MSML610: Advanced Machine Learning

Lesson 09.3: Multi-armed Bandits

Instructor: Dr. GP Saggese, gsaggese@umd.edu

References:

- ***Introduction***
- Algorithms
- Bayesian Approaches

SCIENCE ACADEMY

What are Multi-Armed Bandits?

- Imagine you're in a casino facing K slot machines (aka "one-armed bandits")
  - Each machine has a different unknown probability of winning
  - Limited budget of plays
  - *Goal*: Maximize total winnings
- **Multi-Armed Bandit (MAB)** is a sequential decision-making problem where:
  - At each time step $t = 1, 2, \ldots, T$, choose one action (arm) from K options
  - After choosing arm i, receive a random reward $r_t \sim P_i$
  - Reward distributions $P_1, \ldots, P_K$ are unknown
  - Learn which arms are best while collecting rewards
- Sequential decisions with uncertain outcomes and limited feedback
- **Key Challenge**: Balance:
  - *Exploration*: Try different arms to learn their reward distributions
  - *Exploitation*: Play the arm you believe is best

SCIENCE ACADEMY

Why Study Bandits?

- **Fundamental Problem in Sequential Decision-Making**
  - Core model for learning under uncertainty
  - Simplest non-trivial reinforcement learning setting
  - Rich theoretical framework with practical algorithms
- **Tractable Analysis**
  - Stateless: No long-term consequences of actions
  - Clean mathematical formulation
  - Provable regret bounds and optimality results
  - Serves as building block for more complex RL problems
- **Wide Applicability**
  - Online advertising (which ad to show?)
  - Recommendation systems (which movie to suggest?)
  - Clinical trials (which treatment to test?)
  - Website optimization (A/B testing)
  - Resource allocation (which server to query?)

Applications: Real-World Decision-Making

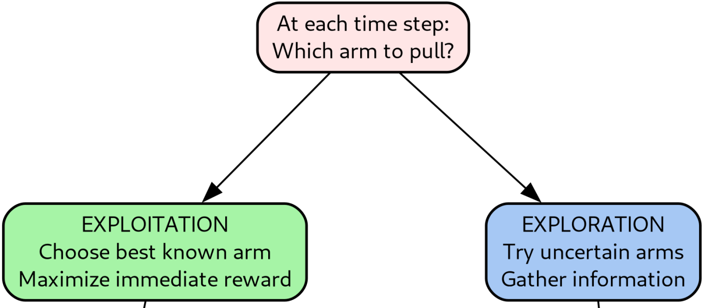
- **Online Advertising**
  - Platform must choose which ad to display
  - Each ad click generates revenue
  - Must learn which ads perform best
  - Trade-off: Show proven ads vs. test new ones
- **Clinical Trials**
  - Allocate patients to treatment arms
  - Each treatment has unknown efficacy
  - *Ethical imperative*: Minimize patients receiving inferior treatments
  - *Exploration-exploitation* is life-or-death
- **A/B Testing**
  - Test website designs, features, UI changes
  - Each variant has unknown conversion rate
  - Want to identify best variant quickly
  - Minimize opportunity cost during testing

Applications: Real-World Decision-Making

- **Recommendation Systems**
  - Netflix/Spotify: Which content to recommend?
  - Each item has unknown appeal to user
  - Must balance popular items vs. discovery
  - Learn user preferences over time
- **Network Routing**
  - Choose which server/path to route request
  - Each option has variable latency/reliability
  - Conditions change over time
  - Need fast, adaptive decisions
- **Financial Portfolio Allocation**
  - Allocate capital among investment options
  - Each asset has unknown future returns
  - Balance risk and reward
  - Market conditions non-stationary

The Exploration–Exploitation Tradeoff

- The "Exploration–Exploitation Tradeoff" is the central dilemma of sequential learning
- **Why This Tradeoff is Unavoidable**:
  - *Pure exploitation*: Never try suboptimal-looking arms $\rightarrow$ may stick with suboptimal choice forever
  - *Pure exploration*: Always try random arms $\rightarrow$ collect lots of information but low total reward
  - *Must do both*: Explore enough to find good arms, exploit enough to gain high reward



