CS 683 — Learning, Games, and Electronic Markets **Spring 2007**

Notes from Week 2: Prediction algorithms and zero-sum games

Instructor: Robert Kleinberg *29 Jan – 2 Feb 2007*

1 Summary of Week 1

Here are some closing observations about sequential prediction algorithms.

- For the problem of predicting a sequence using expert advice, we saw two algorithms: a deterministic algorithm which satisfies a mistake bound

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

  and a randomized algorithm which satisfies a mistake bound

$$M \leq \left(\frac{1}{1-\varepsilon}\right) m + \left(\frac{1}{\varepsilon}\right) \ln n,$$

  where n is the number of experts, m is the number of mistakes made by the best expert, and $\varepsilon > 0$ is a parameter which is pre-configured by the algorithm designer.

- You should think of $1/(1-\varepsilon)$ as being equivalent to $1 + O(\varepsilon)$. (For example, when $\varepsilon < 1/2$ we have $1/(1-\varepsilon) < 1 + 2\varepsilon$.) Hence when $m \gg \ln(n)$, the randomized prediction algorithm comes *very* close to making the same number of mistakes as the best expert.

- Neither algorithm needs to know the value of m in order to achieve these mistake bounds.

- The best randomized algorithm makes half as many mistakes as the best deterministic algorithm. This seems to be a recurring theme in sequential prediction problems. (You saw another example of it on your homework.) The factor of 2 is not because it's a binary prediction problem; randomization also saves a factor of 2 when predicting the data in a k-ary sequence for $k > 2$.

- The randomized prediction algorithm actually applies to a more general problem — the "best expert" problem — in which there is a cost associated with each expert in each time step, and the cost of every expert (a real number between 0 and 1) is revealed only after the algorithm picks one expert.

- The analysis of both algorithms follows the same rough outline. It's important to remember this rough outline: every time the algorithm accrues one unit of cost, there is a corresponding multiplicative decrease in the total "weight" of the experts. Since the total weight can never sink below the weight of the best expert — and it starts out only n times larger than the best expert's weight — the algorithm can only accrue $O(\log n)$ more units of cost than the best expert.

2 Regret

We've seen a randomized algorithm Hedge for the best-expert problem whose expected cost relative to the best expert x^* satisfies

$$\mathbf{E}[\mathsf{Cost}(\mathsf{Hedge})] \leq (1+2\varepsilon)\mathsf{Cost}(x^*) + \frac{\ln(n)}{\varepsilon}, \tag{1}$$

for an arbitrary constant $\varepsilon \in (0, \frac{1}{2})$. Whenever you see a bound that says that an algorithm computes a solution whose cost is within a $1+2\varepsilon$ factor of the optimum, for an arbitrarily small $\varepsilon > 0$, a natural follow-up question is whether one can actually get the cost to be $1 + o(1)$ times the optimum, and if so how small can we make the $o(1)$ term?

Let's rewrite (1) as

$$\mathbf{E}[\mathsf{Cost}(\mathsf{Hedge}) - \mathsf{Cost}(x^*)] \leq 2\varepsilon\,\mathsf{Cost}(x^*) + \frac{\ln(n)}{\varepsilon}. \tag{2}$$

Definition 1 (Informal definition of regret.). The *regret* of an online learning algorithm ALG is the maximum (over all input instances) of the expected difference in cost between the algorithm's choices and the best choice.

Definition 2 (Formal definition of regret, valid for this lecture.). The *regret* of an online learning algorithm ALG relative to a class of adversaries $\mathcal{G}$ is

$$R(\mathsf{ALG}, \mathsf{ADV}) = \sup_{G \in \mathcal{G}} \mathbf{E}\left[\max_{x \in [n]} \sum_{t=1}^{\infty} c_t(x_t) - c_t(x)\right],$$

where c_t denotes the (random) cost function chosen at time t by adversary G playing against algorithm ALG (i.e. the component i_{t+1} in the transcript $\mathsf{Trans}(\mathsf{ALG}, G)$), and x_t denotes the expert chosen at time t by ALG playing against G (i.e. the component o_t in the transcript $\mathsf{Trans}(\mathsf{ALG}, G)$).

Inequality (2) gives a useful upper bound on regret in cases where it is possible to bound, *a priori*, the cost of the best expert x^*. Two such cases are the following.

Finite time horizon Say that an adversary G has *time horizon* T if $c_t(x) = 0$ for all $t > T$ and $x \in [n]$, regardless of the choices made by the algorithm. Denote the set of all such adversaries by $\mathcal{G}[T]$. For an adversary in $\mathcal{G}[T]$, $\mathsf{Cost}(x^*) \leq T$.

Geometric discounting Say that an adversary G has *discount factor* $\delta = 1 - r$ if r is a positive constant and there is another adversary $hatG$ — with cost functions $\hat{c}_t$ taking values between 0 and 1 — such that $c_t = \delta^t \hat{c}_t$. Denote the set of all such adversaries by $\mathcal{G}\langle 1 - r\rangle$. For an adversary in $\mathcal{G}\langle 1 - r\rangle$, $\mathsf{Cost}(x^*) \leq 1/r$.

