Data Driven Online Algorithms

Project Presentation

Department of Electrical & Computer Engineering
National Technical University of Athens

Christos Tsoufis

Monday, 12/7/2021

Contents

- Online algorithms (Paging Problem)
- Learning-Augmented Online Algorithms & Examples
- Analysis is based on the following Papers:
  - Competitive caching with machine learned advice [Lykouris, Vassilvitskii, ICML ‘18]
  - Near-Optimal Bounds for Online Caching with Machine Learned Advice [Rohatgi, SODA ‘20]
  - *Improvements*: Better and Simpler Learning-Augmented Online Caching [Wei, APPROX ‘20]

Online algorithms

- Traditional online algorithms encapsulate decision making under uncertainty and give ways to hedge against all possible future events, while guaranteeing a nearly optimal solution, as compared to an offline optimum.
- They process input piece-by-piece in a serial fashion.
- They provide guidance on how to act without any knowledge of future inputs.
- Competitive ratio is used to measure how good is an online algorithm (and may be overly cautious, which translates to high competitive ratios even for seemingly simple problems).
- Examples: Paging, Online Sorting

Competitive ratio

Definition:
Competitive ratio is the ratio between its performance and the offline algorithm's performance.

An online algorithm is α-competitive, if ∀ input x then

$$\text{online\_algorithm}(x) \leq \alpha \ \text{offline\_optimum\_algorithm}(x)$$

Example: Paging (Caching) Problem

- A series of requests (sequence of memory accesses) arrives one at a time to a server equipped with a small amount of memory.
- Upon processing a request, the server places the answer in the memory.
- Since the local memory has limited size (memory of size k or k pages), the server must decide which of the current elements to evict.
- If arriving item in the cache: cost = 0
- If arriving item not in the cache: cost = 1.
- Page replacement strategy: decide which page to evict (LRU (empirically good), FIFO (theoretically good), OPT (Belady's rule)(for offline greedy algorithm)).

Paging: deterministic algorithms

- Unfortunately, the lower bound for every deterministic algorithm is: $\Omega(k)$ – competitive.
- However, there are other techniques for further analysis (e.g. Resource augmentation).

Paging: Randomized algorithms

A randomized online algorithm is α-competitive, if ∀ input x then

$$E[\text{online\_algorithm}(x)] \leq \alpha \, \text{offline\_optimum\_algorithm}(x)$$

Here, the lower bound for randomized algorithms is: Ω(logk) - competitive

Paging: Marker algorithm

Analysis

- At the beginning of a phase all elements are unmarked.
- When an element arrives, mark it.
- When need to evict, pick a random unmarked element.
- When all elements are marked, start a new phase, and unmark all elements.

Related Work [Fiat et al. ‘91]:
Theorem: 2 logk - competitive

Machine Learning

Machine learning algorithms are in the business of extrapolating patterns found in the data to predict the future, and usually come with strong guarantees on the expected generalization error.

Despite the success and prevalence of machine learned systems across many application domains, there are still a lot of hurdles that one needs to overcome to deploy an ML system in practice. As these systems are rarely perfect, a key challenge is dealing with errors that inevitably arise.

Machine Learning - Preliminaries

Given a feature space X, describing the salient characteristics of each item and a set of labels Y, an example is a pair (x, y) where $x \in X$ describes the specific features of the example and $y \in Y$ gives the corresponding label.
In the binary search example, x can be thought as the query element q searched and y as its true position t(x).
A hypothesis is a mapping h: X→Y and can be probabilistic in which case the output on $x \in X$ is some probabilistically chosen $y \in Y$. In binary search, h(x) corresponds to the predicted position of the query.
The performance of a hypothesis is defined as ℓ:Y×Y → $\mathbb{R}^{\geq 0}$.
When labels lie in a metric space loss is defined as $\ell_1 = |y - \hat{y}|$ and squared loss as $\ell_2 = (y - \hat{y})^2$

Dealing with Uncertainty

Online Algorithms

- Full input is unknown (take decisions in order to be good in the worst case scenario – affects performance)
- Design algorithms for worst possible future
- Can not fully exploit patterns in data

+

Machine Learning

- But they are not robust
- Build models to predict the future
- Observe past data
- Uses predictors but has no theoretical guarantees

Learning-Augmented Online Algorithms

Desired characteristics:

- Consistency: Performance should improve with better predictions (if the predictor is perfect, this algorithm is ex-post optimal).
- Robustness: Performance should degrade gracefully with bad predictions. In the worst case that all of the predictions, it should return to the previously discussed competitive ratio.

