Master in Data Science, Iscte-IUL

Data-Driven Strategy Optimization / Otimização de Estratégias Orientada por Dados

Week 3 - October, 2023

Diana A. Mendes

diana.mendes@iscte-iul.pt

Agenda

- Online and Offline RL
- Exploration vs. Exploitation
- Multi-Armed Bandit Problem

References

- David Silver RL slides
- Richard S. Sutton and Andrew G. Barto (2018, 2nd edition): Reinforcement Learning: An Introduction. MIT Press. http://incompleteideas.net/sutton/book/the-book-2nd.html

Core concepts refresh

- **Model**: Mathematical models of dynamics and reward (MRP, MDP)
- **Policy**: Function mapping states to actions
- **Value function**: future rewards from being in a state and/or action when following a particular policy
- Value Evaluation, Policy Evaluation - *Iterative Algorithms*

Online and Offline RL

- *How data (collection of experiences) is generated*?
- **Online RL**
  - the agent gathers data directly - collects experience by interacting with the environment.
  - uses this experience immediately to learn from it (update its policy).
  - this implies to train the agent directly in the real world or to have a simulator
  - the simulator can be very complex, expensive, and insecure

Online and Offline RL

- **Offline RL**
  - The agent only uses data collected from other agents or human demonstrations.
  - It does not interact with the environment.
  - Static dataset of fixed interactions
  - Must learn the best policy it can using this dataset

Exploration vs. Exploitation

- Online decision-making involves a fundamental choice:
  - Exploitation: Make the best decision given current information
  - Exploration: Gather more information (Increase knowledge)
- Greedy strategy is to always choose best known option -> Using this we may get stuck in a local optimum
- Sample efficiency and exploration remain major challenges in online reinforcement learning (RL).
- A powerful approach that can be applied to address these issues is the inclusion of offline data, such as prior trajectories from a human expert or a sub-optimal exploration policy.

Exploration vs. Exploitation

- *Restaurant Selection*
  - **Exploitation**: Go to your favorite restaurant
  - **Exploration**: Try a new restaurant
- *Online Banner Advertisement*
  - **Exploitation**: Show the most successful advertisement
  - **Exploration**: Show a new advertisement
- *Learning to play a game*
  - **Exploitation**: Play the move you believe is best
  - **Exploration**: Play an experimental move

Prediction and Control

- **Prediction**: evaluate the future (for a given policy)
- **Control**: optimise the future (find the best policy)

Learning and Planning

Two fundamental problems in *sequential decision making*

- **Reinforcement Learning**:
  - The environment is initially unknown,
  - The agent interacts with the environment,
  - The agent improves its policy
- **Planning**:
  - A model of the environment is given (known),
  - The agent plans/performs computations with this model (without external interaction),
  - The agent improves its policy
  - E.g., deliberation, reasoning, pondering, thought, search

Model-free and Model-based RL

- **Model-Free RL**: No model, Learn value functions from experience.
  - Monte Carlo Learning
  - Temporal-Difference Learning
- **Model-Based RL**: Learn a model from experience, Plan value functions using the learned model.
  - Linear Expectation Model
  - Deep Neural Network Model

Examples

- Bandits: how to trade-off exploration and exploitation.
- Dynamic Programming (model-based): how to solve prediction and control given full knowledge of the environment (Assume a model and solve the model, no need to interact with the environment)
- Model-free prediction and control: how to solve prediction and control from interacting with the environment

Multi-Armed Bandit (MAB)

- *One-armed bandit* - slang for a slot machine in a casino
  - Put in a coin and pull a lever (the arm)
  - With high probability, lose your coin (the bandit steals your money)
  - With low probability, get varying reward, rewards follow some probability distribution
- *k-armed bandit*
  - Each arm has a different reward probability
  - Goal is to maximize total reward over a sequence of plays

Multi-Armed Bandit (MAB)