By choosing $\varepsilon = \sqrt{\ln(n)/2T}$ in the case of a finite time horizon T, or $\varepsilon = \sqrt{r\ln(n)/2}$ in the case of discount factor $1 - r$, we obtain upper bounds on the regret of Hedge against the adversary sets $\mathcal{G}[T]$ and $\mathcal{G}\langle 1 - r\rangle$.

$$R\left(\mathsf{Hedge}\left(\sqrt{\ln(n)/2T}\right), \mathcal{G}[T]\right) \leq 2\sqrt{2T\ln(n)} \tag{3}$$

$$R\left(\mathsf{Hedge}\left(\sqrt{r\ln(n)/2}\right), \mathcal{G}\langle 1 - r\rangle\right) \leq 2\sqrt{2\ln(n)/r} \tag{4}$$

2.1 The doubling trick

To achieve the regret bound (3), the algorithm designer must know the time horizon T in advance, to specify the appropriate value of ε when the algorithm is initialized. We can avoid this assumption that T is known in advance, at the expense of a constant factor, using the *doubling trick*. Whenever we reach a time step t such that t is a power of 2, we restart the algorithm (forgetting all of the information gained in the past) setting ε to $\sqrt{\ln(n)/2t}$. Let us denote this algorithm by Hedge^*. If $2^k \leq T < 2^{k+1}$, the algorithm satisfies the following upper bound on its regret:

$$\begin{aligned}
R(\mathsf{Hedge}^*, \mathcal{G}[T]) &= \sup_{G\in\mathcal{G}[T]} \mathbf{E}\left[\sup_{x\in[n]} \sum_{t=1}^{T} c_t(x_t) - c_t(x)\right] \\
&= \sup_{G\in\mathcal{G}[T]} \mathbf{E}\left[\sup_{x\in[n]} \sum_{j=0}^{k} \sum_{t=2^j}^{2^{j+1}-1} c_t(x_t) - c_t(x)\right] \\
&\leq \sum_{j=0}^{k} \sup_{G\in\mathcal{G}[2^j]} \mathbf{E}\left[\sum_{x\in[n]} \sum_{t=1}^{2^j} c_t(x_t) - c_t(x)\right] \\
&= \sum_{j=0}^{k} R\left(\mathsf{Hedge}\left(\sqrt{ln(n)/2^{j+1}}\right), \mathcal{G}[2^j]\right) \\
&\leq \sum_{j=0}^{k} 2\sqrt{2^{j+1}\ln(n)} \\
&< 2\sqrt{2^{k+1}\ln(n)} \sum_{i=0}^{\infty} 2^{-i/2} \\
&< 7\sqrt{T\ln(n)}.
\end{aligned}$$

We can use this doubling trick whenever we have an algorithm with known time horizon T, whose regret is $O(T^\alpha)$ for some $\alpha > 0$, to obtain another algorithm with unknown time horizon, whose regret is $O(T^\alpha)$ for all time horizons T.

2.2 A lower bound for regret

The regret bound (3) is information-theoretically optimal up to a constant factor: a matching lower bound of $\Omega\left(\sqrt{T\ln(n)}\right)$ arises by considering an input in which the costs $\{c_t(x) : 1 \le t \le T, x \in [n]\}$ costitute a set of Tn independent uniformly distributed random samples from $\{0, 1\}$. The central limit theorem tells us that with high probability, there is an expert whose total cost is $\frac{T}{2} - \Omega\left(\sqrt{T\ln(n)}\right)$. On the other hand, it is obvious that any randomized algorithm will have expected cost $T/2$.

3 Normal-form games, mixed strategies, and Nash equilibria

3.1 Definitions

Definition 3. A normal-form game is specified by:

- A set $\mathcal{I}$ of *players*.
- For each player $i \in \mathcal{I}$, a set A_i of *strategies*.
- For each player $i \in \mathcal{I}$, a *payoff function*

$$u_i : \prod_{i \in \mathcal{I}} A_i \to \mathbb{R}.$$

When a normal-form game has two players, we will generally refer to them as the *row player* (player 1) and the *column player* (player 2) and we will write the payoff functions in a matrix whose rows and columns are indexed by elements of A_1 and A_2, respectively. The entry in row r and column c of the matrix is the ordered pair $(u_1(r, c), u_2(r, c))$.

3.2 Examples of two-player normal-form games

Example 1. (Bach or Stravinsky) Two players have to decide whether to go to a concert of Bach or of Stravinsky. One prefers Bach, the other prefers Stravinsky, but both of them prefer going to a concert together over going alone.

	B	S
B	(2,1)	(0,0)
S	(0,0)	(1,2)

Example 2. (Bach and Stravinsky) Two players live in neighboring rooms. Each must decide whether to play their music at low volume or at high volume. Each one would prefer to play their own music at high volume and would prefer their neighbor's music to be played at low volume.

	Q	L
Q	(3,3)	(1,4)
L	(4,1)	(2,2)

This game is a form of the famous *prisoner's dilemma* game. Each player is better off playing "L" no matter what the opponent's strategy is. Yet the outcome (L,L) is worse for both players than the outcome (Q,Q).

Example 3. (Penalty kick) There are two players: striker and goalie. Each must choose whether to go left or right. If both choose the same direction, the goalie wins. If both choose opposite directions, the striker wins.

	L	R
L	(-1,1)	(1,-1)
R	(1,-1)	(-1,1)

This game is sometimes called *matching pennies*.

