Chapter 2: Multi-armed Bandits

- RL learning key distinction: evaluates actions, does not instruct with correct actions
  + Evaluative: how good action taken was, not if best or worst
  + Instructive: what is the correct action, despite action chosen
  * supervised learning paradigm ↑
- use k-arm bandit to learn basic learning methods

A K-armed Bandit Problem

- Problem: Repeatedly faced with a choice among k different options, or actions. After each choice, receive a numerical reward chosen from a stationary probability distribution that depends on the action taken.
- Obj: Maximize total reward over a given time period (e.g. 1000 actions, or time steps)
  + slot machine analogy, but with k levers, not 1.
- each of k actions has expected value / reward
  + at time t, action chosen is A_t, reward is R_t

+ value of an arbitrary a, $q_*(a)$ is the expected reward given that a is selected

$$q_*(a) : E[R_t \mid A_t = a]$$

- We don't know the values (trivial if we did)
+ start with estimates
+ $Q_t(a)$ is estimated value of a at time t
* goal: get $Q_t(a)$ as close to $q_*(a)$ as possible
- Maintain action values, one is always greatest = **greedy** action
+ **exploiting** current knowledge
- non greedy choice is **exploring**
+ improves non greedy estimates
- exploitation maximizes short term reward, but exploring improves certainty of possible better actions for optimized long term value
+ can't explore and exploit in a single action
- balancing decisions to explore vs exploit is complex and often requires knowledge that is just not practical outside of theory
+ we attempt simple balancing, not sophisticated

Action Value Methods

- how to estimate action values?

+ action-value methods

* True value is mean reward when action is selected
* Natural possible way:

$$Q_t(a) = \frac{\text{sum of rewards of } a \text{ prior to } t}{\text{times } a \text{ prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \times \mathbb{1}_{A_i = a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}}$$

* $\mathbb{1}$predicate → random variable that is 1 if predicate is true, 0 if not.
* if denominator is 0, choose a default value

" if → ∞, law of large numbers, $Q_t(a) \to q_*(a)$ (convergence)

* Known as sample average method
* not best, but good start

- how to select action from this?

+ simplest rule: pick greedy action

* if tie, arbitrary decide

$A_t \doteq \text{argmax}_a Q_t(a)$

* argmax_a = action a that maximizes followed expression

- how to avoid greedy all the time?

+ once in a while, with probability ε, select randomly from all actions with equal probability, ignore action-value

* $\boxed{\varepsilon\text{-greedy}}$ methods (tic-tac-toe impl)

+ in the limit as steps increase, all action will be sampled infinite times, ensuring $Q_t(a)$ for all a converge $q_*(a)$

* still requires probability of selecting optimal action converges to $> 1-\varepsilon$

EXERCISE 2.1: ε-greedy, 2 actions, $\varepsilon = 0.5$, probability that greedy is selected

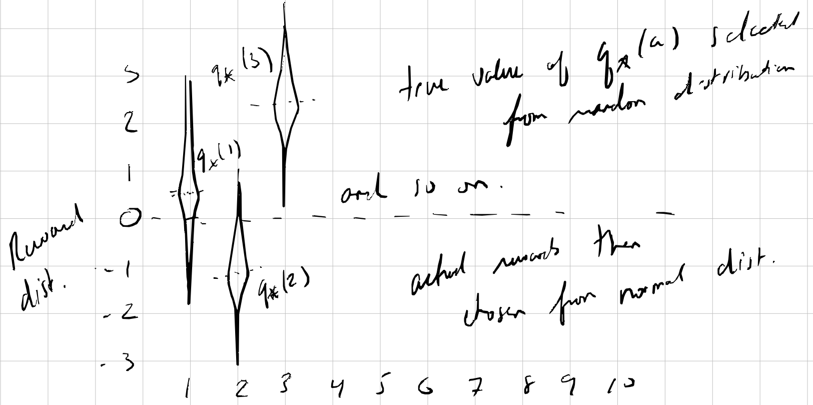
- assume one action is greedy, other not (don't have equal values)
- greedy vs random = 0.5
- random being greedy = 0.5
- probability of greedy = probability of picking greedy outright + (or) probability of randomly picking greedy

