Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.

Is the game framework only interesting in some infinite horizon? No.

Minimize worst expected loss. Best defense.

Any prior distribution on opponent. Best offense.

Rational players should play this way! "Infinite horizon" is just an assumption of rationality.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games. Is the game framework only interesting in some infinite horizon?
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon? No.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon? No.
  Minimize worst expected loss.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games. Is the game framework only interesting in some infinite horizon? No. Minimize worst expected loss. Best defense.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon?
No.
   Minimize worst expected loss. Best defense.
   Any prior distribution on opponent.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games. Is the game framework only interesting in some infinite horizon? No.
- Minimize worst expected loss. Best defense.
- Any prior distribution on opponent. Best offense.
Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon? No.
  Minimize worst expected loss. Best defense.
  Any prior distribution on opponent. Best offense.
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon?
No.
  Minimize worst expected loss. Best defense.
  Any prior distribution on opponent. Best offense.
Rational players should play this way!
Comments on last lecture.

Easy to come up with several Nash for non-zero-sum games.
Is the game framework only interesting in some infinite horizon?
No.
    Minimize worst expected loss. Best defense.
    Any prior distribution on opponent. Best offense.
Rational players should play this way!
“Infinite horizon” is just an assumption of rationality.
Finish Maximum Weight Matching Algorithm.
Today

Finish Maximum Weight Matching Algorithm.
Exact algorithm with dueling players.
Finish Maximum Weight Matching Algorithm.
Exact algorithm with dueling players.
Multiplicative Weights Framework.
Finish Maximum Weight Matching Algorithm.
   Exact algorithm with dueling players.

Multiplicative Weights Framework.
   Very general framework of toll/congestion algorithm.
Matching/Weighted Vertex Cover

Maximum Weight Matching.
Matching/Weighted Vertex Cover

Maximum Weight Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.
Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow \mathbb{R}$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow \mathbb{R}$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow \mathbb{R}$, where for all edges, $e = (u, v)$

$$p(u) + p(v) \geq w(e).$$

Minimize $\sum_{v \in U \cup V} p(u)$. 

Optimal solutions to both if for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and perfect matching.
Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find a vertex cover function of minimum total value.

A function $p : V \rightarrow R$, where for all edges, $e = (u, v)$

\[ p(u) + p(v) \geq w(e) \]

Minimize $\sum_{v \in U \cup V} p(u)$.

Optimal solutions to both if for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and perfect matching.
Matching/Weighted Vertex Cover

Maximum Weight Matching.
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find a maximum weight weight matching.

A matching is a set of edges where no two share an endpoint.

Minimum Weight Cover.
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.
Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow R$, where for all edges, $e = (u, v)$

$p(u) + p(v) \geq w(e)$. 

Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, \( G = (U, V, E) \), with edge weights \( w : E \to \mathbb{R} \), find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, \( G = (U, V, E) \), with edge weights \( w : E \to \mathbb{R} \), find an vertex cover function of minimum total value.

A function \( p : V \to \mathbb{R} \), where for all edges, \( e = (u, v) \)

\[ p(u) + p(v) \geq w(e). \]

Minimize \( \sum_{v \in U \cup V} p(u) \).
Matching/Weighted Vertex Cover

Maximum Weight Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

Minimum Weight Cover.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$

$p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.

Optimal solutions to both if
Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to \mathbb{R}$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to \mathbb{R}$, find an vertex cover function of minimum total value.

A function $p : V \to \mathbb{R}$, where for all edges, $e = (u, v)$

\[ p(u) + p(v) \geq w(e). \]

Minimize $\sum_{v \in U \cup V} p(u)$.

Optimal solutions to both if for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
Matching/Weighted Vertex Cover

**Maximum Weight Matching.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow \mathbb{R}$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

**Minimum Weight Cover.**

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow \mathbb{R}$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow \mathbb{R}$, where for all edges, $e = (u, v)$

$$p(u) + p(v) \geq w(e).$$

Minimize $\sum_{v \in U \cup V} p(u)$.

Optimal solutions to both if for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and perfect matching.
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function $(p \cdot)$
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."
No augmenting path.
Cut, $(S, T)$, in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from $S \cup T$. Lower prices in $S$, raise prices in $T$, all explored edges still tight, backward edges still feasible...

What's delta?
$w(e) > p(u) + p(v) \rightarrow \delta = \min_{e \in (S \cup T \times T)} w(e) - p(u) - p(v)$.
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function ($\rho(\cdot)$)
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
**Maximum Weight Matching**

Goal: perfect matching on tight edges.

**Algorithm**

Start with empty matching, feasible cover function \((\rho(\cdot))\)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.

\[ \text{Algorithm} \]

\[
\text{Start with empty matching, feasible cover function } (\rho(\cdot)) \\
\text{Add tight edges to matching.} \\
\text{Use alt./aug. paths of tight edges.}
\]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

\[ p(u) + p(v) - \delta \leq w(e) \]

\[
\delta = \min_{e \in (S \cup T \times V)} w(e) - p(u) - p(v).
\]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm

Start with empty matching, feasible cover function \( (\rho(\cdot)) \)
Add tight edges to matching.
    Use alt./aug. paths of tight edges.
        "maximum matching algorithm."
No augmenting path.

Algorithm

Start with empty matching, feasible cover function \( (\rho(\cdot)) \)
Add tight edges to matching.
    Use alt./aug. paths of tight edges.
        "maximum matching algorithm."
No augmenting path.
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function ($\rho(\cdot)$)
Add tight edges to matching.
   Use alt./aug. paths of tight edges.
   "maximum matching algorithm."
No augmenting path.
   Cut, $(S, T)$, in directed graph of tight edges!

What's $\delta$?

$$\delta = \min_{e \in (S \cup T \times T \cup S)} w(e) - \rho(u) - \rho(v).$$
Maximum Weight Matching
Goal: perfect matching on tight edges.

### Algorithm

Start with empty matching, feasible cover function \((\rho(\cdot))\)

Add tight edges to matching.

Use alt./aug. paths of tight edges.

