Minimizing regret in repeated play of games

Uriel Feige

May 12, 2013

1 Introduction

These notes are largely based on a survey on *learning, regret minimization and equilibria* written by Avrim Blum and Yishay Mansour, and appearing as Chapter 4 of [NRTV07]. The main difference are in the presentation rather than in the content, and in the inclusion of Section 1.5. The notes correspond to two 2-hour lectures in the course on algorithmic game theory given in the Weizmann Institute in May 2013.

Suppose player P is faced with a game for which P knows his own payoff matrix, but does not know the payoff matrix of the other players (or equivalently, knows payoff matrices for the other players, but has no reason to assume that other players are rational). In this general setting, one may represent (essentially without loss of generality) all remaining players as one player Q, and hence the game may be assumed to be a 2-player game. How should P play? If P has a dominant strategy, then playing it is a natural policy. But in the absence of dominant strategies, solution concepts such as Nash equilibrium involve the payoffs of both players, and hence P would not know which strategy corresponds to such a solution concept. Moreover, it might be the case that Q is not willing to play some strategies available to Q (e.g., they might be dominated by other strategies), so it could be that the payoff matrix for P (say, as a row player) contains columns that are not really part of the game. At this full generality of this situation, it is difficult to provide good advice for P what to play. The content of these notes is a variation on the above situation, in which some basic game G (for which P knows only his own payoff matrix) is repeated for T rounds, and P wishes to maximizes the sum of expected payoffs from all rounds. In this situation, P does have hope of playing intelligently, because at any given round P knows what Q played in previous rounds, and P can use this information in deciding what to play in the given round.

1.1 A formal model

There is a two player game G that is repeated for T rounds. In G, the row player P has a set of N actions (strategies) and the column player Q has an arbitrary number of actions. The payoff matrices of P and Q might contain arbitrarily many columns, and hence neither matrix is given to P. Instead, whenever Q performs an action, the corresponding column of P's payoff matrix is revealed to P. (The fact that P need not know his payoff matrix in advance makes the results applicable in a wider range of settings.) The multi-round game proceeds as follows. At a given round, first P chooses a probability distribution p over the rows, then Q chooses a probability distribution over the columns, then a row and a column are selected according to these distributions, P collects his payoff for the round and learns the entire column played by Q (what P's payoff would have been for any other row). P's goal is to play a multi-round strategy that maximizes the expected sum of payoffs collected from all rounds.

It turns out that even the task of approximately maximizing P's payoff is hopeless. Consider a game in which the payoff for P completely depends on Q's action (what P plays is irrelevant), and the payoff for Q is identically 0. Let Q declare his multi-round strategy in advance: Q plays strategy 1 in round 1, and in all future rounds Q plays the same strategy that P played on his first round. Hence the payoff for P is completely determined by P's action in the first round. But not knowing his own payoff matrix, P has no idea what to play in the first round.

To overcome the above difficulty, we change our goal in the above setting. Instead of maximizing expected payoff for P, we wish to minimize the *regret* of P, compared to some *benchmark*. Let us provide more details. First, without loss of generality, we assume that all payoffs in the payoff matrix of P are scaled to be in the range $[0, 1]$. Given an execution of the T-round game, the *regret* for P is the difference between the maximum payoff offered by the benchmark and that actually obtained by P. We consider the following benchmark:

External regret. The benchmark is that of fixed action strategies against a nonadaptive player Q. Namely, for each of the N actions of P one computes the sum of payoffs that would have been obtained had this action been played in all rounds, assuming that Q would play in each round the same action played on that round in the given execution. The maximum of these N values is taken as the benchmark.

The benchmark we compare against makes the assumption that Q's play is unaffected by P's play. However, as we shall see, the results we obtain relative to this benchmark are informative even in (some) situations in which Q's play is affected by P's play.

Observe that on a particular execution the external regret may be negative (the player does better than the benchmark).

Our goal is to design algorithms that achieve external regret that is sublinear in T. This would imply that the average regret per round tends to 0 as T grows.

As a convention, the term regret will mean external regret.

