The **Hedge**(η)Algorithm

Yoav Freund

January 6, 2010

Outline

Halving Algorithm

Outline

Halving Algorithm

Hedge(η)Algorithm

Hedging vs. Halving

Outline

Halving Algorithm

Hedge(η)Algorithm

- Hedging vs. Halving

Bound on total loss

- Upper bound on $\sum_{i=1}^{N} w_i^{T+1}$
- Lower bound on $\sum_{i=1}^{N} w_i^{T+1}$
- Combining Upper and Lower bounds

Outline

Halving Algorithm

Hedge(η)Algorithm

Hedging vs. Halving

Bound on total loss

Upper bound on $\sum_{i=1}^{N} w_i^{T+1}$

Lower bound on $\sum_{i=1}^{N} w_i^{T+1}$

Combining Upper and Lower bounds

tuning η

Outline

Halving Algorithm

Hedge(η)Algorithm
- Hedging vs. Halving

Bound on total loss
- Upper bound on $\sum_{i=1}^{N} w_i^{T+1}$
- Lower bound on $\sum_{i=1}^{N} w_i^{T+1}$
- Combining Upper and Lower bounds

tuning η

Lower Bounds

The halving algorithm

- Our goal is to predict a binary sequence:
  $x_1, x_2, \ldots, x_t, \ldots \quad x_t \in \{0, 1\}$

The halving algorithm

- Our goal is to predict a binary sequence:
  $x_1, x_2, \ldots, x_t, \ldots \quad x_t \in \{0, 1\}$
- We have N experts, at each time each expert makes a binary prediction.

The halving algorithm

- Our goal is to predict a binary sequence:
  $x_1, x_2, \ldots, x_t, \ldots \quad x_t \in \{0, 1\}$
- We have N experts, at each time each expert makes a binary prediction.
- **We know that one of the experts is perfect**

Example trace for Halving Algorithm

Example trace for Halving Algorithm

expert1
expert2
expert3
expert4
expert5
expert6
expert7
expert8

alg.

Example trace for Halving Algorithm

expert1
expert2
expert3
expert4
expert5
expert6
expert7
expert8

alg.
outcome

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	
outcome	

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	1
outcome	

Example trace for Halving Algorithm

	$t = 1$
expert1	1
expert2	1
expert3	0
expert4	1
expert5	1
expert6	0
expert7	1
expert8	1
alg.	1
outcome	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	
outcome	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	0
outcome	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$
expert1	1	1
expert2	1	0
expert3	0	-
expert4	1	0
expert5	1	0
expert6	0	-
expert7	1	1
expert8	1	1
alg.	1	0
outcome	1	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	
outcome	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	1
outcome	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$
expert1	1	1	1
expert2	1	0	-
expert3	0	-	-
expert4	1	0	-
expert5	1	0	-
expert6	0	-	-
expert7	1	1	1
expert8	1	1	1
alg.	1	0	1
outcome	1	1	1

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	
outcome	1	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	1
outcome	1	1	1	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$
expert1	1	1	1	1
expert2	1	0	-	-
expert3	0	-	-	-
expert4	1	0	-	-
expert5	1	0	-	-
expert6	0	-	-	-
expert7	1	1	1	1
expert8	1	1	1	0
alg.	1	0	1	1
outcome	1	1	1	0

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	
outcome	1	1	1	0	

Example trace for Halving Algorithm

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	0
outcome	1	1	1	0	

Example trace for Halving Algorithm

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
expert1	1	1	1	1	-
expert2	1	0	-	-	-
expert3	0	-	-	-	-
expert4	1	0	-	-	-
expert5	1	0	-	-	-
expert6	0	-	-	-	-
expert7	1	1	1	1	0
expert8	1	1	1	0	-
alg.	1	0	1	1	0
outcome	1	1	1	0	0

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.
- No stochastic assumptions whatsoever.

Mistake bound for Halving algorithm

- Each time algorithm makes a mistakes, the pool of perfect experts is halved (at least).
- We assume that at least one expert is perfect.
- Number of mistakes is at most $\log_2 N$.
- No stochastic assumptions whatsoever.
- Proof is based on combining a lower and upper bounds on the number of perfect experts.

The hedging problem

- N possible actions

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:
  - Algorithm chooses a distribution $\mathbf{p}^t$ over actions.

