n\left(R_n - Q_n\right)$$ GENERAL RULE/FORM.

$$= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2\alpha R_{n-2} + \ldots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n\alpha Q_1$$

$$= (1-\alpha)^n Q_1 + \sum_{i=1}^{n}\alpha(1-\alpha)^n R_i$$

INITIAL PREDICTION

REWARDS FURTHER BACK IN TIME CONTRIBUTE EXPONENTIALLY LESS.

Explore/Exploit Tradeoff

EXPLORATION IMPROVE KNOWLEDGE FOR LONG-TERM BENEFIT. INCREASE THE ACCURACY OF THE ESTIMATED ACTION VALUES

EXPLOITATION ACT UPON CURRENT KNOWLEDGE OF BEST FOR SHORT-TERM BENEFIT, CHOOSE THE GREEDY ACTION WHICH VERY EARLY ON IS NOT SO LIKELY TO HAVE THE BEST REW. DISTRIBUTION.

Optimistic Initial Values

this encourages early exploration, instead of assuming the original estimated action values Q to be 0, use a large over-estimation in all of them to make them appealing to greedy selection.

Q $Q_{n+1} = Q_n + \alpha(R_n - Q_n)$ $\alpha = 0.5$ (EXPONENTIAL DECAY)

① ∅ 2.0 +1 $Q_t = 1.5$

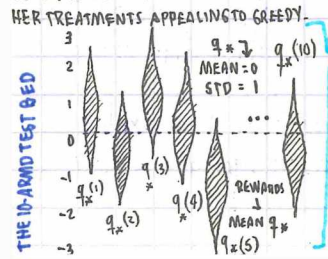
② ∅ 2.0 +0 $Q = 1.0$

③ ∅ 2.0 +1 $Q = 1.5$ +1 $Q = 1.75$ +0 $Q = 0.625$

ENCOURAGES GREEDY-CHOOSING ALL THREE TREATMENTS EARLY ON TIME IT IS A VERY USEFUL HEURISTIC!

IN FIRST ROUND, EITHER EXPLOIT OR EXPLORE WILL CHOOSE AT RANDOM

THE REWARD WAS +1, BECAUSE THE INITIAL ESTIMATE WAS LARGER THAN POSSIBLE REWARDS. THIS WILL BRING ESTIMATE DOWN AND MAKE THE OTHER TREATMENTS APPEALING TO GREEDY.



from textbook, 10 armed bandit. 10 reward distributions sampled from Normal Dist, an initial estimated value of 5 is highly optimistic, all vals in plot are less than 3. Whenever agent selects an action, the observed reward will likely be smaller than original estimation, thus bringing it down, other actions will become more appealing in comparisson.

* ONLY DRIVES EARLY EXPLORATION, STOPS LATER

* HAS ISSUES WITH NON-STATIONARY PROBLEMS

* WE MAY NOT KNOW HOW HIGH TO SET THIS VALUE

Epsilon Greedy Action Selection

to exploit knowledge but not get in a trap of not learning, we can have an exploration ratio, or epsilon greedy, to randomly decide

AGENT → TAKES AN ACTION → DICE ROLL

80% EXPLOITATION

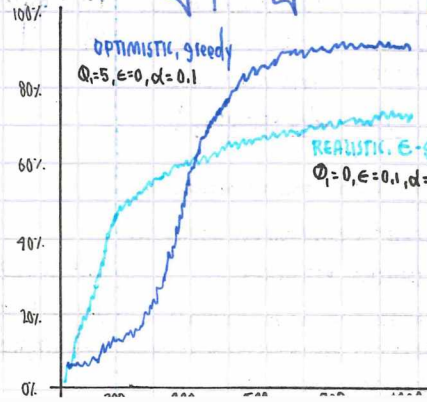
20% EXPLORATION

ITS THE PROBABILITY OF CHOOSING TO EXPLORE

$$A_t \leftarrow \begin{cases} \operatorname{argmax}_a Q_t(a) & \text{with probability } 1-\epsilon \\ a \sim \text{Random Uniform}(\{a, \ldots a_k\}) & \text{with probability } \epsilon \end{cases}$$

ACTION SELECTED AT TIME STEP t

% OPTIMAL ACTION — Percentage of time agent chooses optim. action, avg'd over runs



In early learning, optimistic agent performs worse, because it explores more, its exploration decreases w/ time, because the optimism in it's estim. washes out w/ more samples. but in long run it selects most of the time the most optimal action compared with realistic agent because it explored more early on time.

E-greedy just keeps exploring.

Practice Quiz

What is the incremental rule (sample average) for action values?

R/ $$\underbrace{Q_{n+1}}_{\text{NEW ESTIMATE}} = \underbrace{Q_n}_{\text{OLD ESTIMATE}} + \underbrace{\alpha(n)}_{\frac{1}{n}} \left(\underbrace{R_n}_{\text{TARGET}} - \underbrace{Q_n}_{\text{OLD ESTIMATE}} \right)$$

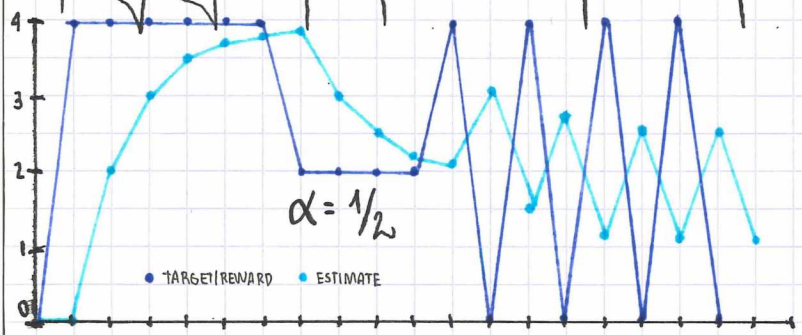
(the bracketed term is the ERROR)

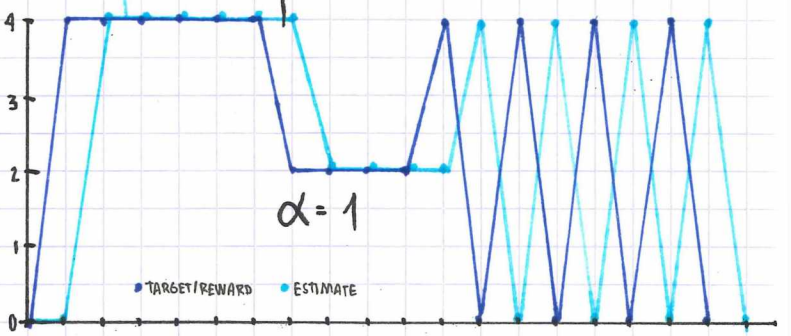
At each time step the agent moves its prediction in the direction of the error by the step size, in the case of the sample-average by $1/n$.

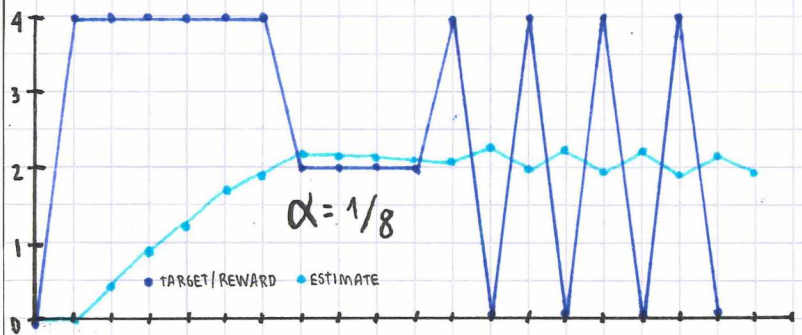
What is the exploration/exploitation tradeoff?

