Learning-Augmented Online Algorithms

Bertrand Simon

IN2P3 Computing Center / CNRS, Villeurbanne

Journées CALAMAR – January 2024
Motivating example: binary search

\[ n \text{ elements} \]

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>11</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>25</td>
<td>30</td>
<td>36</td>
<td>40</td>
<td>43</td>
<td>46</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>53</td>
<td>54</td>
<td>56</td>
<td>59</td>
<td>60</td>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ q = 16 \]
Motivating example: binary search

\[ n \text{ elements} \]

\[
\begin{array}{cccccccccccccccc}
8 & 11 & 14 & 16 & 18 & 25 & 30 & 36 & 40 & 43 & 46 & 49 & 50 & 53 & 54 & 56 & 59 & 60 & 63 \\
\end{array}
\]

\[ q = 16 \]
Motivating example: binary search

$n$ elements

\[
\begin{array}{cccccccccccccccc}
8 & 11 & 14 & 16 & 18 & 25 & 30 & 36 & 40 & 43 & 46 & 49 & 50 & 53 & 54 & 56 & 59 & 60 & 63 \\
\end{array}
\]

$q = 16$

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$

Classic: $\Theta(\log n)$

Practical applications

[KraskaBeutalChiDeanPolyzotis'18]
Motivating example: binary search

$n$ elements

$q = 16$
Motivating example: binary search

Consider an array of $n$ elements: $8, 11, 14, 16, 18, 25, 30, 36, 40, 43, 46, 49, 50, 53, 54, 56, 59, 60, 63$.

Let's search for the element $q = 16$. In a classic binary search, the prediction is $h(q)$, and the error is $\eta = |h(q) - \text{index}(q)|$.

The prediction can be optimized with learning, leading to $\Theta(\log \eta)$ predictions in practice.

The algorithm is described in [KraskaBeutalChiDeanPolyzotis'18].
Motivating example: binary search

\[
\begin{array}{ccccccccc}
8 & 11 & 14 & 16 & 18 & 25 & 30 & 36 & 40 & 43 & 46 & 49 & 50 & 53 & 54 & 56 & 59 & 60 & 63 \\
\end{array}
\]

\( n \) elements

\( q = 16 \)
Motivating example: binary search

### Motivating Example: Binary Search

Consider an array of *n* elements:

| 8 | 11 | 14 | 16 | 18 | 25 | 30 | 36 | 40 | 43 | 46 | 49 | 50 | 53 | 54 | 56 | 59 | 60 | 63 |

Let `q = 16`.

**Prediction:** position `h(q)`

**Error:**

\[
\eta = |h(q) - \text{index}(q)|
\]
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$  
Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

$n$ elements

| 8 | 11 | 14 | 16 | 18 | 25 | 30 | 36 | 40 | 43 | 46 | 49 | 50 | 53 | 54 | 56 | 59 | 60 | 63 |

$q = 16$

Prediction: position $h(q)$  
Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

\[ n \text{ elements} \]

| 8 | 11 | 14 | 16 | 18 | 25 | 30 | 36 | 40 | 43 | 46 | 49 | 50 | 53 | 54 | 56 | 59 | 60 | 63 |
\[ q = 16 \]

Prediction: position \( h(q) \)  
Error: \( \eta = |h(q) - \text{index}(q)| \)
Motivating example: binary search

Prediction: position $h(q)$  
Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$  
Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$
Motivating example: binary search

| 8 | 11 | 14 | 16 | 18 | 25 | 30 | 36 | 40 | 43 | 46 | 49 | 50 | 53 | 54 | 56 | 59 | 60 | 63 |

$n$ elements

$q = 16$

Prediction: position $h(q)$

Error: $\eta = |h(q) - \text{index}(q)|$

Classic: $\Theta(\log n)$

Predictions $\Theta(\log \eta)$

Practical applications [KraskaBeutalChiDeanPolyzotis’18]
Properties we seek

- competitive ratio / complexity / …

Algorithms are oblivious to $\eta$

Prediction $h$ should be *learnable*, e.g., compact
Properties we seek

- Competitive ratio / complexity / ...
- Consistency
- Robustness

Algorithms are oblivious to $\eta$

Prediction $h$ should be learnable, e.g., compact
Properties we seek

- Competitive ratio / complexity / ...