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:
  - Algorithm chooses a distribution $\mathbf{p}^t$ over actions.
  - Losses $0 \leq \ell_i^t \leq 1$ of all actions $i = 1, \ldots, N$ are revealed.

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:
  - Algorithm chooses a distribution $\mathbf{p}^t$ over actions.
  - Losses $0 \leq \ell_i^t \leq 1$ of all actions $i = 1, \ldots, N$ are revealed.
  - Algorithm suffers **expected** loss $\mathbf{p}^t \cdot \ell^t$

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:
  - Algorithm chooses a distribution $\mathbf{p}^t$ over actions.
  - Losses $0 \leq \ell_i^t \leq 1$ of all actions $i = 1, \ldots, N$ are revealed.
  - Algorithm suffers **expected** loss $\mathbf{p}^t \cdot \ell^t$
- **Goal:** minimize total expected loss

The hedging problem

- N possible actions
- At each time step $t = 1, 2, \ldots, T$:
  - Algorithm chooses a distribution $\mathbf{p}^t$ over actions.
  - Losses $0 \leq \ell_i^t \leq 1$ of all actions $i = 1, \ldots, N$ are revealed.
  - Algorithm suffers **expected** loss $\mathbf{p}^t \cdot \ell^t$
- **Goal:** minimize total expected loss
- Here we have stochasticity - but only in **algorithm**, not in **outcome**

Hedging vs. Halving

- Like halving - we want to zoom into best action (expert).

Hedging vs. Halving

- Like halving - we want to zoom into best action (expert).
- Unlike halving - no action is perfect.

Hedging vs. Halving

- Like halving - we want to zoom into best action (expert).
- Unlike halving - no action is perfect.
- Basic idea - reduce probability of lossy actions,
  but not all the way to zero.

Hedging vs. Halving

- Like halving - we want to zoom into best action (expert).
- Unlike halving - no action is perfect.
- Basic idea - reduce probability of lossy actions, but not all the way to zero.
- **Modified Goal:** minimize difference between expected total loss and minimal total loss of repeating one action.

$$\sum_{t=1}^{T} \mathbf{p}^t \cdot \ell^t - \min_i \left(\sum_{t=1}^{T} \ell_i^t \right)$$

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.
- action i = use prediction of expert i

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.
- action i = use prediction of expert i
- Now each iteration of game consistst of three steps:

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.
- action i = use prediction of expert i
- Now each iteration of game consistst of three steps:
  - Experts make predictions $e_i^t \in \{0, 1\}$

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.
- action i = use prediction of expert i
- Now each iteration of game consistst of three steps:
  - Experts make predictions $e_i^t \in \{0,1\}$
  - Algorithm predicts 1 with probability $\sum_{i:e_i^t=1} p_i^t$.

Using hedge to generalize halving alg.

- Suppose that there is no perfect expert.
- action i = use prediction of expert i
- Now each iteration of game consistst of three steps:
  - Experts make predictions $e_i^t \in \{0, 1\}$
  - Algorithm predicts 1 with probability $\sum_{i:e_i^t=1} p_i^t$.
  - outcome o_i^t is revealed. $\ell_i^t = 0$ if $e_i^t = o_i^t$, $\ell_i^t = 1$ otherwise.

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- Weight:

$$w_i^t = w_i^1 e^{-\eta L_i^t}$$

Note freedom to choose initial weight (w_i^1) $\sum_{i=1}^n w_i^1 = 1$.

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- Weight:

$$w_i^t = w_i^1 e^{-\eta L_i^t}$$

Note freedom to choose initial weight (w_i^1) $\sum_{i=1}^{n} w_i^1 = 1$.

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- Weight:

$$w_i^t = w_i^1 e^{-\eta L_i^t}$$

Note freedom to choose initial weight (w_i^1) $\sum_{i=1}^n w_i^1 = 1$.

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$
- Probability:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_i^t},$$

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- Weight:

$$w_i^t = w_i^1 e^{-\eta L_i^t}$$

Note freedom to choose initial weight (w_i^1) $\sum_{i=1}^n w_i^1 = 1$.

