Artificial Intelligence

CS 165A
Mar 4, 2019

Instructor: Prof. Yu-Xiang Wang

→ Contextual bandits / Off-policy evaluation
→ Reinforcement Learning
Announcement

- Two weeks before the final!

- Due to a half lecture delay in the lecture material.
  - HW3 Q6 is now Q1 for HW4.

- This Thursday (Mar 7)
  - HW3 Q1-Q5 due.
  - HW4 released

- Next Thursday (Mar 14)
  - HW4 due
  - MP2 due
  - Review lecture
Recap: off-policy evaluation under the Contextual bandits model

- **Contexts:**
  - $x_1, \ldots, x_n \sim \lambda$ drawn iid, possibly infinite domain

- **Actions:**
  - $a_i \sim \mu(a|x_i)$ Taken by a randomized “Logging” policy

- **Reward:**
  - $r_i \sim D(r|x_i, a_i)$ Revealed only for the action taken

- **Value:**
  - $v_\mu = \mathbb{E}_{x \sim \lambda} \mathbb{E}_{a \sim \mu(\cdot|x)} \mathbb{E} D[r|x, a]$ 

- We collect data $(x_i, a_i, r_i)_{i=1}^n$ by the above processes.
Recap: Off-policy Evaluation and Learning

**Off-policy evaluation**

- Estimate the value of a fixed target policy $\pi$

  $$v_\pi := \mathbb{E}_\pi [\text{Reward}]$$

**Off-policy learning**

- Find $\pi \in \Pi$ that maximizes $v_\pi$

- Using data $(x_i, a_i, r_i)_{i=1}^n$

- Often the policy $\mu$ or logged propensities $(\mu_i)_{i=1}^n$
Recap: Clinical Trial and ATE estimation

Average Treatment Effect

\[ \text{ATE} = E[Y \mid T = 1] - E[Y \mid T = 0] \]

**Ignorability Assumption**

- **Patient Information**
  - Patient information is entered into a computer

- **Random Selection**
  - The computer randomly assigns patients to two or more groups, helping to prevent bias

- **Control group**
  - Receives standard therapy

- **Investigational group**
  - Receives new treatment
Recap: Average Treatment Effect (ATE) estimation is a special case of off-policy evaluation

- **a**: Action ⟷ T: Treatment  \{0,1\}
- **r**: Reward ⟷ Y: Response variable
- **x**: Contexts ⟷ X: covariates

- Take \( a = \{0,1\}, \pi = [0.5,0.5] \)
- \( r(x,a) = [2Y(X,T=1), -2Y(X,T=0)] \)

- Then, the value of \( \pi = \) ATE
Today

1. Estimators for off-policy evaluation

2. Tabular Reinforcement Learning
Direct Method / Regression estimator

• Fit a regression model of the reward

\[ \hat{r}(x, a) \approx \mathbb{E}(r|x, a) \] using the data

• Then for any target policy

\[ \hat{u}_{DM}^{\pi} = \frac{1}{n} \sum_{i=1}^{n} \sum_{a \in A} \hat{r}(x_i, a) \pi(a|x_i) \]

Pros:
• Low-variance.
• Can evaluate on unseen context × action

Cons:
• Often high bias
• The model can be wrong/hard to learn
Inverse propensity scoring / Importance sampling

(Horvitz & Thompson, 1952)

\[
\hat{\nu}_{\text{IPS}} = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(a_i | x_i)}{\mu(a_i | x_i)} r_i
\]

Importance weights \( \equiv \rho_i \)

Pros:
- No assumption on rewards
- Unbiased
- Computationally efficient

Cons:
- High variance when the weight is large
Best of both worlds: Doubly robust estimator

$$\hat{\nu}_\text{DR}^\pi = \hat{\nu}_\text{DM}^\pi + \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} (r_i - \hat{r}(x, a))$$

Main goal:
- Reduce variance
- Same idea as the use of a heuristic in A* search.

