Competitive Caching with Machine Learned Advice
Seminar on Online Algorithms

Gal Wiernik
Tel Aviv University
May 25, 2022
1 Introduction

2 Online Algorithms with ML Advice

3 The Predictive Marker Algorithm

4 Extensions

5 Experiments
In recent years machine learning algorithms have been wildly successful. Yet, in practice:

- Very difficult to deploy
- Prone to errors
Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.
Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.
Machine Learning vs Online Algorithms

Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in practice:
Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.
- Yet, in practice:
  - Are very difficult to deploy
Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in practice:
  - Are very difficult to deploy
  - Are prone to errors
Motivation - Machine Learning

- In recent years machine learning algorithms have been **wildly successful**.

- Yet, in practice:
  - Are very difficult to deploy
  - Are prone to errors
Online algorithms act without any knowledge of the future.
Online algorithms act without any knowledge of the future.

- Are robust against any input
Online algorithms act without any knowledge of the future.
- Are robust against any input
- Have a provable guarantee on performance
Online algorithms act without any knowledge of the future.
  - Are robust against any input
  - Have a provable gaurantee on performance

Yet, overly cautious
### Machine Learning vs Online Algorithms

#### Comparison

<table>
<thead>
<tr>
<th>ML Algorithms</th>
<th>Online Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>attempt to <strong>predict the unknown</strong></td>
<td>act without any knowledge</td>
</tr>
<tr>
<td>susceptible to <strong>large errors</strong></td>
<td><strong>robust against any input</strong></td>
</tr>
<tr>
<td>exploit <strong>patterns</strong></td>
<td><strong>overly cautious</strong></td>
</tr>
</tbody>
</table>
What if we could combine the **predictive power** of ML with the **robustness** of online algorithms?
First example

Example: Binary Search

Textbook problem - Sorted array $A$ of size $n$, and query $q$.
What is the query cost?
First example

Example: Binary Search
Textbook problem - Sorted array $A$ of size $n$, and query $q$. What is the query cost?

ML Approach
First example

Example: Binary Search

Textbook problem - Sorted array $A$ of size $n$, and query $q$. What is the query cost?

ML Approach

- train a classifier $h(q)$ to predict $t(q)$. 
Example: Binary Search

Textbook problem - Sorted array $A$ of size $n$, and query $q$. What is the query cost?

ML Approach

- train a classifier $h(q)$ to predict $t(q)$.
- How can we use such a classifier?
$t(q)$
\[ \epsilon_q = |h(q) - t(q)| \]
The expected cost is $2 \log(\epsilon_q)$. 

$\epsilon_q = |h(q) - t(q)|$
The expected cost is \(2 \cdot \log(\epsilon_q)\).

Is this any good?
The caching problem

- Our focus will be the **caching problem**.
The caching problem

- Our focus will be the **caching problem**.
- Best **determinstic** algorithm for online caching - $\Theta(k)$ competitive ratio.
Our focus will be the **caching problem**.

- Best **deterministic** algorithm for online caching - $\Theta(k)$ competitive ratio
- Best **randomized** algorithm - $\Theta(\log k)$
The caching problem

- Our focus will be the **caching problem**.
- Best **deterministic** algorithm for online caching - $\Theta(k)$ competitive ratio
- Best **randomized** algorithm - $\Theta(\log k)$
  - In reality, observed competitive ratio is much lower.
Our focus will be the **caching problem**.

- **Best deterministic** algorithm for online caching - $\Theta(k)$ competitive ratio
- **Best randomized** algorithm - $\Theta(\log k)$
  - In reality, observed competitive ratio is much lower.

The **machine-learning assisted algorithm** reaches a competitive ratio of $2 + O\left(\min\left(\sqrt{e}, \log k\right)\right)$. 
1 Introduction

2 Online Algorithms with ML Advice

3 The Predictive Marker Algorithm

4 Extensions

5 Experiments
To achieve our results we have to define the playing ground for a **new genre of algorithms**: competitive algorithms with machine learning advice.
Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$
ML scenarios consist of:

- Feature space - \( \mathcal{X} \), and labels - \( \mathcal{Y} \)
- hypothesis \( h : \mathcal{X} \rightarrow \mathcal{Y} \)
- loss function: \( \ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0} \)
- The loss can be: absolute \( (\ell_1 (y, \hat{y}) = |y - \hat{y}|) \), squared \( (\ell_2 (y, \hat{y}) = (y - \hat{y})^2 ) \), or generally: \( \ell_c (y, \hat{y}) = 1_{y \neq \hat{y}} \)
Preliminaries

- **ML scenarios** consist of:
  - Feature space \(- \mathcal{X}\), and labels \(- \mathcal{Y}\)
  - hypothesis \(h: \mathcal{X} \rightarrow \mathcal{Y}\)
  - loss function: \(\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}\)
  - The loss can be: absolute \((\ell_1(y, \hat{y}) = |y - \hat{y}|)\), squared \((\ell_2(y, \hat{y}) = (y - \hat{y})^2)\), or generally: \(\ell_c(y, \hat{y}) = 1_{y \neq \hat{y}}\)

- **Online scenarios** consist of an algorithm \(A\) and sequences \(\sigma\).
Preliminaries

- **ML scenarios** consist of:
  - Feature space - $\mathcal{X}$, and labels - $\mathcal{Y}$
  - hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$
  - loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$
  - The loss can be: absolute ($\ell_1 (y, \hat{y}) = |y - \hat{y}|$), squared ($\ell_2 (y, \hat{y}) = (y - \hat{y})^2$), or generally: $\ell_c (y, \hat{y}) = 1_{y \neq \hat{y}}$

- **Online scenarios** consist of an algorithm $\mathcal{A}$ and sequences $\sigma$.

