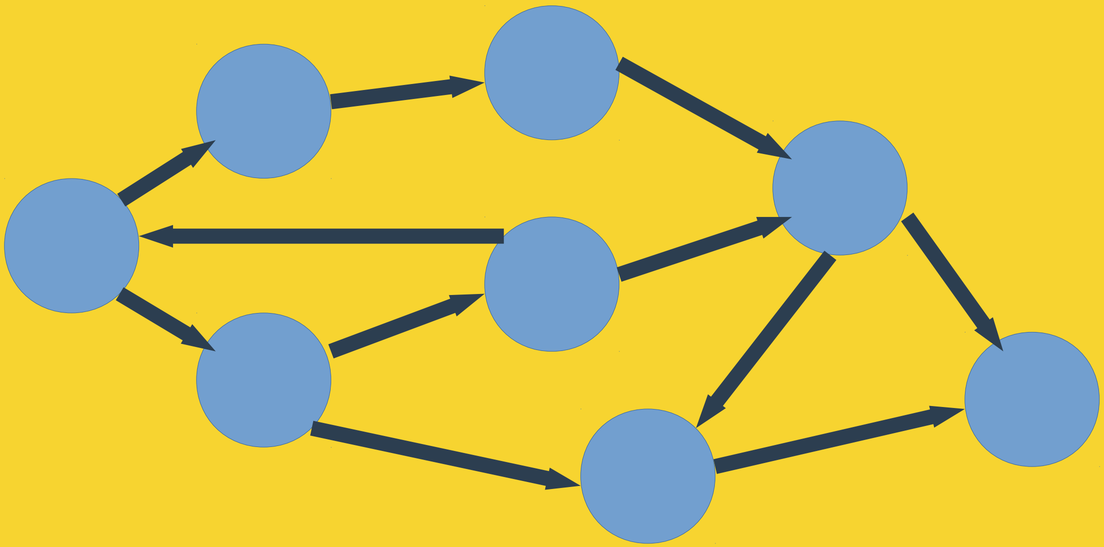
PyData
Charlotte
@pydatacharlotte



Planning ahead: using Markov decision processes to optimize your life

Eric Cotner
Data Scientist, American Tire Distributors
Email: 2.71828cotner@gmail.com

