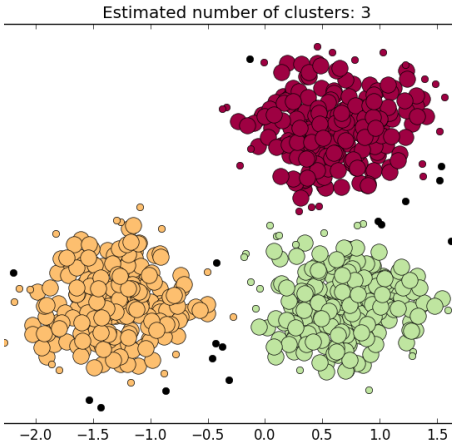
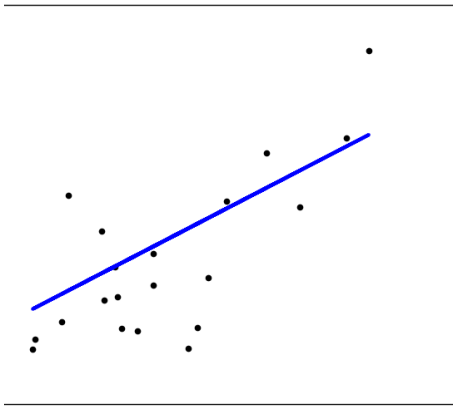
Reinforcement Learning for Data Scientists

Brian Farris
Capital One Labs

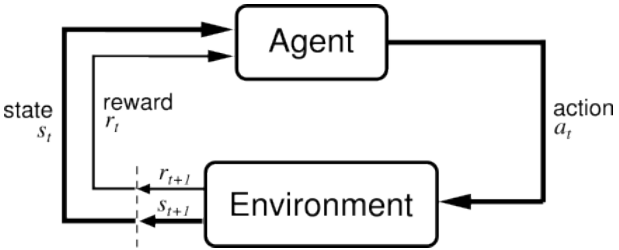
Types of Learning



Unsupervised



Supervised

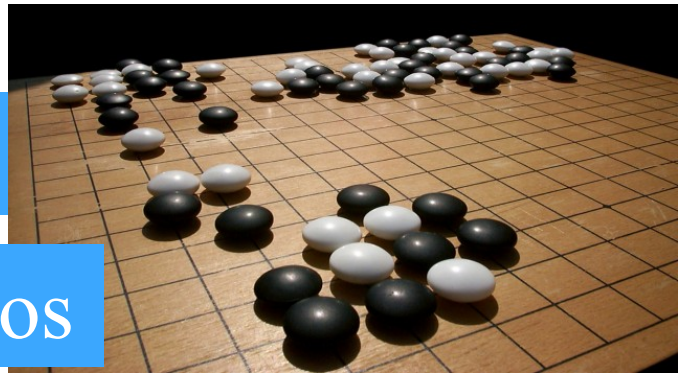


Reinforcement

> “A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour.”

- Sutton and Barto, 2012

“In fact, it played so well that it was almost scary ... One of the greatest virtuosos of the middle game had just been upstaged in black and white clarity.”



- David Ormerod, 2016

Reinforcement Learning vs Supervised Learning

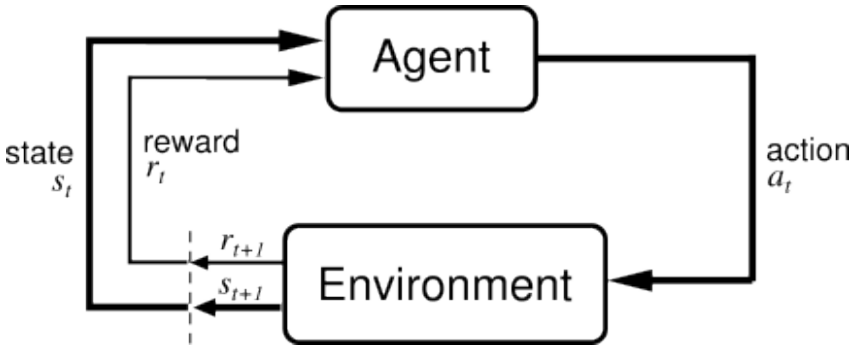
Supervised Learning

- "learning from examples provided by a knowledgeable external supervisor"
- For any state that the agent may be in, the supervisor can supply enough relevant examples of the outcomes which result from similar states so that we may make an accurate prediction.