3.3 Mixed strategies

A *mixed strategy* for a player of a normal-form game is a rule for picking a random strategy from its strategy set. More formally, for a finite set A, let $\mathbf{\Delta}(A)$ denote the set of all probability distributions on A, i.e.

$$\mathbf{\Delta}(A) = \left\{ p : A \to [0,1] \;\middle|\; \sum_{a \in A} p(a) = 1 \right\}.$$

(This definition can be extended to infinite sets using measure theory, but the formalism required to deal with this extension is outside the scope of this course.) The elements of $\mathbf{\Delta}(A_i)$ are called *mixed strategies* of player i.

Elements of $\prod_{i \in \mathcal{I}} A_i$ are called *pure strategy profiles*. Elements of $\prod_{i \in \mathcal{I}} \mathbf{\Delta}(A_i)$ are called *mixed strategy profiles*. The payoff function of a game can be extended from pure strategy profiles to mixed strategy profiles by defining the payoff of a mixed strategy profile to be the expected payoff when every player samples their random strategy independently:

$$u_i(p_1, p_2, \ldots, p_{|\mathcal{I}|}) = \sum_{\vec{a}} u_i(\vec{a}) p_1(a_1) p_2(a_2) \ldots p_{|\mathcal{I}|}(a_{|\mathcal{I}|}),$$

where the sum runs over all pure strategy profiles $\vec{a} = (a_1, a_2, \ldots, a_{|\mathcal{I}|})$.

3.4 Nash equilibrium

Let $k = |\mathcal{I}|$. For a strategy profile $\vec{a} = (a_1, a_2, \ldots, a_k)$, and an element $a'_i \in A_i$, we introduce the notation (a'_i, a_{-i}) to denote:

$$(a'_i, a_{-i}) = (a_1, a_2, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_k).$$

In words, (a'_i, a_{-i}) is the strategy profile obtained by changing player i's strategy from a_i to a'_i.

Definition 4 (Nash equilibrium). A mixed strategy profile $\vec{p} = (p_1, p_2, \ldots, p_k)$ is a *mixed Nash equilibrium* (or, simply, Nash equilibrium) if it is the case that for all $i \in \mathcal{I}$ and all $q_i \in \mathbf{\Delta}(A_i)$,

$$u_i(q_i, p_{-i}) \leq u_i(p_i, p_{-i}).$$

If each p_i is a pure strategy (i.e. a mixed strategy which assigns probability 1 to a single element of A_i) then $\vec{p}$ is a *pure Nash equilibrium.*

Example 4. There are two pure Nash equilibria of "Bach or Stravinsky", namely (B,B) and (S,S). There is also another mixed Nash equilibrium in which player 1 chooses B with probability 2/3, S with probability 1/3; and player 2 chooses B with probability 1/3, S with probability 2/3. Interestingly, in the mixed equilibrium the payoff for both players is 2/3, so *both* of the pure equilibria are better for *both* players. (In game-theoretic terms, the mixed Nash equilibrium is *Pareto dominated* by the pure equilibria. If x and y are two outcomes of a game, we say that x Pareto dominates y when at least one player strictly prefers x to y, and no player strictly prefers y to x.) This example illustrates three of the critiques of the Nash equilibrium concept.

1. In situations like this where there are multiple Nash equilibria, we can't predict which equilibrium the players will choose (if any).

2. Moreover, the different equilibria imply different payoffs for the two players, so we can't even predict their payoffs.

3. The contention that players will select a Nash equilibrium of the game seems to rely on circular reasoning. Player 1 wants to play its side of the equilibrium strategy profile because it believes that player 2 will play its own side of the same strategy profile, and vice-versa. In situations where there are multiple equilibria, why should we assume that both players will be able to coordinate their beliefs in this way?

These critiques of Nash equilibrium are not the end of the story; they are the beginning of a very interesting story in which game theorists have tried to enhance the theory in various ways to address these critiques and improve the predictive power of game-theoretic equilibrium concepts. In particular, some game theorists have tried

addressing critique (3) using models in which players arrive at an equilibrium by playing the game repeatedly and using learning rules to adapt to their opponent's past behavior. This theory of learning in games will be one of the main topics we address in the coming weeks.

Example 5. The "Bach and Stravinsky" game (i.e. the prisoner's dilemma) has only one Nash equilibrium, the pure strategy profile (L,L). This is a *dominant strategy equilibrium*, meaning that each player is better off playing L than Q, *no matter what the other player does.*

Example 6. The "penalty kick" game has no pure Nash equilibrium. In the unique mixed Nash equilibrium, each player assigns probability 1/2 to both strategies.

4 Two-player zero-sum games and von Neumann's minimax theorem

Definition 5. A *two-player zero-sum game* is one in which $\mathcal{I} = \{1, 2\}$ and $u_2(a_1, a_2) = -u_1(a_1, a_2)$ for all pure strategy profiles (a_1, a_2).

A famous theorem of von Neumann illustrates that the equilibria of two-player zero-sum games are much simpler than the equilibria of general two-player games.

