ECE 8803: Online Decision Making in Machine Learning **Fall 2022**

Lecture 6: September 8

Lecturer: Vidya Muthukumar

So far, we have learned about the fundamental tradeoff involved in prediction of a sequence in the worst case: between leveraging the information present in past data, and randomizing to avoid being exploited by an adversarially/maliciously designed sequence. We began by looking closely at the case of predicting a binary sequence of 0's and 1's, where this tradeoff becomes very clear through examples. We then introduced the *multiplicative weights algorithm* that provably navigates this tradeoff and achieve a $\mathcal{O}(\sqrt{T})$ regret with respect to the *best constant predictor (i.e.* 0 *on every round, or* 1 *on every round)* in hindsight. Last lecture, we saw how these ideas can be greatly generalized to a setting of *decision-making using expert advice*, where we can choose one out of K experts to make this prediction. We saw that the multiplicative weights algorithm can be directly generalized to this richer setting, and here yields a $\mathcal{O}(\sqrt{T \cdot \log K})$ regret guarantee.

In these two lectures, we will learn about an entire family of algorithms that also optimally navigates this tradeoff. At the heart of this family of algorithms is a somewhat different approach that essentially follows the leader *with random perturbations added.* This algorithm is popularly called *Follow-the-Perturbed-Leader*, and is introduced below.

6.1. Recap: Decision making using expert advice

We initially introduced the problem of online decision making in the simplest canonical example of *binary prediction*, i.e. predicting a sequence of 0's and 1's. There, we used the notation $\mathbb{I}[X_t \neq x]$ to indicate the "instantaneous" losses that are incurred by predicting 0 and 1 respectively. In the more general decision making setting, our decision constitutes the choice $\widehat{X}_t \in [K] := \{1, \ldots, K\}$ (which can and will be randomized) on every round, and every expert $x \in [K]$ incurs a loss of $\ell_{t,x}$. Each of the losses incurred by each expert is revealed, but only after our choice of expert $\widehat{X}_t$ is made. 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. *For the purposes of regularity, we typically assume that the loss functions remain bounded between* 0 *and* 1*, i.e.* $\ell_{t,x} \in [0,1]$ *for all experts* $x \in [K]$ *and all rounds* t*.*

Accordingly, we as the online learner now choose a distribution over all K experts, given by $\boldsymbol{p}_t := \begin{bmatrix} p_{t,1} & \ldots & p_{t,K} \end{bmatrix}$. Our expected loss is now defined as $H_T := \mathbb{E}[\sum_{t=1}^T \ell_{t,\widehat{X}_t}] = \sum_{t=1}^T \sum_{x \in [K]} p_{t,x} \ell_{t,x}$. The regret is defined in a similar manner as before, but now with

respect to the *best expert in hindsight*:

$$R_T := H_T - \min_{x \in \{1,\ldots,K\}} L_{T,x}. \tag{6.1}$$

6.2. The Follow-the-Perturbed-Leader algorithm

In Lecture 3 (August 30), we introduced the Follow-the-Leader (FTL) algorithm for binary sequence prediction, which was given by

$$\widehat{X}_t := \arg\min_{x \in \{0,1\}} L_{t-1,x}, \tag{6.2}$$

in other words, *predicting the letter out of $x \in \{0,1\}$ that was seen more often thus far*. FTL can be naturally generalized to the setting of decision-making with expert advice — our decision will now be given by

$$\widehat{X}_t := \arg\min_{x \in [K]} L_{t-1,x}. \tag{6.3}$$

We saw that FTL has significant disadvantages even in the simplest setting of binary prediction. While it performed very well against a stochastic Bernoulli sequence, but very poorly against an adversarially designed sequence as it would incur a loss of 1 on every round. (In HW 1, you will additionally show that this implies linear in T regret.)

