Regret Minimization and Posterior Thompson Sampling

Monday Morning Session, CDT Summer School

Outline

- Regret
- Intuition behind various learning algorithms
    - Multi-arm bandits
    - Contextual bandits
    - Markov decision process
- Some details of Bayesian posterior sampling
- Coding

Regret of a learning algorithm

$$\text{Regret} = \max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$$

Sum of rewards that the optimal policy π can get

Sum of rewards that the learning algorithm gets

So, regret is the excess reward that the algorithm misses compared to the best possible policy in a given class

Question: Does the optimal quantity depend on the learning algorithm? What other things does it depend on?

$$\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}]$$

Question: Does the optimal quantity depend on the learning algorithm? What other things does it depend on?

$$\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}]$$

The nature of underlying environment – this object does not depend on the learning algorithm.

Regret, optimal policy, and learning algorithms for two-arm bandits

Optimal policy: Multi-arm bandits (2-arms)

- Two potential outcomes $\{R_{t+1}(1), R_{t+1}(0)\}$ drawn from some distribution $\mathscr{P}$.

We assume stationarity across time

- At time t, take action A_t and receive outcome R_{t+1}.

- Consistency in observed outcomes, i.e., $R_{t+1} = R_{t+1}(A_t)$.

- Observed data: $(A_1, R_2), (A_2, R_3), (A_3, R_4), \ldots, (A_T, R_{T+1})$

Optimal policy: Multi-arm bandits (2-arms)

- Two potential outcomes $\{R_{t+1}(1), R_{t+1}(0)\}$ drawn from some distribution $\mathscr{P}$.
- At time t, take action A_t and receive outcome R_{t+1}.
- Consistency in observed outcomes, i.e., $R_{t+1} = R_{t+1}(A_t)$.
- Observed data: $(A_1, R_2), (A_2, R_3), (A_3, R_4), \ldots, (A_T, R_{T+1})$

We assume stationarity across time

- Regret minimization: Suffices to search over deterministic stationary policies
- Let $r(a) \triangleq \mathbb{E}[R_{j+1}(a)]$ be the stationary mean potential outcome, then

$$\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] = \Sigma_{j=1}^{T} r(a^\star) \text{ where } a^\star = \arg\max_{a} r(a)$$

Exercise: Prove!

Learning optimal policy: 2-arm bandits

- Regret = $\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$ = $T \max(r(1), r(0)) - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$

- Regret would be minimized, if we knew the maximum reward or optimal policy

Learning optimal policy: 2-arm bandits

- Regret = $\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}] = T \max(r(1), r(0)) - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$

- Regret would be minimized, if we knew the maximum reward or optimal policy

- **Question**: Knowledge of what quantity suffices to learn the optimal policy?

    - $r(1)$ and $r(0)$?

Learning optimal policy: 2-arm bandits

- Regret = $\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$ = $T \max(r(1), r(0)) - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$
- Regret would be minimized, if we knew the maximum reward or optimal policy
- **Question**: Knowledge of what quantity suffices to learn the optimal policy?
    - $r(1)$ and $r(0)$?
    - $r(1) - r(0)$?

Learning optimal policy: 2-arm bandits

- Regret = $\max_\pi \mathbb{E}_\pi[\Sigma_{j=1}^T R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^T R_{j+1}] = T \max(r(1), r(0)) - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^T R_{j+1}]$

- Regret would be minimized, if we knew the maximum reward or optimal policy

- **Question**: Knowledge of what quantity suffices to learn the optimal policy?

    - $r(1)$ and $r(0)$?

    - $r(1) - r(0)$?

    - sign$(r(1) - r(0))$?

Learning optimal policy: 2-arm bandits

- Regret = $\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}] = T \max(r(1), r(0)) - \mathbb{E}_{\pi^L}[\Sigma_{j=1}^{T} R_{j+1}]$
- Regret would be minimized, if we knew the maximum reward or optimal policy
- **Question**: Knowledge of what quantity suffices to learn the optimal policy?
    - $r(1)$ and $r(0)$?
    - $r(1) - r(0)$?
    - sign$(r(1) - r(0))$?
  - Is there an advantage of one term over the other?