$= 0.5 + (0.5 \times 0.5) = 0.5 + 0.25 = \boxed{0.75}$

The 10-armed Testbed

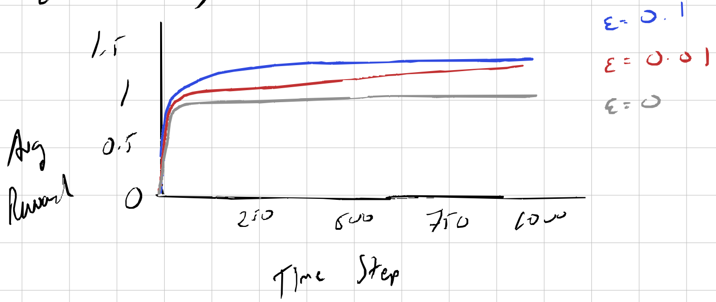
- Assess relative affectiveness of greedy vs ε-greedy
+ 2000 k-armed bandits, k = 10
+ $q_*(a)$ = 1 … 10, each with normal (Gaussian) distribution of mean 0, unit variance
+ Learning applied to problem with selected action A_t at time t, R_t selected from normal distribution with mean $q_*(A_t$

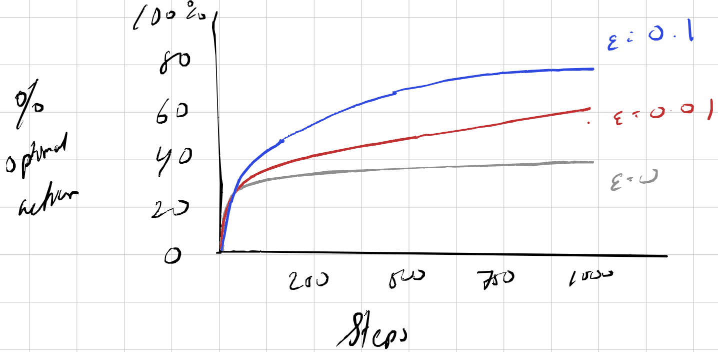
* 10-bandit testbed
* any learning method can be performance tested over 1000 time steps (1 run)
* 2000 runs, reveal learning algo's avg. behaviour



Action

- compare greedy with two ε-greedy (ε=0.01 & ε=0.1)





- all use action-value method of sampling average
+ default = 0

- best possible reward per time step was 1.54, greedy stuck at 1.
- $\varepsilon = 0.1$ found optimal move 91% of time.
+ $\varepsilon = 0.01$ did worse at first, but will do better than both over more time
* can alter ε over time to optimize search
- ε-greedy has advantages in some scenarios
+ larger reward variance? 1 → 10. More exploration required for optimal solution
+ little to no variance? greedy finds best solution almost immediately
- usually exploratory is better given weaker assumptions, like nonstationary rewards
+ constant exploration required to understand if current greedy option is no longer good
- RL balances exploration with exploitation

EXERCISE 2.2: K-armed bandit, $K = 4$ $(1, 2, 3, 4)$

ε-greedy select, sample average action-value estimate (default $Q_1(a) = 0$ for all a)

Initial sequence of actions/rewards

$A_1 = 1$, $R_1 = -1$

$A_2 = 2$, $R_2 = 1$ On some time steps

$A_3 = 2$, $R_3 = -2$ ε-case occurred (explored)

$A_4 = 2$, $R_4 = 2$

$A_5 = 3$, $R_5 = 0$

Which steps did they definitely occur? Which did they possibly?

$Q_x(a)$

t	a	1	2	3	4
init		0	0	0	0
1	1	-1	0	0	0
2	2	-1	1 $\left(\frac{1}{1}\right)$	0	0
3	2	-1	-1/2 $\left(\frac{1-2}{2}\right)$	0	0
4	2	-1	1/3 $\left(\frac{1-2+2}{3}\right)$	0	0
5	3	-1	1/3	0	0

Actions on time steps 4 & 5 are ε driven because the optimal reward was known elsewhere

All actions COULD be ε driven by pure chance. Unlikely, but still could.

EXERCISE 2.3: Which method will perform the best in terms of cumulative reward & probability of selecting best action?