"maximum matching algorithm."

No augmenting path.

Cut, \((S, T)\), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

\[\delta = \min_{e \in (S \cup T \times T)} w(e) - \rho(u) - \rho(v).\]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \( \rho(\cdot) \)
Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    ”maximum matching algorithm.”
No augmenting path.
  Cut, \( (S, T) \), in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from \( S_U, T_V \).
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."
No augmenting path.
  Cut, \((S, T)\), in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from \(S_U, T_V\).
**Maximum Weight Matching**

**Goal:** perfect matching on tight edges.

---

**Algorithm**

Start with empty matching, feasible cover function \((\rho(\cdot))\)

Add tight edges to matching.

- Use alt./aug. paths of tight edges.
  - "maximum matching algorithm."

No augmenting path.

Cut, \((S, T)\), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from \(S_U, T_V\).

Lower prices in \(S_U\),
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm

Start with empty matching, feasible cover function ($p(\cdot)$)
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, ($S$, $T$), in directed graph of tight edges!

All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from $S_U$, $T_V$.
Lower prices in $S_U$, raise prices in $S_T$. 

What's $\delta$?

\[ w(e) > p(u) + p(v) \rightarrow \delta = \min_{e \in (S_U \times T_V)} w(e) - p(u) - p(v). \]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \( (p(\cdot)) \)
Add tight edges to matching.
Use alt./aug. paths of tight edges.
”maximum matching algorithm.”
No augmenting path.
Cut, \((S, T)\), in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from \(S_U, T_V\).
Lower prices in \(S_U\), raise prices in \(S_T\),
all explored edges still tight,
backward edges still feasible
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."
No augmenting path.
Cut, \((S, T)\), in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from \(S_U, T_V\).
Lower prices in \(S_U\), raise prices in \(S_T\),
all explored edges still tight,
backward edges still feasible
... and get new tight edge!
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
   Use alt./aug. paths of tight edges.
   "maximum matching algorithm."
No augmenting path.
   Cut, \((S, T)\), in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from \(S_U, T_V\).
Lower prices in \(S_U\), raise prices in \(S_T\),
   all explored edges still tight,
   backward edges still feasible
... and get new tight edge!
What’s delta?
Maximum Weight Matching
Goal: perfect matching on tight edges.
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm

Start with empty matching, feasible cover function \((\rho(\cdot))\)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, \((S, T)\), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from \(S_U, T_V\).

Lower prices in \(S_U\), raise prices in \(S_T\),
all explored edges still tight,
backward edges still feasible

... and get new tight edge!

What’s delta? \(w(e) > \rho(u) + \rho(v) \rightarrow \)
\(\delta = \min_{e \in (S_U \times T_V)} w(e) - \rho(u) - \rho(v).\)
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$. 

Feasible! Value = 0.

Beginning “Coverer” Solution:

$p(u) = \text{maximum incident edge for } u \in U$, 0 otherwise.

Main Work: breadth first search from unmatched nodes finds cut.

Update prices (find minimum delta.)

Simple Implementation:

Each bfs either augments or adds node to $S$ in next cut.

$O(n)$ iterations per augmentation.

$O(n)$ augmentations.

$O(n^2 m)$ time.
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible!
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \emptyset$.

Feasible! Value = 0.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \),
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).
Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$. Feasible! Value = 0.

Beginning “Coverer” Solution: $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.

Main Work:
- breadth first search from unmatched nodes finds cut.
- Update prices (find minimum delta.)
Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$. 
Feasible! Value = 0.

Beginning “Coverer” Solution: $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work: 
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).
Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to \( S \) in next cut.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).

Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \) maximum incident edge for \( u \in U \), 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to \( S \) in next cut.
  \( O(n) \) iterations per augmentation.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: $M = \{\}$. Feasible! Value = 0.

Beginning “Coverer” Solution: $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:  
  breadth first search from unmatched nodes finds cut.  
Update prices (find minimum delta.)

Simple Implementation:  
  Each bfs either augments or adds node to $S$ in next cut.  
  $O(n)$ iterations per augmentation.  
  $O(n)$ augmentations.
Some details/Runtime

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning “Matcher” Solution: \( M = \{ \} \).
Feasible! Value = 0.

Beginning “Coverer” Solution: \( p(u) = \text{maximum incident edge for } u \in U \), 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to \( S \) in next cut.
  \( O(n) \) iterations per augmentation.
  \( O(n) \) augmentations.

\( O(n^2m) \) time.
Example

All matched edges tight. Perfect matching. Feasible price function. Values the same. Optimal!

Notice: no weights on the right problem. Retains previous matching through price changes. Retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.

Perfect matching.

Feasible price function.

Values the same.

Optimal!

Notice:

no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!
Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.
Example

All matched edges tight.
Perfect matching.
Feasible price function.
Values the same.
Optimal!

Notice:
- no weights on the right problem.
- retain previous matching through price changes.
- retains edges in failed search through price changes.
Example

All matched edges tight.

Perfect matching.

Feasible price function.

Values the same.

Optimal!

Notice:

no weights on the right problem.

retain previous matching through price changes.

retains edges in failed search through price changes.
Example

All matched edges tight.
Example

All matched edges tight. Perfect matching.
All matched edges tight.
Perfect matching. Feasible price function.
Example

All matched edges tight.
Perfect matching. Feasible price function. Values the same.
Example

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!
Example

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
Example

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
Example

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
- no weights on the right problem.
- retain previous matching through price changes.
- retains edges in failed search through price changes.
The multiplicative weights framework.
Just finished the analysis of deterministic weighted majority algorithm in class.
Expert’s framework.

$n$ experts.

Every day, each offers a prediction. “Rain” or “Shine.” Whose advise do you follow? “The one who is correct most often.” Sort of. How well do you do?
Expert’s framework.

$n$ experts.
Every day, each offers a prediction.
$n$ experts.

Every day, each offers a prediction.

“Rain” or “Shine.”
$n$ experts.

Every day, each offers a prediction.

“Rain” or “Shine.”