Typical bandit algorithm has two components (2-arm): These components may be explicit or implicit

1. **Learning algorithm:**

    - Uses the data so far to produce estimates $\hat{r}(1)$ and $\hat{r}(0)$ with uncertainty $\hat{\sigma}(1)$ and $\hat{\sigma}(0)$ for these quantities

2. **Optimization algorithm:**

    - Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

Typical bandit algorithm has two components (2-arm): These components may be explicit or implicit

1. **Learning algorithm:**
   - Uses the data so far to produce estimates $\hat{r}(1)$ and $\hat{r}(0)$ with uncertainty $\hat{\sigma}(1)$ and $\hat{\sigma}(0)$ for these quantities
   - where, *typically* $\hat{r}(a) \rightarrow r(a)$ and $\hat{\sigma}(a) \rightarrow 0$ as the number of pulls of arm $a \rightarrow \infty$
2. **Optimization algorithm:**
   - Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

Typical bandit algorithm has two components (2-arm): These components may be explicit or implicit

1. **Learning algorithm:**
    - Uses the data so far to produce estimates $\hat{r}(1)$ and $\hat{r}(0)$ with uncertainty $\hat{\sigma}(1)$ and $\hat{\sigma}(0)$ for these quantities
    - where, <u>*typically*</u> $\hat{r}(a) \to r(a)$ and $\hat{\sigma}(a) \to 0$ as the number of pulls of arm $a \to \infty$
2. **Optimization algorithm:**
    - Uses the estimates to construct a “learning” policy that pulls the arm in the next iteration
    - Typically, the policy is a “good guess” of the optimal policy based on the data so far

Typical bandit algorithm has two components (2-arm): These components may be explicit or implicit

1. **Learning algorithm:**
    - Uses the data so far to produce estimates $\hat{r}(1)$ and $\hat{r}(0)$ with uncertainty $\hat{\sigma}(1)$ and $\hat{\sigma}(0)$ for these quantities
    - where, *typically* $\hat{r}(a) \to r(a)$ and $\hat{\sigma}(a) \to 0$ as the number of pulls of arm $a \to \infty$
2. **Optimization algorithm:**
    - Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration
    - Typically, the policy is a "good guess" of the optimal policy based on the data so far

- The data collected by executing the learning policy is then fed into the learning algorithm to produce next set of estimates

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a “learning” policy that pulls the arm in the next iteration

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a “learning” policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$

Is this a good choice?

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

Is this a good choice?

➡ $\pi^L = \arg\max_a \hat{r}(a)$

This is the optimal policy if $\hat{r} = r$

What can go wrong when $\hat{r} \neq r$

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$

This is the optimal policy if $\hat{r} = r$

➡ $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$ — This is the optimal policy if $\hat{r} = r$

➡ $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$ — Rationale behind Upper Confidence Bound

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:
Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

- $\pi^L = \arg\max_a \hat{r}(a)$ — This is the optimal policy if $\hat{r} = r$
- $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$ — Rationale behind Upper Confidence Bound
- $\pi^L = \begin{cases} \arg\max_a \hat{r}(a) & \text{with high probability} \\ \text{the other action} & \text{with low probability} \end{cases}$

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a “learning” policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$ This is the optimal policy if $\hat{r} = r$

➡ $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$ Rationale behind Upper Confidence Bound

➡ $\pi^L = \begin{cases} \arg\max_a \hat{r}(a) & \text{with high probability} \\ \text{the other action} & \text{with low probability} \end{cases}$ Rationale behind $\varepsilon-$greedy

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:
Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$ — This is the optimal policy if $\hat{r} = r$

➡ $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$ — Rationale behind Upper Confidence Bound

➡ $\pi^L = \begin{cases} \arg\max_a \hat{r}(a) & \text{with high probability} \\ \text{the other action} & \text{with low probability} \end{cases}$ — Rationale behind $\varepsilon-$greedy

➡ If $Z(a) \sim \mathcal{N}(\hat{r}(a), \hat{\sigma}(a))$ denotes a posterior on $r(a)$,

$$\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(1) > Z(0)) \\ 0 & \text{otherwise}\,. \end{cases}$$

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:
Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

➡ $\pi^L = \arg\max_a \hat{r}(a)$ — This is the optimal policy if $\hat{r} = r$

➡ $\pi^L = \arg\max_a \hat{r}(a) + c\hat{\sigma}(a)$ — Rationale behind Upper Confidence Bound

➡ $\pi^L = \begin{cases} \arg\max_a \hat{r}(a) & \text{with high probability} \\ \text{the other action} & \text{with low probability} \end{cases}$ — Rationale behind ε−greedy

➡ If $Z(a) \sim \mathcal{N}(\hat{r}(a), \hat{\sigma}(a))$ denotes a posterior on $r(a)$,

$$\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(1) > Z(0)) \\ 0 & \text{otherwise}\,. \end{cases}$$

Rationale behind Posterior/Thompson sampling

Discussion questions

- What would happen to the learning policies in the bandit algorithms if one arm has significantly higher mean reward than the other? Which variant of algorithms would you prefer if the two arms have similar means?