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$
- Probability:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_i^t},$$

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- Weight:

$$w_i^t = w_i^1 e^{-\eta L_i^t}$$

Note freedom to choose initial weight (w_i^1) $\sum_{i=1}^{n} w_i^1 = 1$.

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$
- Probability:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_i^t}, \quad \mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{j=1}^{N} w_i^t}$$

Choosing the initial weights

- Giving an action high initial weight makes alg perform well if that action performs well.

Choosing the initial weights

- Giving an action high initial weight makes alg perform well if that action performs well.
- If good action has low initial weight, our total loss will be larger.

Choosing the initial weights

- Giving an action high initial weight makes alg perform well if that action performs well.
- If good action has low initial weight, our total loss will be larger.
- As $\sum_{i=1}^{n} w_i^1 = 1$ increasing one weight implies decreasing some others.

Choosing the initial weights

- Giving an action high initial weight makes alg perform well if that action performs well.
- If good action has low initial weight, our total loss will be larger.
- As $\sum_{i=1}^{n} w_i^1 = 1$ increasing one weight implies decreasing some others.
- Plays a similar role to prior distribution in Bayesian algorithms.

Bound on the loss of **Hedge**(η)Algorithm

Bound on the loss of **Hedge**(η)Algorithm

- Theorem (main theorem)

  For any sequence of loss vectors $\ell^1, \ldots, \ell^T$*, and for any* $i \in \{1, \ldots, N\}$*, we have*

$$L_{\textbf{Hedge}(\eta)} \leq \frac{-\ln(w_i^1) + \eta L_i}{1 - e^{-\eta}}.$$

Bound on the loss of **Hedge**(η)Algorithm

- Theorem (main theorem)
  For any sequence of loss vectors $\ell^1, \ldots, \ell^T$, and for any $i \in \{1, \ldots, N\}$, we have

$$L_{\textbf{Hedge}(\eta)} \leq \frac{-\ln(w_i^1) + \eta L_i}{1 - e^{-\eta}}.$$

  - Proof: by combining upper and lower bounds on $\sum_{i=1}^{N} w_i^{T+1}$

Upper bound on $\sum_{i=1}^{N} w_i^{T+1}$

Lemma (upper bound)

For any sequence of loss vectors $\ell^1, \ldots, \ell^T$ *we have*

$$\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right) \leq -(1-e^{-\eta})L_{\mathbf{Hedge}(\eta)}.$$

Proof of upper bound (slide 1)

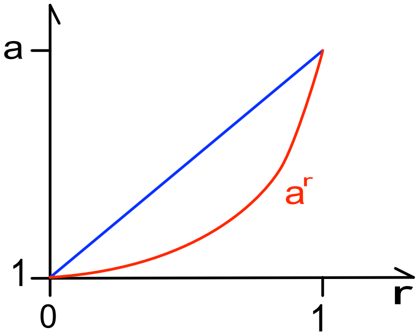
- If $a \geq 0$ then a^r is convex.

Proof of upper bound (slide 1)

- If $a \geq 0$ then a^r is convex.
- For $r \in [0, 1]$, $a^r \leq 1 - (1 - a)r$

Proof of upper bound (slide 1)

- If $a \geq 0$ then a^r is convex.
- For $r \in [0,1]$, $a^r \leq 1 - (1-a)r$



-

Proof of upper bound (slide 2)

Applying $a^r \leq 1 - (1-a)^r$ where $a = e^{-\eta}, r = \ell_i^t$

$$\sum_{i=1}^{N} w_i^{t+1} = \sum_{i=1}^{N} w_i^t e^{-\eta \ell_i^t}$$

Proof of upper bound (slide 2)

Applying $a^r \leq 1 - (1-a)^r$ where $a = e^{-\eta}, r = \ell_i^t$

$$\begin{aligned}\sum_{i=1}^N w_i^{t+1} &= \sum_{i=1}^N w_i^t e^{-\eta \ell_i^t} \\ &\leq \sum_{i=1}^N w_i^t \left(1 - (1 - e^{-\eta})\ell_i^t\right)\end{aligned}$$

Proof of upper bound (slide 2)