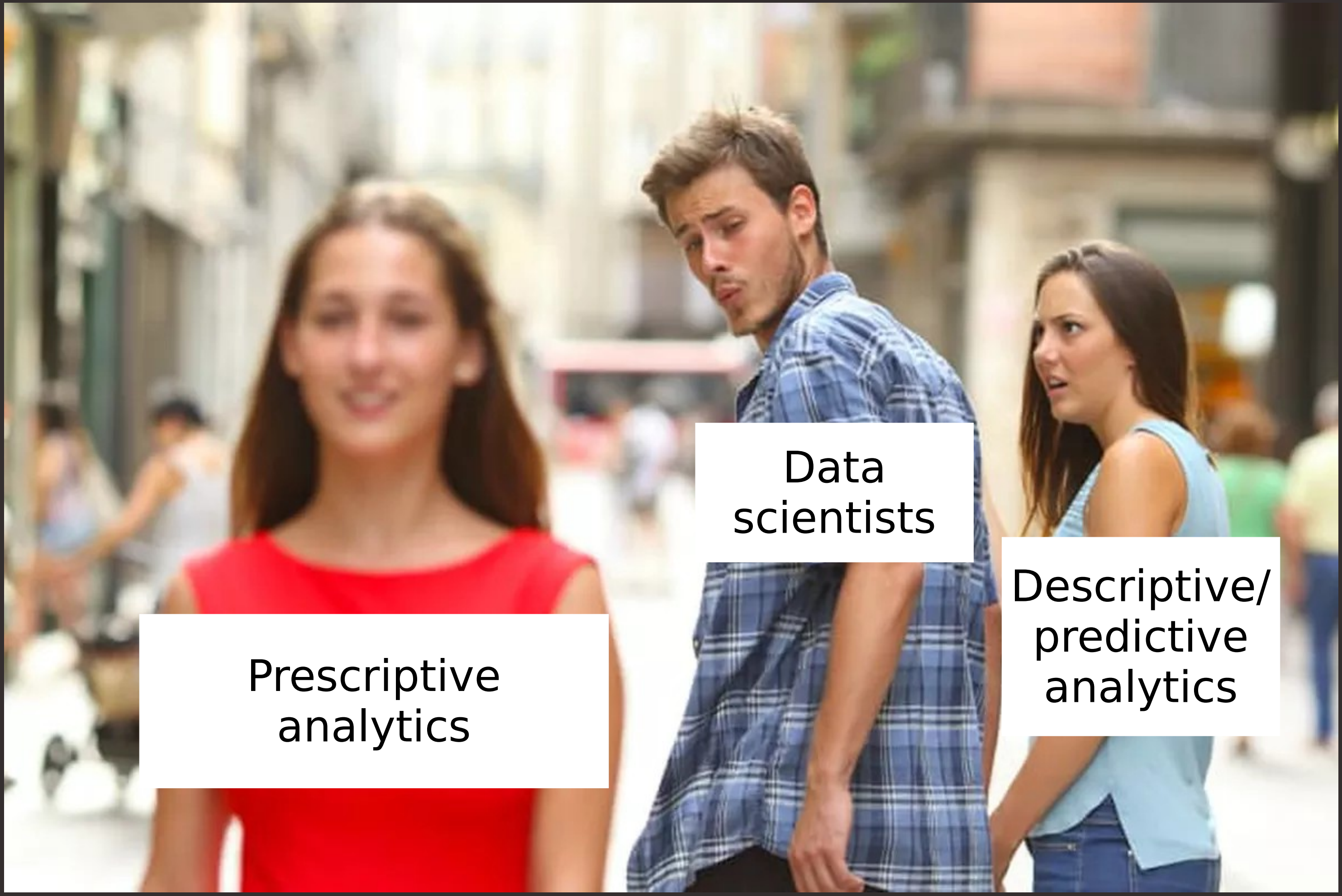
$ whoami

- Former theoretical physicist – studied dark matter and black holes in grad school
- Now I do data science at American Tire Distributors
- Focus on supply chain and logistics problems
  - Optimizing delivery routes/inventory
- Enjoy hiking, astronomy, and scuba diving in my free time

Outline

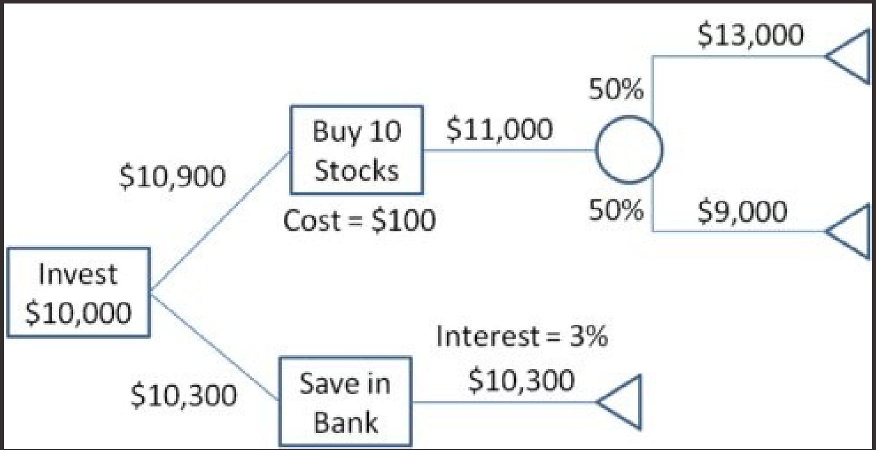
- **What is a Markov decision process (MDP)?**
- **How to quantify risk/reward?**
- **Under the hood**
  - States, actions, transitions, rewards, policy, value functions
- **An example MDP**
- **Solving MDP's using `pymdptoolbox` – a demo**
- **Working with more complex problems**
- **Reinforcement learning**

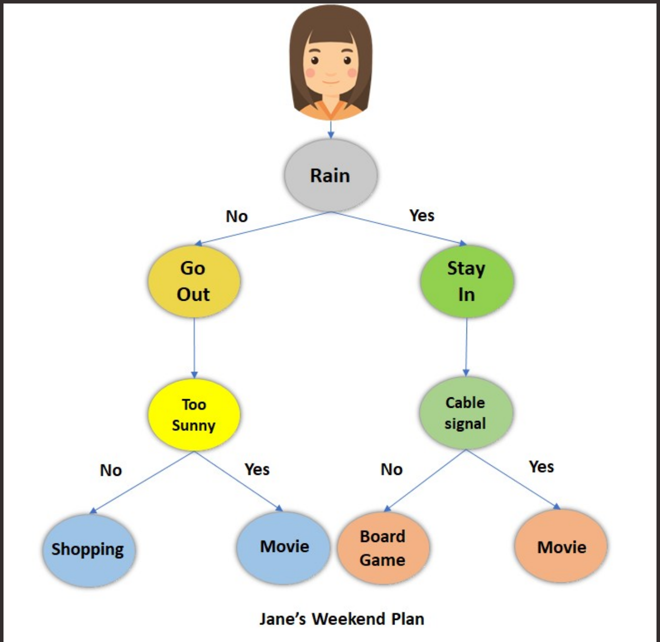
What is a Markov decision process?