Reinforcement Learning

- No supervisor exists
- Agent must learn from experience as it explores the range of possible states
- Continuously update policy in response to new information.

Background



RL algorithm overview

- *Agent* interacts dynamically with its *environment*, moves from one *state* to another.
- Based on the *action*s taken by the *agent*, *rewards* are given.
- Guidelines for which *action* to take in each state is called a *policy.*
- Try to efficiently find an optimal *policy* in which *rewards* are maximized.

Examples

agent	environment	actions	rewards	policy
Robot arm	Set of arm positions	bend elbow, close hand, extend arm, etc.	reward when door successfully opened	most efficient set of movements to open door
Board game player	Set of game configs	legal moves	winning the game	optimal strategy
Mouse	maze	running, turning	cheese	most direct path to cheese
Credit card company	set of all customers in default	set of collections actions	cost for each attempt, reward for successful collection	optimal strategy for debt collections
Marketing team	sets of potential customers and ads that can be shown	showing an ad to a potential customer	cost of placing ad, value of customer's business	optimal ad placement strategy
Call center	status of each customer in queue	connecting customers to representatives	customer satisfaction	optimal queueing strategy
Website Designer	Set of possible layout options	changing layout	Increased click-through rate	ideal layout

Applications

Business operations

- Inventory management: how much to purchase of inventory, spare parts
- Resource allocation: e.g. in call center, who to service first
- Routing problems: e.g. for management of shipping fleet, which trucks/truckers to assign to which cargo

Schulman and Abeel, Lecture Notes, CS294

Applications

Finance

- Investment decisions
- Portfolio design
- Option/asset pricing

Schulman and Abeel, Lecture Notes, CS294

Applications

E-commerce / media

- What content to present to users (using click-through / visit time as reward)
- What ads to present to users (avoiding ad fatigue)
- Medicine
- What tests to perform, what treatments to provide

Schulman and Abeel, Lecture Notes, CS294

Example: Article Recommendation Engine - Yahoo! Labs

"A Contextual-Bandit Approach to Personalized News Article Recommendation" (Li et al, 2012)

- Challenge: Adapt advertisements, news articles, etc. to individual users based on content and user information
- Solution: Model personalized recommendation of news articles as a contextual bandit problem
- Tested algorithm on Yahoo! Front Page Today Module dataset containing over 33 million events
- Found 12.5% click lift compared to a standard context-free bandit algorithm
- Argue that any bandit algorithm can be reliably evaluated offline using previously recorded random traffic

Example: Google Analytics Content Experiments

- Provides a framework for testing up to 10 versions of a single page.
- Uses Multi-Armed Bandits under the hood
- Overview of methods provided:
  https://support.google.com/analytics/answer/2844870?hl=en&ref_topic=1745207
- Written by Steven L Scott, Senior Economic Analyst at Google and former Director of Statistical Analysis at Capital One

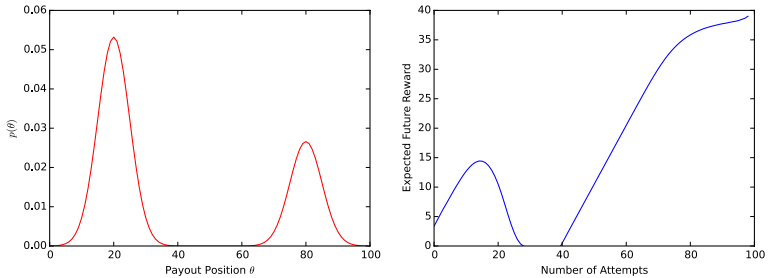
Example: Reinforcement Learning for NY State Tax Collection

"Optimizing Debt Collections Using Constrained Reinforcement Learning"