  - $\mathcal{A}$ has competitive ratio $\text{CR}$ if for every $\sigma$:
    - $\text{cost}_\mathcal{A} (\sigma) \leq \text{CR} \cdot \text{OPT} (\sigma)$
The Online ML-Assisted Framework

Now we can define the combined framework.
Now we can define the combined framework.

- We have a **universe** $\mathcal{Z}$ and **feature** space $\mathcal{X}$
Now we can define the combined framework.

We have a universe $\mathcal{Z}$ and feature space $\mathcal{X}$

The input is a sequence of items $\sigma = (\sigma_1, \sigma_2, \ldots)$
Each item $\sigma_i$ is associated with an element $z_i \in \mathcal{Z}$ and with features $x_i \in \mathcal{X}$
Each item $\sigma_i$ also has a label $y_i \in \mathcal{Y}$
The Online ML-Assisted Framework

- Each item $\sigma_i$ also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \rightarrow \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find $y_i$
Each item $\sigma_i$ also has a label $y_i \in \mathcal{Y}$

A predictor $h : \mathcal{X} \rightarrow \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find $y_i$

- The total loss of $h$ on $\sigma$ is:

$$\eta_\ell(h, \sigma) = \sum_i \ell(h(\sigma_i), y_i)$$
The Online ML-Assisted Framework

- Each item $\sigma_i$ also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \rightarrow \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find $y_i$
  - The total loss of $h$ on $\sigma$ is:

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

**Question**

How can we define $h$ to have a general accuracy of $\epsilon$?
The Online ML-Assisted Framework

- Each item $\sigma_i$ also has a label $y_i \in \mathcal{Y}$
- A predictor $h : \mathcal{X} \rightarrow \mathcal{Y}$ returns $h(\sigma_i)$, attempts to find $y_i$
  - The total loss of $h$ on $\sigma$ is:

\[
\eta_{\ell}(h, \sigma) = \sum_i \ell(h(\sigma_i), y_i)
\]

**Question**
How can we define $h$ to have a general accuracy of $\epsilon$?

**Definition**
We say that $h$ is $\epsilon$-accurate if for every $\sigma$,
\[
\eta_{\ell}(h, \sigma) \leq \epsilon \cdot \text{OPT}(\sigma).
\]
The Online ML-Assisted Framework

Define $cr_A(e)$ to be the competitive ratio of algorithm $A$ that uses any predictor $h$ that is $e$-accurate.
Define $CR_A(\epsilon)$ to be the **competitive ratio** of algorithm $A$ that uses any predictor $h$ that is $\epsilon$-accurate.
The Online ML-Assisted Framework

Define $\text{CR}_A(\epsilon)$ to be the competitive ratio of algorithm $A$ that uses any predictor $h$ that is $\epsilon$-accurate.
Define $CR_A(\varepsilon)$ to be the **competitive ratio** of algorithm $A$ that uses any predictor $h$ that is $\varepsilon$-accurate.

- What would we like $CR_A(0)$ to be?
The Online ML-Assisted Framework

Define $\text{CR}_A(\epsilon)$ to be the **competitive ratio** of algorithm $A$ that uses any predictor $h$ that is $\epsilon$-accurate.

**Consistency**

$A$ is $\beta$-consistent if $\text{CR}_A(0) = \beta$. 
Define $CR_{A}(\epsilon)$ to be the **competitive ratio** of algorithm $A$ that uses any predictor $h$ that is $\epsilon$-accurate.

**Consistency**

$A$ is $\beta$-consistent if $CR_{A}(0) = \beta$.

**Robustness**

$A$ is $\alpha$-robust for some function $\alpha$ if $CR_{A}(\epsilon) = O(\alpha(\epsilon))$. 
Define $\text{CR}_A(\epsilon)$ to be the **competitive ratio** of algorithm $A$ that uses any predictor $h$ that is $\epsilon$-accurate.

- **Consistency**: $A$ is $\beta$-consistent if $\text{CR}_A(0) = \beta$.

- **Robustness**: $A$ is $\alpha$-robust for some function $\alpha$ if $\text{CR}_A(\epsilon) = O(\alpha(\epsilon))$.

- **Competitiveness**: $A$ is $\gamma$-competitive if $\text{CR}_A(\epsilon) \leq \gamma \cdot \text{OPT}$ for all $\epsilon$. 
The Online ML-Assisted Framework

The aim is to find an algorithm $A$ which simultaneously optimizes all three properties.

