Competitive Caching with Machine Learned Advice

T. Lykouris, S. Vassilvitskii

ICML 2018

Presenter: Fang, Chi-Hao

July 2,2020

Outline

Outline

Introduction

Input: A sequence of natural numbers $X = x_1, x_2, ...,$, where x_i is the number of the page requested by the CPU at time i.

Given a cache of size k, initially the cache is empty. The cache is simply an array of size k, s.t. a single page can be stored at each position in the array.

For each arriving x_i, if x_i is in the cache, the algorithm moves on to the next element. If not, the algorithm specifies an index $y_i \in [k]$, which points to a location in the cache where page x_i is to be stored evicting any existing page.

Introduction

We will measure the performance by competitive ratio - the ratio of the number of the cache misses of an online algorithm to the minimum number of cache misses achieved by an optimal offline algorithm that sees the entire sequence in advance.

Competitive Analysis

Definition

Let σ be the input sequence of elements for a particular online decision-making problem, *cost*$_{\mathcal{A}}(\sigma)$ *be the cost incurred by an online algorithm* $\mathcal{A}$ *on* σ*, and OPT(*σ*) be the cost incurred by the optimal offline algorithm.*
Then $\mathcal{A}$ *has competitive ratio* CR *if* $\forall\sigma\ , cost_{\mathcal{A}}(\sigma) \leq CR \times OPT(\sigma)$*.*

Natural Algorithms for Paging Problems

First In First Out(FIFO)
If the cache is full and a cache miss occurs, this algorithm evicts the page from the cache that **was inserted the earliest.**

Least Recently Used(LRU)
If the cache is full and a cache miss occurs, this algorithm evicts the page from the cache that **was accessed least recently.**

Outline

Introduction

In this section we define a more general class of online problems, called **request-answer games** and then define the competitive ratio of deterministic online algorithms w.r.t. request-answer games.

Request-Answer Games

Definition

A request-answer game for a minimization problem consists of a request set $\mathcal{R}$ and an answer set $\mathcal{A}$ and cost functions $f_n : \mathcal{R}^n \times \mathcal{A}^n \to \mathbb{R} \cup \{\infty\}$ for $n = 0, 1, ...$

```
1 On an instance I, including an ordering of the data items $(x_1, ..., x_n)$;
2 $i := 1$;
3 while there are unprocessed data items do
4     The algorithm receives $x_i \in \mathcal{R}$ and makes an irrevocable
       decision $d_i \in \mathcal{A}$ for $x_i$;
5     (based on $x_i$ and all previously seen data items and decisions);
6     $i := i + 1$;
7 end
```

Algorithm 1: Online Algorithm Template

Outline

Introduction

This section shows how randomness can help in solving an online problem.

To do so, we need to extend the definition of competitive ratio to randomized algorithms, since randomness leads to different kinds of adversaries such as **oblivious**, **adaptive** (or **adaptive online**), and **fully adaptive**(or **adaptive offline**).

Thus we can't measure the performance of randomized algorithms as how we do it in measuring that of the deterministic algorithms.

Template

```
1 $\mathcal{R} \leftarrow$ infinite tape of random bits;
2 On an instance *I*, including an ordering of the data items $(x_1, ..., x_n)$;
3 $i := 1$;
4 **while** *there are unprocessed data items* **do**
5     The algorithm receives $x_i \in \mathcal{R}$ and makes an irrevocable
       decision $D_i := D_i(x_1, ..., x_i, \mathcal{R})$ for $x_i$;
6     (based on $x_i$ and all previously seen data items and $\mathcal{R}_i$);
7     $i := i + 1$;
8 **end**
```

Algorithm 2: Randomized Online Algorithm Template

Template

A randomized online algorithm generalizes the deterministic paradigm by allowing the decision in step5 of the template to be a randomized decision.

We view the algorithm as having access to an infinite tape of random bits.

We denote the contents of the tape by $\mathcal{R}$,i.e.,$\mathcal{R}_i \in \{0, 1\}$ for $i \geq 1$, and the $\mathcal{R}_i$ are distributed uniformly and independently of each other.

Types of Adversaries

Oblivious: The adversary designs the input sequence σ at the beginning. It does not know any randomness used by the algorithm $\mathcal{A}$.

Adaptive: At each time t, the adversary knows all randomness used by algorithm $\mathcal{A}$ thus far. In particular, it knows the exact state of the algorithm. With this in mind, it then picks the $(t + 1)$-th element in the input sequence.

Fully adaptive: The adversary knows all possible randomness that will be used by the algorithm $\mathcal{A}$ when running on the full input sequence σ.