The K-Armed Bandit Problem: Formal Definition

- **Setting**:
  - K arms (actions), indexed $i = 1, \ldots, K$
  - Time horizon: T rounds (sometimes $T = \infty$)
  - At round $t = 1, \ldots, T$:
    1. Agent selects arm $A_t \in \{1, \ldots, K\}$
    2. Environment reveals reward $R_t \sim P_{A_t}$
    3. Agent observes only R_t (not rewards from other arms)
- **Stochastic Bandit Assumption**:
  - Each arm i has a fixed reward distribution P_i with mean $\mu_i = \mathbb{E}[R|A = i]$
  - Rewards are independent across rounds: R_t iid
  - The means $\mu_1, \ldots, \mu_K$ are unknown to the agent
- **Goal**: Design a policy (algorithm) that selects $A_1, A_2, \ldots, A_T$ to maximize

$$\text{Total Reward} = \sum_{t=1}^{T} R_t$$

- **Key Insight**: Can't maximize reward with certainty (rewards are random), but can minimize *regret*

Reward Distributions

- Each arm i has an associated **reward distribution** P_i
  - When arm i is pulled, reward $R \sim P_i$ is sampled
  - Common distributions: Bernoulli, Gaussian, bounded support
- **Expected Reward** (Mean):
  - Each arm has true mean reward: $\mu_i = \mathbb{E}[R|A = i]$
  - This is the quantity we care about for maximizing total reward
  - The agent does not know μ_i initially
- **Unknown Parameters**: The Challenge
  - The means $\mu_1, \ldots, \mu_K$ are unknown
  - Also variance, shape of distribution may be unknown
  - Agent must estimate these from observed rewards
  - Must balance learning (exploration) and earning (exploitation)
- **Example**: Bernoulli Bandits
  - Each arm i returns reward $R_t = 1$ with probability μ_i, else $R_t = 0$
  - μ_i is the "success probability" or "conversion rate"
  - Common in A/B testing: $\mu_i =$ click-through rate of ad i

Performance Metrics

- How do we measure algorithm performance when rewards are random?
- **Optimal Arm**: Define $i^* = \text{argmax}_i \mu_i$
  - The arm with highest expected reward
  - If we knew $\mu_1, \ldots, \mu_K$, we would always pull arm i^*
  - Optimal expected reward per round: $\mu^* = \mu_{i^*}$
- **Instantaneous Regret** at time t:

$$\ell_t = \mu^* - \mu_{A_t}$$

  - The difference between optimal mean and mean of chosen arm
  - How much expected reward we "lose" by not playing i^*
  - Note: $\ell_t \geq 0$ always
- **Cumulative Regret** over T rounds:

$$L_T = \sum_{t=1}^{T} \ell_t = T\mu^* - \sum_{t=1}^{T} \mu_{A_t}$$

  - Total expected loss from suboptimal decisions
  - Measures how much worse the algorithm is vs. always pulling i^*

Performance Metrics: Intuition

- **Why Regret?**
  - Cannot maximize reward with certainty (randomness in rewards)
  - But can minimize expected loss relative to optimal strategy
  - Regret = "opportunity cost" of learning
- **Key Property**: Regret depends only on how many times we pull each arm
  - Let $N_i(T) = \sum_{t=1}^{T} I[\{A_t = i\}]$ = number of times arm i pulled
  - Then:

$$L_T = \sum_{i=1}^{K} (\mu^* - \mu_i) N_i(T) = \sum_{i=1}^{K} \Delta_i \, N_i(T)$$

  - Where $\Delta_i = \mu^* - \mu_i$ is the "suboptimality gap" of arm i
- **Implications**:
  - To minimize regret, pull suboptimal arms as few times as possible
  - But we don't know which arms are suboptimal initially!
  - Must explore to identify i^*, but exploration increases regret

Regret Lower Bounds

- **Why Lower Bounds Matter**
  - Shows fundamental limits of *any* algorithm
  - No algorithm can do better (up to constants)
  - Tells us if an algorithm is near-optimal
  - Guides algorithm design: what's achievable?
- **Intuition**: Why Can't Regret Be Constant?
  - Need to distinguish between arms with close means
  - If $\mu_1 \approx \mu_2$, need many samples to tell which is better
  - Must pull suboptimal arms enough to be confident they're suboptimal
  - Trade-off: confidence vs. regret

Regret Lower Bounds: Lai–Robbins Bound

- **Lai and Robbins (1985)** proved:
  - For any "consistent" algorithm (one that identifies optimal arm eventually)
  - The cumulative regret must satisfy:

$$\liminf_{T\to\infty} \frac{L_T}{\log T} \geq \sum_{i:\Delta_i>0} \frac{\Delta_i}{D(\mu_i \| \mu^*)}$$