Whose advise do you follow?
Expert’s framework.

$n$ experts.
Every day, each offers a prediction.
“Rain” or “Shine.”
Whose advise do you follow?
“The one who is correct most often.”
$n$ experts.

Every day, each offers a prediction.

“Rain” or “Shine.”

Whose advise do you follow?

“The one who is correct most often.”

Sort of.
Expert’s framework.

$n$ experts.

Every day, each offers a prediction.

“Rain” or “Shine.”

Whose advise do you follow?

“The one who is correct most often.”

Sort of.

How well do you do?
Infallible expert.

One of the expert's is infallible!
Infallible expert.

One of the expert’s is infallible!

Your strategy?
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..never!
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..never! Never see a mistake.
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..never! Never see a mistake.

Better model?
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..never! Never see a mistake.

Better model?

How many mistakes could you make?
Infallible expert.
One of the expert’s is infallible!
Your strategy?
Choose any expert that has not made a mistake!
How long to find perfect expert?
Maybe..never! Never see a mistake.
Better model?
How many mistakes could you make? Mistake Bound.
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe..never! Never see a mistake.

Better model?

How many mistakes could you make? Mistake Bound.

(A) 1
(B) 2
(C) log \( n \)
(D) \( n - 1 \)

Adversary designs setup to watch who you choose, and make that expert make a mistake.
Infallible expert.

One of the expert’s is infallible!

Your strategy?

Choose any expert that has not made a mistake!

How long to find perfect expert?

Maybe.. never! Never see a mistake.

Better model?

How many mistakes could you make? **Mistake Bound.**

(A) 1
(B) 2
(C) \(\log n\)
(D) \(n - 1\)

Adversary designs setup to watch who you choose, and make that expert make a mistake.

\(n - 1\)!
Concept Alert.

Note.
Concept Alert.

Note.

Adversary:
Concept Alert.

Note.

Adversary:
  makes you want to look bad.
Concept Alert.

Note.
Adversary:
  makes you want to look bad.
  "You could have done so well"...
Concept Alert.

Note.

Adversary:
  makes you want to look bad.
  ”You could have done so well”...
  but you didn’t!
Concept Alert.

Note.

Adversary:
  makes you want to look bad.
"You could have done so well"...
  but you didn’t! ha..
Concept Alert.

Note.

Adversary:
- makes you want to look bad.
  "You could have done so well"
- but you didn’t! ha..ha!
Note.

Adversary: makes you want to look bad.
   "You could have done so well"...
   but you didn’t! ha..ha!

Analysis of Algorithms: do as well as possible!
Back to mistake bound.

Infallible Experts.
Infallible Experts.

Alg: Choose one of the perfect experts.
Infallible Experts.

Alg: Choose one of the perfect experts.

Mistake Bound: $n - 1$
Infallible Experts.

Alg: Choose one of the perfect experts.

**Mistake Bound: \( n - 1 \)**
- Lower bound: adversary argument.
Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

**Mistake Bound: \( n - 1 \)**

- Lower bound: adversary argument.
- Upper bound:
Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

**Mistake Bound: \( n - 1 \)**

- Lower bound: adversary argument.
- Upper bound: every mistake finds fallible expert.
Back to mistake bound.

Infallible Experts.
Alg: Choose one of the perfect experts.

**Mistake Bound:** $n - 1$
- Lower bound: adversary argument.
- Upper bound: every mistake finds fallible expert.

Better Algorithm?
Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

Mistake Bound: $n - 1$
  - Lower bound: adversary argument.
  - Upper bound: every mistake finds fallible expert.

Better Algorithm?

Making decision, not trying to find expert!
Infallible Experts.

Alg: Choose one of the perfect experts.

Mistake Bound: $n - 1$
  - Lower bound: adversary argument.
  - Upper bound: every mistake finds fallible expert.

Better Algorithm?

Making decision, not trying to find expert!

Algorithm: Go with the majority of previously correct experts.
Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

**Mistake Bound: $n - 1$**
- Lower bound: adversary argument.
- Upper bound: every mistake finds fallible expert.

Better Algorithm?

Making decision, not trying to find expert!

Algorithm: Go with the majority of previously correct experts.

What you would do anyway!
Alg 2: find majority of the perfect

How many mistakes could you make?
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) $\log n$
(D) $n - 1$
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) \( \log n \)
(D) \( n - 1 \)

At most \( \log n \)!
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) $\log n$
(D) $n - 1$

At most $\log n$!

When alg makes a mistake, "perfect" experts drop by a factor of two.
Alg 2: find majority of the perfect

How many mistakes could you make?

(A) 1
(B) 2
(C) \( \log n \)
(D) \( n - 1 \)

At most \( \log n \)!

When alg makes a mistake, 
|“perfect” experts| drops by a factor of two.

Initially \( n \) perfect experts
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) log \( n \)
(D) \( n - 1 \)

At most \( \log n \)!

When alg makes a mistake, 
|“perfect” experts| drops by a factor of two.

Initially \( n \) perfect experts \( \rightarrow \leq n/2 \) perfect experts
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) \( \log n \)
(D) \( n - 1 \)

At most \( \log n \)!

When alg makes a mistake,
|“perfect” experts| drops by a factor of two.

Initially \( n \) perfect experts → \( \leq n/2 \) perfect experts
mistake → \( \leq n/4 \) perfect experts
Alg 2: find majority of the perfect

How many mistakes could you make?

(A) 1
(B) 2
(C) log $n$
(D) $n - 1$

At most log $n$!

When alg makes a mistake, the number of “perfect” experts drops by a factor of two.

Initially $n$ perfect experts → $\leq n/2$ perfect experts

$\vdots$

mistake → $\leq n/4$ perfect experts

$\vdots$
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) $\log n$
(D) $n - 1$

At most $\log n$!

When alg makes a mistake, 
|“perfect” experts| drops by a factor of two.

