Bayesian Decision Making

May 7, 2020

Sid Banerjee, ORIE, Cornell
## Bayesian decision theory in learning

Given prior $F$ on $\theta$, choose ‘action’ $\hat{\theta}$ to minimize loss function $\mathbb{E}_F[L(\theta, \hat{\theta})]$

**Examples**

- **$L_0$ loss**: $L(\theta, \hat{\theta}) = 1_{\{\theta \neq \hat{\theta}\}} \Rightarrow \hat{\theta}_{L_0} = \text{mode of } F$
  
  *(Ex: spam filtering)*

- **$L_1$ loss**: $L(\theta, \hat{\theta}) = ||\theta - \hat{\theta}||_1 \Rightarrow \hat{\theta}_{L_1} = \text{median of } \theta$ under $F$

- **$L_2$ loss**: $L(\theta, \hat{\theta}) = ||\theta - \hat{\theta}||_2 \Rightarrow \hat{\theta}_{L_2} = \mathbb{E}_F[\theta]$

## Decision theory in ‘decision-making’

Given prior $F$ on $X$, choose ‘action’ $a \in A$ to minimize loss, i.e.

$$a^* = \text{arg min}_{a \in A} \mathbb{E}_{X \sim F}[L(a, X)]$$

*Posterior for $X$ given data*$

*Ex: $X \sim \text{stock price in 1 day}$

$p = \text{price of stock today}$

$a \in \{0, 1\}$: ‘buy’

$L(0, x) = 0, L(1, x) = (x - p)^+ x$
example: Bayesian optimization

\[
\text{Aim: } \max_{A} \mathbb{E}[f(A)], \quad f \text{ unknown final choice of } X
\]

- Choose points \( X_1, X_2, \ldots, X_s \)
- Pick \( A \in \mathbb{R} \) s.t. \( \max f(A) \)

- Decision problem: choice of \( X_1, X_2, \ldots, X_s, A \)
  - Easier problem: Pick \( X_s, A \) given \( X_1, \ldots, X_{s-1} \)
  - "Heuristic": Pick \( X_s \) to maximize \( \mathbb{E}[f(A) | X_1, \ldots, X_s] \)
  - Pick \( A \) to \( \max \mathbb{E}[f(A) | X_1, \ldots, X_s] \)

As an MDP: \( X_1 \rightarrow f(x_1) \rightarrow X_2 = \Phi(x_1, f(x_1)) \rightarrow f(x_2) \rightarrow \ldots \rightarrow f(x_s) \rightarrow A \rightarrow f(A) \)
next, we play a game  

[stochastic variant of Nim]

• Setup: A pile of 10 toothpicks

• You will be playing against an oblivious random adversary (called Computer).

• A Sequence of Events in Each Iteration:
  – You start first. You can take either 1 or 2 toothpicks from the pile.
  – After you make the decision, I will flip a random fair coin. If the coin lands HEAD, the Computer will remove 1 toothpick from the pile. Otherwise, the Computer will remove 2 toothpicks.

• The game proceeds until all toothpicks are removed from the pile.
• If you end up holding the last toothpick, you win $20. Otherwise, you get nothing.

Courtesy: Paat Rusmevichientong

(note: this is a variant of a game called Nim; see Youtube video)
talking of playing games (in memorium)

for more on such games, see *winning ways for mathematical plays*

Conway, Berlekamp, Guy
analyze the game

(sequential decision making)

divide game into rounds:
- in each round, you go first followed by COMPUTER
- In $k^{th}$ round, computer picks $X_k \sim Unif\{1,2\}$ toothpicks
analyzing the game

divide game into rounds:

