ECE 8803: Online Decision Making in Machine Learning **Fall 2021**

Lecture 7: September 15

Lecturer: Vidya Muthukumar

Thus far, we have been examining the toy setting of prediction of a binary sequence of 0's and 1's. We have looked at explicit algorithms that predict this sequence as accurately as possible without making any assumptions on how the sequence was generated; in fact, the sequence could be adversarial.

While we discussed these ideas in this simple prediction setting, the ideas can be framed in a much more general decision-making setting. In this lecture, we build a bridge to decision-making, and introduce the setting of online optimization in its most basic form.

7.1. From binary sequence prediction to decision-making using experts

We have been using the notation $l_{t,x} := \mathbb{I}[X_t \neq x]$ to indicate the "instantaneous" losses that are incurred by predicting 0 and 1 respectively. It turns out that all of the ideas that we have explored in class can be significantly extended to a setting with K "experts", and loss functions $l_{t,x}$ for each expert $x \in \{1, \dots, K\}$. The goal remains the same as before: to aggregate the performance of all experts in an optimal way regardless of how these loss functions are chosen (as long as they are bounded, e.g. between 0 and 1).

Accordingly, we as the online learner now choose a distribution over all K experts, given by $\boldsymbol{p}_t := \begin{bmatrix} p_{t,1} & \dots & p_{t,K} \end{bmatrix}$. Concretely, the regret with respect to the best expert in hindsight is defined just as before:

$$R_T := H_T - \min_{x \in \{1,\dots,K\}} L_{T,x}. \tag{7.1}$$

Generalizations of FTL, MWA and FTPL, the three algorithms that we studied in depth in this class, can be obtained to this setting of decision-making using K experts. Since FTL failed to obtain sublinear in T regret even in the binary prediction setting, it clearly will not succeed in this more general K-experts setting. However, MWA and FTPL continue to enjoy a $\mathcal{O}(\sqrt{T})$ regret guarantee. Of particular interest, though, is the dependence on the number of experts K. A meta-theorem statement is as follows:

Theorem 1 (Cesa-Bianchi et al. (1997); Freund and Schapire (1997); Kalai and Vempala (2005
Consider the K-experts setting with losses $l_{t,x} \in [0,1]$ for all $x \in \{1, \dots, K\}$. MWA and FTPL, both with learning rate $\eta = 1/\sqrt{T}$, achieve the regret guarantee

$$R_T = \mathcal{O}(\sqrt{T \log K}).$$

Thus, the regret grows logarithmically in the number of experts K. We will not examine this dependence on K in depth in this course, but you are welcome to learn about it on your own if you are really interested. It is actually a worthwhile exercise to try to generalize the proofs that we did in class to this more general K-expert setting; the multiplicative weights proof actually works as is, but the FTPL proof turns out to be significantly more complicated.

7.1.1 Applications of the experts paradigm

This paradigm of *decision-making using expert advice* (DEA) was first considered in the 1950's Hannan (1957) with a relatively abstract motivation of game playing[1]: the learner has K actions available to her, and her loss functions are induced by the actions of an opponent who is adversarial to her. However, this paradigm has several applications that are significantly less abstract. DEA saw a significant resurgence in the 1980's and 1990's through the motivations of binary sequence compression (the Lempel-Ziv compression scheme Ziv and Lempel (1977) is actually an instance of online learning!), financial forecasting Cover (2011), and ensemble methods in machine learning Freund and Schapire (1997). DEA-based applications are heavily used in all of these domains. Here, we provide a high-level and informal description of each of these applications in the DEA framework.

Example 1 (A simplified[2] variant of portfolio optimization) *Suppose we have K assets that we wish to invest in, and a fixed capital that we can spread across these assets — we denote our investment at "time step" t by the K-dimensional probability vector $\boldsymbol{p}_t$, where $p_{t,x}$ denotes the fraction of capital that we invest in asset x. The value of these assets changes across time, and at every round we can denote the profit that we would have made had we invested in asset x by $r_{t,x}$. Naturally, our goal here is to maximize our long-term profit.*

The stock market is a quintessential example of unpredictability: assets heavily fluctuate in their value, and accurately forecasting them is extremely difficult (although statistical models for this do exist, their relative success continues to be heavily debated). Thus, the adversarial model actually makes sense to assume: in particular, a single asset may not consistently be the best across time.