1.2 Some probabilistic preliminaries

To achieve sub-linear regret, the strategy for P might need to be randomized. This can easily be seen by taking the basic game to be matching pennies. Hence we shall consider randomized strategies for P, and compute expected regret. The matching pennies game shows that even in the case $N = 2$ and 0/1 payoffs, the expected regret is at least $\Omega(\sqrt{T})$. This can be proved as follows. Suppose that Q is playing uniformly at random. Then the expected payoff for P is exactly $T/2$. On the other hand, the expected payoff for the external regret benchmark if $T/2 + \Omega(\sqrt{T})$, because Q has constant probability of playing one of his two actions at least $\Omega(\sqrt{T})$ times more often that the other action (the standard deviation of T coin flips is $\Theta(\sqrt{T})$).

One can similarly show games for which the expected regret is at least $\Omega(\log N)$ (even for $T = O(\log N)$). Consider a game with 0/1 payoffs that proceeds for $T = \log_2 N$ rounds and in which the payoff matrix for P contains the 2^N strings $\{0, 1\}^N$ as columns. If Q plays completely at random in each round, then regardless of what P plays, the expected payoff for P is exactly $\frac{\log N}{2}$. However, there is likely to be one action of P that would achieve a payoff of 1 in every round (each action has probability $1/N$ of achieving this, independently of other actions). It is not hard to show that the expected value of the benchmark is larger than $\log N - 1$, giving $\Omega(\log N)$ expected regret.

More generally, if there are N actions each giving random 0/1 payoffs, adaptation of the arguments above show that every algorithm will have expected regret $\Theta(\sqrt{T \log N})$. The examples above do not require Q to be aware of what P is playing.

A simplifying trick in the analysis of randomized strategies is to replace them by deterministic fractional strategies, where the payoff collected from an action in a given round is the probability with which it is played (its fractional value) times its payoff. The expected payoff in fractional and probabilistic strategies is exactly the same, and the value of the external regret benchmark is not affected by this change.

1.3 The randomized weighted majority algorithm

This is an algorithm suggested by Littlestone and Warmouth, and variations of it are used in many different contexts. We present here the special case in which payoffs are either 0 or 1. The algorithm involves a parameter $0 < \epsilon < 1$.

Initially, each action gets weight 1. Let $w^t(i)$ denote the weight of action i prior to round t. Hence $w^1(i) = 1$ for all i. In round t, action i is played with probability $\frac{w^t(i)}{\sum_j w^t(j)}$. Then, the weight of those actions that give 0-payoff at round t remains unchanged (namely, $w^{t+1}(j) = w^t(j)$), whereas the weight of those actions that give 1-payoff at that round is multiplied by $(1+\epsilon)$ (namely, $w^{t+1}(j) = (1+\epsilon)w^t(j)$).

We now analyze the regret of the RWM algorithm. For simplicity of the analysis, we analyze the fractional version of RWM (that we call FWM). Set $W^t = \sum_{i=1}^N w^t(i)$, let f_t denote the payoff obtained by FWM in round t, let $f = \sum_{t=1}^T f_t$ be the total payoff obtained by FWM, and let b denote the total payoff obtained by the benchmark.

Proposition 1.1 $W^{T+1} = N \prod_{t=1}^T (1 + f_t \epsilon)$.

Proof: Proof by induction on t. The base case is $t = 0$ for which $W^1 = N$. Consider round t. In hindsight, let S be the set of actions that gave payoff 1 in round t. Then on the one hand $f_t = \frac{\sum_{i \in S} w^t(i)}{\sum_{j \in [N]} w^t(j)}$, and on the other hand $W^{t+1} = W^t + \epsilon \sum_{i \in S} w^t(i)$. Hence $\frac{W_{t+1}}{W_t} = 1 + \epsilon f^t$, as required by the inductive step. $\square$

Proposition 1.2 $W^{T+1} \geq (1+\epsilon)^b$.