$CR_A(\epsilon)$

- Consistent
- Robust
- Competitive

$\epsilon$
The holy grail is an algorithm $\mathcal{A}$ which simultaneously optimizes all three properties.
The Online ML-Assisted Framework

The Caching Scenario

Each $s_i$ is a request, $z_i$ is the requested page, and $x_i$ is features of the request.

Question: What should the labels $Y$ be?

Hint: The optimal caching algorithm is LFD.

$y(s_i)$ will be the next appearance of $z_i$.

$h(s_i)$ will try to predict $y_i$.

$Y = \text{NP}$
The Caching Scenario

- Each $\sigma_i$ is a request
The Caching Scenario

- Each $\sigma_i$ is a request
- $z_i$ is the requested page, and $x_i$ is features of the request.
The Online ML-Assisted Framework

The Caching Scenario

- Each $\sigma_i$ is a request
- $z_i$ is the requested page, and $x_i$ is features of the request.

Question

What should the labels $\mathcal{Y}$ be?

Hint: The optimal caching algorithm is LFD.
The Caching Scenario

- Each $\sigma_i$ is a request
- $z_i$ is the requested page, and $x_i$ is features of the request.

**Question**

What should the labels $\mathcal{Y}$ be?

*Hint:* The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearance of $z_i$. 
The Caching Scenario

- Each $\sigma_i$ is a request
- $z_i$ is the requested page, and $x_i$ is features of the request.

**Question**

What should the labels $\mathcal{Y}$ be?

Hint: The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearance of $z_i$.
- $h(\sigma_i)$ will try to predict $y_i$. 
The Caching Scenario

- Each $\sigma_i$ is a request
- $z_i$ is the requested page, and $x_i$ is features of the request.

**Question**

What should the labels $Y$ be?

**Hint:** The optimal caching algorithm is LFD.

- $y(\sigma_i)$ will be the next appearance of $z_i$.
- $h(\sigma_i)$ will try to predict $y_i$.
- $Y = \mathbb{N}^+$
The Online ML-Assisted Framework

The Caching Scenario

Example:

\[ y: a \ b \ c \ b \ a \ a \ c \ d \ldots \]
The Caching Scenario

Example:

\[ y: a b c b a a c d \ldots \]
The Caching Scenario

Example:

\[ y: a b c b a a c d \ldots \]

\[ h: a b c b a a c d \ldots \]
1. Introduction

2. Online Algorithms with ML Advice

3. The Predictive Marker Algorithm

4. Extensions

5. Experiments
Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can’t we just use it?
Initial Attempts

Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can’t we just use it?

Lemma

Define $\mathcal{B}$ the algorithm that blindly follows an $\epsilon$-accurate predictor. Then $\mathcal{B}$ has a competitive ratio $\text{CR}_\mathcal{B}(\epsilon) = \Omega(\epsilon)$. 
Initial Attempts

Attempt #1 - Blindly Following the Predictor

• If the predictor is good, can’t we just use it?

Lemma

Define $B$ the algorithm that blindly follows an $\epsilon$-accurate predictor. Then $B$ has a competitive ratio $CR_B(\epsilon) = \Omega(\epsilon)$.

• Assume $k = 2$ and three elements, $a, b, c$. 
Initial Attempts

Attempt #1 - Blindly Following the Predictor

- If the predictor is good, can’t we just use it?

**Lemma**

*Define $B$ the algorithm that blindly follows an $\epsilon$-accurate predictor. Then $B$ has a competitive ratio $\text{CR}_B(\epsilon) = \Omega(\epsilon)$.*

- Assume $k = 2$ and three elements, $a, b, c$.
- Example follows.
a b c b c ... b c
Predictor $h$: true, except $h(\sigma_1) = 2$

$a$ $b$ $c$ $b$ $c$ ... $b$ $c$
Predictor $h$: true, except $h(\sigma_1) = 2$

$\sigma_1$ b c b c ... b c
Predictor $h$: true, except $h(\sigma_1) = 2$

$a \ b \ c \ b \ c \ \ldots \ b \ c$
Predictor $h$: true, except $h(\sigma_1) = 2$

\[ a \quad c \quad a \ b \ c \ b \ c \ldots \ b \ c \]
Predictor $h$: true, except $h(\sigma_1) = 2$

```
  a  c  a b c b c ... b c
```
Predictor $h$: true, except $h(\sigma_1) = 2$
Predictor $h$: true, except $h(\sigma_1) = 2$
Predictor $h$: true, except $h(\sigma_1) = 2$
Predictor $h$: true, except $h(\sigma_1) = 2$
Predictor $h$: true, except $h(\sigma_1) = 2$
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum:
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum: $OPT = 1$
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum: $OPT = 1$

Performance of $B$: 
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: $\epsilon$
Predictor $h$: true, except $h(\sigma_1) = 2$

\[ a \quad c \quad a \ b \ c \ b \ c \ldots \ b \ c \]

Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: $\epsilon$

Absolute Loss: $\eta(h, \sigma) = \epsilon$
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum: $OPT = 1$

Performance of $\mathcal{B}$: $\epsilon$

Absolute Loss: $\eta(h, \sigma) = \epsilon \leq \epsilon \cdot OPT$
Predictor $h$: true, except $h(\sigma_1) = 2$

Offline Optimum: \( OPT = 1 \)

Performance of $\mathcal{B}$: $\epsilon$

Absolute Loss: \( \eta(h, \sigma) = \epsilon \leq \epsilon \cdot OPT \)

Competitive ratio: \( CR_{\mathcal{B}}(\epsilon) = \frac{\epsilon}{1} = \Omega(\epsilon) \)
Initial Attempts

**Attempt #2 - Reacting to Predictor Mistakes**

- We trusted the predictor too much. Can we do better?
We trusted the predictor too much. Can we do better?