  - Where $D(\mu_i \| \mu^*)$ is the KL divergence between reward distributions
- **Simplified Version** (for bounded rewards):
  - For any consistent algorithm:

$$L_T = \Omega\left(\sum_{i:\Delta_i>0} \frac{\log T}{\Delta_i}\right)$$

  - Regret must grow at least logarithmically with T
- **Interpretation**:
  - Cannot achieve constant regret (must be $\Omega(\log T)$)
  - Smaller gap Δ_i $\rightarrow$ more pulls needed $\rightarrow$ higher regret
  - Best achievable: $L_T = O(\log T)$ (logarithmic regret)

Regret Lower Bounds: Implications for Algorithm Design

- **Design Goal**: Achieve $O(\log T)$ regret
  - Matches lower bound (up to constants)
  - Called "order-optimal" or "asymptotically optimal"
- **Key Insight**: Must pull each suboptimal arm i roughly $O(\frac{\log T}{\Delta_i^2})$ times
  - Arms with smaller gaps Δ_i need more exploration
  - Total regret: $\sum_i \Delta_i \cdot O(\frac{\log T}{\Delta_i^2}) = O(\frac{\log T}{\Delta_i})$
- **What Won't Work**:
  - Fixed exploration schedule (e.g., pull each arm n times): $\Omega(K)$ regret
  - Random sampling: $\Omega(\sqrt{T})$ regret
  - Need adaptive exploration that concentrates on promising arms

SCIENCE ACADEMY

- Introduction
- ***Algorithms***
- Bayesian Approaches

Baseline Strategies

- Before studying sophisticated algorithms, consider naive approaches
- **Greedy Algorithm**:
  - At each time t:
    1. Compute empirical mean reward for each arm: $\hat{\mu}_i(t) = \frac{1}{N_i(t)} \sum_{\tau: A_\tau = i} R_\tau$
    2. Pull arm with highest empirical mean: $A_t = \text{argmax}_i \hat{\mu}_i(t)$
  - Pure exploitation: always pull current best arm
- **Random Selection**:
  - At each time t: Choose A_t uniformly at random from $\{1, \ldots, K\}$
  - Pure exploration: no use of observed rewards

Baseline Strategies: Limitations

- **Greedy Algorithm Fails**:
  - *Problem*: May get "stuck" on suboptimal arm
  - *Example*: Suppose arm 1 has $\mu_1 = 0.6$, arm 2 has $\mu_2 = 0.7$
    - If we happen to pull arm 1 first and get reward 1
    - Then $\hat{\mu}_1(1) = 1$, and we'll keep pulling arm 1
    - Never try arm 2, which is actually better
  - *Regret*: $L_T = \Omega(T)$ (linear regret, very bad!)
- **Random Selection Fails**:
  - *Problem*: Never uses information from rewards
  - Pulls all arms equally, even after learning which are bad
  - *Regret*: $L_T = \Theta(\sqrt{KT})$ (grows with T, suboptimal)
- **Key Lesson**: Need to balance exploration and exploitation
  - Greedy: too much exploitation
  - Random: too much exploration
  - Need adaptive strategy that explores less over time

ϵ-Greedy Algorithm

- **Simple Idea**: With small probability ϵ, explore; otherwise exploit
- **Algorithm**:
  - At each time t:
    1. Compute empirical means: $\hat{\mu}_i(t) = \frac{\sum_{\tau < t : A_\tau = i} R_\tau}{N_i(t)}$
    2. With probability ϵ: Choose arm uniformly at random (explore)
    3. With probability $1 - \epsilon$: Choose arm $\text{argmax}_i \hat{\mu}_i(t)$ (exploit)
- **Hyperparameter**: $\epsilon \in (0, 1)$
  - Small ϵ (e.g., 0.01): mostly exploit, little exploration
  - Large ϵ (e.g., 0.5): explore half the time
  - Typical choice: $\epsilon = 0.1$