Theorem 1 (von Neumann's minimax theorem). *For every two-player zero-sum game with finite strategy sets A_1, A_2, there is a number $v \in \mathbb{R}$, called the* game value*, such that:*

1. $$v = \max_{p \in \mathbf{\Delta}(A_1)} \min_{q \in \mathbf{\Delta}(A_2)} u_1(p, q) = \min_{q \in \mathbf{\Delta}(A_2)} \max_{p \in \mathbf{\Delta}(A_1)} u_1(p, q)$$

2. *The set of mixed Nash equilibria is nonempty. A mixed strategy profile (p, q) is a Nash equilibrium if and only if*
$$\begin{aligned} p &\in \arg\max_p \min_q u_1(p, q) \\ q &\in \arg\min_q \max_p u_1(p, q) \end{aligned}$$

3. *For all mixed Nash equilibria (p, q), $u_1(p, q) = v$.*

Remark 1. Among other things, the theorem implies that two-player zero-sum games don't suffer from critiques (2) and (3) discussed above. Although there can be multiple equilibria, part (3) of the theorem says that all equilibria result in the same payoffs for both players. Moreover, the players don't need to coordinate with each other in order to play an equilibrium: by part (2) it is sufficient for each of them to choose a mixed strategy in their respective arg max or arg min set, without knowing which equilibrium mixed strategy the opponent is going to choose.

4.1 Some extra observations about Hedge

Before giving the proof of Theorem 1 we must point out two simple properties of the Hedge algorithm which were not derived in previous lectures.

4.1.1 Using Hedge for maximization problems.

Although we defined and analyzed Hedge in the context of online cost-minimization problems, it is easy to adapt the algorithm to the setting of online payoff maximization. The simplest way to do this is to just transform payoff functions into cost functions. If $g_t : [n] \to [0,1]$ is the payoff function at time t, then define $c_t(x) = 1 - g_t(x)$ and use Hedge to compute a sequence of experts x_t which approximately minimize $\sum_{t=1}^{\infty} c_t(x_t)$, which is the same as approximately maximizing $\sum_{t=1}^{\infty} g_t(x_t)$. This suffices for the purpose of bounding the additive regret when using Hedge for maximization problems, since the additive difference between Hedge and the best expert is unaffected by the transformation $c_t(x) = 1 - g_t(x)$. (In other words, if x^* is the best expert, then $c_t(x_t) - c_t(x^*)$ is equal to $g_t(x^*) - g_t(x_t)$; summing over t we find that the algorithm's regret is the same in both the maximization and minimization contexts.)

For future reference, Appendix A presents a slightly different version of Hedge, denoted by MaxHedge, which is suitable for maximization problems. In the appendix we prove the following multiplicative bound which is analogous to Theorem 3 from last week's notes.

Theorem 2. *For every randomized adaptive adversary, for every $T > 0$, the expected payoff gained by* MaxHedge(ε) *satisfies*

$$\mathbf{E}\left[\sum_{t=1}^{T} g_t(x_t)\right] > (1-\varepsilon)\,\mathbf{E}\left[\max_{x\in[n]} \sum_{t=1}^{T} g_t(x)\right] - \left(\frac{1}{\varepsilon}\right)\ln(n). \tag{5}$$

Corollary 3. *For every $T > 0$, if $\varepsilon = \sqrt{\ln(n)/T}$, the expected payoff gained by* MaxHedge(ε) *against any adaptive adversary satisfies*

$$\mathbf{E}\left[\sum_{t=1}^{T} g_t(x_t)\right] > \mathbf{E}\left[\max_{x\in[n]} \sum_{t=1}^{T} g_t(x)\right] - 2\sqrt{T\ln(n)}. \tag{6}$$

In the rest of these notes, we will not distinguish between the algorithms Hedge and MaxHedge.

4.1.2 Hedge tracks the best mixture of experts

For the best-expert problem, we can extend each payoff function g_t from a function on $[n]$ to a function on $\mathbf{\Delta}([0,1])$ by averaging:

$$g_t(p) = \sum_{x\in[n]} p(x) g_t(x).$$

Observe that for any $p \in \mathbf{\Delta}([0,1])$,

$$\sum_{t=1}^{T} g_t(p) \leq \max_{x \in [n]} \sum_{t=1}^{T} g_t(x),$$

since the left side is a weighted average of the values of $\sum_{t=1}^{T} g_t(x)$ as x runs over all elements of $[n]$. Using this fact, we see that the bounds (5) and (6) extend to distributions:

$$\mathbf{E}\left[\sum_{t=1}^{T} g_t(x_t)\right] > (1-\varepsilon)\,\mathbf{E}\left[\max_{p \in \mathbf{\Delta}([n])} \sum_{t=1}^{T} g_t(p)\right] - \left(\frac{1}{\varepsilon}\right)\ln(n). \tag{7}$$

$$\mathbf{E}\left[\sum_{t=1}^{T} g_t(x_t)\right] > \mathbf{E}\left[\max_{p \in \mathbf{\Delta}([n])} \sum_{t=1}^{T} g_t(p)\right] - 2\sqrt{T\ln(n)}. \tag{8}$$

4.2 The main lemma

The hardest step in the proof of von Neumann's minimax theorem is to prove that

$$\max_p \min_q u_1(p,q) \geq \min_q \max_p u_1(p,q).$$

We will prove this fact using online learning algorithms. The basic idea of the proof is that if the players are allowed to play the game repeatedly, using Hedge to adapt to the other player's moves, then low-regret property of Hedge guarantees that the time-average of each player's mixed strategy is nearly a best response to the time-average of the other player's mixed strategy.

Lemma 4. *For any two-player zero-sum game,*

$$\max_{p \in \mathbf{\Delta}(A_1)} \min_{q \in \mathbf{\Delta}(A_2)} u_1(p,q) \geq \min_{q \in \mathbf{\Delta}(A_2)} \max_{p \in \mathbf{\Delta}(A_1)} u_1(p,q).$$

Proof. Without loss of generality, assume that $0 \leq u_1(a_1,a_2) \leq 1$ for all $a_1 \in A_1, a_2 \in A_2$. (If the original payoff function u_1 doesn't satisfy these bounds, we can replace u_1 with the function $bu_1 + c$ for suitable constants $b, c > 0$.)

