Optimistic Discrete Caching with Switching Costs

Machine Learning Algorithms for Caching Systems

Lucian Tosa

Supervisor: Naram Mhaisen

Responsible Professor: Dr. Georgios Iosifidis

L.Tosa@student.tudelft.nl

TU Delft

Introduction

- The main purpose of a caching system is to improve the latency of requests by keeping a subset of items closer to the customer. Therefore, the efficiency of caching systems is extremely important due to increased usage of its applications.
- Online learning approaches (Online Gradient Ascend, Follow-The-Perturbed-Leader) perform better than well-known eviction policies: LRU, LFU [1].
- Introduction of predictors (of unknown accuracy) for the requests led to *optimistic* caching algorithms: Optimistic FTPL (OFTPL) [2], the current state-of-the-art for discrete caching.
- Evaluation metric: Cache hit - a file that is in the cache is requested
- Different evaluation metric: Switching cost - a penalty incurred by the algorithm when removing or adding an item in the cache.
- The switching cost can have a greater impact than the number of cache hits in some scenarios. Under specific conditions, they can grow linearly.
- What strategies can we use to limit this switching cost?
- Proposed 2 modifications to the OFTPL algorithm in [2].

System Model

- Regret: a metric that evaluates the number of cache hits against a "best-in-hindsight" policy (the best static caching policy assuming knowledge of the future requests)
- Switching cost: an accumulated sum of difference in the caching states, D represents the weight of the penalty.
- Switching regret: sum of regret and switching cost
- Perturbation factor: term in the OFTPL algorithm that adds noise, increases with the prediction errors

$$R_T(\{x\}_T) \triangleq \sup_{\{f_t\}_{t=1}^T} \left\{ \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(x_t) \right\}$$

$$S_T(\{x\}_T) = \frac{D}{2} \sum_{t=2}^{T} \|y_t - y_{t-1}\|_1$$

$$SR_T(\{x\}_T) = R_T + S_T$$

$$\eta_t = \frac{1.3}{\sqrt{C}} \left(\frac{1}{\ln(Ne/C)} \right)^{\frac{1}{4}} \sqrt{\sum_{\tau=1}^{t-1} \left\| \theta_\tau - \tilde{\theta}_{\tau-1} \right\|_1^2}$$

Bounded perturbation

- Add a lower threshold to the perturbation factor, heuristically chosen
- The perturbations never hit 0, therefore ensuring noise even when predictions are 100% accurate

Switching cost informed perturbations

- The perturbation factor should increase with switching costs
- As switching costs increase, the algorithm should add more noise

$$\eta_t = \frac{1.3}{\sqrt{C}} \left(\frac{1}{\ln(Ne/C)} \right)^{\frac{1}{4}} \sqrt{\sum_{\tau=1}^{t-1} \left\| \theta_\tau - \tilde{\theta}_{\tau-1} \right\|_1^2 + S_{t-1}}$$

References

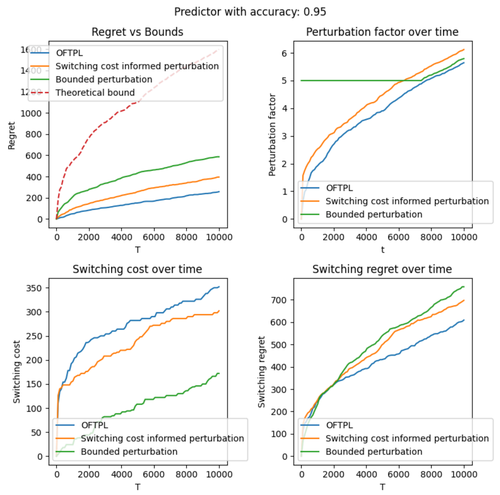
[1] Rajarshi Bhattacharjee, Subhankar Banerjee, and Abhishek Sinha. 2020. Fundamental Limits on the Regret of Online Network-Caching. Proc. ACM Meas. Anal. Comput. Syst. 4, 2 (2020), 31 pages.

[2] Naram Mhaisen, Abhishek Sinha, Georgios Paschos, and George Iosifidis. Optimistic no-regret algorithms for discrete caching. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 6(3):1–28, December 2022.

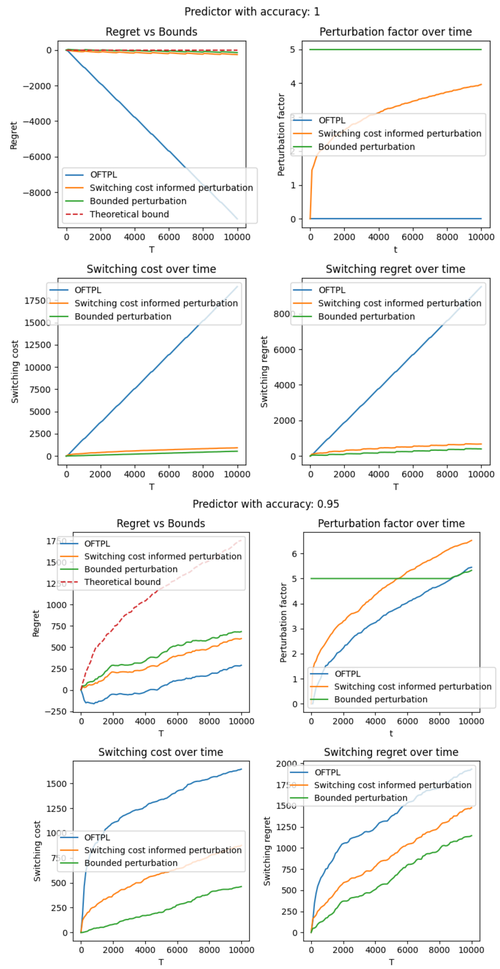
Experiments

- Generated synthetic data
- Used MovieLens dataset
- Generated predictors with varying accuracy

Zipf



Round-robin



MovieLens

Observations

- High accuracy -> low perturbation -> high switching cost
- High accuracy -> low regret
- High perturbation -> low switching cost / high regret

Results

- The perfect predictor offers us negative regret, representing numerous cache hits, but at the same time, our caching algorithm will always change the cache state at each time step, therefore incurring a linearly growing switching cost.
- In the real-world dataset, we see a massive improvement (up to 50% for the first approach, up to 31% for the second) from the two solutions compared to the original implementation.
- The Zipf distribution behaves completely differently than round-robin. Due to the highly localized requests, the switching cost does not increase with the accuracy of the predictions, our solutions perform the same or worse than the initial OFTPL. In this case the perturbation factor adds unwanted noise, leading to higher switching costs.
- Generally, the first approach may perform better than the second due to the high threshold we have chosen, but at the expense of higher regret, meaning less cache hits.

Conclusion

- We have proposed two heuristics to limit the switching cost. Performance is better than initial OFPTL implementation in terms of switching cost, switching regret, while keeping the regret withing bounds.
- Limitation: highly localized requests like Zipf, lead to worse performance because of unwanted noise.
- The first approach seems to achieve better results, however it requires manual tuning of the preset threshold, and it does not adapt to changes in the request pattern or prediction accuracy. The second approach being more adaptive.
- Improvements: theoretical analysis, varying D, weighted files, different cost for removing than adding, optimism for switching cost.