- **Multi-Armed Bandit** is a set of distributions $\{\mathcal{R}_a \mid a \in \mathcal{A}\}$
- A Multi-Armed Bandit problem is a 2-tuple $(\mathcal{A}, \mathcal{R})$, where:
  - $\mathcal{A}$ is a known set of m actions (known as "**arms**")
  - $\mathcal{R}_a(r) = \mathbb{P}[r \mid a]$ is an *unknown probability distribution over rewards*, given action a
- At each step t, the agent (algorithm) selects an action $A_t \in \mathcal{A}$
- Then the environment generates a reward $R_t \sim \mathcal{R}_{A_t}$
- The agent's goal is to maximize the Cumulative Reward: $\sum_{t=1}^{T} R_i$
- We do this by learning a policy: a distribution on $\mathcal{A}$
- Note that the environment doesn't have a notion of State (if assume states -> contextual bandits)

Multi-armed bandits - Values and Regret

- The Action Value, $Q(a)$, for action a is the expected reward

$$Q(a) = E\left[R_t \mid A_t = a\right]$$

- The Optimal Value, V_*, is defined as:

$$V_* = \max_{a \in \mathcal{A}} Q(a) = \max_a E\left[R_t \mid A_t = a\right]$$

- The Regret, I_t, is the opportunity loss on a single step t

$$I_t = E\left[V_* - Q\left(a_t\right)\right]$$

- The Total Regret, L_T, is the total opportunity loss

$$L_T = \sum_{t=1}^{T} I_t = \sum_{t=1}^{T} E\left[V_* - Q\left(a_t\right)\right]$$

- Maximizing Cumulative Reward is same as Minimizing Total Regret

Multi-Armed Bandit

- Let $N_t(a)$ be the (random) number of selections of a across t steps
- The $Count_t$ of action a is defined as $E\left[N_t(a)\right]$
- Define Gap Δ_a of a as the value difference between a and optimal a^*, that is:

$$\Delta_a = V_* - Q(a)$$

- Total Regret is sum-product (over actions) of Gaps and Counts

$$L_T = \sum_{t=1}^{T} E\left[V_* - Q\left(a_t\right)\right]$$

$$= \sum_{a \in \mathcal{A}} E\left[N_T(a)\right] \cdot \left(V_* - Q(a)\right) = \sum_{a \in \mathcal{A}} E\left[N_T(a)\right] \cdot \Delta_a$$

- A good algorithm ensures small counts for large gaps

Multi-Armed Bandit - Algorithms

- We consider algorithms that estimate the value function: $\hat{Q}_t(a) \approx Q(a)$
- Most frequent algorithms:
  - Greedy
  - ϵ-greedy
  - UCB (Upper Confidence Bound)
  - EXP3 (Exponential-weight algorithm for Exploration and Exploitation)
- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret

Multi-Armed Bandit - Algorithms

- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}\left(a_t = a\right)$$

- The Greedy algorithm selects the action with highest estimated value

$$A_t = \underset{a \in \mathcal{A}}{\arg\max} \hat{Q}_{t-1}(a)$$

- Greedy algorithm can lock onto a suboptimal action forever
- Hence, Greedy algorithm has linear total regret

Multi-Armed Bandit - Algorithms

- The ϵ-Greedy algorithm continues to explore forever
- At each time-step t :
  - With probability $1-\epsilon$, select $a_t = \arg\max_{a \in \mathcal{A}} \hat{Q}_{t-1}(a)$
  - With probability ϵ, select a random action (uniformly) from $\mathcal{A}$
- Constant ϵ ensures a minimum regret proportional to mean gap

$$I_t \geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

- Hence, ϵ-Greedy algorithm has linear total regret

MAB Algorithms - ϵ-greedy

- Decaying ϵ_t-Greedy Algorithm
- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

$$
\begin{gathered}
c > 0 \\
d = \min_{a|\Delta_a > 0} \Delta_a \\
\epsilon_t = \min\left(1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{gathered}
$$

- Decaying ϵ_t-Greedy algorithm has logarithmic total regret
- Unfortunately, above schedule requires advance knowledge of gaps
- Practically, implementing some decay schedule helps considerably

MAB Algorithms - UCB

- Estimate an upper confidence $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
- This depends on the number of times N(a) has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate)
- Select action maximising Upper Confidence Bound (UCB)

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \, \hat{Q}_t(a) + \hat{U}_t(a)$$

- UCB1 algorithm

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \, Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Enjoy data science!

Take care!

Diana A. Mendes

E-mail:

diana.mendes@iscte-iul.pt

GitHub:

https://github.com/deam0304