

Lecture 9: Exploration and Exploitation

David Silver

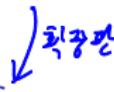
Outline

1 Introduction

2 Multi-Armed Bandits

3 Contextual Bandits

4 MDPs



Exploration vs. Exploitation Dilemma

Exploration vs. Exploitation

- Online decision-making involves a fundamental choice:
 - (Exploitation) Make the best decision given current information
 - (Exploration) Gather more information
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Examples

- Restaurant Selection
 - Exploitation Go to your favourite restaurant
 - Exploration Try a new restaurant
- Online Banner Advertisements
 - Exploitation Show the most successful advert
 - Exploration Show a different advert
- Oil Drilling
 - Exploitation Drill at the best known location
 - Exploration Drill at a new location
- Game Playing
 - Exploitation Play the move you believe is best
 - Exploration Play an experimental move

Principles

▢ Naive Exploration (*Random exploration*)

- Add noise to greedy policy (e.g. ϵ -greedy)

▢ Optimistic Initialisation

- Assume the best until proven otherwise

▢ Optimism in the Face of Uncertainty

- Prefer actions with uncertain values

▢ Probability Matching

- Select actions according to probability they are best

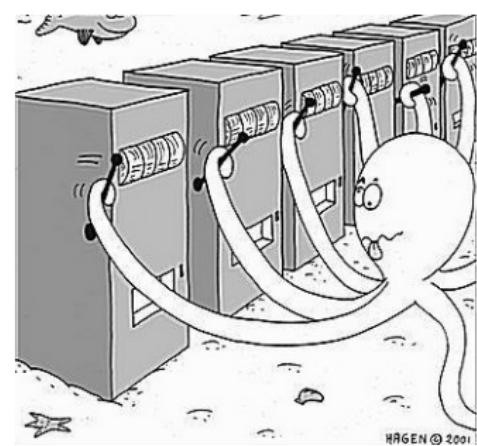
▢ Information State Search

- Lookahead search incorporating value of information

The Multi-Armed Bandit

MDP for 2-armed bandit. state t & all. One step MDP update

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- \mathcal{A} is a known set of m actions (or “arms”)
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^t r_\tau$



Regret

- The *action-value* is the mean reward for action a ,

$$Q(a) = \mathbb{E}[r|a]$$

Q - returns to you, one step up reward is yours

- The *optimal value* V^* is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The *regret* is the opportunity loss for one step

action value $I_t = \mathbb{E}[V^* - Q(a_t)]$

- The *total regret* is the total opportunity loss

$$L_t = \mathbb{E} \left[\sum_{\tau=1}^t V^* - Q(a_\tau) \right]$$

- Maximise cumulative reward \equiv minimise total regret

Counting Regret

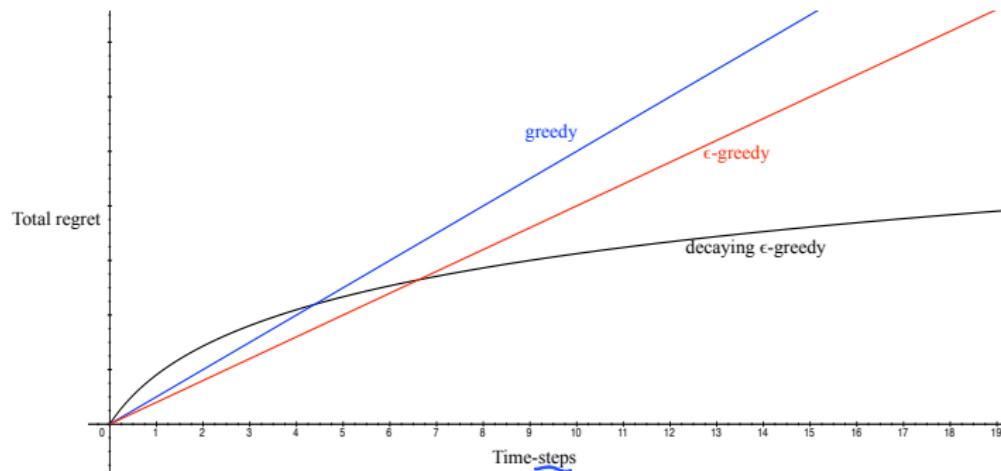
action selection

- The *count* $N_t(a)$ is expected number of selections for action a
- The *gap* Δ_a is the difference in value between action a and optimal action a^* , $\Delta_a = V^* - Q(a)$
- Regret is a function of gaps and the counts

$$\begin{aligned} L_t &= \mathbb{E} \left[\sum_{\tau=1}^t V^* - Q(a_\tau) \right] \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)] (V^* - Q(a)) \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)] \Delta_a \end{aligned}$$