- in each round, you go first followed by COMPUTER
- In $k^{th}$ round, computer picks $X_k \sim \text{Unif}\{1, 2\}$ toothpicks

observations

- if the game starts with 1 or 2 toothpicks, then we win!
  (if game starts with 0 toothpicks, assume we lose.)
analyze the game

divide game into rounds:
- in each round, you go first followed by COMPUTER
- In \( k^{th} \) round, computer picks \( X_k \sim \text{Unif}\{1, 2\} \) toothpicks

observations

- if the game starts with 1 or 2 toothpicks, then we win!
  (if game starts with \( \leq 0 \) toothpicks, assume we lose.)
  \( \text{ie, if } S_k \leq 0, \text{ then loss} \)
- suppose after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks left, and let \( S_{k+1} \) be number of toothpicks left when we play next:
  - if we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - if we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

\[
S_k \rightarrow \text{player picks } X_k \rightarrow S_{k+1}
\]
analyzing the game

divide game into **rounds**:
– in each round, you go first followed by COMPUTER
– in \( k^{th} \) round, computer picks \( X_k \sim \text{Unif}\{1, 2\} \) toothpicks

**observations**

- if the game starts with 1 or 2 toothpicks, then we win!
  (if game starts with 0 toothpicks, assume we lose.)
- suppose after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks left, and let \( S_{k+1} \) be number of toothpicks left when we play next:
  - if we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - if we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

to ‘solve’ this game, we use **dynamic programming**.
analyzing the game

- if after $k - 1$ rounds, game has $S_k \geq 3$ toothpicks, and $S_{k+1}$ is number of toothpicks when we play next:
  - If we pick 1 match, then $S_{k+1} = S_k - 1 - X_k$
  - If we pick 2 match, then $S_{k+1} = S_k - 2 - X_k$

((where $X_k \sim \text{Unif}\{1, 2\}$)

let $V(x) = \max \mathbb{E}[\text{Reward}]$ if round starts with $x$ toothpicks (Value fn)
analyzing the game

- if after $k - 1$ rounds, game has $S_k \geq 3$ toothpicks, and $S_{k+1}$ is number of toothpicks when we play next:
  - If we pick 1 match, then $S_{k+1} = S_k - 1 - X_k$
  - If we pick 2 match, then $S_{k+1} = S_k - 2 - X_k$

(where $X_k \sim Unif\{1, 2\}$)

let $V(x) = \max \mathbb{E}[\text{Reward}]$ if round starts with $x$ toothpicks

- $V(-1) = V(0) = 0, V(1) = V(2) = 20$. Want to find $V(10)$
analyzing the game

- if after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks, and \( S_{k+1} \) is number of toothpicks when we play next:
  - If we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - If we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

(\( X_k \sim Unif\{1, 2\} \))

let \( V(x) = \max \mathbb{E}[\text{Reward}] \) if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, \ V(1) = V(2) = 20 \). Want to find \( V(10) \)
- \( V(3) = \max \mathbb{E}[\text{Reward}] \) if round starts with 3 toothpicks
an analyzing the game

• if after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks, and \( S_{k+1} \) is number of toothpicks when we play next:
  - If we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - If we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

(\text{where } X_k \sim \text{Unif}\{1, 2\})

let \( V(x) = \max \mathbb{E}[\text{Reward}] \) if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, \ V(1) = V(2) = 20 \). Want to find \( V(10) \)
- \( V(3) = \max \mathbb{E}[\text{Reward}] \) if round starts with 3 toothpicks
  \[ = \max \left\{ \mathbb{E}[R \text{ if we pick 1 of 3}], \mathbb{E}[R \text{ if we pick 2 of 3}] \right\} \]
analyzing the game

- if after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks, and \( S_{k+1} \) is number of toothpicks when we play next:
  - If we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - If we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

  (where \( X_k \sim Unif\{1, 2\} \))

let \( V(x) = \max \mathbb{E}[\text{Reward}] \) if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, V(1) = V(2) = 20. \) Want to find \( V(10) \)
- \( V(3) = \max \mathbb{E}[\text{Reward}] \) if round starts with 3 toothpicks
  \[ = \max \left\{ \mathbb{E}[R \text{ if we pick 1 of 3}], \mathbb{E}[R \text{ if we pick 2 of 3}] \right\} \]
  \[ = \max \left\{ \mathbb{E}[V(3 - 1 - X)], \mathbb{E}[V(3 - 2 - X)] \right\} \]
analyzing the game