Baseline
Correction of the baseline using IPS
Performance of IPS, DM, DR on datasets

Recall from STAT101: \[ \text{MSE} = \text{Bias}^2 + \text{Variance} \]

Doubly robust with estimated propensity

\[ \hat{v}_{DR}^{\pi} = \hat{v}_{DM}^{\pi} + \frac{1}{n} \sum_{i=1}^{n} \pi \left( \frac{a_i | x_i}{\hat{\mu}_i} \right) (r_i - \hat{r}(x, a)) \]

- Why is it called Doubly Robust?

- Doubly robustness to model misspecification.
  - Consistency: as \(n \rightarrow \infty\), estimator \(\rightarrow\) true parameter
  - Consistent if either the reward model is unbiased or the propensity is consistent.
You can use IPS to construct a near-optimal algorithm for Contextual Bandits!

• Policy elimination
  – Use constructed IPS confidence interval to estimate regret and eliminate policies
  – Use logging policy that yields low variance IPS for all remaining policies.

• Motivates a coordinate descent algorithm
  – That makes efficient use of an “argmax oracle”.

• Out of scope for this class
So far

• RL applications, basic problem setup
  – Tabular MDP (illustrated using a Grid World example)

• Simplifying the RL and decomposing the challenges:
  – Multi-arm bandits: Challenges in the need to actively collect data
    • Solution: Explore-exploit, eps-greedy, UCB.
  – Contextual bandits: Challenges in large/continuous state-space.
    • Solution: Work with a policy class.

• It remains to address the challenges in planning
  – Tabular MDP
Recap: Reinforcement learning problem setup

- **State, Action, Reward and Observation**
  \[ S_t \in S \quad A_t \in A \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O} \]

- **Policy:**
  - When the state is observable: \[ \pi : S \rightarrow A \]
  - Or when the state is not observable
    \[ \pi_t : (\mathcal{O} \times A \times \mathbb{R})^{t-1} \rightarrow A \]

- **Learn the best policy that maximizes the expected reward**
  
  - **Finite horizon (episodic) RL:** \[ \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{T} R_t] \]
  
  - **Infinite horizon RL:** \[ \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \right] \]

  \[ \gamma: \text{discount factor} \]

  \[ T: \text{horizon} \]
Tabular MDP

- State, Action, Reward and Observation
  \[ S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O} \]

- Policy:
  - When the state is observable: \( \pi : \mathcal{S} \rightarrow \mathcal{A} \)
  - Or when the state is not observable
    \[ \pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A} \]

- Learn the best policy that maximizes the expected reward
  - Finite horizon (episodic) RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{T} R_t \right] \)
  - Infinite horizon RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right] \)

\( \gamma \): discount factor
Tabular MDP

• Simplifications in a Tabular MDP:
  – Finite State, Action (S States, A actions)
  – Observable state
  – Bounded reward (or at least bounded variance)
  – No restriction on the policy class

• Parameters of a tabular MDP:
  – Transition dynamics $P(s'| s,a)$   How many parameters?  $S(S-1)A$
  – Expected immediate reward $E[r | s,a]$   How many parameters?  $SA$
  – Initial state distribution $P(s)$   How many parameters?  $S-1$

• Do we need to accurately estimate all $S^2A$ parameters?

Potential Final exam question:
1. How to draw a BayesNet graphical model for an MDP?
2. How to draw a State-space diagram of an MDP?
Value functions

- state value function: \( V^\pi(s) \)
  - expected return when starting in \( s \) and following \( \pi \)

- state-action value function: \( Q^\pi(s,a) \)
  - expected return when starting in \( s \), performing \( a \), and following \( \pi \)

- useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using \( Q^\pi(s,a) \)

- Bellman equation

\[
V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} \left[ r^a_{ss'} + \gamma V^\pi(s') \right] = \sum_a \pi(s, a) Q^\pi(s, a)
\]
Optimal value functions