Relationships between Adversaries

Theorem

For a minimization problem and a randomized online algorithm ALG we have

$$p_{OBL}(ALG) \leq p_{ADON}(ALG) \leq p_{ADOFF}(ALG).$$

How Much Can Randomness Help

We start by showing that the gap between the best competitive ratio achieved by a randomized algorithm and a deterministic algorithm can be arbitrarily large.
Fix any gap function $g : \mathbb{N} \to \mathbb{R}$.
Consider the following maximization problem:

Modified Bit Guessing Problem

Input: $(x_1, x_2, ..., x_n)$, where $x_i \in \{0, 1\}$.
Output: $z = (z_1, z_2, ..., z_n)$, where $z_i \in \{0, 1\}$.
Objective: To find z s.t. $z_i = x_{i+1}$ for some $i \in [n-1]$. If such i exists, the payoff is $\frac{g(n)}{1 - \frac{1}{2^{n-1}}}$, otherwise the payoff is 1.

How Much Can Randomness Help

Theorem

Every deterministic algorithm ALG achieves objective value 1 on the Modified Bit Guessing Problem.
There is a randomized algorithm that achieves expected objective value $g(n)$ against an oblivious adversary on inputs of length n for the Modified Bit Guessing Problem.

How Much Can Randomness Help

Proof.

Consider a deterministic algorithm ALG. The adversarial strategy is as follows:
Present $x_1 = 0$ as the first input item. The algorithm replies with z_1.
The adversary defines $x_2 = \neg z_1$. Continue this process for $n - 2$ more steps. In other words, the adversary defines $x_i = \neg x_{i-1}$ for $i = 2, ..., n$, making sure that the algorithm does not guess any of the bits. Thus, the algorithm achieves objective function value 1. □

How Much Can Randomness Help

Proof.

Consider the randomized algorithm that selects z_i uniformly at random. The probability that it picks $z_1, ..., z_{n-1}$ to be different from $x_2, ..., x_n$ in each coordinate is exactly $\frac{1}{2^{n-1}}$. Therefore w.p. $1 - \frac{1}{2^{n-1}}$ it guesses at least one bit correctly. Therefore the expected value of the objective function is **at least**(we omit the part of $1 \cdot \frac{1}{2^{n-1}}$)

$$\frac{g(n)}{1 - \frac{1}{2^{n-1}}} \cdot (1 - \frac{1}{2^{n-1}}) = g(n).$$

□

How Much Can Randomness Help

Corollary

The gap between p_{OBL} *and* p_{ADOFF} *can be arbitrarily large.*

Lower Bound for Paging

We revisit the paging problem and we want to get a lower bound on the competitive ratio achieved by any randomized algorithm.
First, we need the following definition.

Definition

The nth harmonic number, denoted by H_n, is defined as

$$H_n = 1 + \frac{1}{2} + ... + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i}$$

Note that $H_n \approx \ln(n)$

Lower Bound for Paging

Theorem

Let ALG be a randomized online algorithm for the paging problem with cache size k. Then we have

$$p_{OBL}(ALG) \geq H_k.$$

Upper Bound for Paging

Next, we see an algorithm called Mark that achieves the competitive ratio $\leq 2H_k$ against an oblivious adversary for the paging problem.

Upper Bound for Paging

```
$C[1, ..., k]$ stores cache contents;
$M[1, ..., k]$ stores a Boolean flag for each page in the cache;
Initialize $C[i] \leftarrow -1$ for all $i \in [k]$ to indicate that cache is empty;
Initialize $M[i] \leftarrow False$ for all $i \in [k]$;
$j \leftarrow 1$
while $j \leq n$ do
    if $x_j$ is in $C$ then
        Compute $i$ s.t. $C[i] = x_j$;
        if $M[i] = False$ then
            $M[i] \leftarrow True$
        end
    else
        if $M[i] = True$ for all $i$ then
            $M[i] \leftarrow False$ for all $i$
        end
        $S \leftarrow \{i | M[i] = False\}$ ;
        $i \leftarrow$ uniformly random element of $S$ ;
        Evict $C[i]$ from the cache; $C[i] \leftarrow x_j$; $M[i] \leftarrow True$;
    end
    $j \leftarrow j + 1$
end
```

Theorem

$p_{OBL}(Mark) \leq 2H_k$

Outline

Objective of the study

- Combine the predictive power of machine learning with the online algorithm.

Machine Learning Basics

First, we recall some definitions.

Definition

Given a feature space $\mathcal{X}$ and a set of labels $\mathcal{Y}$, the pair (x, y), where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is specific features and the corresponding label.