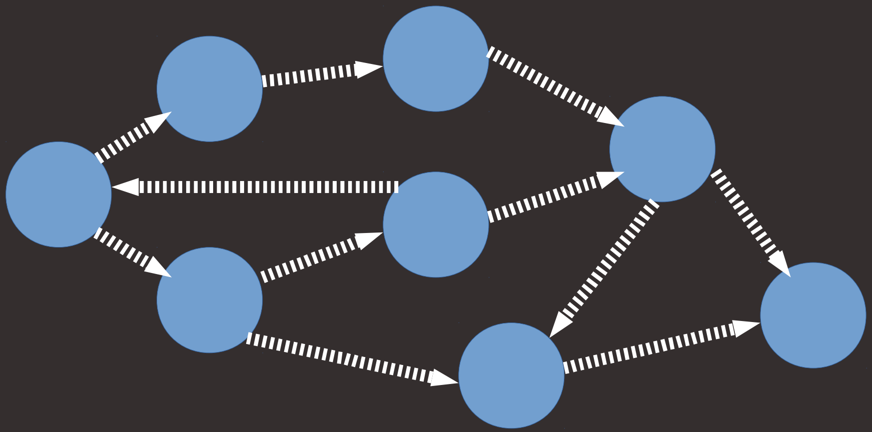
What is a Markov decision process?

Decision tree:



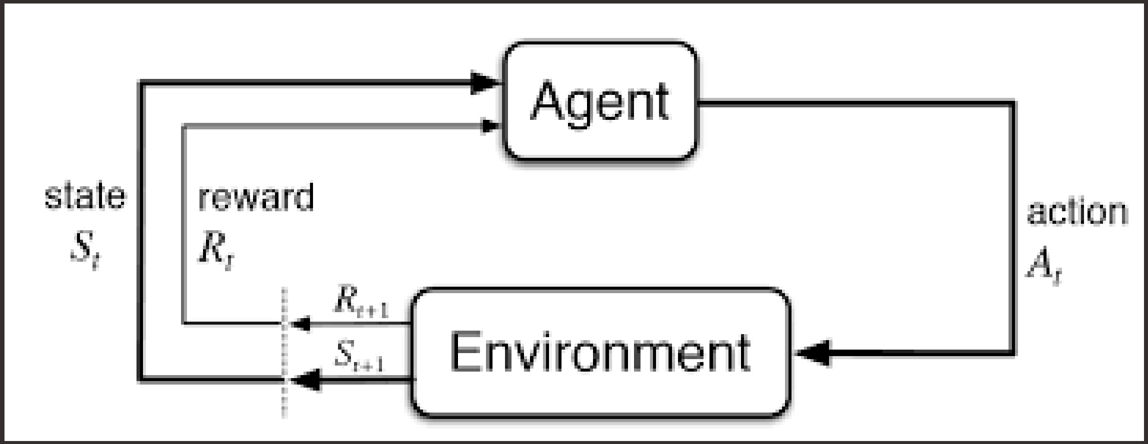


MDP:



What is a Markov decision process?

- **MDP's are a mathematical formalism for solving sequential decision problems**
  - Satisfies the *Markov* property: all states and rewards do *not* depend on past history
- **Defined in terms of *states*, *actions*, *rewards*, and *transitions*, which are fixed by the nature of the problem/environment**
  - Capital letters denote random variables, lower case denote actual samples
- **Things to solve for are the *policy* (tells you which action to take) and *value function(s)* (how much reward you expect to get from taking an action)**



How to quantify risk/reward/value?

- **Reward: some thing of value gained (money, game points, reddit karma, personal satisfaction, etc.) by achieving certain goals**
- **Risk = (reward) x (probability of reward)**

  Example: roll a 6-side die; {1,4}=-$2.15, {2,3,5,6}=+$1.34; risks associated with these two outcomes are (-$2.15)x(2/6)= -$0.72 and (+$1.34)x(4/6)=+$0.89

- **Value: sum of all risks; i.e. the expected/average reward**
  - Previous example: value = (-$0.72) + (+$0.89) = +$0.17

How to quantify risk/reward/value?

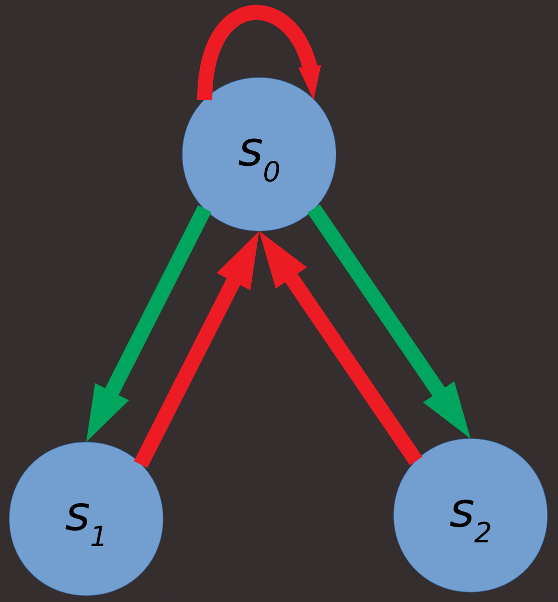
- **Previous example's usefulness is limited**
  - What if you need to plan multiple steps ahead?
  - What if rewards change with time or are random?
  - What if there is more than one available course of action?
  - What if actions don't always result in intended consequences?
- **Value is (discounted) sum of expected future rewards:**

$$v_t = \mathbb{E}\left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}\right]$$