It is clear that this a natural manifestation of the DEA paradigm; in particular, "hedging" across assets (using MWA, FTPL, or some other no-regret algorithm) makes intuitive sense. Using these algorithms for portfolio optimization implies that we can achieve a profit that is almost as good *(in the sense of regret) as the best fixed allocation across time in hindsight. See Cover (2011) for a description of "universal portfolio" algorithms that achieve this kind of guarantee (in a more complex and realistic setting than the one presented here).*

Example 2 (Boosting in machine learning) *The classification task, which involves predicting a discrete-valued label y from input x, is a classical goal of machine learning (ML). Suppose that we have a training dataset given by $\{x_i, y_i\}_{i=1}^n$. Frequently, we have a scenario in which we have K "weak" classifiers that do well on some of the training examples, but not others. More generally, if the training examples are drawn from a distribution, each of these classifiers may do slightly better than random guessing on average, i.e. for every*

1. We will return to this game-theoretic motivation at the end of the course.

$k \in \{1, \ldots, K\}$ we have $\mathbb{E}[\mathbb{I}[f_k(x) = y]] \geq \frac{1}{2} + \gamma$ for some $\gamma > 0$. It turns out that we can get much *better performance (in fact, arbitrarily close to 1!), by aggregating these K classifiers using an adaptive online majority-voting scheme that is directly inspired by MWA Freund et al. (1997); Freund and Schapire (1997). This scheme is commonly called* ***boosting****, and constitutes an ensemble method in ML as it aggregates several "weak learners" optimally to produce a "strong learner". We will touch upon this application of boosting later in the course, in conjunction with connections between online learning and game theory.*

Ideas from DEA continue to be used in diverse applications today. In particular, DEA is used to aggregate "experts" in a number of ML and algorithmic applications. Two notable modern applications of DEA are below:

- The *meta-learning* application, which constitutes aggregating the output of ML algorithms trained for different sub-tasks, to generalize well on a new task. Here, the various ML algorithms are the "experts". See Finn et al. (2019) for a recent example of DEA applied to meta-learning.
- A *meta-algorithm* for a challenging NP-hard optimization problem (such as the k-satisfiability problem (k-SAT)) that optimally combines heuristics that are each good for certain subclasses of instances. Here, the heuristics are the "experts". See Xu et al. (2008) and code there-in for a description of the meta-algorithm *SAT-Zilla*, which has won several SAT solver competitions.

7.2. Online linear optimization (OLO)

While the DEA framework has a very diverse set of applications, its use is limited in certain contexts. For example, it only allows us to choose among a *finite* set of options, while several applications in practice may need us to make decisions from an infinite or continuous set. This motivates the much more general *online optimization* paradigm. We now look at an explicit example of this paradigm, which is also motivated by finance.

7.2.1 An example: Buying/selling stocks

A common problem in finance applications is to decide which stocks to buy, and which to sell, over time. The intuition is roughly as follows: if a stock is likely to *increase* in value in the future, then we would like to buy more of it now so that we can later sell it at a profit. On the other hand, if a stock likely to *decrease* in value in the future, then we would like to sell it now rather than later to avoid a loss. Our overall goal, of course, is to "play the market" in order to maximize long-run profit.

We can model this problem as an online optimization problem. Suppose we are playing the stock market for T days, and considering d stocks that fluctuate in value over time. Each morning when the market opens, we go over this list of stocks, and choose to either buy or sell some amount of each one. This decision is represented by a vector $\boldsymbol{w}_t \in [-D, D]^d$, whose definitely we now unpack. Here, t indexes the day, and $w_{t,i}$ represents the amount corresponding to stock i. If $w_{t,i}$ is positive, then we have chosen to buy stock; if negative, we have chosen to sell or "short" stock. For example, if $w_{t,i} = 3$, we decided to buy 3 shares of stock on the morning of day t; if $w_{t,i} = -2$, we decided to short 2 shares of stock on the

Loss incurred	Value increased by $\ell_{t,i}$	Value decreased by $\ell_{t,i}$
Buy k units	$-k\ell_{t,i}$	$k\ell_{t,i}$
Short k units	$k\ell_{t,i}$	$-k\ell_{t,i}$

morning of day t. Finally, D represents our limit on overall buying power on each day: we cannot buy/short unlimited amounts of stock. Concretely, we will assume a limited buying power of the form[3] $\sqrt{\sum_{i=1}^{d} w_{t,i}^2} \leq D$.