3 methods: $\varepsilon = 0$, $\varepsilon = 0.01$ $\varepsilon = 0.1$

limit as $t \to \infty$

$\varepsilon = 0$, uses immediately known value

cum reward: $1 \times \infty$

$\varepsilon = 0.01$, 1% of the time uses non optimal value
99% of the time use optimal value

cum reward: $(0.01 \times Z + 0.99 \times 1.54) \times \infty$

$\varepsilon = 0.1$ 10% take nonoptimal $Z < 1.54$, 90% take optimal

cum reward: $0.1Z + 0.9(1.54) \times \infty$

$\varepsilon = 0$	0.01	0.1
∞	$(0.01Z + 1.52)\infty$	$(0.1Z + 1.386)\infty$

if you assume Z is no worse than $\varepsilon = 0$ greedy reward $(Z = 1)$ (strong assumption)

∞	1.53∞	1.486∞

reorder

$$\infty < 1.486\infty < 1.53\infty$$

on the strong assumption that $Q_{t \to \infty}(a) = q_*(a)$

$\varepsilon = 0.01$ is best in cumulative long run

% optimal action taken as $t \to \infty$

assumes

not sure. if $\varepsilon = 0.1$, wouldn't it eventually perform optimally 90% of the time. $\varepsilon = 0.01$, 99% convergence, again assuming $Q_{t \to \infty}(a) \approx q_*(a)$?

Incremental Implementation

- Compute sample average in efficient manner, constant space & constant per-time computation
- Single action simplification.
  + $R_i \to$ reward recieved after i^{th} selection of <u>this</u> action
  + $Q_n \to$ action value estimate after said action has been selected $n-1$ times
  + $Q_n \doteq \dfrac{R_1 + R_2 + \ldots R_{n-1}}{n-1}$ ($\doteq$ is definition)
- How to make this space & computation efficient like so

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

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

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

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

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

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

- holds for $n = 1 \Rightarrow Q_2 = R_1$ for arbitrary Q_1
+ needs memory for Q_n an n
+ General form:

New Estimate $\leftarrow$ Old Estimate + Step Size[Target - Old Estimate]

- [Target - Old Estimate] is $\boxed{\text{error}}$
- $\frac{1}{n}$ is step-size, changing every timestep

+ generically call step size $\boxed{\alpha}$ or $\boxed{\alpha_t(a)}$
(have $\alpha = \frac{1}{n}$)

- pseudocode for a simple bandit with incremental computed sample average, ε-greedy selection:

```
initialize for $a = 1 \dots k$
  $Q(a) = 0$
  $N(a) = 0$
Loop forever:
  $A = \begin{cases} \text{argmax}_a Q(a) & \text{with pr } (1-\varepsilon) \\ \text{random action} & \text{with pr } (\varepsilon) \end{cases}$
  $R = \text{bandit}(a)$
  $N(A) = N(A) + 1$
  $Q(A) = Q(A) + \frac{1}{N(A)} [R - Q(A)]$
```

Tracking a Nonstationary Problem

- averaging good when reward probabilities do not change
- often not the case, must weight reward by recency
- popular way: constant step-size $\alpha \in (0, 1]$

$$
\begin{aligned}
Q_{n+1} &\doteq Q_n + \alpha(R_n - Q_n) \\
&= \alpha R_n + Q_n - \alpha Q_n \\
&= \alpha R_n + (1-\alpha) Q_n \\
&= \alpha R_n + (1-\alpha)\left[Q_{n-1} + \alpha(R_{n-1} - Q_{n-1})\right] \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha) Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} + \dots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} (1-\alpha)^{n-i} \alpha R_i
\end{aligned}
$$

- now a weighted average
+ sum of weights $\Rightarrow (1-\alpha)^n + \sum_{i=1}^{n} \alpha (1-\alpha)^{n-i} = 1$
* weight $\alpha(1-\alpha)^{n-i}$ on R_i changes based on how recent i is compared to n
* $(1-\alpha) < 1$, so always decreasing
* weight decays exponentially because of $(1-\alpha)^{n-i}$
* sometimes called exponential-recency weighted average