A hypothesis is a mapping $h : \mathcal{X} \rightarrow \mathcal{Y}$.

Define a loss function $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+ \cup \{0\}$.

OMLA

Let $\mathcal{H}$ be a hypothesis class and $h \in \mathcal{H}$ be a machine learned predictor.
The online input consists of a set of elements from $\mathcal{Z}$.
For any input σ, its elements are denoted by $z(\sigma_1)$, $z(\sigma_2)$, ... with length $|\sigma|$.
The error of the predictor h on σ w.r.t. the loss function ℓ is:

$$\eta_\ell(h, \sigma) = \sum_i \ell(y(\sigma_i), h(\sigma_i)).$$

Definition (Online with Machine Learned Advice(OMLA))

The OMLA instance consists of :
(i) An input $\sigma = \{z(\sigma_1), z(\sigma_2), ..., z(\sigma_{|\sigma|})$; each $z(\sigma_i) \in \mathcal{Z}$ has features $x(\sigma_i) \in \mathcal{X}$ and $y(\sigma_i) \in \mathcal{Y}$. $\}$
(ii) A predictor $h : \mathcal{X} \to \mathcal{Y}$ that predicts a label $h(\sigma_i)$ for each $x(\sigma_i) \in \mathcal{X}$.
(iii) The error of predictor h at sequence σ w.r.t. loss ℓ, $\eta_\ell(h, \sigma)$.

OMLA

Our goal is to create online algorithms that can use its advice to achieve an improved competitive ratio when augmented with a predictor h.

Definition (ε-accurate)

Given an optimization problem Π, let $OPT_\Pi(\sigma)$ denote the value of the optimal solution on the input σ. We say that a predictor h is ε-accurate w.r.t. a loss function ℓ for Π if for any σ:

$$\eta_\ell(h, \sigma) \leq \varepsilon \cdot OPT_\Pi(\sigma).$$

We use $\mathcal{H}_\ell(\varepsilon)$ to denote the class of ε-accurate predictors, omitting Π for notation clarity since it is fixed.

Definition (ε-assisted)

An algorithm is ε-assisted if it has access to an ε-accurate predicor.

OMLA

Remark: From the definitions above, we can see that the competitive ratio of an ε-assisted algorithm is a function of ε.

Definition (Competitive Ratio)

Let $CR_{\mathcal{A}(h)}(\sigma)$ be the competitive ratio of algorithm $\mathcal{A}$ which uses a predictor h on input sequence σ. The competitive ratio of an ε-assisted algorithm $\mathcal{A}$ is:

$$CR_{\mathcal{A},\ell}(\varepsilon) = \max_{\sigma, h \in \mathcal{H}_\ell(\varepsilon)} CR_{\mathcal{A}(h)}(\sigma).$$

.

Definition

$\mathcal{A}$ is β-consistent if $CR_{\mathcal{A}(h)}(0) = \beta$.
$\mathcal{A}$ is α-robust for a function $\alpha(\cdot)$ if $CR_{\mathcal{A}(h)}(\sigma) = \mathcal{O}(\alpha(\varepsilon))$.
$\mathcal{A}$ is γ-competitive if $\forall \varepsilon, CR_{\mathcal{A},\ell}(\varepsilon) \leq \gamma$.

Paging with ML Advice

Here we define the framework for the paging problem.

Evicting element predicted the furthest in the future.

Theorem

Let ℓ_1 be the absolute loss. The competitive ratio of ε-assisted algorithm $\mathcal{B}$ is $CR_{\mathcal{B},\ell_1}(\varepsilon) = \Omega(\varepsilon)$

Remark

When the error of the predictor is much worse than the offline optimum, the competitive ratio becomes unbounded.

Evicting elements with proven wrong predictions

Theorem

The competitive ratio of ε*-assisted algorithm* $\mathcal{W}$ *is* $CR_{\mathcal{W},\ell_1}(\varepsilon) = \Omega(\varepsilon)$.

Predictive Marker Algorithm

Let $H_k = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{k}$ be the k-th harmonic number.
Recall the classic marker algorithm:

- All elements are unmarked at the beginning of each phase.
- When an element arrives and it is already in the cache, then it is marked; if not in the cache, evicting a random unmarked element and put this newly arrived element in the cache and mark it.
- Once all elements are marked and a new cache miss occurs, the phase ends and we unmark all the elements.

Definition

An element is called *clean* in phase r if it appears during phase r, but does not appear during phase $r - 1$.
Otherwise, it is *stale* i.e.it appears during phase r and $r - 1$.