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

Linear or Sublinear Regret



- If an algorithm **forever** explores it will have linear total regret
- If an algorithm **never** explores it will have linear total regret
- Is it possible to achieve sublinear total regret? **Yes!**

Greedy Algorithm

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

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

- The *greedy* algorithm selects action with highest value

$$a_t^* = \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- ⇒ Greedy has linear total regret

ϵ -Greedy Algorithm

- The ϵ -greedy algorithm continues to explore forever
 - With probability $1 - \epsilon$ select $a = \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}(a)$
 - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

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

- $\Rightarrow \epsilon$ -greedy has linear total regret

Optimistic Initialisation

a value
가장 높은 초기값으로 초기화

- Simple and practical idea: initialise $Q(a)$ to high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- Encourages systematic exploration early on 초기에는 exploration을 하는 경향
- But can still lock onto suboptimal action $\text{optimal}(a_t)$ 보다 낮은 값
- \Rightarrow greedy + optimistic initialisation has linear total regret
- \Rightarrow ϵ -greedy + optimistic initialisation has linear total regret

Decaying ϵ_t -Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \dots$
- Consider the following schedule

$$\epsilon_t = \min \left\{ 1, \frac{c|A|}{d^3 t} \right\}$$

c > 0

1. 흐름 2. 계산

repeat (시작) $d = \min_{a|\Delta_a > 0} \Delta_a$ *action* d 로 ϵ_t 를 계산해보면 ϵ_t 가 0이 되면 exploration이 끝나고, d 로 ϵ_t 를 계산해보면 ϵ_t 가 1이 되면 exploitation이 시작된다.

- Decaying ϵ_t -greedy has *logarithmic* asymptotic total regret!
- Unfortunately, schedule requires advance knowledge of gaps
- Goal: find an algorithm with *sublinear* regret for any multi-armed bandit (without knowledge of \mathcal{R})

log t 이 t 에 비해 *sublinear!*
 $\frac{\log t}{t}$ 은 t 에 비해 *sublinear!*

Lower Bound

설명

최적의 팔과 다른 팔의 확률 분포의 차이

- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have similar-looking arms with different means
- This is described formally by the gap Δ_a and the similarity in distributions $KL(\mathcal{R}^a || \mathcal{R}^{a*})$

KL divergence는 확률 분포의 차이를 0에서 1 사이로 정의합니다.

Theorem (Lai and Robbins)

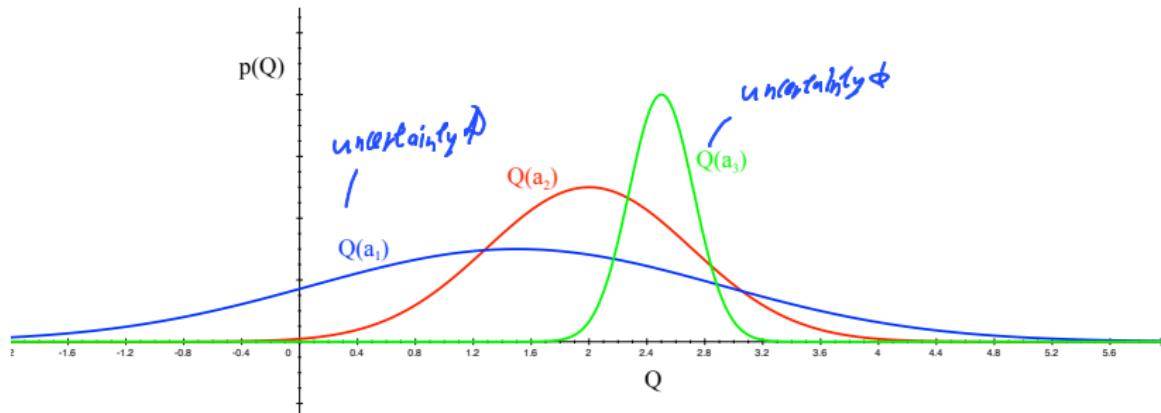
Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t \rightarrow \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a || \mathcal{R}^{a*})}$$