Learning-Augmented Online Algorithms

Let $c(A, \eta)$ be the competitive ratio of algorithm A when the predictor has error rate η

- An algorithm A is β-consistent $\Leftrightarrow c(A, 0) = \beta$
- An algorithm A is γ-robust $\Leftrightarrow \forall \eta \in (0, 1): c(A, \eta) \leq \gamma$

Example: Faster Binary Search

We have a black-box predictor h that predicts the index of query element q. We will show how we can we take advantage of it.

Algorithm:

- Compare q with A[h(q)]
- If q == A[h(q)]
  - Finish
- Else if q > A[h(q)]
  - Find min k
    s.t. $A[h(q) + 2^k] \geq q$
    by looking at A[h(q)+2], A[h(q)+4], …
  - Binary search on $[h(q) \ldots h(q) + 2^k]$
- Else
  - Similar to the latter

Example: Faster Binary Search

Analysis:

- t(q): true index of q
- h(q): predicted index of q
- Predictor's error: $\eta = |h(q)-t(q)|$

- Find the interval: $O(\log(\eta))$
- Binary search: $O(\log(\eta))$
- O(1)-consistent
- O(log(n))-robust

Paper 1: Competitive caching with machine learned advice [Lykouris, Vassilvitskii, ICML '18]

Question: "What if the online algorithm is equipped with a machine learned oracle? How can we use the oracle to combine the predictive power of machine learning with the robustness of online algorithms?"

In this paper, they developed a framework for augmenting online algorithms with a machine learned oracle to achieve competitive ratios that provably improve upon unconditional worst case lower bounds when the oracle has low error. Our approach treats the oracle as a complete black box, and is not dependent on its inner workings, or the exact distribution of its errors.

Target

- Make minimal assumptions on the oracle. Specifically, since most machine learning guarantees are on the expected performance, our results are parametric as a function of the error of the oracle, η, and not the distribution of the error.
- Robustness: better oracle leads to better performance (smaller competitive ratio). A better oracle (one with lower η) results in a smaller competitive ratio.
- Worst-case competitive: in worst case the algorithm behaves comparably to the best online algorithm.

Measuring the prediction error of the oracle

Definitions:

- Input sequence: $z = z_1, z_2, \ldots, z_n$
- $h(z_i)$ estimates t_i = next appearance of z_i
- $Err(z_i) = |h(z_i) - t_i|$
- $\eta = \sum_i Err(z_i)$

Blindly follow the predictions of the oracle

Best case scenario:

- Evicting element predicted the furthest in the future (optimal algorithm)

Worst case scenario:

- Unbounded competitive ratio (even worse than the online algorithm)

Predictive Marker algorithm

Analysis:

- At the beginning of a phase, all elements unmarked.
- When an element arrives, mark it.
- When need to evict, pick the unmarked element predicted furthest in the future.
- When all elements are marked, start a new phase, and unmark all elements.

(Worst case: k-competitive)

Marker algorithm (Analysis)

Definition:

An input element is called:

- Clean for some phase r of the algorithm if it appeared in phase r but not in r-1.
- Stale if it appeared in phase r and phase r-1.

Lemma: Let L be the number of clean elements in an execution of a marker-based algorithm on some input σ. Then $L/2 \leq OPT(\sigma) \leq L$.

Definition: An arrival in phase r is an element which has not previously appeared in the phase.

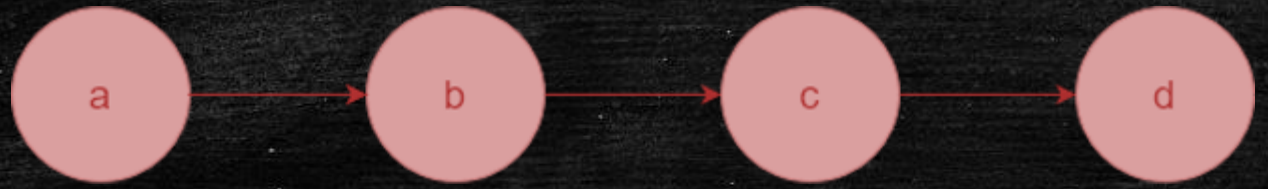
Marker algorithm (Analysis)

- This analysis uses the Eviction Chains (Blame graph).
- Each clean arrival in the phase causes a cache miss. Therefore yields a chain of evictions.

