Computational Learning Theory
11 : Online Learning with Expert Advice

Lecturer: Varun Kanade

In this lecture, we will move away from the *prediction* learning framework, either binary or real-valued, and consider a more abstract sequential decision making framework. The general framework turns out to be very powerful, and as applications, we will re-derive a boosting algorithm similar to AdaBoost, as well as see a proof of von Neumann's Min-Max theorem.

1 Learning with Expert Advice

We consider a sequential decision making problem as a repeated game between a learning algorithm or a *decision maker* (DM) and an environment. The game will be played for T rounds and we assume that T is known to the decision maker.[1] The decision maker has to choose between n options at each time step; the decision maker can pick a probability distribution over n options, i.e. a point from the probability simplex,

$$\Delta_n = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n, x_i \geq 0, \sum_i x_i = 1\}.$$

These options are sometimes called *experts*; this is because we can view this as the decision maker having advice from n different experts and seeking to combine the advice to come up with its own decision rule. Hence the name learning with *expert advice*. The game proceeds in rounds as follows. For $t = 1, 2, \ldots, T$:

- At time t, DM picks $\mathbf{x}_t \in \Delta_n$.
- The environment picks a loss vector $\mathbf{l}_t \in [0,1]^n$.
- The loss suffered by DM at time t is $\mathbf{x}_t \cdot \mathbf{l}_t$ and DM observes the entire loss vector, $\mathbf{l}_t$.

First, we consider the choice of the decision maker. The decision maker may choose $\mathbf{x}_t$ at time t based on all the available information, i.e. $\mathbf{x}_1, \mathbf{l}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{l}_{t-1}$. In principle, we would allow $\mathbf{x}_t$ to be an arbitrary computable function of the history, though the algorithms we study turn out to be computationally efficient.

Second, we can consider the loss vectors $\mathbf{l}_t$ produced by the environment. We will not constrain the environment beyond the requirement that each entry of $\mathbf{l}_t$ be in the interval $[0,1]$. In fact, the environment may chose the loss vector $\mathbf{l}_t$ in response to all the available history, including the choice $\mathbf{x}_t$ made by the decision maker at time t.

How do we measure performance of the decision maker in this case? Clearly simply requiring the DM to minimise loss across the T time-steps is not sufficient as the environment could simply require all n options to have a loss of 1 at each time-step. Instead, we will be interested in a notion of relative performance with respect to a class of strategies. We will look at the simplest set of strategies, i.e. strategies that don't change over time. However, we will find the *best* such strategy in hindsight. We define the notion of *regret* of the decision maker, DM, as,

$$\text{Regret}(\mathsf{DM}) = \sum_{t=1}^{T} \mathbf{x}_t \cdot \mathbf{l}_t - \min_{\mathbf{x} \in \Delta_n} \sum_{t=1}^{T} \mathbf{x} \cdot \mathbf{l}_t.$$

[1] This is not a serious restriction. For tricks to remove this restriction refer to Cesa-Bianchi and Lugosi (2006).

It is worth commenting on the notion of regret before we design algorithms to minimise regret. The decision maker has the advantage of being able to change its decisions every time-step but has the disadvantage of not knowing the future. On the other hand, the performance comparison is to a fixed strategy which has the benefit of hindsight. (It is worth observing that regret may be negative in this case.) What this suggests is that if there was a fixed strategy that would have performed "well", then having low regret would suggest that the decision maker performs almost as well. Thus, if the environment was relatively benign with loss vectors not changing very rapidly and the same options doing well over time, we would expect the decision maker to eventually figure out a good distribution over options. On the other hand, if the environment was very adversarial and no fixed strategy could have achieved a good performance then we would not expect the decision maker to have low loss even if it had low regret. Of course, one may take the view that the comparison to fixed strategies is too restrictive and we would like the decision maker to have low regret with more complex strategies. Indeed, such problems can be considered and the interested reader is referred to the excellent book by Cesa-Bianchi and Lugosi (2006) for several generalisations and extensions.