- **How do you determine the optimal sequence of actions to take? → Use MDP's!**

States and actions

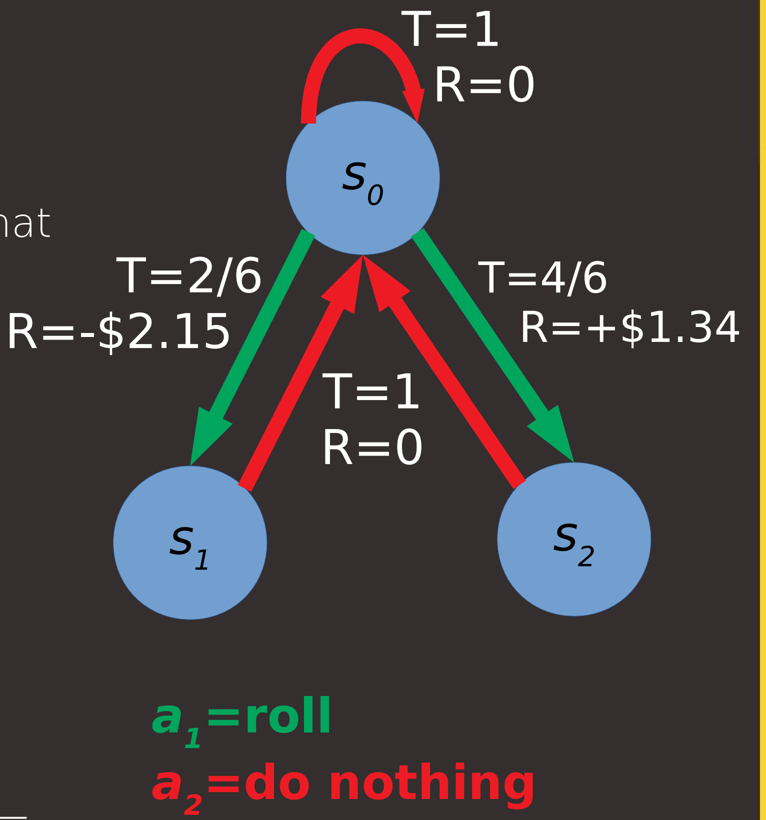
- ***States*** **represent the current state of the environment**
  - Usually denoted by symbols *s*, S_t (sometimes *x* in control theory); set of all states: $\mathcal{S} = \{s_1, s_2, \ldots\}$
  - Previous die example: s_1=1, s_2=2, …, s_6=6 (or s_1={1,4}, s_2={2,3,5,6})
- ***Actions*** **represent the decisions you can make which will cause you to transition from state to state**
  - Usually denoted by symbols *a*, A_t (sometimes *u* in control theory); set of all actions: $\mathcal{A} = \{a_1, a_2, \ldots\}$
  - Previous die example: a_1=roll, a_2=do nothing



a_1=roll
a_2=do nothing

State transitions and rewards

- **Only some states lead to other states; sometimes taking one action can lead to multiple potential states**
- **Model this using the transition matrix/function *T(s';s,a)***
  - *T(s';s,a)* = the probability to transition to state *s'* given that you take action *a* from state *s*
  - 6-sided die example: $T(s_1;s_0,a_1)$=2/6, $T(s_2;s_0,a_1)$=4/6, $T(s_0;*,a_2)=1$, otherwise *T*=0
- **Sometimes rewards can be stochastic too; model this using reward matrix/function R(r;s,a)**
  - *R(r;s,a)* = probability of getting reward *r* when taking action *a* from state *s*
  - If reward is deterministic, reward matrix *R(s,a)* = reward received from taking action *a* from state *s*



The policy

- **How do you decide what to do in a given state?**
- **The *policy* function π determines which action to take**
  - Could be deterministic: $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$
  - Could be stochastic: $\pi(a|s) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$
  - (deterministic is special case of stochastic where all probability mass is located at one action)
- **The optimal policy is denoted by π*, which is the policy that maximizes the sum of future expected rewards (value function)**

Value functions

- **Two kinds of value functions:**
  - The state-value function $V_\pi(s)$: the future reward you expect when starting from state s and following policy π
  - The action-value function $Q_\pi(s,a)$: the future reward you expect when starting from state s and taking action a, and following policy π thereafter

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a)$$

$$Q_\pi(s,a) = \mathbb{E}_{A\sim\pi}\left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} \middle| S_t = s_t, A_t = a\right]$$

How do you actually solve these equations?

