The Polynomial Weights Algorithm: Warmup

Aaron Roth

University of Pennsylvania

February 6 2024

Overview

- We've studied when Best Responds Dynamics converges...

Overview

- We've studied when Best Responds Dynamics converges...
- But it doesn't always, even in simple games.

Overview

- We've studied when Best Responds Dynamics converges...
- But it doesn't always, even in simple games.
- Even if they have pure strategy Nash equilibria! (Example)

Overview

- We've studied when Best Responds Dynamics converges...
- But it doesn't always, even in simple games.
- Even if they have pure strategy Nash equilibria! (Example)
- In such games, how should players behave?

Overview

- We've studied when Best Responds Dynamics converges...
- But it doesn't always, even in simple games.
- Even if they have pure strategy Nash equilibria! (Example)
- In such games, how should players behave?
- This lecture: learning in games.

Overview

- We've studied when Best Responds Dynamics converges...
- But it doesn't always, even in simple games.
- Even if they have pure strategy Nash equilibria! (Example)
- In such games, how should players behave?
- This lecture: learning in games.
- First we'll abstract away the game...

Sequential Prediction

A simple example—Stock prediction:

Sequential Prediction

A simple example—Stock prediction:

1. Every day GME goes *up* or *down*.

Sequential Prediction

A simple example—Stock prediction:

1. Every day GME goes *up* or *down*.
2. Your goal: Predict direction each day before the market opens (so you can buy or short)

Sequential Prediction

A simple example—Stock prediction:

1. Every day GME goes *up* or *down*.
2. Your goal: Predict direction each day before the market opens (so you can buy or short)
3. The market can behave arbitrarily/adversarially... So no way you can promise to do *well*.

Sequential Prediction

A simple example—Stock prediction:

1. Every day GME goes *up* or *down*.
2. Your goal: Predict direction each day before the market opens (so you can buy or short)
3. The market can behave arbitrarily/adversarially... So no way you can promise to do *well*.
4. But... You get advice.

Sequential Prediction

Expert Advice:

1. Before the bell every day, N experts whisper in your ear a guess: (U)p or (D)own.

Sequential Prediction

Expert Advice:

1. Before the bell every day, N experts whisper in your ear a guess: (U)p or (D)own.
2. Self-proclaimed “experts” — no promise they know what they are talking about.

Sequential Prediction

Expert Advice:

1. Before the bell every day, N experts whisper in your ear a guess: (U)p or (D)own.
2. Self-proclaimed "experts" — no promise they know what they are talking about.
3. Your goal: Aggregate expert advice so that after awhile you do (almost) as well as the *best* expert in hindsight.

Sequential Prediction

Expert Advice:

1. Before the bell every day, N experts whisper in your ear a guess: (U)p or (D)own.
2. Self-proclaimed "experts" — no promise they know what they are talking about.
3. Your goal: Aggregate expert advice so that after awhile you do (almost) as well as the *best* expert in hindsight.
4. Lets start with an easier case.

An Easier Case

- There are N experts who will make predictions in T rounds.

An Easier Case

- There are N experts who will make predictions in T rounds.
- At each round t, each expert i makes a prediction $p_i^t \in \{U, D\}$ (up or down).

An Easier Case

- There are N experts who will make predictions in T rounds.
- At each round t, each expert i makes a prediction $p_i^t \in \{U, D\}$ (up or down).
- We (the algorithm) aggregate these predictions somehow, to make our own prediction $p_A^t \in \{U, D\}$. Then we learn the true outcome $o^t \in \{U, D\}$. If we predicted incorrectly (i.e. $p_A^t \neq o^t$), then we *made a mistake*.

