Comments on last lecture.

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.

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.

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.

Today

Finish Maximum Weight Matching Algorithm.

Today

Finish Maximum Weight Matching Algorithm.
  Exact algorithm with dueling players.

Today

Finish Maximum Weight Matching Algorithm.
    Exact algorithm with dueling players.

Multiplicative Weights Framework.

Today

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 $w : E \rightarrow R$, find a maximum weight matching.

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

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.

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.

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 an vertex cover function of minimum total value.

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

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

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 an 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 $w : E \to R$, 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 an 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

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 $w : E \to R$, 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 an 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

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.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

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

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

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

Add tight edges to matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.

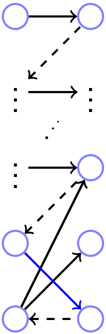
Algorithm

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

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

Maximum Weight Matching

Goal: perfect matching on tight edges.



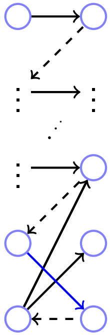
Algorithm

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

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

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

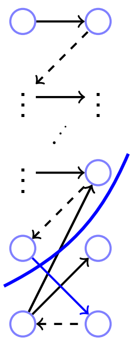
Start with empty matching, feasible cover function ($p(\cdot)$)

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

No augmenting path.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

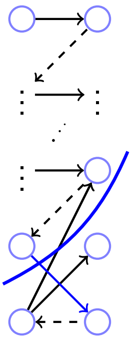
Start with empty matching, feasible cover function ($p(\cdot)$)

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!

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

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

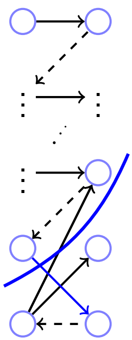
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

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

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

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."

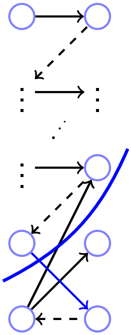
No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

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

Nontight edges leaving cut, go from S_U, T_V.

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."

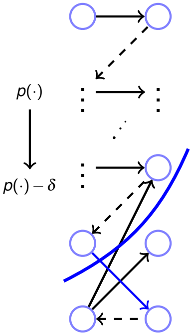
No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

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

Nontight edges leaving cut, go from S_U, T_V.

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!

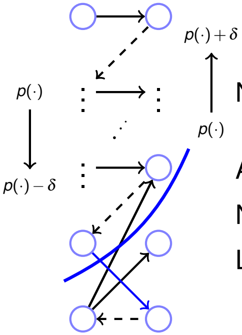
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U,

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!

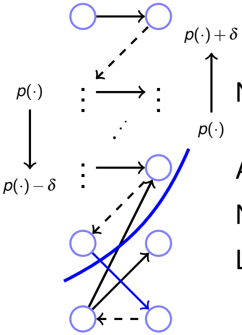
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_T,

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!

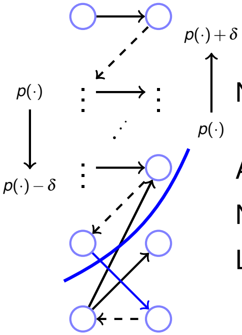
All edges across cut are not tight. (loose?)

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 ($p(\cdot)$)

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?)

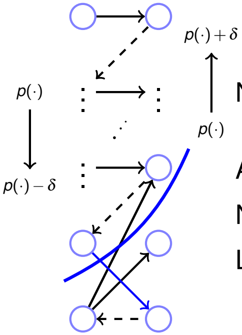
Nontight edges leaving cut, go from S_U, T_V.

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

... and get new tight edge!

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

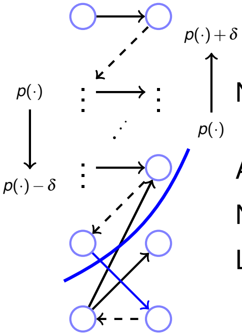
... and get new tight edge!
What's delta?