Initially $n$ perfect experts mistake $\rightarrow \leq n/2$ perfect experts
mistake $\rightarrow \leq n/4$ perfect experts

\vdots

mistake $\rightarrow \leq 1$ perfect expert
Alg 2: find majority of the perfect

How many mistakes could you make?
(A) 1
(B) 2
(C) \( \log n \)
(D) \( n - 1 \)

At most \( \log n \)!

When alg makes a mistake,
|“perfect” experts| drops by a factor of two.

Initially \( n \) perfect experts
mistake \( \rightarrow \leq n/2 \) perfect experts
mistake \( \rightarrow \leq n/4 \) perfect experts

\vdots

mistake \( \rightarrow \leq 1 \) perfect expert
Alg 2: find majority of the perfect

How many mistakes could you make?

(A) 1
(B) 2
(C) $\log n$
(D) $n - 1$

At most $\log n$!

When alg makes a mistake, the number of "perfect" experts drops by a factor of two.

Initially $n$ perfect experts $\rightarrow \leq n/2$ perfect experts
mistake $\rightarrow \leq n/4$ perfect experts
$\vdots$
mistake $\rightarrow \leq 1$ perfect expert
$\geq 1$ perfect expert
Alg 2: find majority of the perfect

How many mistakes could you make?

(A) 1
(B) 2
(C) $\log n$
(D) $n - 1$

At most $\log n$!

When alg makes a mistake, $|\text{“perfect” experts}|$ drops by a factor of two.

Initially $n$ perfect experts mistake $\rightarrow \leq n/2$ perfect experts mistake $\rightarrow \leq n/4$ perfect experts

$\vdots$

mistake $\rightarrow \leq 1$ perfect expert

$\geq 1$ perfect expert $\rightarrow$ at most $\log n$ mistakes!
Imperfect Experts

Goal?
Imperfect Experts

Goal?
Do as well as the best expert!
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm.
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Best expert is penalized the least.
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Best expert is penalized the least.

1. Initially: \( w_i = 1 \).
Imperfect Experts

Goal?
Do as well as the best expert!
Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Best expert is penalized the least.

1. Initially: \( w_i = 1 \).
2. Predict with weighted majority of experts.
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Best expert is penalized the least.

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Imperfect Experts

Goal?
Do as well as the best expert!

Algorithm. Suggestions?
Go with majority?
Penalize inaccurate experts?
Best expert is penalized the least.

1. Initially: \( w_i = 1 \).
2. Predict with weighted majority of experts.
3. \( w_i \rightarrow w_i/2 \) if wrong.
Analysis: weighted majority

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i / 2$ if wrong.

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$.

Initially $n$. For best expert, $b$, $w_b \geq 1 / 2^m$.

Each mistake: total weight of incorrect experts reduced by $-1$?

Potential function decreased by $3 / 4$.

We have $1 / 2^m \leq \sum_i w_i \leq (3 / 4)^M n$.

where $M$ is number of algorithm mistakes.
Analysis: weighted majority

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i/2$ if wrong.

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum w_i$.

Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2}m$.

Each mistake: total weight of incorrect experts reduced by $-\frac{1}{2}$ factor of $\frac{1}{2}$ each incorrect expert weight multiplied by $\frac{1}{2}$ total weight decreases by factor of $\frac{3}{4}$ mistake $\rightarrow \geq$ half weight with incorrect experts.

Mistake $\rightarrow$ potential function decreased by $\frac{3}{4}$.

We have $\frac{1}{2}m \leq \sum w_i \leq (\frac{3}{4})^m$.

where $M$ is number of algorithm mistakes.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function:

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$.

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.

We have $\frac{1}{2} m \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n$.

where $M$ is number of algorithm mistakes.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.
Potential function: $\sum_i w_i$. Initially $n$.

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

1. Initially: \( w_i = 1 \).

2. Predict with weighted majority of experts.

3. \( w_i \rightarrow w_i / 2 \) if wrong.

Goal: Best expert makes \( m \) mistakes.

Potential function: \( \sum_i w_i \). Initially \( n \).

For best expert, \( b \), \( w_b \geq \frac{1}{2^m} \).

Each mistake:

\[
\text{We have } \frac{1}{2^m} \leq \sum_i w_i \leq (\frac{3}{4})^M n.
\]
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
  total weight of incorrect experts reduced by

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
  - total weight of incorrect experts reduced by $-1$.

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \to w_i/2$ if wrong.

$\frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n$. where $M$ is number of algorithm mistakes.
Analysis: weighted majority

Goal: Best expert makes \( m \) mistakes.

Potential function: \( \sum_i w_i \). Initially \( n \).

For best expert, \( b \), \( w_b \geq \frac{1}{2^m} \).

Each mistake:
- total weight of incorrect experts reduced by \(-1? \ -2?\)

1. Initially: \( w_i = 1 \).
2. Predict with weighted majority of experts.
3. \( w_i \rightarrow w_i/2 \) if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.
Potential function: $\sum_i w_i$. Initially $n$.
For best expert, $b$, $w_b \geq \frac{1}{2^m}$.
Each mistake:
  total weight of incorrect experts reduced by $-1? -2? \text{ factor of } \frac{1}{2}$?

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes \( m \) mistakes.

Potential function: \( \sum_i w_i \). Initially \( n \).

For best expert, \( b \), \( w_b \geq \frac{1}{2^m} \).

Each mistake:
- total weight of incorrect experts reduced by \(-1? \) \(-2? \) factor of \( \frac{1}{2} \)?
- each incorrect expert weight multiplied by \( \frac{1}{2} \)!

1. Initially: \( w_i = 1 \).
2. Predict with weighted majority of experts.
3. \( w_i \to w_i/2 \) if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
- total weight of incorrect experts reduced by $\frac{1}{2}$?
- each incorrect expert weight multiplied by $\frac{1}{2}$!
- total weight decreases by

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
- total weight of incorrect experts reduced by $-1$? $-2$? factor of $\frac{1}{2}$?
  - each incorrect expert weight multiplied by $\frac{1}{2}$!
- total weight decreases by factor of $\frac{1}{2}$? factor of $\frac{3}{4}$?