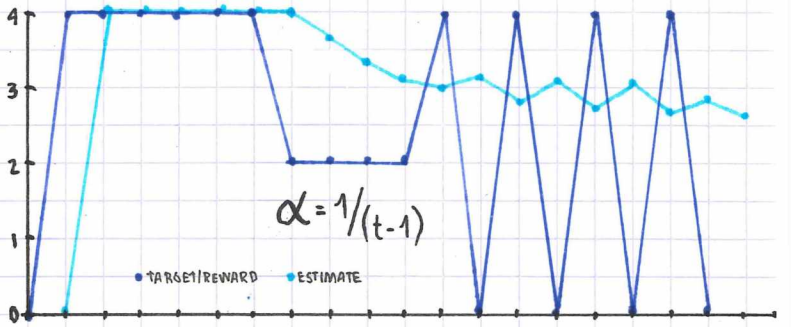
R/ the agent wants to explore to get more accurate estimates of its action values, the agent also wants to exploit its knowledge to get more reward, but it can't choose to do both things simultaneously.

By using the general form of the estimate-update rule, find the α (step size) that updates estimates based on rewards.

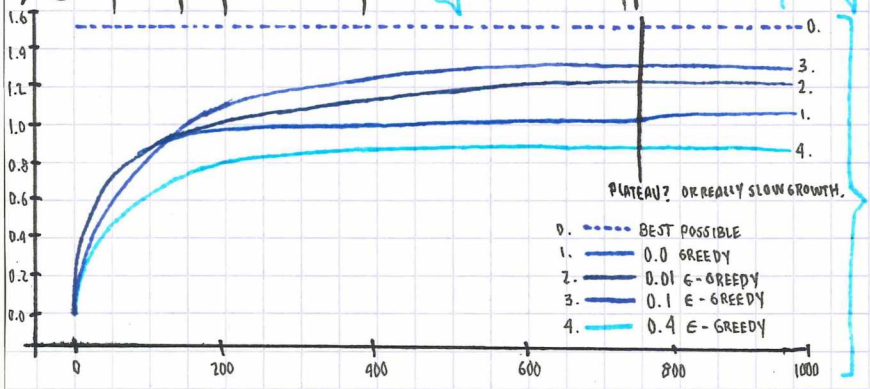








Compare performance of an agent with different ϵ (epsilon greedy values).



0 is the best possible

1 doesn't explore, at first performs better but long run has not much growth. 2 starts similarly but better later.

3 is not so close to 0, 0, starts slow, later really good.

4 explores so much it never quite takes off, need to exploit more the information it has acquired.

Upper-Confidence Bound (UCB) Action Selection

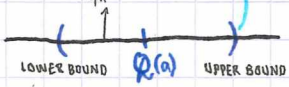
UCB ACTION SELECTION:
$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[\underbrace{Q_t(a)}_{\text{HIGHEST ESTIMATED VALUE}} + \underbrace{c\sqrt{\frac{\ln t}{N_t(a)}}}_{\text{UPPER CONFIDENCE BOUND EXPLORATION TERM. SO MIXES SOMEHOW THE EXPLORATION AND EXPLOIT.}} \right]$$

$$A_t \doteq \begin{cases} \underset{a}{\operatorname{argmax}}\, Q_t(a) & \text{prob } 1-\epsilon \\ a \sim \text{Uniform}(\{a_1, \dots a_k\}) & \text{prob } \epsilon \end{cases}$$

REGULAR EPSILON GREEDY METHOD

the exploratory actions are selected uniformly. could it be done better? If we had a notion of uncertainty in our value estimates, we could potentially select actions on a more intelligent way.

CONFIDENCE INTERVAL AROUND $q_*(a)$ — WE ARE CONFIDENT THE VALUE OF ACTION a LIES SOMEWHERE IN THIS REGION



UCB → OPTIMISM IN FACE OF UNCERTAINTY

IF WE ARE UNCERTAIN ABOUT SOMETHING, WE SHOULD OPTIMISTICALLY ASSUME IT IS GOOD. EXPLORE THE ACTION WITH HIGHEST UPPER BOUND. IF VALUE IS BIG, GREAT, IF NOT, WE LEARNED ABOUT IT & NARROWED.

c: USER SPECIFIED PARAMETER, CONTROLS EXPLORATION.

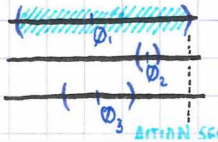
$\ln t$: ln OF AMOUNT OF TIMESTEPS t

$N_t(a)$: NUMBER OF TIMES AITION a HAS BEEN TAKEN UNTIL TIME t

A LESS TAKEN ACTION HAS MORE UNCERTAINTY, SO EXPLORATION TERM FOR IT WILL BE LARGER.

$$c\sqrt{\frac{\ln 10000}{5000}} \Rightarrow 0.043c$$

$$c\sqrt{\frac{\ln 1000}{100}} \Rightarrow 0.303c \quad \sim 10\text{x value!}$$



UCB REDUCES EXPLORATION OVER TIME AS INTERVALS ARE NARROWED, BETTER PERFORMANCE LONG TERM, LIKES TO EXPLORE LESS TAKEN ACTIONS.

Introduction to Markov Decision Processes

LESSON 01 / INTRODUCTION TO MARKOV DECISION PROCESSES
* understand MARKOV DECISION PROCESSES, or MDPs.
* describe how the dynamics of an MDP are defined.
* understand the graphical representation of an MDP.
* explain how diverse processes can be written in terms of the MARKOV DECISION PROCESS framework.

LESSON 02 / GOAL OF REINFORCEMENT LEARNING
* describe how rewards relate to the goal of an agent.
* understand episodes and identify episodic tasks.

LESSON 03 / CONTINUING TASKS
* formulate returns for cont. tasks using discounting.
* describe how returns at successive timesteps are related.
* understand when to formulate a task as episodic or cont.

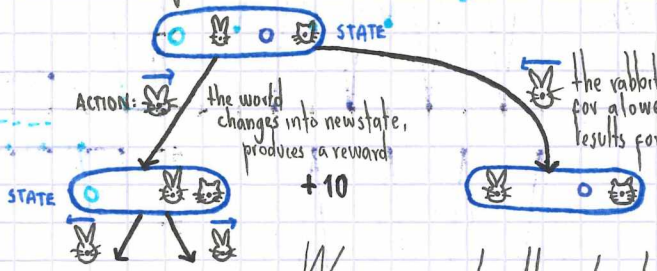
WEEK 02 - PAGES 47 - 56 OF RL-BOOK → LECTURES

Markov Decision Processes

The k-armed bandit problem doesn't include many aspects of real-world problems.
The agent is presented with same situation and each time the action is optimal.
In many real-life problems, different situations call for different responses.
The action we choose now, affect the amount of reward we can get in the future.
The Markov Decision Process formalism captures this two aspects of real-world problems.

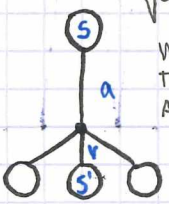
FOR RABBIT IMMEDIATE NEXT STEP WITH MOST REWARD IS THE CARROT, BUT THAT WILL LEAD IT ON A PATH WHERE THE TIGER WILL EAT THEM AND GET A HUGE PENALTY / DEATH. IN LONG TERM IT MAKES THE MOST SENSE THEN TO GO FOR THE BROCCOLI.

A bandit rabbit will only care about immediate reward, but we can make a better decision by considering the long-term impact of our decisions. The situation changes as the rabbit takes actions, we'll call the situations "states".



The Dynamics of an MDP

As in bandits, the outcomes are stochastic, so we have to use the language of probabilities.



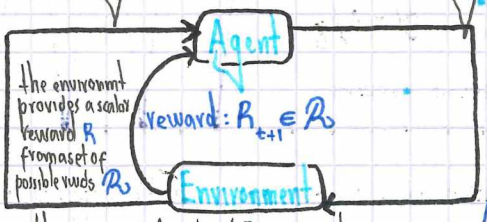
WHEN AGENT TAKES AN ACTION IN A STATE, THERE ARE MANY POSSIBLE NEXT STATES AND ASSOCIATED REWARDS.

transition dynamics function p

$$p(s', r \mid s, a)$$

Given a state s and an action a, the function p tells us the joint probability of next state s' and reward r are. In this course we will typically assume that the set of states, actions, rewards are finite, but will learn algorithms that handle inf or uncountable sets.