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 ($p(\cdot)$)

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

... and get new tight edge!
What's delta? $w(e) > p(u) + p(v) \rightarrow$
$\delta = min_{e \in (S_U \times T_V)} w(e) - p(u) - p(v)$.

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: $M = \{\}$.

Some details/Runtime

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

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

Feasible!

Some details/Runtime

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

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

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 $u \in U$,

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.

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:

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.

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.)

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.)

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

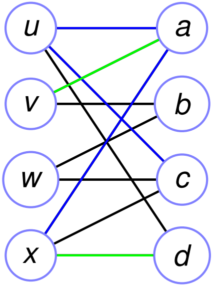
Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

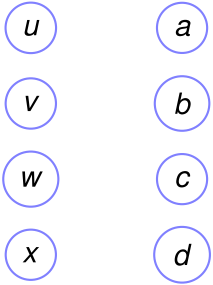
Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.
$O(n)$ augmentations.

$O(n^2 m)$ time.

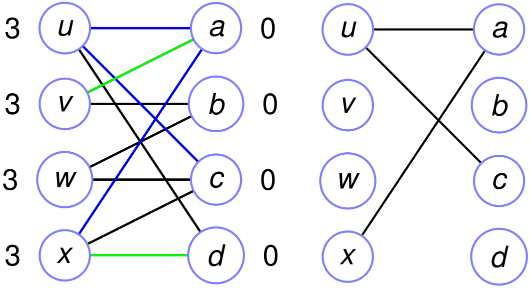
Example

Weight legend:
black 1, green 2, blue 3



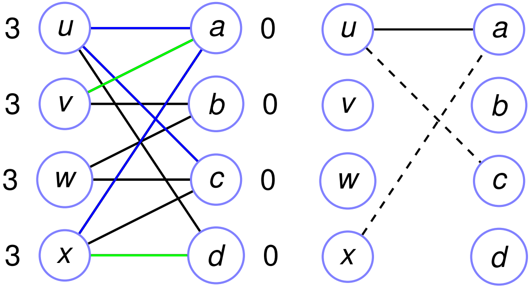


Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.

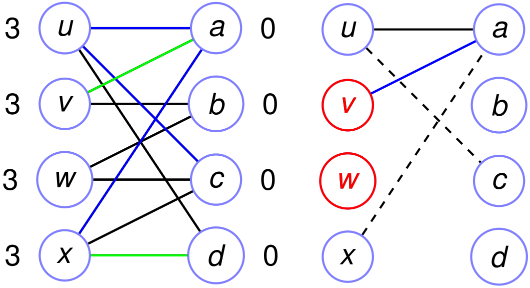
Example



Weight legend:
black 1, green 2, blue 3

Max matching in tight edges.
dashed means matched.

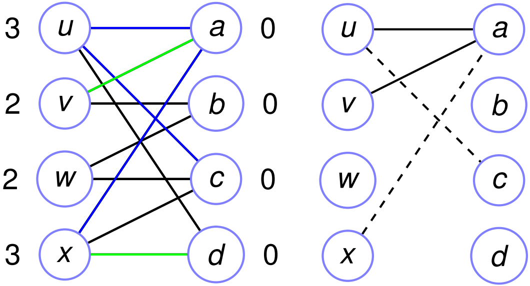
Example



Weight legend:
black 1, green 2, blue 3

No augmenting path →
reachable: $S = \{u, v\}$
Blue edges soon
to be tight!

Example

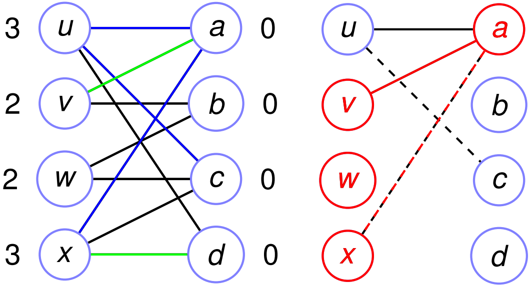


Weight legend:
black 1, green 2, blue 3

Adjust prices...
new tight edges.