(Abe et al, 2010)

- Challenge: Optimize the debt collections process for NY state taxes.
- Outperforms standard approach of combining data modeling and constrained optimization
- State expects savings of about 100 million dollars in the next three years

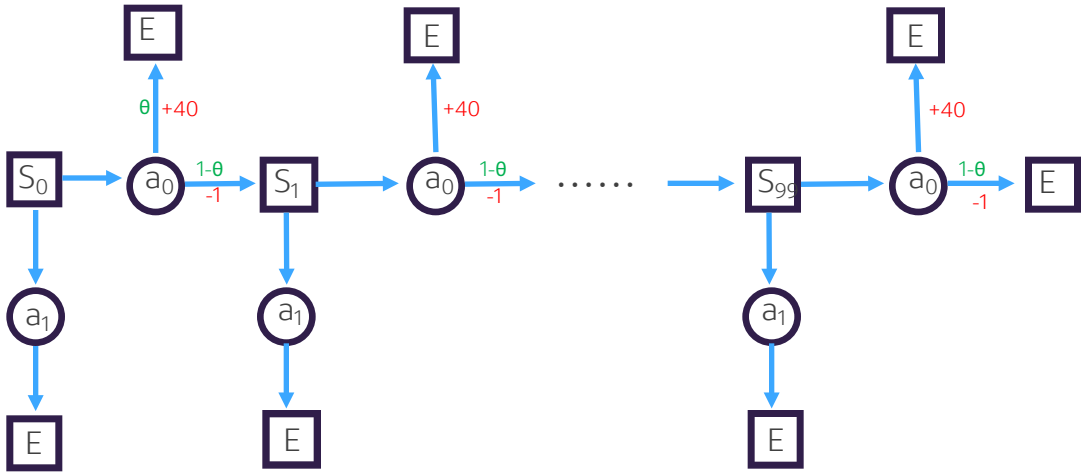
Recast as debt collection problem



Debt collection policy

- Each individual owes $40, will pay if asked θ times by the *agent*.
- In each *state*, the *agent* chooses between 2 *actions*, trying to collect, or giving up.
- Each collection attempt has 2 possible outcomes:
    1. it receives a negative *reward* of $1 and moves into the next *state*.
    2. it receives a positive *reward* of $40 and the game terminates
- The set of all possible *states* is the *environment*.
- We don't know the distribution of θ a priori.
- Can we learn what the best collections *policy* is?
- Can we learn what the expected *reward* is for any given *state*?

Markov Decision Process

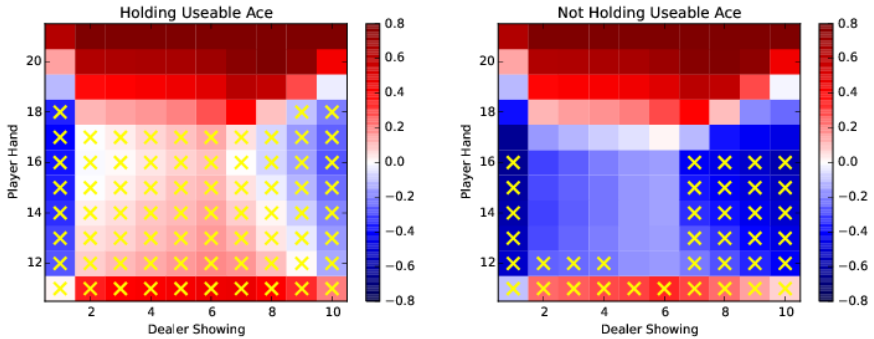


Defined by:

- Set of all *states* and actions available to *agent*
- Probability of transitions between *states*
- For each transition, there is an expected *reward* for entering new *state*
- Transition probability depends only on current *state* and *action* (Markov Property)

Environment defined by specifying $\mathbf{Pr}\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\}$, for all r, s', S_t and A_t

Example: Blackjack



- What is the optimal policy when playing Blackjack?
- What is the expected future reward in each state?