ϵ-Greedy: Exploration Schedules

- **Fixed ϵ**:
  - Use same ϵ for all t
  - *Problem*: Keeps exploring even after identifying best arm
  - *Regret*: $L_T = \Omega(T)$ (linear, not optimal)
- **Decaying ϵ_t**:
  - Use time-varying ϵ_t that decreases with t
  - Example: $\epsilon_t = \frac{1}{t}$ or $\epsilon_t = \min(1, \frac{K}{t})$
  - *Intuition*: Explore a lot early, less over time
  - *Regret*: Can achieve $L_T = O(T^{2/3})$ with right schedule (still not $O(\log T)$)
- **Limitations**:
  - Hard to choose optimal schedule without knowing Δ_i's
  - Explores uniformly (doesn't focus on promising arms)
  - Better algorithms exist (UCB, Thompson Sampling)

ϵ-Greedy: Regret Behavior

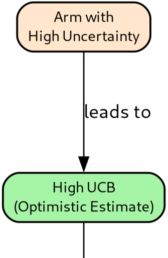
- **Fixed** ϵ: $L_T = \Theta(T)$
  - Explores ϵ fraction of time forever
  - Linear regret: $\epsilon T \sum_i \Delta_i / K$
- **Decaying** $\epsilon_t = O(1/t^\alpha)$:
  - If α too small: not enough exploration $\rightarrow$ may miss i^*
  - If α too large: too much exploration $\rightarrow$ high regret
  - Best tuned: $L_T = O(T^{2/3})$ (worse than $O(\log T)$ lower bound)
- **Practical Performance**:
  - Simple, easy to implement
  - Works reasonably well in practice
  - Often used as baseline
  - But theoretically suboptimal (cannot achieve $O(\log T)$)

Optimism in the Face of Uncertainty

- **Core Idea**: "Be optimistic about uncertain options"
  - If unsure about arm's value, assume it could be the best
  - Optimism drives exploration of uncertain arms
  - As arms are explored, optimism diminishes (uncertainty shrinks)
- **Why Optimism Works**:
  - If an arm *is* optimal: Optimism ensures we keep trying it
  - If an arm *is not* optimal: We'll eventually learn and stop pulling it
  - Natural exploration-exploitation balance without explicit randomization
- **Contrast with ϵ-Greedy**:
  - ϵ-greedy: Random exploration (ignores uncertainty)
  - Optimism: Directed exploration toward uncertain arms

Optimism: Confidence Intervals

- **Confidence Bound** for arm i after n pulls:
  - Empirical mean: $\hat{\mu}_i = \frac{1}{n} \sum_{\tau : A_\tau = i} R_\tau$
  - Uncertainty radius: $c_i(n) = \sqrt{\frac{2 \log T}{n}}$ (example)
  - Upper confidence bound: $\text{UCB}_i = \hat{\mu}_i + c_i(n)$
- **Interpretation**:
  - With high probability, true mean $\mu_i \leq \text{UCB}_i$
  - Fewer samples $n \rightarrow$ larger $c_i(n) \rightarrow$ more optimistic
  - More samples $n \rightarrow$ smaller $c_i(n) \rightarrow$ less optimistic
- **Optimistic Strategy**:
  - Pull arm with highest upper confidence bound: $A_t = \text{argmax}_i \text{UCB}_i(t)$
  - Automatically explores uncertain arms and exploits promising arms



Upper Confidence Bound (UCB)

- **UCB1 Algorithm** (Auer, Cesa-Bianchi, Fischer, 2002):
  - For first K rounds: Pull each arm once
  - For round $t > K$:
    1. For each arm i, compute UCB:

    $$\text{UCB}_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{2\log t}{N_i(t)}}$$

    2. Pull arm: $A_t = \text{argmax}_i \text{UCB}_i(t)$
- **Components**:
  - $\hat{\mu}_i(t)$: Empirical mean (exploitation term)
  - $\sqrt{\frac{2\log t}{N_i(t)}}$: Exploration bonus (uncertainty term)
  - $N_i(t)$: Number of times arm i pulled so far

SCIENCE ACADEMY

Upper Confidence Bound: Arm Selection Rule

- **UCB Index**:

$$\text{UCB}_i(t) = \underbrace{\hat{\mu}_i(t)}_{\text{exploitation}} + \underbrace{\sqrt{\frac{2\log t}{N_i(t)}}}_{\text{exploration bonus}}$$

- **How It Works**:
  - Arms with high empirical mean $\hat{\mu}_i$ have high UCB (exploitation)
  - Arms pulled few times (N_i small) have high UCB (exploration)
  - Bonus $\propto \frac{1}{\sqrt{N_i}}$: Shrinks with pulls
  - Bonus $\propto \sqrt{\log t}$: Grows slowly with time
- **Behavior Over Time**:
  - *Early*: All arms have few pulls $\rightarrow$ large bonuses $\rightarrow$ explores
  - *Middle*: Focuses on promising arms, occasionally revisits uncertain arms
  - *Late*: Mostly pulls optimal arm (bonus becomes negligible)