An Easier Case

- There are N experts who will make predictions in T rounds.
- At each round t, each expert i makes a prediction $p_i^t \in \{U, D\}$ (up or down).
- We (the algorithm) aggregate these predictions somehow, to make our own prediction $p_A^t \in \{U, D\}$. Then we learn the true outcome $o^t \in \{U, D\}$. If we predicted incorrectly (i.e. $p_A^t \neq o^t$), then we *made a mistake*.
- To make things easy, we will assume at first that there is one *perfect* expert who never makes a mistake (but we don't know who he is).

An Easier Case

- There are N experts who will make predictions in T rounds.
- At each round t, each expert i makes a prediction $p_i^t \in \{U, D\}$ (up or down).
- We (the algorithm) aggregate these predictions somehow, to make our own prediction $p_A^t \in \{U, D\}$. Then we learn the true outcome $o^t \in \{U, D\}$. If we predicted incorrectly (i.e. $p_A^t \neq o^t$), then we *made a mistake*.
- To make things easy, we will assume at first that there is one *perfect* expert who never makes a mistake (but we don't know who he is).

Can we find a strategy that is guaranteed to make at most $\log(N)$ *mistakes?*

The Halving Algorithm

Algorithm 1 The Halving Algorithm

Let $S^1 \leftarrow \{1, \ldots, N\}$ be the set of all experts.

for $t = 1$ to T **do**

Let $S_U^t = \{i \in S : p_i^t = U\}$ be the set of experts in S^t who predict up, and $S_D^t = S^t \setminus S_U^t$ be the set who predict down.

Predict with the majority vote: If $|S_U^t| > |S_D^t|$, predict $p_A^t = U$, else predict $p_A^t = D$.

Eliminate all experts that made a mistake: If $o^T = U$, then let $S^{t+1} = S_U^t$, else let $S^{t+1} = S_D^t$

end for

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

Proof.

1. The algorithm predicts with the majority vote, so every time it makes a mistake at some round t, at least half of the remaining experts have made a mistake and are eliminated.

□

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

Proof.

1. The algorithm predicts with the majority vote, so every time it makes a mistake at some round t, at least half of the remaining experts have made a mistake and are eliminated.
2. Hence $|S^{t+1}| \leq |S^t|/2$.

□

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

Proof.

1. The algorithm predicts with the majority vote, so every time it makes a mistake at some round t, at least half of the remaining experts have made a mistake and are eliminated.
2. Hence $|S^{t+1}| \leq |S^t|/2$.
3. On the other hand, the perfect expert is never eliminated.

□

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

Proof.

1. The algorithm predicts with the majority vote, so every time it makes a mistake at some round t, at least half of the remaining experts have made a mistake and are eliminated.
2. Hence $|S^{t+1}| \leq |S^t|/2$.
3. On the other hand, the perfect expert is never eliminated.
4. Hence $|S^t| \geq 1$ for all t.

□

The Halving Algorithm

Theorem

If there is at least one perfect expert, the halving algorithm makes at most $\log N$ *mistakes.*

Proof.

1. The algorithm predicts with the majority vote, so every time it makes a mistake at some round t, at least half of the remaining experts have made a mistake and are eliminated.
2. Hence $|S^{t+1}| \leq |S^t|/2$.
3. On the other hand, the perfect expert is never eliminated.
4. Hence $|S^t| \geq 1$ for all t.
5. Since $|S^1| = N$, this means there can be at most $\log N$ mistakes.

□

The Halving Algorithm

Is this bound any good?

The Halving Algorithm

Is this bound any good?

1. Of course we've made a big assumption: a perfect expert.

The Halving Algorithm

Is this bound any good?

1. Of course we've made a big assumption: a perfect expert.
2. But $\log N$ is pretty small even if N is large (e.g. if $N = 1024$, $\log N = 10$, if $N = 1,048,576$, $\log N = 20$)

The Halving Algorithm

Is this bound any good?

1. Of course we've made a big assumption: a perfect expert.
2. But $\log N$ is pretty small even if N is large (e.g. if $N = 1024$, $\log N = 10$, if $N = 1,048,576$, $\log N = 20$)
3. And the bound doesn't grow with T, so even with a huge number of experts, the average number of mistakes made by this algorithm is tiny.