- There are two ways to implement posterior sampling:

  - Sample action 1 with probability $p \triangleq \mathbb{P}\left[\mathcal{N}\left(\hat{r}(1) - \hat{r}(0), \hat{\sigma}_1^2 + \hat{\sigma}_2^2\right) > 0\right]$ and action 0 with $1 - p$.

  - Sample two random variables $Z(a) \sim \mathcal{N}(\hat{r}(a), \hat{\sigma}^2(a))$ and assign $\arg\max_a Z(a)$

  - Are both approaches correct? Is there a benefit of one approach over the other?

  - **Hint**: In which case do we have explicit probabilities of randomization? What happens when we don't have Gaussian posterior?

Same ideas applied to contextual bandits

Contextual bandits

- State, two potential outcomes $\{S_t, R_{t+1}(1), R_{t+1}(0)\}$ drawn from some distribution $\mathscr{P}$
- At time t, take action A_t and receive outcome R_{t+1}
- Consistency in observed outcomes, i.e., $R_{t+1} = R_{t+1}(A_t)$
- Observed data: $(S_1, A_1, R_2), (S_2, A_2, R_3), (S_3, A_3, R_4), \ldots, (S_T, A_T, R_{T+1})$

At each time, a state/context variable is observed (still assuming stationarity across time points)

Contextual bandits

- State, two potential outcomes $\{S_t, R_{t+1}(1), R_{t+1}(0)\}$ drawn from some distribution $\mathscr{P}$
- At time t, take action A_t and receive outcome R_{t+1}
- Consistency in observed outcomes, i.e., $R_{t+1} = R_{t+1}(A_t)$
- Observed data: $(S_1, A_1, R_2), (S_2, A_2, R_3), (S_3, A_3, R_4), \ldots, (S_T, A_T, R_{T+1})$
- Regret minimization: Would like to use the states, so the policy class is $\{\pi : \mathscr{S} \to \boldsymbol{\Delta}^{|A|-1}\}$
- Let $r(s, a) \triangleq \mathbb{E}[R_{j+1}(a) \,|\, S_j = s]$ be the stationary mean reward outcome, then

At each time, a state/context variable is observed (still assuming stationarity across time points)

$$\max_{\pi} \mathbb{E}_{\pi}[\Sigma_{j=1}^{T} R_{j+1}] = \mathbb{E}_{\pi^\star}[\Sigma_{j=1}^{T} r(S_j, \pi^\star(S_j))] \text{ where } \pi^\star(s) = \arg\max_{a} r(s, a)$$

Exercise: Prove!

Typical "contextual" bandit algorithm has two components (2-arm): These components may be explicit or implicit

1. **Learning algorithm:**

    - Uses the data so far to produce estimates $\hat{r}(s,1)$ and $\hat{r}(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$ for these quantities

    - where, <u>*typically*</u> $\hat{r}(s,a) \to r(s,a)$ and $\hat{\sigma}(s,a) \to 0$ as the number of pulls of arm a under state $s \to \infty$

2. **Optimization algorithm:**

    - Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

    - Typically, the policy is a "good guess" of the optimal policy based on the data so far

- The data collected by executing the learning policy is then fed into the learning algorithm to produce next set of estimates

Typical "contextual" bandit algorithm has two components (2-arm): These components may be explicit or implicit

Typical approach: Model $r(s,a) = \beta_a^\top \phi(s,a)$ and estimate β_a

1. **Learning algorithm:**
   - Uses the data so far to produce estimates $\hat{r}(s,1)$ and $\hat{r}(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$ for these quantities
   - where, *typically* $\hat{r}(s,a) \to r(s,a)$ and $\hat{\sigma}(s,a) \to 0$ as the number of pulls of arm a under state $s \to \infty$
2. **Optimization algorithm:**
   - Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration
   - Typically, the policy is a "good guess" of the optimal policy based on the data so far

- The data collected by executing the learning policy is then fed into the learning algorithm to produce next set of estimates

Different principled heuristics in the optimization algorithm lead to different algorithms

Optimization algorithm:

Uses the estimates to construct a "learning" policy that pulls the arm in the next iteration

➡ $\pi^L(s) = \arg\max_a \hat{r}(s,a)$

This is the optimal policy if $\hat{r} = r$

➡ $\pi^L(s) = \arg\max_a \hat{r}(s,a) + c\hat{\sigma}(s,a)$

Rationale behind Upper Confidence Bound

➡ $\pi^L = \begin{cases} \arg\max_a \hat{r}(s,a) & \text{with high probability} \\ \text{the other action} & \text{with low probability} \end{cases}$