Proof: Consider the action that maximizes the benchmark. In b distinct rounds its weight was multiplied by a factor of $(1+\epsilon)$. $\square$

Combining the two propositions, we deduce that $N \prod_{t=1}^T (1+f_t\epsilon) \geq (1+\epsilon)^b$. As $1+\delta < e^\delta$ for all $0 < \delta < 1$, the left hand side can be replaced by $Ne^{\epsilon f}$. For $0 < \epsilon < 1/2$ we have that $1 + \epsilon > e^{\epsilon - \epsilon^2}$ (this can easily be verified by the Taylor expansion of e^x). Hence the right hand side is at most $e^{(\epsilon-\epsilon^2)b}$. This gives $Ne^{\epsilon f} > e^{(\epsilon-\epsilon^2)b}$. Taking logarithms on both sides and rearranging we obtain that $f > b - \epsilon b - \frac{\log N}{\epsilon}$. As $b \leq T$ we may choose (in the

RWM algorithm) $\epsilon = \sqrt{\frac{\log N}{T}}$ obtaining an external regret term of $2\sqrt{T \log N}$. We have thus established:

Theorem 1.3 *When payoffs are 0/1, the randomized weighted majority algorithm has expected external regret at most* $2\sqrt{T \log N}$.

Extending Theorem 1.3 to payoffs in $[0, 1]$ is left as homework.

1.4 The minimax theorem revisited

Consider a 2-person 0-sum finite game G. Recall the minimax theorem: if for every mixed strategy of Q player P has a reply (pure strategy) with expected payoff at least p, than P can announce a mixed strategy with respect to which no reply of Q gives P expected payoff smaller than p. We previously proved the minimax theorem using linear programming duality. Here we show that it is also a consequence of sublinear external regret in repeated games.

Consider a T-fold repeated version of G. Suppose that both players use an algorithm of sublinear external regret. Then on average (per round), P gets payoff (denote it by p) at least equal (up to terms that tend to 0 as T grows) as his best response to the empirical mixed strategy of Q. Moreover, Q gets payoff at least equal (up to terms that tend to 0 as T grows) as his best response to the empirical mixed strategy of P. Hence the empirical mixed strategies of P and Q must be minimax strategies. (Both players can announce these mixed strategies in advance and no player has an incentive to deviate.)

The above implies that the RWM algorithm can be used in order to compute mixed strategies certifying the minimax theorem. Observe that the convergence to minimax strategies is not in the sense that towards the last rounds players play their minimax strategies, but rather that the minimax strategies are the long-time averages of what players empirically play.

1.5 Linear programming revisited

Recall that we used linear programming to prove the minimax theorem. Now that we have an alternative proof for the minimax theorem, we revisit linear programming.

As mentioned in the past, linear programs come in many equivalent forms. Consider the following *game form* of a linear program. There is no objective function (the objective is added as a constraint). Variables are nonnegative (this is fairly standard), and in addition we have the constraint $\sum_i x_i = 1$. (This can be enforced for bounded linear programs by scaling and adding a slackness variable.) Every other constraint j is of the form $\sum_i a_{ij} x_i \geq 1$. A constraint of the form $\sum_i a_{ij} x_i \leq b_j$ can be converted to the required from by first negating it, obtaining $\sum_i -a_{ij} x_i \geq -b_j$, and then adding to it the constraint $\sum_i (b_j + 1) x_i = b_j + 1$ (implied by $\sum_i x_i = 1$). Hence the *game form* of a linear program is testing feasibility of:

minimize 0
subject to
$A\mathbf{x} \geq \mathbf{1}$
$\sum_{i=1}^n x_i = 1$
$\mathbf{x} \geq \mathbf{0}$

(**Boldface** notation denotes vectors.)

Consider a game between a variable player and a constraint player. The variable player chooses a variable i, the constraint player chooses a constraint j, and the payoff to the variable player is a_{ij}. If the LP is feasible, a feasible solution gives the variable player a mixed strategy of expected payoff at least 1. For any mixed strategy that corresponds to a nonfeasible solution, the constraint player has a reply (the violated constraint) that gives the variable player expected payoff less than 1.

