

In this simulation study, I compared the performance of two bandit algorithms. The algorithm framework is given in [Algorithm Framework](#). OLS bandit uses OLS update in the beta updating stage.

Why these DGPs? Why these sample sizes/conditions?

Truncated normal contextual vector : Following the theoretical part of [Online Decision Making with High-Dimensional Covariates](#), we require the l-2 norm of contextual vector is bounded.

Two different uniformly distributed beta : With assumption 2, we expect the arms are "distinguishable" with high probability. Uniform distribution can be easily generated with some overlapping.

Heavy-tailed error : We expect the quantile update have robustness property compared with OLS update.

Sample size : Time frame T may grow large, but up to some point, the proportion of forced-sampling exploration becomes too small compared with the total number of samples.

How did you ensure your simulation design was fair and unbiased?

I tried several different degrees of freedom for the error student t-distribution to avoid cherry-picking.

What are the limitations of your simulation study?

I did not explore the impact of different β distribution and also the high-dimensional cases.

What scenarios did you not include, and why might they matter?

When the target focuses on different quantile level, the results may differ a lot. However, I do not have a good benchmark algorithm to compare with. It can be applied for conservative decision-making.

How do your results inform practice or theory?

It aligns with the theory that a quantile estimator in this case provides a more robust and more accurate β estimation. But this can be inferred from the quantile regression property.

What would you investigate next if you had more time/resources?

I may want to work on high-dimensional setting simulation, as the main contribution of the paper was based on high-dimensional contextual data.

Which aspects of the implementation were most challenging?

Modularize my code is definitely a pain. When I write them in functions, I get confused about which one should be called first. Also, the structure must be designed more carefully compared with working directly with interface notebook. Making the results reproducible -- adding a random seed in each part and creating a makefile are quite challenging.

How confident are you in your results? What could undermine that confidence?

I feel quite confident in my simulation results.