A natural interpretation of the above formulation is in terms of betting over n differen options which may change in value over time in uncertain ways. We can spread a unit amount of money every day and we would like our long term performance to be not much worse than the single best option (or single distribution) in hindsight. In fact, because the loss function $\mathbf{l} \cdot \mathbf{x}$ is linear in $\mathbf{x}$, the minimum is (also) achieved at a vertex of the simplex Δ_n. So we may in fact simply compare our performance with the best single option and rewrite the definition of regret as,

$$\text{Regret}(\mathsf{DM}) = \sum_{t=1}^{T} \mathbf{x}_t \cdot \mathbf{l}_t - \min_{i \in [n]} \sum_{t=1}^{T} l_{t,i}.$$

2 Follow The Leader

We will use the notation $\mathbf{L}_t = \sum_{s=1}^{t-1} \mathbf{l}_s$. A natural strategy is to pick $\mathbf{x}_t$ at time t that minimises the *cumulative* historical loss, i.e. pick $\mathbf{x}_t \in \Delta_n$ that minimises $\mathbf{x}_t \cdot \mathbf{L}_t$. In fact, this $\mathbf{x}_t$ can be chosen to be one of the vertices of the simplex Δ_n, i.e. we may pick $\mathbf{x}_t$ with $x_{t,i} = 1$ for $i \in \text{argmin}_i L_{t,i}$ and $x_{t,j} = 0$ for $j \neq i$. This algorithm is sometimes called *Follow-the-Leader* as it picks the strategy that has historically proved the best.

It turns out however that a strategy that would have perfomed best on historical observations, may not perform well in the future, and indeed this algorithm does not get a good bound on the regret. The following example with only two options, when T is even, establishes the problem with this strategy.

	$\mathbf{l}_1$	$\mathbf{l}_2$	$\mathbf{l}_3$	$\mathbf{l}_4$	...	$\mathbf{l}_T$
Option 1	0.5	0	1	0	...	0
Option 2	0	1	0	1	...	1

The strategy may pick either option at time $t = 1$, but subsequently it would always pick option 2 in even time steps and option 1 in odd time steps. Thus, it would incur a total loss of at least $T - 1$. However, the first option only has a loss of $T/2 - 1/2$. Thus, the regret incurred is at least $T/2 - 1/2$ which grows linearly in T. At least asymptotically this is a completely trivial regret bound as no algorithm incurs a loss of more than T as the losses are constrained to be in $[0, 1]$ at each time step.

Although the Follow-the-Leader strategy does not work well in this particular case, there are online optimization problems in which the strategy does give non-trivial (and in fact close to optimal) regret bounds. The problem in this specific setting is that the decision rule is too unstable, i.e. it may change drastically at every time step based on small changes to the loss

vectors. In the next section, we will see a small modification to the algorithm that is more stable that does actually give optimal regret bounds.

3 The Multiplicative Weight Update Algorithm (MWUA)

Algorithm 1 presents the Multiplicative Weight Update Algorithm (MWUA) for the sequential decision making problem. This algorithm, or close variants, go by various names including Weighted Majority, Exponentially-weighted Forecaster, Hedge, etc. The reason for these names and indeed variants is that very similar abstractions of this sequential decision making problem have been studied in various fields including information theory, game theory, learning theory, complexity theory, among others. The excellent survey article by Arora et al. (2012) gives several variants of this algorithm and applications to approximation algorithms and optimization. The book by Cesa-Bianchi and Lugosi (2006) considers various problems in learning theory, game theory, information theory and optimization.

Algorithm 1 The Multiplicative Weight Update Algorithm (MWUA)

State $\mathbf{w}_1 = (1, \ldots, 1) \in \mathbb{R}^n$
for $t = 1, 2, \ldots, T$ **do**
  $\mathbf{x}_t = \frac{\mathbf{w}_t}{Z_t}$, $Z_t = \sum_{i=1}^{n} w_{t,i}$
  Play $\mathbf{x}_t$
  Receive loss $\mathbf{x}_t \cdot \mathbf{l}_t$
  Observe $\mathbf{l}_t$
  For each i, let $w_{t+1,i} = w_{t,i} \cdot \exp(-\eta l_{t,i})$
end for

Before we analyse the algorithm, let us see in what sense this algorithm is a tweak of the *Follow-the-leader Algorithm*. Let $\mathbf{L}_t = \sum_{s=1}^{t-1} \mathbf{l}_s$ as defined above. Then the Follow-the-leader strategy simply picks $\mathbf{x}_t$ that minimises $\mathbf{L}_t$, which can be represented by vectors in $\operatorname{argmin}_{\mathbf{x} \in \Delta_n} \mathbf{x} \cdot \mathbf{L}_t$. In fact, all vectors in $\operatorname{argmin}_{\mathbf{x} \in \Delta_n} \mathbf{x} \cdot \mathbf{L}_t$ are exactly the subgradients of the function $f : \mathbb{R}^n \to \mathbb{R}$, given by $f(\mathbf{z}) = \min_i z_i$ at the point $\mathbf{L}_t$. Instead, if we define the function,