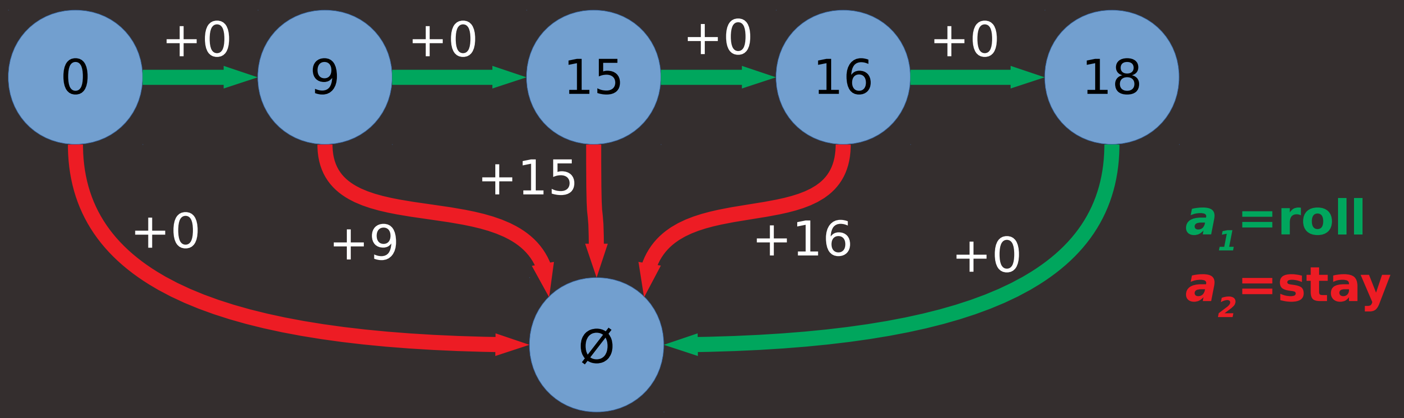
- **Many different algorithms exist depending on size of state/action spaces, length of episodes, structure of reward/transition matrices…**
- **All of them use some form of the Bellman equation:**

$$Q_\pi(s,a) \leftarrow \mathbb{E}_\pi\left[R(s,a) + \gamma Q(s',a')\right]$$

- **Consider one of the simplest: value iteration:**
  - Randomly initialize *Q(s,a)*
  - Select policy by maximizing value function: $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$
  - Use Bellman equation to update estimate of *Q*
  - Repeat 2-3 until convergence

An example MDP

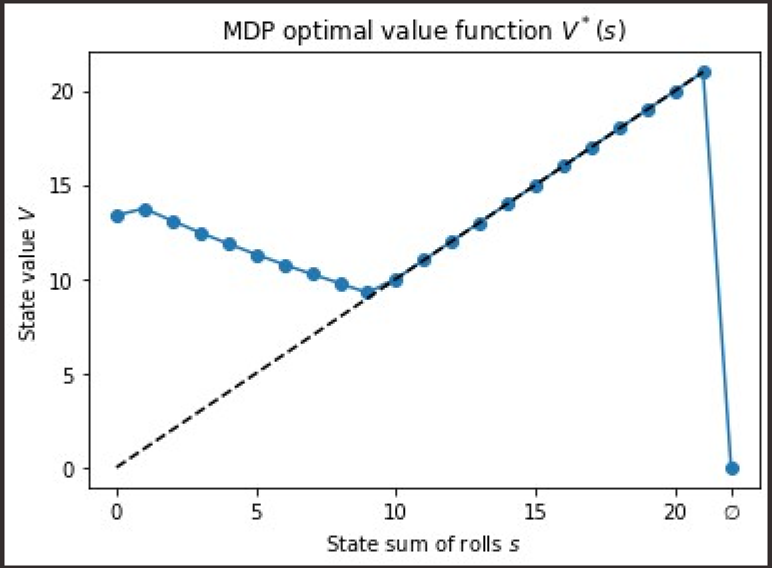
- Let's look at a relatively simple example to solve – roll or stay?
- You have a 20-sided die, and you get to roll repeatedly until the sum of your rolls either gets as close as possible to 21 or you bust
- Your score is the numerical value of the sum of your rolls; if you bust, you get zero.
- What is the optimal strategy?