The central issue with FTL is that it is too *sensitive* to the past data. Recall the adversarial sequence that was designed for FTL, which was given by $101010\ldots$. We saw that this example induces a change of the identity of the leader on every single round! This instability of FTL is the central reason behind its high loss (and therefore, high regret) in a malicious setting. We now examine a very simple approach to mitigate this instability. This constitutes continuing to follow the leader, but adding a significant *random perturbation* to each of the losses at every round. Concretely, we draw random variables $N_{t,x}$ that are independently and identically distributed (iid) across $t = 1, \ldots, T$ *and* $x \in [K]$, and we add them to the cumulative losses for each letter. We then follow this perturbed leader in the following way:

$$\widehat{X}_t := \arg\min_{x \in [K]} \left[L_{t-1,x} + N_{t,x} \right]. \tag{6.4}$$

Clearly, $\widehat{X}_t$ is randomized because $N_{t,x}$'s are random. We can continue to denote $p_{t,x} := \mathbb{P}[\widehat{X}_t = x]$, although we will not work with these probabilities directly in the next two lectures.

Let us now elaborate on the distribution that we choose for each of the perturbations given by $N_{t,x}$. For each value of $x \in [K]$ we will consider $N_{t,x}$ to be an exponential random variable[1] with parameter equal to η; in other words, the PDF of $N_{t,x}$ is given by $p(n) := \eta \exp(-\eta n)$ for all $n \geq 0$. The parameter η is called a learning rate and its choice critically impacts the behavior of the FTPL algorithm. To see this, we note that the variance of an exponential random variable is given by $\frac{1}{\eta^2}$. Now, we can consider two extremes:

1. See `https://en.wikipedia.org/wiki/Exponential_distribution` for basic factoids on the exponential distribution, including weird quantities like kurtosis.

- When $\eta \to \infty$, we have variance$(N_{t,x}) = 0$, and so there is no noise introduced into the algorithm. FTPL becomes exactly FTL.
- When $\eta \to 0$, we have variance$(N_{t,x}) = \infty$, and so the effect of the random perturbation completely "drowns out" the effect of the cumulative losses. Effectively, FTPL becomes like "pure guessing" in this case, making it equally likely to decide on any of the K choices on every round.

This intuition tells us that the smaller the value of η, the more randomization we are introducing into the algorithm. In fact, this parameter η is actually very related to the parameter that was chosen in MWA. Moreover, FTPL constitutes a *family* of algorithms: the exponential distribution is not the only choice we could have made. See the bibliographical notes for more details on possible alternative choices of distribution for the random perturbations.

6.2.1 Possible computational benefits

6.3. Why it works

We will now see, through a series of steps, why FTPL can ensure low regret. We begin with a critical simplification of Equation (6.4) and go through our two canonical examples to get some intuition for what FTPL really does.

6.3.1 A critical simplification: Random perturbation at the start

To illustrate our examples, we will go back to the binary sequence prediction paradigm to build intuition. Recall that for this problem, we measure our performance through *expected* loss, given by $H_T := \mathbb{E}\left[\sum_{t=1}^{T} \ell(\widehat{X}_t; X_t)\right]$, where the expectation is taken over the randomness in the predictions $\{\widehat{X}_t\}$. We first notice that FTPL is identical in expectation to running FTL on a *loss sequence* of length $(T+1)$ given by:

$$\{N_x, l_{1,x}, \ldots, l_{T,x}\} \text{ for each } x \in \{0,1\},$$

where for each $x \in \{0,1\}$, N_x is a single random variable (drawn according to the same distribution as what was earlier $N_{t,x}$). Note that this essentially yields updates given by

$$\widehat{X}_t = \arg\min_{x \in \{0,1\}} \left[L_{t-1,x} + N_x\right]. \tag{6.5}$$

Exercise 1 (Optional) *Verify that the predictions generated by Equation* (6.4) *and* (6.5) *would yield the exact same expected total loss, given by H_T, for any sequence $\{X_t\}_{t=1}^T$ that is fixed in advance and does not depend on the value of N_x. Use linearity of expectation (i.e. $\mathbb{E}[A+B] = \mathbb{E}[A] + \mathbb{E}[B]$), and a calculation of $\widehat{P}_t$ in terms of the cumulative distribution function (CDF) of an appropriate random variable.*

For the rest of this lecture and next lecture, we will work with this simplified version of FTPL with "one-shot" noise for the reason that it preserves expected performance[2]. The

2. There is a caveat that we will not discuss in lecture that can lead to differing performance between these variants of FTPL. The high-level reason is as follows: in the "one-shot" version of FTPL, the same