**Lemma**

*Define $\mathcal{W}$ the algorithm that follows an $\epsilon$-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio $\text{CR}_{\mathcal{W}}(\epsilon) = \Omega(\epsilon)$.**
Initial Attempts

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

Lemma

Define $\mathcal{W}$ the algorithm that follows an $\epsilon$-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio $CR_{\mathcal{W}}(\epsilon) = \Omega(\epsilon)$.

- Assume $k = 3$ and four elements, $a, b, c, d$. 
Initial Attempts

Attempt #2 - Reacting to Predictor Mistakes

- We trusted the predictor too much. Can we do better?

**Lemma**

Define $\mathcal{W}$ the algorithm that follows an $\epsilon$-accurate predictor, but evicts wrong predictions. Then $\mathcal{W}$ has a competitive ratio 

$$\text{CR}_{\mathcal{W}}(\epsilon) = \Omega(\epsilon).$$

- Assume $k = 3$ and four elements, $a, b, c, d$.
- Example follows.
\[dabcabc...abc\]
Predictor $h$: predict $d$ correctly
   but predict $a, b, c$ two steps too early

\[ d \ a \ b \ c \ a \ b \ c \ldots \ a \ b \ c \]
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

\[ d \, a \, b \, c \, a \, b \, c \ldots a \, b \, c \]
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

$d\ a\ b\ c\ a\ b\ c\ \ldots\ a\ b\ c$
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

\[
\begin{array}{ccc}
\text{a} & \text{b} & \text{c} \\
\end{array}
\begin{array}{cccc}
d & a & b & c \\
a & b & c & \ldots & a & b & c
\end{array}
\]
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

\[
\begin{array}{c}
\downarrow \\
a & b & c & d & a & b & c & a & b & c \ldots a & b & c
\end{array}
\]
Predictor $h$: predict $d$ correctly
    
    but predict $a, b, c$ two steps too early

\[ \downarrow \]

$a$ $b$ $d$ $d$ $a$ $b$ $c$ $a$ $b$ $c$ $\ldots$ $a$ $b$ $c$
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

\[
\begin{array}{c}
    a \quad b \quad d \\
    d \ a \ b \ c \ a \ b \ c \ \ldots \ a \ b \ c
\end{array}
\]
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

c b d d a b c a b c ... a b c
Predictor $h$: predict $d$ correctly
but predict $a, b, c$ two steps too early

d $a$ $b$ $c$ $a$ $b$ $c$ $\ldots$ $a$ $b$ $c$
Predictor \( h \): predict \( d \) correctly
but predict \( a, b, c \) two steps too early

c a d

\[ d \, a \, b \, c \, a \, b \, c \ldots a \, b \, c \]
Initial Attempts

Attempt #3 - Popular Heuristics

- In the examples we saw that there is some element that should have been evicted.
Initial Attempts

**Attempt #3 - Popular Heuristics**

- In the examples we saw that there is some element that **should** have been evicted.
- LRU and FIFO both provide strong heuristics for such cases.
Initial Attempts

**Attempt #3 - Popular Heuristics**

- In the examples we saw that there is some element that **should** have been evicted.
- LRU and FIFO both provide strong heuristics for such cases.
  - However, their **strict** (deterministic) policy leads to weak guarantees.
Reminder - Classic Marking Algorithm

- Recall the classic Marking Algorithm.
abc c d c d a b b e a b

[Diagram with green squares and arrows indicating movement]
abc c d c d a b b e a b

\[ a \rightarrow a \rightarrow a b \]
A clean element is an element that didn’t arrive in phase $r - 1$ and arrives in phase $r$. 

Reminder - Classic Marking Algorithm
Reminder - Classic Marking Algorithm

- A **clean** element is an element that didn’t arrive in phase \( r - 1 \) and arrives in phase \( r \).
- A **stale** element is an element that arrived in phase \( r - 1 \) and also in phase \( r \).
abc
d
CLEAN
Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$. 

Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:
Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

Claim 1
Let $L$ be the number of clean elements in $\sigma$. Then $\text{OPT}(\sigma) \geq \frac{L}{2}$. 
Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

**Claim 1**
Let $L$ be the number of clean elements in $\sigma$. Then $\text{OPT}(\sigma) \geq \frac{L}{2}$.

**Claim 2**
Let $L$ be the number of clean elements in $\sigma$.
Then $\mathbb{E}[\text{MARK}(\sigma)] \leq L \cdot H_k$. 
Reminder - Classic Marking Algorithm

- We saw that the marking algorithm has competitive ratio $O(\log k)$.
- This came from 2 claims:

**Claim 1**

Let $L$ be the number of clean elements in $\sigma$. Then $\text{OPT}(\sigma) \geq \frac{L}{2}$.