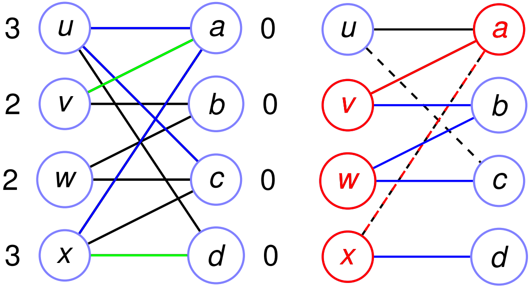
Example

Weight legend:
black 1, green 2, blue 3



Still no augmenting path.
Reachable $S = \{v, w, x, a\}$

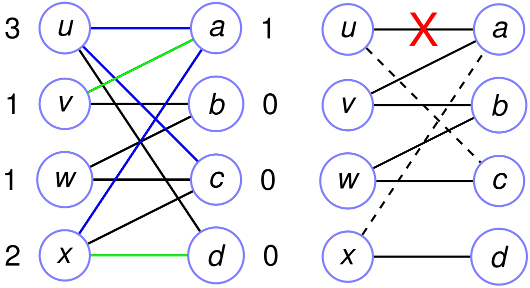
Example



Weight legend:
black 1, green 2, blue 3

Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.

Example

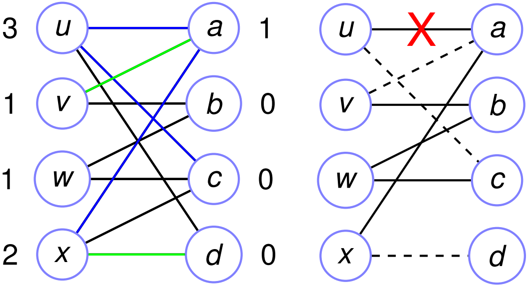


Weight legend:
black 1, green 2, blue 3

Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.

Example

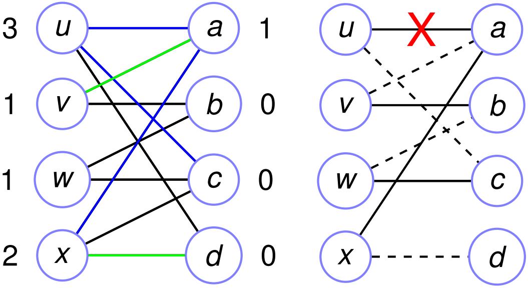
Weight legend:
black 1, green 2, blue 3



..and another augmentation...

Example

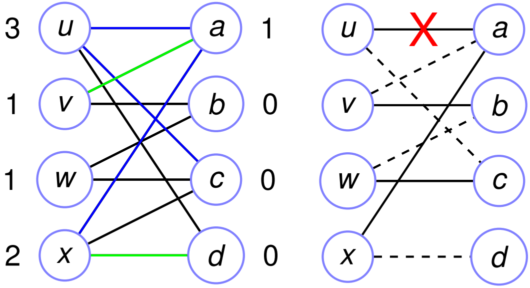
Weight legend:
black 1, green 2, blue 3



..and finally: a perfect matching.

Example

Weight legend:
black 1, green 2, blue 3

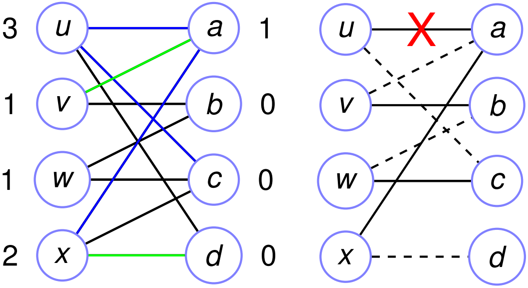


..and finally: a perfect matching.