- if after \( k - 1 \) rounds, game has \( S_k \geq 3 \) toothpicks, and \( S_{k+1} \) is number of toothpicks when we play next:
  - If we pick 1 match, then \( S_{k+1} = S_k - 1 - X_k \)
  - If we pick 2 match, then \( S_{k+1} = S_k - 2 - X_k \)

(\( \text{where } X_k \sim \text{Unif}\{1, 2\} \))

let \( V(x) = \max \mathbb{E}[\text{Reward}] \) if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, V(1) = V(2) = 20. \) Want to find \( V(10) \)
- \( V(3) = \max \mathbb{E}[\text{Reward}] \) if round starts with 3 toothpicks
  = \( \max \left\{ \mathbb{E}[R \text{ if we pick 1 of 3}], \mathbb{E}[R \text{ if we pick 2 of 3}] \right\} \)
  = \( \max \left\{ \mathbb{E}[V(3 - 1 - X)], \mathbb{E}[V(3 - 2 - X)] \right\} \)
  = \( \max \left\{ \left( \frac{V(1) + V(0)}{2} \right), \left( \frac{V(0) + V(-1)}{2} \right) \right\} \)
  = \( 10 \)
analyzing the game

\[ V(x) = \max \mathbb{E}[\text{Reward}] \] if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, \ V(1) = V(2) = 20 \). Want to find \( V(10) \)
- \( V(3) = \max \left\{ 0.5(V(1) + V(0)), 0.5(V(0) + V(-1)) \right\} = 10 \)
- \( V(4) = \max \left\{ 0.5(V(2) + V(1)), 0.5(V(1) + V(0)) \right\} = 20 \)
- \( V(5) = \max \left\{ 0.5(V(3) + V(2)), 0.5(V(2) + V(1)) \right\} = 20 \)
- \( V(6) = \max \left\{ 0.5(V(4) + V(3)), 0.5(V(3) + V(2)) \right\} = 15 \)
- \( V(7) = \max \left\{ 0.5(V(5) + V(4)), 0.5(V(4) + V(3)) \right\} = 20 \)
- \( V(8) = \max \left\{ 0.5(V(6) + V(5)), 0.5(V(5) + V(4)) \right\} = 20 \)
- \( V(9) = \max \left\{ 0.5(V(7) + V(6)), 0.5(V(6) + V(5)) \right\} = 17.5 \)
- \( V(10) = \max \left\{ 0.5(V(8) + V(7)), 0.5(V(7) + V(6)) \right\} = 20 \)

optimal policy: move to nearest multiple of 3

we always win if \( x \neq 0 \mod(3) \)
analyzing the game

\[ V(x) = \max \mathbb{E}[\text{Reward}] \] if round starts with \( x \) toothpicks

- \( V(-1) = V(0) = 0, \ V(1) = V(2) = 20 \). Want to find \( V(10) \)
- \( V(3) = \max \left\{ 0.5(V(1) + V(0)), 0.5(V(0) + V(-1)) \right\} = 10 \)
- \( V(4) = \max \left\{ 0.5(V(2) + V(1)), 0.5(V(1) + V(0)) \right\} = 20 \)
- \( V(5) = \max \left\{ 0.5(V(3) + V(2)), 0.5(V(2) + V(1)) \right\} = 20 \)
- \( V(6) = \max \left\{ 0.5(V(4) + V(3)), 0.5(V(3) + V(2)) \right\} = 15 \)
- \( V(7) = \max \left\{ 0.5(V(5) + V(4)), 0.5(V(4) + V(3)) \right\} = 20 \)
- \( V(8) = \max \left\{ 0.5(V(6) + V(5)), 0.5(V(5) + V(4)) \right\} = 20 \)
- \( V(9) = \max \left\{ 0.5(V(7) + V(6)), 0.5(V(6) + V(5)) \right\} = 17.5 \)
- \( V(10) = \max \left\{ 0.5(V(8) + V(7)), 0.5(V(7) + V(6)) \right\} = 20 \)