Marker algorithm (Analysis)

Eviction Chains

- Each element in the chain after the first clean element must be stale.
- Total number of cache misses in a phase is the length of the eviction chains.
- # chains = # of clean elements
- The graph is a series of paths (chains) as shown below.



Marker algorithm (Analysis)

Eviction Chains

- Main result: the length of each chain is: $O(\sqrt{n})$
- Result: Given a predictor with error η, Predictive Marker achieves competitive ratio:

$$O(1+\sqrt{1+4\frac{\eta}{OPT}})$$

Handling bad predictions

- If the predictions are bad, switch to randomized evictions for the remainder of the phase. (If the chain does not switch to random evictions, it has Belady evictions and incurs at most S_l misses from stale elements).
- Lemma: Randomization adds an additional O(logk) evictions to each chain in expectation.

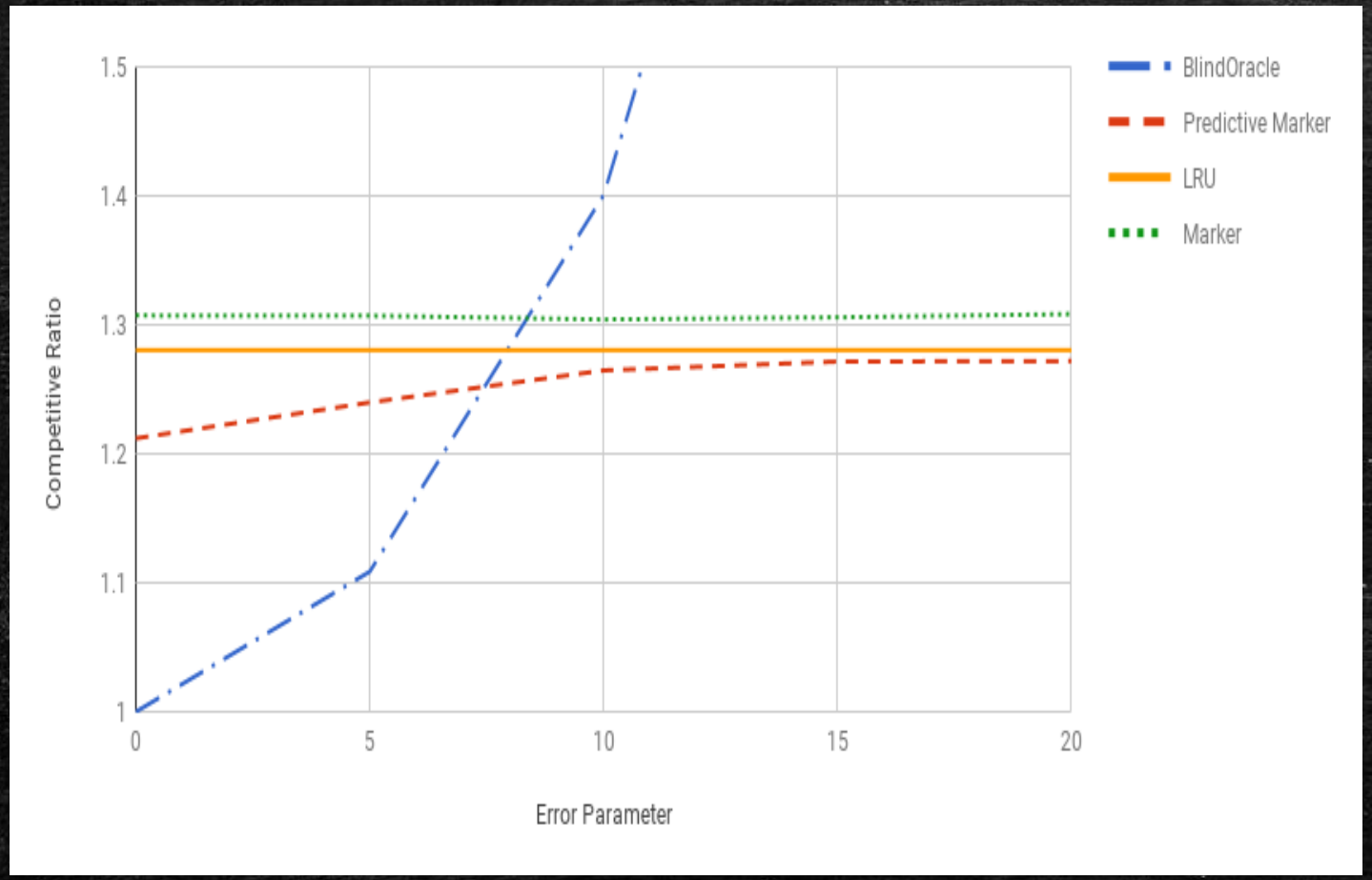
Final result

Finally, it is proved that the Predictive Marker achieves competitive ratio:

$$\min\left(2+2\sqrt{1+4\frac{\eta}{OPT}},\ 4\log k\right)$$

(Worst case: it is as good as online algorithm)