All matched edges tight.

Example

Weight legend:
black 1, green 2, blue 3

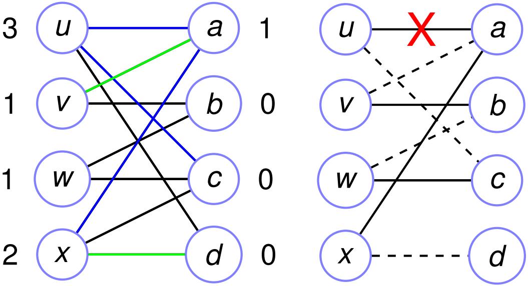


..and finally: a perfect matching.

All matched edges tight.
Perfect matching.

Example

Weight legend:
black 1, green 2, blue 3

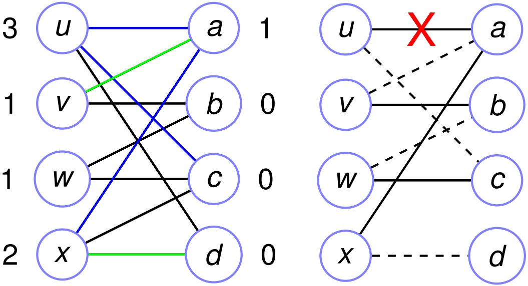


..and finally: a perfect matching.

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

Example

Weight legend:
black 1, green 2, blue 3



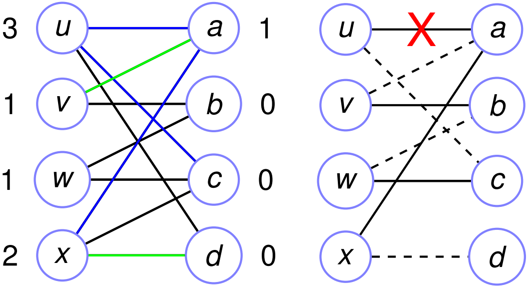
..and finally: a perfect matching.

All matched edges tight.

Perfect matching. Feasible price function. Values the same.

Example

Weight legend:
black 1, green 2, blue 3



..and finally: a perfect matching.

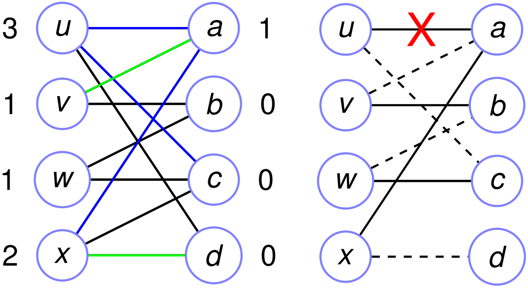
All matched edges tight.

Perfect matching. Feasible price function. Values the same.

Optimal!

Example

Weight legend:
black 1, green 2, blue 3



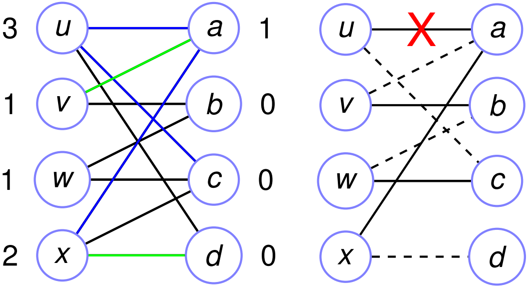
..and finally: a perfect matching.

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

Notice:

Example

Weight legend:
black 1, green 2, blue 3



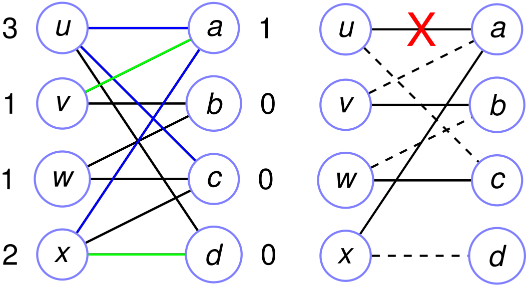
..and finally: a perfect matching.

