UPenn NETS 412: Algorithmic Game Theory January 25, 2018

Lecture 4

Lecturer: Bo Waggoner *Scribe: Aaron Roth, Bo Waggoner*

Learning from Expert Advice

In the last several lectures, we have seen several games in which best response dynamics inevitably converges to a Nash equilibrium; however, there are also plenty of games for which best response dynamics does not converge (even if they have pure strategy Nash equilibria!). In such games, what should players do? What should we expect to happen? And how should we *compute* equilibria?

In this lecture, we will introduce a model of a single agent repeatedly making decisions and observing feedback over time. This can capture playing a game repeatedly, but it can also apply to more general scenarios. So for a while, we will move away from considering games and consider this more general problem. Then, we will come back and see how it can be useful for games.

As a simple example to keep in mind, consider the following toy model of predicting the stock market: every day the market goes *up* or *down*, and you must predict what it will do before it happens (so that you can either buy or short shares). You don't have any information about what the market will do, and it may behave arbitrarily, so you can't hope to do well in an absolute sense. However, every day, before you make your prediction, you get to hear the advice of a bunch of experts, who make their own predictions. These "experts" may or may not know what they are talking about, and you start off knowing nothing about them. Nevertheless, you want to come up with a rule to aggregate their advice so that you end up doing (almost) as well as the best expert (whomever she might turn out to be) in hindsight. Sounds tough.

Lets start with an even 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 she is).

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

We can, using the simple 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
```

Its not hard to see that the halving algorithm makes at most $\log N$ mistakes under the assumption that one expert is perfect:

Theorem 1 *If there is at least one perfect expert, the halving algorithm makes at most* $\log N$ *mistakes.*

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

Not bad – $\log N$ is pretty small even if N is large (e.g. if $N = 1024$, $\log N = 10$, if $N = 1,048,576$, $\log N = 20$), and doesn't grow with T, so even with a huge number of experts, the average number of mistakes made by this algorithm is tiny.

What if no expert is perfect? Suppose the best expert makes OPT mistakes. Can we find a way to make not too many more than OPT mistakes?

The first approach you might try is the iterated halving algorithm:

Algorithm 2 The Iterated Halving Algorithm

```
Let $S^1 \leftarrow \{1, \dots, N\}$ be the set of all experts.
for $t = 1$ to $T$ do
  If $|S^t| = 0$ Reset: Set $S^t \leftarrow \{1, \dots, N\}$.
  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
```

Theorem 2 *The iterated halving algorithm makes at most* $\log(N)(OPT + 1)$ *mistakes.*

Proof As before, whenever the algorithm makes a mistake, we eliminate half of the experts, and so the algorithm can make at most $\log N$ mistakes between any two resets. But if we reset, it is because since the last reset, *every* expert has made a mistake: in particular, between any two resets, the *best* expert has made at least 1 mistake. This gives the claimed bound. ■

We should be able to do better, though. The iterated halving algorithm is wasteful in that every time we reset, we forget what we have learned! The weighted majority algorithm can be viewed as a "softer" version of the halving algorithm: rather than eliminating experts who make mistakes, we just down-weight them:

Algorithm 3 The Weighted Majority Algorithm

```
Set weights $w_i^1 \leftarrow 1$ for all experts $i$.
for $t = 1$ to $T$ do
  Let $W_U^t = \sum_{i:p_i^t=U} w_i$ be the weight of experts who predict up, and $W_D^t = \sum_{i:p_i^t=D} w_i$ be the weight of those who predict down.
  Predict with the weighted majority vote: If $W_U^t > W_D^t$, predict $p_A^t = U$, else predict $p_A^t = D$.
  Down-weight experts who made mistakes: For all $i$ such that $p_i^t \neq o^t$, set $w_i^{t+1} \leftarrow w_i^t/2$
end for
```