Empirical results



Algorithm	Competitive Ratio on BK	Competitive Ratio on Citi
Blind Oracle	2.049	2.023
LRU	1.280	1.859
Marker	1.310	1.869
Predictive Marker	1.266	1.810

Predictive marker outperforms LRU

Paper 2: Near-Optimal Bounds for Online Caching with Machine Learned Advice [Rohatgi, SODA ‘20]

Beyond Predictive Marker:

- We are following the prediction until eviction chains have grown considerably.
- The number of eviction chains is equal to the number of clean arrivals, which is asymptotically equal to OPT.
- Therefore bounding the average length gives us a bound for the competitive ratio.

(We will trust the oracle less.)

LMARKER

Analysis:

- If the incoming element is clean, then trust the predictor (evict the unmarked element with highest predicted arrival time).
- If the incoming element is stale, evict a random unmarked element.

Intuition

- We are trusting the predictor only once per chain.
- We will have $\Theta(\eta/OPT)$ error in each of the $\Theta(OPT)$ chains
- If the evicted element comes back, it will be at most $\Theta(\eta/OPT)$ away from the end of the phase.
- It will result to at most $O(\log(\eta/OPT))$ more cache misses for that chain.

LMARKER

Analysis

- Let $N_a(b)$ be the number of unmarked stale elements in cache at $t=i_a$, that will arrive after i_b.
- When a clean element arrives at time t, we evict e, the element predicted farthest in the future. At this time there are $N_t(e(t))$ unmarked stale elements in the cache.
- The expected length of the eviction chain begun at t is at most:
  - Order the $m = N_t(e(t))$ elements by true arrival time: $1+\log(N_t(e(t)))$.
  - If $z_{j_e}(t)$ evicts the j-th element, then there are only m-j other possible to evict.
  - The chain is bounded by $R_{N_t(e(t))}$.

$$R_m = 1 + \frac{1}{m}\sum_{j=1}^{m} R_{m-j}$$

LMARKER

Analysis

It is proved that:

$$\text{E[\# of misses in phase r]} \leq \sum_{e \in r} 1 + \log(N_c(e(c)))$$

By taking the sum of all of the phases and having in mind that $\eta_r \geq \frac{N}{2}$, the total number of misses is:

$$O(1 + \min(\log(\frac{\eta}{OPT}), \log k))$$

LNONMARKER

Algorithm

- If the incoming element is clean, evict the unmarked element with the highest predicted arrival time.
- If the incoming element is stale, evict a uniformly random element of the cache.

Intuition:

If $\eta/_{OPT}$ << k, a uniformly random cache element (potentially unmarked) will probably not appear after e, so evicting it might terminate the eviction chain at length O(1)

$$O(1 + \min(1, \log(\frac{\eta/_{OPT}}{k}), \log k))$$

Lower Bounds

- Lower bound from [Rohatgi SODA ‘20]:

$$\Omega(\log \min(\frac{\epsilon}{klogk}, k)), \forall \epsilon \geq \eta/_{OPT}$$

- ***Improved*** Lower Bound from [Wei, APPROX ‘20]:

$$1 + \Omega(\min(\frac{1}{k} \frac{\eta}{\mathrm{OPT}}, k))$$

(This is an algorithm that combines algorithms that follow an oracle with Marking-based algorithms.)

Additional Information

- ***Improved*** competitive ratio that matches the lower bound [Wei, APPROX ‘20]:

$$O(1 + \min(\frac{1}{k}, \frac{\eta}{\mathrm{OPT}}, \log k))$$

- ***Useful technique*** that can be extended beyond caching from (Combining robustness and competitiveness in a black-box manner) [Lykouris, Vassilvitskii, ICML ‘18]:

  For the caching problem, let A be an α-robust algorithm and B be a γ-competitive algorithm. We can then create a black-box algorithm ALG that is both 9α-robust and 9γ-competitive.

End of Presentation

Thank you!