regret

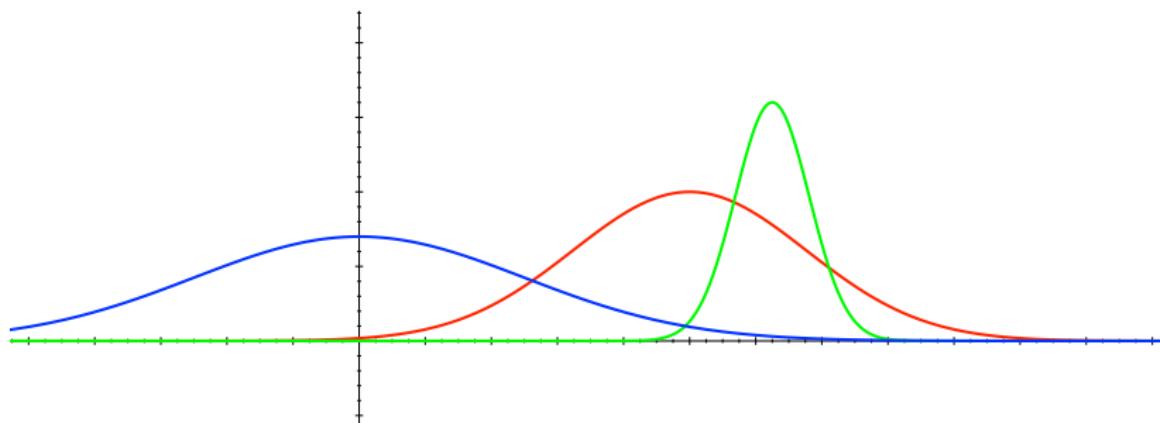
Lower Bound - 최소한의 총 후회

Optimism in the Face of Uncertainty



- Which action should we pick?
- The more uncertain we are about an action-value
- The more important it is to explore that action
- It could turn out to be the best action

Optimism in the Face of Uncertainty (2)



- After picking **blue** action
- We are less uncertain about the value
- And more likely to pick another action
- Until we home in on best action

Upper Confidence Bounds

의미

의미

- 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 $N_t(a)$ has been selected
 - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain) 의미는 확장
 - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate) 의미는 확장
- Select action maximising Upper Confidence Bound (UCB)

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

Hoeffding's Inequality

할인 할 때 쓰는 법.

Theorem (Hoeffding's Inequality)

Let X_1, \dots, X_t be i.i.d. random variables in $[0,1]$, and let

$\bar{X}_t = \frac{1}{t} \sum_{\tau=1}^t X_\tau$ be the sample mean. Then

$$\mathbb{P} [\mathbb{E}[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

ex (0, 124598) $\rightarrow [0, 1]$
 $(-\infty, \infty) \not\rightarrow$

- We will apply Hoeffding's Inequality to rewards of the bandit
- conditioned on selecting action a

UCB은 예상.

$$\mathbb{P} [Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds

- Pick a probability p that true value exceeds UCB
- Now solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Reduce p as we observe more rewards, e.g. $p = t^{-4}$
- Ensures we select optimal action as $t \rightarrow \infty$

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

UCB1

- This leads to the UCB1 algorithm

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

Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \rightarrow \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