Let $n = \max\{|A_1|, |A_2|\}$ and for any positive $\delta > 0$ let

$$\begin{aligned} T &= \lceil 4\ln(n)/\delta^2 \rceil \\ \varepsilon &= \sqrt{\ln(n)/T}. \end{aligned}$$

Suppose that each player uses Hedge(ε) (with "experts" corresponding to elements of the player's strategy set) to define a T-step sequence of mixed strategies in response to the other player's sequence of mixed strategies. More precisely, player 1 defines a sequence of mixed strategies $p_1, p_2, \ldots, p_T$ and player 2 defines a sequence of mixed

strategies $q_1, q_2, \ldots, q_T$, according to the following prescription. Player 1 runs the payoff-maximizaton version of $\mathsf{Hedge}(\varepsilon)$, defining the payoff function at time t by $g_t(x) = u_1(x, q_t)$. The mixed strategy p_t is taken to be the distribution from which the algorithm samples at time t, i.e. $p_t(x) = w_{xt} \Big/ \left(\sum_{y \in A_1} w_{yt}\right)$, where w_{xt} is the weight which $\mathsf{Hedge}(\varepsilon)$ assigns to strategy x at time t. Similarly, player 2 runs the payoff-maximization version of $\mathsf{Hedge}(\varepsilon)$, defining the payoff function at time t by $g_t(x) = 1 - u_1(p_t, x)$, and q_t is taken to be the distribution from which the algorithm samples at time t.

Our choice of T and ε guarantees that each player's regret is at most δT, using Corollary 3. Hence we have

$$\min_q \frac{1}{T}\sum_{t=1}^{T} u_1(p_t, q) + \delta \;\;\geq\;\; \frac{1}{T}\sum_{t=1}^{T} u_1(p_t, q_t) \;\;\geq\;\; \max_p \frac{1}{T}\sum_{t=1}^{T} u_1(p, q_t) - \delta \tag{9}$$

where the first inequality follows from considering Player 2's regret, and the second inequality follows from considering Player 1's regret. Letting

$$\overline{p} = \frac{1}{T}\sum_{t=1}^{T} p_t, \qquad \overline{q} = \frac{1}{T}\sum_{t=1}^{T} q_t,$$

we can rewrite (9) as

$$\min_q u_1(\overline{p}, q) + \delta \;\;\geq\;\; \max_p u_1(p, \overline{q}) - \delta. \tag{10}$$

Trivially, we have

$$\max_p \min_q u_1(p, q) + \delta \;\;\geq\;\; \min_q u_1(\overline{p}, q) + \delta \tag{11}$$

$$\max_p u_1(p, \overline{q}) - \delta \;\;\geq\;\; \min_q \max_p u_1(p, q) - \delta \tag{12}$$

and combining (10)-(12) we find that

$$\max_p \min_q u_1(p, q) + \delta \geq \min_q \max_p u_1(p, q) - \delta. \tag{13}$$

The lemma follows because δ can be made arbitrarily close to zero. □

It will be useful to considering the following alternate proof of Lemma 4, which is almost identical to the first proof.

Alternate proof of Lemma 4. As before, assume without loss of generality that $0 \leq u_1(a_1, a_2) \leq 1$ for all strategy profiles (a_1, a_2). Let $\delta > 0$ be an arbitrarily small positive number, and define n, T, ε as above. Player 1 still uses $\mathsf{Hedge}(\varepsilon)$ to define a sequence of mixed strategies $p_1, p_2, \ldots, p_T$ in response to the payoff function induced

by the opponent's sequence of strategies. But player 2 now chooses its strategies adversarially, according to the prescription

$$q_t \in \arg\min_q u_1(p_t, q). \tag{14}$$

Note that the set of mixed strategies minimizing player 1's payoff always contains a pure strategy, so we may assume q_t is a pure strategy if desired.

Define $\overline{p}, \overline{q}$ as above. We find that

$$\begin{aligned}
\max_p \min_q u_1(p,q) &\geq \min_q u_1(\overline{p}, q) \\
&= \min_q \frac{1}{T}\sum_{t=1}^{T} u_1(p_t, q) \\
&\geq \frac{1}{T}\sum_{t=1}^{T} \min_q u_1(p_t, q) \\
&= \frac{1}{T}\sum_{t=1}^{T} u_1(p_t, q_t) \\
&\geq \max_p \frac{1}{T}\sum_{t=1}^{T} u_1(p, q_t) - \delta \\
&= \max_p u_1(p, \overline{q}) - \delta \\
&\geq \min_q \max_p u_1(p,q) - \delta.
\end{aligned}$$

□

4.3 Proof of Theorem 1

In this section we complete the proof of von Neumann's minimax theorem.

Proof of Theorem 1. For any mixed strategy profile $(\hat{p}, \hat{q})$ we have

$$u_1(\hat{p}, \hat{q}) \leq \max_p u_1(p, \hat{q}).$$

Taking the minimum of both sides as $\hat{q}$ ranges over $\mathbf{\Delta}(A_2)$ we find that

$$\min_q u_1(\hat{p}, q) \leq \min_q \max_p u_1(p,q).$$

Taking the maximum of both sides as $\hat{p}$ ranges over $\mathbf{\Delta}(A_1)$ we find that

$$\max_p \min_q u_1(p,q) \leq \min_q \max_p u_1(p,q).$$

The reverse inequality was proven in Lemma 4. Thus we have established part (1) of Theorem 1.