- often good to change step size, $\alpha_n(a)$
+ $\alpha_n(a) = \frac{1}{n}$ guaranteed convergence

* stochastic approximation theory dictates conditions for convergence

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{AND} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

$$\Downarrow \qquad\qquad\qquad\qquad \Downarrow$$

large enough steps to overcome initial conditions or fluctuations

steps become small enough to assure convergence

- $\alpha_n(a) = \frac{1}{n}$ meets above, $\alpha_n(a) = \alpha$ does not

+ desirable though to continually understand new rewards in nonstationary env.
- convergence often extremely slow, require tuning, and used in theoretical, not practical, work

EXERCISE 2.4: if now $\alpha_n(a)$ not constant, what is Q_n?

(assuming for single a r.n.)

$$Q_{n+1} = Q_n + \alpha_n [R_n - Q_n]$$

$$= Q_n + \alpha_n R_n - \alpha_n Q_n \Rightarrow \alpha_n R_n + (1-\alpha_n) Q_n$$

$$= \alpha_n R_n + (1-\alpha_n)[\alpha_{n-1} R_{n-1} + (1-\alpha_{n-1}) Q_{n-1}]$$

$$= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1}) Q_{n-1}$$

$$= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1}) [\alpha_{n-2} R_{n-2} + (1-\alpha_{n-2}) Q_{n-3}]$$

$$= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2} R_{n-2} + (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2}) Q_{n-2}$$

$$= Q_1 \prod_{i=1}^{n} 1-\alpha_i + \sum_{i=1}^{n} \alpha_i R_i \prod_{j=i+1}^{n} 1-\alpha_j$$

weighting on R_i is $\boxed{\alpha_i \prod_{j=i+1}^{n} 1-\alpha_j}$

Optimistic Initial Values

- everything depends on $Q_1(a)$

+ biased by it

* sample-average removes bias after all actions are seen once

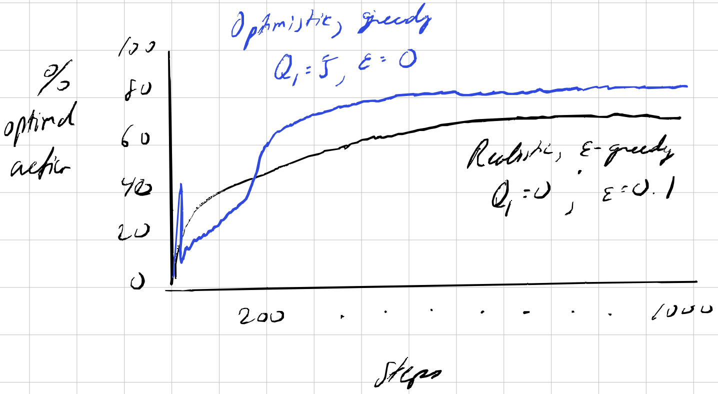
* constant α, bias always exists but matters less over time

+ becomes a parameter in need of tuning

- however, great for encouraging exploration using optimistic values (e.g. $Q_1 = 5$)

+ almost every reward will be less, causing a negative update, incentivising even greedy methods to pick other actions

- optimistic initial values great in stationary but does not generalize well



- combination of simple methods can generalize well

EXERCISE 2.6: Figure above is quite reliable because it averages over 2000 individual, random, 10 arm bandits. Why the peaks and oscillations in the very beginning?

A: Because of the optimistic initialization, every bandit explores early and quickly hits the "optimal action". But this doesn't last after the first exploration because now its expectations are tempered and doesn't move from action to action without more certainty

NOT QUITE

Stochastic nature of reward + greedy method means some occasionally get very lucky and see performance boosts. On the other hand, they get stuck in suboptimal zones if the reward biased negative more

EXERCISE 2.7: Can we avoid bias of constant step size while adapting to nonstationary problems

Step size:

$$\beta_n \doteq \frac{\alpha}{\bar{o}_n}$$

$\alpha > 0$ (conventional step)

$\bar{o}_n$ trace of 1 starting at 0.