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

Notice:
no weights on the right problem.

Example

Weight legend:
black 1, green 2, blue 3



..and finally: a perfect matching.

All matched edges tight.

Perfect matching. Feasible price function. Values the same.

Optimal!

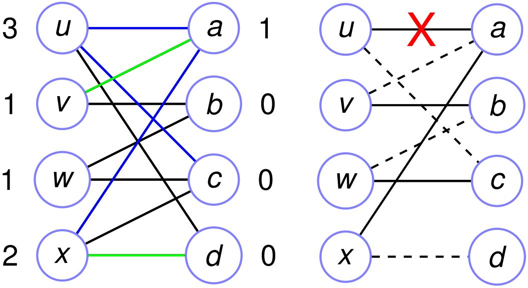
Notice:

no weights on the right problem.

retain previous matching through price changes.

Example

Weight legend:
black 1, green 2, blue 3



..and finally: a perfect matching.

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.

Expert's framework.

n experts.

Expert's framework.

n experts.

Every day, each offers a prediction.

Expert's framework.

n experts.

Every day, each offers a prediction.

"Rain" or "Shine."

Expert's framework.

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."

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.

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!

Concept Alert.

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.

Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

Back to mistake bound.

Infallible Experts.

Alg: Choose one of the perfect experts.

Mistake Bound: $n-1$

Back to mistake bound.

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!

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.

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 <u>mistake</u>,
$|$"perfect" experts$|$ drops 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 mistake $\rightarrow$ $\leq n/2$ 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 mistake $\rightarrow$ $\leq n/2$ perfect experts
mistake $\rightarrow$ $\leq n/4$ 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 <u>mistake</u>,
$|$“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

$\vdots$

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 $\rightarrow$ ≤ 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 <u>mistake</u>,
$|$"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 $\rightarrow$ ≤ 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 $\rightarrow$ ≤ 1 perfect expert

≥ 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 <u>mistake</u>,
$|$"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 $\rightarrow$ ≤ 1 perfect expert

≥ 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

Analysis: weighted majority

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.

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:

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. $w_i \to w_i/2$ if wrong.

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. $w_i \to w_i/2$ if wrong.

Analysis: weighted majority

Goal: Best expert makes m mistakes.

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

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

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:

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. 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. $w_i \rightarrow w_i/2$ 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?

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
-1? -2? factor of $\frac{1}{2}$?

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
-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. $w_i \rightarrow w_i/2$ 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

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. $w_i \rightarrow w_i/2$ 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? 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 $\frac{3}{4}$.

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 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.

$-m \leq -M \log(4/3) + logn.$

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.

$$-m \leq -M \log(4/3) + logn.$$

Solve for M.

$$M \leq (m + logn)/\log(4/3)$$

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.

$$-m \leq -M \log(4/3) + logn.$$

Solve for M.

$$M \leq (m + logn)/\log(4/3) \leq 2.4(m + \log n)$$

Analysis: continued.

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

m - best expert mistakes M algorithm mistakes.

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

Take log of both sides.

$-m \le -M \log(4/3) + logn.$

Solve for M.

$M \le (m + logn)/\log(4/3) \le 2.4(m + \log n)$

Multiple by $1 - \varepsilon$ for incorrect experts...