Note that the sets $B_1 = \arg\max_p \min_q u_1(p,q)$ and $B_2 = \arg\min_q \max_p u_1(p,q)$ are both nonempty. (This follows from the compactness of $\mathbf{\Delta}(A_1)$ and $\mathbf{\Delta}(A_2)$, the contintuity of u_1, and the finiteness of A_1 and A_2.) If $p \in B_1$ and $q \in B_2$ then

$$v = \min_q u_1(p,q) \leq u_1(p,q) \leq \max_p u_1(p,q) = v$$

hence $u_1(p,q) = v$. Moreover, since $q \in B_2$, player 1 can't achieve a payoff greater than v against q by changing its mixed strategy. Similarly, since $p \in B_1$, player 2 can't force player 1's payoff to be less than v by changing its own mixed strategy. Hence (p,q) is a Nash equilibrium. Conversely, if (p,q) is a Nash equilibrium, then

$$u_1(p,q) = \max_p u_1(p,q) \geq v \tag{15}$$

$$u_1(p,q) = \min_q u_1(p,q) \leq v \tag{16}$$

and this implies that in each of (15), (16), the inequality on the right side is actually an *equality*, which in turn implies that $p \in B_1$ and $q \in B_2$. This completes the proof of (2) and (3). □

The proof of the minimax theorem given here, using online learning, differs from the standard proof which uses ideas from the theory of linear programming. The learning-theoretic proof has a few advantages, some of which are spelled out in the following remarks.

Remark 2. The procedure of using Hedge to approximately solve a zero-sum game is remarkably fast: it converges to within δ of the optimum using only $O(\log(n)/\delta^2)$ steps, provided the payoffs are between 0 and 1. (By "converges to within δ", we mean that it outputs a pair of mixed strategies, $(\overline{p}, \overline{q})$ such that $\min_q u_1(\overline{p}, q) \geq v - \delta$ and $\max_p u_1(p, \overline{q}) \leq v + \delta$, where v is the game value.) This is especially important when one of the players has a strategy set whose size is exponentially larger than the size of the natural representation of the game. See Example 7 below for an example of this.

Recall that in the second proof of Lemma 4 we remarked that player 2's strategies q_t could be taken to be pure strategies. Note also that the Hedge algorithm used by player 1 only needs to look at the scores in a column of the payoff matrix if the corresponding strategy has been used by player 2 some time in the past. Thus, as long as we have an oracle for finding player 2's best response to any mixed strategy, we need only look at a very sparse subset of the payoff matrix — a set of $O(\log(n)/\delta^2)$ columns — to compute a mixed strategy for player 1 which obtains an additive δ-approximation to the game value. Again, see Example 7 for an example in which it is reasonable to assume that we don't have an explicit representation of the payoff matrix, but we can examine any desired column and we have an oracle for finding player 2's best response to any mixed strategy.

Example 7 (The VPN eavesdropping game). Let $G = (V, E)$ be an undirected graph. In the "VPN eavesdropping game", player 1 chooses an edge of G, and player 2 chooses a spanning tree of G. For any edge e and spanning tree T, the payoff of player 1 is

$$u_1(e, T) = \begin{cases} 1 & \text{if } e \in T \\ 0 & \text{otherwise.} \end{cases}$$

(We can think of player 1 as an eavesdropper who can listen on any single edge of G, and player 2 as someone who is setting up a virtual private network on the edges of T, to join together all the nodes of G. The game is a win for player 1 if he or she eavesdrops on an edge which is part of the VPN.)

Note that, in general, the cardinality of player 2's strategy set is exponential in the size of G. Thus the parameter n appearing in the proof of Lemma 4 will be exponential in the size of the game's natural representation. However, it is easy to examine any particular column of the payoff matrix u_1: the column corresponding to a spanning tree T will be a vector of 0's and 1's, with 1's in the rows corresponding to the edges of T. Moreover, it is easy to compute player 2's best response to any mixed strategy of player 1: one simply computes a minimum spanning tree of G, where the weight of each edge is equal to the probability of player 1 picking that edge.

Consequently, there is an algorithm for approximately solving the game (up to an additive error of δ) which requires only $O(\log(M)/\delta^2)$ minimum spanning tree computations, where M is the total number of minimum spanning trees of G. (If G has V vertices, then by Cayley's formula $M \leq V^{V-2}$. Hence $\log(M)$ is always polynomial — in fact, nearly linear — in the number of vertices of G.)

Remark 3. Another consequence of the second proof of Lemma 4 is that player 2 has a mixed strategy which has *sparse support* — i.e. at most $O(\log(n)/\delta^2)$ strategies have positive probability — yet it achieves an additive δ-approximation to the game value. By symmetry, player 1 also has a mixed strategy with sparse support which achieves an additive δ-approximation to the game value. Hence the game has an approximate Nash equilibrium in which both players use sparsely-supported mixed strategies.

Remark 4. If player 2 is not playing rationally, by using Hedge player 1 comes close to achieving the best possible payoff against whatever distribution of strategies player 2 happens to be using. This property would not be ensured in repeated play if player 1 instead solved the game offline, picked a strategy in $\arg\max_p \min_q u_1(p, q)$, and always used this strategy.

