Final Project: Multi-Algorithm Quantile Bandit Comparison

Author: Siyu Xie **Date:** December 2025

a. Motivation

Problem Statement

In Projects 2-3, I developed a Risk-Aware Contextual Bandit using quantile regression. However, one question remained: **Is Forced Sampling the best practical choice, or are there better alternatives?**

While Forced Sampling has theoretical guarantees, its exploration strategy (forced sampling every 2^n rounds) is inefficient and leads to slow convergence. Modern algorithms (LinUCB, Thompson Sampling, Epsilon-Greedy) perform well in practice but are typically designed for mean-based rewards (OLS), not quantile regression.

Why It Matters

It answers my group's research question. Also, it might be helpful in operations practice.

Impact: Algorithm choice affects: - Final performance (cumulative regret) - Parameter estimation quality - Computational efficiency - Convergence speed

b. Project Description

What I Built

Four Algorithms (all use quantile regression, differ in exploration):

Algorithm	Exploration Strategy	Key Feature
Forced Sampling	Forced every 2^n rounds	Baseline
LinUCB	Upper confidence bound	Conservative
Epsilon-Greedy	Decay $\epsilon = 1/\sqrt{t}$	Simple
Thompson Sampling	Posterior sampling	Bayesian

Framework Features: Flexible beta generation (decoupled from simulation); comprehensive tracking (regret, beta error, runtime); experimental infrastructure (6 scenarios, parameter sweeps).

Pipeline: User input → Generate parameters → For each replication/algorithm/timestep: generate context, choose action, observe reward, update via quantile regression, track metrics → Aggregate & analyze.

Course Concepts Applied:

- **Software Engineering:** Modular design, documentation, version control
- **Numerical Stability:** Safe operations, regularization, condition checking
- **Automation:** Makefile workflows, batch processing
- **Optimization:** Vectorization, efficient data structures, profiling

c. Results and Demonstration

Setup: All results use 5 simulations, 50 rounds, K=2 arms, d=10 dimensions (quick test mode).

Main Finding: No Universal Winner

Algorithm	Default	Gaussian	Sparse	Heavy-Tail	Wins	Avg Rank
LinUCB	27.77 (3rd)	**52.28 (1st)**	**28.43 (1st)**	**59.55 (1st)**	**3**	**1.50**
Epsilon-Greedy	**22.16 (1st)**	57.80 (2nd)	30.47 (2nd)	67.30 (2nd)	**1**	**1.75**
Thompson Sampling	23.50 (2nd)	95.30 (3rd)	52.11 (3rd)	76.76 (3rd)	0	2.75
Forced Sampling	40.95 (4th)	255.49 (4th)	62.99 (4th)	110.12 (4th)	0	4.00

Scenarios: (1) Default: Uniform beta - Epsilon wins; (2) Gaussian: Zero-mean - LinUCB wins 79.5% better; (3) Sparse: Weak signals - LinUCB wins; (4) Heavy-tail: Extreme values - LinUCB wins.

Key Insights: (1) LinUCB wins 3/4 scenarios - most robust; (2) Epsilon-Greedy simple and effective; (3) Thompson sensitive to assumptions; (4) Forced obsolete (31-80% worse); (5) Good estimation low regret - exploration efficiency matters more.

Runtime: All algorithms similar (~15s); bottleneck is quantile regression (70%). Choose based on performance, not speed.

d. Lessons Learned

Technical Challenges: (1) Adapting algorithms to quantile regression - used asymptotic variance theory for LinUCB/Thompson; (2) Flexible beta generation - decoupled generation from simulation, enabling diverse testing.

Course Impact: Transformed from single script with 1000+ warnings to modular structure with: separation of concerns, reusable components, numerical stability (safe operations, regularization), systematic testing, and comprehensive documentation.

Caveats: Simple data generating processes where estimation easier than exploration. Rankings may reverse in complex settings.

Future Work: High-dimensional settings (d » 100), theoretical analysis, Neural Thompson Sampling, alternative robustness measures (CVaR, worst-case).