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes \( m \) mistakes.

Potential function: \( \sum_i w_i \). Initially \( n \).

For best expert, \( b \), \( w_b \geq \frac{1}{2^m} \).

Each mistake:
- total weight of incorrect experts reduced by \(-1\)? \(-2\)? factor of \( \frac{1}{2} \)?
- each incorrect expert weight multiplied by \( \frac{1}{2} \)!
- total weight decreases by factor of \( \frac{1}{2} \)? factor of \( \frac{3}{4} \)?
- mistake \( \rightarrow \geq \) half weight with incorrect experts.

1. Initially: \( w_i = 1 \).
2. Predict with weighted majority of experts.
3. \( w_i \rightarrow \frac{w_i}{2} \) if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
- total weight of incorrect experts reduced by $-1\? -2\?$ factor of $\frac{1}{2}\?$
  - each incorrect expert weight multiplied by $\frac{1}{2}$!
- total weight decreases by factor of $\frac{1}{2}\?$ factor of $\frac{3}{4}\?$
- mistake $\rightarrow \geq$ half weight with incorrect experts.

Mistake $\rightarrow$ potential function decreased by $\frac{3}{4}$.

1. Initially: $w_i = 1$.
2. Predict with weighted majority of experts.
3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: weighted majority

Goal: Best expert makes $m$ mistakes.

Potential function: $\sum_i w_i$. Initially $n$.

For best expert, $b$, $w_b \geq \frac{1}{2^m}$.

Each mistake:
- total weight of incorrect experts reduced by $-1^2$ $-2^2$ factor of $\frac{1}{2}$?
  - each incorrect expert weight multiplied by $\frac{1}{2}$!
- total weight decreases by factor of $\frac{1}{2}^2$ factor of $\frac{3}{4}$?

mistake $\rightarrow \geq$ half weight with incorrect experts.

Mistake $\rightarrow$ potential function decreased by $\frac{3}{4}$.

We have

$$\frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n.$$

where $M$ is number of algorithm mistakes.

1. Initially: $w_i = 1$.

2. Predict with weighted majority of experts.

3. $w_i \rightarrow w_i/2$ if wrong.
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]
Analysis: continued.

\[
\frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n.
\]

\(m\) - best expert mistakes
Analysis: continued.

\[
\frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n.
\]

$m$ - best expert mistakes  \( M \) algorithm mistakes.
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left( \frac{3}{4} \right)^M n. \]

\( m \) - best expert mistakes \quad \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left( \frac{3}{4} \right)^M n. \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left( \frac{3}{4} \right)^M n. \]

\( m \) - best expert mistakes \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left( \frac{3}{4} \right)^M n. \]

Take log of both sides.
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]

\( m \) - best expert mistakes  \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left(\frac{3}{4}\right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left( \frac{3}{4} \right)^M n. \]

\( m \) - best expert mistakes \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left( \frac{3}{4} \right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]

Solve for \( M \).

\[ M \leq (m + \log n)/\log(4/3) \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]

\( m \) - best expert mistakes \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left(\frac{3}{4}\right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]

Solve for \( M \).

\[ M \leq (m + \log n)/\log(4/3) \leq 2.4(m + \log n) \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left( \frac{3}{4} \right)^M n. \]

$m$ - best expert mistakes  \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left( \frac{3}{4} \right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]

Solve for $M$.

\[ M \leq (m + \log n) / \log(4/3) \leq 2.4(m + \log n) \]

Multiple by $1 - \varepsilon$ for incorrect experts...
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq (\frac{3}{4})^M n. \]

- \( m \) - best expert mistakes
- \( M \) - algorithm mistakes.

\[ \frac{1}{2^m} \leq (\frac{3}{4})^M n. \]

Take log of both sides.

\[-m \leq -M \log(4/3) + \log n.\]

Solve for \( M \).

\[ M \leq (m + \log n)/\log(4/3) \leq 2.4(m + \log n) \]

Multiple by \( 1 - \varepsilon \) for incorrect experts...

\[ (1 - \varepsilon)^m \leq (1 - \frac{\varepsilon}{2})^M n. \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]

\( m \) - best expert mistakes \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left(\frac{3}{4}\right)^M n. \]

Take log of both sides.

\[-m \leq -M \log(4/3) + \log n.\]

Solve for \( M \).

\[ M \leq (m + \log n)/\log(4/3) \leq 2.4(m + \log n) \]

Multiple by \(1 - \epsilon\) for incorrect experts...

\[ (1 - \epsilon)^m \leq (1 - \frac{\epsilon}{2})^M n. \]

Massage...
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]

- \( m \) - best expert mistakes
- \( M \) - algorithm mistakes.

\[ \frac{1}{2^m} \leq \left(\frac{3}{4}\right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]

Solve for \( M \).

\[ M \leq (m + \log n) / \log(4/3) \leq 2.4(m + \log n) \]

Multiple by \( 1 - \varepsilon \) for incorrect experts...

\[ (1 - \varepsilon)^m \leq (1 - \varepsilon/2)^M n. \]

Massage...

\[ M \leq 2(1 + \varepsilon)m + \frac{2\ln n}{\varepsilon} \]
Analysis: continued.

\[ \frac{1}{2^m} \leq \sum_i w_i \leq \left(\frac{3}{4}\right)^M n. \]

- \( m \) - best expert mistakes
- \( M \) algorithm mistakes.

\[ \frac{1}{2^m} \leq \left(\frac{3}{4}\right)^M n. \]

Take log of both sides.

\[ -m \leq -M \log(4/3) + \log n. \]

Solve for \( M \).

\[ M \leq \frac{m + \log n}{\log(4/3)} \leq 2.4(m + \log n) \]

Multiple by \( 1 - \varepsilon \) for incorrect experts...

\[ (1 - \varepsilon)^m \leq (1 - \varepsilon^2)^M n. \]

Massage...

\[ M \leq 2(1 + \varepsilon)m + \frac{2\ln n}{\varepsilon} \]

Approaches a factor of two of best expert performance!
Best Analysis?