Remark 5. If we think about our intuition of how human beings learn to play games against each other, the process is probably more similar to a learning algorithm such as Hedge than to a linear programming algorithm such as the simplex method. Hence another benefit of the learning-theoretic proof is that it gives an intuitive justification for why human beings are able to find the equilibria of zero-sum games.

4.4 Yao's lemma

The von Neumann minimax theorem has an important consequence in computer science. Suppose we have a computational problem with a finite set of possible inputs I, and we are considering a finite set of possible algorithms A. For example, I might be the set of all n-bit binary strings, and A might be the set of all Boolean circuits of size at most n^3 which accept an n-bit input and return a valid output for the problem under consideration. Suppose we have a parameter $t(i, a)$ which corresponds to the cost of running algorithm a on input i. For example, $t(i, a)$ could denote the algorithm's running time, or the cost of the solution it computes.

We may interpret this scenario as a two-player zero-sum game in which player 1 specifies an input, player 2 specifies an algorithm, and $t(i, a)$ is the payoff for player 1. Let $\mathcal{D} = \mathbf{\Delta}(I)$ denote the set of all probability distributions on inputs, and let $\mathcal{R} = \mathbf{\Delta}(A)$ denote the set of all probability distributions on algorithms, i.e. the set of all randomized algorithms. We can extend the function t to mixed strategy profiles in the usual way, i.e.

$$t(d, r) = \sum_{i \in I} \sum_{a \in A} t(i, a) d(i) r(a).$$

Lemma 5 (Yao's Lemma).

$$\max_{d \in \mathcal{D}} \min_{a \in A} t(d, a) = \max_{d \in \mathcal{D}} \min_{r \in \mathcal{R}} t(d, r) = \min_{r \in \mathcal{R}} \max_{d \in \mathcal{D}} t(d, r) = \min_{r \in \mathcal{R}} \max_{i \in I} t(i, r).$$

Proof. The second equality is a restatement of von Neumann's minimax theorem. The first and third equalities follow from the fact that for any mixed strategy of one player, the other player always has a best response which is a pure strategy, i.e.

$$\begin{aligned} \forall d \in \mathcal{D} \ \min_{r \in \mathcal{R}} t(d, r) &= \min_{a \in A} t(d, a) \\ \forall r \in \mathcal{R} \ \max_{d \in \mathcal{D}} t(d, r) &= \max_{i \in I} t(i, r). \end{aligned}$$

□

5 Learning equilibria in non-zero sum games

Here we present a short example to illustrate that the dynamics of learning processes in non-zero-sum games can be much more complicated. We will consider a variant of the well-known "rock, paper, scissors" game, with the following payoff matrix.

	R	P	S
R	(-5,-5)	(-1,1)	(1,-1)
P	(1,-1)	(-5,-5)	(-1,1)
S	(-1,1)	(1,-1)	(-5,-5)

Consider the dynamics studied in the second proof of Lemma 4, i.e. player 1 uses Hedge(ε) to choose mixed strategies p_t, and player 2 responds adversarially with a pure strategy $q_t \in \arg\min_q u_1(p_t, q)$. If one looks at the strategies of both players during the course of an infinite time history, the timeline is divided into epochs during which player 2 is always picking the same strategy in {R,P,S}, and player 1 is adjusting its mixed strategy accordingly. The lengths of these epochs are approximated by a geometric progression. For example, in an epoch when player 2 is always picking R, player 1 is increasing the probability of P, decreasing the probability of S, and *rapidly* decreasing the probability of R. At some point when the probability of S is small enough and the probability of P is large enough, player 2 will shift to playing S. Player 2 will continue playing S until player 1 increases the probability of R enough to make P more attractive than S for player 2. However, this shift from S to P takes longer (by a constant factor) than the previous shift from R to S, because player 1 decreased the weight assigned to R very rapidly during the period when player 2 was playing R, and player 1 increased the weight assigned to R much more slowly during the period when player 2 was playing S.

As a consequence of these observations, we see that the average of the strategies chosen by player 2 (the mixed strategy denoted by $\overline{q}$ in the proof of Lemma 4) never converges! Similarly, the average of the strategies chosen by player 1 never converges. So the description of the type of equilibrium achieved by this process (if any) must be significantly more complicated than Nash equilibrium. During the next few weeks we will be discussing equilibrium concepts for non-zero-sum games and analyzing the types of equilibria which arise as the limiting outcomes of different learning processes.

A Appendix: The MaxHedge algorithm

The algorithm MaxHedge(ε) — a version of Hedge suited for payoff maximization rather than cost minimization — is presented in Figure 1. In this section we analyze the algorithm, proving Theorem 2 and Corollary 3.

Lemma 6. *For $x > 0$,*

$$\frac{1}{x}\ln(1+x) > 1 - x. \tag{17}$$

Proof. We have

$$\ln\left(\frac{1}{1+x}\right) = \ln\left(1 - \frac{x}{1+x}\right) < -\left(\frac{x}{1+x}\right).$$

Multiplying both sides by $-1/x$,

$$\frac{1}{x}\ln(1+x) > \frac{1}{1+x}.$$

Finally, the inequality $1 > (1-x)(1+x)$ implies that $\frac{1}{1+x} > 1 - x$, which concludes the proof of the lemma. □

Algorithm MaxHedge(ε)

/* Initialization */
$w_x \leftarrow 1$ **for** $x \in [n]$

/* Main loop */
for $t = 1, 2, \ldots$
  /* Define distribution for sampling random strategy */
  for $x \in [n]$
    $p_t(x) \leftarrow w_x \Big/ \left(\sum_{y=1}^{n} w_y\right)$
  end
  Choose $x_t \in [n]$ at random according to distribution p_t.
  Observe payoff function g_t.