Bellman Equations

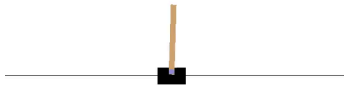
$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\
&= \sum_{s'} p(s'|s,a)\Big[r(s,a,s') + \gamma \max_{a'} q_*(s',a')\Big].
\end{aligned}
$$

- System of algebraic equations
- Can be solved directly (uncommon) or iteratively (e.g. "policy iteration" or "value iteration")

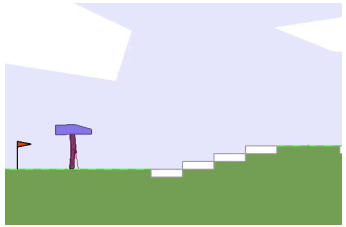
1. Dynamic programming Approach
    1. Assume complete knowledge of environment (incl. transition probabilities)
    2. Solve eqns exactly
2. Monte Carlo Approach
    - Sample sequences of states, actions, and rewards from interaction with environment
    - For each state-action pair, compute average subsequent rewards over many "episodes"
3. Temporal Difference Approach (e.g. Q-learning)
    - Combine MC and DP ideas.
    - Update estimates based in part on other learned estimates (bootstrapping)
4. Policy Gradient Approach (deep RL)
    - Model policy with neural net
    - Update parameters of policy with gradient descent

Recommendation: Check out gym.openai.com

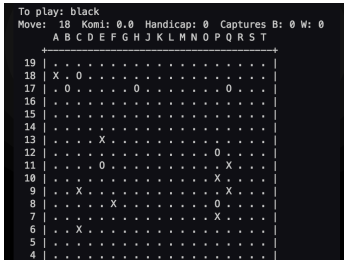
"A toolkit for developing and comparing reinforcement learning algorithms."



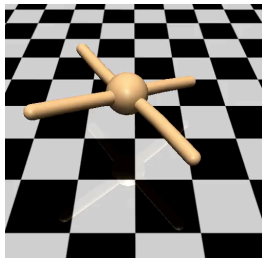
cartpole



Bipedal walker



go



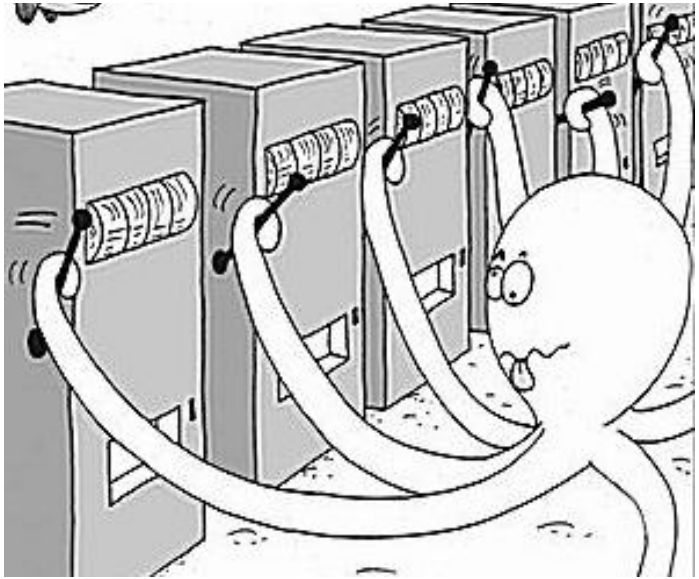
MuJoCo

OpenAI

"OpenAI is a non-profit artificial intelligence research company. Our goal is to advance digital intelligence in the way that is most likely to benefit humanity as a whole, unconstrained by a need to generate financial return."

Sponsors:
- Sam Altman
- Greg Brockman
- Reid Hoffman
- Jessica Livingston
- Elon Musk
- Peter Thiel

Multi-Armed Bandit: Single-state RL



- Given N different arms to choose from
- Each with an unknown reward
- How do we:
    - Explore and learn the values of each arm?
    - Exploit our current knowledge to maximize profit?
- Model as Markov Decision Process with:
    - 1 state
    - N actions
- Ideal approach for testing with real time feedback

MAB: A Framework for Testing

- As Data Scientists we try to:
  - *exploit* our knowledge of the business environment (using models)
  - *explore* the business environment through testing (to make the models accurate)
- Often, there is a trade-off between *exploration* and *exploitation*.
- Testing can be very expensive, and we need to balance the resources that we allocate to each.
- Multi-Armed Bandits provide a principled way of striking this balance in an optimal, automated way.

