4/9/2024: Algorithms for Bandit Problems

Exploration vs. **Exploitation**

Exploration: Want to try all the possible actions enough times to gain knowledge of which action is best

"Try various medicines"

Exploitation: Use current knowledge to do what seems optimal

"Prescribe the best treatment"

Algorithm: Upper Confidence Bound (UCB)

Idea:

- Player is estimating $\mu(a)$ ← expected reward when playing action a
- Estimates are uncertain
  - ↳ UCB: represent uncertainty as a confidence interval
  - "I think $\mu(a)$ is between 0.5 and 0.8"
  - 0.5 ← lower bound; 0.8 ← upper bound
- At each time t,
  play action with largest upper bound

Why? Optimistic in the face of uncertainty

↳ Choose action with highest potential to be good

Actual Algorithm:

- Define $n_t(a)$ = # of times we tried action a up until time t — "size of dataset" for action a

$n_8(1) = 3$, $n_8(2) = 4$

- Define $\hat{\mu}_t(a)$ = sample mean of rewards when taking action a up until time t

t	A_t	R_t
1	1	1
2	2	0
3	1	0
4	1	0
5	2	1
6	2	1
7	2	1
8	?	

$\hat{N}_8(1) = 1/3$ $\hat{N}_8(2) = 3/4$

How uncertain are our estimates $\hat{N}_t(a)$? iid.

A: In general, sample mean over n datapoints has variance of $\frac{\boxed{\sigma^2}}{n}$ ← variance of one sample

$\Rightarrow$ standard deviation is $\frac{\sigma}{\sqrt{n}}$ $O\left(\frac{1}{\sqrt{n}}\right)$

For UCB: For each action, we use a confidence interval of $\hat{N}_t(a) \pm \sqrt{\frac{2 \log t}{\boxed{n_t(a)}}}$

it's $O\left(\frac{1}{\sqrt{n_t(a)}}\right)$

ie

$$N(a) \in \left[\hat{N}_t(a) - \sqrt{\frac{2\log t}{n_t(a)}},\ \overbrace{\underbrace{\hat{N}_t(a)} + \underbrace{\sqrt{\frac{2\log t}{n_t(a)}}}}^{= UCB_t(a)}\right]$$

Only doing exploitation
choosing action based on this
only uses prior knowledge,
not trying to learn more
Can get stuck playing
suboptimal action

\+

Only doing exploration
choose actions that
we've tried fewer times

= balance

Why is it $\sqrt{\frac{\boxed{2 \log t}}{\boxed{n_t(a)}}}$?

Gets bigger over time (slowly)
$\Rightarrow$ Never completely rule out an action
If we avoid an action, its UCB grows over time until we take it again

Gets bigger as we collect more data
$\Rightarrow$ this term gets smaller
$\Rightarrow$ over time, do less exploration

Full UCB algorithms:

1. For $t = 1, \dots, K$: Try each action once
2. For $t = K+1, \dots, T$: choose $A_t = \arg\max_a UCB_t(a)$

Theorem: If all rewards are in $[0, 1]$:

Regret of UCB is $O\left(\sqrt{KT \log T}\right)$

Importantly: this is sublinear in T

Alternatively: Define Average Regret as $\frac{\text{Regret}}{T}$

(amount of regret per timestep)

The average regret of UCB is $O\left(\frac{\sqrt{KT \log T}}{T}\right)$

this $\to 0$ as $T \to \infty$

Reinforcement Learning

- Actions determine what rewards you observe (also in bandits)
- AND actions also can change state (of yourself, of world) (absent in bandits)

$\boxed{\text{Class selection}}$

- Action: Take some classes, not others each semester
- Reward: Enjoyment, job
- State: What subjects do you know

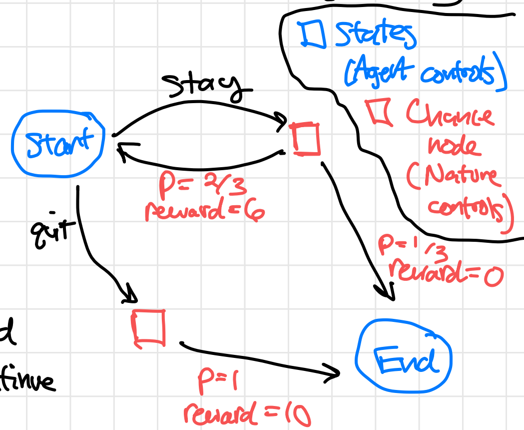
$\boxed{\text{Other examples}}$:

- Robotics
- Video games

1. Formalism to define a world (no learning yet)
2. Learn how to act in this world

① Assume the world is a Markov Decision Process (MDP)

Example MDP:



At each timestep:

- Agent can stay or quit
- If quit: Receive $10, game ends
- If stay:
  - Probability 1/3: Get $0, end
  - Probability 2/3: Get $6, continue

Formal description of MDPs:

- Set of States S (e.g. possible configurations / locations of robot)
- Starting state S_{start}
- Actions(s): Set of possible actions in state s
- $T(s, a, s')$: Probability of transitioning to state s' after taking action a in state s (e.g. $T(\text{start}, \text{stay}, \text{start}) = 2/3$)
- Reward(s, a, s'): Reward received when transitioning to state s' after taking action a in state s (e.g. Reward(start, stay, start) = 6)

Unknown during learning

- IsEnd(s): Is this an end state? Game ends when reaching end state

---

What should an agent do if MDP is known?

Policy: Strategy used by an agent, denoted π

mapping from states to actions

$$\pi(s) \rightarrow a \in \text{Actions}(s)$$

Current state   Chosen action

Value Function: The value $V_\pi(s)$ for policy π and state s is expected discounted sum of rewards starting at s, playing policy π

Discounting: Future rewards are less valuable
- At each timestep, probability of survival < 1

introduce a discount factor $\gamma \in [0, 1]$
$=$ prob. of survival at each timestep

If we got rewards $r_1, r_2, r_3, \dots$
discounted sum is $r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$