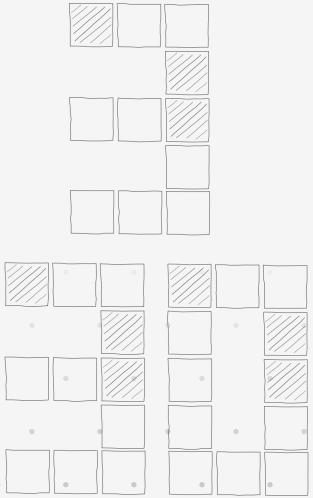
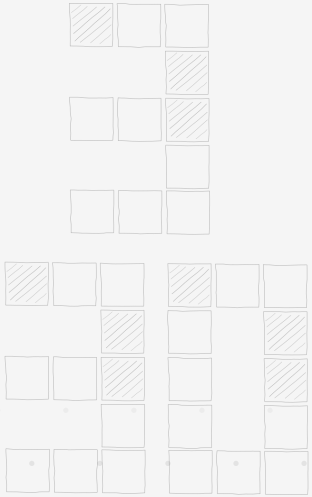


New Computational Models ▸ Algorithms with Predictions

Lecture 16

- Algorithms with Predictions Framework
- Ski Rental with Predictions

PRINCETON COMPUTER SCIENCE



New Computational Models ▸ Algorithms with Predictions

Lecture 16

- Algorithms with Predictions Framework
- Ski Rental with Predictions

Machine Learning Meets Algorithm Design

Key Question: Can we use ML predictions to design *better* algorithms?

Challenges:

- How do we incorporate predictions into classical algorithms?
- What if the predictions are wrong? Can we still guarantee good performance?
- Can we get the best of both worlds?
  - Great performance when predictions are good
  - Worst-case guarantees when predictions are bad

Key Properties: Consistency and Robustness

Goal: Tie algorithm performance to the quality of the prediction

Intuition:

- We want algorithms to be consistent: perform well when predictions are accurate
  - Ideally recover offline optimal performance with error-free predictions
- We want algorithms to be robust: maintain guarantees even with bad predictions
  - ML systems sometimes have very large errors!
  - Performance should not be much worse than standard online algorithms with no predictions

Remark

Tradeoff: Trusting predictions (consistency) vs. worst-case protection (robustness)

Example: Binary Search with Predictions

Classic problem: Given an array A sorted in increasing order with n elements and a query element q, find the index of q in A or report that it is not in the array.

Standard binary search:

- Compare q to middle element, recurse on correct half
- Cost: $O(\log n)$ comparisons (worst-case optimal)

What if we have a prediction?

- Suppose an ML model predicts where q is likely to be
- Can we use this to beat $O(\log n)$ if the prediction is good?
- But still maintain $O(\log n)$ worst-case if prediction is wrong?

Binary Search with Predictions

Assume we have a predictor h:

- For every query q, predictor returns $h(q) =$ best guess for position of q in array

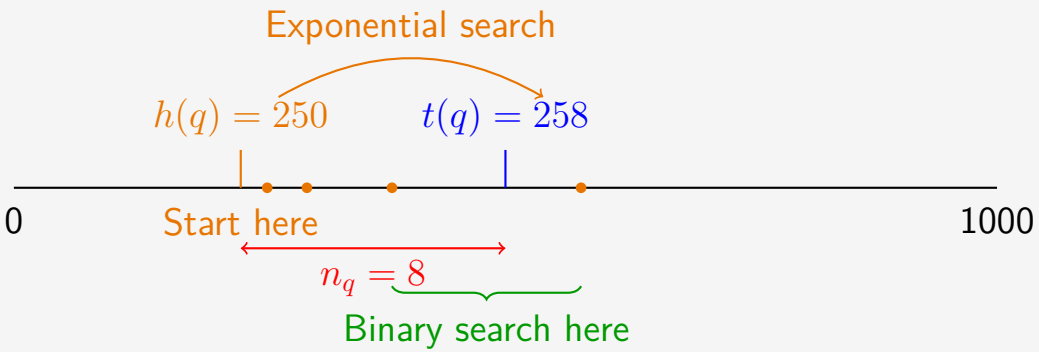
Algorithm: Binary Search with Predictions

1. Probe location $A[h(q)]$
2. If q found, return the index
3. Otherwise, we know whether q is smaller or larger than $A[h(q)]$
4. Suppose $q > A[h(q)]$ (the other case is symmetric):
   - Probe $h(q)+1$, $h(q)+2, h(q)+4, h(q)+8, \ldots$ (exponential search)
   - Stop when we find element $\geq q$ (or hit end of array)
5. Apply binary search on the interval guaranteed to contain q

Example: How It Works

Scenario: Array of size $n = 1000$, searching for element q

Prediction: $h(q) = 250$ True position: $t(q) = 258$ Error: $n_q = 8$



Total: ≈ 6 comparisons vs. $\log_2 1000 \approx 10$ for standard binary search!

