Interlude

1

OpenAI – GPT2

- Language models – unigrams, bigrams, Markov models, ELMO
- GPT2 – transformer based NN
  - 1.5 Billion parameters
  - Trained on 40 GB Internet text (no supervision)

2

Generate Synthetic Text

Human Prompt:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT continues… (best of 10 tries)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. …

3

Generate Synthetic Text

Human Prompt:

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown

GPT continues… (best of 10 tries)

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

"The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation," said Tom Hicks, the U.S. Energy Secretary, in a statement. "Our top priority is to secure the theft and ensure it doesn't happen again."

The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

4

Zero Shot Learning on other Tasks

Winigrad Schema Challenge

The trophy would not fit in the brown suitcase because it was too **big** (*small*).

What was too **big** (*small*)?

Answer 0: the trophy

Answer 1: the suitcase

The town councilors refused to give the demonstrators a permit because they **feared** (*advocated*) violence.

Who **feared**(*advocated*) violence?

Answer 0: the town councilors

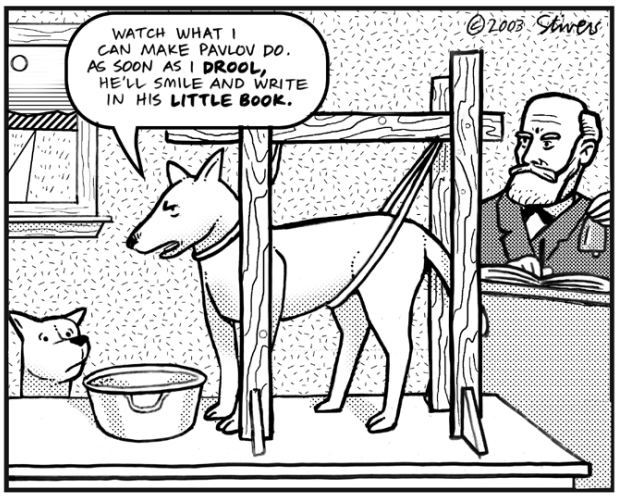
Answer 1: the demonstrators

GPT	70.7% accuracy
Previous record:	63.7%
Human:	92%+

5

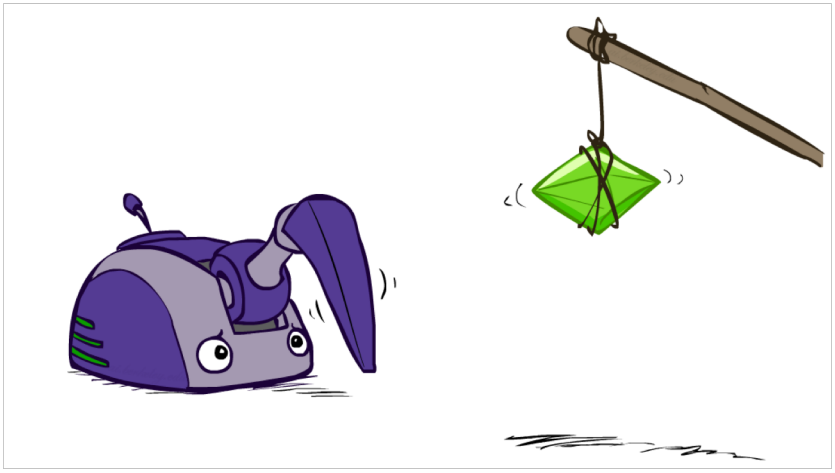
CSE P 573: Artificial Intelligence

Reinforcement Learning

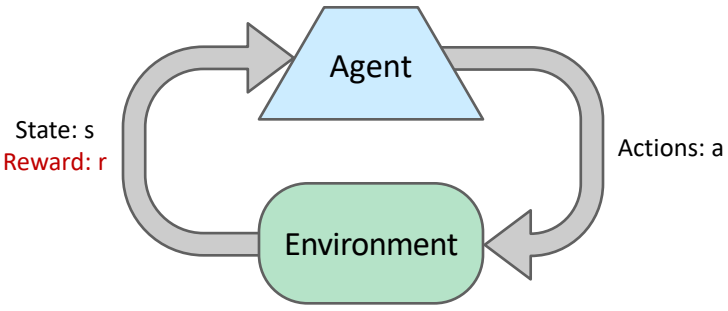


[Many slides taken from Dan Klein and Pieter Abbeel / CS188 Intro to AI at UC Berkeley – materials available at http://ai.berkeley.edu.]

Reinforcement Learning



Reinforcement Learning



- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

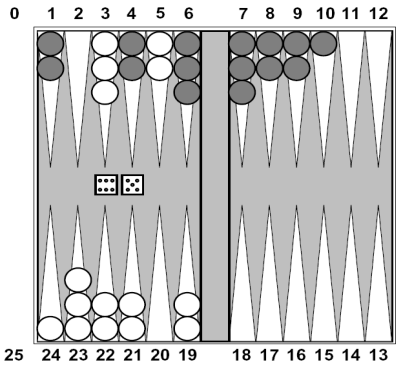
Example: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistication debated
- Example: foraging
  - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
  - Bees have a direct neural connection from nectar intake measurement to motor planning area

Example: Backgammon

- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to V(s) using a neural network
- Combined with depth 3 search, one of the top 3 players in the world

- You could imagine training Pacman this way…
- … but it's tricky! (It's also PS 4)



Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – initial]

Example: Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – finished]

Example: Sidewinding

[Andrew Ng]

[Video: SNAKE – climbStep+sidewinding]

"Few driving tasks are as intimidating as parallel parking….

https://www.youtube.com/watch?v=pB_iFY2jIdI

15

Parallel Parking

"Few driving tasks are as intimidating as parallel parking….

https://www.youtube.com/watch?v=pB_iFY2jIdI

16

Other Applications

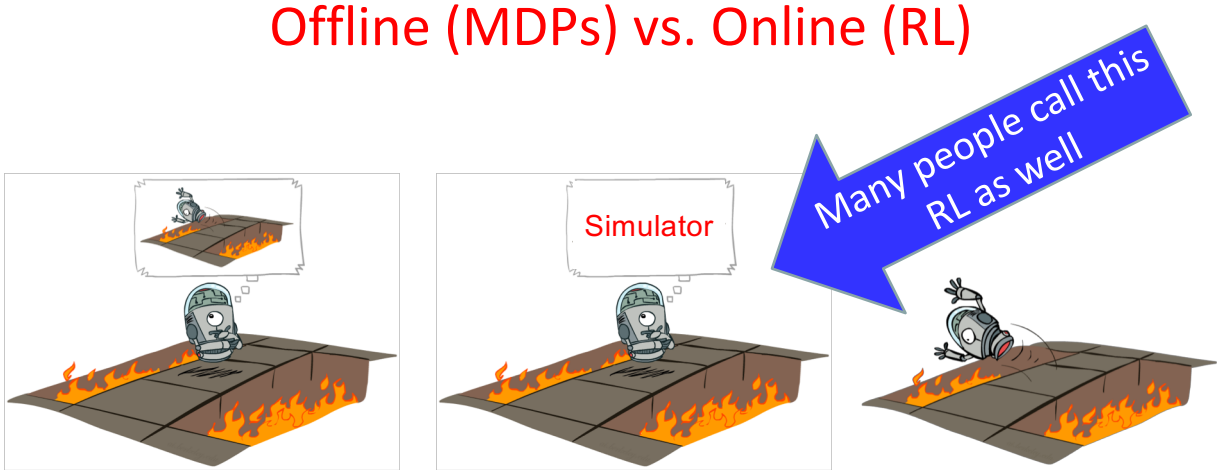
- Robotic control
  - helicopter maneuvering, autonomous vehicles
  - Mars rover - path planning, oversubscription planning
  - elevator planning
- Game playing - backgammon, tetris, checkers, chess, go
- Computational Finance, Sequential Auctions
- Assisting elderly in simple tasks
- Spoken dialog management
- Communication Networks – switching, routing, flow control
- War planning, evacuation planning, forest-fire treatment planning

