Chapter 2: Multi-armed Bandits

Seungjae Ryan Lee
One-armed Bandit

- Slot machine
- Each spin (action) is independent
Multi-armed Bandit problem

- Multiple slot machines to choose from
- Simplified setting to avoid complexities of RL problems
  - No observation
  - Action does not have delayed effect
10-armed Testbed

- 10 actions, 10 reward distributions
- Reward $R_t$ chosen from stationary probability distributions
Expected Reward

- Knowing expected reward trivializes the problem
- Estimate $q_*(a)$ with $Q_t(a)$

$$q_*(a) \doteq \mathbb{E}(R_t \mid A_t = a)$$
Sample-average

- Estimate $q_*(a)$ by averaging received rewards
- Default value (ex. 0) if action was never selected
- $Q_t(a)$ converges to $q_*(a)$ as denominator goes to infinity

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot 1_{A_i=a}}{\sum_{i=1}^{t-1} 1_{A_i=a}}$$
Greedy method

- Always select *greedily*: $A_t \doteq \arg\max_a Q_t(a)$
- No exploration
- Often stuck in suboptimal actions

Diagram:

- Agent
- Eat the usual cereal? (with an arrow labeled 1 pointing to the right)
\(\epsilon\)-greedy method

- Select random action with probability \(\epsilon\)
- All \(Q_t(a)\) converges to \(q_*(a)\) as denominator goes to infinity
Greedy vs. $\varepsilon$-greedy
Incremental Implementation

- Don’t store reward for each step

\[ Q_{n+1} = \frac{R_1 + R_2 + \ldots + R_n}{n} \]

- Compute incrementally

\[ Q_{n+1} = Q_n + \frac{1}{n} \left[ R_n - Q_n \right] \]

\[ \text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize} \left[ \text{Target} - \text{OldEstimate} \right] \]
Nonstationary problem

- $q_*(a)$ changes over time
- Want to give new experience more weight

$q_*(A_1 = a) \sim N(0, 1)$ 
$q_*(A_2 = a) \sim N(3, 1)$
Exponentially weighted average

- Constant step-size parameter $\alpha$
- Give more weight to recent rewards

\[
Q_{n+1} = Q_n + \alpha \left[ R_n - Q_n \right]
\]

\[
= (1 - \alpha)^n Q_1 + \sum_{i=1}^{n} \alpha (1 - \alpha)^{n-i} R_i
\]
### Sample-average

\[ Q_{n+1} = Q_n + \frac{1}{n} \left[ R_n - Q_n \right] \]

- Guaranteed convergence
- Converge slowly: need tuning
- Seldomly used in applications

### Weighted average

\[ Q_{n+1} = Q_n + \alpha \left[ R_n - Q_n \right] \]

- Never completely converges
- Desirable in nonstationary problems
Optimistic Initial Values

- Set initial action values optimistically (ex. +5)
-Temporarily encourage exploration
- Doesn’t work in nonstationary problems

![Diagram showing optimistic initial values]

<table>
<thead>
<tr>
<th>Value</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>R = 0</td>
</tr>
<tr>
<td>+4.5</td>
<td>R = 0.1</td>
</tr>
<tr>
<td>+4.06</td>
<td>R = -0.1</td>
</tr>
<tr>
<td>+3.64</td>
<td>R = 0</td>
</tr>
<tr>
<td>+3.28</td>
<td></td>
</tr>
</tbody>
</table>
Optimistic Greedy vs. Realistic $\epsilon$-greedy

Optimistic, greedy: $Q_1 = 5$, $\epsilon = 0$

Realistic, $\epsilon$-greedy: $Q_1 = 0$, $\epsilon = 0.1$

% Optimal action

Steps

0% 20% 40% 60% 80% 100%

1 200 400 600 800 1000
Upper Confidence Bound (UCB)

- Take into account each action’s **potential** to be optimal
- Selected less $\rightarrow$ more potential
- Difficult to extend beyond multi-armed bandits

$$A_t = \text{argmax} \left[ Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$
UCB vs. \( \varepsilon \)-greedy

![Graph showing the comparison between UCB and \( \varepsilon \)-greedy strategies. The UCB strategy with \( c = 2 \) is compared to the \( \varepsilon \)-greedy strategy with \( \varepsilon = 0.1 \). The graph depicts the average reward over steps.]
Gradient Bandit Algorithms

- Learn a numerical preference $H_t(a)$ for each action
- Convert to probability with softmax:

$$
\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b \in A} e^{H_t(b)}}
$$
Gradient Bandit: Stochastic Gradient Descent

- Update preference $H_t(a)$ with SGD

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) \quad \text{for all } a \neq A_t$$

- Baseline $\bar{R}_t$: average of all rewards $R_1, R_2, \ldots, R_t$
  - Increase probability if reward is above baseline
  - Decrease probability if reward is below baseline
Gradient Bandit: Results

![Graph showing the performance of the Gradient Bandit algorithm with different values of $\alpha$ and the presence or absence of a baseline.](image)
Parameter Study

- Check performance in best setting
- Check hyperparameter sensitivity
Associative Search (Contextual Bandit)

- Observe some *context* that can help decision
- Intermediate between multi-armed bandit and full RL problem
  - Need to learn a *policy* to *associate* observations and actions
  - Each action only affects immediate reward
Thank you!

Original content from

- [Reinforcement Learning: An Introduction by Sutton and Barto](#)

You can find more content in

- [github.com/seungjaeryanlee](#)
- [www.endtoend.ai](#)