Online Metric Algorithms with Untrusted Predictions

Antonios Antoniadis[1] Christian Coester[2] Marek Elias[3]
Adam Polak[4] **Bertrand Simon**[5]

1: MPI, Saarbrucken (Germany).
2: CWI, Amsterdam (Netherlands).
3: EPFL, Lausanne (Switzerland).
4: Jagiellonian University, Kraków (Poland).
5: University of Bremen (Germany).

Dagstuhl – February 2020

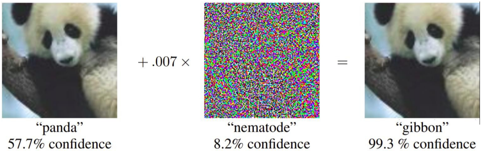
Motivation

Online algorithms

- Guaranteed competitive ratio
- Bad performance on easy instances, overly pessimistic

Machine Learning predictions

- Often relevant information
- No guarantee, can be arbitrarily bad



arxiv.org/abs/1412.6572

Prediction-augmented algorithms

- Target competitive ratio: $O(\ \min\{\ 1 + f(\eta/\textsc{Opt})\ ,\ \mathit{ONLINE}\ \}\ )$

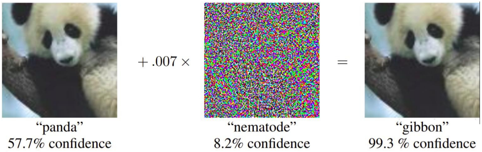
Motivation

Online algorithms

- Guaranteed competitive ratio
- Bad performance on easy instances, overly pessimistic

Machine Learning predictions

- Often relevant information
- No guarantee, can be arbitrarily bad



arxiv.org/abs/1412.6572

Prediction-augmented algorithms

- Target competitive ratio: $O(\ \min\{\ 1 + f(\eta/\textsc{Opt})\ ,\ \mathit{ONLINE}\ \}\ )$

Motivation

Online algorithms

- Guaranteed competitive ratio
- Bad performance on easy instances, overly pessimistic

Machine Learning predictions

- Often relevant information
- No guarantee, can be arbitrarily bad

absolute error η

$\eta = \sum_t \eta_t$

Prediction-augmented algorithms

- Target competitive ratio: $O(\ \min\{\ 1 + f(\eta/\textsc{Opt})\ ,\ \mathit{ONLINE}\ \}\ )$

Prediction-augmented algorithms

Some previously studied problems

- Ski rental: predict #days we will ski [KumarPS'18]
- Non-clairvoyant scheduling: predict processing times [KumarPS'18]
- Restricted assignment: predict machine weights [LattanziMLV'20]
- Caching: predict next arrival time [LykourisV'18, Rohatgi'20]

Issue: lack of generality, predictions tailored to specific problems

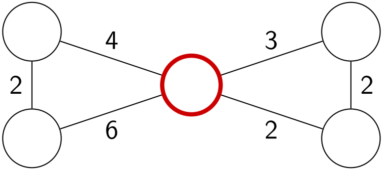
Lemma

Previously used caching predictions are not useful for weighted caching.

$\Longrightarrow$ need for a *more general* prediction setup

The Metrical Task System (MTS) problem

Definition by picture



Round 0
Cost incurred: 0

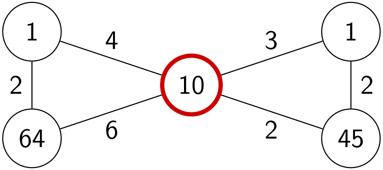
Note: generalizes caching, k-server, convex body chasing...

Prediction setup

- Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t. Error: $\eta = \sum_t d(o_t, p_t)$

The Metrical Task System (MTS) problem

Definition by picture



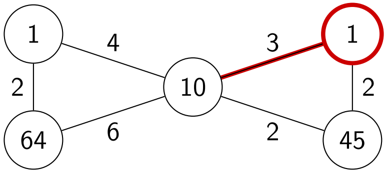
Round 1 before serving
Cost incurred: 0

Note: generalizes caching, k-server, convex body chasing...

Prediction setup

- Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \dots$
- At time t, $p_t :=$ prediction of o_t. Error: $\eta = \sum_t d(o_t, p_t)$

Definition by picture



Round 1 after serving
Cost incurred: 3+1

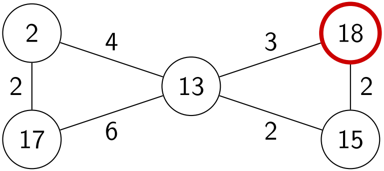
Note: generalizes caching, k-server, convex body chasing...

Prediction setup

- Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t. Error: $\eta = \sum_t d(o_t, p_t)$

The Metrical Task System (MTS) problem

Definition by picture



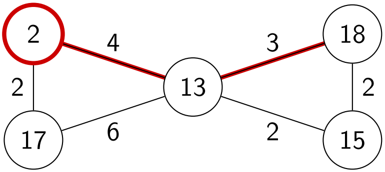
Round 2 before serving
Cost incurred: 3+1

Note: generalizes caching, *k*-server, convex body chasing...