Upper Confidence Bound: Key Intuition

- **Why UCB Works**:
  - The bonus $\sqrt{\frac{2\log t}{N_i(t)}}$ is carefully calibrated:
    - Derived from Hoeffding's concentration inequality
    - Ensures with high probability: $\mu_i \leq \text{UCB}_i(t)$
    - "Optimism": We act as if each arm is as good as its UCB
- **Self-Correcting**:
  - If we pull a suboptimal arm too much: Its UCB decreases (more data $\rightarrow$ less uncertainty)
  - If we neglect the optimal arm: Its UCB increases ($\log t$ grows, while N_i stays fixed)
  - Automatically balances exploration and exploitation
- **No Tuning Parameters**:
  - Unlike ϵ-greedy (requires choosing ϵ)
  - UCB has no hyperparameters (besides confidence constant)
  - Adapts automatically to problem structure

SCIENCE ACADEMY

Regret Analysis of UCB

- **Theorem** (Auer et al., 2002):
  - For UCB1 with rewards in $[0, 1]$, the cumulative regret satisfies:

$$L_T \leq 8 \sum_{i:\Delta_i > 0} \frac{\log T}{\Delta_i} + \left(1 + \frac{\pi^2}{3}\right) \sum_{i=1}^{K} \Delta_i$$

  - **Simplified**: $L_T = O\left(\sum_{i:\Delta_i>0} \frac{\log T}{\Delta_i}\right)$
- **Interpretation**:
  - Logarithmic regret: $L_T = O(\log T)$
  - Matches Lai–Robbins lower bound (up to constants)
  - *Order-optimal*: Cannot do better asymptotically
- **Dependence on Δ_i**:
  - Smaller gaps $\Delta_i \rightarrow$ higher regret (harder to distinguish arms)
  - Arms with $\Delta_i = 0$ (i.e., optimal arms) don't contribute to regret bound

Regret Analysis: Role of Concentration Inequalities

- **Key Tool**: Hoeffding's Inequality
  - For rewards $R_1, \ldots, R_n$ iid in $[0, 1]$ with mean μ:

$$\Pr\left(|\hat{\mu} - \mu| \geq \epsilon\right) \leq 2\exp(-2n\epsilon^2)$$

- **Application to UCB**:
  - Choose radius $c = \sqrt{\frac{2\log t}{n}}$ so that:

$$\Pr\left(\mu_i > \hat{\mu}_i + \sqrt{\frac{2\log t}{N_i(t)}}\right) \leq \frac{1}{t^4}$$

  - With high probability, the true mean is below the UCB
- **Union Bound**: Across all arms and time steps:
  - Probability that any UCB is violated at any time $\leq \sum_{t=1}^{T} \sum_{i=1}^{K} \frac{1}{t^4} < \infty$
  - UCB indices are "optimistic" with high probability
- **Consequence**: If optimal arm has high UCB, we'll pull it; if suboptimal arm has high UCB, we'll pull it until UCB decreases

SCIENCE ACADEMY

Regret Comparison: UCB vs. ϵ-Greedy

ϵ-Greedy

- **Regret**: $L_T = O(T)$ (fixed ϵ) or $O(T^{2/3})$ (decaying ϵ)
- **Exploration**: Uniform random sampling
- **Pros**:
  - Simple to implement
  - Easy to understand
- **Cons**:
  - Linear or polynomial regret
  - Ignores uncertainty
  - Requires tuning ϵ

UCB

- **Regret**: $L_T = O(\log T)$
- **Exploration**: Directed by uncertainty
- **Pros**:
  - Logarithmic regret (optimal)
  - No hyperparameters
  - Adaptive
- **Cons**:
  - Slightly more complex
  - Requires confidence bounds

- **Practical Takeaway**: UCB is theoretically and empirically superior for stationary problems