Cost Analysis

Notation:

- Let $t(q) =$ true position of q in the array
  - (or position of largest element smaller than q if q not in array)
- Let $n_q = |h(q) - t(q)| =$ error of the predictor on query q

How the algorithm works:

- Start at $h(q)$, then search exponentially: $h(q)+1, h(q)+2, h(q)+4, h(q)+8, \ldots$
- After $\log n_q$ probes, we bracket an interval of size $\approx n_q$ containing q
- Binary search on that interval: another $\log n_q$ comparisons

Total cost: At most $2 \log n_q$ comparisons

Remark

Key insight: Cost depends on **prediction error** n_q, not array size n!
If $n_q = O(\text{polylog}\, n)$, we get $O(\log \log n)$ performance

Robustness Observation

What if the predictor is terrible?

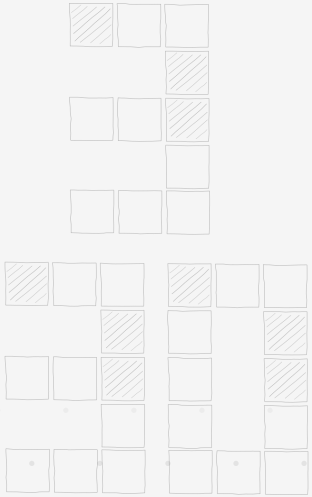
- Error n_q is always bounded: $n_q \leq n$
- Worst-case cost: $2 \log n_q \leq 2 \log n = O(\log n)$
- This matches standard binary search!

Robustness property: Even an exceptionally bad predictor cannot do much harm!

Remark

Takeaway:

- Good predictions help (can achieve $\log \log$ performance)
- Bad predictions don't hurt (worst case is still $O(\log n)$)



New Computational Models ▸ Algorithms with Predictions

Lecture 16

- Algorithms with Predictions Framework
- Ski Rental with Predictions

Recall: Ski Rental Problem (Lecture 14)

Setup:

- Rent skis for \$1 per day (today $R = 1$), or buy for $\$B$ (one-time cost)
- Will ski for d days (unknown in advance)

Value of optimal offline algorithm:

$$\text{OPT} = \min\{d, B\}$$

- If $d < B$: rent all days (pay $\$d$)
- If $d \geq B$: buy on day 1 (pay $\$B$)

From Lecture 14:

- Better-Late-Than-Never algorithm: rent for $B - 1$ days, then buy
- Achieves competitive ratio $2 - 1/B$
- **No deterministic algorithm does better!**

Ski Rental with Predictions: Setup

New assumption: We have a predictor!

- Predictor gives $p =$ predicted number of days we'll ski
- Error bound: $|d - p| \leq \eta$ where d is true number of days

Formal definitions:

- α-**consistency:** Competitive ratio $\to \alpha$ as prediction error $\to 0$
- β-**robustness:** Competitive ratio $\leq \beta$ even with arbitrarily bad predictions

Question: Can we use p to beat the 2-competitive ratio when predictions are good, while still maintaining robustness when predictions are bad?

Trust ML Algorithm

First attempt: Blindly follow the prediction

Algorithm: Trust ML Algorithm

- If $p \geq B$: Buy skis immediately on day 1
- Otherwise: Rent for however long we end up skiing

Intuition:

- If prediction says we'll ski many days, buy early
- If prediction says we'll ski few days, keep renting

Question: Does this satisfy our consistency and robustness goals?

Trust ML Algorithm Fails Robustness

Lemma: Lemma

The Trust ML algorithm is not β-robust for any finite β.

Proof

Consider scenario where $d = \eta + 1 > B$ but $p = 1$.

Cost analysis:

- Trust ML algorithm rents for all $d = \eta + 1$ days (since $p = 1 < B$)
- Trust ML pays: $\$(\eta + 1)$
- OPT buys on day 1 (since $d > B$), pays: $\$B$
- Competitive ratio: $\text{CR} = \frac{\eta+1}{B} \to \infty$ as $\eta \to \infty$

Therefore, no finite β can bound the competitive ratio.