$$f^\eta(\mathbf{z}) = -\frac{1}{\eta} \ln \left(\sum_{i=1}^{n} e^{-\eta z_i} \right),$$

then it is easy to see that in the limit as $\eta \to \infty$ $f^\eta(\mathbf{z}) = \min_i z_i$. Thus, f^η is from a family of *soft min* functions, which are differentiable; the gradient of f^η is called the argsoftmin function and indeed the MWUA algorithm sets $\mathbf{x}_t = \nabla f^\eta(\mathbf{L}_t)$. The *softening* of the min function can be viewed as a form of regularization. Yet another view of the $\mathbf{x}_t$ picked by the algorithm is the following, $\mathbf{x}_t$ is the constrained minimum of the following optimization problem,

$$\min_{\mathbf{x} \in \Delta_n} \mathbf{L}_t \cdot \mathbf{x} - \frac{1}{\eta} H(\mathbf{x}).$$

where $H(\eta) = -\sum_{i=1}^{n} x_i \ln x_i$ is the entropy function. The *negative* entropy term above acts as a regularizer. Minimising the function $\mathbf{L}_t \cdot \mathbf{x}$ pushes the solution to a corner of the simplex, whereas minimising the negative entropy term pushes it towards the centre of the simplex, i.e. encourages more *hedging*.

Indeed, many analyses of the regret bound of MWUA are possible using the interpretations above, some of which lead to more insightful proofs. We will consider a short potential-based proof which uses the Z_t as a potential function.

Theorem 1. *For the* MWUA *algorithm run with* $\eta = \sqrt{\frac{n}{T}}$*, we have,*

$$\text{Regret}(\mathsf{MWUA}) \leq 2\sqrt{T \ln n}.$$

Proof. Consider the following,

$$\frac{Z_{t+1}}{Z_t} = \frac{\sum_{i=1}^n w_{t+1,i}}{Z_t} = \frac{\sum_{i=1}^n w_{t,i} \exp(-\eta l_{t,i})}{Z_t}$$

By the convexity of $z \mapsto e^{-\eta z}$, we have $\exp(-\eta z) \leq 1 + (e^{-\eta} - 1)z$ for $z \in [0, 1]$. Thus, we have,

$$\begin{aligned}\frac{Z_{t+1}}{Z_t} &\leq \sum_{i=1}^t \frac{w_{t,i}}{Z_t} \cdot (1 + (e^{-\eta} - 1) l_{t,i}) \\ &= 1 + (e^{-\eta} - 1)\mathbf{x}_t \cdot \mathbf{l}_t.\end{aligned}$$

Above, we have used the definition $\mathbf{x}_t = \mathbf{w}_t / Z_t$. Further, using the fact that $1 + x \leq e^x$, we have,

$$\frac{Z_{t+1}}{Z_t} \leq \exp\left((e^{-\eta} - 1)\mathbf{x}_t \cdot \mathbf{l}_t\right).$$

We can multiply the ratios Z_{t+1}/Z_t for $t = 1, \ldots, T$, and use the bound above to obtain,

$$\frac{Z_{T+1}}{Z_1} \leq \exp\left((e^{-\eta} - 1) \cdot \sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t\right). \tag{1}$$

On the other side, we note that $w_{t,i} = \exp(-\eta \sum_{s=1}^{t-1} l_{s,i})$. So we have the following for each $i \in [n]$,

$$\frac{Z_{T+1}}{Z_1} \geq \frac{w_{T+1,i}}{n} = \frac{\exp\left(-\eta \sum_{t=1}^T l_{t,i}\right)}{n}. \tag{2}$$

Combining Equations (1) and (2), and taking natural logarithms, we have for each $i \in [n]$,

$$-\eta \sum_{t=1}^T l_{t,i} - \ln n \leq (e^{-\eta} - 1) \sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t.$$

Moving $\ln n$ to the RHS and adding $\eta \sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t$ to both sides, we get,

$$\eta \left(\sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t - \sum_{t=1}^T l_{t,i}\right) \leq (e^{-\eta} - (1 - \eta)) \sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t + \ln n.$$

Using the fact that $e^{-\eta} \leq 1 - \eta + \eta^2$ for $\eta \in [0, 1]$ and dividing both sides by η, we have for all i,

$$\sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t - \sum_{t=1}^T l_{t,i} \leq \eta \sum_{t=1}^T \mathbf{x}_t \cdot \mathbf{l}_t + \frac{\ln n}{\eta}.$$