Example: UCB vs. ϵ -Greedy On 10-armed Bandit

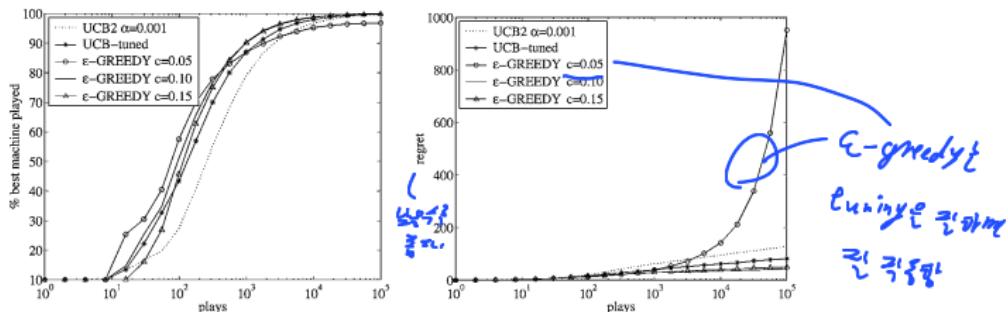


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, ..., 0.6).

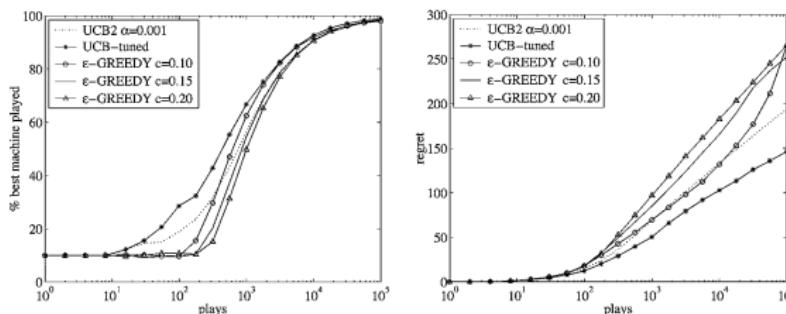


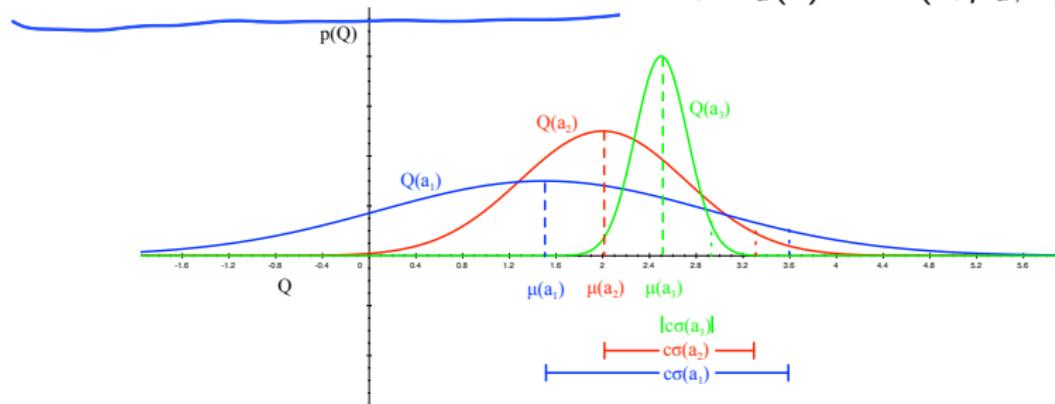
Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Bayesian Bandits

- So far we have made no assumptions about the reward distribution \mathcal{R}
 - Except bounds on rewards
- **Bayesian bandits** exploit prior knowledge of rewards, $p[\mathcal{R}]$
- They compute posterior distribution of rewards $p[\mathcal{R} | h_t]$
 - where $h_t = a_1, r_1, \dots, a_{t-1}, r_{t-1}$ is the history
- Use posterior to guide exploration
 - Upper confidence bounds (Bayesian UCB)
 - Probability matching (Thompson sampling)
- Better performance if prior knowledge is accurate

Bayesian UCB Example: Independent Gaussians

- Assume reward distribution is Gaussian, $\mathcal{R}_a(r) = \mathcal{N}(r; \mu_a, \sigma_a^2)$



- Compute Gaussian posterior over μ_a and σ_a^2 (by Bayes law)

$$p[\mu_a, \sigma_a^2 | h_t] \propto p[\mu_a, \sigma_a^2] \prod_{t \mid a_t=a} \mathcal{N}(r_t; \mu_a, \sigma_a^2)$$