We make this decision $\boldsymbol{w}_t$ on the morning of day t. At the end of day t, we sell whatever stock we bought, and we buy back the stock that we shorted. We make or lose money depending on how much the value of the stock changed over the course of the day. Qualitatively, there are four cases:

- If we bought stock and its value decreases, then we make a loss. For example, if we bought k units of stock i and the value of the stock decreases by ℓ_i, we make a *loss* of $k\ell_{t,i}$ on that stock.
- If we bought stock and its value increases, then we make a profit. For example, if we bought k units of stock i and the value of the stock increases by $\ell_{t,i}$, we make a *profit* of $k\ell_{t,i}$ on that stock; thus a loss of $-k\ell_{t,i}$.
- If we shorted stock and its value decreases, then we make a profit. For example, if we shorted k units of stock i and the value of the stock decreases by $\ell_{t,i}$, we make a *profit* of $k\ell_{t,i}$ on that stock; thus a loss of $-k\ell_{t,i}$.
- If we shorted stock and its value increases, then we make a loss. For example, if we shorted k units of stock i and the value of the stock increases by $\ell_{t,i}$, we make a *loss* of $k\ell_{t,i}$ on that stock.

These four cases are represented pictorially in Table 7.2.1. It is then easy to verify that we can write a *loss function* in terms of the overall decision vector $\boldsymbol{w}_t$: we write $\boldsymbol{\ell}_t := \begin{bmatrix} \ell_{t,1} & \dots & \ell_{t,d} \end{bmatrix}$ as the vector that represents how much each stock depreciated, and the overall loss incurred on that day will be represented as $\langle \boldsymbol{w}_t, \boldsymbol{\ell}_t \rangle$. From the above discussion, note that the elements of $\boldsymbol{\ell}_t$ can also be either positive or negative. We will also assume them to be bounded, i.e. $||\boldsymbol{\ell}_t||_2 \leq G$; this simply represents the fact that the overall aggregate value of the stocks cannot astronomically crash or rise[4].

Our formulation now becomes an *online linear optimization* formulation. In other words, we wish to design decisions $\{\boldsymbol{w}_t\}_{t=1}^{T}$ to minimize the total loss, given by

$$H_T := \sum_{t=1}^{T} \langle \boldsymbol{w}_t, \boldsymbol{\ell}_t \rangle. \tag{7.2}$$

3. Note that there are several ways in which we could restrict our overall buying power: we could also constrain the buying power of each stock more directly (i.e. $|w_{t,i}| \leq D$), or the buying power of the sums (i.e. $\sum_{i=1}^{t} |w_{t,i}| \leq D$). All of these constraints will turn out to yield highly similar answers to the problem. We choose the ℓ_2-norm constraint for relative simplicity.
4. Like for the constraints on decisions, we could model this boundedness through an alternative set of constraints, such as $||\boldsymbol{\ell}_t||_\infty \leq G$ or $||\boldsymbol{\ell}_t||_1 \leq G$. These will turn out to yield similar answers to the one provided here.

We wish to do this subject to the constraint that $|w_{t,i}| \leq D$, i.e. $||\boldsymbol{w}_t||_\infty \leq D$. Thus, we can write our decision set $\mathcal{B}$ as the ℓ_∞ ball. (It will turn out that the methodology that we develop for this problem extends more generally to any compact decision set.)