  /* Update score for each strategy */
  for $x \in [n]$
    $w_x \leftarrow w_x \cdot (1+\varepsilon)^{g_t(x)}$
  end
end

Figure 1: The algorithm MaxHedge(ε).

Proof of Theorem 2. Let w_{xt} denote the value of w_x at the beginning of the t-th iteration of the main loop, and let $W_t = \sum_{x=1}^{n} w_{xt}$. Note that w_{xt}, W_t are random variables, since they depend on the adversary's choices which in turn depend on the algorithm's random choices in previous steps. For an expert $x \in [n]$, let $g_{1..T}(x)$ denote the total payoff

$$g_{1..T}(x) = \sum_{t=1}^{T} g_t(x).$$

Let $x^* = \arg\max_{x \in [n]} g_{1..T}(x)$. We have

$$W_T > w_{x^* t} = (1+\varepsilon)^{g_{1..T}(x^*)}$$

and after taking logarithms of both sides this becomes

$$\ln(W_T) > \ln(1+\varepsilon) g_{1..T}(x^*) \tag{18}$$

On the other hand, we can bound the expected value of $\ln(W_T)$ from above, using an inductive argument. Let w_{*t} denote the vector of weights $(w_{1t}, \ldots, w_{nt})$.

$$\mathbf{E}(W_{t+1} \,|\, w_{*t}) \;=\; \sum_{x=1}^{n} \mathbf{E}\left((1+\varepsilon)^{g_t(x)} w_{xt} \,|\, w_{*t}\right) \tag{19}$$

$$
\begin{aligned}
&\leq \sum_{x=1}^{n} \mathbf{E}\left((1+\varepsilon g_t(x))w_{xt} \mid w_{*t}\right) && (20)\\
&= \sum_{x=1}^{n} w_{xt} + \varepsilon \mathbf{E}\left(\sum_{x=1}^{n} g_t(x) w_{xt} \mid w_{*t}\right) && (21)\\
&= W_t \cdot \left(1 + \varepsilon \mathbf{E}\left(\sum_{x=1}^{n} g_t(x) p_t(x) \mid w_{*t}\right)\right) && (22)\\
&= W_t \cdot (1 + \varepsilon \mathbf{E}(g_t(x_t) \mid w_{*t})) && (23)\\
\mathbf{E}(\ln(W_{t+1}) \mid w_{*t}) &\leq \ln(W_t) + \ln(1 + \varepsilon \mathbf{E}(g_t(x_t) \mid w_{*t})) && (24)\\
&\leq \ln(W_t) + \varepsilon \mathbf{E}(g_t(x_t) \mid w_{*t}) && (25)\\
\mathbf{E}(\ln(W_{t+1}) \mid w_{*t}) - \ln(W_t) &\leq \varepsilon \mathbf{E}(g_t(x_t) \mid w_{*t}) && (26)\\
\mathbf{E}(\ln(W_{t+1})) - \mathbf{E}(\ln(W_t)) &\leq \varepsilon \mathbf{E}(g_t(x_t)) && (27)\\
\mathbf{E}(\ln(W_T)) - \ln(n) &\leq \varepsilon \mathbf{E}\left(\sum_{t=1}^{T} g_t(x_t)\right) && (28)
\end{aligned}
$$

Here, (20) is derived using the identity $(1+\varepsilon)^x \leq 1+\varepsilon x$, which is valid for $\varepsilon > 0$ and $0 \leq x \leq 1$. Step (22) is derived using the fact that $p_t(x) = w_{xt}/W_t$, (23) is derived using the observation that x_t is a random element sampled from the probability distribution $p_t(\cdot)$ on $[n]$, (24) is derived using Jensen's inequality, (27) is derived by taking the unconditional expectation of both sides of the inequality, and (28) is derived by summing over t and recalling that $W_0 = n$.

Combining (18) and (28) we obtain

$$
\begin{aligned}
\varepsilon \mathbf{E}\left(\sum_{t=1}^{T} g_t(x_t)\right) &> \ln(1+\varepsilon)\mathbf{E}(g_{1..T}(x^*)) - \ln(n)\\
\mathbf{E}\left(\sum_{t=1}^{T} g_t(x_t)\right) &> \frac{1}{\varepsilon}\ln\left(\frac{1}{1+\varepsilon}\right)\mathbf{E}(g_{1..T}(x^*)) - \left(\frac{1}{\varepsilon}\right)\ln(n)\\
\mathbf{E}\left(\sum_{t=1}^{T} g_t(x_t)\right) &> (1-\varepsilon)\,\mathbf{E}(g_{1..T}(x^*)) - \left(\frac{1}{\varepsilon}\right)\ln(n) \qquad (29)
\end{aligned}
$$

where the last line is derived using identity (17) from the Lemma above. □

Proof of Corollary 3. The corollary follows by combining (29) above with the trivial bound $\mathbf{E}(g_{1..T}(x^*)) \leq T$. □