Predictive Marker Algorithm

Claim (1)

Let Q be the number of clean elements. Then the optimal algorithm has at least $\frac{Q}{2}$ cache misses.

Claim (2)

Let Q be the number of clean elements. Then the expected number of cache misses of the marker algorithm is $Q \cdot H_k$.

Predictive Marker Algorithm

The classic marker algorithm is part of a larger family of marking algorithm which never evict marked elements when there are unmarked elements present. And any algorithm in this family has a worst case competitive ratio of k.

Our next goal is to reduce the worst case competitive ratio to $\mathcal{O}(H_k)$.

Predictive Marker Algorithm

We combine the prediction-based tie-breaking rule with the random tie-breaking rule.

Suppose an element e is evicted during this phase. We construct a graph to understand the reason why e is evicted.
There are 2 cases:

- either it was evicted when a clean element c arrived, then we add a directed edge from e to c.
- it was evicted when a stale element s arrived but s was previously evicted. In this case we add a directed edge from e to s.

Predictive Marker Algorithm

Remark

1.The graph is a set of chains(paths).
2.The total length of the chains represents the total number of evictions incurred by the algorithm during the phase.
3.The number of distinct chains is the number of clean elements.

Predictive Marker Algorithm

The modification in this paper is that when a stale element arrives

- Evict a new element in a prediction-based manner if the chain containing it has length $< H_k$.
- Otherwise, evicting a random unmarked element.

Analysis

First, we define the growth rate of the loss function in order to analyze the performance of the proposed algorithm.

Definition

Let $A_T = \{a_1, a_2, ..., a_T\}$ be a sequence of strictly increasing integers of length T, and $B_T = \{b_1, b_2, ..., b_T\}$ be a sequence of non-increasing reals of length T. For a fixed loss function ℓ , we define its spread $S_\ell : \mathbb{N}^+ \rightarrow \mathbb{R}^+$ by:

$$S_\ell(m) = \min\{T| \min_{A_T, B_T} \ell(A_T, B_T) \geq m\}.$$

Analysis

Lemma

For absolute loss, $\ell_1(A,B) = \sum_i |a_i - b_i|$*, the spread of* ℓ_1 *is* $S_{\ell 1}(m) \leq \sqrt{5m}$*.*
For squared loss, $\ell_2(A,B) = \sum_i (a_i - b_i)^2$*, the spread of* ℓ_2 *is* $S_{\ell 2}(m) \leq \sqrt[3]{14m}$*.*

Analysis

Theorem

Let a loss function ℓ with spread bounded by S_ℓ. If S_ℓ is concave, the competitive ratio of ε-assisted Predictive Marker PM is bounded by:

$$CR_{PM,\ell}(\varepsilon) \leq 2 \cdot \min(1 + 2S_\ell(\varepsilon), 2H_K) \; .$$

Analysis

Lemma

Using algorithm SM, at most H_K special elements cause cache misses per phase in expectation.

Analysis

Lemma

For any loss function ℓ, any phase r, the expected length of any chain is at most $1 + S_\ell(\eta_{r,c})$, where $\eta_{r,c}$ is the cumulative error of the predictor on the elements in the chain and S_ℓ is the spread of the loss.

Analysis

Lemma

For any loss function ℓ, any phase r, the expected length of any chain is at most $\min(1 + 2S_\ell(\eta_{r,c}), 2\log K)$, where $\eta_{r,c}$ is the cumulative error of the predictor on the elements in the chain and S_ℓ is the spread of the loss.

Analysis

Corollary

The competitive ratio of ε-assisted Predictive Marker w.r.t. the absolute loss ℓ_1 is bounded by $CR_{PM,\ell_1}(\varepsilon) \leq 2 \cdot \min(1 + 2\sqrt{5\varepsilon}, 2H_K)$.

Corollary

The competitive ratio of ε-assisted Predictive Marker w.r.t. the squared loss ℓ_2 is bounded by $CR_{PM,\ell_2}(\varepsilon) \leq 2 \cdot \min(1 + 2\sqrt[3]{14\varepsilon}, 2H_K)$.

Tightness of Analysis

We want to show that the analysis is tight i.e. any marking algorithm that uses the predictor in a deterministic way cannot achieve an better guarantee w.r.t. robustness.

Theorem

Any deterministic ε-assisted marking algorithm $\mathcal{A}$, that only uses the predictor in the tie-breaking among unmarked elements in a deterministic fashion, has a competitive ratio of $CR_{\mathcal{A},\ell}(\varepsilon) = \Omega(\min(S_\ell(\varepsilon), k))$.