Algorithms are oblivious to $\eta$

Prediction $h$ should be *learnable*, e.g., compact
Properties we seek

- Competitive ratio /
- Complexity /
- \( \eta \)

Algorithms are oblivious to \( \eta \)

Prediction \( h \) should be *learnable*, e.g., compact
Properties we seek

- Competitive ratio / complexity / …
- Prediction error ($\eta$)
- Competitive ratio / complexity / …
- Prediction error ($\eta$)
- Robustness

Names vary in the literature

Algorithms are oblivious to $\eta$

Prediction $h$ should be *learnable*, e.g., compact
"Classic" Beyond worst-case analysis

Future instance: \( X_1; X_2; X_3; X_4; X_5; \ldots \)

- **Lookahead**
  \( X_1 = 5 \)

- **Semi-online**
  \( \sum_i X_i = 30 \)

- **Random arrival**
  Advice: 1101110

- **Stochastic input**
  \( X_i \sim \mathcal{N}(10, 5) \)

- **Robust analysis**
  \( X_1 = 5 \pm 2, X_2 = 7 \pm 3, \ldots \)

😊 Strong assumptions, needs some perfect information (oracle)

---

**HERE:** no assumption on the predictor allows plug-and-play predictors

Bertrand Simon | Learning-Augmented Online Algorithms
--- | ---
Landscape preview

100s publications since 2018
https://algorithms-with-predictions.github.io/
Most common framework used

Input

Input

Input

Input

Input

Online algorithm $A$

time

Objective: "minimize" competitive ratio

$c_A(\eta) = \max \frac{\text{cost}_A(I)}{\text{OPT}(I)}$

Consistency $c_A(0)$

Robustness $c_A(\infty)$

Smoothness "slope" of $c_A(\eta)$
Most common framework used

Input
Input
Input
Input
Input

error( ) = \eta

Online algorithm A

time
Most common framework used

Online algorithm $A$

$$\text{error}(\text{Input}) = \eta$$

Objective: "minimize" competitive ratio

$$c_A(\eta) = \max_{I} \frac{\text{cost}_A(I)}{\text{OPT}(I)}$$

Consistency

Robustness

Smoothness: "slope" of $c_A(\eta)$
Most common framework used

**Objective:** “minimize” competitive ratio

\[ c_A(\eta) = \max_i \frac{\text{cost}_A(i)}{\text{OPT}(i)} \]

- **Consistency**
  \[ c_A(0) \]

- **Robustness**
  \[ c_A(\infty) \]

- **Smoothness**
  “slope” of \[ c_A(\eta) \]
Outline

1. Introduction
2. Ski rental & extensions
3. Preview of Paging and Graph Algorithms
4. Conclusion
First example: Ski rental

Cost to buy skis: $b$

Daily rent price: $1$

$\times$ ? # ski days (unknown)

Online algorithm: minimize $\max \frac{\text{cost}}{\text{OPT}}$

Best deterministic algo: buy at day $\approx b$

Worst-case = stop after day $b$:

$\text{OPT} = b$

$\text{cost} = 2b$

$\Rightarrow$ competitive ratio $= 2$
First example: Ski rental

What should $h$ predict?

- 😞 $h \rightarrow 0/1$: rent or buy? cannot measure $\eta$
- ☝️ $h \rightarrow x$ with $\eta = |h - x|$

What should the algorithm do?

Cost to buy skis: $b$

Daily rent price: $1$

# ski days (unknown): $x$
First example: Ski rental

What should $h$ predict?

- 🙁 $h \rightarrow 0/1$: rent or buy? cannot measure $\eta$
- ☺ $h \rightarrow x$ with $\eta = |h - x|$

What should the algorithm do?

**Naive**: if $h \geq b$ then buy on day 1 else rent forever
First example: Ski rental

What should \( h \) predict?

- ☹ \( h \rightarrow 0/1: \) rent or buy? cannot measure \( \eta \)
- ☻ \( h \rightarrow x \) with \( \eta = |h - x| \)

What should the algorithm do?