Finally, noticing that $\mathbf{x}_t \cdot \mathbf{l}_t \leq 1$ for each t, setting η as in the statement of the theorem, and observing that because the above holds for each $i \in [n]$, by linearity, it applies to each $\mathbf{x} \in \Delta_n$, we get the result as follows:

$$\sum_{t=1}^{T} \mathbf{x}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} \sum_{i=1}^{n} x_i l_{t,i} \leq \eta T + \frac{\ln n}{\eta} \leq 2\sqrt{T \ln n}.$$

As the above applies to each $\mathbf{x} \in \Delta_n$, we are done. □

We remark that the choice $\mathbf{x}_t$ made by the algorithm at time t only depends on the loss vectors $\mathbf{l}_1, \ldots, \mathbf{l}_{t-1}$ given by the environment (in fact only on their sum). There is no dependence whatsoever on the past choices made by the algorithm, and the algorithm itself is completely deterministic. Hence, even if the environment completely adversarially picks the loss vectors $\mathbf{l}_t$ the algorithm still retains the guarantee. In fact, there is no need for the environment to wait to see the choice made by the algorithm as it can be computed directly using the previous loss vectors. In this sense, the MWUA algorithm achieves guarantees against adaptive and adversarial environments. This particular aspect will be useful in applications to game theory.

3.1 Lower Bound

In this section, we will give a sketch of why the bound achieved in Theorem 1 is essentially tight (up to constant factors). We will not give a formal proof, though writing a formal proof is not very difficult. We will also only establish a lower bound of $\Omega(\sqrt{T})$ for deterministic algorithms, when there are only $n = 2$ options. The environment at time t picks the loss vector $(1, 0)$ or $(0, 1)$ with equal probability independently at each time step. Now for any algorithm, its expected loss is exactly $T/2$. On the other hand the expectation of $\min_{i \in \{1,2\}} \sum_{t=1}^{T} l_{t,i}$ is $T/2 - \Omega(\sqrt{T})$. This follows from a simple exercise: if $X \sim \text{Binomial}(T, 1/2)$, what is the $\mathbb{E}\left[\min\{X, T - X\}\right]$? This shows that the $\sqrt{T}$ factor in the regret bound is tight; a slightly more detailed construction shows that $\sqrt{\ln n}$ is also required in the regret bound as a multiplicative factor. The formal proof appears in Cesa-Bianchi and Lugosi (2006).

4 Application: Boosting Algorithm

In this section, we will see an application of the MWUA algorithm to get a boosting algorithm. As in the case of AdaBoost, we will assume that the weak learner returns hypotheses from a hypothesis class, H, of finite VC dimension and focus on finding a (combined) hypothesis that is consistent with the training data. We will not worry about generalization error here as it can be done by bounding the VC dimension of the resulting classifier exactly as in the case of AdaBoost. In fact, there are many similarities between the AdaBoost algorithm and MWUA and indeed in their analyses.

Let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be a training dataset with m points, where $y_i = c(\mathbf{x}_i)$ for some $c \in C$. Here C is the concept class for which we have a weak learning algorithm available.

We are going to treat this as a sequential decision making problem with m options. At each time t, the decision maker, DM, will use MWUA to pick a distribution, which we will call $\mathbf{d}_t \in \Delta_m$ to distinguish it from the data $(\mathbf{x}, y)$.

We will also simulate the environment. For this reason, we will use the weak learning algorithm, WeakLearn. We shall assume for simplicity that the weak learning algorithm succeeds with probability 1 instead of probability $1 - \delta$. At time t, let h_t be the hypothesis returned by the weak learning algorithm with respect to distribution $\mathbf{d}_t \in \Delta_m$. Note that the distribution $\mathbf{d}_t$, puts probability mass $d_{t,i}$ on $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, m$. In particular, we have $\text{err}(h_t; \mathbf{d}_t) \leq \frac{1}{2} - \gamma$,

where γ is the parameter of WeakLearn. We define the loss vector $\mathbf{l}_t \in [0,1]^m$ as follows, $l_{t,i} = 1$ if $h_t(\mathbf{x}_i) = y_i$ and $l_{t,i} = 0$ otherwise. Based on the construction and the property of the weak learner, we have $\mathbf{d}_t \cdot \mathbf{l}_t = 1 - \mathrm{err}(h_t; \mathbf{d}_t) \geq \frac{1}{2} + \gamma$ for every t.