**optimal policy**: move to nearest multiple of 3

We always win if \( x \neq 0 \mod (3) \)
**sequential decision making**

**Markov decision process (MDP)**

general paradigm for sequential decision making

**problem:** \( \max_{a: "Actions"} \mathbb{E}_X[f(X_1, a_1, X_2, a_2, \ldots, X_T, a_T)] \)

- **state:** \( S \) - summary of history
- **value function:** \( V(\cdot) \) - 'value-to-go' for given state
- **Bellman equation** (or dynamic program equation):
  \[
  V(S_t) = \max_{a_t: \text{actions}} \mathbb{E}
  \left[
  R_t(S_t, a_t) + V(S_{t+1}(S_t, a_t))
  \right]
  \]
- **optimal policy:** pick any \( a_t \) that is a maximizer of above eqn
Markov chain vs. Markov decision process

Markov chain

MDP

'Solution' to an MDP

\[ T = \{1, 2, \ldots, T\} \]
\[ S_t \in \{1, 2, \ldots, S^3\} \]
\[ V_t(s) \]

for state \( s \) at time \( t \)

store: \[ V_t(s) = \varepsilon \]

\[ \alpha_t^*(s) = \operatorname{argmax}_a \left( \mathbb{E} R_t(s_a, a) + V_{t+1}(s) \right) \]
(finite horizon) MDP

sequential decision making: \( \max_{a: \text{"Actions"}} \mathbb{E}_X[f(a, X)] \)

main concepts

- **horizon**: \( T \) - discrete ‘decision periods’ \( t = \{1, 2, \ldots, T\} \)
**finite horizon** MDP

Sequential decision making: \( \max_{a: \text{"Actions"}} \mathbb{E}_X[f(a, X)] \)

**Main concepts**

- **Horizon**: \( T \) - discrete ‘decision periods’ \( t = \{1, 2, \ldots, T\} \)
- **State**: \( s_t \in S_t \) - concise summary of history
(finite horizon) MDP

Sequential decision making: \( \max_{a: "Actions"} \mathbb{E}_X[f(a, X)] \)

Main concepts

- **Horizon**: \( T \) - discrete 'decision periods' \( t = \{1, 2, \ldots, T\} \)
- **State**: \( s_t \in \mathcal{S}_t \) - concise summary of history
- **Action**: \( a_t \in \mathcal{A}(s_t) \) - allowed set actions in each period
(finite horizon) MDP

sequential decision making: \( \max_{a: \text{“Actions”}} \mathbb{E}[f(a, X)] \)

main concepts

- **horizon**: \( T \) - discrete ‘decision periods’ \( t = \{1, 2, \ldots, T\} \)
- **state**: \( s_t \in S_t \) - concise summary of history
- **action**: \( a_t \in A(s_t) \) - allowed set actions in each period
- **randomness/disturbance**: \( X_t \) - determines state transition probability \( p(s_{t+1} | s_t, a_t) \) (or \( s_{t+1} = f(s_t, a_t, X_t) \))
(finite horizon) MDP

sequential decision making: \( \max_{a: "Actions"} \mathbb{E}_X[f(a, X)] \)

main concepts

- **horizon**: \( T \) - discrete ‘decision periods’ \( t = \{1, 2, \ldots, T\} \)
- **state**: \( s_t \in S_t \) - concise summary of history
- **action**: \( a_t \in A(s_t) \) - allowed set actions in each period
- **randomness/disturbance**: \( X_t \) - determines state transition probability \( p(s_{t+1}|s_t, a_t) \) (or \( s_{t+1} = f(s_t, a_t, X_t) \))
- **Reward**: \( R_t(s_t, a_t, X_t) \) (or \( R_t(s_{t+1}|s_t, a_t) \))
‘solving’ an MDP

dynamic programming

- **value function**: \( V_t(s) \triangleq \text{maximum expected expected reward over periods } \{t, t+1, \ldots, T\} \text{ starting from state } s \)
- **terminal conditions**: \( V_T(s) \) for all \( s \)
- **Bellman equation** (or dynamic program equation):
  \[
  V_t(S_t) = \max_{a_t: \text{actions}} \mathbb{E} \left[ R_t(S_t, a_t) + V_{t+1}(S_{t+1}(S_t, a_t)) \right]
  \]
  **optimal policy**: pick any \( a_t \) that is a maximizer of above eqn
example: distributing food to soup kitchens