- Pick action that maximises standard deviation of $Q(a)$

$$a_t = \operatorname{argmax}_a \mu_a + c\sigma_a/\sqrt{N(a)}$$

Probability Matching

- Probability matching selects action a according to probability that a is the optimal action

$$\pi(a \mid h_t) = \mathbb{P} [Q(a) > Q(a'), \forall a' \neq a \mid h_t]$$

- Probability matching is optimistic in the face of uncertainty
 - Uncertain actions have higher probability of being max
- Can be difficult to compute analytically from posterior

Thompson Sampling

probability matching vs Bayes

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(a \mid h_t) &= \mathbb{P} [Q(a) > Q(a'), \forall a' \neq a \mid h_t] \\ &= \mathbb{E}_{\mathcal{R} \mid h_t} \left[\mathbf{1}(a = \operatorname{argmax}_{a \in \mathcal{A}} Q(a)) \right]\end{aligned}$$

- Use Bayes law to compute posterior distribution $p[\mathcal{R} \mid h_t]$
- Sample a reward distribution \mathcal{R} from posterior
- Compute action-value function $Q(a) = \mathbb{E} [\mathcal{R}_a]$
- Select action maximising value on sample, $a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q(a)$
- Thompson sampling achieves Lai and Robbins lower bound!

Value of Information

- Exploration is useful because it gains information
- Can we quantify the value of information?
 - How much reward a decision-maker would be prepared to pay in order to have that information, prior to making a decision
 - Long-term reward after getting information - immediate reward
- Information gain is higher in uncertain situations
- Therefore it makes sense to explore uncertain situations more
- If we know value of information, we can trade-off exploration and exploitation *optimally*

Information State Space

state of information \tilde{s}

- We have viewed bandits as *one-step* decision-making problems
- Can also view as *sequential* decision-making problems
- At each step there is an *information state* \tilde{s}
 - \tilde{s} is a statistic of the history, $\tilde{s}_t = f(h_t)$
 - summarising all information accumulated so far
- Each action a causes a transition to a new information state \tilde{s}' (by adding information), with probability $\tilde{P}_{\tilde{s}, \tilde{s}'}^a$
- This defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

Example: Bernoulli Bandits

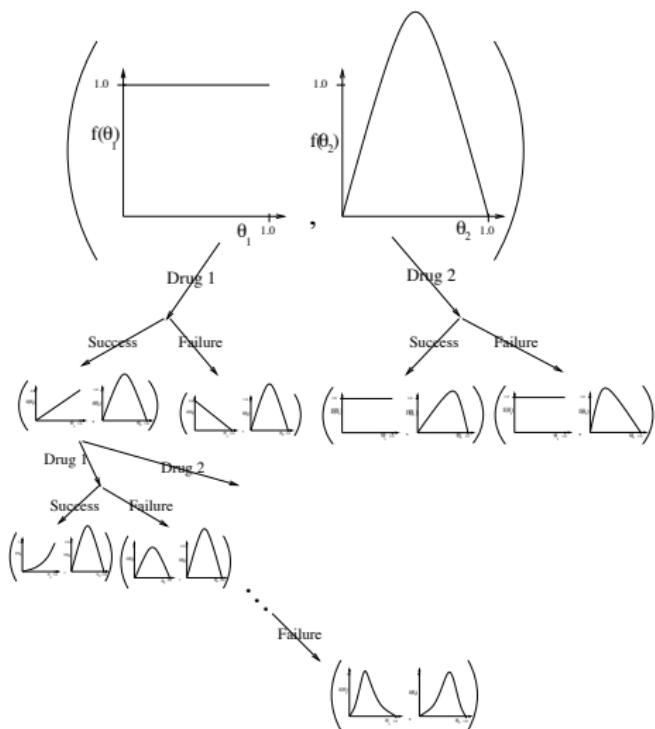
- Consider a Bernoulli bandit, such that $\mathcal{R}^a = \mathcal{B}(\mu_a)$
- e.g. Win or lose a game with probability μ_a
- Want to find which arm has the highest μ_a
- The information state is $\tilde{s} = \langle \alpha, \beta \rangle$
 - α_a counts the pulls of arm a where reward was 0
 - β_a counts the pulls of arm a where reward was 1