Two experts: A, B
Best Analysis?

Two experts: A, B

Bad example?
Best Analysis?

Two experts: A, B

Bad example?

Which is worse?

(A) A right on even, B right on odd.
(B) A right first half of days, B right second
Best Analysis?

Two experts: A, B

Bad example?

Which is worse?

(A) A right on even, B right on odd.
(B) A right first half of days, B right second

Best expert performance: $T/2$ mistakes.
Best Analysis?

Two experts: A, B

Bad example?

Which is worse?

(A) A right on even, B right on odd.

(B) A right first half of days, B right second

Best expert performance: $T/2$ mistakes.

Pattern (A): $T - 1$ mistakes.
Two experts: A, B

Bad example?
Which is worse?
(A) A right on even, B right on odd.
(B) A right first half of days, B right second

Best expert performance: $T/2$ mistakes.
Pattern (A): $T - 1$ mistakes.
Factor of (almost) two worse!
Randomization!!!

Better approach?
Randomization!!!!

Better approach?
Use?
Randomization!!!!

Better approach?
Use?
    Randomization!
Randomization!!!!

Better approach?
Use?
  Randomization!
That is, choose expert $i$ with prob $\propto w_i$
Randomization!!!

Better approach?
Use?

  Randomization!

That is, choose expert $i$ with prob $\propto w_i$

Bad example: A, B, A, B, A...
Better approach?
Use?

Randomization!
That is, choose expert $i$ with prob $\propto w_i$

Bad example: A,B,A,B,A...

After a bit, A and B make nearly the same number of mistakes.
Randomization!!!

Better approach?
Use?

Randomization!

That is, choose expert $i$ with prob $\propto w_i$

Bad example: A,B,A,B,A...

After a bit, A and B make nearly the same number of mistakes.
Choose each with approximately the same probabilit.
Randomization!!!!

Better approach?
Use?
  Randomization!
That is, choose expert $i$ with prob $\propto w_i$
Bad example: A,B,A,B,A...
After a bit, A and B make nearly the same number of mistakes.
Choose each with approximately the same probabilty.
Make a mistake around $1/2$ of the time.
Randomization!!!

Better approach?
Use?
  Randomization!
That is, choose expert $i$ with prob $\propto w_i$
Bad example: A,B,A,B,A...
After a bit, A and B make nearly the same number of mistakes.
Choose each with approximately the same probability.
Make a mistake around $1/2$ of the time.
Best expert makes $T/2$ mistakes.
Randomization!!!

Better approach?
Use?

Randomization!
That is, choose expert $i$ with prob $\propto w_i$

Bad example: A,B,A,B,A...

After a bit, A and B make nearly the same number of mistakes.
Choose each with approximately the same probability.
Make a mistake around 1/2 of the time.
Best expert makes $T/2$ mistakes.
Roughly
Randomization!!!!

Better approach?
Use?

Randomization!
That is, choose expert $i$ with prob $\propto w_i$

Bad example: A,B,A,B,A...

After a bit, A and B make nearly the same number of mistakes. Choose each with approximately the same probability.

Make a mistake around $1/2$ of the time.

Best expert makes $T/2$ mistakes.

Roughly optimal!
Randomized analysis.

Some formulas:
Randomized analysis.

Some formulas:
For $\varepsilon \leq 1, x \in [0, 1]$,
Randomized analysis.

Some formulas:

For $\varepsilon \leq 1, x \in [0, 1],$

$$(1 + \varepsilon)^x \leq (1 + \varepsilon x)$$
$$(1 - \varepsilon)^x \leq (1 - \varepsilon x)$$
Randomized analysis.

Some formulas:
For $\varepsilon \leq 1, x \in [0,1]$,
\[
(1 + \varepsilon)^x \leq (1 + \varepsilon x)
\]
\[
(1 - \varepsilon)^x \leq (1 - \varepsilon x)
\]
For $\varepsilon \in [0, \frac{1}{2}]$, 
\[
\ln(1 - \varepsilon^2) \leq -\varepsilon - \varepsilon^2 \leq \ln(1 + \varepsilon)
\]
Randomized analysis.

Some formulas:

For $\varepsilon \leq 1$, $x \in [0, 1]$,

$$ (1 + \varepsilon)^x \leq (1 + \varepsilon x) $$
$$ (1 - \varepsilon)^x \leq (1 - \varepsilon x) $$

For $\varepsilon \in [0, \frac{1}{2}]$,

$$ -\varepsilon - \frac{\varepsilon^2}{2} \ln(1 - \varepsilon) \leq -\varepsilon $$
$$ \varepsilon - \frac{\varepsilon^2}{2} \ln(1 + \varepsilon) \leq \varepsilon $$
Randomized analysis.

Some formulas:

For $\varepsilon \leq 1, x \in [0, 1],$

$$(1 + \varepsilon)^x \leq (1 + \varepsilon x)$$

$$(1 - \varepsilon)^x \leq (1 - \varepsilon x)$$

For $\varepsilon \in [0, \frac{1}{2}],$

$$-\varepsilon - \frac{\varepsilon^2}{2} \ln(1 - \varepsilon) \leq -\varepsilon$$

$$\varepsilon - \frac{\varepsilon^2}{2} \ln(1 + \varepsilon) \leq \varepsilon$$

Proof Idea: $\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots$
Randomized algorithm

Losses in $[0, 1]$. 

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $w_i W$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \epsilon )$ 

Best expert loses $L^*$ total.

$W(t) \geq (1 - \epsilon ) L^*$.

$L_t = w_i \ell_i$ expected loss of alg. in time $t$.

Claim:

$W(t+1) \leq W(t) (1 - \epsilon L_t)$ 

Proof:

$W(t+1) \leq \sum_i (1 - \epsilon \ell_i) w_i = \sum_i w_i - \epsilon \sum_i w_i \ell_i = \sum_i w_i (1 - \epsilon \sum_i w_i \ell_i) = W(t) (1 - \epsilon L_t)$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$. 
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$. 
Randomized algorithm

Losses in $[0,1]$.