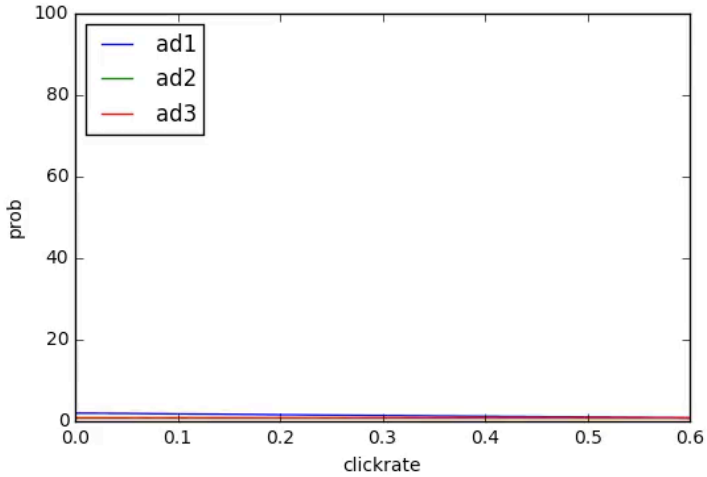
Example: Thompson Sampling

- Model probability distribution for each arm. For example, choose beta distribution:

  $P(\theta \mid y) = \text{Beta}(y + 1, n - y + 1)$

- Sample a number from each distribution
- Choose arm corresponding to winner
- Arms with higher click-rate will naturally be chosen more often
- Pull the arm, measure the outcome, and update the corresponding posterior.

Example: Thompson Sampling



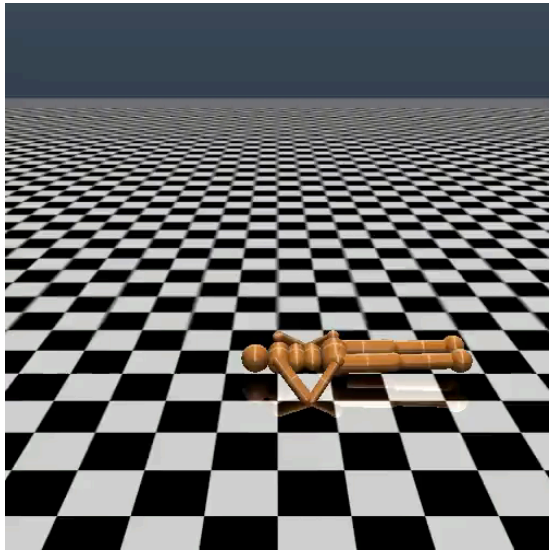
Posterior distributions for clickrate

Deep RL: Policy Gradients

- Let the agent's policy be expressed by a parametrized function $\pi (a | s, \theta)$
- Try to find θ which maximizes:
  $E [R | \pi (a | s, \theta)]$
- Compute gradients $\nabla_\theta E [R | \pi (a | s, \theta)]$
- Find θ which minimizes loss

- Let $\pi (a | s, \theta)$ be a deep neural net
- θ are the weights in the neural net

- See John Schulman and Pieter Abbeel's Deep RL course page for more info:
  http://rll.berkeley.edu/deeprlcourse/



humanoid

Summary

Gazelle – all grown up

- RL is a framework for tackling many AI tasks.
- Useful for any business challenge in which a tradeoff between *exploration* and *exploitation must be managed*
- The Multi-Armed Bandit problem is a special case of RL with a single state. Particularly relevant for marketing applications
- Deep RL uses neural nets to model policies in RL problems. Recent algorithms have been successful on a wide range of problems using relatively generic algorithms