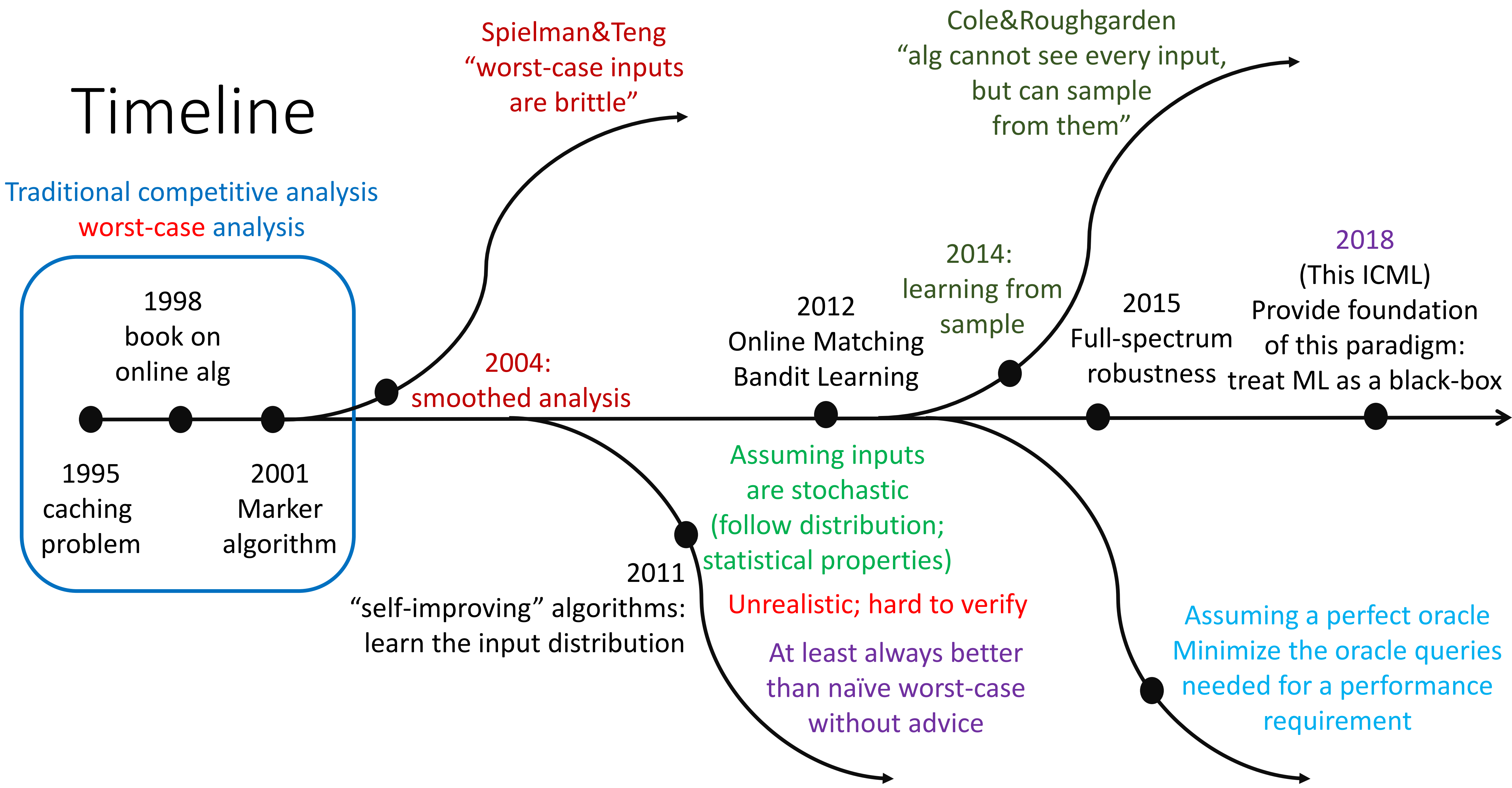
Online Algorithms with ORACLE and ML Advice (and why these are different)

Joint Study Group, 20200709


Timeline
Traditional competitive analysis
worst-case analysis
1998
book on
online alg
1995
caching
problem
2001
Marker
algorithm
Spielman&Teng
“worst-case inputs
are brittle”
2004:
smoothed analysis
2011
“self-improving” algorithms:
learn the input distribution
2012
Online Matching
Bandit Learning
Assuming inputs
are stochastic
(follow distribution;
statistical properties)
Unrealistic; hard to verify
At least always better
than naïve worst-case
without advice
Cole&Roughgarden
“alg cannot see every input,
but can sample
from them”
2014:
learning from
sample
2015
Full-spectrum
robustness
2018
(This ICML)
Provide foundation
of this paradigm:
treat ML as a black-box
Assuming a perfect oracle
Minimize the oracle queries
needed for a performance
requirement

Online Alg with PERFECT ORACLE Advice

Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, Jesper W. Mikkelsen: Online Algorithms with Advice: A Survey. SIGACT News 47(3): 93-129 (2016)

- Input is adversarial
- Decisions are irrevocable
- Algorithm knows ~~nothing~~ about the inputs before arrival

some information bits from ORACLE

1. How many bits of advice are necessary and sufficient to obtain a competitive ratio c?
2. Given a fixed B bits of advice, how good can c be?