Expert $i$ loses $\ell_i^t \in [0,1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$. 

Claim:

$W(t+1) \leq W(t)(1 - \varepsilon L_t)$
Randomized algorithm

Losses in $[0, 1]$. 

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$. 

1. Initially $w_i = 1$ for expert $i$. 
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$. 
3. $w_i \leftarrow w_i (1 - \epsilon)\ell_i^t$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. 

Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i(1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) =$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)\ell_i^t$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell^t_i \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)^{\ell^t_i}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total.
Randomized algorithm

Losses in $[0,1]$. Expert $i$ loses $\ell_i^t \in [0,1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)^{L^*}$. 
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $w_i / W$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \epsilon)\ell_i^t$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \epsilon)L^*$.

$L_t = \frac{w_i \ell_i^t}{W}$ expected loss of alg. in time $t$. 
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i \ell_i^t}{W}$ expected loss of alg. in time $t$.

Claim: $W(t + 1) \leq W(t)(1 - \varepsilon L_t)$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell^t_i \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i(1 - \varepsilon)^{\ell^t_i}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i\ell^t_i}{W}$ expected loss of alg. in time $t$.

Claim: $W(t + 1) \leq W(t)(1 - \varepsilon L_t)$

Proof:
Randomized algorithm
Losses in $[0,1]$.
Expert $i$ loses $\ell_i^t \in [0,1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i(1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i\ell_i^t}{W}$ expected loss of alg. in time $t$.

Claim: $W(t+1) \leq W(t)(1 - \varepsilon L_t)$

Proof: $W(t+1) \leq \sum_i (1 - \varepsilon \ell_i^t) w_i$
Randomized algorithm

Losses in $[0, 1]$. Expert $i$ loses $\ell_i^t \in [0, 1]$ in round t.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i(1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i\ell_i^t}{W}$ expected loss of alg. in time $t$.

Claim: $W(t + 1) \leq W(t)(1 - \varepsilon L_t)$

Proof:

$$W(t + 1) \leq \sum_i (1 - \varepsilon\ell_i^t)w_i = \sum_i w_i - \varepsilon \sum_i w_i\ell_i^t$$
Randomized algorithm

Losses in $[0, 1]$.
Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon) \ell_i^t$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$
Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i \ell_i^t}{W}$ expected loss of alg. in time $t$.

Claim: $W(t+1) \leq W(t)(1 - \varepsilon L_t)$

Proof:

$$W(t + 1) \leq \sum_i (1 - \varepsilon \ell_i^t) w_i = \sum_i w_i - \varepsilon \sum_i w_i \ell_i^t$$

$$= \sum_i w_i \left(1 - \varepsilon \frac{\sum_i w_i \ell_i^t}{\sum_i w_i} \right)$$
Randomized algorithm

Losses in $[0, 1]$.

Expert $i$ loses $\ell_i^t \in [0, 1]$ in round $t$.

1. Initially $w_i = 1$ for expert $i$.
2. Choose expert $i$ with prob $\frac{w_i}{W}$, $W = \sum_i w_i$.
3. $w_i \leftarrow w_i (1 - \varepsilon)^{\ell_i^t}$

$W(t)$ sum of $w_i$ at time $t$. $W(0) = n$

Best expert loses $L^*$ total. $\rightarrow W(T) \geq (1 - \varepsilon)L^*$.

$L_t = \frac{w_i \ell_i^t}{W}$ expected loss of alg. in time $t$.

Claim: $W(t + 1) \leq W(t)(1 - \varepsilon L_t)$

Proof:

$$W(t + 1) \leq \sum_i (1 - \varepsilon \ell_i^t)w_i = \sum_i w_i - \varepsilon \sum_i w_i \ell_i^t = \sum_i w_i \left(1 - \varepsilon \frac{\sum_i w_i \ell_i^t}{\sum_i w_i}\right) = W(t)(1 - \varepsilon L_t)$$
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod(1 - \varepsilon L_t)\]

Take logs
\[L^* \ln(1 - \varepsilon) \leq \ln n + \sum \ln(1 - \varepsilon L_t)\]
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs

\[L^* \ln (1 - \varepsilon) \leq \ln n + \sum \ln (1 - \varepsilon L_t)\]

Use \(-\varepsilon - \varepsilon^2 \leq \ln (1 - \varepsilon) \leq \varepsilon\)
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs
\[L^* \ln (1 - \varepsilon) \leq \ln n + \sum \ln (1 - \varepsilon L_t)\]

Use \[-\varepsilon - \varepsilon^2 \leq \ln (1 - \varepsilon) \leq \varepsilon\]
\[-L^* (\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs
\[L^* \ln (1 - \varepsilon) \leq \ln n + \sum \ln (1 - \varepsilon L_t)\]

Use \[-\varepsilon - \varepsilon^2 \leq \ln (1 - \varepsilon) \leq \varepsilon\]
\[-L^*(\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]

And
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs
\[L^* \ln (1 - \varepsilon) \leq \ln n + \sum \ln (1 - \varepsilon L_t)\]

Use \(-\varepsilon - \varepsilon^2 \leq \ln (1 - \varepsilon) \leq \varepsilon\)
\[-L^* (\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]

And
\[\sum L_t \leq L^* (1 + \varepsilon) + \frac{\ln n}{\varepsilon}.\]
Analysis

\[(1 - \varepsilon)L^* \leq W(T) \leq n\prod(1 - \varepsilon L_t)\]

Take logs
\[L^* \ln(1 - \varepsilon) \leq \ln n + \sum \ln(1 - \varepsilon L_t)\]

Use \[-\varepsilon - \varepsilon^2 \leq \ln(1 - \varepsilon) \leq \varepsilon\]
\[-L^*(\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]

And
\[\sum L_t \leq L^*(1 + \varepsilon) + \frac{\ln n}{\varepsilon}\]

Left hand side is the total expected loss of the experts algorithm.
(1 − ε)^L* ≤ W(T) ≤ n\Pi(1 − εL_t)

Take logs

L^* \ln(1 − ε) ≤ \ln n + \sum \ln(1 − εL_t)

Use −ε − ε^2 ≤ \ln(1 − ε) ≤ ε

−L^*(ε + ε^2) ≤ \ln n − ε \sum L_t

And

\sum L_t ≤ L^*(1 + ε) + \frac{\ln n}{ε}.