Reinforcement Learning

- Still assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s') & discount γ
- Still looking for a policy $\pi(s)$

- New twist: don't know T or R
  - I.e. we don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

Offline (MDPs) vs. Online (RL)



Offline Solution (Planning) | Monte Carlo Planning | Online Learning (RL)

Diff: 1) dying ok; 2) (re)set button

Demo

- Stanford Helicopter
  https://www.youtube.com/watch?v=Idn10JBsA3Q

20

Four Key Ideas for RL

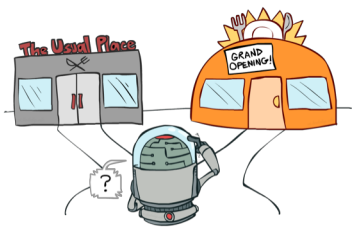
- Credit-Assignment Problem
  - What was the real cause of reward?
- Exploration-exploitation tradeoff
- Model-based *vs* model-free learning
  - What function is being learned?
- Approximating the Value Function
  - Smaller → easier to learn & better generalization

Credit Assignment Problem

22

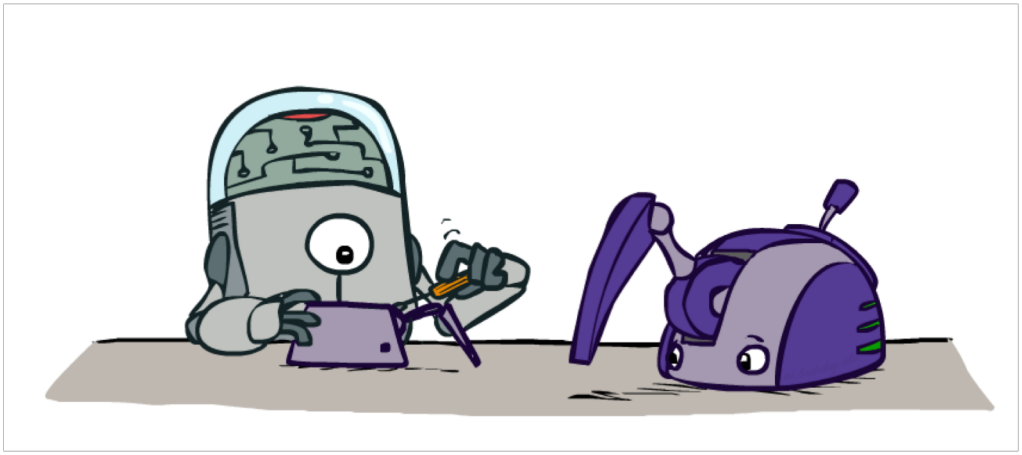
Exploration-Exploitation tradeoff

- You have visited part of the state space and found a reward of 100
  - is this the best you can hope for???
- **Exploitation**: should I stick with what I know and find a good policy w.r.t. this knowledge?
  - at risk of missing out on a better reward somewhere
- **Exploration**: should I look for states w/ more reward?
  - at risk of wasting time & getting some negative reward



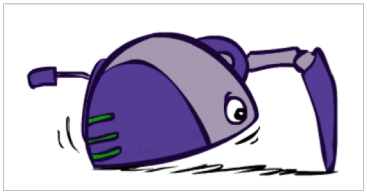
23

Model-Based Learning



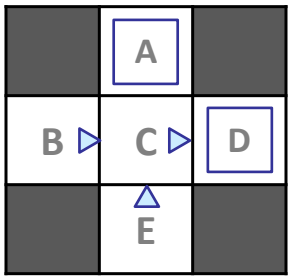
Model-Based Learning

- Model-Based Idea:
  - Learn an approximate model based on experiences
  - Solve for values as if the learned model were correct
- Step 1: Learn empirical MDP model
  - Explore (e.g., move randomly)
  - Count outcomes s' for each s, a and estimate $\hat{T}(s, a, s')$
  - Normalize to give an estimate of
  - Discover each $\hat{R}(s, a, s')$ when we experience (s, a, s')
- Step 2: Solve the learned MDP
  - For example, use value iteration, as before



Example: Model-Based Learning

Random π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$\hat{T}(s, a, s')$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
...

$\hat{R}(s, a, s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
...

Convergence

- If policy explores "enough" – doesn't starve any state
- Then T & R converge

- So, VI, PI, Lao* *etc.* will find optimal policy
  - Using Bellman Equations

- When can agent start exploiting??
  - (We'll answer this question later)

27

Two main reinforcement learning approaches

- Model-based approaches:
  - explore environment & learn model, T=P(**s**'|**s**,**a**) and R(**s**,**a**), (almost) everywhere
  - use model to plan policy, MDP-style
  - approach leads to strongest theoretical results
  - often works well when state-space is manageable
- Model-free approach:
  - ***don't*** learn a model of T&R; instead, learn ***Q-function*** (or policy) directly
  - weaker theoretical results
  - often works better when state space is large

28

Two main reinforcement learning approaches

- Model-based approaches:

  Learn T + R

  $|S|^2|A| + |S||A|$ parameters (40,400)

- Model-free approach:

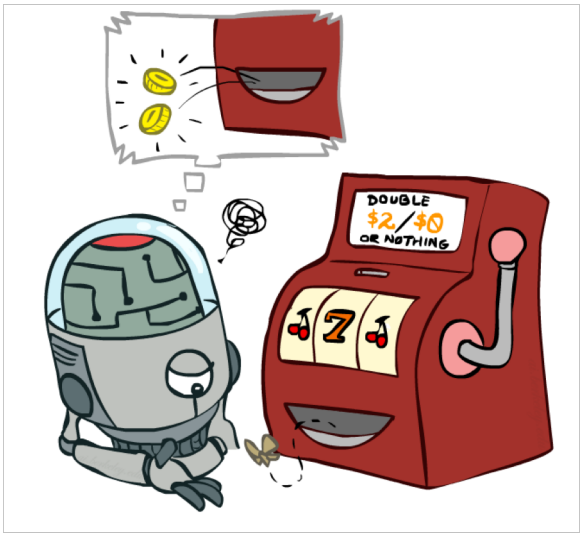
  Learn Q

  $|S||A|$ parameters (400)

Suppose 100 states, 4 actions

29

Model-Free Learning



Nothing is Free in Life!