$$\bar{o}_n \doteq \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}) \text{ for } n > 0, \quad \bar{o}_0 \doteq 0$$

$$Q_{n+1} = Q_n + \beta_n[R_n - Q_n] = Q_n + \frac{\alpha}{\bar{o}_n}[R_n - Q_n]$$

$$= \underbrace{\frac{\alpha}{\bar{o}_n}}_{1} R_n + \left(1 - \frac{\alpha}{\bar{o}_n}\right) Q_n \left.\vphantom{\frac{\alpha}{\bar{o}_n}}\right\} 2$$

$$\frac{\alpha}{\bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1})} R_n$$

from 2:

$$1 - \frac{\alpha}{\bar{o}_n}\left(\frac{\alpha}{\bar{o}_{n-1}} R_{n-1} + \left(1 - \frac{\alpha}{\bar{o}_{n-1}}\right) Q_{n-1}\right)$$

$$\underbrace{\left(1 - \frac{\alpha}{\bar{o}_n}\right)\frac{\alpha}{\bar{o}_{n-1}}}_{3} R_{n-1} + \underbrace{\left(1 - \frac{\alpha}{\bar{o}_n}\right)\left(1 - \frac{\alpha}{o_{n-1}}\right)}_{4} Q_{n-1}$$

from 1 & 3:

$$= \frac{\alpha}{o_n} R_n + \frac{\alpha}{\bar{o}_{n-1}}\left(1 - \frac{\alpha}{\bar{o}_n}\right) R_{n-1} +$$

$$\frac{\alpha}{\bar{o}_{n-2}}\left(1 - \frac{\alpha}{o_n}\right)\left(1 - \frac{\alpha}{o_{n-1}}\right) R_{n-2} + \dots$$

$$= \sum_{i-1}^{n} \frac{\alpha}{\bar{o}_n} R_n \prod_{j=i+1}^{n} 1 - \frac{\alpha}{\bar{o}_n}$$

from 4.

$$= Q_0 \prod_{i=1}^{n} 1 - \frac{\alpha}{\bar{o}_i}$$

$$o_i = o_{i-1} + \alpha(1 - \bar{o}_i - 1)$$

$$= 1 - \frac{\alpha}{\bar{o}_{i-1} + \alpha(1 - o_{i-1})} = 1 - \frac{\alpha}{\alpha + \underbrace{(1-\alpha)\bar{o}_{i-1}}_{5}}$$

from 5.

$$(1-\alpha)[\bar{o}_{i-2} + \alpha(1 - \bar{o}_{i-2})]$$

$$(1-\alpha)[\alpha + (1-\alpha)\bar{o}_{i-2}]$$

$$(1-\alpha)\alpha + (1-\alpha)^2\bar{o}_{i-2}$$

plug back in

$$1 - \frac{\alpha}{\underbrace{\alpha + (1-\alpha)\alpha + (1-\alpha)^2\bar{o}_{i-2}}_{6}}$$

from 6:

$$\sum_{k=1}^{i} \alpha(1-\alpha)^{k-1}$$

omit last term because

$$\alpha(1-\alpha)^{i-1}\bar{o}_{i-i}$$

$$\Downarrow$$

SHOULD HAVE STOPPED HERE $\alpha(1-\alpha)^{i-1}\bar{O}_0 = \underline{0}$

plug back in

$$1-\frac{\alpha}{\sum_{k=1}^{i}\alpha(1-\alpha)^{k-1}} = 1-\frac{1}{\sum_{k=1}^{i}(1-\alpha)^{k-1}}$$

$$1-\frac{\alpha}{\bar{O}_n} = 1-\left[\sum_{k=1}^{i}(1-\alpha)^{k-1}\right]^{-1}$$

IGNORE

$$= \sum_{i=1}^{n}\frac{\alpha}{\bar{O}_n}R_n\prod_{j=1}^{n}1-\frac{\alpha}{\bar{O}_n} +$$

$$O_0\prod_{i=1}^{n}1-\left[\sum_{j=1}^{i}(1-\alpha)^{j-1}\right]^{-1}$$

Holy moly help me

STUPID

The product term on O_0 goes to 0 for the first i, always zeroing it out. feels super long winded to get to that conclusion though.