• there’s a set of optimal policies
  – $V^\pi$ defines partial ordering on policies
  – they share the same optimal value function

$$V^*(s) = \max_{\pi} V^\pi(s)$$

• Bellman optimality equation

$$V^*(s) = \max_a \sum_{s'} P_{ss'}^a \left[ r_{ss'}^a + \gamma V^*(s') \right]$$

  – system of n non-linear equations
  – solve for $V^*(s)$

$$Q^*(s, a) = \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma \max_a Q^*(s', a)]$$

• having $Q^*(s, a)$ makes it even simpler

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$
The Bellman equation implies that

- If we can estimate $Q^*(s,a)$, then the greedy policy on $Q^*(s,a)$ is the optimal policy!

- Recall in bandits problem:
  - The optimal policy is the greedy policy on $E[r | x, a]$

- The problem reduces to estimating $Q^*(s,a)$
  - You don’t need to estimate $Q^*(s,a)$ well for all s, a pairs
  - Only need to identify the best a for every s.

How many parameters do we need to represent $Q^*(s,a)$?  Answer: SA
Example from Lecture 14: Robot in a room

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- what’s the strategy to achieve max reward?
- what if the actions were deterministic?
Let’s work out the Value function for a specific policy

actions: UP, DOWN, LEFT, RIGHT

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

\[
V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} \left[ r^a_{ss'} + \gamma V^\pi(s') \right] = \sum_a \pi(s, a) Q^\pi(s, a)
\]

1.0 + \( \frac{8}{9} \times (1-0.04 + 0) \)

\( \frac{1}{9} \times (-0.04 + V^\pi(s')) \)
Dynamic programming

• main idea
  – use value functions to structure the search for good policies
  – need a perfect model of the environment

• two main components
  – policy evaluation: compute $V^\pi$ from $\pi$
  – policy improvement: improve $\pi$ based on $V^\pi$

  – start with an arbitrary policy
  – repeat evaluation/improvement until convergence
Policy evaluation/improvement

• policy evaluation: $\pi \rightarrow V^\pi$
  - Bellman eqn’s define a system of n eqn’s
  - could solve, but will use iterative version
    \[
    V_{k+1}(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma V_k(s')]
    \]
  - start with an arbitrary value function $V_0$, iterate until $V_k$ converges

• policy improvement: $V^\pi \rightarrow \pi'$
  \[
  \pi'(s) = \arg \max_a Q^\pi(s, a)
  = \arg \max_a \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma V^\pi(s')]
  \]
  - $\pi'$ either strictly better than $\pi$, or $\pi'$ is optimal (if $\pi = \pi'$)
Policy/Value iteration

- **Policy iteration**
  \[ \pi_0 \to E \ V^{\pi_0} \to I \ \pi_1 \to E \ V^{\pi_1} \to I \ldots \to I \ \pi^* \to E \ V^* \]
  - two nested iterations; too slow
  - don’t need to converge to \( V^{\pi_k} \)
    - just move towards it

- **Value iteration**
  \[ V_{k+1}(s) = \max_a \sum_{s'} P^{a}_{ss'} \left[ r^{a}_{ss'} + \gamma V_k(s') \right] \]
  - use Bellman optimality equation as an update
  - converges to \( V^* \)
Drawback of the Dynamic Programming Approach

• need complete model of the environment and rewards
  – robot in a room
    • state space, action space, transition model

• ExploreFirst:
  – Randomly sample for a while, estimate the MDP parameters.
  – Switch to Greedy using DP.