**Claim 2**

Let $L$ be the number of clean elements in $\sigma$. Then $\mathbb{E}[\text{MARK} (\sigma)] \leq L \cdot H_k$.

- Combined, we got:
  $$\mathbb{E}[\text{MARK} (\sigma)] \leq L \cdot H_k \leq 2 \log k \cdot \text{OPT}(\sigma).$$
If we’ll use the marking algorithm, we’ll gain $O(\log k)$ competitive ratio. How can we improve that?
If we'll use the marking algorithm, we’ll gain $O(\log k)$ competitive ratio. How can we improve that?

The natural thing to do then is to **break ties in eviction** using the predictor.
Chains
Chains, Explained

```plaintext
abdcdefafceabca
```

```plaintext
acbd
```
Chains, Explained
Chains, Explained

a b d c d e f a f c e a b c a

a c b d e e c b d
Chains, Explained
Chains, Explained

[Diagram showing chains and their connections]

[Text diagram showing the sequence and connections]
Chains, Explained

a b d c d e f a f c e a b c a

e a b d e c b d f e c f d a e a f d
Chains, Explained
Chains, Explained

- a → b → d → c → d → e → f → a → f → c → e → a → b → c → a
- a → c → b → d → e → c → b → d → e → c → f → d → e → a → f → d → e → a → f → c
Chains, Explained
Chains, Explained

\[ \omega_1 \]
Chains, Explained
Chains, Explained
Chains, Explained

\[ \omega_1 = 4 \]
\[ \omega_2 = 2 \]

\[ n_1 = 4 \]
\[ n_2 = 2 \]
Now to the algorithm:

\[ \mathcal{PM} \] – *Predictive Marker*
Now to the algorithm:

\[ \mathcal{PM} \text{ – Predictive Marker} \]

- At each phase, unmark all elements and save them as potentially stale.
Now to the algorithm:

**PM – Predictive Marker**

- At each phase, unmark all elements and save them as potentially stale.
- In a cache miss for $z_i$:
The Predictive Marker Algorithm

Now to the algorithm:

**PM – Predictive Marker**

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for \( z_i \):
  - If the element \( z_i \) is **clean**, create a new chain.
The Predictive Marker Algorithm

Now to the algorithm:

\[
\text{PM} - \text{Predictive Marker}
\]

- At each phase, unmark all elements and save them as potentially \textit{stale}.
- In a cache miss for \( z_i \):
  - If the element \( z_i \) is \textbf{clean}, create a new chain.
  - If \( z_i \) is \textbf{stale}, find its chain \( z_i = \omega_c \).
Now to the algorithm:

\[ \mathcal{PM} \rightarrow \text{Predictive Marker} \]

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for \( z_i \):
  - If the element \( z_i \) is **clean**, create a new chain.
  - If \( z_i \) is **stale**, find its chain \( z_i = \omega_c \).
    - If the chain length is \( n_c \leq H_k \):
      - evict unmarked element with highest predicted time \( e \).
The Predictive Marker Algorithm

Now to the algorithm:

\[\text{PM} \rightarrow \text{Predictive Marker}\]

- At each phase, unmark all elements and save them as potentially \textit{stale}.
- In a cache miss for \(z_i\):
  - If the element \(z_i\) is \textit{clean}, create a new chain.
  - If \(z_i\) is \textit{stale}, find its chain \(z_i = \omega_c\).
    - If the chain length is \(n_c \leq H_k\):
      - evict unmarked element with highest predicted time \(e\).
    - Else:
      - evict random unmarked element \(e\).
Now to the algorithm:

\[ \mathcal{PM} – Predictive Marker \]

- At each phase, unmark all elements and save them as potentially **stale**.
- In a cache miss for \( z_i \):
  - If the element \( z_i \) is **clean**, create a new chain.
  - If \( z_i \) is **stale**, find its chain \( z_i = \omega_c \).
    - If the chain length is \( n_c \leq H_k \):
      - evict unmarked element with highest predicted time \( e \).
    - Else:
      - evict random unmarked element \( e \).
  - Increase the chain: \( n_c \leftarrow n_c + 1, \omega_c \leftarrow e. \)
The Predictive Marker Analysis

Main Theorem

Theorem

Consider the caching scenario, with the prediction model $\mathcal{H}$ and the loss function $\ell_1$.

The competitive ratio is of the $\epsilon$-assisted Predictive Marker Algorithm $\mathcal{PM}$ is bounded by:

$$CR_{\mathcal{PM}}(\epsilon) \leq 2 \cdot \min \left(1 + \sqrt{5\epsilon}, 2H_k \right)$$
Proof - Chain Lengths

- Every cache miss is a link in a chain.
The Predictive Marker Analysis

Proof - Chain Lengths

- Every cache miss is a link in a chain.
- Long chains means there are many misses. We would like to bound that length.
The Predictive Marker Analysis

Proof - Chain Lengths

- Every cache miss is a link in a chain.
- Long chains means there are many misses. We would like to bound that length.