using derivation

$$O_n = \alpha \sum_{i=0}^{n} (1-\alpha)^{n} \Rightarrow \text{finite geometric series}$$

$$= \alpha \left[\frac{1-(1-\alpha)^n}{1-(1-\alpha)} \right] = \cancel{\alpha} \left[\frac{1-(1-\alpha)^n}{\cancel{\alpha}} \right]$$

$$O_n = 1-(1-\alpha)^n$$

$$Q_0 \prod_{i=1}^{n} 1 - \frac{\alpha}{1-(1-\alpha)^i} \Rightarrow 1 \quad \text{for } \alpha < 1,\ n \to \infty$$

so as $n \to \infty$

$$Q_0 \prod_{i=1}^{n} 1-\alpha \Rightarrow Q(1-\alpha)^n$$

$\lim_{n \to \infty} (1-\alpha)^n = 0$, so Q_0 decays exponentially

Upper Confidence-Bound Action Selection

- how do we better explore actions than just randomly?
  + take into account potential optimality and possible uncertainty:

$$A_t = \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right]$$

$N_t(a)$: number of times a has been taken prior to time t.

$c > 0$ degree of exploration

if $N_t(a) = 0$, a is a maximizing action

- sqrt measures uncertainty
  + denominator increases, uncertainty ↑ Otherwise ↓
  + ln makes increase smaller over time, but still unbounded
  + ensures low estimate values or high certainty are selected less over time (confidence interval)
- $\boxed{\text{UCB}}$
  + "upper-bound" because uncertainty term goes to 0 as uncertainty decreases, bounded at $Q_t(a)$
- UCB outperforms ε-greedy in stationary problems, but often impractical outside bandits

EXERCISE 2.8: Why does reward spike on 11th step (figure 2.4):

A: Most likely tries all 10 actions, then hits the most rewarded first, jumping away from Q_0, before going back to exploring the uncertain actions that had less reward. bigger c = bigger spike

Gradient-Based Algorithms

- how about a numerical preference for each action as opposed to argmaxing?
+ call $H_t(a) \in \mathbb{R}$
+ larger preference more often action is taken, but no interpretation on reward
* prob is relative; add 1000 to action prefs, no diff in prob!
- $\boxed{\text{soft-max distribution}}$ (Gibbs/Boltzmann)

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

- $H_1(a) = 0$ for all a

EXERCISE 2.9. Show $\pi_t(a)$ is the same as sigmoid function (logistic) for 2 actions, used in neural nets

Sigmoid: $\sigma(a) = \frac{1}{1+e^{-a}}$

$$\pi_t(a_1) = \frac{e^{H_t(a_1)}}{e^{H_t(a_1)} + e^{H_t(a_2)}}$$

$H_t(a_1) = H_t(a_2) + \Delta H_t$ $\quad$ $H_t(a_2) = H_t(a_1) - \Delta H_t$

$$\pi_t(a_1) = \frac{e^{H_t(a_1)}}{e^{H_t(a_1)} + e^{(H_t(a_1) - \Delta H_t)}}$$

$$= \frac{\cancel{e^{H_t(a_1)}}}{\cancel{e^{H_t(a_1)}}\left(1 - e^{-\Delta H_t}\right)}$$

$$= \frac{1}{1-e^{-\Delta H}} \Rightarrow \sigma(\Delta H_t)$$

- natural learning via stochastic gradient ascent

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1-\pi(A_t)),$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha(R - \bar{R}_t)\pi(a) \quad \forall\, a \neq A_t$$

$\alpha > 0$, $\bar{R}_t \in \mathbb{R} \Rightarrow$ average of rewards up to, not including, t

$\bar{R}_1 \doteq R_1$

- $\bar{R}_t$ is baseline comparison, probability of taking A goes up or down based on comparison $\forall\, a \neq A$ goes the opposite

- understanding stochastic approximation to gradient ascent.