Solving Information State Space Bandits

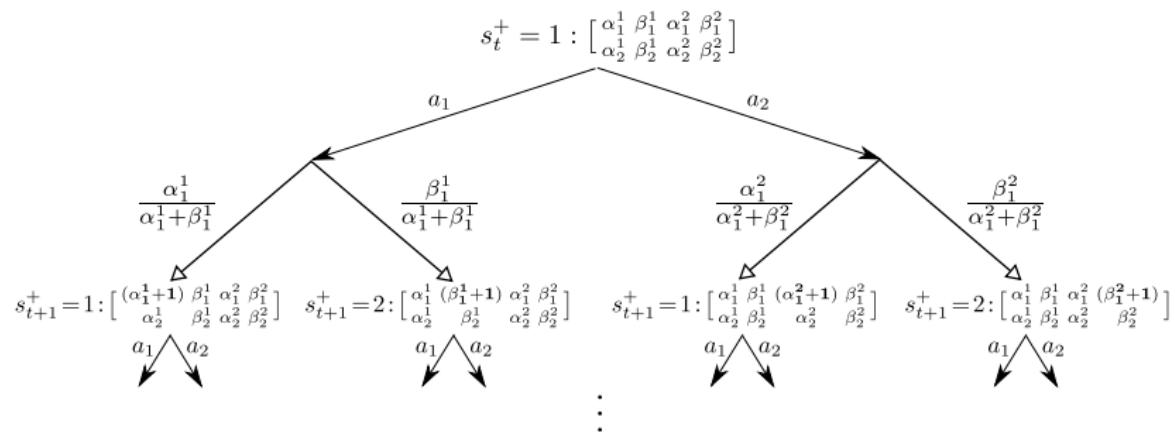
- We now have an infinite MDP over information states
- This MDP can be solved by reinforcement learning
- Model-free reinforcement learning
 - e.g. Q-learning (Duff, 1994)
- Bayesian model-based reinforcement learning
 - e.g. Gittins indices (Gittins, 1979)
 - This approach is known as *Bayes-adaptive RL*
 - Finds Bayes-optimal exploration/exploitation trade-off with respect to prior distribution

Bayes-Adaptive Bernoulli Bandits

- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function \mathcal{R}^a
- Each time a is selected, update posterior for \mathcal{R}^a
 - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
 - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{\mathcal{P}}$ for the Bayes-adaptive MDP
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model $Beta(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update



Bayes-Adaptive MDP for Bernoulli Bandits



Gittins Indices for Bernoulli Bandits

- Bayes-adaptive MDP can be solved by dynamic programming
- The solution is known as the *Gittins index*
- Exact solution to Bayes-adaptive MDP is typically intractable
 - Information state space is too large
- Recent idea: apply simulation-based search (Guez et al. 2012)
 - Forward search in information state space
 - Using simulations from current information state

Contextual Bandits

- A contextual bandit is a tuple $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$
- \mathcal{A} is a known set of actions (or “arms”)
- $\mathcal{S} = \mathbb{P}[s]$ is an unknown distribution over states (or “contexts”)
- $\mathcal{R}_s^a(r) = \mathbb{P}[r|s, a]$ is an unknown probability distribution over rewards
- At each step t
 - Environment generates state $s_t \sim \mathcal{S}$
 - Agent selects action $a_t \in \mathcal{A}$
 - Environment generates reward $r_t \sim \mathcal{R}_{s_t}^{a_t}$
- Goal is to maximise cumulative reward $\sum_{\tau=1}^t r_\tau$