Applying $a^r \leq 1-(1-a)r$ where $a = e^{-\eta}, r = \ell_i^t$

$$
\begin{aligned}
\sum_{i=1}^N w_i^{t+1} &= \sum_{i=1}^N w_i^t e^{-\eta \ell_i^t} \\
&\leq \sum_{i=1}^N w_i^t \left(1-(1-e^{-\eta})\ell_i^t\right) \\
&= \left(\sum_{i=1}^N w_i^t\right)\left(1-(1-e^{-\eta})\frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}\cdot \ell^t\right)
\end{aligned}
$$

Proof of upper bound (slide 2)

Applying $a^r \leq 1 - (1-a)r$ where $a = e^{-\eta}, r = \ell_i^t$

$$\begin{aligned}
\sum_{i=1}^{N} w_i^{t+1} &= \sum_{i=1}^{N} w_i^t e^{-\eta \ell_i^t} \\
&\leq \sum_{i=1}^{N} w_i^t \left(1 - (1 - e^{-\eta})\ell_i^t\right) \\
&= \left(\sum_{i=1}^{N} w_i^t\right)\left(1 - (1 - e^{-\eta})\frac{\mathbf{w}^t}{\sum_{i=1}^{N} w_i^t} \cdot \ell^t\right) \\
&= \left(\sum_{i=1}^{N} w_i^t\right)\left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right)
\end{aligned}$$

Proof of upper bound (slide 3)

- Combining

$$\sum_{i=1}^{N} w_i^{t+1} \leq \left(\sum_{i=1}^{N} w_i^{t}\right) \left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right)$$

Proof of upper bound (slide 3)

- Combining

$$\sum_{i=1}^{N} w_i^{t+1} \leq \left(\sum_{i=1}^{N} w_i^{t}\right) \left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right)$$

- for $t = 1, \ldots, T$

Proof of upper bound (slide 3)

- Combining

$$\sum_{i=1}^{N} w_i^{t+1} \leq \left(\sum_{i=1}^{N} w_i^t\right) \left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right)$$

- for $t = 1, \ldots, T$
- yields

$$\sum_{i=1}^{N} w_i^{T+1} \leq \prod_{t=1}^{T} (1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t)$$

Proof of upper bound (slide 3)

- Combining

$$\sum_{i=1}^N w_i^{t+1} \leq \left(\sum_{i=1}^N w_i^t\right)(1-(1-e^{-\eta})\mathbf{p}^t \cdot \ell^t)$$

- for $t = 1, \ldots, T$
- yields

$$\sum_{i=1}^N w_i^{T+1} \quad \leq \quad \prod_{t=1}^T (1-(1-e^{-\eta})\mathbf{p}^t \cdot \ell^t)$$

Proof of upper bound (slide 3)

- Combining

$$\sum_{i=1}^{N} w_i^{t+1} \leq \left(\sum_{i=1}^{N} w_i^t\right) \left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right)$$

- for $t = 1, \ldots, T$
- yields

$$\begin{aligned} \sum_{i=1}^{N} w_i^{T+1} &\leq \prod_{t=1}^{T}\left(1 - (1 - e^{-\eta})\mathbf{p}^t \cdot \ell^t\right) \\ &\leq \exp\left(-(1 - e^{-\eta})\sum_{t=1}^{T} \mathbf{p}^t \cdot \ell^t\right) \end{aligned}$$

since $1 + x \leq e^x$ for $x = -(1 - e^{-\eta})$.

Lower bound on $\sum_{i=1}^{N} w_i^{T+1}$

For any $j = 1, \ldots, N$:

$$\sum_{i=1}^{N} w_i^{T+1} \geq w_j^{T+1} = w_j^1 e^{-\eta L_j}$$

Combining Upper and Lower bounds

- Combining bounds on $\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right)$

$$\ln w_j^1 - \eta L_j \leq \ln \sum_{i=1}^{N} w_i^{T+1} \leq -(1-e^{-\eta}) \sum_{t=1}^{T} \mathbf{p}^t \cdot \ell^t$$