Finally, just like in the case of sequence prediction, this is a setting in which we may not be able to make any assumptions on the evolution of the loss vectors $\{\ell_t\}_{t\geq 1}$. This is actually a very realistic concern in the case of financial applications, as the stock market is in fact highly unpredictable. Accordingly, our goal is to minimize regret with respect to the best *fixed* decision on stock trading in hindsight, i.e.

$$R_T := H_T - \min_{\boldsymbol{w}\in\mathcal{B}} \sum_{t=1}^{T} \langle \boldsymbol{w}, \boldsymbol{\ell}_t \rangle \tag{7.3}$$

Think about what the best fixed decision is doing: essentially, it gets to see the *overall* evolution of the value of each stock, and take a decision accordingly. So if stock i overall increased in value, it would be better to keep buying and selling it. On the other hand, if stock i overall decreased in value, it would be better to keep shorting and buying it.

Just like in online sequence prediction, our goal is to obtain sublinear in T regret with respect to this benchmark. We will now review basic intuition and design an algorithm that achieves this.

7.2.2 The equivalent of FTL and why it doesn't work

We can now define the natural equivalent of FTL, and provide an intuitive example to show why it doesn't work.

Definition 2 *The Follow-the-Leader algorithm in online optimization simply chooses*

$$\boldsymbol{w}_t := \arg\min_{\boldsymbol{w}\in\mathcal{B}} \left[L_{t-1}(\boldsymbol{w}) \right] := \sum_{s=1}^{t-1} \langle \boldsymbol{w}, \boldsymbol{\ell}_s \rangle$$

at every round t.

Let us look at our finance example for the case $d = 1$ (i.e. only one stock) to get a closer look at what FTL is doing. Recall that our buying/selling power every day is D units, and ℓ_t denotes the depreciation in the stock value on day t. We can then verify that FTL will make the following binary decision:

- If the stock overall depreciates over time, i.e. $\sum_{s=1}^{t-1} \ell_t > 0$, then the FTL algorithm will decide to sell *all* the stock, i.e. decide $w_t = -D$.
- If the stock increases in value over time, i.e. $\sum_{s=1}^{t-1} \ell_t < 0$, then the FTL algorithm will decide to buy *all* the stock, i.e. decide $w_t = D$.

There are situations in which FTL can indeed be verified to do well. Suppose, for example, that the depreciation of the stock followed the following pattern:

$$\begin{bmatrix} -5 & 10 & -5 & 10 & \dots & 10 \end{bmatrix}. \tag{7.4}$$

Then, while the stock value is fluctuating, it overall drifts in the direction of increase over time. It is clear that our best bet should be to buy (and sell at a profit) all the stock, and this is exactly what FTL does. This is a situation in which FTL can be verified to incur sublinear regret, and is somewhat analogous to the Bernoulli sequences that we studied in previous lectures.

On the other hand, the stock's overall amount of depreciation/increase could fluctuate around 0 in the following way:

$$\begin{bmatrix} 10 & -10 & 10 & -10 & \dots & -10 \end{bmatrix}, \tag{7.5}$$

and this is a situation in which FTL would do very poorly. To see this, note that on all *even-numbered* days, FTL would decide to buy D units of stock as the value appears to be increasing until then; however, the stock would then depreciate by 10 points, leading to a loss of $10D$. Overall, FTL will incur a loss of $10D \cdot \frac{T}{2} = 5DT$. On the other hand, with the benefit of hindsight, we would see that this stock is heavily fluctuating and we should neither have bought or sold it. The best decision in hindsight would be to neither buy nor sell on each round, which would incur no loss.

Putting these together, we see that FTL would incur a regret exactly equal to $5DT$ in this case.

7.2.3 How to achieve stability: Regularization

The above example demonstrates that the instability issue of FTL that we first observed in binary sequence prediction persists for the case of online optimization! In binary sequence prediction, we saw that adding randomization *stabilizes* the updates and makes them less vulnerable, on average, to unpredictability in the process. How do we achieve stability here? We first provide a mathematical explanation, and then motivate it through the stock market example. Notice, first, that the decision vector $\boldsymbol{w}_t = \mathbf{0}$ is included in the decision set $\mathcal{B}$ (for any norm, not just the ℓ_2-norm). Also observe that regardless of the loss sequence $\{\boldsymbol{\ell}_t\}_{t=1}^T$, we would have $\langle \boldsymbol{w}_t, \boldsymbol{\ell}_t \rangle = 0$ for all t. Thus, such a simple decision is extremely stable to perturbations in the loss vectors.