**Naive**: if \( h \geq b \) then buy on day 1 else rent forever

**Lemma**

The competitive ratio of **Naive** is \( 1 + \eta / \text{Opt} \).
A robust algorithm for Ski Rental  [Purohit Switkina Kumar’18]

Intuition: if $h < x$, we should not buy at day 1

How long should we rent? depends on the predictor’s “trustworthiness”

$\lambda = 0$ Consistent  $\lambda = 1$ Robust

$\text{SkiPred}(\lambda)$:  
- If $h \geq b$: rent $\lceil \lambda b \rceil$ days
- Else: rent $\lceil b/\lambda \rceil$ days

Theorem

$\text{SkiPred}\left(\frac{1}{2}\right)$ is:  
\[
\min \left(3, \frac{5}{3} + 2 \cdot \frac{\eta}{\text{OPT}} \right) - \text{competitive}.
\]
A robust algorithm for Ski Rental  [Purohit Switkina Kumar’18]

Intuition: if $h \leq x$, we should not buy at day 1

How long should we rent? depends on the predictor’s “trustworthiness”

Consistent

$\lambda = 0$

Robust

$\lambda = 1$

$\text{SkiPred}(\lambda)$:  

- If $h \geq b$: rent $\lceil \lambda b \rceil$ days
- Else: rent $\lceil b/\lambda \rceil$ days

SkiPred$(\frac{1}{2})$ is: $\min \left( 3, 1.5 + 2 \cdot \frac{\eta}{\text{OPT}} \right)$ - competitive.
A robust algorithm for Ski Rental  [Purohit Switkina Kumar’18]

Intuition: if \( h \geq b \), we should not buy at day 1

How long should we rent? depends on the predictor’s “trustworthiness”

\[
\text{SkiPred}(\lambda): \begin{cases} 
\text{If } h \geq b: \text{ rent } \lceil \lambda b \rceil \text{ days} \\
\text{Else: rent } \lceil b/\lambda \rceil \text{ days}
\end{cases}
\]

\[\text{Theorem} \quad \text{SkiPred}(\lambda) \text{ is: } \min \left( \frac{1+\lambda}{\lambda}, (1+\lambda) + \frac{1}{1-\lambda} \cdot \frac{\eta}{OPT} \right) \text{ - competitive.}\]
Randomized ski rental

Classic randomized ski rental $\rightarrow \frac{e}{e-1} \approx 1.58$-competitive
Randomized ski rental

Classic randomized ski rental $\rightarrow \frac{e}{e-1} \approx 1.58$-competitive

Theorem

There is a $O\left(\min\left(\frac{1}{1-e^\lambda}, \frac{\lambda}{1-e^{-\lambda}} \left(1 + \frac{\eta}{\Omega_{\text{OPT}}}\right)\right)\right)$-competitive algorithm.

e.g., $\lambda = 1/2$
Consistency vs Robustness

Lower bounds:

- Randomized: matches UB [WeiZhang’20]
- Deterministic: LB a bit lower but
  [AngelopoulosDürrJinKamaliRenault’19]
Multi-round ski rental

[Antoniadis Coester Elias Polak Simon ’21]

Dynamic Power Management: shift of focus

- focus first on 2 states
- is “free” over many rounds (experts framework)
Dynamic Power Management: shift of focus

- focus first on 2 states

- is “free” over many rounds (experts framework)
Dynamic Power Management: shift of focus

- focus first on 2 states
- is “free” over many rounds (experts framework)
Multi-round ski rental  [Antoniadis Coester Elias Polak Simon ’21]

Dynamic Power Management: shift of focus

- focus first on 2 states

- is “free” over many rounds (experts framework)
New tradeoff: \( \text{NAIVE} \) vs \( \text{ONLINE} \) in cost = \( \text{NAIVE} \cdot \text{OPT} + \text{ONLINE} \cdot \eta \)

Use the whole prediction

Example for a \( \approx 1.16 \)-consistent, 0.38-smooth solution:
New tradeoff: $\text{vs}$ in cost $= \cdot \text{OPT} + \cdot \eta$

Use the whole prediction

Example for a $\approx 1.16$-consistent, $0.38$-smooth solution:
New tradeoff: $\eta \cdot \text{Opt}$

Use the whole prediction

Example for a $\approx 1.16$-consistent, $0.38$-smooth solution:

1.0 1.1 1.2 1.3 1.4 1.5
0.0 0.2 0.4 0.6 0.8
NAIVE
ONLINE

Pareto frontier: smoothness = $f(\text{consistency})$

1.0 1.1 1.2 1.3 1.4 1.5
0.0 0.2 0.4 0.6 0.8
1.0
0.0
0.2
0.4
0.8
1.2
1.4

Time ($\times b$)