Rationale behind $\varepsilon-$greedy

➡ If $Z(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(a)$,

$\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(s,1) > Z(s,0)) \\ 0 & \text{otherwise}. \end{cases}$

Rationale behind Posterior/Thompson sampling

Moving on to MDPs

Markov Decision process

- Sequence of states $\{S_0, S_1(1), S_1(0), S_2(1), S_2(0), \ldots\}$ drawn from some distribution $\mathscr{P}$, satisfying Markovian property

Markov Decision process

- Sequence of states $\{S_0, S_1(1), S_1(0), S_2(1), S_2(0), \ldots\}$ drawn from some distribution $\mathscr{P}$, satisfying Markovian property

Assuming stationarity in the transitions between states

Markov Decision process

- Sequence of states $\{S_0, S_1(1), S_1(0), S_2(1), S_2(0), \ldots\}$ drawn from some distribution $\mathscr{P}$, satisfying Markovian property

Assuming stationarity in the transitions between states

- At time $t = 0,1,\ldots,$ take action A_t, observe S_{t+1}, and receive outcome R_{t+1}.

- Consistency in observed states, i.e., $S_{t+1} = S_{t+1}(A_t)$ and reward $R_{t+1} = h(S_{t+1}) = h(S_{t+1}(A_t))$.

- Observed data: $(S_1, A_1, R_2), (S_2, A_2, R_3), (S_3, A_3, R_4), \ldots,$

Markov Decision process

- Sequence of states $\{S_0, S_1(1), S_1(0), S_2(1), S_2(0), \ldots\}$ drawn from some distribution $\mathscr{P}$, satisfying Markovian property

Assuming stationarity in the transitions between states

- At time $t = 0,1,\ldots,$ take action A_t, observe S_{t+1}, and receive outcome R_{t+1}.
- Consistency in observed states, i.e., $S_{t+1} = S_{t+1}(A_t)$ and reward $R_{t+1} = h(S_{t+1}) = h(S_{t+1}(A_t))$.
- Observed data: $(S_1, A_1, R_2), (S_2, A_2, R_3), (S_3, A_3, R_4), \ldots,$
- Now the benchmark is discounted sum of rewards $\mathbb{E}_\pi[\Sigma_{j=0}^{\infty}\gamma^j R_{j+1}]$
- Would like to use the states, so the policy class is $\{\pi : \mathcal{S} \to \mathbf{\Delta}^{|A|-1}\}$.
    - Here the policy class is restricted based on the fact that under stationary transitions, the optimal policy is stationary

Unpacking the optimal policy for Markov decision process

- Let $r(s, a) \triangleq \mathbb{E}[R_{j+1} \,|\, S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \,|\, S_j = s]$

- Suppose initial state is deterministically s, then

  $\mathbb{E}_\pi[\Sigma_{j=0}^{\infty} \gamma^j R_{j+1}]$ =

Unpacking the optimal policy for Markov decision process

- Let $r(s, a) \triangleq \mathbb{E}[R_{j+1} \,|\, S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \,|\, S_j = s]$

- Suppose initial state is deterministically s, then

$$\mathbb{E}_\pi[\Sigma_{j=0}^{\infty}\gamma^j R_{j+1}] = \Sigma_a \, \pi(a \,|\, s)\Big[r(s, a) + \underbrace{\gamma \sum_{j=1}^{\infty} \mathbb{E}[\gamma^{j-1} R_{j+1} \,|\, S_0 = s, A_0 = a]}_{\triangleq H^\pi(s,a)}\Big]$$

Unpacking the optimal policy for Markov decision process

- Let $r(s,a) \triangleq \mathbb{E}[R_{j+1} \,|\, S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \,|\, S_j = s]$

- Suppose initial state is deterministically s, then

$$\mathbb{E}_\pi[\Sigma_{j=0}^{\infty} \gamma^j R_{j+1}] = \Sigma_a \, \pi(a \,|\, s)\Big[r(s,a) + \gamma \underbrace{\sum_{j=1}^{\infty} \mathbb{E}[\gamma^{j-1} R_{j+1} \,|\, S_0 = s, A_0 = a]}_{\triangleq H^\pi(s,a)}\Big]$$

$$= \Sigma_a \, \pi(a \,|\, s)\big[r(s,a) + \gamma H^\pi(s,a)\big]$$

$$= \Sigma_a \, \pi(a \,|\, s)\big[Q^\pi(s,a)\big]$$

Unpacking the optimal policy for Markov decision process

- Let $r(s, a) \triangleq \mathbb{E}[R_{j+1} \,|\, S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \,|\, S_j = s]$