- mobile food pantry has $C$ meals to distribute between $H$ soup kitchens
- kitchen $i$ has demand $D_i \sim F_i$ ($F_i$ is known)
- can choose to give $X_i \geq 0$ units of food
- **objective**: maximize sum of log fill ratios $\sum_{i=1}^{H} \log \left( \frac{X_i}{D_i} \right)$

- Check: If $D_1 = D_2 = \ldots = D_H > C/H$
  - optimal $X_i = C/H$

**State**
- $S_t = C_t = \text{Amount of food left for } [t, t+1, \ldots, H]$
- $A_t = X_t = \text{Amount } i \mapsto \text{ given to location } i$

$$V_t(C_t) = \max_{X_t: X_t \in [0, C_t]} \mathbb{E} \left[ \log \left( \min \left( \frac{X_t}{D_t}, 1 \right) \right) + V_{t+1}(C_t - X_t) \right]$$
'Solution' - Threshold $\Theta_t$ s.t. $X_t = \min(D_t, C_t, \Theta_t)$

$R_t(C_t, X_t) + V_{t+1}(C_t - X_t)$
example: distributing food to soup kitchens

- mobile food pantry has $C_j$ cans of item $j \in \{1, 2, \ldots, d\}$ to distribute between $H$ soup kitchens
- kitchen $i$ has demand $D_{ij} \sim F_i$ for item $j$
- can choose to give $X_{ij} \geq 0$ units of each item
- objective: maximize product of utilities $\prod_{i=1}^{H} \left( U_i \left( \sum_{j} v_{ij} \frac{X_{ij}}{D_{ij}} \right) \right) \approx \sum_{j=1}^{C_j} \log \left( U_i \left( X_{ij}, D_{ij} \right) \right)$

\begin{align*}
\text{If } j &= 1 \\
C &
\end{align*}

\begin{align*}
\text{If } j &= 2 \\
C_1 &
\end{align*}

\begin{align*}
\text{For general } j = C_1 C_2 \ldots C_j H \\
\text{complexity } &\approx C_1 C_2 \ldots C_j H
\end{align*}
‘solving’ real MDPs

- exact solution via DP
  - newsvendor problem, selling single item (‘convexity’)
  - ‘index’ policies (greedy policies) - Gitlin’s index

- approximate methods (Thompson sampling)
  - Expected improvement / KG for Bayesian Opt

- iterative methods (value/policy iteration, Q learning)
  - approximate $V^*(s)$ (or $Q^*(s)$) via same iteration
  - Q-learning (more generally, RL) - solve the MDP approx. without knowing $R$, transitions
example: the **multi-armed bandit** problem

- $K$ actions, $H$ horizon
- action $a \in [K]$ has reward $R(a) = Ber(\theta_a)$, with unknown $\theta$
- aim: maximize $\sum_{t=1}^{H} R(A_t)$

---

**Q:** If you know $\Theta_a$, what is your policy?

**A:** pick highest $\theta_a$

- **Exploration vs. Exploitation**

- Examples of "bad" policies:
  - Equal play, fix arm
  - play each arm $N$ times, for remaining $H-3N$, pick arm with highest MLE for $\Theta_a$

- These perform badly ($\Theta_1, H - E[\text{Reward}] = \text{Regret} \approx cH$)
example: the **multi-armed bandit** problem  

**Idea** - Assume $\Theta_a \sim \text{Beta}(1, 1)$

- Choose $A_t$ via some rule
- Update posterior $\Theta_a \sim \text{Beta}(1 + S_a, 1 + F_a)$

---

**Fact 1** - If $H \sim \text{Geom}(\theta)$ then optimal solution for the MDP is known (Gittin's index)

**Fact 2** - For fixed $H$, if we sample $\Theta_a \sim \text{Beta}(1 + S_a, 1 + F_a)$ and pick $A_t = \arg\max \{\Theta_a\}$ then $\mathbb{E}[\text{Regret}] \approx c K \log H$