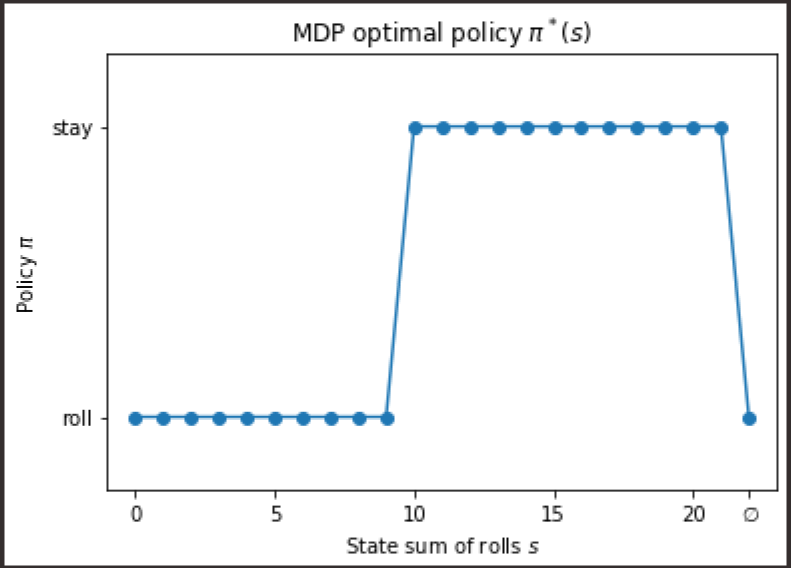


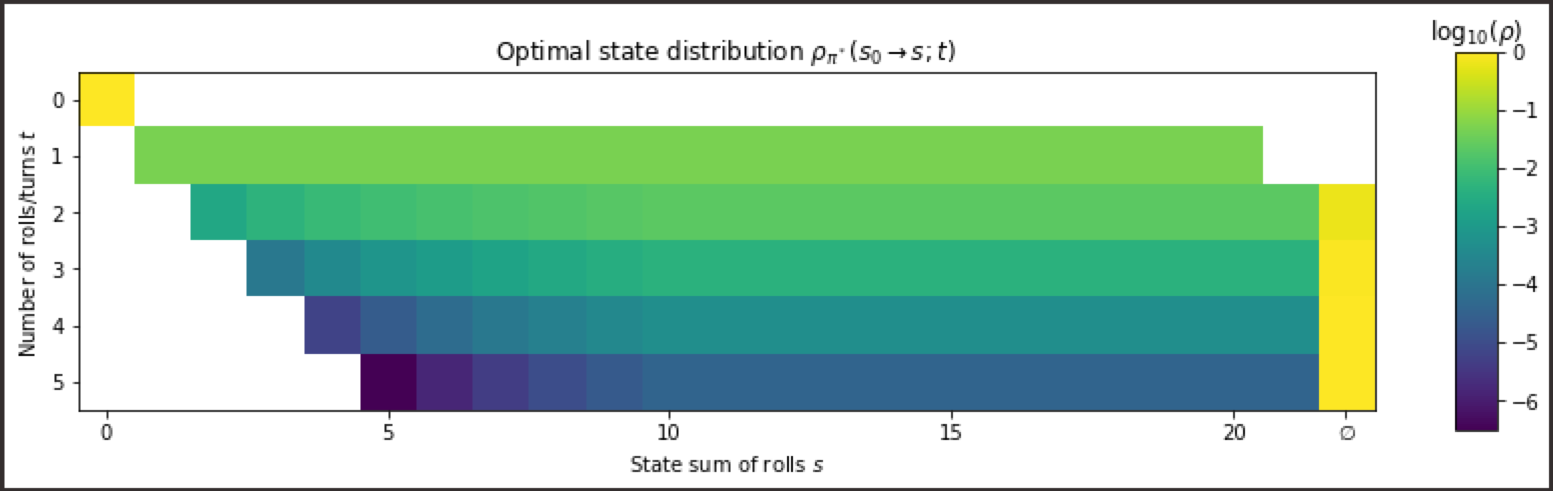
Using `pymdptoolbox`

- **Let's solve the example from the previous slide in python.**
- **We will use *pymdptoolbox*, a toolkit for solving discrete-time MDP's**
- **For more details, see the github repo or the API documentation**
- **Install with \`*pip install pymdptoolbox*\`**
- **Check out my demo jupyter notebook**