- Suppose initial state is deterministically s, then

$$\mathbb{E}_\pi[\Sigma_{j=0}^{\infty} \gamma^j R_{j+1}] = \Sigma_a \, \pi(a \,|\, s)\Big[r(s, a) + \gamma \underbrace{\sum_{j=1}^{\infty} \mathbb{E}[\gamma^{j-1} R_{j+1} \,|\, S_0 = s, A_0 = a]}_{\triangleq H^\pi(s,a)}\Big]$$

$$= \Sigma_a \, \pi(a \,|\, s)\big[r(s, a) + \gamma H^\pi(s, a)\big]$$

$$= \Sigma_a \, \pi(a \,|\, s)\big[Q^\pi(s, a)\big]$$

- Hence the optimal policy $\pi^\star$ satisfies

Unpacking the optimal policy for Markov decision process

- Let $r(s, a) \triangleq \mathbb{E}[R_{j+1} \,|\, S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \,|\, S_j = s]$

- Suppose initial state is deterministically s, then

$$\mathbb{E}_\pi[\Sigma_{j=0}^{\infty} \gamma^j R_{j+1}] = \Sigma_a \pi(a \,|\, s) \Big[r(s, a) + \gamma \underbrace{\sum_{j=1}^{\infty} \mathbb{E}[\gamma^{j-1} R_{j+1} \,|\, S_0 = s, A_0 = a]}_{\triangleq H^\pi(s,a)} \Big]$$

$$= \Sigma_a \pi(a \,|\, s) \Big[r(s, a) + \gamma H^\pi(s, a) \Big]$$

$$= \Sigma_a \pi(a \,|\, s) \Big[Q^\pi(s, a) \Big]$$

- Hence the optimal policy $\pi^\star$ satisfies

$$\pi^\star(a \,|\, s) = \arg\max_a Q^{\pi^\star}(s, a) = \arg\max_a r(s, a) + H^{\pi^\star}(s, a)$$

Exercise: Prove!

Unpacking the optimal policy for Markov decision process

- Let $r(s,a) \triangleq \mathbb{E}[R_{j+1} \mid S_j = s, A_j = a] = \mathbb{E}[h(S_{j+1}(a)) \mid S_j = s]$

- Suppose initial state is deterministically s, then

$$\mathbb{E}_\pi[\Sigma_{j=0}^{\infty} \gamma^j R_{j+1}] = \Sigma_a \pi(a \mid s)\Big[r(s,a) + \gamma \underbrace{\sum_{j=1}^{\infty} \mathbb{E}[\gamma^{j-1} R_{j+1} \mid S_0 = s, A_0 = a]}_{\triangleq H^\pi(s,a)}\Big]$$

$$= \Sigma_a \pi(a \mid s)\big[r(s,a) + \gamma H^\pi(s,a)\big]$$

$$= \Sigma_a \pi(a \mid s)\big[Q^\pi(s,a)\big]$$

- Hence the optimal policy $\pi^\star$ satisfies

$$\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s,a) \quad = \arg\max_a r(s,a) + H^{\pi^\star}(s,a)$$

Exercise: Prove!

- **Typical goal in MDP:** Find this optimal policy!

Key ideas behind Q-learning

- **Question**: Suppose someone gave us the values of $Q^{\pi^\star}$ but not $\pi^\star$. How do we recover $\pi^\star$?

Key ideas behind Q-learning

- **Question**: Suppose someone gave us the values of $Q^{\pi^\star}$ but not $\pi^\star$. How do we recover $\pi^\star$?
    - $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s, a)$

Key ideas behind Q-learning

- **Question**: Suppose someone gave us the values of $Q^{\pi^\star}$ but not $\pi^\star$. How do we recover $\pi^\star$?
  - $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s, a)$
- **Question:** What if we had an estimate $\hat{Q}^{\pi^\star}$ with uncertainty $\hat{\sigma}$?

One class of RL algorithms for MDPs

1. **Learning algorithm:**

    - Uses the data so far to produce estimates $\hat{Q}^{\pi^\star}r(s,1)$ and $\hat{Q}^{\pi^\star}r(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$ for these quantities

    - where, <u>*typically*</u> $\hat{Q}^{\pi^\star}(s,a) \to Q^{\pi^\star}(s,a)$ and $\hat{\sigma}(s,a) \to 0$ as the number of pulls of action a under state $s \to \infty$

2. **Optimization algorithm:**

    - Uses the estimates to construct a "learning" policy that samples the action for the next iteration

    - $\pi^L(s) = \arg\max_a \hat{Q}^{\pi^\star}(s,a) + c\hat{\sigma}(s,a)$

    - If $Z(s,a) \sim \mathcal{N}(\hat{Q}^{\pi^\star}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $Q^{\pi^\star}(s,a)$,
    $$\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(s,1) > Z(s,0)) \\ 0 & \text{otherwise} . \end{cases}$$

A simpler RL algorithm class with a different target

- Recall $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s, a) = \arg\max_a r(s, a) + H^{\pi^\star}(s, a)$. Suppose someone gave us $H^{\pi^\star}$ but not r. How do we estimate $\pi^\star$?