**Lemma**

*If $h$ has error $\leq \eta$ on chain $\omega_c$, then the chain’s length is bounded by $n_c \leq 1 + \sqrt{5\eta}$.***
\[
h(s_1) \rightarrow s_1 \rightarrow c \rightarrow s_2 \rightarrow h(s_2) \rightarrow s_3
\]
$h(s_1) \geq h(s_2)$
\[ h(s_1) \geq h(s_2) \geq h(s_3) \]
\[ h(s_1) \geq h(s_2) \geq h(s_3) \]
\[ h(s_1) \geq h(s_2) \geq h(s_3) \leq y(s_1) \leq y(s_2) \leq y(s_3) \]
\[ h(s_1) \geq h(P) \geq h(s_2) \geq h(s_3) \leq y(s_2) \leq y(s_3) \leq y(P) \]
\[ h(s_1) \geq h(s_2) \geq h(s_3) \geq \cdots \]
\[ y(s_1) \leq y(s_2) \leq y(s_3) \geq \cdots \]
Proof - Chain Lengths

\[ H, h(s_1), \ell(H,Y), y(s_1) \]
The Predictive Marker Analysis

Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?
Proof - Chain Lengths

If the error $\ell(H, Y)$ is bounded, how long can the chain get?

![Diagram showing chain lengths with variables $H$, $Y$, and $T$.]
Proof - Chain Lengths

If the error $\ell(H, Y)$ is bounded, how long can the chain get?

For such chain of length $T$ we get the error:

$$\ell(H, Y) = 2 \sum_{i=1}^{\frac{T-1}{2}} i = 2 \frac{\frac{T-1}{2} \cdot \frac{T+1}{2}}{2}$$

$$= \frac{T^2 - 1}{4}$$
Proof - Chain Lengths

- If the error $\ell(H, Y)$ is bounded, how long can the chain get?

For such chain of length $T$ we get the error:

$$\ell(H, Y) = 2 \sum_{i=1}^{\frac{T-1}{2}} i = 2 \frac{T-1}{2} \cdot \frac{T+1}{2}$$

$$= \frac{T^2 - 1}{4}$$

- So every chain of length $T$ has at least $\frac{T^2 - 1}{4}$ error.
Proof - Chain Lengths

If the error $\ell (H, Y)$ is bounded, how long can the chain get?

For such chain of length $T$ we get the error:

$$\ell (H, Y) = 2 \sum_{i=1}^{T-1} i = 2 \frac{T-1}{2} \cdot \frac{T+1}{2}$$

$$= \frac{T^2 - 1}{4}$$

So every chain of length $T$ has at least $\frac{T^2 - 1}{4}$ error.

Corollary

If a chain has error $\leq \eta_c$, its length is at most $\sqrt{4\eta_c + 1} \leq \sqrt{5\eta_c}$. 
We want to figure out the total cost.
We want to figure out the total cost.
We want to figure out the total cost.

For a single chain $c$ with error $\eta_c$: 
Proof - Continued

- We want to figure out the total cost.
- For a single chain $c$ with error $\eta_c$:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
Proof - Continued

- We want to figure out the total cost.
- For a single chain $c$ with error $\eta_c$:
  - If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
  - If we did: there are at most $2H_k$ misses.
We want to figure out the total cost.

For a single chain $c$ with error $\eta_c$:
- If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
- If we did: there are at most $2H_k$ misses.

We can bound the evictions from the chain $c$ of the $r$th phase in expectation by $\min(1 + \sqrt{5\eta_{r,c}}, 2H_k)$. 
We want to figure out the total cost.

For a single chain $c$ with error $\eta_c$:

- If we never switched to random evictions: there are at most $1 + \sqrt{5\eta_c}$ cache misses.
- If we did: there are at most $2H_k$ misses.

We can bound the evictions from the chain $c$ of the $r$th phase in expectation by $\min(1 + \sqrt{5\eta_{r,c}}, 2H_k)$.

So the total cost is bounded by

$$\text{cost}_{PM}(e) \leq \sum_{r,c} \min(1 + \sqrt{5\eta_{r,c}}, 2H_k) \leq ?$$
Proof.

Let $L$ be the number of clean elements ($=$ number of chains).
Proof - Continued

Proof.

Let $L$ be the number of clean elements (≡ number of chains).
Proof. 

Let $L$ be the number of clean elements (equal to number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$. 

Proof.

Let $L$ be the number of clean elements (i.e., number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min (1 + \sqrt{5\eta_c}, 2H_k)$. 

Proof. 

Let $L$ be the number of clean elements (i.e., number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min (1 + \sqrt{5\eta_c}, 2H_k)$. 

Which means $CR_{PM}(e) \leq 2 \cdot \min (1 + \sqrt{5\eta_c}, 2H_k)$.
Proof.