Consider a repeated game in which the LP is the basic game, and both players play algorithms with low external regret. Then if the LP is feasible the empirical average play of the variable player converges to a feasible solution to the LP. If the LP is infeasible, the empirical average play of the constraint player converges to an unbounded solution of the dual LP as explained now. There are m dual y variables (one for each row of A) and one dual z variable (for the constraint $\sum_{i=1}^{n} x_i = 1$). The dual LP is:

maximize $\sum_j y_j + z$

subject to

$A^T\mathbf{y} + \mathbf{1}z \leq 0$

$\mathbf{y} \geq \mathbf{0}$

If the constraint player has a strategy that keeps the expected payoff to the variable player smaller than $\delta < 1$, then using this strategy as a solution to $\mathbf{y}$ and setting $z = -\delta$ gives a feasible solution to the dual LP of value $1-\delta > 0$. Scaling this solution by arbitrarily large multiplicative positive constants shows that the dual LP is unbounded.

The research direction of using multiplicative update rules in the design of algorithms is surveyed in [AHK12].

1.6 Swap regret

The notion of external regret might appear to be weak in the sense that the benchmark only compares against strategies that are fixed throughout all rounds in the repeated game. However, due to the fact that the regret depends only on the logarithm of the number of actions available, it is easy to extend the benchmark to much larger classes of strategies while still maintaining sublinear regret. For example, if one wants to consider strategies that may switch the fixed action midway through the repeated game, one simply considers a benchmark containing N^2 strategies instead on N, where strategy a_{ij} (for $1 \leq i, j \leq N$) means playing action i in the first $T/2$ rounds and action j in the last $T/2$ rounds. As $\log N^2 = 2\log N$ the regret does not significantly change.

It is desirable in some cases to consider *internal regret* rather than *external regret.* The difference between the two is that in external regret the benchmark is set in advance independently of the actions of the algorithm, whereas in internal regret the benchmark depends on the algorithm.

Swap regret. One considers all N^N mappings from $[N]$ to $[N]$. For each such mapping f, one sums over all rounds the payoff that P would get if the action played at that round would be swapped to the one indicated by applying f, assuming that Q does not change his actions. The maximum of these N^N values is taken as the benchmark.

Recall that on a particular execution the external regret may be negative (the player does better than the benchmark). In contrast, the swap regret is nonnegative (taking f to be the identity function).

As swap regret includes a benchmark of N^N strategies, one may expect to be able to achieve swap regret of $O(\sqrt{T \log N^N}) = O(\sqrt{TN \log N})$. However, swap regret is an internal regret notion rather than an external one, hence we cannot use as a blackbox algorithms with low external regret. Moreover, even if we could, this might give an algorithm of complexity N^N. Blum and Mansour provide an elegant way of circumventing these difficulties.

To simplify the proof that follows, let us introduce a notion of *fractional swap regret.* In each round t, the player P has a probability distribution p^t over actions, and then chooses one action from this probability distribution. His fractional payoff (if actions are played fractionally) is equal to his expected payoff. For fractional internal regret we do not replace the realizations of action i by action j, but rather the probability p_i^t is changed to 0 and the probability p_j^t is raised by p_i^t. In expectation, this does not change the expected payoff of a particular benchmark swap strategy. However, in the swap benchmark one takes the swap strategy of highest payoff. The expectation of the maximum is in general larger than the maximum of the expectation. Hence expected swap regret is in general larger than fractional swap regret. (For example, consider N actions that each give total payoff of $T/2$ in T rounds. Action a_1 gives payoff 1/2 in every round, and actions $a_2, \ldots a_N$ give essentially random 0/1 payoffs. Playing strategy a_1 with probability 1/2 in every round has 0 fractional swap regret, but $\Omega(\sqrt{T \log N})$ expected swap regret.) Nevertheless, the difference between the two can be shown to be $O(N\sqrt{\frac{T}{N}\log N}) = O(\sqrt{TN \log N})$. (Sketch of proof. There are $N(N-1)$ basic swaps of the form "action i is replaced by action j". Large deviation bounds imply that for every action j that replaces i the probability of more than $O(\sqrt{\log N})$ standard deviations from expectation is smaller than $1/N^2$. By a convexity argument, the worst sum over i, j of standard deviations is when $\sum_t p_i^t = T/N$ for all actions i. Then there are N events each with standard deviation $\sqrt{T/N}$.) It follows that a bound of $O(\sqrt{TN \log n})$ on fractional swap regret implies a similar bound of expected swap regret.