A simpler RL algorithm class with a different target

- Recall $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s, a) = \arg\max_a r(s, a) + H^{\pi^\star}(s, a)$. Suppose someone gave us $H^{\pi^\star}$ but not r. How do we estimate $\pi^\star$?

- **Learning algorithm:** Produce estimates $\hat{r}(s,1)$ and $\hat{r}(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$.

A simpler RL algorithm class with a different target

- Recall $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s,a) = \arg\max_a r(s,a) + H^{\pi^\star}(s,a)$. Suppose someone gave us $H^{\pi^\star}$ but not r. How do we estimate $\pi^\star$?

- **Learning algorithm:** Produce estimates $\hat{r}(s,1)$ and $\hat{r}(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$.

- **Optimization algorithm:** Uses the estimates to construct a policy.

  - $\pi^L = \arg\max_a \hat{r}(s,a) + c\hat{\sigma}(s,a) + \gamma H^{\pi^\star}(s,a)$

  - If $Z'(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,

$$\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}\left[Z'(s,1) - Z'(s,0) > -\gamma\left[H^{\pi^\star}(s,1) - H^{\pi^\star}(s,0)\right]\right] \\ 0 & \text{otherwise}\,. \end{cases}$$

A simpler RL algorithm class with a different target

- Recall $\pi^\star(a \mid s) = \arg\max_a Q^{\pi^\star}(s,a) = \arg\max_a r(s,a) + H^{\pi^\star}(s,a)$. Suppose someone gave us $H^{\pi^\star}$ but not r. How do we estimate $\pi^\star$?
- **Learning algorithm:** Produce estimates $\hat{r}(s,1)$ and $\hat{r}(s,0)$ with uncertainty $\hat{\sigma}(s,1)$ and $\hat{\sigma}(s,0)$.
- **Optimization algorithm:** Uses the estimates to construct a policy.
  - $\pi^L = \arg\max_a \hat{r}(s,a) + c\hat{\sigma}(s,a) + \gamma H^{\pi^\star}(s,a)$
  - If $Z'(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,

$$\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}\left[Z'(s,1) - Z'(s,0) > -\gamma\left[H^{\pi^\star}(s,1) - H^{\pi^\star}(s,0)\right]\right] \\ 0 & \text{otherwise}\,. \end{cases}$$

Discuss: Why is this a simpler algorithm class?

Steps towards the HeartSteps RL algorithm

Action 1 = **Send** a notification vs Action 0 = **Do nothing**

If $Z'(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,

$$\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}\left[Z'(s,1) - Z'(s,0) > -\gamma\left[H^{\pi^\star}(s,1) - H^{\pi^\star}(s,0)\right]\right] \\ 0 & \text{otherwise}\,. \end{cases}$$

HeartSteps further clips the probabilities between 0.2 and 0.8

Discuss:
Why does this make intuitive sense?
What are the quantities on the LHS and RHS?
Why is this a simpler algorithm class compared to estimating $Q^{\pi^\star}$?

This is the HeartSteps RL algorithm

Action 1 = **Send** a notification vs Action 0 = **Do nothing**

If $Z'(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,

$$\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}\left[Z'(s,1) - Z'(s,0) > -\gamma\left[H^{\pi^\star}(s,1) - H^{\pi^\star}(s,0)\right] \right] \\ 0 & \text{otherwise}\,. \end{cases}$$

LHS is like the immediate treatment effect of sending a notification

RHS is like the delayed effects (denoted as $\eta(s)$)

This is the HeartSteps RL algorithm

Action 1 = **Send** a notification vs Action 0 = **Do nothing**

If $Z'(s, a) \sim \mathcal{N}(\hat{r}(s, a), \hat{\sigma}(s, a))$ denotes a posterior on $r(s, a)$,

$$\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}\left[Z'(s,1) - Z'(s,0) > -\gamma\left[H^{\pi^\star}(s,1) - H^{\pi^\star}(s,0)\right] \right] \\ 0 & \text{otherwise} . \end{cases}$$