Variants of UCB

- **UCB1**: Original algorithm (assumes bounded rewards in $[0, 1]$)
- **UCB-V** (Variance-Aware UCB):
  - Incorporates empirical variance of rewards
  - Index:
  $$\text{UCB-V}_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{2\hat{\sigma}_i^2(t)\log t}{N_i(t)}} + \frac{3\log t}{N_i(t)}$$
  - Where $\hat{\sigma}_i^2(t)$ is sample variance of arm i
  - **Benefit**: Better for arms with low variance (tighter bounds)
- **KL-UCB**:
  - Uses Kullback–Leibler divergence for tighter confidence bounds
  - Index:
  $$\text{KL-UCB}_i(t) = \max\{q : N_i(t)\, D(\hat{\mu}_i(t) \| q) \leq \log t\}$$
  - Where $D(p\|q)$ is KL divergence
  - **Benefit**: Asymptotically optimal constants (better than UCB1)

Variants of UCB: When They Matter

- **UCB1**: Good default choice
  - Simple, no tuning, strong guarantees
  - Works for any reward distribution with bounded support
- **UCB-V**: Use when variance varies across arms
  - Example: Some ads have consistent CTR, others highly variable
  - Can reduce regret if some arms have low variance
- **KL-UCB**: Use when asymptotic performance matters
  - Better constants in regret bound
  - More computationally expensive (requires solving equation for index)
  - Recommended for Bernoulli or Gaussian bandits
- **Practical Advice**: Start with UCB1; switch to variants if:
  - You have domain knowledge about variance structure (UCB-V)
  - You need best possible asymptotic performance (KL-UCB)

- Introduction
- Algorithms
- ***Bayesian Approaches***

Bayesian Bandits

- So far: **Frequentist** approach (UCB, ϵ-greedy)
  - Treat μ_i as fixed unknown constants
  - Use confidence bounds and empirical means
- **Bayesian Approach**: Treat μ_i as random variables
  - Start with prior distribution $p(\mu_i)$ for each arm
  - Update beliefs using Bayes' rule after each observation
  - Make decisions based on posterior distributions
- **Bayesian Framework**:
  - Prior: $p(\mu_i)$ (e.g., $\mu_i \sim \text{Beta}(1, 1)$ for Bernoulli arm)
  - Likelihood: $p(R|\mu_i)$ (e.g., $R \sim \text{Bernoulli}(\mu_i)$)
  - Posterior: $p(\mu_i|R_{1:t}) \propto p(R_{1:t}|\mu_i)p(\mu_i)$ (by Bayes' rule)

Bayesian Bandits: Priors and Posteriors

- **Prior Distribution**: Encodes initial beliefs about μ_i
  - Uniform prior: $\mu_i \sim \text{Beta}(1, 1)$ (no prior knowledge)
  - Informative prior: $\mu_i \sim \text{Beta}(\alpha, \beta)$ (domain knowledge)
- **Posterior Update**: After observing rewards from arm i
  - Suppose arm i has been pulled n times: s successes, f failures
  - Posterior (for Bernoulli arm with Beta prior):

$$p(\mu_i|\text{data}) = \text{Beta}(\alpha + s, \beta + f)$$

  - Conjugate prior: Posterior has same form as prior (easy to update)
- **Bayesian Inference**: Posterior encodes all information about μ_i
  - Mean: $\mathbb{E}[\mu_i|\text{data}] = \frac{\alpha+s}{\alpha+\beta+n}$
  - Uncertainty: $\mathbb{V}[\mu_i|\text{data}] = \frac{(\alpha+s)(\beta+f)}{(\alpha+\beta+n)^2(\alpha+\beta+n+1)}$
  - More data $\rightarrow$ sharper posterior (less uncertainty)

Bayesian Decision-Making

- **How to Choose Arm?** Several strategies:
- **Greedy Bayesian**: Choose arm with highest posterior mean

$$A_t = \text{argmax}_i \mathbb{E}[\mu_i | \text{data}_t]$$

  - Problem: No exploration (same issue as frequentist greedy)

- **Bayes-UCB**: Use posterior quantile as UCB

$$A_t = \text{argmax}_i Q_i(1 - \frac{1}{t})$$

  - Where $Q_i(p)$ is the p-th quantile of posterior $p(\mu_i | \text{data}_t)$
  - Similar to UCB, but uses Bayesian uncertainty

- **Probability of Best Arm**: Choose arm most likely to be optimal

$$A_t = \text{argmax}_i \Pr(\mu_i = \max_j \mu_j | \text{data}_t)$$

- **Thompson Sampling**: Sample from posteriors and play best sample (next slide)

