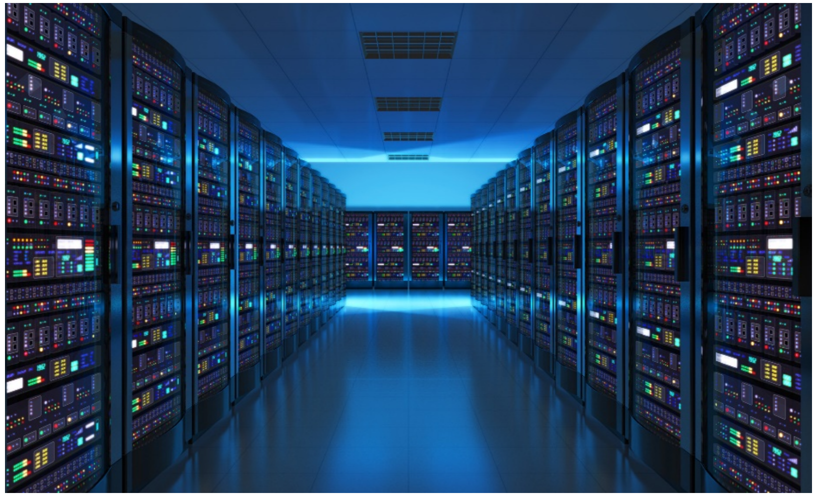
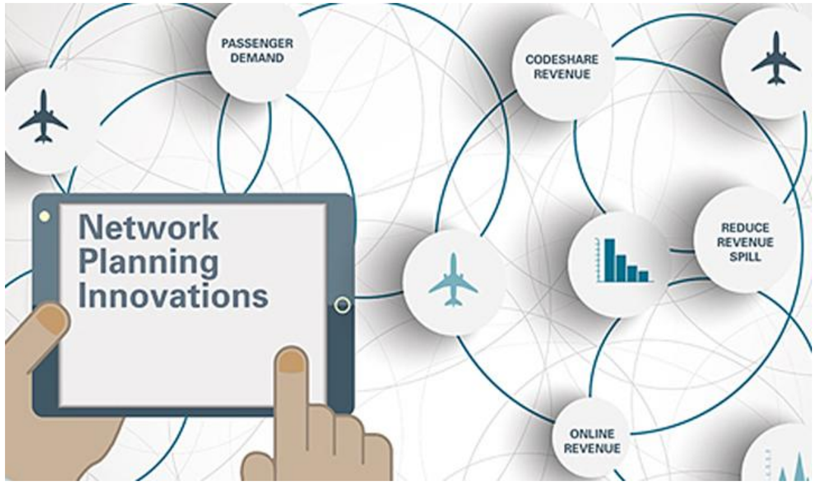
Online Algorithms for Rent or Buy with Expert Advice

Sreenivas Gollapudi
Google

Debmalya Panigrahi
Duke University

How to optimize for an unknown future?





How to optimize for an unknown future?

Online Algorithms

- Optimize for the worst possible (adversarial) future
- Competitive ratio = Online Algorithm / Offline Optimum

+ Very robust (guarantees hold no matter what)

- Pessimistic (nature is not adversarial!)

Machine Learning

- Use the past to predict the future, and optimize for the predicted future
- Approximation ratio = Offline Algorithm / Offline Optimum

+ Optimistic (approx. ratio << comp. ratio for most problems)

- Not robust (no guarantees if predictions are inaccurate)

Online Algorithms with Predictions

- A. M. Medina and S. Vassilvitskii. Revenue optimization with approximate bid predictions. NeurIPS 2017.
- T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. SIGMOD 2018.
- T. Lykouris and S. Vassilvitskii. Competitive caching with machine learned advice. ICML 2018.
- M. Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. NeurIPS 2018.
- M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ML predictions. NeurIPS 2018.
- C.-Y. Hsu, P. Indyk, D. Katabi, and A. Vakilian. Learning-based frequency estimation algorithms. ICLR 2019.

Consistency: If the prediction are accurate, then the algorithm should perform as well as the best offline solution

Robustness: Irrespective of the accuracy of the prediction, the algorithm should perform as well as the best online solution

Graceful degradation: The performance of the algorithm should gracefully degrade with the accuracy of the prediction

Online Algorithms with Multiple Predictions

- Multiple ML models/human experts make predictions about the future
- The predictions may be completely different from one another
- The algorithm has no information about the *absolute* or *relative* quality of the predictions

Consistency: If *any* of the predictions is accurate, then the algorithm should perform as well as the best offline solution

Robustness: Irrespective of the accuracy of the predictions, the algorithm should perform as well as the best online solution

Graceful degradation: The performance of the algorithm should gracefully degrade with the accuracy of the *best* prediction

A Single Parameter Problem: Rent or Buy (a.k.a. Ski-rental)

- It costs
  - \$1 to rent skis for a day
  - \$B to buy skis for the season
- Length of ski season is S
- Offline optimum
  - If S ≥ B, buy on day 1
  - If S < B, rent every day
- Unknown future: The algorithm gets to know S only when the ski season ends
- Online algorithm (existing results)
  - Competitive ratio of 2 for deterministic algorithms
  - Competitive ratio of $\frac{e}{e-1}$ for randomized algorithms

A Single Parameter Problem: Rent or Buy (a.k.a. Ski-rental)

- It costs
  - \$1 to rent skis for a day
  - \$B to buy skis for the season
- Length of ski season is S
- Offline optimum
  - If S ≥ B, buy on day 1
  - If S < B, rent every day
- Unknown future: The algorithm gets to know S only when the ski season ends
- Online algorithm (existing results)
  - Competitive ratio of 2 for deterministic algorithms
  - Competitive ratio of $\frac{e}{e-1}$ for randomized algorithms

