How should I use my model predictions?
TACKLING UNCERTAINTY

Online Algorithms

- Full input is unknown
- Design algorithms for worst-possible future
- Pessimistic
- Cannot exploit patterns / predictability in data.
TACKLING UNCERTAINTY

Online Algorithms
- Full input is unknown
- Design algorithms for worst-possible future
- Pessimistic
- Cannot exploit patterns / predictability in data.

Machine Learning
- Observe past data
- Build robust models to predict the future
- Highly successful!
- Trained for good average performance
- Not robust to outliers
TACKLING UNCERTAINTY

Online Algorithms
- Full input is unknown
- Design algorithms for worst-possible future
- Pessimistic
- Cannot exploit patterns / predictability in data.

Machine Learning
- Observe past data
- Build robust models to predict the future
- Highly successful!
- Trained for good average performance
- Not robust to outliers
OUTLINE

- Desiderata
  - Robustness
  - Consistency
- Example
- Ski Rental Problem
- Ski Rental with a Genie
- Other Models
Online Algorithms with Predictions

Independence
- Algorithm should be independent of the predictor
- No assumptions about error types / distribution

Consistency
- Performance should improve with better predictions

Robustness
- Performance should degrade gracefully with bad predictions
TOY EXAMPLE: BINARY SEARCH

(Classical Setting)
- Sorted array A with n elements
- Query element q
- Find the index of q in A (if it exists)

(Binary Search)
- Compare q with A[mid]
- Recurse on appropriate half of A
- # probes ≤ log₂(n)
TOY EXAMPLE : BINARY SEARCH

(Classifier h)

- Predicts the index of query element q

- How can we use such a classifier?
TOY EXAMPLE : BINARY SEARCH

(LYKOURIS & VASSILVITSKII, 2018)

- Classifier h
  - Predicts the index of query element q

- How can we use such a classifier?

- Algorithm:
  - Compare q with A[h(q)]
  - If q > A[h(q)]
    - Until we find a larger element
      - Probe A[h(q) + 2], A[h(q) + 4], A[h(q) + 8], ...
    - Binary search on found interval

Binary Search
TOY EXAMPLE : BINARY SEARCH

(LYKOURIS & VASSILVITSKII, 2018)

- Analysis
  - Let $t(q) \leftarrow$ true index of $q$ in $A$
  - Let $\epsilon = |h(q) - t(q)|$ be the error

- # probes to find interval $\leq \log_2 \epsilon$
- # probes for binary search $\leq \log_2 \epsilon$

- Independence
- Consistency
- Robustness
Ski Rental

- A skier wishes to ski for $x$ days
- On each day:
  - “Rent” skis for $1$
  - “Buy” skis for $b$ and ski for free hereafter
- When should she buy the skis?
- “Rent or buy” captures many online decision making scenarios
  - When should I buy a house?
  - Should I rent servers from the cloud?
  - Should I buy an annual subscription?
A skier wishes to ski for $x$ days

If $x$ is known in advance, the problem is trivial!

Optimal Strategy:
- If $x > b$: Buy on day 1
- Else: Rent on all days

Cost = $\min(b, x)$

What can one do if $x$ is unknown?
SKI RENTAL

- Competitive Ratio = $\max_x \frac{Alg(x)}{Opt(x)}$

- Algorithm:
  - Buy on $b^{th}$ visit
  - In other words, rent for the first $b - 1$ days and then buy
SKI RENTAL

- Competitive Ratio $= \max_x \frac{Alg(x)}{Opt(x)}$

- Algorithm:
  - Buy on $b^{th}$ visit
  - In other words, rent for the first $b - 1$ days and then buy

- Analysis:
  - If $x < b$
    - $Opt(x) = Alg(x) = x$
  - If $x \geq b$
    - $Opt(x) = b$
    - $Alg(x) = b - 1 + b = 2b - 1$
**SKI RENTAL**

- **Competitive Ratio** = \( \max_x \frac{Alg(x)}{Opt(x)} \)

- **Algorithm:**
  - Buy on \( b^{th} \) visit
  - In other words, rent for the first \( b - 1 \) days and then buy

- **Analysis:**
  - If \( x < b \)
    - \( Opt(x) = Alg(x) = x \)
  - If \( x \geq b \)
    - \( Opt(x) = b \)
    - \( Alg(x) = b - 1 + b = 2b - 1 \)

**No deterministic algorithm can be better than 2-competitive!**
 Observation: Any deterministic algorithm performs worst when $x = k$ where $k$ is the day when the skis are bought

 Randomize $k$!

 Algorithm Sketch:
  Choose $i \in \{1,2,\ldots\}$ with probability $p_i$,
   Buy on the $i^{th}$ day
Observation: Any deterministic algorithm performs worst when $x = k$ where $k$ is the day when the skis are bought

Randomize $k$!