Let $L$ be the number of clean elements ( = number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min (1 + \sqrt{5\eta}, 2H_k)$.

- Since $\sqrt{\cdot}$ and $\min (\cdot)$ are both concave:
Proof.

Let $L$ be the number of clean elements (= number of chains). The total error is $\eta = e \cdot \text{OPT}$. We want to know how big is $\sum_{r,c} \min (1 + \sqrt{5\eta_c}, 2H_k)$.

- Since $\sqrt{\cdot}$ and $\min (\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.
Proof.

Let $L$ be the number of clean elements (\(=\) number of chains). The total error is $\eta = e \cdot \text{OPT}$. We want to know how big is $\sum_{r,c} \min (1 + \sqrt{5\eta_c}, 2H_k)$.

- Since $\sqrt{\cdot}$ and $\min (\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.
- The total length of all chains is then $L \cdot \min \left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k \right)$. 

By Lemma 1, $L \leq \text{OPT} (s)$. Trivially, $\text{OPT} (s) \leq L$. So:

$$\text{cost}_{PM} (e) (s) \leq 2 \cdot \text{OPT} (s) \cdot \min \left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k \right).$$

Which means $CR_{PM} (e) \leq 2 \cdot \min \left(1 + \sqrt{5\frac{\eta}{L}}, 2\log k \right)$. 

Proof.

Let $L$ be the number of clean elements (= number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$. We want to know how big is $\sum_{r,c} \min \left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.

- The total length of all chains is then $L \cdot \min \left(1 + \sqrt{\frac{5\eta}{L}}, 2H_k\right)$.

- By Lemma 1, $\frac{L}{2} \leq \text{OPT}(\sigma)$. Trivially, $\text{OPT}(\sigma) \leq L$. 
Proof.

Let $L$ be the number of clean elements ($= $ number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min(1 + \sqrt{5\eta_c}, 2H_k)$.

- Since $\sqrt{\cdot}$ and $\min(\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.

- The total length of all chains is then $L \cdot \min \left(1 + \sqrt{\frac{5\eta}{L}}, 2H_k \right)$.

- By Lemma 1, $\frac{L}{2} \leq \text{OPT} (\sigma)$. Trivially, $\text{OPT} (\sigma) \leq L$.

- So: $\text{cost}_{P_M}(\epsilon) (\sigma) \leq 2 \cdot \text{OPT} (\sigma) \cdot \min \left(1 + \sqrt{\frac{\eta}{\text{OPT}(\sigma)}}, 2H_k \right)$. 
Proof.

Let $L$ be the number of clean elements (number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min \left(1 + \sqrt{5\eta_c}, 2H_k\right)$.

- Since $\sqrt{\cdot}$ and $\min (\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.

- The total length of all chains is then $L \cdot \min \left(1 + \sqrt{5\frac{\eta}{L}}, 2H_k\right)$.

- By Lemma 1, $\frac{L}{2} \leq \text{OPT} (\sigma)$. Trivially, $\text{OPT} (\sigma) \leq L$.

- So: $\text{cost}_{\text{PM}}(\epsilon) (\sigma) \leq 2 \cdot \text{OPT} (\sigma) \cdot \min \left(1 + \sqrt{5\frac{\eta}{\text{OPT}(\sigma)}}, 2H_k\right)$. 
Proof.

Let $L$ be the number of clean elements (= number of chains). The total error is $\eta = \epsilon \cdot \text{OPT}$.

We want to know how big is $\sum_{r,c} \min \left( 1 + \sqrt{5\eta_c}, 2H_k \right)$.

- Since $\sqrt{\cdot}$ and $\min (\cdot)$ are both concave:
  - the way to maximize the expression is to divide the error $\eta$ equally across all chains, with $\frac{\eta}{L}$ each.

- The total length of all chains is then $L \cdot \min \left( 1 + \sqrt{5\frac{\eta}{L}}, 2H_k \right)$.

- By Lemma 1, $\frac{L}{2} \leq \text{OPT} (\sigma)$. Trivially, $\text{OPT} (\sigma) \leq L$.

- So: $\text{cost}_{P,M}(\epsilon) (\sigma) \leq 2 \cdot \text{OPT} (\sigma) \cdot \min \left( 1 + \sqrt{5\frac{\eta}{\text{OPT}(\sigma)}}, 2H_k \right)$.

Which means $CR_{P,M} (\epsilon) \leq 2 \cdot \min \left( 1 + \sqrt{5\epsilon}, 2 \log k \right)$. $\square$
Tightness of analysis

**Theorem (without proof)**

Any deterministic $e$-assisted marking algorithm $A$, that only uses the predictor in tie-breaking among unmarked elements in a deterministic fashion, has a competitive ratio of $\text{cr}_A(e) = W_{\min \{p \in \mathbb{N}, k \}}$. 