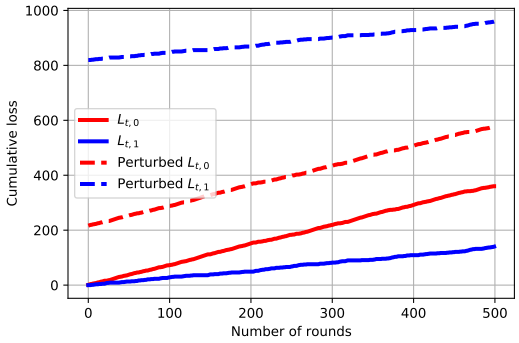


Figure 6.1: Visualizing the impact of a random perturbation on the evolution of the cumulative losses for an iid Bernoulli(0.7) sequence. Notice that the random perturbation's effect is so strong that it flips the identity of the best predictor from 1 to 0, which is a totally undesired outcome!

same intuition for how increased variance (randomization) in N_x's influences performance continues to hold: when $\eta \to \infty$, $N_x = 0$ and the algorithm reduces to FTL; but when $\eta \to 0$, the variance of N_x goes to ∞ and will dominate the algorithm's performance.

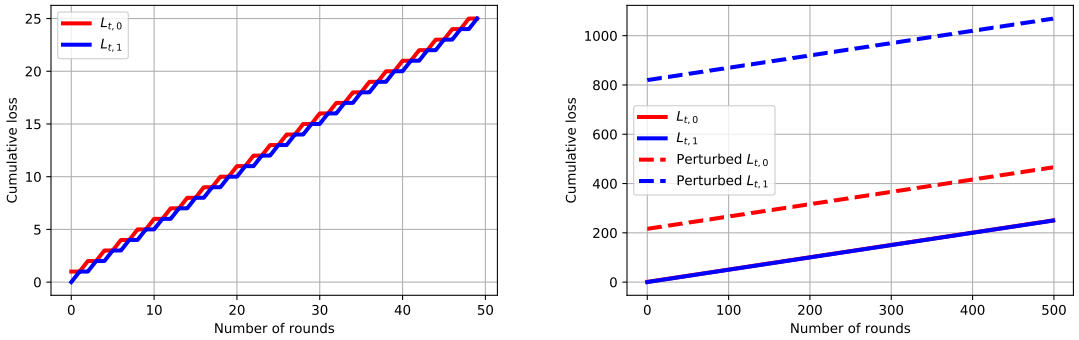
Just as we saw in the analysis of MWA, we neither want to over-exploit the true information present in the sequence, nor do we want to over-randomize. We will now visualize the impact of adding high-variance random perturbations $\{N_x\}_{x\in\{0,1}$ through our two canonical examples. In both examples, we will consider the total number of rounds $T = 500$.

Example 1 (iid Bernoulli(0.7) sequence) *We start with our canonical example of the iid Bernoulli sequence, and first plot (for a* typical *realization of the sequence) the cumulative losses of predicting* 0 *and* 1 *(respectively marked in solid red and solid blue) as a function of the number of rounds played. Notice that we consistently have $L_{t,1} < L_{t,0}$ (in fact, the gap between them grows with t). Therefore, after a "warm-start" period*[3]*,* 1 *is always the leader (and is also the best in hindsight). The solid lines in Figure 6.1 illustrate why FTL is such a good algorithm on this sequence: it will figure out very quickly that* 1 *is the correct letter to predict, and it will always predict it.*

Now, we evaluate what will happen when we add perturbations N_0 and N_1 to the respective loss sequences. In particular, we will consider a special realization of the random variables, N_0 and N_1, drawn from the exponential distribution with parameter $\eta = 0.01$, such that N_1 is really large but N_0 is really small. The dashed lines (in red and blue respectively) in Figure 6.1 plot the cumulative losses of predicting 0 *and* 1 that are perturbed by N_0 and

realization of the noise N_x influences the decisions made on all rounds. Therefore, an adversary that is allowed to adapt *after* the algorithm has started running would be able to intuit this value of N_x by observing past predictions made, and eventually exploit it. Such an adversary is more commonly called a "non-oblivious" adversary in the literature. In fact, the principal reason for needing the perturbations to be drawn independently at round t is precisely to avoid such exploitation by an adaptive adversary.