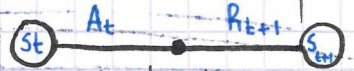
We can generalize this interaction with a general framework:



state: $S_t \in \mathcal{S}$
At each t, the agent receives a state S_t from the environment from a set of possible states $\mathcal{S}$.

action: $A_t \in \mathcal{A}(S_t)$
based on a state, the agent selects an action A_t from a set of possible actions for state $\mathcal{A}(S_t)$.

Agent and Environment interact at discrete timesteps.



The agent-environment interaction generates a trajectory of experience consisting of
* states
* actions
* rewards

The Markov Property

Since p is a probability distribution, it must be non-negative and it's sum over all possible next states and rewards must equal 1. Future state and reward only depends on the current state and action. the present state is sufficient and remembering earlier states would not improve predictions about the future.

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$$

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

Recycling Robot MDP

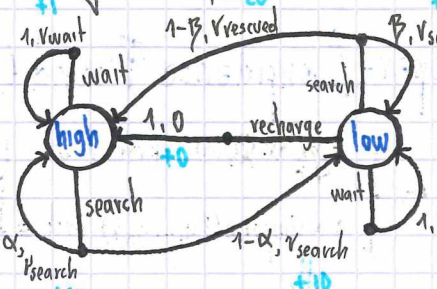
collects empty cans in a office environment. It can detect cans, grab them, and drop them in a recycling bin. The robot runs on rechargeable battery. It seeks to collect as much cans.

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

SENSOR CAN ONLY DETECT TWO CHARGE STATES, LOW AND HIGH.

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

ONLY ALLOW RECHARGING FROM THE LOW ENERGY STATE.



α: probability of remaining high after search.
$1-\alpha$: prob of going to low from high after search.
β: probability of remaining low after search.
$1-\beta$: prob of needing to be rescued after search in low, takes you back to high.

search: done in any state, can bring a reward of 10 but can trigger a state change.
wait: done in any state, can bring a reward of 1, doesn't trigger a state change.
recharge: no reward, done in low to go to high, no alternatives
rescued: if run out of battery will be recharged but involves a penalty.

The Goal of Reinforcement Learning

In RL, the agent's objective is to maximize future rewards. Bandits maximize immediate reward, but this won't work on an MDP. Let's formally define what maximizing **total future reward** is:

RANDOM VARIABLE BECAUSE DYNAMICS OF MDP CAN BE STOCHASTIC.

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

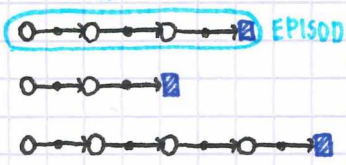
the return at time step t is the sum of rewards obtained onwards

$$\mathbb{E}[G_t] = \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \cdots] + R_T$$

due to randomness in individual rewards an state transitions, lets maximize the expected return, many diff traj. from same state are possible. FOR THIS TO BE WELL DEFINED, SUM OF REWARDS MUST BE FINITE

R_T: FINAL TIMESTEP T, WHERE THE INTERACTN ENDS.

what happens when the interaction ends? EPISODIC TASKS.



In Chess, each game starts at the same state with all pieces reset, it ends with check, draw, or resignation. Each game constitutes an episode.

EPISODIC TASKS	CONTINUING TASKS
* INTERACTION BREAKS NATURALY INTO EPISODES.	* INTERACTION GOES ON CONTINUALLY.
* EACH EPISODE ENDS IN A TERMINAL STATE.	* THERE ARE NO TERMINAL STATES.
* EPISODES ARE INDEPENDENT	* LIKE A THERMOSTAT ALWAYS INTERACTING WITH ENVIRONMENT, STATE IS TEMP, ACTION IS ON/OFF, NEG REWARD IF MANUAL CHANGE.
$G_t = \sum_{i=0}^{n} R_{t+i+1} \mid R_{t+n+1} = R_T$ (TERMINAL STATE)	$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid 0 \le \gamma < 1$

Practice Quiz

* the learner and decision maker is the **agent**.
* at each time step the agent takes an **action**.
* MDP stands for **MARKOV DECISION PROCESS**.

IMAGINE THE AGENT IS LEARNING IN AN EPISODIC PROBLEM:
the number of steps is stochastic, each episode can have a different number of steps occurring.

WHAT IS THE DIFFERENCE BETWEEN SMALL & LARGE GAMMA:
with a larger discount factor, the agent is more far-sighted and considers rewards farther into the future.

WHAT IS THE REWARD HYPOTHESIS:
goals and purposes can be thought of as the maximization of the expected value of the cumulative sum of rewards received.

FORMULATE AS EPISODIC: when agent-environment interaction naturally breaks into sequences. each sequence begins independently of how the episode ended.

FORMULATE AS CONTINUING: when agent-environment interaction does not naturally break into sequences. each new episode begins independently of how the previous episode ended.

WHAT EQUATION(S) DEFINE $q_\pi(s,a)$ IN TERMS OF SUBS. REWARDS?

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

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

or

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \cdots \mid S_t = s, A_t = a]$$

IF REWARD IS ALWAYS 1, THE SUM OF DISCOUNTED INFINITE RETURN WHEN $\gamma < 1$ IS:

$$G_t = 1 \cdot \frac{1}{1-\gamma}$$ CONVERGING GEOMETRIC SERIES WHEN $\gamma < 1$.

$$G_t = \sum_{k=0}^{\infty} \gamma^k * 1$$ CONSTANT

Continuing Tasks

In many problems, the agent-environment interaction continues without end, such problems can be formulated as **CONTINUING TASKS**. How do we formulate the return as we did for episodic tasks, without letting it become infinite?

$0 \le \gamma < 1$ DISCOUNT RATE.

return $G_t \doteq R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{K-1} R_{t+K} + \cdots$

the return is the sum of discounted rewards obtained onwards. immediate rewards contribute more to the sum, far into future less. $1 today is worth more than $1 in a year.

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$ FINITE AS LONG AS $0 \le \gamma < 1$

assume R_{max} is the max reward an agent can receive, so return is bounded like:

$$\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \le \sum_{k=0}^{\infty} \gamma^k R_{max} \to R_{max} \left(\sum_{k=0}^{\infty} \gamma^k\right) \to R_{max} * \frac{1}{1-\gamma}$$

CONSTANT. CONVERGING GEOMETRIC SERIES WHEN $\gamma < 1$

$R_{max} * \frac{1}{1-\gamma}$ * IS FINITE * IS AN UPPER BOUND ON G_t $\therefore$ G_t IS FINITE.

Effect of γ on Agent Behavior

$\gamma = 0$ $G_t = R_{t+1}$ agent only cares about immediate reward, short-sighted.

$\gamma \to 1$ immediate and future R are weighted nearly equally, more far-sighted.

Recursive Nature of The Returns

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

IN FUTURE WE'LL EXPLOIT THIS EQUATION TO DESIGN LEARNING ALGORITHMS.

SUPPOSE $\gamma = 0.8$ AND REWARDS $R_1 = -3$, $R_2 = 5$, $R_3 = 2$, $R_4 = 7$, $R_5 = 1$. $T = 5$, WHAT IS G_0?

$-3 + 5*0.8 + 2*0.8^2 + 7*0.8^3 + 1*0.8^4 = 6.2736$

SUPPOSE $\gamma = 0.5$ AND REWARD SEQUENCE $R_1 = 5$ FOLLOWED BY INF 10s. WHAT IS G_0?

think recursive: $G_2 = \frac{10}{1-0.5} = 50$. $G_1 = 10 + 0.8*(50) = 50$ $G_0 = 5 + 0.8*(50) = 45$

IMAGINE AN AGENT IN A MAZE-LIKE GRID WORLD, WANT THEM TO FIND GOAL ASAP. GIVE A REWARD ON GOAL OF +1. DISCOUNT IS 1 (EPISODIC). WHEN RUN, AGENT FINDS GOAL, BUT DOES NOT CARE HOW LONG IT TOOK. HOW TO FIX THIS?

* give -1 reward at each step * use a discount < 1, something like 0.9
this will incentivize agent to reach goal faster to maximize the return.