On the Rate of Robustness in Paging

Theorem3.4 shows that the analysis of Predictive Marker is tight w.r.t. the rate of robustness, and suggests that algorithms that use the predictor in a deterministic manner suffer from similar rates.

Question: Whether there is a better rate can be achieved using the predictor in a randomized way?

Randomized Predictors

Definition

For a fixed optimization problem Π, let $OPT_{\Pi}(\sigma)$ denote the value of the optimal solution on input σ. Assume that the predictor is probabilistic and therefore the error of the predictor at σ is a random variable $\eta_\ell(h,\sigma)$. We say that a predictor h is ε- accurate in expectation for Π if:

$$\mathbb{E}[\eta_\ell(h,\sigma)] \leq \varepsilon \cdot \mathbb{E}[OPT_{\Pi}(\sigma)] \; .$$

Theorem

Let a loss function ℓ with spread bounded by S_ℓ. If S_ℓ is concave, the competitive ratio of ε-assisted $mathcalPM$ is:

$$CR_{\mathcal{PM},\ell}(\varepsilon) \leq 2 \cdot \min(1 + 2S_\ell(\varepsilon), 2H_K) \; .$$

Section Introduction

So far we have shown a caching problem with $\mathcal{O}(\sqrt{\varepsilon})$ competitive ratio for the absolute loss and ε-accurate predictor.

Now we give a finer trade-off of competitiveness and robustness.

Robustness vs Competitive Trade-offs

One of the free parameters in Predictive Marker algorithm is **the length of the chain** when the algorithm switches from the predictor to random evictions.

If the switch occurs after the chains grow to length(γH_k), this provides a trade-off between competitiveness robustness.

Theorem

For $\gamma > 0$, let $\mathcal{PM}(\gamma)$ be the algorithm that uses γ_k as switching point(line 18 in Algorithm 1). Let a loss function ℓ with spread bounded by s_ℓ. If S_ℓ is concave, the competitive ratio of ε-assisted $\mathcal{PM}(\gamma)$ is bounded by:

$$CR_{\mathcal{PM}(\gamma),\ell}(\varepsilon) \leq 2 \cdot \min(1 + \frac{1+\gamma}{\gamma} S_\ell(\varepsilon), \gamma H_k, k) \; .$$

Robustness vs Competitive Trade-offs

Remark

When $\gamma \to 0$ then the algorithm is more conservative(switching to random evictions earlier); this reduces the worst-case competitive ratio but at the cost of abandoning the predictor unless it is extremely accurate.

On the other hand, setting $\gamma >> 0$ makes the algorithm trust the predictor more, reducing the competitive ratio when the predictor is accurate at the expense of a worst guarantee when the predictor in unreliable.

Practical Traits of Predictive Marker

Locality: From Theorem3.3, the competitive ratio is bounded as a function of the quality of the prediction.

One may concerns that if the predictions have a small number of very large errors, then the applicability of Predictive Marker may be limited. The following shows that this is not the case.

Theorem

Consider a loss function ℓ with spread S_ℓ. If S_ℓ is concave, the competitive ratio of Predictive Marker PM at sequence σ when assisted by a predictor h is at most:

$$CR_{\mathcal{PM},\ell} \leq \frac{\sum_r CL(r,\sigma) \cdot \min(1 + 2S_\ell(\eta_{\ell,r}(h,\sigma), 2H_k)}{\sum_r CL(r,\sigma)} .$$

Practical Traits of Predictive Marker

If the predictor h is really bad for a period of time then the clean chains of the corresponding phases will contribute the second term. But the other phases will provide enhanced performance using the predictor's advice.

In this way, the algorithm adapts to the quality of the predictions, and bad errors do not propagate beyond the end of a phase. This is useful when the most patterns are generally well-predicted but there may be some unforeseen sequences.

Combining Robustness and Competitive in a Black-Box Manner

In previous sections wee have shown:

- how we can slightly modify a competitive algorithm while
- retain the worst-case competitiveness guarantees o.w.

Next, we see that achieving the requirements individually is enough. In particular, we can combine an algorithm that is robust and one that is worst-case competitive in a black-box manner.

Theorem

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

Datasets and Metrics

Datasets

- **BK**: data extracted from BrightKite
- **Citi**: data extracted form CitiBike

Metric

Competitive ratio of the algorithm: Defined as the number of misses incurred by the particular strategy divided by the optimum number of misses

Predictions

Running the experiments with the following 2 methods:

- **Synthetic Predictions**
- **PLECO Predictions**

Algorithm

Consider the following algorithms for evaluation.

- **LRU**
- **Marker**
- **Predictive Marker**
- **Blind Oracle**

Results