Example MDP solution







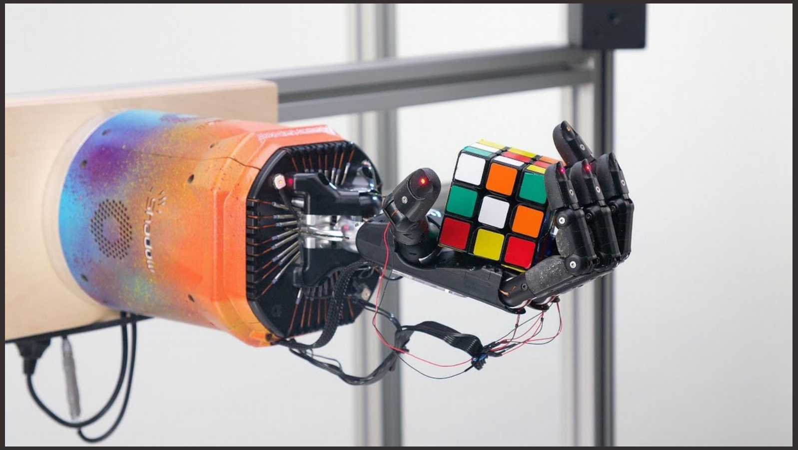
Working with more complex problems

- **Previous example was very simple compared to most realistic problems**
- **For example, maybe you're trying to plan purchase orders for your supply chain**
  - If you have 10 warehouses, each with 1,000 different products which can have stocking levels between 0-100, and you want to plan your orders for the next couple months… that's a 1M-dimensional state space and 1M-dimensional action space – would need a transition matrix with 10^18 elements…
- **For very large systems, you may have to resort to approximate solutions**
  - Bucket similar products together
  - Find better state/action representations – use dimensional reduction
  - Use function approximation to represent value/policy functions
- **Some problems have continuous state/action spaces and can't be formulated with matrices at all**
  - For example: self-driving car - state is position+orientation+velocity vectors, actions are amount of gas/brake/wheel angle; all probably continuous variables
  - Would need to use a completely different approach from the *value iteration* method we used

How do you handle these more complex problems?

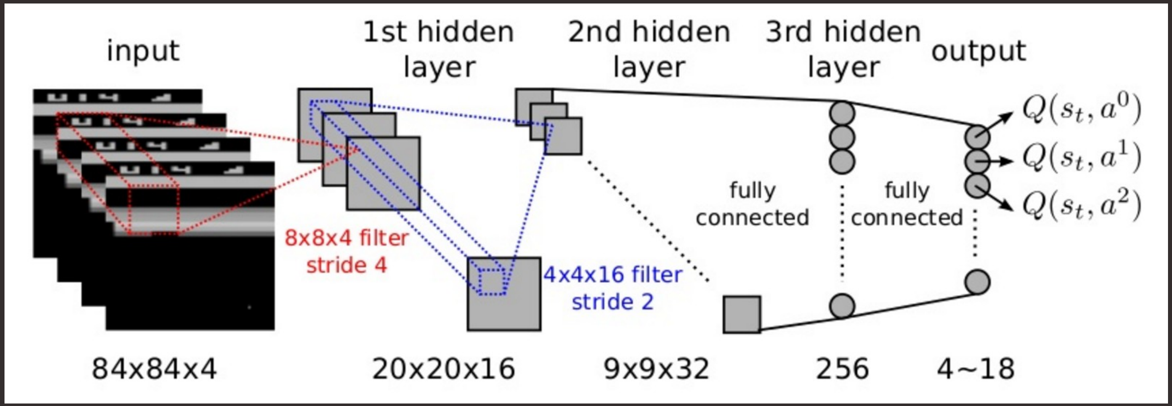
Reinforcement learning

- **MDP + ML = RL**
- **Type of machine learning that learns how to act optimally in an environment to maximize rewards**
- **MDP’s are the underlying mathematical framework for RL – if you know MDP’s you can easily learn RL**

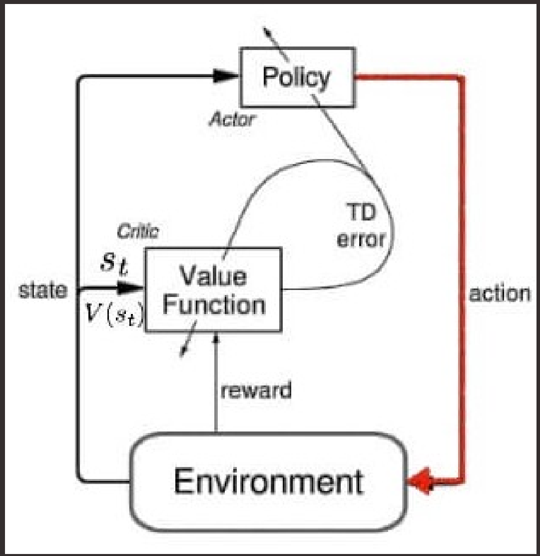


Reinforcement learning

- **Very advanced algorithms that try to solve MDP's in environments with enormous state spaces by only sampling states and actions encountered in simulation**
- **Can use neural networks to approximate complex value and/or policy functions**