What exactly is “fully characterize”?

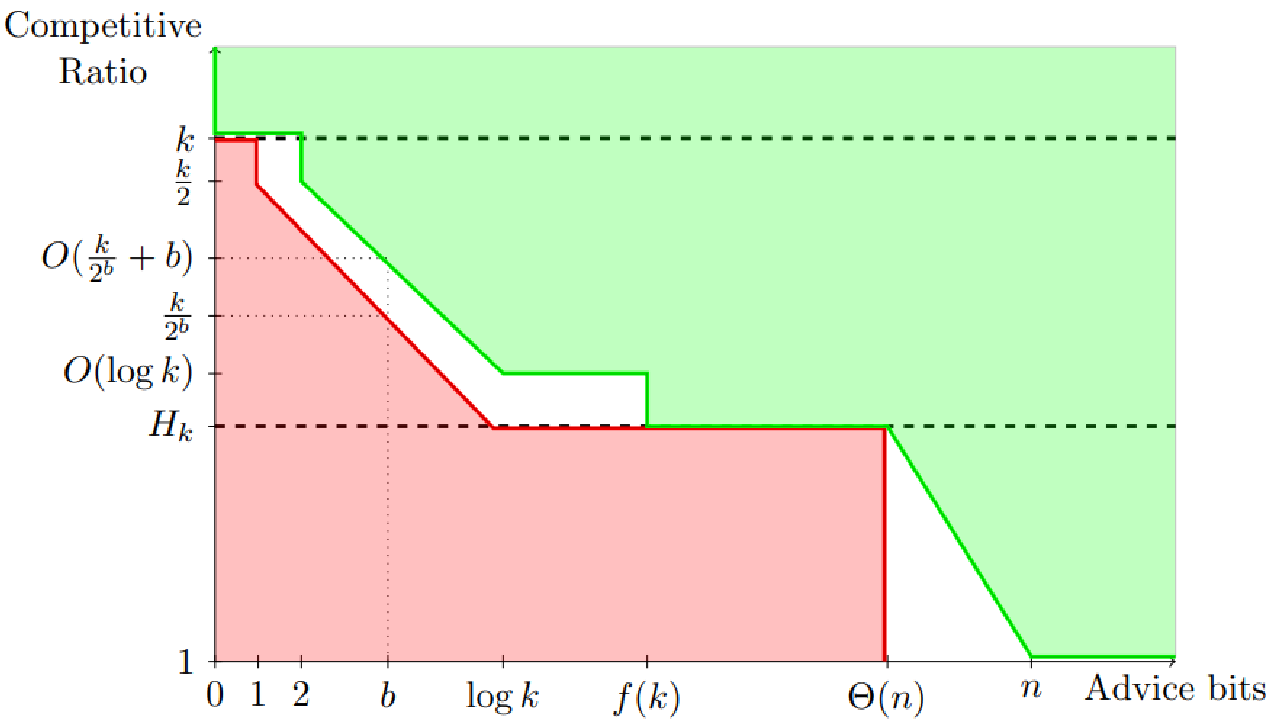


Figure 1: The (asymptotic) trade-off between competitive ratio and advice for PAGING. The function $f(k)$ is a rapidly growing function of k (but does not depend on n). Consider a trade-off point (b, c) where b is a number of advice bits and c is a competitive ratio. The red area shows those trade-offs which provably cannot be achieved. The green area shows those trade-offs that we currently have algorithms achieving. It is an open problem if trade-offs in the white area are achievable or not. The horizontal dashed lines are the best possible competitive ratios of deterministic and randomized algorithms without advice.

Some insights for PERFECT ORACLE

- Connection between randomized online algorithms
  - Amount of advice ≈ amount of randomness needed (“guessing” advice)
- De-online
  - Use an online algorithm with b bits of advice in time $O(T(n))$
  - Obtain an offline approximation algorithm in time $O(2^b T(n))$
- Leads to rigorous “Complexity Classes” for online problems
  - “advice complexity”

Online optimization with imperfect ML advice

- Key difference: These ML-based “predictors” are NOT ORACLES because they are estimators

- Want to show:
  incorporating the advice enhances the quality of optimization task with the quality of enhancement depending on the quality of advice
  - Robustness
    - At least as good as naïve online algorithm, if predictor is bad
  - Consistency
    - At least as good as optimal offline algorithm, if predictor is good/powerful