The Predictive Marker Analysis

Tightness of analysis

**Theorem (without proof)**

Any deterministic $\epsilon$-assisted marking algorithm $A$, that only uses the predictor in tie-breaking among unmarked elements in a deterministic fashion, has a competitive ratio of

$$\text{CR}_A (\epsilon) = \Omega \left( \min \left( \sqrt{\epsilon}, k \right) \right)$$
1 Introduction

2 Online Algorithms with ML Advice

3 The Predictive Marker Algorithm

4 Extensions

5 Experiments
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
- What about $\gamma H_k$ for some $\gamma > 0$?
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
- What about $\gamma H_k$ for some $\gamma > 0$?

**Theorem**

Suppose that, for $\gamma > 0$, the algorithm uses $\gamma H_k$ as switching point. Denote this algorithm by $\mathcal{PM}(\gamma)$. 

What does a low $g$ mean? What does a high $g$ mean?
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
- What about $\gamma H_k$ for some $\gamma > 0$?

Theorem

Suppose that, for $\gamma > 0$, the algorithm uses $\gamma H_k$ as switching point. denote this algorithm by $\mathcal{PM}(\gamma)$. Then, the competitive ratio of $\epsilon$-assisted $\epsilon$-assisted $\mathcal{PM}(\gamma)$ is bounded by:

$$CR_{\mathcal{PM}(\gamma), \ell}(\epsilon) \leq 2 \min \left( 1 + \frac{1 + \gamma}{\gamma} \sqrt{5\epsilon}, (1 + \gamma) H_k, k \right)$$
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
- What about $\gamma H_k$ for some $\gamma > 0$?

**Theorem**

Suppose that, for $\gamma > 0$, the algorithm uses $\gamma H_k$ as switching point. denote this algorithm by $\mathcal{P} \mathcal{M}(\gamma)$.

Then, the competitive ratio of $\epsilon$-assisted $\epsilon$-assisted $\mathcal{P} \mathcal{M}(\gamma)$ is bounded by:

$$CR_{\mathcal{P} \mathcal{M}(\gamma), \ell}(\epsilon) \leq 2 \min \left(1 + \frac{1 + \gamma}{\gamma} \sqrt{5\epsilon}, (1 + \gamma) H_k, k\right)$$

- What does a low $\gamma$ mean?
Free parameter in the algorithm

- So far, we chose $H_k$ as a switching point for the algorithm.
- What about $\gamma H_k$ for some $\gamma > 0$?

**Theorem**

Suppose that, for $\gamma > 0$, the algorithm uses $\gamma H_k$ as switching point. Denote this algorithm by $PM(\gamma)$.

Then, the competitive ratio of $\epsilon$-assisted $\epsilon$-assisted $PM(\gamma)$ is bounded by:

$$CR_{PM(\gamma), \ell}(\epsilon) \leq 2 \min \left( 1 + \frac{1 + \gamma}{\gamma} \sqrt{5\epsilon}, (1 + \gamma) H_k, k \right)$$

- What does a low $\gamma$ mean?
- What does a high $\gamma$ mean?
Robustifying LRU

- How can we use LRU in the predictive marker setting?
Robustifying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$. 
Robustifiying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

**Fact**

*If we never switched to random evictions, this is exactly LRU.*
Robustifying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

**Fact**

*If we never switched to random evictions, this is exactly LRU.*

- LRU is deterministic and therefore has only a bound of $\Theta(k)$, but is very good in practice.
Robustifying LRU

- How can we use LRU in the predictive marker setting?
- Define $h(\sigma_i) = -i$.

**Fact**

*If we never switched to random evictions, this is exactly LRU.*

- LRU is deterministic and therefore has only a bound of $\Theta(k)$, but is very good in practice.
- This new setting reduces the analysis of LRU from $\Theta(k)$ to $O(\log k)$, while still exploiting its predictive power.
Comparing results

- Let's take a look at some real world datasets:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num Sequences</th>
<th>Sequence Length</th>
<th>Unique Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK</td>
<td>100</td>
<td>2,101</td>
<td>67– 800</td>
</tr>
<tr>
<td>Citi</td>
<td>24</td>
<td>25,000</td>
<td>593 – 719</td>
</tr>
</tbody>
</table>
Comparing results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Competitive Ratio on BK</th>
<th>Competitive Ratio on Citi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blind Oracle</td>
<td>2.049</td>
<td>2.023</td>
</tr>
<tr>
<td>LRU</td>
<td>1.280</td>
<td>1.859</td>
</tr>
<tr>
<td>Marker</td>
<td>1.310</td>
<td>1.869</td>
</tr>
<tr>
<td>Predictive Marker</td>
<td>1.266</td>
<td>1.810</td>
</tr>
</tbody>
</table>
We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.
We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.

We saw how we can maintain **robustness** while exploiting predictions to improve **competitiveness**.
We saw a general framework for combining online algorithms with guidance of machine learning - OMLA.

We saw how we can maintain robustness while exploiting predictions to improve competitiveness.

We discussed the analysis of robustness vs competitiveness and looked at real-world examples.
• We saw a general framework for combining **online algorithms** with guidance of **machine learning** - OMLA.

• We saw how we can maintain **robustness** while exploiting predictions to improve **competitiveness**.

• We discussed the analysis of **robustness vs competitiveness** and looked at real-world examples.

• Thank you!