We can make use of the regret guarantee. Note that $l_{t,i} = \mathbb{1}(h_t(\mathbf{x}_i) = y_i)$. Theorem 1 gives us that provided the DM uses MWUA to generate the decisions $\mathbf{d}_t \in \Delta_m$, we have for each i,

$$\sum_{t=1}^{T} \mathbf{d}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} \mathbb{1}(h_t(\mathbf{x}_i) = y_i) \leq 2\sqrt{T \ln m}.$$

Using, the fact that $\mathbf{d}_t \cdot \mathbf{l}_t \geq 1/2 + \gamma$ for each t, and rearranging, we get,

$$T \cdot \left(\frac{1}{2} + \gamma\right) - 2\sqrt{T \ln m} \leq \sum_{t=1}^{T} \mathbb{1}(h_t(\mathbf{x}_i) = y_i).$$

Hence,

$$\frac{T}{2} + \sqrt{T} \cdot (\gamma\sqrt{T} - 2\sqrt{\ln m}) \leq \sum_{t=1}^{T} \mathbb{1}(h_t(\mathbf{x}_i) = y_i).$$

Observe that for $T > 4 \ln m/\gamma^2$, the LHS is strictly larger than $T/2$. This means that the majority of the hypotheses h_t classify the i^{th} example $(\mathbf{x}_i, y_i)$ correctly for every i. Thus simply outputting majority$\{h_1(\mathbf{x}), \ldots, h_T(\mathbf{x})\}$ will give a hypothesis that is consistent with the training set. Note that the number of calls made to the weak learning algorithm is of the same order as that made by Adaboost and as the majority function can be written as thresholds of hypotheses from the hypothesis class used by WeakLearn, the generalization error can be bounded in exactly the same way as in the analysis of AdaBoost.

5 Application: von Neumann's Min-Max Theorem

As another application of the MWUA algorithm, we will give a proof von Neumann's Min-Max theorem. We will be brief and terse in our description of two player zero-sum games. Readers unfamiliar with these notions may wish to read the introductory chapter in a book on game theory.

We will consider finite two player zero-sum games with bounded payoffs. We will assume that all payoffs are bounded in $[0, 1]$. The game has two players, typically called a row player and a column player. The game is specificed as an $n \times m$ matrix, A, with n rows and m columns. The row player has to pick one of n options and the column player has to pick one of m options. If the row player picks i and the column player picks j, then the row player gets a payoff of A_{ij}. (As it is a zero-sum game, the payoff to the column player is $-A_{ij}$.) Clearly, there is an advantage to moving *second* as you can observe the option picked by the other player. Suppose the row player moves first: if they pick i, the column player will certainly pick j that will minimize the payoff of the row player (as it is a zero sum game), resulting in a payoff of $\min_j A_{ij}$; thus to maximise their payoff they will pick i for which this quantity is the largest. In other words, the best achievable payoff for the row player is $\max_i \min_j A_{ij}$.

An entirely identical arugment shows that the best payoff achieved by the row player if they go second is $\min_j \max_i A_{ij}$. This is of course assuming that both players are playing rationally. Thus, clearly we have that,

$$\max_i \min_j A_{ij} \leq \min_j \max_i A_{ij}.$$

When players are forced to pick a single option, something which is known as *pure strategies*, the above inequality can indeed be strict. The following simple game in which both players have two options establishes a strict inequality.

	1	2
1	3	2
2	1	4

For this game, $\max_i \min_j A_{ij} = 2$ and $\min_j \max_i A_{ij} = 3$. However, we can also consider *mixed* strategies, where rather than picking a single option, each player picks a distribution over their options. The other player can see the distribution of their competitor, but not the actual random choice. For example, think of playing rock-paper-scissors with someone, where you know their (random) strategy, but can't predict what particular random choice they will make when actually playing the game. Now suppose the row player has a strategy $\mathbf{x} \in \Delta_n$ and a column player has strategy $\mathbf{y} \in \Delta_m$, then the payoff to the row player (in expectation) is,

$$\mathop{\mathbb{E}}_{i \sim \mathbf{x}, j \sim \mathbf{y}} \left[A_{ij} \right] = \mathbf{x}^\top A \mathbf{y}.$$