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.

$$-m \leq -M \log(4/3) + logn.$$

Solve for M.

$$M \leq (m + logn)/\log(4/3) \leq 2.4(m + \log n)$$

Multiple by $1 - \varepsilon$ for incorrect experts...

$$(1-\varepsilon)^m \leq \left(1 - \frac{\varepsilon}{2}\right)^M n.$$

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.

$$-m \leq -M \log(4/3) + logn.$$

Solve for M.

$$M \leq (m + logn)/\log(4/3) \leq 2.4(m + \log n)$$

Multiple by $1 - \varepsilon$ for incorrect experts...

$$(1-\varepsilon)^m \leq \left(1 - \frac{\varepsilon}{2}\right)^M n.$$

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.

$$-m \leq -M \log(4/3) + logn.$$

Solve for M.

$$M \leq (m + logn)/\log(4/3) \leq 2.4(m + \log n)$$

Multiple by $1-\varepsilon$ for incorrect experts...

$$(1-\varepsilon)^m \leq \left(1 - \frac{\varepsilon}{2}\right)^M n.$$

Massage...

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

Analysis: continued.

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

m - best expert mistakes M algorithm mistakes.

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

Take log of both sides.

$$-m \le -M\log(4/3) + logn.$$

Solve for M.

$$M \le (m + logn)/\log(4/3) \le 2.4(m + \log n)$$

Multiple by $1-\varepsilon$ for incorrect experts...

$$(1-\varepsilon)^m \le \left(1-\frac{\varepsilon}{2}\right)^M n.$$

Massage...

$$M \le 2(1+\varepsilon)m + \frac{2\ln n}{\varepsilon}$$

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 peformance: $T/2$ 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 peformance: $T/2$ mistakes.

Pattern (A): $T-1$ 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 peformance: $T/2$ 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...

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.

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.

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 probabilty.

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 probabilty.

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

Best expert makes $T/2$ mistakes.

Rougly

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.

Best expert makes $T/2$ mistakes.

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

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

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

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

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

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

Randomized algorithm

Losses in $[0, 1]$.

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 $\frac{w_i}{W}$, $W = \sum_i w_i$.

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. $w_i \leftarrow w_i(1-\varepsilon)^{\ell_i^t}$

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 $\ell_i^t \in [0,1]$ 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, b, 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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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 $\frac{w_i}{W}$, $W = \sum_i w_i$.
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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

$L_t = \sum_i \frac{w_i \ell_i^t}{W}$ 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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

$L_t = \sum_i \frac{w_i \ell_i^t}{W}$ 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 $\ell_i^t \in [0,1]$ 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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

Proof:

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

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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

Proof:

$$
\begin{aligned}
W(t+1) \leq \sum_i (1-\varepsilon \ell_i^t) w_i &= \sum_i w_i - \varepsilon \sum_i w_i \ell_i^t \\
&= \sum_i w_i \left(1 - \varepsilon \frac{\sum_i w_i \ell_i^t}{\sum_i w_i}\right)
\end{aligned}
$$

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, b, loses L^* total. $\rightarrow W(T) \geq w_b \geq (1-\varepsilon)^{L^*}$.

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

Claim: $W(t+1) \leq W(t)(1-\varepsilon L_t)$ Loss $\rightarrow$ weight loss.

Proof:

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

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \;\; \prod_t (1-\varepsilon L_t)$$

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \;\; \prod_t (1-\varepsilon L_t)$$

Take logs

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

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \;\; \prod_t (1-\varepsilon L_t)$$

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

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \ \prod_t (1-\varepsilon L_t)$$

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

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \;\; \prod_t (1-\varepsilon L_t)$$

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

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \ \ \prod_t (1-\varepsilon L_t)$$

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

$$\sum_t L_t \leq (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \ \ \prod_t (1-\varepsilon L_t)$$

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

$$\sum_t L_t \leq (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

$\sum_t L_t$ is total expected loss of algorithm.

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \ \ \prod_t (1-\varepsilon L_t)$$

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

$$\sum_t L_t \leq (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

$\sum_t L_t$ is total expected loss of algorithm.

Within $(1+\varepsilon)$

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \;\; \prod_t (1-\varepsilon L_t)$$

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

$$\sum_t L_t \leq (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

$\sum_t L_t$ is total expected loss of algorithm.