- What exactly is Free???
  - No model of T
  - No model of R
  - (Instead, just model Q)

31

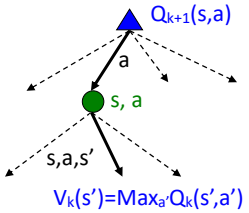
Reminder: Q-Value Iteration

- **Forall s, a**
  - **Initialize $Q_0(s, a) = 0$** *no time steps left means an expected reward of zero*
- **K = 0**
- **Repeat** *do Bellman backups*

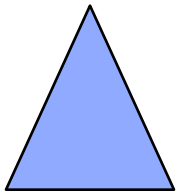
  For every (s,a) pair:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

  K += 1
- **Until convergence** *I.e., Q values*



We know this….



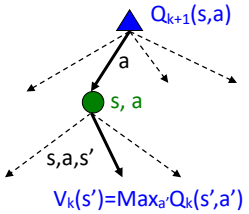
Puzzle: Q-Learning

- **Forall s, a**
  - **Initialize $Q_0(s, a) = 0$** *no time steps left means an expected reward of zero*
- **K = 0**
- **Repeat** *do Bellman backups*

  For every (s,a) pair:

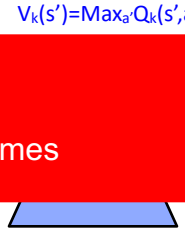
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

  K += 1
- **Until convergence**



Q: How can we compute without R, T ?!?

A: Compute averages using sampled outcomes



Simple Example: Expected Age

Goal: Compute expected age of CSE students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \quad = 0.35 \times 20 +$$

Note: never know **P(age=22)**

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Why does this work? Because eventually you learn the right model.

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work? Because samples appear with the right frequencies.

Anytime Model-Free Expected Age

Goal: Compute expected age of CSE students

Let A=0
Loop for i = 1 to ∞
  a_i ← ask "what is your age?"
  A ← (1-α)*A + α*a_i

Without P(A), instead collect samples [a_1, a_2, … a_N]

Let A=0
Loop for i = 1 to ∞
  a_i ← ask "what is your age?"
  A ← (i-1)/i * A + (1/i) * a_i

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Anytime Model-Free Expected Age

Goal: Compute expected age of CSE students

Let A=0
Loop for i = 1 to ∞
  a_i ← ask "what is your age?"
  A ← (1-α)*A + α*a_i

Without P(A), instead collect samples [a_1, a_2, … a_N]

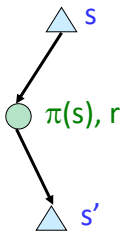
Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Let A=0
Loop for i = 1 to ∞
  a_i ← ask "what is your age?"
  A ← (i-1)/i * A + (1/i) * a_i

Sampling Q-Values

- Big idea: learn from every experience!
  - Follow exploration policy a ← π(s)
  - Update Q(s,a) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often



- Update towards running average:

Get a sample of Q(s,a): *sample* = r + γ $Max_{a'}$ Q(s', a')

Update to Q(s,a): Q(s,a) ← (1-***α***)Q(s,a) + (***α***)*sample*

Same update: Q(s,a) ← Q(s,a) + ***α***(*sample* – Q(s,a))

Rearranging: Q(s,a) ← Q(s,a) + ***α***(difference)
Where difference = (r + γ $Max_{a'}$ Q(s', a')) - Q(s,a)

} Equivalently

Q Learning

- **Forall s, a**
  - **Initialize Q(s, a) = 0**
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

  difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

  Q(s,a) ← Q(s,a) + α(difference)

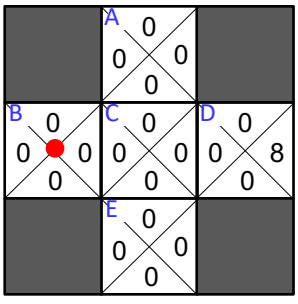
Trial

Note parallel to RTDP
- Both have trials
- But no T, R

Example

Assume: γ = 1, α = 1/2

Observed Transition: B, east, C, -2



In state B. What should you do?

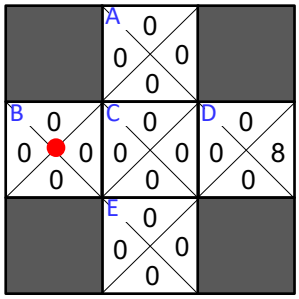
Suppose (for now) we follow a random exploration policy

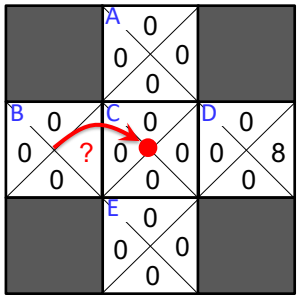
 → "Go east"

Example

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transition: B, east, C, -2





difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

-2 -2 0 0

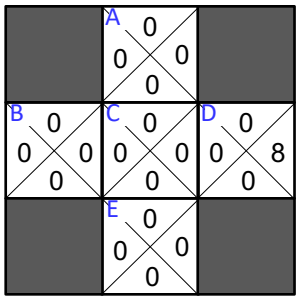
Q(s,a) ← Q(s,a) + α(difference)

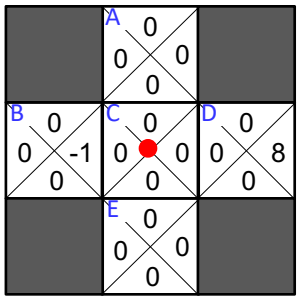
-1 0 ½ (-2)

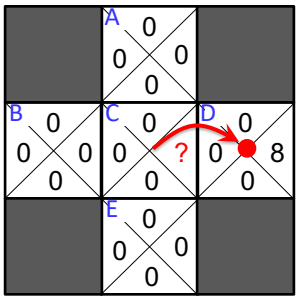
Example

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transition: B, east, C, -2 C, east, D, -2







difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

6 -2 8 0

Q(s,a) ← Q(s,a) + α(difference)