Combining Upper and Lower bounds

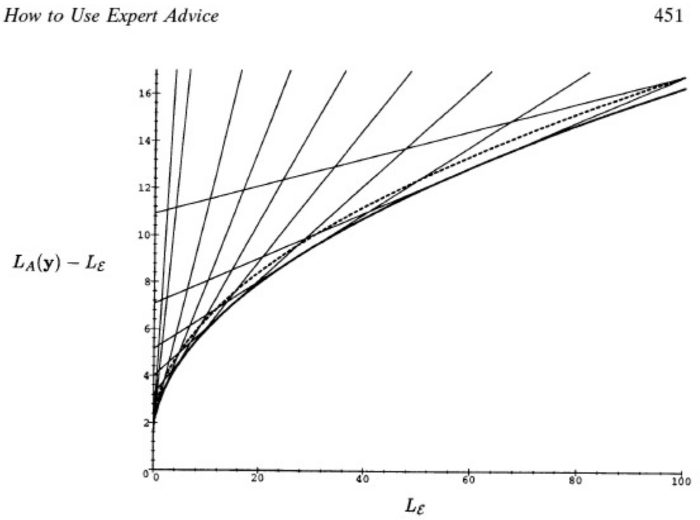
- Combining bounds on $\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right)$

$$\ln w_j^1 - \eta L_j \leq \ln \sum_{i=1}^{N} w_i^{T+1} \leq -(1-e^{-\eta}) \sum_{t=1}^{T} \mathbf{p}^t \cdot \ell^t$$

- Reversing signs, using $L_{\textbf{Hedge}(\eta)} = \sum_{t=1}^{T} \mathbf{p}^t \cdot \ell^t$ and reorganizing we get

$$L_{\textbf{Hedge}(\eta)} \leq \frac{-\ln(w_i^1) + \eta L_i}{1 - e^{-\eta}}$$

Tuning η


How to Use Expert Advice
451
$L_A(\mathbf{y}) - L_{\mathcal{E}}$
$L_{\mathcal{E}}$
0
2
4
6
8
10
12
14
16
0
20
40
60
80
100

Tuning η

- Suppose $\min_i L_i \leq \tilde{L}$

Tuning η

- Suppose $\min_i L_i \leq \tilde{L}$
- set
$$\eta = \ln\left(1 + \sqrt{\frac{2 \ln N}{\tilde{L}}}\right) \approx \sqrt{\frac{2 \ln N}{\tilde{L}}}$$

Tuning η

- Suppose $\min_i L_i \le \tilde{L}$
- set
$$\eta = \ln\left(1 + \sqrt{\frac{2\ln N}{\tilde{L}}}\right) \approx \sqrt{\frac{2\ln N}{\tilde{L}}}$$
- use uniform initial weights $\mathbf{w}^1 = \langle 1/N, \ldots, 1/N \rangle$

Tuning η

- Suppose $\min_i L_i \leq \tilde{L}$
- set

$$\eta = \ln\left(1 + \sqrt{\frac{2\ln N}{\tilde{L}}}\right) \approx \sqrt{\frac{2\ln N}{\tilde{L}}}$$

- use uniform initial weights $\mathbf{w}^1 = \langle 1/N, \ldots, 1/N \rangle$
- Then

$$L_{\textbf{Hedge}(\eta)} \leq \frac{-\ln(w_i^1) + \eta L_i}{1 - e^{-\eta}} \leq \min_i L_i + \sqrt{2\tilde{L}\ln N} + \ln N$$

Tuning η as a function of T

- trivially $\min_i L_i \leq T$, yielding

$$L_{\textbf{Hedge}(\eta)} \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$

Tuning η as a function of T

- trivially $\min_i L_i \leq T$, yielding

$$L_{\textbf{Hedge}(\eta)} \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$

- per iteration we get:

$$\frac{L_{\textbf{Hedge}(\eta)}}{T} \leq \min_i \frac{L_i}{T} + \sqrt{\frac{2 \ln N}{T}} + \frac{\ln N}{T}$$

How good is this bound?

- Very good! There is a closely matching lower bound!

How good is this bound?

- Very good! There is a closely matching lower bound!
- There exists a stochastic adversarial strategy such that with high probability for any hedging strategy **S** after T trials

$$L_{\mathbf{S}} - \min_i L_i \geq (1 - o(1))\sqrt{2T \ln N}$$

How good is this bound?

- Very good! There is a closely matching lower bound!
- There exists a stochastic adversarial strategy such that with high probability for any hedging strategy **S** after T trials

$$L_{\mathbf{S}} - \min_i L_i \geq (1 - o(1))\sqrt{2T \ln N}$$

- The adversarial strategy is random, extremely simple, and does not depend on the hedging strategy!