Algorithm Sketch:
- Choose $i \in \{1, 2, \ldots\}$ with probability $p_i$,
  - Buy on the $i^{th}$ day

There is a randomized $(e/(e-1))$-competitive algorithm for ski rental
SKI RENTAL (RANDOMIZED)

- Observation: Any deterministic algorithm performs worst when
  the day when the skis are bought

- Randomize $k$!

- Algorithm Sketch:
  - Choose $i \in \{1,2, \ldots \}$ with probability $p_i$,
  - Buy on the $i^{th}$ day

There is a randomized $(e/(e-1))$-competitive algorithm for ski rental

Refer to notes by Claire Mathieu
http://cs.brown.edu/~claire/Talks/skirental.pdf
Ski Rental with Prediction

You shall ski for 15 days!

You sure?

Eh!

Why even tell me then?
SKI RENTAL WITH PREDICTION

- **Notation**
  - $y \leftarrow$ Predicted number of days
  - $\eta = |x - y| = \text{Prediction error}$

- **Competitive Ratio**
  - Function of the error
    - $\frac{\text{Alg}(l)}{\text{opt}(l)} \leq c(\eta(l))$

- **Consistency**
  - Algorithm is $\beta$-consistent if $c(0) = \beta$

- **Robustness**
  - Algorithm is $\gamma$-robust if $c(\eta) \leq \gamma$ for all $\eta$
ATTEMPT 1

- Attempt 1
  (a.k.a Trust the genie)

- If $y \geq b$
  - Buy on day 1

- Else
  - Rent every day

Claim: $\text{ALG} \leq \text{OPT} + \eta$
ATTEMPT 1

- Attempt 1
  (a.k.a Trust the genie)

- If $y \geq b$
  - Buy on day 1

- Else
  - Rent every day

Claim: $ALG \leq OPT + \eta$

Proof:

If ($y \geq b$ and $x \geq b$) or ($y < b$ and $x < b$)

$ALG = OPT$

If ($y \geq b$ and $x < b$)

$ALG = b \leq x + (y - x) = OPT + \eta$

If ($y < b$ and $x \geq b$)

$ALG = x \leq b + (x - y) = OPT + \eta$
ATTEMPT 1

- Attempt 1  
  (a.k.a Trust the genie)

- If $y \geq b$
  - Buy on day 1

- Else
  - Rent every day

Claim: $\text{ALG} \leq \text{OPT} + \eta$
ATTEMPT 1

- Attempt 1 (a.k.a Trust the genie)

  - If $y \geq b$
    - Buy on day 1
  - Else
    - Rent every day

Claim: $\text{ALG} \leq \text{OPT} + \eta$

1 – consistent!
ATTEMPT 1

- Attempt 1 (a.k.a Trust the genie)
  
  - If $y \geq b$
    - Buy on day 1
  
  - Else
    - Rent every day

Claim: $\text{ALG} \leq \text{OPT} + \eta$

1 – consistent!

Not Robust
ATTEMPT 2

- Attempt 2
  (a.k.a Trust the genie, but cautiously)

- Let $\lambda \in (0,1)$ be a hyperparameter

- If $y \geq b$
  - Buy on day $\lfloor \lambda b \rfloor$

- Else
  - Buy on day $\left\lfloor \frac{b}{\lambda} \right\rfloor$

Claim:\ $\frac{\text{ALG}}{\text{OPT}} \leq \min \left\{ 1 + \lambda + \frac{\eta}{(1-\lambda) \text{OPT}}, \frac{1+\lambda}{\lambda} \right\}$
ATTEMPT 2

- Attempt 2
  (a.k.a Trust the genie, but cautiously)

- Let $\lambda \in (0,1)$ be a hyperparameter
- If $y \geq b$
  - Buy on day $\lceil \lambda b \rceil$
- Else
  - Buy on day $\left\lfloor \frac{b}{\lambda} \right\rfloor$

Claim: $\frac{\text{ALG}}{\text{OPT}} \leq \min \left\{ 1 + \lambda + \frac{\eta}{(1-\lambda)\text{OPT}}, \frac{1+\lambda}{\lambda} \right\}$

$(1 + \lambda) - \text{consistent!}$

$\left(\frac{1+\lambda}{\lambda}\right) - \text{robust}$
ATTEMPT 2

- Attempt 2
  (a.k.a Trust the genie, but cautiously)

- $\lambda \in (0,1)$ gives a tradeoff between consistency and robustness

- Small $\lambda$
  - Larger trust in the prediction
  - Higher consistency
ATTEMPT 2

- Attempt 2 (a.k.a Trust the genie, but cautiously)

- $\lambda \in (0,1)$ gives a tradeoff between consistency and robustness

- Small $\lambda$
  - Larger trust in the prediction
  - Higher consistency

Can we do better?
Shall we randomize?