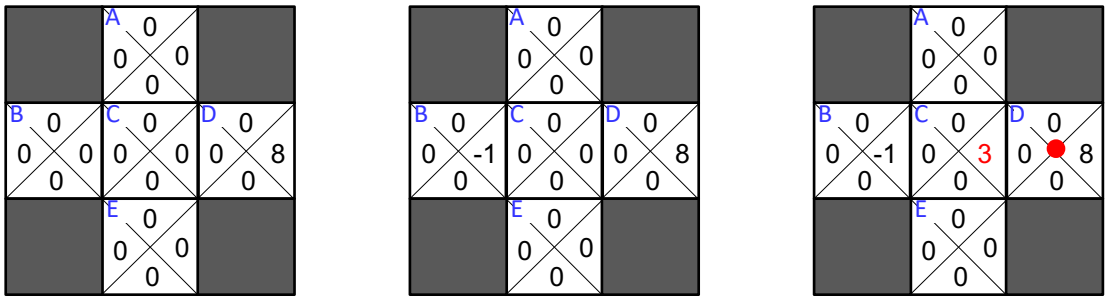
3 0 ½ (6)

Example

Assume: $\gamma = 1$, $\alpha = 1/2$

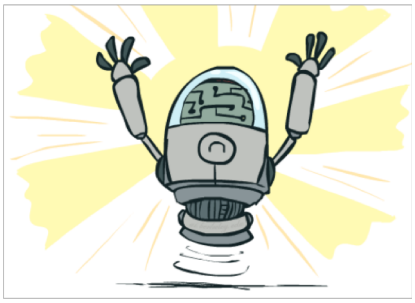
Observed Transition: B, east, C, -2

C, east, D, -2



Q-Learning Properties

- Q-learning converges to optimal Q function (and hence ***learns*** optimal policy)
  - even if you're acting suboptimally!
  - This is called off-policy learning
- Caveats:
  - *You have to explore enough*
  - *You have to eventually shrink the learning rate, α*
  - *… but not decrease it too quickly*
- And… if you want to ***act*** optimally
  - You have to switch from explore to exploit



[Demo: Q-learning – auto – cliff grid (L11D1)]

Video of Demo Q-Learning Auto Cliff Grid

Q Learning

- **Forall s, a**
  - **Initialize Q(s, a) = 0**
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

$$\text{difference} \leftarrow [r + \gamma \text{Max}_{a'} Q(s', a')] - Q(s,a)$$

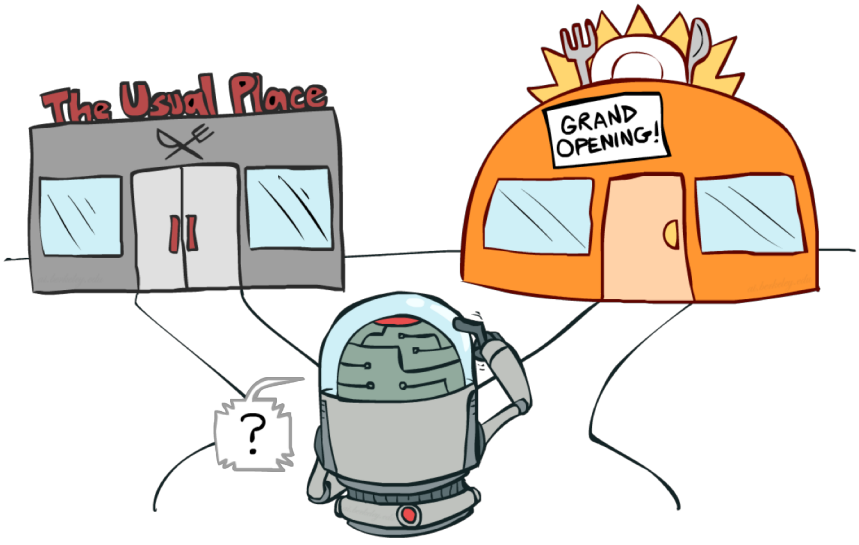
$$Q(s,a) \leftarrow Q(s,a) + \alpha(\text{difference})$$

Demos

- Inverted Pendulum https://www.youtube.com/watch?v=Lt-KLtkDlh8
- Stanford Helicopter https://www.youtube.com/watch?v=Idn10JBsA3Q

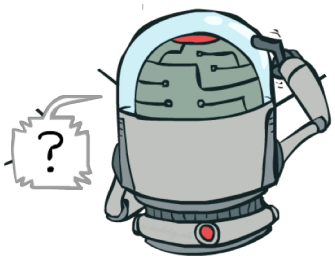
47

Exploration vs. Exploitation



Questions

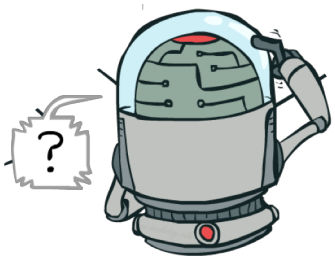
- How to explore?



- When to exploit?
- How to even think about this tradeoff?

Questions

- How to explore?
  - Random Exploration
  - Uniform exploration
  - Epsilon Greedy
    - With (small) probability ε, act randomly
    - With (large) probability 1-ε, act on ***current policy***
  - Exploration Functions (such as UCB)
  - Thompson Sampling
- When to exploit?
- How to even think about this tradeoff?

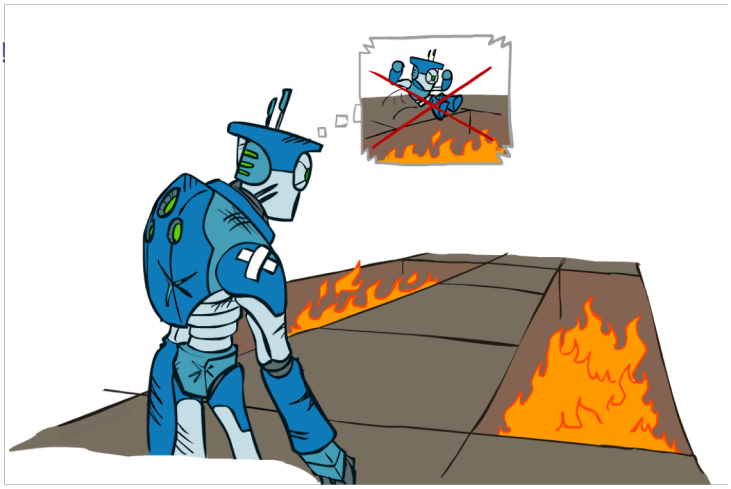


Video of Demo Crawler Bot

More demos at: http://inst.eecs.berkeley.edu/~ee128/fa11/videos.html

Regret

- Even if you learn the optimal policy, you still make mistakes along the way!
- Regret is a measure of your total ***mistake cost***: the difference between your (expected) rewards, including youthful sub-optimality, and optimal (expected) rewards
- Minimizing regret goes beyond learning to be optimal – it requires ***optimally learning to be optimal***