Prudent Trust ML Algorithm

Key idea: Add a trust parameter $\lambda \in [0, 1]$ to control how much we trust the prediction

Algorithm: Prudent Trust ML Algorithm

Input: Prediction p, parameter $\lambda \in [0, 1]$

- If $p \geq B$: Buy on the start of day $\lceil \lambda \cdot B \rceil$
- Else: Buy on the start of day $\left\lceil \frac{B}{\lambda} \right\rceil$

Interpreting the Trust Parameter λ

What does λ mean?

Examine extreme cases:

- $\lambda = 1$**:** Prudent Trust ML simplifies to:
  - Buy on the start of day B (if still skiing)
  - This is exactly the Better Late Than Never algorithm from Lecture 14!
  - Competitive ratio: $2 - 1/B$ (no use of prediction)

- $\lambda = 0$**:** Prudent Trust ML simplifies to:
  - Buy immediately if $p \geq B$, otherwise rent forever
  - This is the Trust ML algorithm (blind trust in prediction)

λ controls how much we **trust** the prediction:

- $\lambda = 1$: zero trust (ignore prediction)
- $\lambda = 0$: complete trust (follow prediction blindly)
- $\lambda \in (0, 1)$: balanced trust

Robustness Lemma

Lemma: Robustness Lemma

The Prudent Trust ML algorithm is $\frac{1+\lambda}{\lambda}$-robust:

$$\text{Competitive Ratio} \leq \frac{1+\lambda}{\lambda}$$

for any prediction error η (independent of η).

What does this tell us?

- No matter how bad the prediction is, CR $\leq \frac{1+\lambda}{\lambda}$
- Worst-case guarantee independent of prediction quality
- Example: $\lambda = 1/2$ gives CR ≤ 3 (vs. 2 for no predictions)

Proof: Robustness Bound for Case $p \geq B$

Proof

In this case, Prudent Trust ML buys on the start of day $\lceil \lambda \cdot B \rceil$

Case 1: $d < \lceil \lambda \cdot B \rceil$ Both rent all days, CR $= 1$

Case 2: $\lceil \lambda \cdot B \rceil \leq d \leq B$

- Prudent Trust ML pays: $\$(\lceil \lambda \cdot B \rceil - 1 + B)$, OPT pays: $\$d$
- CR $= \frac{\lceil \lambda \cdot B \rceil - 1 + B}{d} \leq \frac{\lambda \cdot B + B}{\lambda \cdot B} = \frac{1+\lambda}{\lambda}$

Case 3: $d \geq B$

- Prudent Trust ML pays: $\$(\lceil \lambda \cdot B \rceil - 1 + B)$, OPT pays: $\$B$
- CR $= \frac{\lceil \lambda \cdot B \rceil - 1 + B}{B} \leq 1 + \lambda \leq \frac{1+\lambda}{\lambda}$

Proof: Case $p < B$

Proof

In this case, Prudent Trust ML buys on the start of day $\lceil B/\lambda \rceil$

Case 1: $d \leq B$ Both rent all days, CR $= 1$

Case 2: $B < d < \lceil B/\lambda \rceil$

- Prudent Trust ML rents every day and pays: $\$d$
- OPT buys immediately and pays: $\$B$
- CR $= \dfrac{d}{B} \leq \dfrac{B/\lambda}{B} = \dfrac{1}{\lambda} \leq \dfrac{1+\lambda}{\lambda}$

Case 3: $d \geq \lceil B/\lambda \rceil$

- Prudent Trust ML pays: $\$(\lceil B/\lambda \rceil - 1 + B)$
- OPT pays: $\$B$
- CR $= \dfrac{\lceil B/\lambda \rceil - 1 + B}{B} \leq \dfrac{B/\lambda + B}{B} = 1 + \dfrac{1}{\lambda} = \dfrac{1+\lambda}{\lambda}$

□

Consistency Lemma

Lemma: Consistency Lemma

The Prudent Trust ML algorithm is $(1+\lambda)$-consistent:

$$\text{Competitive Ratio} \leq 1 + \lambda + \frac{\eta}{(1-\lambda) \cdot \text{OPT}}$$

What does this tell us?