In the context of the stock market example, this represents a very conservative or "low-risk" decision: we simply decide not to play the market! This is in stark constrast to the case of FTL, which goes "all-in" by maximally exploiting the information present in past observations of the stock market. Of course, this kind of low-risk decision-making is optimal in some settings, such as the example of highly fluctuating stock values in Equation (7.5). However, it is also unnecessarily conservative in a more favorable situation where the stock value drifts in one direction or the other, such as in Equation (7.4). In this scenario, we really should be taking advantage of the drift and deciding to buy the stock. Here, purely low-risk decision-making would incur a linear regret, as the best-loss-in-hindsight would be given by $5DT$ (the overall loss incurred by deciding to buy the stock on every day).

7.2.4 The Follow-the-Regularized-Leader algorithm

The discussion above makes it clear that we want to balance high-risk elements (FTL) and low-risk elements (the stable decision of $\mathbf{0}$) in our decision making. We can easily achieve this by adding a level of *regularization* to FTL! Let us describe this algorithm below.

Definition 3 *The Follow-the-Regularized-Leader (FTRL) algorithm chooses*

$$\boldsymbol{w}_t := \arg\min_{\boldsymbol{w}\in\mathcal{B}} \left[L_{t-1}(\boldsymbol{w}) + \frac{1}{\eta}||\boldsymbol{w}||_2^2 \right],$$

*where $\eta > 0$ is a **learning rate** parameter, analogous to the one that we picked in binary prediction.*

Notice that η naturally measures the amount of risk we take: a higher value of η leads to less regularization, and more weight placed on the past observations (therefore, higher risk), while a lower value of η leads to more regularization, and less weight placed on the past observations (therefore, lower risk). Two extreme cases are below:

- If $\eta \to \infty$, FTRL reduces to the highest-risk option, FTL.
- If $\eta \to 0$, FTRL reduces to the lowest-risk option of picking the most stable vector $\boldsymbol{w}_t = \mathbf{0}$.

From what we have seen in class so far, we should expect that we can trade off the high-risk and low-risk elements and achieve a sublinear regret guarantee using the FTRL algorithm. Next lecture, we will see this explicitly. Moreover, we will see that for the choice of decision set and constraint on loss functions defined here (both constrained ℓ_2-ball), the FTRL algorithm has a particularly nice closed-form expression that connects to fundamental algorithms used in convex optimization.

7.3. Additional references

- For a historical perspective on the discoveries of various online learning algorithms as well as historical motivation of the online learning paradigm, see the excellent survey by Foster and Vohra (1999).
- For an excellent machine-learning-oriented survey and history of online learning, see Blum (1998). This includes the inspiration for MWA (or the weighted-majority algorithm) from the perceptron, "winnowing" algorithms and mistake bounds.
- For recent perspective on how MWA is used for algorithm design, see the excellent survey by Arora et al. (2012).
- For another example of online linear optimization motivated by online advertisement, see the lecture notes: `https://courses.cs.washington.edu/courses/cse599s/14sp/scribes/lecture2/scribeNote.pdf` from a course on online optimization that was offered at UW.
- For some simulations that were run for the finance example (and a rougher description of this example), see the lecture notes from a previous iteration of this course: `https://inst.eecs.berkeley.edu/~ee290s/fa18/scribe_notes/EE290S_Lecture_Note_8.pdf`.

References

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

Avrim Blum. On-line algorithms in machine learning. In *Online algorithms*, pages 306–325. Springer, 1998.

Nicolo Cesa-Bianchi, Yoav Freund, David Haussler, David P Helmbold, Robert E Schapire, and Manfred K Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44 (3):427–485, 1997.

Thomas M Cover. Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*, pages 181–209. World Scientific, 2011.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

Dean P Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 334–343, 1997.

James Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.