LHS is like the immediate treatment effect of sending a notification

RHS is like the delayed effects (denoted as $\eta(s)$)

HeartSteps estimates the RHS and then clips the probabilities between 0.2 and 0.8

Bayesian Thompson Sampling with Gaussian linear model

Bayesian Thompson sampling with linear model

- We didn't discuss so far, how the estimate of $\hat{r}$ is generated
- Model: $r(s, a) \sim \alpha_a^\top s + \varepsilon$ for $a \in \{0,1\}$ where ε is Gaussian noise
    - can choose non-linear model, like $\alpha_a^\top \phi(s, a)$
    - or non-Gaussian noise
- Or we can also model: $r(s, a) \sim \alpha^\top g(s) + a\beta^\top f(s) + \varepsilon$ –> HeartSteps

Bayesian Thompson sampling for 2-arm bandits

- Parameters unknown, put a Gaussian prior on them
- Gaussian prior + Gaussian likelihood –> Gaussian posterior
- We put a prior $\mathcal{N}(\bar{\alpha}_a, \bar{\sigma}_a^2)$ on $r(a)$, and observe (A_1, R_2). What are the posteriors?

Bayesian Thompson sampling for 2-arm bandits

- Parameters unknown, put a Gaussian prior on them
- Gaussian prior + Gaussian likelihood –> Gaussian posterior
- We put a prior $\mathcal{N}(\bar{\alpha}_a, \bar{\sigma}_a^2)$ on $r(a)$, and observe (A_1, R_2). What are the posteriors?
    - $\tilde{\sigma}_a^2 = \left(\frac{1}{\bar{\sigma}_a^2} + \frac{\mathbf{1}(A_1 = a)}{\sigma^2} \right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\sigma}_a^2 \left(\frac{\alpha_a}{\sigma_a^2} + \frac{\mathbf{1}(A_1 = a) R_2}{\sigma^2} \right)$

Bayesian Thompson sampling for 2-arm bandits

- Parameters unknown, put a Gaussian prior on them
- Gaussian prior + Gaussian likelihood –> Gaussian posterior
- We put a prior $\mathcal{N}(\bar{\alpha}_a, \bar{\sigma}_a^2)$ on $r(a)$, and observe (A_1, R_2). What are the posteriors?
  - $\tilde{\sigma}_a^2 = \left(\frac{1}{\bar{\sigma}_a^2} + \frac{\mathbf{1}(A_1 = a)}{\sigma^2}\right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\sigma}_a^2 \left(\frac{\alpha_a}{\sigma_a^2} + \frac{\mathbf{1}(A_1 = a)R_2}{\sigma^2}\right)$
- Now what happens after observing T tuples? Can use induction with – posterior being the new prior

Bayesian Thompson sampling for 2-arm bandits

- Parameters unknown, put a Gaussian prior on them
- Gaussian prior + Gaussian likelihood –> Gaussian posterior
- We put a prior $\mathcal{N}(\bar{\alpha}_a, \bar{\sigma}_a^2)$ on $r(a)$, and observe (A_1, R_2). What are the posteriors?
    - $\tilde{\sigma}_a^2 = \left(\frac{1}{\bar{\sigma}_a^2} + \frac{\mathbf{1}(A_1 = a)}{\sigma^2} \right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\sigma}_a^2 \left(\frac{\alpha_a}{\sigma_a^2} + \frac{\mathbf{1}(A_1 = a)R_2}{\sigma^2} \right)$
- Now what happens after observing T tuples? Can use induction with – posterior being the new prior
    - $\tilde{\sigma}_a^2 = \left(\frac{1}{\sigma_a^2} + \frac{\sum_{j=1}^{T} \mathbf{1}(A_j = a)}{\sigma^2} \right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\sigma}_a^2 \left(\frac{\alpha_a}{\sigma_a^2} + \frac{\sum_{j=1}^{T} \mathbf{1}(A_j = a)R_{j+1}}{\sigma^2} \right)$