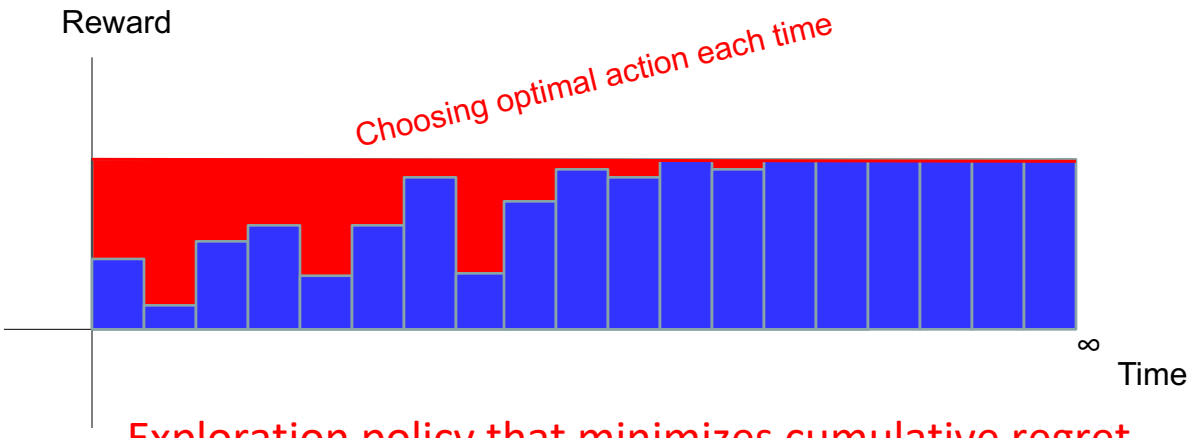


Two KINDS of Regret

- **Cumulative Regret:**
  - Goal: achieve near optimal cumulative lifetime reward
    (in expectation)
- **Simple Regret:**
  - Goal: quickly identify policy with high reward
    (in expectation)

54

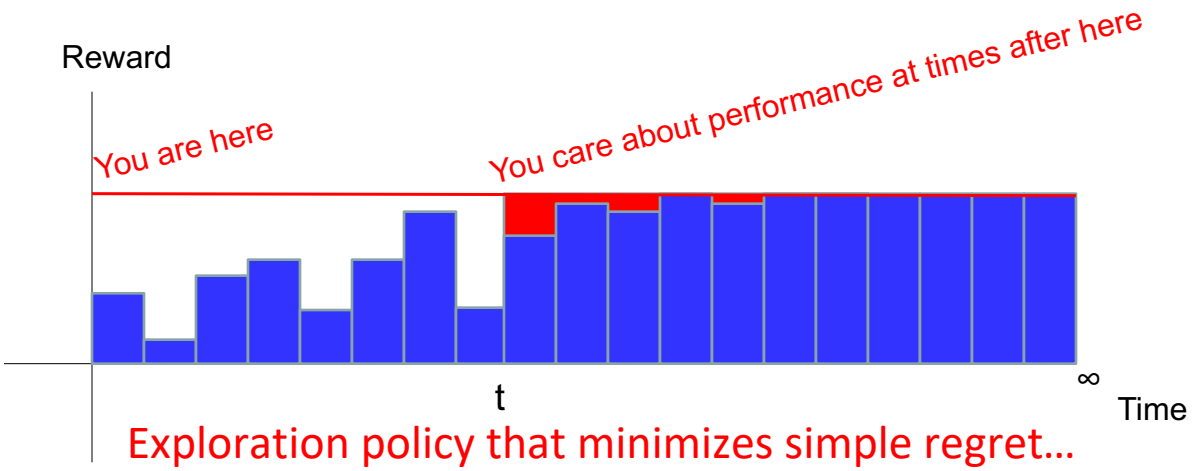
Regret



Exploration policy that minimizes cumulative regret
Minimizes red area

55

Regret

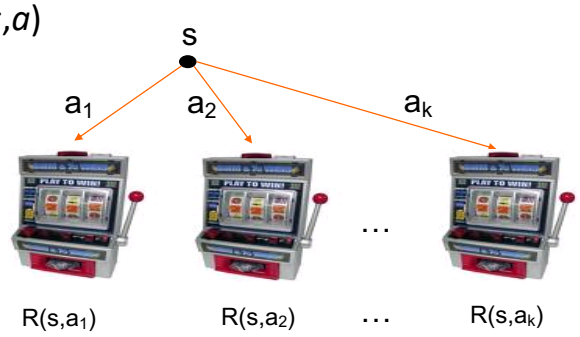


Exploration policy that minimizes simple regret…
Given a time, t, in the ***future***, explore in order to ***minimize red area after t***

56

RL on Single State MDP

- Suppose MDP has a single state and k actions
  - Can sample rewards of actions using call to simulator
  - Sampling action *a* is like pulling slot machine arm with random payoff function *R*(*s*,*a*)



57

Multi-Armed Bandit Problem

Slide adapted from **Alan Fern** (OSU)

Multi-Armed Bandits

- Bandit algorithms are not just useful as components for RL & Monte-Carlo planning
- Pure bandit problems arise in many applications
- Applicable whenever:
  - set of independent options with unknown utilities
  - cost for sampling options or a limit on total samples
  - Want to find the best option or maximize utility of samples

Slide adapted from **Alan Fern** (OSU)

Multi-Armed Bandits: Example 1

- **Clinical Trials**
  - Arms = possible treatments
  - Arm Pulls = application of treatment to individual
  - Rewards = outcome of treatment
  - Objective = maximize cumulative reward = maximize benefit to trial population (or find best treatment quickly)

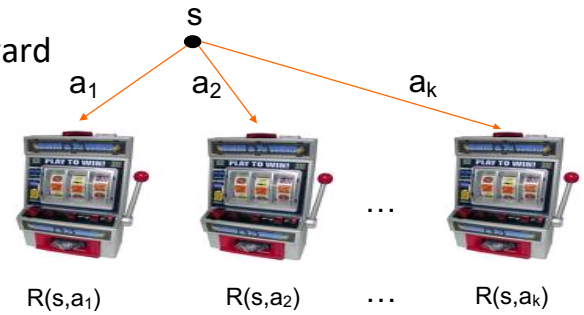
Slide adapted from **Alan Fern** (OSU)

Multi-Armed Bandits: Example 2

- **Online Advertising**
  - Arms = different ads/ad-types for a web page
  - Arm Pulls = displaying an ad upon a page access
  - Rewards = click through
  - Objective = maximize cumulative reward = maximum clicks (or find best ad quickly)

Multi-Armed Bandit: Possible Objectives

- **PAC Objective:**
  - find a near optimal arm w/ high probability
- **Cumulative Regret:**
  - achieve near optimal cumulative reward over lifetime of pulling (in expectation)
- **Simple Regret:**
  - quickly identify arm with high reward
  - (in expectation)

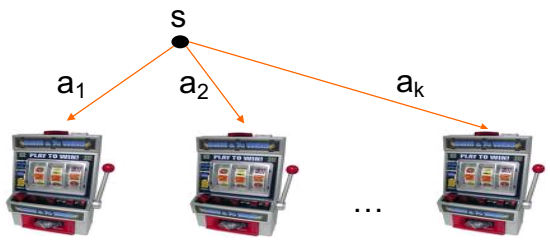


61

Slide adapted from **Alan Fern** (OSU)

Cumulative Regret Objective

- **Problem:** find arm-pulling strategy such that the expected total reward at time *n* is ***close*** to the best possible (one pull per time step)
  - Optimal (in expectation) is to pull optimal arm *n* times
  - Pull arms uniformly? (UniformBandit) ??