Within $(1+\varepsilon)$ ish

Analysis

$$(1-\varepsilon)^{L^*} \leq W(T) \leq n \ \prod_t (1-\varepsilon L_t)$$

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

$$\sum_t L_t \leq (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

$\sum_t L_t$ is total expected loss of algorithm.

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

Analysis

$$(1-\varepsilon)^{L^*} \le W(T) \le n \ \prod_t (1-\varepsilon L_t)$$

Take logs

$$(L^*)\ln(1-\varepsilon) \le \ln n + \sum \ln(1-\varepsilon L_t)$$

Use $-\varepsilon - \varepsilon^2 \le \ln(1-\varepsilon) \le -\varepsilon$

$$-(L^*)(\varepsilon+\varepsilon^2) \le \ln n - \varepsilon \sum L_t$$

And

$$\sum_t L_t \le (1+\varepsilon)L^* + \frac{\ln n}{\varepsilon}.$$

$\sum_t L_t$ is total expected loss of algorithm.

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

No factor of 2 loss!

Gains.

Why so negative?

Gains.

Why so negative?

Each day, each expert gives gain in $[0, 1]$.

Gains.

Why so negative?

Each day, each expert gives gain in $[0,1]$.

Multiplicative weights with $(1+\varepsilon)^{g_i^t}$.

Gains.

Why so negative?

Each day, each expert gives gain in $[0,1]$.

Multiplicative weights with $(1+\varepsilon)^{g_i^t}$.

$$G \geq (1-\varepsilon)G^* - \frac{\log n}{\varepsilon}$$

where G^* is payoff of best expert.

Gains.

Why so negative?

Each day, each expert gives gain in $[0,1]$.

Multiplicative weights with $(1+\varepsilon)^{g_i^t}$.

$$G \geq (1-\varepsilon)G^* - \frac{\log n}{\varepsilon}$$

where G^* is payoff of best expert.

Scaling:

Gains.

Why so negative?

Each day, each expert gives gain in $[0,1]$.

Multiplicative weights with $(1+\varepsilon)^{g_i^t}$.

$$G \geq (1-\varepsilon)G^* - \frac{\log n}{\varepsilon}$$

where G^* is payoff of best expert.

Scaling:

Not $[0,1]$, say $[0,\rho]$.

Gains.

Why so negative?

Each day, each expert gives gain in $[0,1]$.

Multiplicative weights with $(1+\varepsilon)^{g_i^t}$.

$$G \geq (1-\varepsilon)G^* - \frac{\log n}{\varepsilon}$$

where G^* is payoff of best expert.

Scaling:

Not $[0,1]$, say $[0,\rho]$.

$$L \leq (1+\varepsilon)L^* + \frac{\rho \log n}{\varepsilon}$$

Summary: multiplicative weights.

Summary: multiplicative weights.

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

Summary: multiplicative weights.

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

Perfect Expert: $\log n$ mistakes.

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

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

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

Real numbered losses: Best loses L^* total.

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

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

Real numbered losses: Best loses L^* total.

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

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

Real numbered losses: Best loses L^* total.

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

Strategy:

Choose proportional to 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: $2(1+\varepsilon)m + \frac{\log n}{\varepsilon}$

Real numbered losses: Best loses L^* total.

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

Strategy:

Choose proportional to weights

multiply weight 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: $2(1+\varepsilon)m + \frac{\log n}{\varepsilon}$

Real numbered losses: Best loses L^* total.

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

Strategy:

Choose proportional to weights

multiply weight 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: $2(1+\varepsilon)m + \frac{\log n}{\varepsilon}$

Real numbered losses: Best loses L^* total.

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

Strategy:

Choose proportional to weights

multiply weight 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: $2(1+\varepsilon)m + \frac{\log n}{\varepsilon}$

Real numbered losses: Best loses L^* total.

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

Strategy:
Choose proportional to weights
multiply weight by $(1-\varepsilon)^{\text{loss}}$.

Multiplicative weights framework!

Applications next!