Bayesian Thompson sampling with linear model for contextual bandits

- Suppose $r(s, a) \sim \alpha_a^\top s + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0,\sigma^2)$ and σ is known
- We put a prior $\mathcal{N}(\bar{\alpha}_a, \bar{\Sigma}_a)$ on α_a –> $\alpha_a^\top \phi(s, a) \ \sim \mathcal{N}(\bar{\alpha}_a^\top \phi(s, a), \phi(s, a)^\top \bar{\Sigma}_a \phi(s, a))$
- Observe (A_1, S_1, R_2). What are the posteriors?
- $\tilde{\Sigma}_a = \left(\bar{\Sigma}_a^{-1} + \frac{\mathbf{1}(A_1 = a)\phi(S_1, A_1)\phi(S_1, A_1)^\top}{\sigma^2} \right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\Sigma}_a \left(\bar{\Sigma}_a^{-1}\alpha_a + \frac{\mathbf{1}(A_1 = a)\phi(S_1, A_1)R_2}{\sigma^2} \right)$
- Now what happens after observing T tuples? Can use induction with – posterior being the new prior
- $\tilde{\Sigma}_a = \left(\bar{\Sigma}_a^{-1} + \frac{\sum_{j=1}^T \mathbf{1}(A_j = a)\phi(S_j, A_j)\phi(S_j, A_j)^\top}{\sigma^2} \right)^{-1}$ and $\tilde{\alpha}_a = \tilde{\Sigma}_a \left(\bar{\Sigma}_a^{-1}\alpha_a + \frac{\sum_{j=1}^T \mathbf{1}(A_j = a)\phi(S_j, A_j)R_{j+1}}{\sigma^2} \right)$

Posterior sampling

- If $Z(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,
  $\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(s,1) > Z(s,0)) \\ 0 & \text{otherwise}. \end{cases}$

- Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}((\tilde{\alpha}_1 - \tilde{\alpha}_0)^\top \phi(s,a), \tilde{\Sigma}_1 + \tilde{\Sigma}_2) > 0\right]$ – can compute this probability exactly – closed form for Gaussian case

Posterior sampling

- If $Z(s,a) \sim \mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,
  $$\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(s,1) > Z(s,0)) \\ 0 & \text{otherwise} . \end{cases}$$

- Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}((\tilde{\alpha}_1 - \tilde{\alpha}_0)^\top \phi(s,a), \tilde{\Sigma}_1 + \tilde{\Sigma}_2) > 0\right]$ – can compute this probability exactly – closed form for Gaussian case

  - Or if we model the treatment effect as $\beta^\top f(s)$, then
    Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}(\tilde{\beta}^\top f(s), \tilde{\Sigma}_\beta) > 0\right]$

Posterior sampling

- If $Z(s, a) \sim \mathcal{N}(\hat{r}(s, a), \hat{\sigma}(s, a))$ denotes a posterior on $r(s, a)$,
  $\pi^L(s) = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(Z(s,1) > Z(s,0)) \\ 0 & \text{otherwise} . \end{cases}$

- Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}((\tilde{\alpha}_1 - \tilde{\alpha}_0)^\top \phi(s, a), \tilde{\Sigma}_1 + \tilde{\Sigma}_2) > 0\right]$ – can compute this probability exactly – closed form for Gaussian case

  - Or if we model the treatment effect as $\beta^\top f(s)$, then
    Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}(\tilde{\beta}^\top f(s), \tilde{\Sigma}_\beta) > 0\right]$

  - When we also have delayed effects proxy $\eta(s)$:
    Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}(\tilde{\beta}^\top f(s), \tilde{\Sigma}_\beta) > \eta(s)\right]$

Posterior sampling

- If $\mathcal{N}(\hat{r}(s,a), \hat{\sigma}(s,a))$ denotes a posterior on $r(s,a)$,
  $\pi^L = \begin{cases} 1 & \text{with probability} \quad \mathbb{P}(r(s,1) > r(s,0)) \\ 0 & \text{otherwise}\,. \end{cases}$

- Sample α_1, α_0 from the posterior and assign action $= \arg\max_a \alpha_a^\top \phi(s,a)$
  - Or if we model the treatment effect: Sample β from the posterior and assign $a = \mathbf{1}(\beta^\top f(s) > 0)$.
  - Or if we model the treatment effect with delayed effects proxy as $\eta(s)$: Sample β from the posterior and assign $a = \mathbf{1}(\beta^\top f(s) > \eta(s))$.

Discuss

- Sample action 1 with probability $\mathbb{P}\left[\mathcal{N}((\tilde{\alpha}_1 - \tilde{\alpha}_0)^\top \phi(s,a), \tilde{\Sigma}_1 + \tilde{\Sigma}_2) > 0\right]$
- Sample α_1, α_0 from the posterior and assign action $= \arg\max_a \alpha_a^\top \phi(s,a)$
    - Which of these two approaches are correct?
    - What are the pros and cons of the two approaches?
    - **Hint**: What happens when we don't have Gaussian posterior?

Coding outline

- 2-armed bandits
    - three algorithms: UCB, eps-greedy, BTS
- 2-armed linear contextual bandits
    - Correctly specified BTS with linear model
- 2-armed tabular MDP
    - No delayed effect estimate
    - With some delayed effect estimate