If $y \geq b$

- $k = \lfloor \lambda b \rfloor$
- Define $q_i \leftarrow \left(\frac{b-1}{b}\right)^{k-i} \cdot \frac{1}{b(1 - (1 - 1/b)^k)}$
- Choose $j \in \{1, 2, \ldots, k\}$ randomly from distribution defined by $q_i$.
- Buy on day $j$

Else

- $\ell = \left\lfloor \frac{b}{\lambda} \right\rfloor$
- Define $r_i \leftarrow \left(\frac{b-1}{b}\right)^{\ell-i} \cdot \frac{1}{b(1 - (1 - 1/b)^\ell)}$
- Choose $j \in \{1, 2, \ldots, \ell\}$ randomly from distribution defined by $r_i$.
- Buy on day $j$
Shall we randomize?

If \( y \geq b \)

\[ k = \lfloor \lambda b \rfloor \]

Define \( q_i \leftarrow \frac{(b-1)^{k-i}}{b} \cdot \frac{1}{b(1 - (1-1/b)^k)} \)

Choose \( j \in \{1, 2, \ldots, k\} \) randomly from distribution defined by \( q_i \).

Buy on day \( j \)

Else

\[ \ell = \left\lfloor \frac{b}{\lambda} \right\rfloor \]

Define \( r_i \leftarrow \frac{(b-1)^{\ell-i}}{b} \cdot \frac{1}{b(1 - (1-1/b)^\ell)} \)

Choose \( j \in \{1, 2, \ldots, \ell\} \) randomly from distribution defined by \( r_i \).

Buy on day \( j \)
Many potential models to study ski rental with “learning”

- Binary Classifier
  - Does the skier ski for at least $b$ days?
  - A perfect classifier guarantees an optimal solution every time!
  - Let $h \leftarrow$ probability of correct classification

- If $h$ is known (say via past data), we can design better algorithms!
Many potential models to study ski rental with “learning”

- Binary Classifier
  - Does the skier ski for at least $b$ days?
  - A perfect classifier guarantees an optimal solution every time!
  - Let $h \leftarrow$ probability of correct classification

- If $h$ is known (say via past data), we can design better algorithms!

For known prediction accuracy $h$, there exists a randomized algorithm with competitive ratio $f(h)$ such that $f(1) = 1$ and $f(1/2) \leq \frac{e}{e-1}$
Can we leverage bounded uncertainty in our algorithms?

“I know I’m going to ski for at least 10 days” vs.
“I have no clue how many times I’ll ski!”
More convenient to work with fractional version of the problem

Costs 1 to buy skis

Costs $z$ to rent skis for $z$ time (fractional)

Let $x \leftarrow$ true time required to ski

Prediction: $x \geq y$

There exists a randomized algorithm with competitive ratio $\left( \frac{e}{e-(1-y)e^y} \right)$
**BOUNDDED UNCERTAINTY**

- Let $p_y(z) \leftarrow$ Probability of buying on day $y$
- Let $q(y) \leftarrow$ Probability of buying immediately

- Say we enforce $p_y(z) = 0, \forall z > 1$ \hspace{1cm} (Even the deterministic algorithm does that)

- What’s the algorithm cost for $x$ days?
  - $Cost_y(x) = q(y) + \int_0^x (1 + z)p_y(z)dz + \int_x^1 xp_y(z)dz$
  - Set probabilities so that $\left(\frac{Cost_y(x)}{\min(x,1)}\right) = \left(\frac{Cost_y(x)}{x}\right)$ is a constant
Let $p_y(z) \leftarrow$ Probability of buying on day $y$

Let $q(y) \leftarrow$ Probability of buying immediately

Say we enforce $p_y(z) = 0, \forall z > 1$  (Even the deterministic algorithm does that)

What’s the algorithm cost for $x$ days?

$\text{Cost}_y(x) = q(y) + \int_0^x (1 + z)p_y(z)dz + \int_x^1 xp_y(z)dz$

Set $p_y(z)$ so that $\text{HIJ}_K_L_E_MNO(E, G)$ is a constant

There exists a randomized algorithm with competitive ratio $\frac{e}{e - (1 - y)e^y}$
SKI RENTAL WITH MULTIPLE EXPERTS

(COLLAPUDI & PANIGRAHI, 2019)

You shall ski for 15 days!

No! 30 days!
SKI RENTAL WITH MULTIPLE EXPERTS

(GOLLAPUDI & PANIGRAHI, 2019)

You shall ski for 15 days!

No! 30 days!

70 days!

10 days!
THANKS!