+ exact gradient descent uses derivatives:

$$H_{t+1}(a) = H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \qquad \circledcirc$$

$$\mathbb{E}[R_t] = \sum_x \pi_t(x)\, q_*(x)$$ expected value

- Our $H_{t+1}(a)$ is an <u>instance</u> of stochastic gradient descent, not relying on a known $q_*(a)$

performance gradient:

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(a) \right]$$

$$= \sum_x q_*(a) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

$$= \sum_x [q_*(a) - B_t] \frac{\partial \pi_t(x)}{\partial H_t(a)} \qquad ①$$

- B_t is baseline, scalar not dependent on x
- okay because gradient sum over all actions is 0 (if $H_t(a)$ goes up, $H_t(x)$ $x \neq a$ goes down)
- multiply by $\frac{\pi_t(x)}{\pi_t(x)}$

from 1:

$$= \sum_x \pi_t(x) (q_t(x) - B_t) \frac{\frac{\partial \pi_t(x)}{\partial H_t(a)}}{\pi_t(x)} \qquad ②$$

- now in expectation form (sum ∀a pr(a) × value(a))

from 2:

$$= \mathbb{E}\left[\left(q_*(A_t) - B_t \; \frac{\frac{\partial \pi_t(A_t)}{\partial H_t(a)}}{\pi_t(A_t)} \right]$$

$$= \mathbb{E}\left[(R_t - \bar{R}_t) \; \frac{\frac{\partial \pi_t(A_t)}{\partial H_t(a)}}{\pi_t(A_t)} \right] \quad ③$$

$$\bar{R}_t = B_t \qquad R_t = q_*(A_t) \text{ because } \mathbb{E}(R_t | A_t) = q_*(A_t)$$

- now assume that $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$

$$\mathbb{1}_{a=x} \begin{cases} 1 & a=x \\ 0 & a \neq x \end{cases}$$

from 3:

$$= \mathbb{E}\left[\frac{(R_t - \bar{R}_t)\, \pi_t(A_t)\,(\mathbb{1}_{a=x} - \pi_t(a))}{\pi_t(A_t)} \right]$$

$$= \mathbb{E}\left[(R_t - \bar{R}_t)(\mathbb{1}_{a=x} - \pi_t(a)) \right]$$

- done writing gradient as an expectation of something we can sample
- now to update on each time step

from 0:

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)(\mathbb{1}_{a=x} - \pi_t(a))$$
$$\forall a$$

- how does $\frac{\partial \pi_t(x)}{\partial H_t(a)} \overset{?}{=} (\mathbb{1}_{a=x} - \pi_t(a))\,\pi_t(x)$

  use quotient rule expansion (too mundane to write)
- robust convergence

Associative Search (Contextual Bandits)

- currently don't associate different actions to different situations
- true RL goal is to learn a policy:
- map situations ⇒ best actions
- Suppose k-different bandits, each with a separate task

+ probabilities change

+ can know that you are interfacing with a different bandit

+ thus, learn to associate best action with task in front of you

- associative search task or contextual bandit

+ intermediate RL problem, since reward is still immediate based on task

* grow to full RL problem if action influences next situation & reward

EXERCISE 2.10: 2 armed bandit with changing values every timestep. Values are either

case A: $q_*(a_1) = 10$ $q_*(a_2) = 20$ $Pr = 0.5$

case B: $q_*(a_1) = 90$ $q_*(a_2) = 80$ $Pr = 0.5$

Given no knowledge about env, what is the best course of action?

Greedily stick to a_1 or a_2. Both will have the same expected value in the long run:

$$\mathbb{E}[q_*(a_1)] = 50 \quad \mathbb{E}[q_*(a_2)] = 50$$

If known, we actually learn to pick a_2 in case A and a_1 in case B

$$\mathbb{E}[\text{bandit}] = \frac{90+20}{2} = 55 \quad \checkmark$$

Summary

- several methods described for bandit problem
+ greedy
+ ε-greedy
+ UCB
+ gradient
* which best?
- hard to answer, all individually have a parameter for tuning
+ UCB seems to perform best after parameter study
- these are simple but can be considered state of the art
+ more complexity is achievable, but requires more assumptions
- Balance exploration with exploitation in K-armed bandits?
+ Gittins Index but requires complete prior knowledge

* simply not practical for full RL problem
* Instance of Bayesian Sampling using priors to select next best action
* called posterior sampling or Thompson sampling

- In Bayesian setting understanding known priors can compute optimal explore / exploit balance
+ evolving distribution is information state
* can take, for 1000 steps, 2 actions, 2^{2000} leaves of possibilities. NOT feasible, might be able to approximate