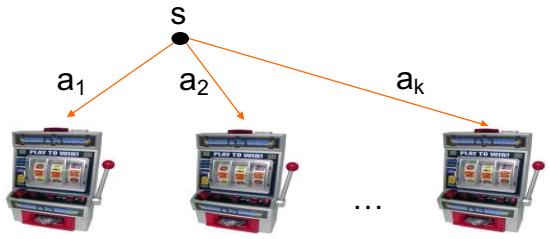


62

Slide adapted from **Alan Fern** (OSU)

Cumulative Regret Objective

- **Problem:** find arm-pulling strategy such that the expected total reward at time *n* is ***close*** to the best possible (one pull per time step)
  - Optimal (in expectation) is to pull optimal arm *n* times
  - UniformBandit is poor choice --- waste time on bad arms
  - Must balance ***exploring*** all arms to find good payoffs and ***exploiting*** current knowledge (pulling best arm)



63

Slide adapted from **Alan Fern** (OSU)

Idea

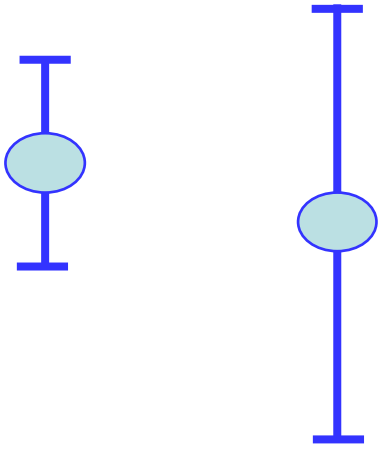
- The problem is uncertainty… How to quantify?
- Error bars

If arm has been sampled n times,
With probability at least 1- δ:

$$|\hat{\mu} - \mu| < \sqrt{\frac{\log(\frac{2}{\delta})}{2n}}$$

Slide adapted from **Travis Mandel** (UW)

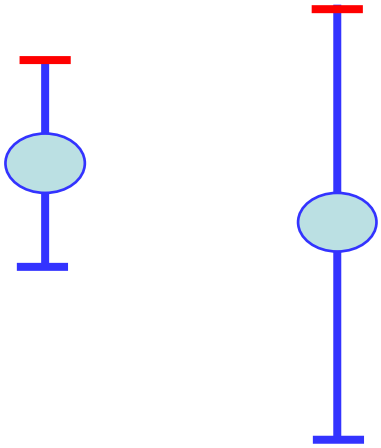
Given Error bars, how do we act?



Slide adapted from **Travis Mandel** (UW)

Given Error bars, how do we act?

- Optimism under uncertainty!
- Why? If bad, we will soon find out!



Slide adapted from **Travis Mandel** (UW)

One last wrinkle

- How to set confidence δ
- Decrease over time

If arm has been sampled n times,
With probability at least 1- δ:

$$|\hat{\mu} - \mu| < \sqrt{\frac{\log(\frac{2}{\delta})}{2n}}$$

$$\delta = \frac{2}{t}$$

Slide adapted from **Travis Mandel** (UW)

Upper Confidence Bound (UCB)

1. Play each arm once

2. Play arm i that maximizes:

$$\hat{\mu}_i + \sqrt{\frac{2\log(t)}{n_i}}$$

3. Repeat Step 2 forever

Slide adapted from **Travis Mandel** (UW)

UCB Performance Guarantee

[Auer, Cesa-Bianchi, & Fischer, 2002]

Theorem: The expected cumulative regret of UCB $\boldsymbol{E}[\boldsymbol{Reg_n}]$ after n arm pulls is bounded by O(log n)

- Is this good?

Yes. The average per-step regret is $O\left(\frac{\log(n)}{n}\right)$

Theorem: No algorithm can achieve a better expected regret (up to constant factors)

70

Slide adapted from **Alan Fern** (OSU)

UCB as Exploration Function in Q-Learning

Let N_{sa} be number of times one has executed a in s; let $N = \sum_{sa} N_{sa}$

Let $Q^e(s,a) = Q(s,a) + \sqrt{\log(N)/(1+n_{sa})}$

- **Forall s, a**
  - Initialize $Q(s, a) = 0$, $n_{sa} = 0$
- **Repeat Forever**

  Where are you? s.

  Choose action with highest Q^e

  Execute it in real world: *(s, a, r, s')*

  Do update:

  N_{sa} += 1;

  difference ← $[r + \gamma \text{Max}_{a'} Q^e(s', a')] - Q^e(s,a)$

  $Q(s,a) \leftarrow Q^e(s,a) + \alpha(\text{difference})$

71

Video of Demo Q-learning – Exploration Function – Crawler

A little history…

William R. Thompson (1933): Was the first to examine MAB problem, proposed a method for solving them

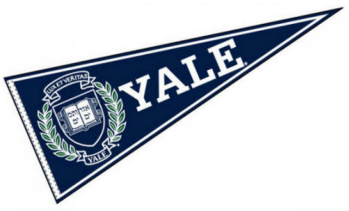
1940s-50s: MAB problem studied intentively during WWII, Thompson was ignored

1970's-1980's: "Optimal" solution (Gittins index) found but is intractable and incomplete. Thompson ignored.

2001: UCB proposed, gains widespread use due to simplicity and "optimal" bounds. Thompson still ignored.

2011: Empricial results show Thompson's 1933 method beats UCB, but little interest since no guarantees.

2013: Optimal bounds finally shown for Thompson Sampling



Slide adapted from **Travis Mandel** (UW)

Bayesian *vs.* Frequentist

- Bayesians: You have a prior, probabilities interpreted as beliefs, prefer probabilistic decisions

- Frequentists: No prior, probabilities interpreted as facts about the world, prefer hard decisions ($p<0.05$)

UCB is a frequentist technique! What if we are Bayesian?

Bayesian review: Bayes' Rule

$$p(\theta \mid data) = \frac{p(data|\theta)p(\theta)}{p(data)}$$

$$p(\theta \mid data) \propto \overbrace{p(data|\theta)p(\theta)}^{\text{Posterior}}$$

Likelihood Prior

Bernoulli Case

What if distribution in the set {0,1}
instead of the range [0,1] ?

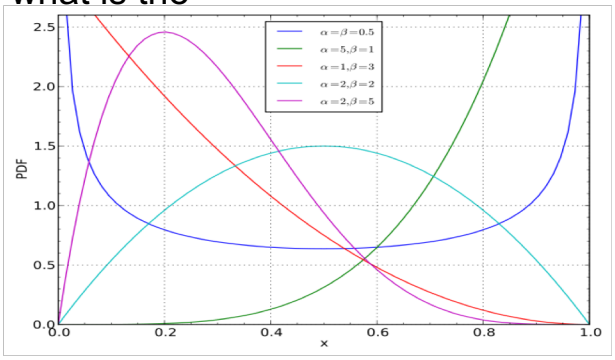
Then we flip a coin with probability p → Bernoulli distribution!