Ski Rental and Non-Clairvoyant Job Scheduling with ML advice

Manish Purohit, Zoya Svitkina, Ravi Kumar: Improving Online Algorithms via ML Predictions. NeurIPS 2018: 9684-9693

Sreenivas Gollapudi, Debmalya Panigrahi: Online Algorithms for Rent-Or-Buy with Expert Advice. ICML 2019: 2319-2327

- Traditional task: Rent or Buy?
  - Deterministic Break-even algorithm: 2-competitive (optimal)
  - Randomized algorithm: $\frac{e}{e-1} \approx 1.58$-competitive (optimal)

x: actual # of skiing days
y: predicted # of skiing days
$\eta = |y - x|$: prediction error

- Traditional task: Task Scheduling (jobs/machines)
  - With uncertainty: running time of job unknown until finishes (
  - Deterministic Round-Robin algorithm: 2-competitive (optimal)

x_i: actual running time of task i
y_i: predicted running time of task i
$\eta_i = |y_i - x_i|$: prediction error of task i

Let $\lambda \in (0, 1)$ be a hyperparameter. For the ski rental problem with a predictor, we first obtain a deterministic online algorithm that is $(1 + 1/\lambda)$-robust and $(1 + \lambda)$-consistent (Section 2.2). We next improve these bounds by obtaining a randomized algorithm that is $(\frac{1}{1-e^{-(\lambda-1/b)}})$-robust and $(\frac{\lambda}{1-e^{-\lambda}})$-consistent, where b is the cost of buying (Section 2.3). For the non-clairvoyant scheduling problem, we obtain a randomized algorithm that is $(2/(1-\lambda))$-robust and $(1/\lambda)$-consistent. Note that the consistency bounds for all these algorithms circumvent the lower bounds, which is possible only because of the predictions.

Q: how about multiple (independent) predictors?

Online Weighted Paging + ML

Theorem 1.1. *For weighted paging with PRP, any deterministic algorithm is $\Omega(k)$-competitive, and any randomized algorithm is $\Omega(\log k)$-competitive.*

Theorem 1.2. *For weighted paging with ℓ-strong lookahead where $\ell \leq n-k$, any deterministic algorithm is $\Omega(k)$-competitive, and any randomized algorithm is $\Omega(\log k)$-competitive.*

SPRP ("strong per-request prediction"): On a request for page p, the predictor gives the next time-step when p will be requested *and all page requests till that request.*

Theorem 1.3. *There is a deterministic 2-competitive for weighted paging with SPRP.*

Theorem 1.4. *For weighted paging with SPRP, there is no deterministic algorithm whose cost is $o(k) \cdot \mathsf{OPT} + o(\ell_{pd})$, and there is no randomized algorithm whose cost is $o(\log k) \cdot \mathsf{OPT} + o(\ell_{pd})$.*

- Propose a new suitable model for weighted paging
- Design new online algorithms for new model
- Show lower bounds for new model

And More…

- Bin Packing/List Update with both ORACLE and untrusted ML advice
  - Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, Marc P. Renault: Online Computation with Untrusted Advice. ITCS 2020: 52:1-52:15

Sidenote: Optimization with ML advice

- Difference: the optimization task is NOT of sequential/online flavor
- Still, there is an optimization-like problem, for which one needs to learn/predict some key information to achieve good results
- *Focusing on the optimization part*, show good results *assuming a black-box predictor exists*

- Example: Andres Muñoz Medina, Sergei Vassilvitskii: Revenue Optimization with Approximate Bid Predictions. NIPS 2017: 1858-1866

Revenue Optimization with Approximate Bid Predictions

- Traditional task: Setting a good reserve price in auctions
  - Traditionally: easy for known-distributional-bidders [Myerson 81]
  - Challenge 1: distribution not known (need to sample)
  - Challenge 2: item is highly heterogeneous (no history/data to learn from)
- Why not directly ML? -> Revenue function is highly non-continuous/non-convex

Our results. We show that given a predictor of the bid with squared loss of η^2, we can construct a reserve function r that extracts all but $g(\eta)$ revenue, for a simple increasing function g. (See

$$\Phi(\mathcal{C}) := \sum_{j=1}^{k} \sqrt{\sum_{i,i':x_i,x_{i'} \in C_k} (b_i - b_{i'})^2} = 2\sum_{j=1}^{k} m_j \widehat{\sigma}_j \tag{2}$$

Theorem 2. *Let $\delta > 0$ and let r_k denote the output of Algorithm 1 then $r_k \in G(h, k)$ and with probability at least $1 - \delta$ over the samples $\mathcal{S}$:*

$$\widehat{S}(r_k) \leq (3\widehat{B})^{1/3}\Big(\frac{1}{2m}\Phi(\mathcal{C}^h)\Big) \leq (3\widehat{B})^{1/3}\Big(\frac{1}{2k} + 2\Big(\eta^2 + \sqrt{\frac{\log 1/\delta}{2m}}\Big)^{1/2}\Big)^{2/3}.$$