Thompson Sampling

- **Posterior Sampling** (Thompson, 1933; Agrawal & Goyal, 2012)
  - At each time t:
    1. Sample $\tilde{\mu}_i \sim p(\mu_i | \text{data}_t)$ for each arm i
    2. Pull arm $A_t = \text{argmax}_i \tilde{\mu}_i$
    3. Observe reward R_t
    4. Update posterior for arm A_t
- **Intuition**: "Probability matching"
  - Pull arm i with probability $\approx \Pr(i \text{ is optimal} | \text{data})$
  - If arm likely to be best, sample will often be highest
  - If arm uncertain, might still be sampled (exploration)
- **Example**: Bernoulli Bandits with Beta Priors
  - Posterior: $p(\mu_i | \text{data}) = \text{Beta}(\alpha_i, \beta_i)$
  - Algorithm:
    1. Sample $\tilde{\mu}_i \sim \text{Beta}(\alpha_i, \beta_i)$ for each i
    2. Pull $A_t = \text{argmax}_i \tilde{\mu}_i$
    3. If reward $R_t = 1$: $\alpha_{A_t} \leftarrow \alpha_{A_t} + 1$; else $\beta_{A_t} \leftarrow \beta_{A_t} + 1$

SCIENCE ACADEMY

Thompson Sampling: Algorithm Steps

Thompson Sampling for Bernoulli Bandits

Input: Number of arms K, time horizon T

Initialize: For each arm $i = 1, \ldots, K$: - $\alpha_i \leftarrow 1, \beta_i \leftarrow 1$ (Beta(1,1) prior)

For $t = 1, 2, \ldots, T$:

1. **Sample** from posteriors:
   - For each arm i: $\tilde{\mu}_i \sim \text{Beta}(\alpha_i, \beta_i)$
2. **Select arm**: $A_t \leftarrow \text{argmax}_i \tilde{\mu}_i$
3. **Observe reward**: R_t
4. **Update posterior** for arm A_t:
   - If $R_t = 1$: $\alpha_{A_t} \leftarrow \alpha_{A_t} + 1$
   - If $R_t = 0$: $\beta_{A_t} \leftarrow \beta_{A_t} + 1$

Thompson Sampling: Practical Performance

- **Theoretical Guarantees**:
  - Achieves $L_T = O(\log T)$ regret (optimal)
  - Matches UCB asymptotically
  - Proofs more involved than UCB (Agrawal & Goyal, 2012)
- **Empirical Performance**:
  - Often *outperforms* UCB in practice (better constants)
  - Faster convergence to optimal arm in many problems
  - More robust to model misspecification
- **Advantages**:
  - Naturally incorporates prior knowledge (via priors)
  - Flexible: Works with any reward distribution (if conjugate prior available)
  - Simple randomized algorithm (no complex optimization)
- **Disadvantages**:
  - Requires choosing prior (though often uniform prior works well)
  - Posterior updates may be non-trivial for complex distributions
  - Theoretically less understood than UCB (until recent years)
- **Current Status**: *De facto* standard in practice (especially industry)

Comparison of UCB and Thompson Sampling

- Frequentist vs. Bayesian views
- Empirical vs. theoretical tradeoffs
- When to use each

Adversarial Bandits

- Motivation for adversarial models
- Difference from stochastic bandits
- Adversary assumptions

EXP3 Algorithm

- Probability-weighted arm selection
- Importance weighting
- Regret guarantees

Stochastic vs. Adversarial Settings

- Model assumptions
- Algorithm robustness
- Use cases

Contextual Bandits

- Using side information
- Feature vectors
- Decision rules with context

Linear Contextual Bandits

- Linear reward model
- Parameter estimation
- Assumptions

LinUCB

- Algorithm structure
- Confidence ellipsoids
- Regret guarantees

Contextual Thompson Sampling

- Bayesian linear models
- Posterior sampling with context
- Comparison to LinUCB

Bandits vs. Reinforcement Learning

- Stateless vs. stateful problems
- Bandits as a special case of RL
- When RL is needed

Non-Stationary Bandits

- Changing reward distributions
- Concept drift
- Practical motivation

Algorithms for Non-Stationary Bandits

- Discounted UCB
- Sliding-window methods
- Restart strategies

Structured Bandits

- Linear bandits
- Combinatorial bandits
- Graph and rank-one structures

Best-Arm Identification

- Pure exploration setting
- Fixed-confidence vs. fixed-budget
- Sample complexity

Practical Issues

- Delayed feedback
- Offline evaluation
- Safety constraints