Left hand side is the total expected loss of the experts algorithm.

Within (1 + ε)
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs
\[L^* \ln(1 - \varepsilon) \leq \ln n + \sum \ln(1 - \varepsilon L_t)\]

Use \(-\varepsilon - \varepsilon^2 \leq \ln(1 - \varepsilon) \leq \varepsilon\)
\[\begin{align*}
-L^*(\varepsilon + \varepsilon^2) &\leq \ln n - \varepsilon \sum L_t \\
\end{align*}\]

And
\[\sum L_t \leq L^*(1 + \varepsilon) + \frac{\ln n}{\varepsilon}\]

Left hand side is the total expected loss of the experts algorithm.

Within \((1 + \varepsilon)\) ish
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod (1 - \varepsilon L_t)\]

Take logs
\[L^* \ln (1 - \varepsilon) \leq \ln n + \sum \ln (1 - \varepsilon L_t)\]

Use \[-\varepsilon - \varepsilon^2 \leq \ln (1 - \varepsilon) \leq \varepsilon\]
\[-L^*(\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]

And
\[\sum L_t \leq L^*(1 + \varepsilon) + \frac{\ln n}{\varepsilon}\]

Left hand side is the total expected loss of the experts algorithm.

Within \((1 + \varepsilon)\) ish of the best expert!
Analysis

\[(1 - \varepsilon)^{L^*} \leq W(T) \leq n \prod(1 - \varepsilon L_t)\]

Take logs
\[L^* \ln(1 - \varepsilon) \leq \ln n + \sum \ln(1 - \varepsilon L_t)\]

Use \(-\varepsilon - \varepsilon^2 \leq \ln(1 - \varepsilon) \leq \varepsilon\)

\[-L^*(\varepsilon + \varepsilon^2) \leq \ln n - \varepsilon \sum L_t\]

And
\[\sum L_t \leq L^*(1 + \varepsilon) + \frac{\ln n}{\varepsilon}\]

Left hand side is the total expected loss of the experts algorithm.

Within \((1 + \varepsilon) ish of the best expert!\)

No factor of 2 loss!
Summary: multiplicative weights.
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.

Perfect Expert: $\log n$ mistakes.

Imperfect Expert: best makes $m$ mistakes.

Deterministic Strategy: $2 \left(1 + \varepsilon\right)^m + \log m \varepsilon$

Real numbered losses: Best loses $L^*$ total.

Randomized Strategy: $\left(1 + \varepsilon\right)^{L^*} + \log m \varepsilon$

Strategy: choose according to weights updated by multiplying by $\left(1 - \varepsilon\right)$ loss.

Multiplicative weights framework!

Applications next!
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.
Perfect Expert: $\log n$ mistakes.

Deterministic Strategy: $2 \left(1 + \varepsilon\right)^m + \log m \varepsilon$

Randomized Strategy: $(1 + \varepsilon)^L + \log m \varepsilon$

Strategy: choose according to weights updated by multiplying by $(1 - \varepsilon) loss$. 

Multiplicative weights framework!
Applications next!
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.
Perfect Expert: $\log n$ mistakes.
Imperfect Expert: best makes $m$ mistakes.
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.
Perfect Expert: $\log n$ mistakes.
Imperfect Expert: best makes $m$ mistakes.
Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$

Real numbered losses: Best loses $L^*$ total.
Randomized Strategy: $(1 + \varepsilon)L^* + \log m + \varepsilon$
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.
Perfect Expert: $\log n$ mistakes.
Imperfect Expert: best makes $m$ mistakes.
Deterministic Strategy: $2(1 + \epsilon)m + \frac{\log m}{\epsilon}$
Real numbered losses: Best loses $L^*$ total.
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.
Perfect Expert: $\log n$ mistakes.
Imperfect Expert: best makes $m$ mistakes.
Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$
Real numbered losses: Best loses $L^*$ total.
Randomized Strategy: $(1 + \varepsilon)L^* + \frac{\log m}{\varepsilon}$
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day. Perfect Expert: $\log n$ mistakes. Imperfect Expert: best makes $m$ mistakes. Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$ Real numbered losses: Best loses $L^*$ total. Randomized Strategy: $(1 + \varepsilon)L^* + \frac{\log m}{\varepsilon}$ Strategy:
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.

Perfect Expert: $\log n$ mistakes.

Imperfect Expert: best makes $m$ mistakes.

Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$

Real numbered losses: Best loses $L^*$ total.

Randomized Strategy: $(1 + \varepsilon)L^* + \frac{\log m}{\varepsilon}$

Strategy:
choose according to weights updated by multiplying by $(1 - \varepsilon)^{\text{loss}}$. 
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.

Perfect Expert: $\log n$ mistakes.

Imperfect Expert: best makes $m$ mistakes.

Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$

Real numbered losses: Best loses $L^*$ total.

Randomized Strategy: $(1 + \varepsilon)L^* + \frac{\log m}{\varepsilon}$

Strategy:
choose according to weights updated by multiplying by $(1 - \varepsilon)^{\text{loss}}$.

Multiplicative weights framework!
Summary: multiplicative weights.

Framework: $n$ experts, each loses different amount every day.

Perfect Expert: $\log n$ mistakes.

Imperfect Expert: best makes $m$ mistakes.

Deterministic Strategy: $2(1 + \varepsilon)m + \frac{\log m}{\varepsilon}$

Real numbered losses: Best loses $L^*$ total.

Randomized Strategy: $(1 + \varepsilon)L^* + \frac{\log m}{\varepsilon}$

Strategy:
choose according to weights updated by multiplying by $(1 - \varepsilon)^\text{loss}$.

Multiplicative weights framework!

Applications next!