To estimate p, we count up numbers of ones and zeros

Given observed ones and zeroes, how do we calculate
the distribution of possible values of p?

Beta-Bernoulli Case

Beta(a,b) → Given a 0's and b 1's, what is the distribution over means?



Prior → pseudocounts

Likelihood → Observed counts

Posterior → pseudocounts + observed counts

How does this help us?

Thompson Sampling:

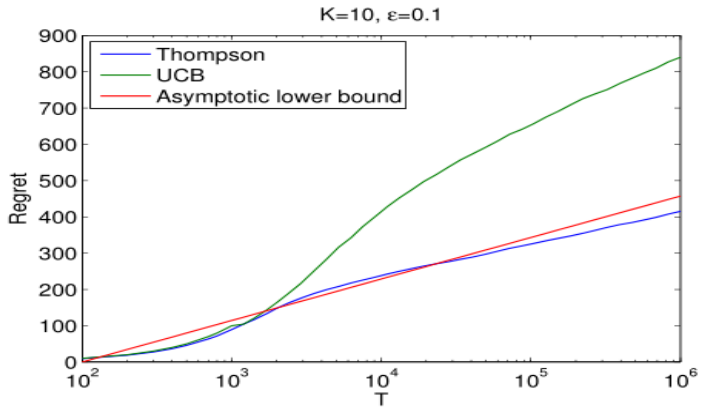
1. Specify prior (e.g., using Beta(1,1))

2. Sample from each posterior distribution to get estimated mean for each arm.

3. Pull arm with highest mean; update posterior

4. Repeat step 2 & 3 forever

Thompson Empirical Results



And shown to have optimal regret bounds just like (and in some cases a little better than) UCB!

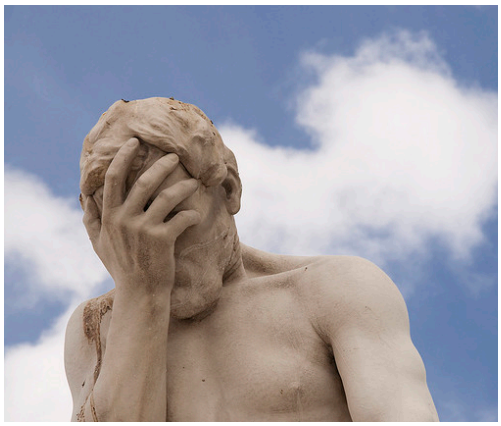
What Else ….

- UCB & Thompson is great when we care about cumulative regret
  - *I.e.,* when the agent is acting in the real world
- But, sometimes all we care about is ***finding a good arm quickly***
  - *E.g.,* when we are training in a simulator
- In these cases, "Simple Regret" is better objective

82

Two KINDS of Regret

- **Cumulative Regret:**
  - achieve near optimal cumulative lifetime reward (in expectation)
- **Simple Regret:**
  - quickly identify policy with high reward (in expectation)



83

Simple Regret Objective

- **Protocol:** At time step n the algorithm picks an "exploration" arm a_n to pull and observes reward r_n and also picks an arm index it thinks is best j_n (a_n, j_n *and* r_n *are random variables*).
  - If interrupted at time n the algorithm returns j_n.
- **Expected Simple Regret ($E[SReg_n]$):** difference between R^* and expected reward of arm j_n selected by our strategy at time n

$$E[SReg_n] = R^* - E[R(a_{j_n})]$$

84

How to Minimize Simple Regret?

What about UCB for simple regret?

Theorem: The expected simple regret of UCB after *n* arm pulls is upper bounded by $O(n^{-c})$ for a constant c.

Seems good, but we can do much better (at least in theory).

- Intuitively: UCB puts too much emphasis on pulling the best arm
- After an arm is looking good, maybe better to see if ∃a better arm

Incremental Uniform (or Round Robin)

Bubeck, S., Munos, R., & Stoltz, G. (2011). Pure exploration in finitely-armed and continuous-armed bandits. Theoretical Computer Science, 412(19), 1832-1852

Algorithm:

- At round n pull arm with index (k mod n) + 1
- At round n return arm (if asked) with largest average reward

Theorem: The expected simple regret of Uniform after *n* arm pulls is upper bounded by $O(e^{-cn})$ for a constant c.

- This bound is exponentially decreasing in n!

  Compared to polynomially for UCB $O(n^{-c})$.

86

Can we do even better?

Tolpin, D. & Shimony, S, E. (2012). MCTS Based on Simple Regret. *AAAI Conference on Artificial Intelligence.*

Algorithm -Greedy **:** (parameter)

- At round n, with probability 1/2 pull arm with best average reward so far, otherwise pull one of the other arms at random.
- At round n return arm (if asked) with largest average reward

Theorem: The expected simple regret of ϵ-Greedy for $\epsilon = 0.5$ after n arm pulls is upper bounded by $O(e^{-cn})$ for a constant c that is larger than the constant for Uniform (this holds for "large enough" n).

87

Summary of Bandits in Theory

PAC Objective:

- **UniformBandit** is a simple PAC algorithm
- **MedianElimination** improves by a factor of log(k) and is optimal up to constant factors

Cumulative Regret:

- **Uniform** is very bad!
- **UCB** is optimal (up to constant factors)
- **Thomson Sampling** also optimal; often performs better in practice

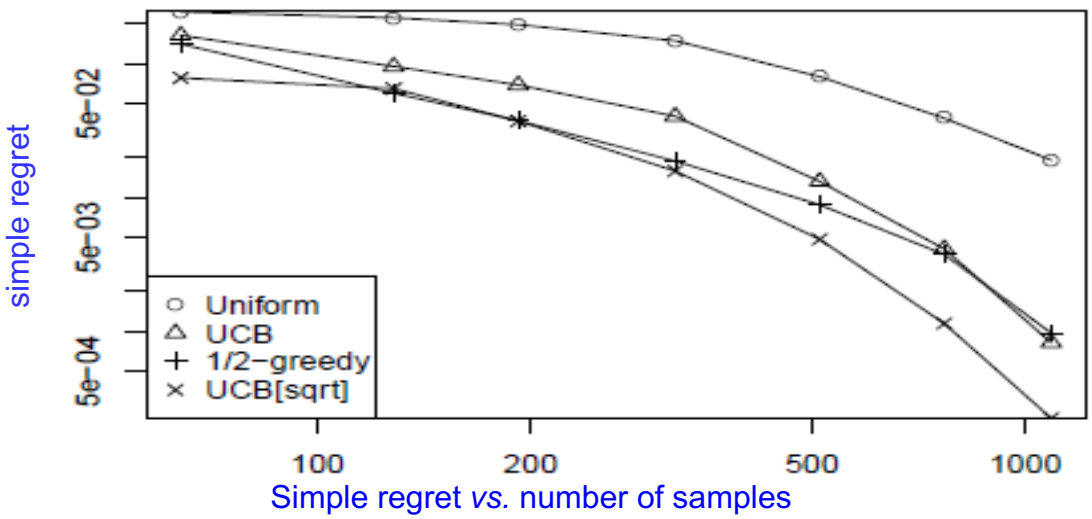
Simple Regret:

- **UCB** shown to reduce regret at polynomial rate
- **Uniform** reduces at an exponential rate
- **0.5-Greedy** may have even better exponential rate

Theory *vs.* Practice

- The established theoretical relationships among bandit algorithms have often been useful in predicting empirical relationships.
- But not always ….