Linear Regression

- Action-value function is expected reward for state s and action a

$$Q(s, a) = \mathbb{E}[r|s, a]$$

- Estimate value function with a linear function approximator

$$Q_\theta(s, a) = \phi(s, a)^\top \theta \approx Q(s, a)$$

- Estimate parameters by least squares regression

$$A_t = \sum_{\tau=1}^t \phi(s_\tau, a_\tau) \phi(s_\tau, a_\tau)^\top$$

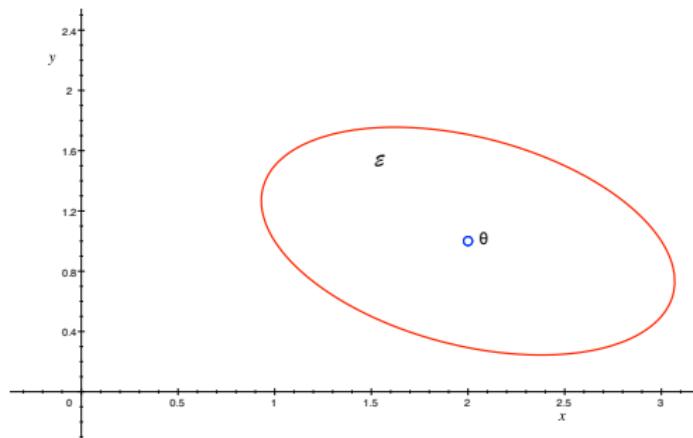
$$b_t = \sum_{\tau=1}^t \phi(s_\tau, a_\tau) r_\tau$$

$$\theta_t = A_t^{-1} b_t$$

Linear Upper Confidence Bounds

- Least squares regression estimates the mean action-value $Q_\theta(s, a)$
- But it can also estimate the variance of the action-value $\sigma_\theta^2(s, a)$
- i.e. the uncertainty due to parameter estimation error
- Add on a bonus for uncertainty, $U_\theta(s, a) = c\sigma$
- i.e. define UCB to be c standard deviations above the mean

Geometric Interpretation



- Define confidence ellipsoid \mathcal{E}_t around parameters θ_t
- Such that \mathcal{E}_t includes true parameters θ^* with high probability
- Use this ellipsoid to estimate the uncertainty of action values
- Pick parameters within ellipsoid that maximise action value

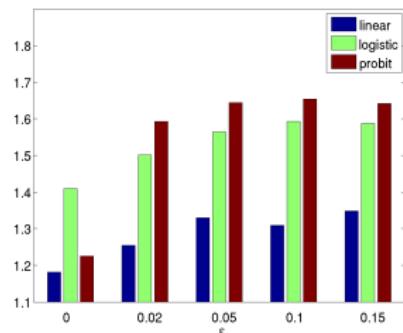
$$\operatorname{argmax}_{\theta \in \mathcal{E}} Q_\theta(s, a)$$

Calculating Linear Upper Confidence Bounds

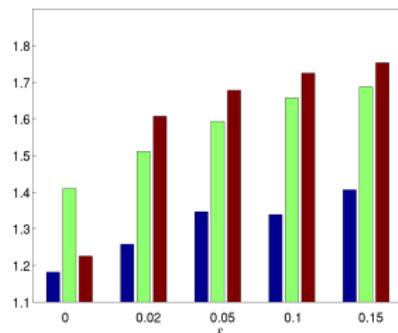
- For least squares regression, parameter covariance is A^{-1}
- Action-value is linear in features, $Q_\theta(s, a) = \phi(s, a)^\top \theta$
- So action-value variance is quadratic,
 $\sigma_\theta^2(s, a) = \phi(s, a)^\top A^{-1} \phi(s, a)$
- Upper confidence bound is $Q_\theta(s, a) + c \sqrt{\phi(s, a)^\top A^{-1} \phi(s, a)}$
- Select action maximising upper confidence bound

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_\theta(s_t, a) + c \sqrt{\phi(s_t, a)^\top A_t^{-1} \phi(s_t, a)}$$

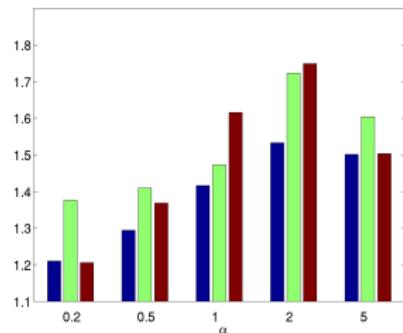
Example: Linear UCB for Selecting Front Page News