IMAGINE YOU ARE A VISION SYSTEM. WHEN 1ST TURNED ON AN IMAGE FLOODS CAMERA. CAN SEE LOTS OF THINGS BUT NOT ALL (OCCLUDED, BEHIND, ETC). AFTER FIRST SCENE, DO YOU HAVE ACCESS TO MARKOV STATE OF ENVIRONMENT? SUPPOSE CAMERA WAS BROKEN THAT DAY, NO IMGS AT ALL DAY, DO YOU HAVE ACCESS TO THE MARKOV STATE THEN?

* you have access to MARKOV STATE before and after damage.
there's no history before first image. it has markov property. the property doesn't tell state tells all that's useful to know, only that hasn't forgotten anything useful. the case with camera broken, future is impoverished but still has the property. all futures are same (black img), nothing needed to remember to predict them.

Policies and Value Functions

LESSON 01/ POLICIES AND VALUE FUNCTIONS
* recognize that a policy is a distribution over actions for each possible state.
* describe the similarities and differences between stochastic & deterministic policies.
* identify the characteristics of a well defined policy.
* generate examples of valid policies for a given MDP.
* describe role of state-value and action-value functions in VI.
* describe the relationship between value functions and policies.
* create examples of valid value functions for a given MDP.

LESSON 02/ BELLMAN EQUATIONS
* derive the Bellman Equation for state-value functions.
* derive the Bellman Equation for action-value functions.
* understand how Bellman Equations relate current and future values.
* use the Bellman Equations to compute value functions.

LESSON 03/ OPTIMALITY (OPTIMAL POLICIES & VALUE FUNCTIONS)
* define an optimal policy.
* understand how a policy can be at least as good as every other policy in every state.
* identify an optimal policy for given MDPs.
* derive the Bellman optimality equation for state-value functions.
* derive the Bellman optimality equation for action-value functions.
* understand how Bellman optimality eqns. relate to previous Bellman eqns.
* understand the connection between the optimal value fn & optimal policies.
* verify the optimal value function for given MDPs.

WEEK 03 - PAGES 58-67 OF RL-BOOK → LECTURES
PAGES 68-69 OF RL-BOOK → SUMMARY BEFORE QUIZZES

Value Functions

Many problems involve some amount of delayed reward. Short-term gain could be maximized, but some strategies may yield better long run rewards. In RL, reward captures the notion of short term gain, the objective is to learn a policy that achieves the most reward in the long-term run. Value functions formalizes what this means.

State-Value Function

"Future reward agent can expect to receive starting from a given particular state"

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

STATE-VALUE FN — EXPECTED RETURN FROM STATE.

π: THE AGENT'S BEHAVIOR WILL DETERMINE HOW MUCH TOTAL REWARD CAN EXPECT, SO A VALUE FUNCTION IS DEFINED WITH RESPECT TO A GIVEN POLICY. π INDICATES V IS CONTINGENT ON THE AGENT SELECTING ACTIONS ACCORDING TO THE POLICY π.

Recall that...

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

RETURN: DISCOUNTED SUM OF FUTURE REWARDS.

Action-Value Function

"Expected return if an agent selects action a and then follows policy π"

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

ACTION-VALUE FN — EXPECTED RETURN WITH a FOLLOWING π ... at a given state s ...

VALUE FUNCTIONS ARE CRUCIAL, ALLOW AN AGENT TO QUERY QUALITY OF ITS CURRENT SITUATION, INSTEAD OF WAITING TO OBSERVE LONG-TERM OUTCOME. THE VALUE FUNCTION SUMMARIZES ALL POSSIBLE FUTURES BY AVG OVER RETURNS. THEY ALSO ENABLE US TO JUDGE THE QUALITY OF DIFFERENT POLICIES.

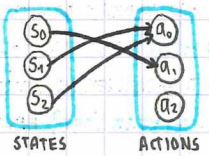
Specifying Policies

"Reinforcement learning is a problem formulation for sequential decision making under uncertainty. The agent's role in this interaction is to choose an action on each time step. The choice of action has an immediate impact in both immediate reward and the next step. policies are how an agent selects these actions.

Deterministic Policies

In simplest case, a policy maps each state to a single action.

$$\pi(s) = a$$



THE AGENT CAN SELECT THE SAME ACTION IN MULTIPLE STATES, AND SOME ACTION MIGHT NOT BE SELECTED IN ANY STATE.

Stochastic Policies

$$\pi(a|s)$$

π OF A GIVEN s, THE PROBABILITY OF SELECTING ACTION a IN STATE s. MULTIPLE ACTIONS COULD BE SELECTED WITH NON-ZERO PROBABILITY.

IN GENERAL, SET OF ACTIONS, $\mathcal{A}$ IS DIFF IN EACH STATE.

$$\sum_{a \in \mathcal{A}(s)} \pi(a|s) = 1$$

THE SUM OVER ALL ACTION PROBABILITIES MUST BE ONE IN EACH STATE NO MATTER THE STATE.

$$\pi(a|s) \geq 0$$

EACH ACTION PROB. MUST NON-NEGT.

↳ DISTRIBUTION OVER ACTIONS, SPECIFIES A SEPARATE DISTRIBUTION FOR EACH DIFFERENT STATE.

It's important that policies depend only on current state, not in time or prev. states. the state defines all the information used to select the current action, think of this as a requirement on the **state**, not a limitation on the **agent**.

* a policy maps the current state onto a set of probabilities for taking each action.
* policies can only depend on the current state, not time or previous states.

Bellman Equations

"In everyday we learn a lot without getting explicit positive or negative feedback. If you're riding a bike and hit a rock, you fell almost and recovered, you may learn to avoid rocks in the future & react more quickly if you do hit one. How do we know hitting a rock is bad even if nothing bad happened this time? we recognize losing balance is bad even without falling/hurting, perhaps we had similar exp. in past.

IN RL A SIMILAR IDEA ALLOWS US TO RELATE THE VALUE OF CURRENT STATE TO VALUE OF FUTURE STATES WITHOUT WAITING TO OBSERVE ALL REWARDS. FORMALIZED W/ BELLMAN EQNS.

State-Value Bellman Equation

"relationship between value of a state and the value of possible successor states"

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \Rightarrow \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

EXPAND EXPECTED RETURN AS A SUM OF POSSIBLE ACTION CHOICES BY AGENT.

$$\sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'] \right]$$

ACTION CHOICE DEPENDS ONLY ON CUR. STATE

POSSIBLE REWARDS AND STATES CONDITIONED ON S & a.

EXPAND OVER POSSIBLE REWARDS AND NEXT STATES CONDITIONED ON STATE S

$v_\pi(s')$: t IS $t+1$, BUT DOESN'T MATTER, BECAUSE NEITHER POLICY OR p DEPEND ON TIME. AND ACTION a.

Action-Value Bellman Equation

"the value of a state-action pair in terms of it's possible successor state-action pairs"

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

ACTION IS ALREADY FIXED.

$$\sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'] \right] \quad q_\pi(s', a')$$

CHANGE EXPECTATION TO BE CONDITIONED ON NEXT STATE AND NEXT ACTION AND THEN SUM OVER ALL POSSIBLE ACTIONS.

$$\sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a'] \right]$$

Why Bellman Equations

the "magic" of value functions is we can use them as stand-in for the average of an infinite num of possible futures.

STATE-VALUE BELLMAN EQUATION

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

- $\sum_a \pi(a|s)$: SUM OVER POSSIBLE ACTION CHOICES MADE BY THE AGENT.
- $\sum_{s'} \sum_r p(s', r \mid s, a)$: SUM OVER POSSIBLE REWARDS AND NEXT STATES CONDITIONED ON STATE s AND ACTION a.

relationship between value of a state and value of possible successor states.

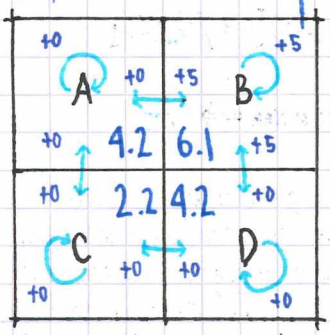
ACTION-VALUE BELLMAN EQUATION

$$q_\pi(s,a) \doteq \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')\right]$$

- DOES NOT BEGIN WITH POLICY SELECTING AN ACTION, IT IS ALREADY FIXED IN THE s, A PAIR.
- SELECT THE IMMEDIATE REWARD r AND THE NEXT STATE s'.
- EXPECTED RETURN ON NEXT STEP IS A SUM OF THE AGENT POSSIBLE ACTION CHOICES, EXPECTATION CONDITIONED ON NEXT STATE, NEXT ACTION, THEN SUM OVER ALL POSSIBLE ACTIONS.

the value of a state-action pair in terms of possible successor state-action pairs.