Theory *vs.* Practice

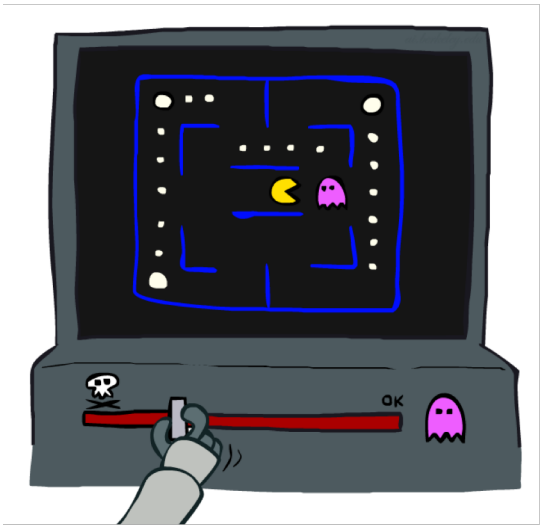


Simple regret *vs.* number of samples

UCB maximizes $Q_a + \sqrt{((2 \ln(n)) / n_a)}$

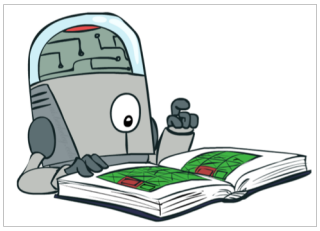
UCB[sqrt] maximizes $Q_a + \sqrt{((2 \sqrt{n}) / n_a)}$

Approximate Q-Learning

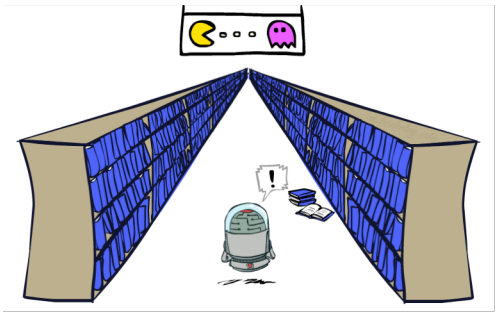


Generalizing Across States

- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory



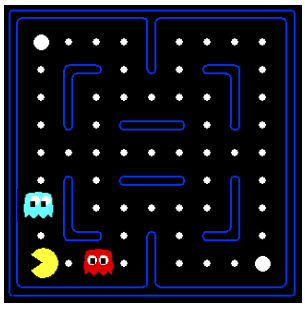
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
  - This is a fundamental idea in machine learning, and we'll see it over and over again



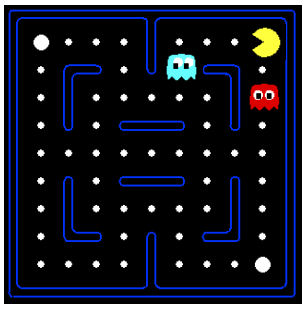
[demo – RL pacman]

Ex: Pacman – Failure to Generalize

Let's say we discover through experience that this state is bad:

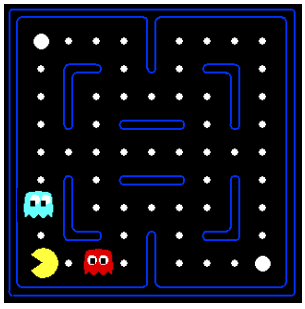


In naïve q-learning, we know nothing about this state:

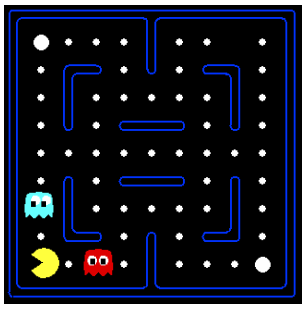


Ex: Pacman – Failure to Generalize

Let's say we discover through experience that this state is bad:



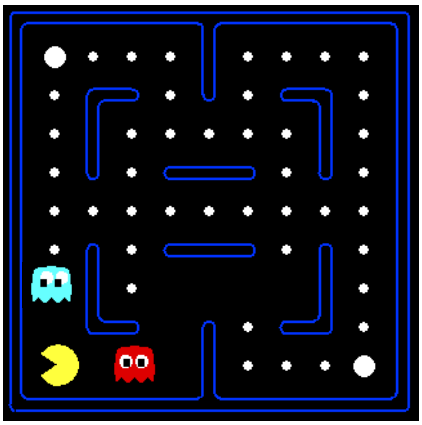
Or even this one!



Feature-Based Representations

Solution: describe a state using a **vector of features** (aka "properties")

- Features = functions from states to R (often 0/1) capturing important properties of the state
- Example features:
  - Distance to closest ghost or dot
  - Number of ghosts
  - 1 / (dist to dot)^2
  - Is Pacman in a tunnel? (0/1)
  - …… etc.
  - Is it the exact state on this slide?
- Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear Combination of Features

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states sharing features may actually have very different values!

Q Learning

- **Forall s, a**
  - **Initialize Q(s, a) = 0**
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

  difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

  Q(s,a) ← Q(s,a) + α(difference)

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- **Forall *i***
  - Initialize $w_i = 0$
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

  difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

  Q(s,a) ← Q(s,a) + α(difference)

Approximate Q Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- **Forall *i***
  - Initialize $w_i = 0$
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

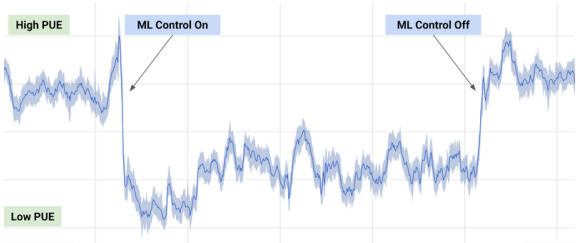
  difference ← [r + γ $\text{Max}_{a'}$ Q(s', a')] - Q(s,a)

  Forall *i* do:

  $$w_i \leftarrow w_i + \alpha \, [\text{difference}] \, f_i(s,a)$$

That's all for Reinforcement Learning!

- Very tough problem: How to perform any task well in an unknown, noisy environment!
- Traditionally used mostly for robotics, but…



Google DeepMind – RL applied to data center power usage

145