3. You characterized the length of this warm-start period in HW 1, Problem 2.



(a) Original. (b) Original + perturbed

Figure 6.2: Visualizing the impact of a random perturbation on the evolution of the cumulative losses for a sequence of alternating 1's and 0's. Notice that the random perturbation's effect is to separate the cumulative losses, which are intricately intertwined in the absence of the perturbation. Thus, the perturbation in this case has the positive effect of "stabilizing" the FTPL updates. Note the difference in y-axis scales between figure (a) and (b).

N_1 respectively. *The figure shows that, unfortunately, these perturbations are adversely large because they create a completely different outcome: now, because* 0 *is always leading, FTPL will always predict* 0 *on every round! This is the complete opposite of the outcome that we want: the original sequence had* 1 *as the leader on almost all rounds. Thus, for this realization of the random perturbation FTPL will, in fact, incur linear in* T *regret. It can be shown formally that the probability of this kind of an adverse realization occurring is actually significant (in particular, not close to* 0*) when the perturbations are of high variance. Therefore, the expected regret (over the randomness in the algorithm) can be shown to be linear.*

Example 2 (Alternating 1's and 0's) *We now consider our other canonical example of alternating* 1*'s and* 0*'s, which was observed to make FTL incur a loss of* 1 *on every single round. Similar to the previous example, Figure 6.2 plots* $L_{t,0}$ *and* $L_{t,1}$ *(in solid red and blue respectively) versus* t*. You can see from Figure 6.2a that the identity of the leader keeps changing from round to round; therefore, FTL is a highly unstable algorithm and (like we saw in class) will incur linear regret.*

We now consider the effect of adding perturbations N_0 *and* N_1*. As before, we will consider a special realization of the random variables,* N_0 *and* N_1*, drawn from the exponential distribution with parameter* $\eta = 0.01$*, such that* N_1 *is really large but* N_0 *is really small. You can see from Figure 6.2b that this disparate perturbation now has the positive effect of "separating out" the evolution of cumulative loss: in particular,* 0 *is always less than* 1 *in the perturbed loss sequence, and so FTPL will always predict* 0*. Thus, FTPL* ***stabilizes*** *the prediction updates. Since the best predictor in hindsight can be either* 0 *or* 1*, this stabilization*

ensures that the regret will be low (and would be low even in the complementary case where instead N_0 were really large, but N_1 were really small).

The two examples highlight the inherent tension in deciding how much perturbation to add to the algorithm. On one hand, Example 2 shows that if the extent of perturbation is insufficient, the updates remain unstable to an adversary (as in the original FTL algorithm). On the other hand, Example 1 shows that if the extent of perturbation is too large, it can change the entire outcome of prediction. Nevertheless, we will now show that we can effectively trade off these two effects and achieve the optimal $\mathcal{O}(\sqrt{T})$ regret guarantee, as below.

Theorem 1 *FTPL with the learning rate $\eta = \frac{1}{\sqrt{T}}$ achieves $R_T = \mathcal{O}(\sqrt{T})$ on any sequence.*

In the next lecture, we will prove Theorem 1 by bounding these two effects.

6.4. Additional references

- The initial version of FTPL and analysis was invented back in the 1950's by the statistician James Hannan (Hannan, 1957). The initial motivation was quite abstract and FTPL was designed to work for zero-sum repeated games with K actions per player.

- More recently, the benefits of FTPL have been explored through the lens of computational efficiency (as there are many real-world settings in which the number of choices K may be exponentially large, but there is special structure in the problem that makes the discrete optimization problem underlying FTL (and, in some cases, FTPL) more efficient than MWA. A more recent analysis of FTPL was conducted by Kalai and Vempala (2005) to work with general decision and action spaces, and show that computationally efficient versions of FTPL could be implemented for several decision problems of a combinatorial nature, including shortest-path problems and prediction using decision trees.

- A natural question in light of this is whether FTPL can be made efficient *whenever* FTL is efficient. This is commonly referred to as the *oracle-efficient online learning* problem. While oracle-efficient online learning is not achievable in the worst case (Hazan and Koren, 2016), recent work (Dudík et al., 2020) showed that this is achievable under certain structural constraints on the decision set.

References

Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. *Journal of the ACM (JACM)*, 67(5):1–57, 2020.

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

Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 128–141, 2016.

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