Gridworld Example



- only four states, A, B, C, D on a grid
- the action space consists of moving ↑, ↓, ←, →, actions which would move agent off-grid just bounce.
- uniform random policy, moves in every direction 25% of the time. π

+5 reward for reaching state B. $\gamma = 0.7$.

$$v_\pi(A) \doteq \mathbb{E}_\pi[G_t \mid S_t = A]$$
$$v_\pi(B) \doteq \mathbb{E}_\pi[G_t \mid S_t = B]$$
$$v_\pi(C) \doteq \mathbb{E}_\pi[G_t \mid S_t = C]$$
$$v_\pi(D) \doteq \mathbb{E}_\pi[G_t \mid S_t = D]$$

value function is defined as expected return under policy π. it's an avg over the return obtained by each seq. of actions an agent could choose.

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

for an action a there's only one state & reward.

$$v_\pi(A) = \sum_a \pi(a|s)\left[r + 0.7 v_\pi(s')\right]$$

$$v_\pi(A) = \frac{1}{4}\left(5 + 0.7 v_\pi(B)\right) \text{ MOVE RIGHT} + \frac{1}{4}\left(0 + 0.7 v_\pi(C)\right) \text{ MOVE DOWN} + \frac{1}{4}\left(0 + 0.7 v_\pi(A)\right) \text{ MOVE LEFT} + \frac{1}{4}\left(0 + 0.7 v_\pi(A)\right) \text{ MOVE UP} \Big\} \frac{1}{2} 0.7 v_\pi(A)$$

EACH MOVEMENT IS EQUALLY LIKELY THANKS TO THE UNIFORM POLICY.

MANY INDOGNITAS, BUILD AN EQUATION SYSTEM TO SOLVE.

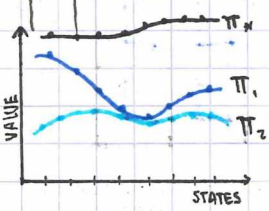
Using Bellman Equation, we can write down an expression for the value of state A in terms of the sum of the four possible actions and the resulting possible successor states. In this case can be simplified because for each action there's only one possible associated state and reward.

$$v_\pi(A) = \frac{1}{4}\left(5 + 0.7 v_\pi(B)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(C)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(A)\right) = 4.1667 \sim 4.2$$
$$v_\pi(B) = \frac{1}{4}\left(5 + 0.7 v_\pi(B)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(A)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(D)\right) = 6.0897 \sim 6.1$$
$$v_\pi(C) = \frac{1}{2}\left(0 + 0.7 v_\pi(C)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(D)\right) + \frac{1}{4}\left(0 + 0.7 v_\pi(A)\right) = 2.2435 \sim 2.2$$
$$v_\pi(D) = \frac{1}{4}\left(0 + 0.7 v_\pi(C)\right) + \frac{1}{4}\left(5 + 0.7 v_\pi(B)\right) + \frac{1}{2}\left(0 + 0.7 v_\pi(D)\right) = 4.1667 \sim 4.2$$

Here, the Bellman Equation reduced an unmanageable infinite sum over possible futures, to a simple linear algebra problem. This may be possible for MDPs of moderate size, in complex problems may end up with 10^{45} (chess) linear equations. However, algorithms based on Bellman Equations can scale up well for large problems.

Optimal Policies

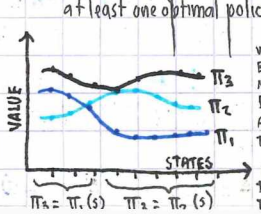
We have talked about a policy as something that is given. It specifies how an agent behaves, given this way of behaving, we then aim to find a value function. But RL does not only seek to evaluate specific policies but we want to obtain a policy that obtains as much reward as possible in long run.



$\pi_1 \geq \pi_2$: π_1 IS AS GOOD AS OR BETTER THAN π_2 IFF THE VALUE ON π_1 IS $\geq$ VALUE ON π_2 FOR ALL POSSIBLE STATES.

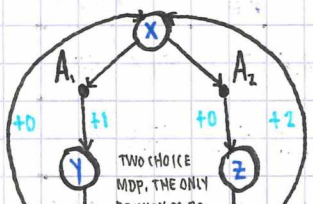
continuous/smooth only for illustrative purposes, really is discrete and values for states on a π may not go smooth.

π_*: optimal policy, good as or better than all the other policies, will have the highest value on every state. there will always exist at least one optimal policy, there can be more than one.



WE COULD HAVE TO POLICIES IN WHICH ONE IS THE BEST IN SOME STATES BUT NOT OTHERS. IS THERE NO OPTIMAL POLICY? WE CAN COMBINE THOSE POLICIES INTO A THIRD POLICY WHICH CHOOSES ACTIONS ACCORDING TO π_1 OR π_2 DEPENDING ON THE CURRENT STATE.

IN SOME STATES π_3 IS STRICTLY GREATER THAN π_1, AND π_2, BEING OPTIMAL BOOSTS THE VALUES OVER MANY STATES.



TWO CHOICE MDP, THE ONLY DECISION TO BE MADE IS AT X

- A_1 offers an immediate reward of +1.
- A_2 offers a better +2 reward after delay.
- there are only two deterministic policies.

$\pi_1(X) = A_1 \qquad \pi_2(X) = A_2$

Optimal policy? ... depends on gamma γ

$\gamma = 0$: $v_{\pi_1}(x) = 1 \quad v_{\pi_2}(x) = 0 \quad \pi_1$!

$\gamma = 0.9$: $v_{\pi_1}(x) = 1 + 0.9 * 0 + 0.9^2 * 1 + \dots$

$5.3 \approx \frac{1}{1 - 0.9^2} \Leftarrow \sum_{k=0}^{\infty} 0.9^{2k}$ (exclude odd steps)

$v_{\pi_2}(x) = 0 + 0.9 * 2 + 0.9^2 * 0 + \dots \quad \pi_2$!

$2 \sum_{k=0}^{\infty} 0.9^{2k+1} \quad (0.9^{2k} * 0.9) \Rightarrow 2 * 0.9 * \frac{1}{1 - 0.9^2} \rightarrow 9.5$

We can only directly solve small MDPs...

$\Longrightarrow$ 2 deterministic policies $\Longrightarrow$ brute-force search

General MDP $\Rightarrow |A|^{|S|}$ deterministic policies $\Rightarrow$ X unfeasible to do brute-force search.

the number of possible policies is equal to the number of possible actions to the number of possible states. luckily there is another way to organize our search of the policy space. the solution comes in the form of another set of bellman equations called the **bellman optimality equations**.

- an **optimal policy** is defined as a policy with the highest **highest possible value function** in **all states**.
- **atleast one** optimal policy always exist, but there may be more than 1.
- the **exponential number** of possible deterministic policies makes searching for the optimal policy by brute force intractable.

Optimal Value Functions

REMINDER: $\pi_1 \geq \pi_2$ if and only if $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all $s \in \mathcal{S}$

An optimal policy is one that is good as or better than any other policy, thus has the greatest value possible in every state.

THIS HOLDS FOR EVERY STATE IN OUR STATE SPACE

$$v_{\pi_*}(s) = \mathbb{E}_{\pi_*}[G_t \mid S_t = s] = \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S} \quad \text{(for all)} \quad \rightarrow v_*$$

the value of optimal policy π_* = maximum value over all policies for all s.

OPTIMAL POLICIES ALSO SHARE SAME OPTIMAL ACTION-VALUE FUNCTION

$$q_{\pi_*}(s,a) = \max_\pi q_\pi(s,a) \quad \text{for all } s \in \mathcal{S} \text{ and all } a \in \mathcal{A} \quad \rightarrow q_*$$

the act-val value of opt. policy π_* = max value over all policies for all s and a.

BELLMAN EQUATION FOR STATE-VALUE FUNCTION ⇒ BELLMAN OPTIMALITY EQN FOR v_*.