Theorem 1.4 *There is a polynomial time sequential optimization algorithm that for* $[0, 1]$ *payoffs achieves fractional swap regret of* $O(\sqrt{TN \log N})$.

Proof: Algorithm A is a coordinator among N algorithms $A_1, \ldots A_N$. Each of the N algorithms is a low regret algorithm (we let R_i denote the regret for algorithm A_i), such as the fractional weighted majority algorithm. In every round t, each algorithm A_i proposes a vector of fractions q_i^t that sums up to 1. The idea is to associate algorithm A_i with action i in two respects that complement each other.

1. A_i updates the weights (fractions) after round t only if action i is played in round t. More precisely, we use a fractional implementation of this idea, in the sense that if the fraction that A assigns to action i is p_i^t, then the payoff vector observed by A_i in round t is $p_i^t L^t$ (where L^t is the payoff vector for round t). The external regret properties of A_i imply:

$$\sum_{t=1}^{T} \sum_{j \in [N]} q_i^t(j) p_i^t L^t(j) \geq \max_j [\sum_{t=1}^{T} p_i^t L^t(j)] - R_i$$

2. In round t, algorithm A receives the vectors of fractions q_i^t from each algorithm A_i and needs to aggregate them into one vector of fractions p^t. This p^t is a weighted average over the q_i^t. Namely $p^t = \sum_{i=1}^N \lambda_i^t q_i^t$. How are the weights λ_i^t chosen? The idea it to choose the vector λ^t such that when solving for p^t one would get $p^t = \lambda^t$. Hence the probability that A takes the advice of A_i is exactly equal to the probability that A chooses to play action i. Proposition 1.5 will show that such a choice of λ^t exists, and moreover, can be found in polynomial time.

The fractional payoff for A is

$$\text{payoff}(A) = \sum_{t=1}^T \sum_{i\in[N]} \lambda_i^t \sum_j q_i^t(j) L^t(j)$$

Consider now a swap function $f : [N] \to [N]$ that for every j plays $f(j)$ instead of j. The fractional payoff for A_f is

$$\text{payoff}(A_f) = \sum_{i\in[N]} \sum_{t=1}^T p_i^t L^t(f(i)) \le \sum_{i\in[N]} \max_j [\sum_{t=1}^T p_i^t L^t(j)]$$

$$\le \sum_{i\in[N]} \left(R_i + \sum_{t=1}^T \sum_{j\in[N]} q_i^t(j) p_i^t L^t(j) \right) = \sum_{i\in[N]} R_i + \sum_{i\in[N]} \sum_{t=1}^T \sum_{j\in[N]} q_i^t(j) \lambda_i^t L^t(j)$$

$$= \text{payoff}(A) + \sum_i R_i$$

Recall that in the fractional weighted majority algorithm the external regret was at most $\epsilon b + \frac{\log N}{\epsilon}$, where b was the value of the benchmark (taken there to be at most T). Hence:

$$\sum_i R_i = \sum_i (\epsilon b_i + \frac{\log N}{\epsilon}) = \epsilon \sum_i b_i + \frac{N \log N}{\epsilon}$$

Observe that every b_i is upper bounded by $\sum_{t=1}^T \lambda_i^t$ and hence $\sum_i b_i \le \sum_{t=1}^T \sum_i \lambda_i^T = T$. Choosing $\epsilon = \sqrt{\frac{\log N}{NT}}$ gives fractional swap regret $2\sqrt{TN \log N}$, as desired. $\square$

It remains to prove the following proposition.