The adversarial strategy

- Adversary sets each loss ℓ_i^t indepedently at random to 0 or 1 with equal probabilities $(1/2, 1/2)$.

The adversarial strategy

- Adversary sets each loss ℓ_i^t indepedently at random to 0 or 1 with equal probabilities $(1/2, 1/2)$.
- Obviously, nothing to learn !
  $L_{\mathbf{S}} \approx T/2$.

The adversarial strategy

- Adversary sets each loss ℓ_i^t independently at random to 0 or 1 with equal probabilities $(1/2, 1/2)$.
- Obviously, nothing to learn !
  $L_{\mathbf{S}} \approx T/2$.
- On the other hand $\min_i L_i \approx T/2 - \sqrt{2T \ln N}$

The adversarial strategy

- Adversary sets each loss ℓ_i^t indepedently at random to 0 or 1 with equal probabilities $(1/2, 1/2)$.
- Obviously, nothing to learn !
  $L_{\mathbf{S}} \approx T/2$.
- On the other hand $\min_i L_i \approx T/2 - \sqrt{2T \ln N}$
- The difference $L_{\mathbf{S}} - \min_i L_i$ is due to unlearnable random fluctuations!

The adversarial strategy

- Adversary sets each loss ℓ_i^t indepedently at random to 0 or 1 with equal probabilities $(1/2, 1/2)$.
- Obviously, nothing to learn !
  $L_{\mathbf{S}} \approx T/2$.
- On the other hand $\min_i L_i \approx T/2 - \sqrt{2T \ln N}$
- The difference $L_{\mathbf{S}} - \min_i L_i$ is due to unlearnable random fluctuations!
- Detailed proof quite involved. See games paper.

Summary

- Given learning rate η the **Hedge**(η)algorithm satisfies

$$L_{\textbf{Hedge}(\eta)} \leq \frac{\ln N + \eta L_i}{1 - e^{-\eta}}$$

Summary

- Given learning rate η the **Hedge**(η)algorithm satisfies

$$L_{\textbf{Hedge}(\eta)} \leq \frac{\ln N + \eta L_i}{1 - e^{-\eta}}$$

- Setting $\eta \approx \sqrt{\frac{2 \ln N}{T}}$ guarantees

$$L_{\textbf{Hedge}(\eta)} \leq \min_i L_i + \sqrt{2 T \ln N} + \ln N$$

Summary

- Given learning rate η the **Hedge**(η)algorithm satisfies

$$L_{\textbf{Hedge}(\eta)} \le \frac{\ln N + \eta L_i}{1 - e^{-\eta}}$$

- Setting $\eta \approx \sqrt{\frac{2 \ln N}{T}}$ guarantees

$$L_{\textbf{Hedge}(\eta)} \le \min_i L_i + \sqrt{2T \ln N} + \ln N$$

- A trivial random data, in which there is nothing to be learned forces any algorithm to suffer this total loss

Some loose threads

- Total Loss of best action usually scales linearly with time, but we can't change η on the fly. NormalHedge will be explained later in the course.

Some loose threads

- Total Loss of best action usually scales linearly with time, but we can't change η on the fly. NormalHedge will be explained later in the course.
- Observing only the loss of chosen action - the multi-armed bandit problem. Will get to that later in the course.

Some loose threads

- Total Loss of best action usually scales linearly with time, but we can't change η on the fly. NormalHedge will be explained later in the course.
- Observing only the loss of chosen action - the multi-armed bandit problem. Will get to that later in the course.
- Send me email yfreund@ucsd.edu

Some loose threads

- Total Loss of best action usually scales linearly with time, but we can't change η on the fly. NormalHedge will be explained later in the course.
- Observing only the loss of chosen action - the multi-armed bandit problem. Will get to that later in the course.
- Send me email yfreund@ucsd.edu
- Register on the Wiki, and post your project ideas.

Some loose threads

- Total Loss of best action usually scales linearly with time, but we can't change η on the fly. NormalHedge will be explained later in the course.
- Observing only the loss of chosen action - the multi-armed bandit problem. Will get to that later in the course.
- Send me email yfreund@ucsd.edu
- Register on the Wiki, and post your project ideas.
- next time: Using experts for estimation and control - a large set of possible projects!