Probability that the algorithm buys before that time
$h = 0.1$
$h = 0.3$
$h = 0.5$
$h = 0.65$
$h = 0.8$
$h = 0.95$
$h = 1.1$
$h = 1.4$
$h = 1.7$

Case 1, $h < 0.5857$
Case 2, $h [0.5857, 1]$
Case 3, $h > 1$

Example of simulation results

4 actual idle states, random instances + predictions

Competitive ratio

Standard online algorithm
Purohit et al.
Our algorithm

Nitrate parameter of the synthetic predictor

1.0 1.1 1.2 1.3 1.4 1.5

Competitive ratio

Noise parameter $\sigma$ of the synthetic predictor

1.0 2.0 4.0 6.0 8.0
Outline

1. Introduction
2. Ski rental & extensions
3. Preview of Paging and Graph Algorithms
4. Conclusion
Paging with predictions

\[ k = 4 \quad \text{misses: } 1 \]

\[ \text{pages } \in \{A, B, \ldots, F\} \]

\[ \text{1} \]

\[ \text{A} \]
Paging with predictions

$k = 4$ misses: 2

pages $\in \{A, B, \ldots, F\}$

\[\begin{array}{c|c}
A & 1 & 2 \\
A & A & B \\
\end{array}\]
Paging with predictions

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

$k = 4$  misses: 2  pages $\in \{A, B, \ldots, F\}$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>
Paging with predictions

k = 4  misses: 3

pages ∈ \{A, B, \ldots, F\}

\[ \begin{array}{c}
\text{A} \\
\text{B} \\
\end{array} \]

1 2 3 4

A B A C
Paging with predictions

$\begin{array}{|c|}
\hline
k = 4 \quad \text{misses: 4} \\
C \\
B \\
A \\
\hline
\end{array}$

$\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
A & B & A & C & D \\
\end{array}$
Paging with predictions

$k = 4$ misses: 5

pages $\in \{A, B, \ldots, F\}$
Paging with predictions

\[ k = 4 \quad \text{misses: 6} \]

\[
\begin{array}{c}
A \\
\hline
B \\
\hline
C \\
\hline
D \\
\hline
E \\
\hline
F
\end{array}
\]

\[
\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
A & B & A & C & D & E & F
\end{array}
\]
Paging with predictions

\[ k = 4 \quad \text{misses: 6} \]

\[
\begin{array}{c}
A \\
B \\
C \\
D \\
E \\
F
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A & B & A & C & D & E & F & A
\end{array}
\]

\[ \text{pages} \in \{A, B, \ldots, F\} \]
Paging with predictions

\[ k = 4 \quad \text{misses: 7} \]

\[
\begin{array}{c}
D \\
F \\
E \\
A \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
A & B & A & C & D & E & F & A & B \\
\end{array}
\]
Paging with predictions

$k = 4$ misses: 7

$$\text{pages } \in \{A, B, \ldots, F\}$$

\[
\begin{array}{c}
A \\
B \\
C \\
D \\
E \\
A
\end{array}
\]

[LykourisVassilvitskii’18]

1 2 3 4 5 6 7 8 9 10

A B A C D E F A B E
Paging with predictions

\[ k = 4 \quad \text{misses: 8} \]

\[
\begin{array}{c}
A \\
B \\
C \\
D \\
E \\
F \\
\end{array}
\]

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
A & B & A & C & D & E & F & A & B & E & F \\
\end{array}
\]
Paging with predictions

$k = 4$  misses: 8

pages $\in \{A, B, \ldots, F\}$

F
B
E
A

1  2  3  4  5  6  7  8  9  10  11

A  B  A  C  D  E  F  A  B  E  F

[LykourisVassilvitskii’18]
Paging with predictions

$k = 4$  misses: 8

\[
\begin{array}{c}
\text{F} \\
\text{B} \\
\text{E} \\
\text{A}
\end{array}
\]

pages $\in \{A, B, \ldots, F\}$

Q: What to predict?