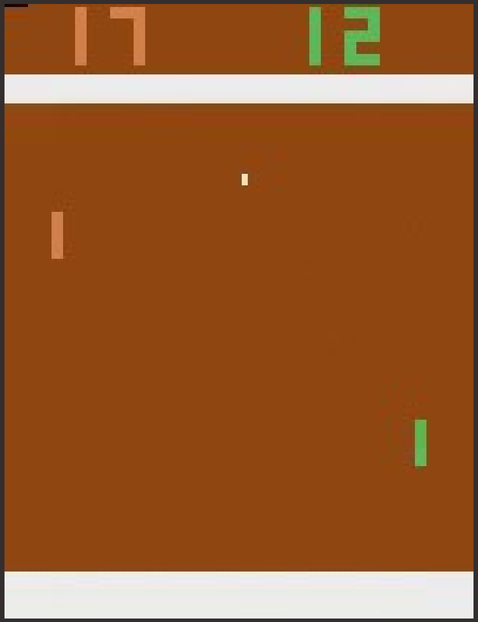
Deep Q-Network (DQN)



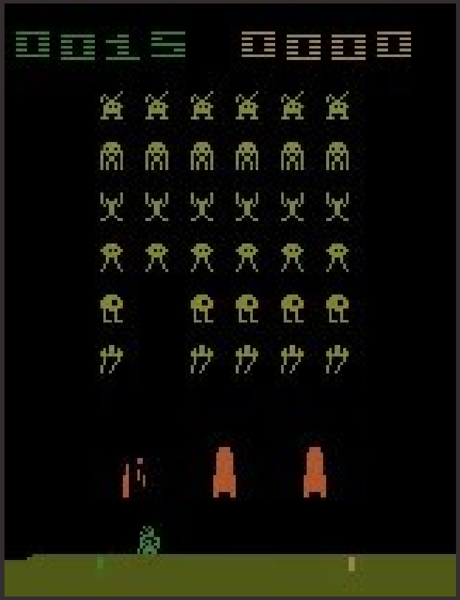
Actor-critic

Reinforcement learning

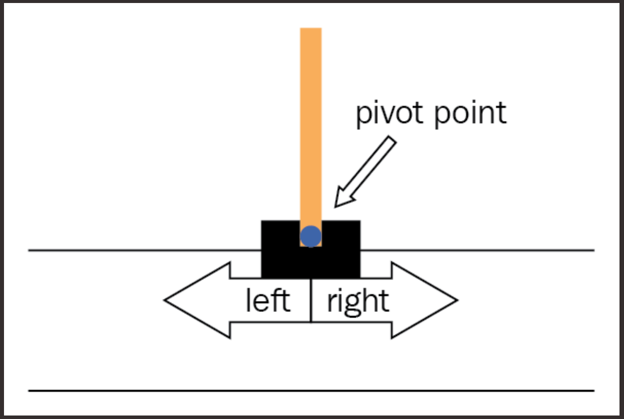
- You can get started with reinforcement learning by using OpenAI’s gym
- Has out-of-the box environments you can experiment with
- Can make your own custom environments



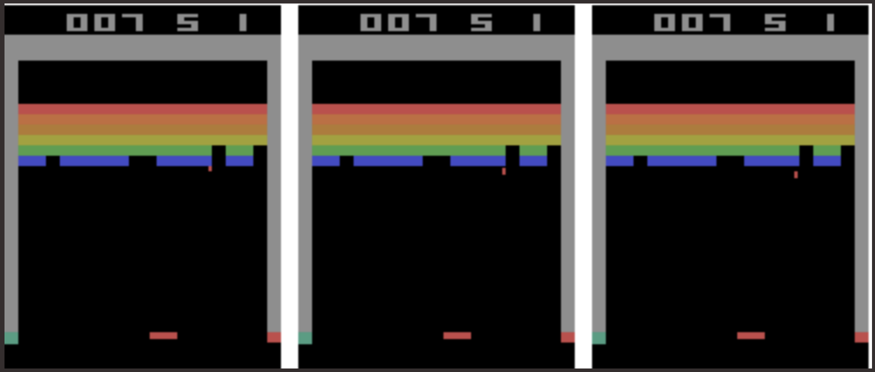
Pong



Space Invaders



Cart-pole



Breakout

Reference material

- ***`pymdptoolbox`*; Steven Cordwell**
  - *https://pymdptoolbox.readthedocs.io/en/latest/api/mdptoolbox.html*
- ***Introduction to Reinforcement Learning*; Sutton, Barto:**
  - http://incompleteideas.net/sutton/book/the-book-2nd.html
- ***`gym`*; OpenAI**
  - *https://gym.openai.com/*
- ***Spinning up in Deep RL*; OpenAI**
  - https://spinningup.openai.com/en/latest/
- **r/reinforcementlearning; Reddit**
  - https://www.reddit.com/r/reinforcementlearning/

Thank you!