We can similarly define optimal strategies for the row player depending on whether they are going first or second. When the row player goes first, the optimal value they can achieve is called the maxmin value, denoted by v_{maxmin}, and defined as,

$$v_{\text{maxmin}} = \max_{\mathbf{x} \in \Delta_n} \min_{\mathbf{y} \in \Delta_m} \mathbf{x}^\top A \mathbf{y}.$$

Similarly, the best value achievable by the row player if they go second is the minmax value, denoted by v_{minmax}, and defined as,

$$v_{\text{minmax}} = \min_{\mathbf{y} \in \Delta_m} \max_{\mathbf{x} \in \Delta_n} \mathbf{x}^\top A \mathbf{y}.$$

As in the previous case, we can immediately see that $v_{\text{maxmin}} \leq v_{\text{minmax}}$.[2] What von Neumann's theorem states is that, for finite two player zero-sum games, $v_{\text{maxmin}} = v_{\text{minmax}}$.

We could of course frame this theorem in terms of losses rather than payoffs, but that would deviate from standard statements of this result. Instead, we make a series of observations. It is without loss of generality to assume that payoffs are in $[0, 1]$. First, we can always make payoffs for the row player to be *positive* because as solution concepts, zero-sum games and constant-sum games are identical. Second, we can always scale payoffs so that they are in the interval $[0, 1]$. Finally, we can replace losses in the sequential decision making problem by payoffs, or rather consider loss vectors as coming from payoff vectors, $\mathbf{p}_t$ and set $\mathbf{l}_t = \mathbf{1} - \mathbf{p}_t$. Then, we can rephrase the statement of Theorem 1 as,

$$\text{Regret}(\mathsf{MWUA}) = \max_{\mathbf{x} \in \Delta_n} \sum_{t=1}^{T} \mathbf{x} \cdot \mathbf{p}_t - \sum_{t=1}^{T} \mathbf{x}_t \cdot \mathbf{p}_t \leq 2\sqrt{T \ln n}. \tag{3}$$

We will simply run the algorithm by using $\mathbf{l}_t = \mathbf{1} - \mathbf{p}_t$ as our loss vectors when we are actually given payoff vectors $\mathbf{p}_t$.

With that out of the way, we are ready to give a proof of von Neumann's Min-Max theorem using the MWUA algorithm. The row player will implement the MWUA algorithm (with payoff

[2] Essentially, this side of the argument is weak duality, and von Neumann's theorem states that strong duality holds.

vectors suitably converted to loss vectors). Let $\mathbf{x}_t \in \Delta_n$ be the strategy picked by MWUA at time t. The column player picks $\mathbf{y}_t$ as,

$$\mathbf{y}_t \in \operatorname*{argmin}_{\mathbf{y}} \mathbf{x}_t^\top A\mathbf{y}.$$

Clearly, we have that,

$$\mathbf{y}_t^\top A\mathbf{y} = \min_{\mathbf{y}\in\Delta_m} \mathbf{x}_t^\top A\mathbf{y} \leq \max_{\mathbf{x}\in\Delta_n} \min_{\mathbf{y}\in\Delta_m} \mathbf{x}^\top A\mathbf{y} = v_{\text{maxmin}}.$$

We set the payoff vector $\mathbf{p}_t = A\mathbf{y}_t \in [0,1]^n$; and note that the above inequalities show that $\mathbf{x}_t \cdot \mathbf{p}_t \leq v_{\text{minmax}}$ for all t. Hence,

$$\sum_{t=1}^{T} \mathbf{x}_t \cdot \mathbf{p}_t \leq T v_{\text{maxmin}}. \tag{4}$$

On the other hand, we have,

$$\begin{aligned}\max_{\mathbf{x}\in\Delta^n} \sum_{t=1}^{T} \mathbf{x} \cdot \mathbf{p}_t &= T \cdot \max_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A \frac{1}{T} \sum_{t=1}^{T} \mathbf{y}_t \\ &\geq T \cdot \min_{\mathbf{y}\in\Delta_m} \max_{\mathbf{x}\in\Delta_n} \mathbf{x}^\top A\mathbf{y} = T v_{\text{minmax}}. \end{aligned} \tag{5}$$

Combining Eqns. (3), (4) and (5), we get that,

$$v_{\text{minmax}} - v_{\text{maxmin}} \leq \frac{1}{T} \cdot \text{Regret}(\mathsf{MWUA}) \leq 2\sqrt{\frac{\ln n}{T}}.$$

Since the above holds for all $T \geq 1$, it must be that $v_{\text{minmax}} \geq v_{\text{maxmin}}$ which completes the proof of the theorem.

References

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University press, 2006.