- As $\eta \to 0$, the bound approaches $1 + \lambda$
- Perfect predictions ($\eta = 0$) give CR $\leq 1 + \lambda$
- Example: $\lambda = 1/2$ with $\eta = 0$ gives CR ≤ 1.5 (vs. 2 for no predictions)

Proof: Consistency Bound for Case $p \geq B$

Proof

Case 1: $d < \lceil \lambda B \rceil$
In this range, both Prudent Trust ML and OPT rent every day, so CR $= 1$.

Case 2: $d \geq \lceil \lambda B \rceil$

- Prudent Trust ML buys on day $\lceil \lambda B \rceil$ and pays

$$B + \lceil \lambda B \rceil - 1 \leq (1 + \lambda)B.$$

- Since $p \geq B$ and $|d - p| \leq \eta$, if $d < B$ then $B \leq p \leq d + \eta = \mathsf{OPT} + \eta$, while if $d \geq B$ then $\mathsf{OPT} = B$, so in all cases:

$$B \leq \mathsf{OPT} + \eta.$$

- Therefore

$$\mathsf{CR} \leq \frac{(1+\lambda)B}{\mathsf{OPT}} \leq \frac{(1+\lambda)(\mathsf{OPT}+\eta)}{\mathsf{OPT}} = 1 + \lambda + \frac{(1+\lambda)\cdot\eta}{\mathsf{OPT}} \leq 1 + \lambda + \frac{\eta}{(1-\lambda)\cdot\mathsf{OPT}}$$

To see the last inequality: for $\lambda \in [0, 1]$, $(1 + \lambda) \cdot (1 - \lambda) = 1 - \lambda^2 \leq 1 \Rightarrow (1 + \lambda) \leq 1/(1 - \lambda)$

Proof: Consistency Bound for Case $p < B$

Proof

Case 1: $d < \lceil \lambda B \rceil$
In this range, both Prudent Trust ML and OPT rent every day, so CR $= 1$.

Case 2: $B < d < \lceil B/\lambda \rceil$
Here Prudent Trust ML rents every day, so it pays d. Using $|d - p| \leq \eta$ and $p < B$,

$$\text{CR} = \frac{d}{\text{OPT}} \leq \frac{p + \eta}{\text{OPT}} < \frac{B + \eta}{\text{OPT}} = 1 + \frac{\eta}{\text{OPT}} \leq 1 + \lambda + \frac{\eta}{(1 - \lambda) \cdot \text{OPT}}$$

Proof: Consistency Bound for Case $p < B$ (continued)

Proof

Case 3: $d \geq \lceil B/\lambda \rceil$

- Prudent Trust ML buys on day $\lceil B/\lambda \rceil$, so

$$\mathsf{CR} = \frac{B + \lceil B/\lambda \rceil - 1}{\mathsf{OPT}} < \frac{B + \frac{B}{\lambda}}{\mathsf{OPT}}$$

- Since $p < B$ and $|d - p| \leq \eta$ with $d \geq \lceil B/\lambda \rceil$,

$$\eta = d - p > \frac{B}{\lambda} - B = \frac{(1-\lambda)B}{\lambda}$$

- So $B < \frac{\lambda}{1-\lambda}\eta$ and

$$\mathsf{CR} < \frac{B + \frac{B}{\lambda}}{\mathsf{OPT}} < 1 + \frac{\eta}{(1-\lambda)\mathsf{OPT}} \leq 1 + \lambda + \frac{\eta}{(1-\lambda) \cdot \mathsf{OPT}}$$

Consistency-Robustness Tradeoff

Putting the two together: Competitive Ratio $\leq \min\left\{\frac{1+\lambda}{\lambda},\ 1+\lambda+\frac{\eta}{(1-\lambda)\cdot \mathsf{OPT}}\right\}$

Small λ (trust predictions more):

- Consistency: $1+\lambda \approx 1$ (excellent!)
- Robustness: $\frac{1+\lambda}{\lambda}$ is large (poor worst-case)

Large λ (trust predictions less):

- Consistency: $\geq \frac{\eta}{\mathsf{OPT}}$ (could be much worse than no predictions if η/OPT large)
- Robustness: $\frac{1+\lambda}{\lambda} \approx 2$ (good worst-case, matches Lecture 14)

In practice: Tune λ based on confidence in predictions

- High confidence $\rightarrow$ small $\lambda \rightarrow$ prioritize consistency
- Low confidence $\rightarrow$ large $\lambda \rightarrow$ prioritize robustness