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

(v_π → v_*; π → π_*; $v_\pi(s')$ → $v_*(s')$)

THIS EQUATION HOLDS FOR ANY POLICY, EVEN THE OPTIMAL POLICY. WE CAN SUBSTITUTE THE OPTIMAL POLICY IN IT AND SIMPLIFY:

because it's an optimal policy, we can rewrite the equation in a special form which does not reference the policy itself.

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

there always exist an optimal deterministic policy, one that selects an optimal action in every state, such a deterministic optimal policy will assign probability 1 for an action that achieves the highest value, and probability 0 for all other actions. we can express this by replacing $\sum_a \pi_*(a|s)$ with a max over a.

BELLMAN EQUATION FOR ACTION-VALUE FUNCTION ⇒ BELLMAN OPTIMALITY EQN FOR q_*.

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

(q_π → q_*; π → π_*; $q_\pi(s',a')$ → $q_*(s',a')$)

THIS EQUATION HOLDS FOR ANY POLICY, INCLUDING TEH OPTIMAL POLICY. WE CAN REMOVE REFERENCES TO POLICY AND SIMPLIFY:

$$q_*(s,a) = \sum_{s'} \sum_r p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right]$$

the Bellman Optimality Equations relate the value of a state or state-action pair, to its possible successors under any optimal policy.

the Bellman Optimality Equations gives us a similar system of equations for the optimal value, but we can't solve this system in a similar way to find the optimal state-value function, as taking the maximum over actions is not a linear operation, so standard L.A. techniques won't apply.

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

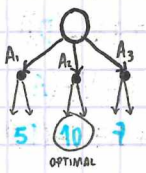
NOT LINEAR

$\pi, p, \gamma \Rightarrow$ LINEAR SYSTEM SOLVER $\Rightarrow v_\pi$

Instead we'll use other techniques based on Bellman EQNs to compute value fns & policies.

Using Optimal Value Functions to Get Optimal Policies

the optimal value functions are not the ultimate goal, but we want to find the optimal policy itself, but given an optimal value function it's possible to find the associated optimal policy (one of them).

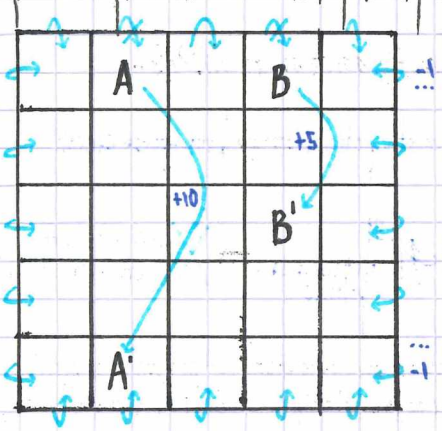


$$v_*(s) = \max_a \sum_{s'} \sum_r p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$ MAXIMUM OVER ALL ACTIONS, DETERMINES VALUE

$$\pi_*(s) = \arg\max_a \sum_{s'} \sum_r p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$ PARTICULAR ACTION TO ACHIEVE MAXIMUM. IF THERE'S A TIE WE CAN BREAK IT STOCHASTICALLY OR WHATEVER WE WANT.

HAVING v_* MAKES IT EASY TO FIND AN OPTIMAL POLICY AS LONG AS WE ALSO HAVE ACCESS TO DYNAMICS FN p.

→ WE NEED ONLY ONE STEP LOOK AHEAD OVER NEXT s' & r.



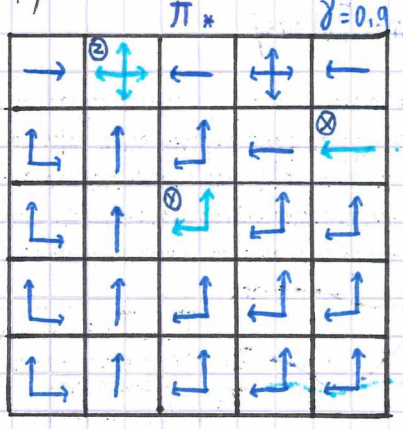
v_* $\gamma = 0.9$

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_* values for all states

π_* $\gamma = 0.9$

Actions that any π_* should take



THE OPTIMAL ACTION IN EACH STATE MUST BE SELECTED BY EVERY OPTIMAL POLICY.

* all actions on state A transition to state A' with a reward of +10
* all actions on state B transition to state B' with a reward of +5
* all 'wall bounces' transition to the same state with a reward of -1
* any other state transition has no rwd.
* the discount factor is $\gamma = 0.9$

* associated optimal values for each step (requires a lot to get this table)
* values at bottom not negative or too low, we're not following anymore a random uniform policy, but an optimal policy given these values.
* the optimal policy won't choose to bump into walls.
* the optimal value of state A is much higher than the immediate reward of +10.

⊗: A ONE STEP LOOK AHEAD CONSIDERS EACH ACTION AND THE POTENTIAL NEXT STATES AND REWARDS. THIS CASE IS SIMPLE BECAUSE EACH ACTION TAKES US DETERMINISTICALLY TO ONE STATE AND ONE REWARD. (s,a ⇒ one s', r).

↑: 0 + 0.9 * 17.5 ←: 0 + 0.9 * 17.8 | ↑: 14.0 ←: 16.0

↓: 0 + 0.9 * 14.4 →: -1 + 0.9 * 16.0 | ↓: 13.0 →: 13.4

FOR MAXIMIZING ← ACTION THE RESULT IS 16.0, WHICH IS EQUAL TO $v_*(s)$.

⊗: IN THIS STATE, TWO DIFFERENT ACTIONS ↑, ← GIVE SAME OPTIMAL VALUE (0 + 0.9 * 19.8) = 17.8. IN THIS STATE THERE ARE TWO DIFFERENT OPTIMAL ACTIONS, AND AN OPTIMAL POLICY IS FREE TO CHOOSE EITHER WITH SOME PROBABILITY.

②: REGARDLESS OF ACTION IN STATE A WE TRANSITION TO A' WITH +10

Policies and Value Functions Summary

policies tell an agent how to behave in their environment

value functions capture the future total reward/return under a particular policy.

DETERMINISTIC POLICIES: state $\xrightarrow{\pi}$ action — maps each state to an action.

STOCHASTIC POLICIES: state $\xrightarrow{\pi}$ actions — maps each state to a distribution over all possible actions. $\pi(s) = a$

STATE-VALUE FUNCTION: $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$ — the expected return from current state under a policy.

ACTION-VALUE FUNCTION: $q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ — the expected return from current state if the agent first selects action a and then follows policy π after.

by definition, a policy depends only on the current state, can't depend on time or previous states. this best thought as a restriction on the state, not on the agent.

Value functions simplify things by aggregating many possible future returns into a single value.

Bellman eqns define a relationship between value of a state/state-action pairs and its possible successors.

the goal: to find a policy that obtains as much reward possible

Bellman equations help us evaluate policies, but they do not achieve the ultimate goal of finding an optimally performing policy.

BELLMAN EQN STATE-VALUE FN: $v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$ — value of current state as a sum of the values of all the successor states & rewards.

BELLMAN EQN ACTION-VALUE FN: $q_\pi(s,a) = \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(a', s')\right]$ — value of state-action pair as a sum over the values of all possible next state-action pairs and rewards.

An optimal policy achieves the highest val possible in all states.

the Bellman Equations can be directly solved (e.g. with a linear system) to find the value functions. however, the optimal Bellman Equations cannot be solved linearly due to the use of 'max' in them.

OPTIMAL STATE-VALUE FN: $v_{\pi_*}(s) = \max_\pi v_\pi(s) \;\forall s \in \mathcal{S}$.

OPTIMAL ACTION-VALUE FN: $q_{\pi_*}(s,a) = \max_\pi q_\pi(s,a) \;\forall s \in \mathcal{S}, a \in \mathcal{A}$.

LIKE ALL VALUE FN, THESE OPTIMAL VAL FN HAVE BELLMAN EQ.

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

$$q_*(s,a) = \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right]$$

these Bellman Equations do not reference a specific policy, this amounts to replacing the policies in the Bellman Equations with a max over all actions.