• Do we need the model? Can we learn the Q function directly?
  – Monte-Carlo methods for estimating $Q^\pi$

Only $S$ parameters for policy evaluation: $V^\pi(s)$

$SA$ parameters for policy improvements: $Q^\pi(s,a)$
Monte Carlo policy evaluation

- want to estimate $V^\pi(s)$
  - expected return starting from $s$ and following $\pi$
  - estimate as average of observed returns in state $s$

- first-visit MC
  - average returns following the first visit to state $s$

\[
V^\pi(s) \approx \frac{2 + 1 - 5 + 4}{4} = 0.5
\]
Monte Carlo control (policy optimization)

- $V^\pi$ not enough for policy improvement
  - need exact model of environment

- estimate $Q^\pi(s,a)$
  $$\pi'(s) = \arg\max_a Q^\pi(s,a)$$

- MC control
  $$\pi_0 \rightarrow^E Q^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E Q^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E Q^*$$
  - update after each episode

- Two problems
  - greedy policy won’t explore all actions  **eps-greedy!**
  - We are maximizing $Q^\pi(s,a)$ but is it a good estimate of $Q^\pi'(s,a)$?

**Importance sampling!**
DP + MC = Temporal Difference Learning

• Idea of Monte Carlo
  \[ V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)] , \]
  Issue: \( G_t \) can only be obtained after the entire episode!

• The idea of TD learning:
  \[ \mathbb{E}_\pi[G_t] = \mathbb{E}_\pi[R_t | S_t] + \gamma V^\pi(S_{t+1}) \]
  We only need one step before we can plug-in and estimate the RHS!

• TD-Policy evaluation
  \[ V(S_t) \leftarrow V(S_t) + \alpha \left[ R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right] \]
  Bootstrapping!
Bootstrap’s origin

• “The Surprising Adventures of Baron Münchausen”
  – Rudolf Erich Raspe, 1785

In statistics: Brad Efron’s resampling methods
In operating systems: Booting…
In RL: It simply means TD learning
TD-control (policy optimization)

- **SARSA**
  - Update the Q function by bootstrapping Bellman Equation
    \[ Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)] \]
  - Choose the next A’ using Q, e.g., eps-greedy.

- **Q-Learning**
  - Update the Q function by bootstrapping Bellman Optimality Eq.
    \[ Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)] \]
  - Choose the next A’ using Q, e.g., eps-greedy.

**Remarks:**
- These are **proven to converge** asymptotically.
- Much more data-efficient in practice, than MC.
- Regret analysis is still active area of research.
Extension (using ideas we’ve already learned)

• Using doubly robust with a heuristic function?
  – Yes, it’s called RL with Function approximation! Part II of Sutton and Barto.

• Sometimes it does not converge…
  – Big open problem: How to use Function approximation with TD while still guarantee convergence?

• Idea from contextual bandits:
  – Parameterize the policy class (A neural network)
  – Direct online policy optimization?
Policy gradient

- Let’s not worry about states, dynamics, Q function.
  - We might not even observe the true state.
  - Let’s specify a class of parametrized policy and hope to compare to the best within this class

- Objective function to maximize: \( J(\theta) = v_{\pi_\theta}(s_0) \),

- Do SGD: \( \theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \),

- Policy gradient theorem:

\[
\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla_\theta \pi(a|s, \theta),
\]
The REINFORCE algorithm (Williams, 1987)

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$
    For each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step $t$
    $\theta \leftarrow \theta + \alpha \gamma^t G \nabla_{\theta} \ln \pi(A_t|S_t, \theta)$

• Essentially a reward-weighted MLE
• When the reward are nonnegative
  – Optimizing a causal policy-improvement Lower Bound (Ma, W., Narayanaswamy, 2019)
Alpha-Go!

- Parameterize the policy networks with CNN
- Supervised learning initialization
- RL using Policy gradient
- Fit Value Network (This is a heuristic function!)
- Monte-Carlo Tree Search

https://www.youtube.com/watch?v=4D5yGiYe8p4

Summary of RL algorithms

• Model-based:
  – Policy iteration / Value iteration
  – Global optimal solution (need to know the dynamics)

• Model-free: (no need to “explicitly” estimate dynamics)
  – SARSA, Q-learning
  – Global optimal solution (when the model’s correctly specified)

• Absolutely model-free (do not even need MDP model)
  – Policy gradient
  – Local search, local optimal solution