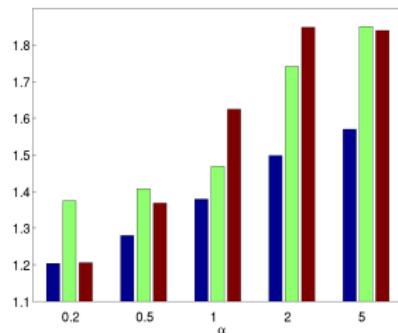
(a)



(b)



(c)



(d)

Exploration/Exploitation Principles to MDPs

The same principles for exploration/exploitation apply to MDPs

- Naive Exploration
- Optimistic Initialisation
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search

Optimistic Initialisation: Model-Free RL

- Initialise action-value function $Q(s, a)$ to $\frac{r_{max}}{1-\gamma}$
- Run favourite model-free RL algorithm
 - Monte-Carlo control
 - Sarsa
 - Q-learning
 - ...
- Encourages systematic exploration of states and actions

Optimistic Initialisation: Model-Based RL

- Construct an **optimistic** model of the MDP
- Initialise transitions to **go to heaven**
 - (i.e. transition to terminal state with r_{max} reward)
- Solve optimistic MDP by favourite planning algorithm
 - policy iteration
 - value iteration
 - tree search
 - ...
- Encourages systematic exploration of states and actions
- e.g. RMax algorithm (Brafman and Tennenholz)

Upper Confidence Bounds: Model-Free RL

- Maximise UCB on action-value function $Q^\pi(s, a)$

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q(s_t, a) + U(s_t, a)$$

- Estimate uncertainty in policy evaluation (easy)
- Ignores uncertainty from policy improvement
- Maximise UCB on optimal action-value function $Q^*(s, a)$

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q(s_t, a) + U_1(s_t, a) + U_2(s_t, a)$$

- Estimate uncertainty in policy evaluation (easy)
- plus uncertainty from policy improvement (hard)

Bayesian Model-Based RL

- Maintain posterior distribution over MDP models
- Estimate both transitions and rewards, $p[\mathcal{P}, \mathcal{R} \mid h_t]$
 - where $h_t = s_1, a_1, r_2, \dots, s_t$ is the history
- Use posterior to guide exploration
 - Upper confidence bounds (Bayesian UCB)
 - Probability matching (Thompson sampling)

Thompson Sampling: Model-Based RL

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(s, a \mid h_t) &= \mathbb{P} [Q^*(s, a) > Q^*(s, a'), \forall a' \neq a \mid h_t] \\ &= \mathbb{E}_{\mathcal{P}, \mathcal{R} \mid h_t} \left[\mathbf{1}(a = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a)) \right]\end{aligned}$$

- Use Bayes law to compute posterior distribution $p[\mathcal{P}, \mathcal{R} \mid h_t]$
- Sample** an MDP \mathcal{P}, \mathcal{R} from posterior
- Solve MDP using favourite planning algorithm to get $Q^*(s, a)$
- Select optimal action for sample MDP, $a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s_t, a)$

Information State Search in MDPs

- MDPs can be augmented to include information state
- Now the augmented state is $\langle s, \tilde{s} \rangle$
 - where s is original state within MDP
 - and \tilde{s} is a statistic of the history (accumulated information)
- Each action a causes a transition
 - to a new state s' with probability $\mathcal{P}_{s,s'}^a$
 - to a new information state \tilde{s}'
- Defines MDP $\tilde{\mathcal{M}}$ in augmented information state space

$$\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{R}, \gamma \rangle$$

Bayes Adaptive MDPs

- Posterior distribution over MDP model is an information state

$$\tilde{s}_t = \mathbb{P}[\mathcal{P}, \mathcal{R} | h_t]$$

- Augmented MDP over $\langle s, \tilde{s} \rangle$ is called **Bayes-adaptive MDP**
- Solve this MDP to find optimal exploration/exploitation trade-off (with respect to prior)
- However, Bayes-adaptive MDP is typically enormous
- Simulation-based search has proven effective (Guez et al.)

Conclusion

- Have covered several principles for exploration/exploitation
 - Naive methods such as ϵ -greedy
 - Optimistic initialisation
 - Upper confidence bounds
 - Probability matching
 - Information state search
- Each principle was developed in bandit setting
- But same principles also apply to MDP setting