Practice Quiz

* A policy is a function which maps from states to probability distributions over actions

  THIS STATEMENT IS GENERAL BECAUSE IT INCLUDES STOCHASTIC POLICIES, BUT ALSO DETERMINISTIC SINCE THE DISTRIBUTION CAN HAVE A SINGLE ACTION WITH PROBABILITY 1.

* the term backup most closely resembles update.

* it is true that at least one deterministic optimal policy exists in every MARKOV DECISION PROCESS.

  IF WE HAVE A POLICY π_1 THAT PERFORMS BETTER IN SOME STATES, AND A POLICY π_2 THAT PERFORMS BETTER IN OTHERS, A SPLICED POLICY π_3 CAN BE FORMED SO THAT DEPENDING ON THE STATE, THE ACTION FROM EITHER π_1 OR π_2 IS TAKEN.

* adding a constant to all rewards changes the set of optimal policies in episodic tasks

  THIS IS BECAUSE CAN MAKE EPISODES LONGER (REDUCED PENALTY) OR ACTIONS LESS ATTRACTIVE DEPENDING ON THE SIGN OF THE CONSTANT.

* adding a constant to all rewards doesn't change the optimal policies in continuing tasks

  SINCE THE TASK IS CONTINUING, THE AGENT WILL ACCUMULATE THE SAME AMOUNT OF EXTRA REWARD W/E BEHAVIOUR.

* the optimal state-value function is unique in every finite Markov Decision Process.

  the Bellman optimality equation is actually a system of equations, one for each state, so if there are N states, then there are N equations with N unknowns. If the dynamics of the environment are ~~unknown~~ known, then in principle one can solve this system of equations for the optimal value function using any one of the variety of methods for solving systems of non-linear equations.

  All optimal policies share the same optimal state-value function.

* Select the eqn that relates v_* to q_*, assume π RANDOM UNIFORM.

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

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

  c) $v_*(s) = \sum_{a,r,s'} \pi(a|s) p(s', r|s, a)\; q_*(s')$

  (d) $v_*(s) = \max_a q_*(s,a)$

* Select the eqn that maps q_* to v_*, using 4-arg fn p.

  a) $q_*(s,a) = \sum_{s',r} p(s', r|s, a)\left[r + V_*(s')\right]$

  b) $q_*(s,a) = \sum_{s',r} p(s', r|s, a)\,\gamma\left[r + V_*(s')\right]$

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

* Give an eqn for some π_* in terms of v_* & 4-arg p.

  a) $\pi_*(a|s) = 1$ if $v_*(s) = \max_{a'} \sum_{s'r} p(s', r|s, a)\left[r + \gamma v_*(s')\right]$ else 0

  b) $\pi_*(a|s) = \max_{a'} \sum_{s'r} p(s', r|s, a')\left[r + \gamma v_*(s')\right]$

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

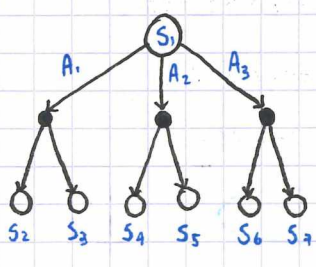
  (d) $\pi_*(a|s) = 1$ if $v_*(s) = \sum_{s'r} p(s', r|s, a)\left[r + \gamma v_*(s')\right]$ else 0

* Write a policy π_* in terms of q_*

  a) $\pi_*(a|s) = q_*(s,a)$

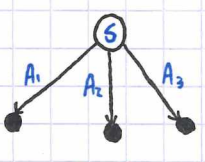
  b) $\pi_*(a|s) = \max_{a'} q_*(s,a')$

  (c) $\pi_*(a|s) = 1$ if $a = \arg\max_{a'} q_*(s,a')$ else 0



We can extract the optimal policy from the optimal state-value function, but to do so we need the one-step dynamics of the MDP. We need then a look ahead to see the values of the states where actions taken from current state takes us to.

$v_*(s)$



$q_*(s,a)$

We can get the optimal policy with much less work if we had the optimal action-value function. We simply select the action with the highest value in each state.

Prediction and Control

LESSON 01 / POLICY EVALUATION (PREDICTION)

* understand the distinction between policy evaluation & control.
* explain the setting of dynamic programming, also limitations.
* outline the iterative policy evaluation algorithm for estimating state values under a given policy.
* apply iterative policy evaluation to compute value functions.

LESSON 02 / POLICY ITERATION (CONTROL)

* understand the policy improvement theorem.
* use a value function for a policy to produce a better policy.
* outline the policy iteration alg. for finding optimal policy.
* understand "the dance of policy and value".
* apply policy iteration to compute optimal policies + value functions.

LESSON 03 / GENERALIZED POLICY ITERATION

* understand the framework of generalized policy iteration.
* outline value iteration, an important example of GPI.
* understand distinction of (a) synchronous programming.
* describe brute-force for searching an optimal policy.
* describe monte-carlo for searching a value function.
* understand the advantage of DP and "bootstrapping" over these alternatives for searching the optimal policy.

WEEK 04 - PAGES 73-88 OF RL-BOOK → LECTURES

Policy Evaluation vs Control

Evaluation is the task of determining the value function for a specific policy.

Control is the task of finding a policy to obtain as much reward as possible, finding a policy that maximizes the value function.

Control is the ultimate goal of RL, but the task of *Evaluation* is necessary 1st step. It's hard to improve our policy if we don't have a way to assess how good is it. Dynamic Programming techniques can be used to solve both these tasks, if we have access to the *dynamics function* p ($p(s',r|s,a)$).

Iterative Policy Evaluation

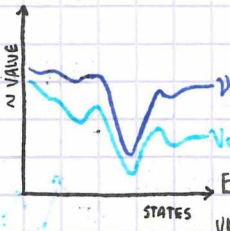
Dynamic Programming are obtained by turning Bellman EQNs into update rules.

THE BELLMAN EQUATION FOR STATE-VALUE GIVES US A RECURSIVE EXPRESSION

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

v_{k+1} ← v_k

INSTEAD OF AN EQUATION THAT HOLDS FOR THE TRUE VALUE FUNCTION, USE IT AS AN UPDATE RULE. THIS PROCEDURE ITERATIVELY APPLIED WILL REFINE THE ESTIMATE OF THE VALUE FUNCTION.



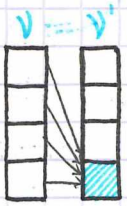
v_0: arbitrary initialization for approximate value function.

v_1: each iteration produces a better approximation using update rule.

Each iteration applies to $\forall s \in \mathcal{S}$, which is called a sweep. Applying update repeated ly leads to a better approx to the s-v fn v_π.

If the update leaves the approx unchanged, $v_{k+1}(s) = v_k(s)\ \forall s \in \mathcal{S}$, then $v_k = v_\pi$ and we have found the value function, this is because v_π is the unique solution to the Bellman Equation, the only way for v_k to be unchanged after update is if it already obeys Bellman EQN.

for any choice of v_0, $\lim_{k\to\infty} v_k = v_\pi$.



We store two arrays, each with an entry for every state. V stores the current approximate value function, the other V' the updated values. Compute new values from old one step at a time, without changing old. At the end of a full sweep compare both arrays, if different, write V' into V & iterate.

IN PLACE UPDATE IS POSSIBLE & WILL CONVERGE FASTER (BECAUSE SOME STATES WILL USE UPDATED STATE VALUES) AND CONSUME LESS MEMORY, BUT TWO ARRAY IS EASIER TO EXPLAIN.

$R = -1$ for moving

$\gamma = 1$ episodic

$\theta = 0.001$

25% 25% 25% 25% π random policy

v_0			
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

* TERMINAL STATES (NEVER UPDATE IT)

v_1			
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

1st sweep

v_2			
0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

2nd sweep

v_π			
0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

final sweep converge

In this case all initial values are set to 0, the dynamics are easy because it's deterministic (a at s takes to one s'), and the sweep order will be left to right and top to bottom. After a while this iterative policy evaluation should converge.