The Halving Algorithm

Is this bound any good?

1. Of course we've made a big assumption: a perfect expert.
2. But $\log N$ is pretty small even if N is large (e.g. if $N = 1024$, $\log N = 10$, if $N = 1,048,576$, $\log N = 20$)
3. And the bound doesn't grow with T, so even with a huge number of experts, the average number of mistakes made by this algorithm is tiny.
4. But what if no expert is perfect? Say the best expert makes OPT mistakes.

The Halving Algorithm

Is this bound any good?

1. Of course we've made a big assumption: a perfect expert.
2. But $\log N$ is pretty small even if N is large (e.g. if $N = 1024$, $\log N = 10$, if $N = 1,048,576$, $\log N = 20$)
3. And the bound doesn't grow with T, so even with a huge number of experts, the average number of mistakes made by this algorithm is tiny.
4. But what if no expert is perfect? Say the best expert makes OPT mistakes.
5. Can we find a way to make not too many more than OPT mistakes?

The Iterated Halving Algorithm

Algorithm 2 The Iterated Halving Algorithm

Let $S^1 \leftarrow \{1, \ldots, N\}$ be the set of all experts.
for $t = 1$ to T **do**
  If $|S^t| = 0$ **Reset**: Set $S^t \leftarrow \{1, \ldots, N\}$.
  Let $S^t_U = \{i \in S : p^t_i = U\}$ be the set of experts in S^t who predict up, and $S^t_D = S^t \setminus S^t_U$ be the set who predict down.
  Predict with the majority vote: If $|S^t_U| > |S^t_D|$, predict $p^t_A = U$, else predict $p^t_A = D$.
  Eliminate all experts that made a mistake: If $o^T = U$, then let $S^{t+1} = S^t_U$, else let $S^{t+1} = S^t_D$
end for

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ *mistakes.*

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ *mistakes.*

Proof.

1. Whenever the algorithm makes a mistake, we eliminate half of the experts.

□

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT} + 1)$ *mistakes.*

Proof.

1. Whenever the algorithm makes a mistake, we eliminate half of the experts.
2. So the algorithm can make at most $\log N$ mistakes between any two resets.

□

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ *mistakes.*

Proof.

1. Whenever the algorithm makes a mistake, we eliminate half of the experts.
2. So the algorithm can make at most $\log N$ mistakes between any two resets.
3. But if we reset, it is because since the last reset, *every* expert has made a mistake.

□

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ *mistakes.*

Proof.

1. Whenever the algorithm makes a mistake, we eliminate half of the experts.
2. So the algorithm can make at most $\log N$ mistakes between any two resets.
3. But if we reset, it is because since the last reset, *every* expert has made a mistake.
4. in particular, between any two resets, the *best* expert has made at least 1 mistake.

□

The Iterated Halving Algorithm

Theorem

The iterated halving algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ *mistakes.*

Proof.

1. Whenever the algorithm makes a mistake, we eliminate half of the experts.
2. So the algorithm can make at most $\log N$ mistakes between any two resets.
3. But if we reset, it is because since the last reset, *every* expert has made a mistake.
4. in particular, between any two resets, the *best* expert has made at least 1 mistake.
5. This gives the claimed bound.

□

The Iterated Halving Algorithm

1. We should be able to do better though.

The Iterated Halving Algorithm

1. We should be able to do better though.
2. The above algorithm is wasteful in that every time we reset, we forget what we have learned!

The Iterated Halving Algorithm

1. We should be able to do better though.
2. The above algorithm is wasteful in that every time we reset, we forget what we have learned!
3. What should we do instead?

The Iterated Halving Algorithm

1. We should be able to do better though.
2. The above algorithm is wasteful in that every time we reset, we forget what we have learned!
3. What should we do instead?
4. To be continued...

Thanks!

See you next class — stay healthy!