Prediction setup

- Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t. Error: $\eta = \sum_t d(o_t, p_t)$

Definition by picture



Round 2 after serving
Cost incurred: 3+1+7+2

Note: generalizes caching, k-server, convex body chasing…

Prediction setup

- Fix Off: offline algorithm (e.g., Opt), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t. Error: $\eta = \sum_t d(o_t, p_t)$

Algorithm for MTS: RobustFtP

Algorithm subroutine (FtP, Follow the Prediction)

- Go to p_t, except if there is a cheap state nearby
- Proposition: this costs at most $\textsc{Off} \cdot (1 + 4\eta/\textsc{Off})$

Combining online algorithms A and B: $comb(A, B)$ [BlumB'00]

- $E(cost_{comb(A,B)}) \leq (1+\varepsilon) \cdot \min\{E(cost_A), E(cost_B)\}$ asymptotically

Theorem (RobustFtP := $comb(ONLINE, \textsc{FtP})$ competitive ratio)

RobustFtP *is* $O(\min\{1 + \eta/\textsc{Off}\ ,\ ONLINE\})$ *- competitive.*

(Recall: $\eta = \sum_t d(o_t, p_t)$*)*

Lemma (Lower bound)

This is tight for some MTS (i.e., $\eta/\textsc{Off}$ *- dependency).*

Logarithmic error dependency for caching

Issue with RobustFtP: too general, not the best for all MTS

Focus on a specific MTS: the caching problem

- Maintain a cache of k pages, pay 1 per cache miss

Algorithm Trust&Doubt: explaining ~~how it works~~ its name

- "Trust" predictor: evict a page not in its cache
- If an evicted page requested: "doubt" this decision
- Regularly (depending on trustworthiness): "trust" again

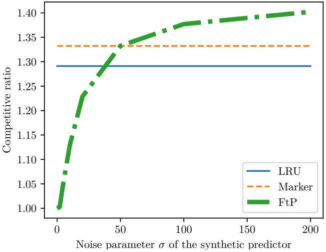
Theorem (Trust&Doubt competitive ratio)

Trust&Doubt *costs at most* $O(\textsc{Off} \cdot \min\{1 + \log \frac{\eta}{\textsc{Off}}, \log k\})$.

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
  deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – *(lower is better)*



Prediction: ground truth + lognorm error

LRU	1.291	
Marker	1.333	
Predictions	PLECO	POPU
FtP	2.081	1.707
L&V [LykourisV'18]	1.340	1.262
LMarker [Rohatgi'20]	1.337	1.264
LNonMarker [Rohatgi'20]	1.339	1.292
RobustFtP	**1.351**	**1.316**
Trust&Doubt	**1.292**	**1.274**

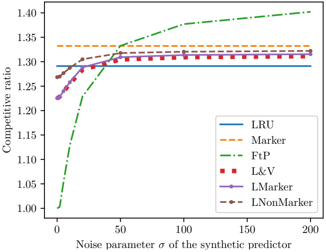
Two *predictors*: simple statistics

Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
  deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – *(lower is better)*



Prediction: ground truth + lognorm error

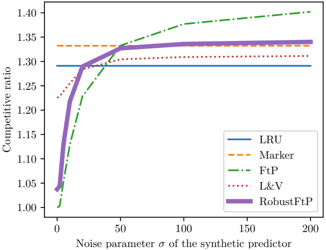
LRU	1.291	
Marker	1.333	
Predictions	PLECO	POPU
FtP	2.081	1.707
L&V [LykourisV'18]	1.340	1.262
LMarker [Rohatgi'20]	1.337	1.264
LNonMarker [Rohatgi'20]	1.339	1.292
RobustFtP	**1.351**	**1.316**
Trust&Doubt	**1.292**	**1.274**

Two *predictors*: simple statistics

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
  deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – *(lower is better)*



Prediction: ground truth + lognorm error

LRU	1.291	
Marker	1.333	
Predictions	PLECO	POPU
FtP	2.081	1.707
L&V [LykourisV'18]	1.340	1.262
LMarker [Rohatgi'20]	1.337	1.264
LNonMarker [Rohatgi'20]	1.339	1.292
RobustFtP	**1.351**	**1.316**
Trust&Doubt	**1.292**	**1.274**

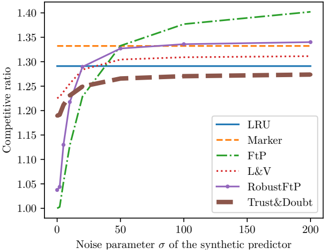
Two *predictors*: simple statistics

Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
  deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – *(lower is better)*



Prediction: ground truth + lognorm error

LRU	1.291	
Marker	1.333	
Predictions	PLECO	POPU
FtP	2.081	1.707
L&V [LykourisV'18]	1.340	1.262
LMarker [Rohatgi'20]	1.337	1.264
LNonMarker [Rohatgi'20]	1.339	1.292
RobustFtP	**1.351**	**1.316**
Trust&Doubt	**1.292**	**1.274**

Two *predictors*: simple statistics

Conclusion

is useful !

- MTS (and beyond): prediction setup and “optimal” algorithm
- Caching: specific algorithm good in theory & practice
- Perspective: other MTS (weighted caching, convex body chasing, k-server)