Lookahead (next $q$ requests)

▶️ 😞 useless in the worst case

Strong Lookahead (next requests until $q$ distinct)

▶️ 😞 huge, hard to predict

Next arrival time of the current request

▶️ 😊 compact, enough to compute $\text{OPT}$, arguably learnable

▶️ error $\eta_i$ at round $i$: distance between predicted time and actual time combined error $\eta = \sum \eta_i$.

▶️ $\implies$ get a $\approx \min(\log k, \log \frac{\eta}{\text{OPT}})$-competitive algorithm
Paging with predictions

\[ k = 4 \quad \text{misses: 8} \]

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
| & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\hline
\text{A} & \text{B} & \text{A} & \text{C} & \text{D} & \text{E} & \text{F} & \text{A} & \text{B} & \text{E} & \text{F} \\
\text{F} & & & & & & & & & & & \\
\text{B} & & & & & & & & & & & \\
\text{E} & & & & & & & & & & & \\
\text{A} & & & & & & & & & & & \\
\end{array}
\]

\text{next: 3 9 8 - - 10 11 - - - -}

\text{Q: What to predict?}

\begin{itemize}
\item \textbf{Lookahead (next } q \text{ requests)}
  \begin{itemize}
  \item 😞 useless in the worst case
  \end{itemize}
\end{itemize}

\begin{itemize}
\item \textbf{Strong Lookahead} (next requests until } q \text{ distinct)
  \begin{itemize}
  \item 😞 huge, hard to predict
  \end{itemize}
\end{itemize}

\text{Next arrival time of the current request}

\begin{itemize}
\item 😊 compact, enough to compute } OPT, \text{ arguably learnable
\item error } \eta_i \text{ at round } i : \text{ distance between predicted time and actual time combined error } \eta = \sum \eta_i. \\
\item \implies \text{ get a } \approx \min(\log k, \log \frac{\eta}{OPT})\text{-competitive algorithm}
\end{itemize}
Paging with predictions – Overview of models

Predictions = time of next occurrence of current page

- Lykouris Vassilvitskii (2021 JACM); Rohatgi (SODA 2020); Wei (APPROX/RANDOM 2020)

Predictions = all pages before next occurrence of current page

- Jiang Panigrahi Su (ICALP 2020)

Predictions = state of $\text{OPT}$ (which pages in cache)

- Antoniadis Coester Elias Polak Simon (ICML 2020)

Multiple predictors — time of next occurrence of current page

- Emek Kutten Shi (ITCS 2020)

Prediction queries — obtain next occurrence of any page in cache

- Im Kumar Petety Purohit (ICML 2022)

Succinct predictions = 1 bit of information ($\approx$ to evicted or not)

- Antoniadis Boyar Eliáš Favrholdt Hoeksma Larsen Polak Simon (ICML 2023)
A general error measure for graph algorithms

[Azar Panigrahi Touitou, Online Graph Algorithms with Predictions, SODA 2022]

Error measure for predicting a set of points in a metric space $\eta = (D, \Delta)$:

$D =$ transportation distance ; $\Delta =$ # outliers

**Theorem**

*For the Steiner Tree and Facility Location problems, if the error of the predicted input (resp. set of terminal and set of clients) is $(D, \Delta)$, there is an algorithm of cost at most $O(\log \Delta) \cdot \text{OPT} + O(D)$.***
Faster matching via learned duals

[Dinitz Im Lavastida Moseley Vassilvitskii NeurIPS 2021]
[Chen Silwal Vakilian Zhang ICML 2022]

Theorem

Given a weighted bipartite graph and predicted dual $\hat{\gamma}$, there exists an algorithm that finds a minimum weight perfect matching in time $O(m \sqrt{n} + (m + n \log n) \| y^* - \hat{\gamma} \|_0)$, where $y^*$ is an optimal dual solution.

Main ideas:

- use predicted dual as a warm start for the Hungarian algorithm
- if this dual is not feasible, adapt it
- actually design the algorithm that get the solution faster if the prediction error is small
- show that the duals can be learned from samples of a probabilistic input
Outline

1. Introduction
2. Ski rental & extensions
3. Preview of Paging and Graph Algorithms
4. Conclusion
Conclusion

Take-back messages

- fresh algorithm concepts
- relevant link ML – algorithms

Newer questions

- improve running time
- ensure learnability (e.g., PAC-learnability) \( \eta \approx \text{loss function} \)
- extensive experiments including ML predictors
- multiple predictors
- wrt renowned heuristics?