ITERATIVE POLICY EVALUATION FOR ESTIMATING $V \approx v_\pi$

$V \leftarrow \vec{0}$, $V' \leftarrow \vec{0}$

loop:

  $\Delta \leftarrow 0$

  for each $s \in S$:

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

    $\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$

  $V \leftarrow V'$

until $\Delta < \theta$ (a small positive number)

Output: $V \approx v_\pi$

Policy Improvement

GIVEN v_* WE CAN FIND THE OPTIMAL POLICY BY CHOOSING THE GREEDY ACTION, WHICH MAXIMIZES THE BELLMAN OPTIMALITY EQN IN EACH STATE.

$$\pi_*(s) = \operatorname*{argmax}_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$

Instead of optimal value function select a greedy action with respect to an arbitrary policy v_π.

New policy π'

* the new policy π' must be different than π. If greedification doesn't improve it, then the policy was already greedy respect to v_π.

$$\text{iff } \pi(s) = \operatorname*{argmax}_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \quad \forall s \in \mathcal{S}$$

v_π obeys Bellman Optimality EQN, π is already optimal policy.

* the new policy π' is a strict improvement on π unless already optimal.

POLICY IMPROVEMENT THEOREM

$$q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s)) \quad \forall s \in \mathcal{S} \text{ then } \pi' \text{ must be better}$$

but then follow π — $\pi'(s)$: an action but taken with diff. policy π'

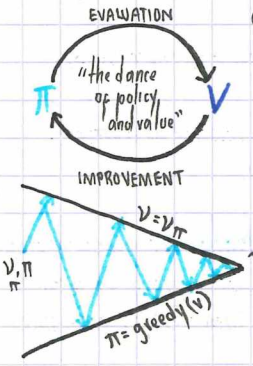
π' IS AT LEAST AS GOOD OR BETTER THAN π IF $\forall s \in \mathcal{S}$ THE VALUE OF ACTION SELECTED BY π' IS $\geq$ THE VALUE OF ACTION SELECTED BY π. STRICTLY BETTER IF VALUE IS $>$ IN AT LEAST ONE STATE.

the policy improvement theorem only guarantees a new policy is an improvement on the original, can't expect to find optimal policy easily.

Policy Iteration, Control/

Policy Iteration/

The value function computed for a given policy can be used to find a better policy. We can find the optimal policy by iteratively evaluating & improving a sequence of policies.



dimensionality ~ geometry of space of policies & value functions is more complicated, but the same intuition holds.

$$\pi'(s) \doteq \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$ PIT

CONSTRUCT A STRICTLY BETTER POLICY BY ACTING GREEDILY WITH RESPECT TO THE VALUE FUNCTION OF A GIVEN POLICY, UNLESS ALREADY OPTIMAL.

E = EVALUATE WITH ITERATIVE POLICY EVALUATION

$$\pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} v_{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E} \cdots \xrightarrow{I} \pi_*$$

- $\pi_1 \to v_{\pi_1}$: OBTAIN STATE VALUES FOR π_1.
- $v_{\pi_1} \to \pi_2$: I = IMPROVE WITH THE POLICY IMPROVEMENT THEOREM. GREEDIFY RESPECT TO π_1 WITH v_{π_1} TO OBTAIN A BETTER POLICY π_2
- $\pi_2 \to v_{\pi_2} \to \pi_3$: KEEP DANCING BETWEEN EVAL AND IMPROVE

Each policy generated this way is deterministic. There are a finite number of deterministic policies, so this iterative improvement must eventually reach an optimal policy and optimal v_{π_*}. Every policy will be greedy with respect to previous value function but not its own if it's not an optimal policy, but if it's an optimal policy then it's greedy with respect to its own value function, meaning should terminate the algorithm since policy won't improve anymore.

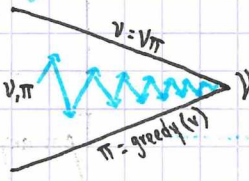
POLICY ITERATION (USING ITER. POL.-EVAL) FOR ESTIMATING $\pi \approx \pi_*$

1. INITIALIZATION:
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily $\forall s \in \mathcal{S}$
2. POLICY EVALUATION: *make V reflect the value of π*
   loop:
   $\Delta \leftarrow 0$
   foreach $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\left[r + \gamma V(s')\right]$ ($a = \pi(s)$)
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation).
3. POLICY IMPROVEMENT: *make a greedy π' respect to V.*
   policy-stable $\leftarrow$ true
   foreach $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$
   if old-action $\neq \pi(s)$, then policy-stable $\leftarrow$ false.
   If policy-stable = true, stop and return $V \approx v_*$, $\pi \approx \pi_*$, else go to 2.

Generalized Policy Iteration/

Regular *policy iteration* is a fairly rigid procedure. Alternates between evaluating the current policy and greedifying it. The *generalized policy iteration* framework allows much more freedom, while maintaining the optimality guarantees of converging to an optimal policy approx.

VALUE ITERATION



Policy Iteration Algorithm runs every step all the way to completion. We can imagine relaxing this, each evaluation step brings the estimate a little closer to the value of the current policy, and then a little more greedy but not totally, but should still converge to π_*, v_*.

In *Value Iteration*, we still sweep over all states and greedify with respect to value function, but we don't run policy evaluation to completion, we perform a single sweep over all states instead of sweeping, until some $\Delta < \theta$.

$$V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$$ UPDATE DOES NOT REFERENCE ANY SPECIFIC POLICY, HENCE THE NAME "VALUE ITERATION".

WE CAN RECOVER THE OPTIMAL POLICY FROM THIS VALUE FUNCTION WITH ARGMAX $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right] \ldots \pi \approx \pi_*$

SYNCRONOUS DP/

METHODS THAT PERFORM SYSTEMATIC SWEEPS OVER ALL STATES ARE CALLED SYNCHRONOUS. THIS CAN BE BAD IF THE STATE SPACE IS LARGE, BUT FOR SMALL VERY PRECISE & STRAIGHTFWD.

ASYNCHRONOUS DP/

UPDATES VALUES OF STATES IN ANY ORDER, NOT SYSTEMATIC, MAY UPDATE SOME STATES MORE, BUT TO GUARANTEE THE CONVERGENCE MUST UPDATE EVERY STATE SOME TIME, CAN UPDATE STATES NEAR THOSE THAT HAVE CHANGED VAL.

MONTE CARLO METHOD updates states randomly, needs a lot returns to average.
POLICY ITERATION polynomial time in $|S|$ and $|A|$, very fast ⇒ USES BOOTSTRAPPING/DP STATES
BRUTE FORCE SEARCH exponential time $|A|^{|S|}$ deterministic pols. ⇒ ALL COMBINATIONS OF ACTIONS

$q(11,\downarrow) = [-1 + (1 * \overset{\text{TERMINAL STATE VALUE}}{0})] = -1$

$p(s',r|s,a)$ will be 1 because state transitions are deterministic.

$q(7,\downarrow) = [-1 + (1 * \overset{\text{STATE 11 STATE VALUE}}{-14})] = -15$

Remember Bellman EQN:

$$v(15) = X = \frac{1}{4}(-1 + \overbrace{X) + \frac{1}{4}(-1+X) + \frac{1}{4}(-1+X}^{\text{bounces}}) + \frac{1}{4}(-1 \underset{\text{STATE 13}}{-20})$$

$X = \frac{3}{4}(X-1) + \frac{1}{4}(-21)$,

$4X = 3X - 24$, $X = -24$

Practice Quiz/

* the value of any state under an optimal policy is greater than or equal to the value of a state under a non-optimal policy.
* It is false that if a policy is greedy with respect to the value function of the uniform random policy then it's guaranteed to be optimal policy.
* It is true that if a policy is greedy with respect to its own value function then it is the optimal policy.
* If π' is greedy respect to v_π, then $v_{\pi'} \geq v_\pi$.
* If $v_{\pi'} = v_\pi$, then π and π' may not be equal, they could have different tie-breaking mechanisms.
* VALUE ITERATION and POLICY ITERATION are special cases of G.P.I.
* It is false that all GENERALIZED POLICY ITER. algorithms are synchronous.
* As described, VALUE ITERATION & POLICY ITERATION are synchronous.
* ASYNCHRONOUS methods scale better than SYNCHRONOUS methods.
* DP algos are planning methods because they use a model to impr. policy.