Proposition 1.5 *For $i \in [N]$, given nonnegative vectors $q_i \in R^N$ with $\sum_j q_i(j) = 1$ for every i, one can associate nonnegative weights λ_i summing up to 1 such that $\lambda_i = \sum_{j\in[N]} \lambda_j q_j(i)$ for every i. Moreover, such a vector λ can be found in polynomial time.*

Proof: Consider the N by N matrix Q whose columns are the vectors q_i. Let $X \in R^N$ be the set of nonnegative vectors whose entries sum up to 1. Observe that X is a compact convex set. The mapping Qx is a continuous mapping from X to itself. Hence by Brouwer's fixpoint theorem it has a fixpoint, which can serve as our λ.

The proof of Brouwer's theorem does not provide a polynomial time algorithm. However, λ is an eigenvector of Q, and hence can be found in polynomial time by algorithms for computing eigenvectors.

We note that if the q_i vectors are strictly positive then there is a unique λ satisfying the proposition (the unique stationary distribution of the random walk implied by the matrix Q). $\square$

1.7 Correlated Nash

We shall consider now ϵ-correlated equilibrium. In a multiplayer game with a correlation device, a set of strategies is in an ϵ-correlated equilibrium in no player can gain more than ϵ in his expected payoff by deviating from the recommended strategy implied by the signal.

Theorem 1.6 *Let G be an arbitrary finite multiplayer game with payoffs in $[0, 1]$, and let G^T be its T-fold repeated version. If players use strategies with sublinear swap regret, then for every $\epsilon > 0$, there is a sufficiently large T such that the resulting empirical joint distribution of actions is an ϵ-correlated equilibrium.*

Let us clarify the action of the correlation device. Given the execution on the T rounds, it picks a round at random, and recommends to each player to player whatever he play on that round, but without revealing the round number. If given a recommendation to play i player P wants to switch to j, this implies that the execution of G^T had significant swap regret for P.

Observe that if the correlation devices can recommend mixed strategies to players rather than pure strategies, then low fractional swap regret suffices in the proof of Theorem 1.6.

Theorem 1.6 implies that algorithms such as RWM can compute correlated equilibria. This is consistent with the findings of Section 1.5 which shows that they can be used in order to solve linear programs in general, and we have seen previously that a correlated equilibrium is a solution to a linear program.

1.8 Bandit algorithms

Online optimization in which all payoffs are observed after each round is referred to as *expert* algorithms. If only the payoff of the action played is observed, this is referred to as the *multi armed bandit* (MAB) setting. As a rule of thumb, expert algorithms can be transformed into MAB algorithms using the paradigm of *exploration and exploitation.* We present one approach of achieving this (though one may obtain better parameters through more complicated approaches).

Partition T into blocks of size B. Within every block, the algorithm sticks to the same distribution over actions, except that each of the actions is tried once in a random round within the block, so as to gather statistics about its performance. Hence $B > N$. Each block is viewed as a "super round", and in between blocks, the update rule is that of the underlying expert algorithm, where observed sampled payoffs of action i are scaled by B, as if the payoff in all rounds of the block is the same as the sampled payoff. There are several sources of regret in this MAB algorithm:

1. There are $\frac{NT}{B}$ exploration steps. This adds a regret term of $\frac{NT}{B}$.
2. The true sum of payoffs of an action differs from the sampled one. Each action is sampled T/B times, giving accuracy of $1 - O(\sqrt{B/T})$, leading to an error term of $O(\sqrt{BT})$, and with high probability not exceeding $\sqrt{BT \log N}$.

3. Compared to the expert setting, now there are T/B super rounds rather than T rounds, and every super round has payoff at most B rather than 1. Hence the regret term of the expert algorithm used in between super rounds translates to $B(\epsilon\frac{T}{B}+\frac{\log N}{\epsilon})$.

Suppressing factors logarithmic in N by $\tilde{O}$ notation, we may choose $B = N^{2/3}T^{1/3}$ and $\epsilon = N^{1/3}/T^{1/3}$, obtaining a regret of $\tilde{O}(N^{1/3}T^{2/3})$.

References

[AHK12] Sanjeev Arora, Elad Hazan, Satyen Kale: The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. Theory of Computing (TOC) 8(1):121–164 (2012).

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay V. Vazirani (Editors), Algorithmic Game Theory, Cambridge University Press, 2007.