- Online algorithm with multiple predictions (this work)
  - **k** predictions
  - **k=1**: consistency of 1 achieved by assuming the expert is accurate and using the offline algorithm [Purohit et al. '18 shows how to achieve robustness in this setting]
  - **k=∞**: experts can make all possible predictions, hence it reduces to the classical setting (without predictions)
  - What can we say for finite **k > 1**? Can we add robustness and graceful degradation for **k > 1**?
  - What is a good value of **k**?
    - Under independent Gaussian error, we show that **k** between **2** and **4** achieves significant improvements over **k < 2**

Rent or Buy with Multiple Predictions

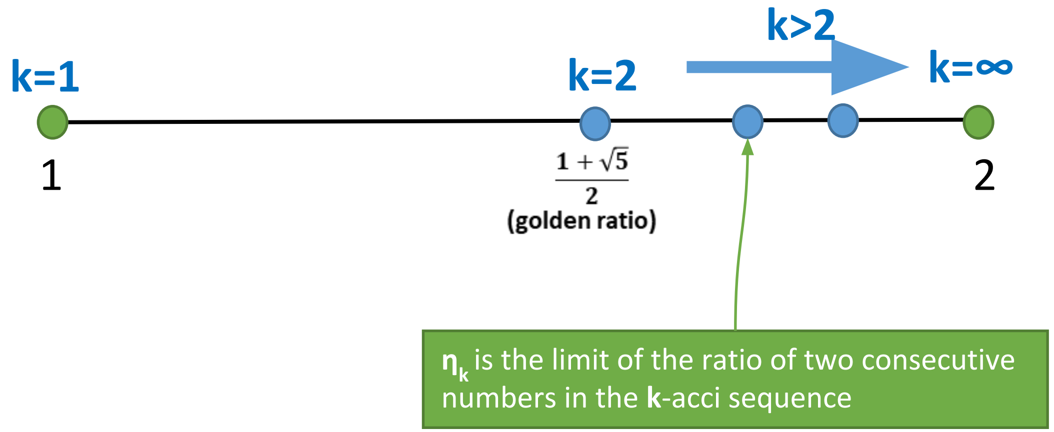
Consistency: For **k** predictions, we give an η_k-consistent deterministic algorithm where:

- $\eta_1 = 1$
- $\lim_{k \to \infty} \eta_k = 2$
- η_k is an increasing sequence
- No deterministic algorithm can achieve consistency better than η_k for **k** predictions

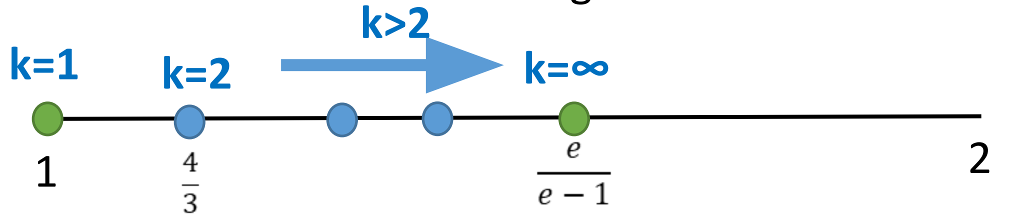
Consistency: For **k** predictions, we give a μ_k-consistent randomized algorithm where:

- $\mu_1 = 1$
- $\lim_{k \to \infty} \mu_k = \frac{e}{e-1}$
- μ_k is an increasing sequence
- No randomized algorithm can achieve consistency better than μ_k for **k** predictions

Deterministic Algorithms



Randomized Algorithms



Rent or Buy with Multiple Predictions

Consistency: For **k** predictions, we give an η_k-consistent deterministic algorithm where:

- $\eta_1 = 1$
- $\lim_{k \to \infty} \eta_k = 2$
- η_k is an increasing sequence
- No deterministic algorithm can achieve consistency better than η_k for **k** predictions

Graceful degradation: For **k** predictions, we give a deterministic algorithm with alg ≤ γ_k (opt + err) where:

- $\gamma_1 = \frac{3}{2}$
- $\lim_{k \to \infty} \gamma_k = 2$
- γ_k is an increasing sequence, $\gamma_k > \eta_k$ for finite **k**
- No deterministic algorithm can achieve a ratio better than γ_k for $\frac{\text{alg}}{\text{opt+err}}$ for **k** predictions

Consistency: For **k** predictions, we give a μ_k-consistent randomized algorithm where:

- $\mu_1 = 1$
- $\lim_{k \to \infty} \mu_k = \frac{e}{e-1}$
- μ_k is an increasing sequence
- No randomized algorithm can achieve consistency better than μ_k for **k** predictions

Robustness: For **k** predictions, we give a deterministic algorithm such for that any $0 < \lambda < 1$:

- alg ≤ $\left(1 + \frac{1}{\lambda}\right)$ opt in all situations
- alg ≤ $\rho_{k,\lambda}$ opt if the best prediction has 0 error
- $\rho_{k,\lambda}$ is an increasing sequence, $\rho_{k,\lambda} > \eta_k$ for finite **k**
- No deterministic algorithm can simultaneously achieve consistency ratio ≤ $\rho_{k,\lambda}$ and robustness ratio ≤ $\left(1 + \frac{1}{\lambda}\right)$ for **k** predictions

Future Work

- Multiple predictions in other online optimization problems
  - Caching (Lykouris and Vassilvitskii consider the single prediction case)
  - Scheduling/Load Balancing (Purohit et al. consider one variant for single prediction, but several variants are open even for single prediction)
  - k-server (single prediction is open)
- Incorporate prediction costs – multi-armed bandit models for online optimization?
- Other interfaces between online algorithms and online learning
  - Smoothed Online Convex Optimization
  - Other models?

thank you

questions?