EE 290S/CS 194: ML for Sequential Decision Making Under Uncertainty
Fall 2018

Lecture 5: Multiplicative Weights II

Lecturers: Anant Sahai, Vidya Muthukumar *Scribe: Julian Chan*

In this lecture note, we will examine the Multiplicative Weights algorithm in more detail. In addition, we will provide a proof that Multiplicative Weights, with an appropriate choice of its learning parameter, can achieve $\mathcal{O}(\sqrt{T})$ regret on universal sequences. We then go on to show that no algorithm can do better than $\mathcal{O}(\sqrt{T})$ regret on universal sequences.

5.1 Recap

Recall our goal was to predict a $\{0,1\}^T$ sequence $X_1, X_2, ..., X_T$. In other words, $X_i \in \{0,1\}$ for $i = 1, ..., T$. We wanted to do well in prediction despite not having a model for the sequence. We saw last time that if we utilized a deterministic strategy like Follow the Leader (FTL), an adversarial environment can force us to incur full loss because it always has a "one-step ahead" advantage. So we needed to randomize.

We then proposed using a fair coin toss to get $\hat{X}$, meaning $\hat{X} \sim Bernoulli(\frac{1}{2})$, which would incur an expected loss of $\frac{T}{2}$ across T trials. In fact, the environment can force us to incur this loss by tossing its own fair coin. This strategy equates to minimizing the worst case loss:

$$\min_{\hat{X}} \max_{X} Loss(\hat{X}, X)$$

How could we do "*better*" with a randomized strategy, and by what standard are we using to say a strategy is "*better*" than another? We then defined this notion of **regret** with respect to a reference class of predictors; that is, at time T, we would look back at how well we performed relative to how the reference class would have performed (in our case, we considered a constant predictor, or one that predicts all 0's or all 1's). We want this difference to be as small as possible in hindsight; in other words, we don't want to regret our decisions.

So we introduced the Multiplicative Weights (MW) algorithm, parametrized by a learning rate η, which essentially is performing a softmax-like operation on the probability of predicting 0 or 1 at time t. How does this softmax operation assign a probability to each class? Let's first define some terms:

$$w_{t,k} = \frac{e^{-\eta L_{t-1,k}}}{Z_t}, k \in \{0,1\}$$

- $w_{t,k}$ = the probability of picking k at time t
- $L_{t-1,k} = \sum_{i=1}^{t-1} \mathbb{1}\{X_i \neq k\}$ **(Hamming Loss)**
- $Z_t = \sum_{k=0}^{1} e^{-\eta L_{t-1,k}}$ **(Normalization Factor)**

Why did we define $w_{t,k}$ as above? If the Hamming Loss $L_{t-1,k}$ is large, then we want $w_{t,k}$, the probability of picking k at time t, to be small. $e^{-\eta L_{t,k}}$ is a good choice because exponentiating the Hamming Loss results in a heavier penalty on incorrect predictions. However, this doesn't satisfy the property that probabilities must be between 0 and 1, so we need to normalize. $Z_t = \sum_{k=0}^{1} e^{-\eta L_{t-1,k}} = e^{-\eta L_{t-1,0}} + e^{-\eta L_{t-1,1}}$ is a suitable normalization factor since we only have 2 classes (0 and 1), so we have that:

$$w_{t,0} = \frac{e^{-\eta L_{t-1,0}}}{Z_t} \tag{5.1}$$

$$w_{t,1} = \frac{e^{-\eta L_{t-1,1}}}{Z_t} \tag{5.2}$$

5.2 Multiplicative Weights Algorithm Analysis

Great! We now have an algorithm to randomize our predictions. But as with any algorithm, we need to examine its performance in general. So now we pose the question: How do we know the Multiplicative Weights algorithm with an "exponential weight update" scheme works well? It turns out that using this algorithm, the regret scales on the order of $\mathcal{O}(\sqrt{T})$ with proper tuning of the learning rate η.

However, the Hamming Loss function doesn't seem like it has any relation to exponentials. So, rather than proving the regret scales on the order of $\mathcal{O}(\sqrt{T})$ with the Hamming Loss function, we will prove that this result holds for some other loss and see that if we perform well on this new loss function, we will perform well on the Hamming Loss function.

5.2.1 Defining a new loss function: Mix Loss

Claim 5.1 *The Multiplicative Weights algorithm, with an appropriate choice of its learning parameter η, yields regret on the order of $\mathcal{O}(\sqrt{T})$ where T is the time horizon of the $\{0,1\}$ sequence.*

Proof: *(Note: Below, we present some intuition as to how we came up with the new loss function. For more details on this loss function, see Freund and Schapire's (1997) paper: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.)*

So how do we come up with this new loss function? We want this intermediate goal loss function to behave as follows:

- For the magical rule that picks X^* (either all 0's or all 1's, whichever is the best one in hindsight) and gets loss L_T^*, we want L_T^* to be the loss
- Should look like $e^{-\eta L_{T,0}} + e^{-\eta L_{T,1}}$ for total loss under Multiplicative Weights

Without loss of generality, let's say $X^* = 1$. We expect that the loss that we get will look something like $e^{-\eta L_{T,0}} + e^{-\eta L_{T,1}}$. We need to do something to this term to get it to look just like $L_{T,1}$. Recall that if $X^* = 1$, this means that $e^{-\eta L_{T,0}} << e^{-\eta L_{T,1}}$ using the softmax idea. We add in an additional factor of $\frac{1}{2}$ to make the math work out later on, but the general form of the loss function is as follows:

$$L_{T,1} \approx -\frac{1}{\eta}\ln(\frac{1}{2}(\underbrace{e^{-\eta L_{T,0}} + e^{-\eta L_{T,1}}}_{\text{Looks like loss at time } T})) \tag{5.3}$$

This loss function must have some connection to the $w_{t,k}$ in Equations (5.1) and (5.2). Notice that Equation (5.3) looks like the loss at the last time T, except it is missing the denominator (normalizing factor at the previous time step $T-1$). So, let's multiply and divide by this factor:

$$L_{T,1} \approx -\frac{1}{\eta}\ln((\underbrace{e^{-\eta L_{T-1,0}} + e^{-\eta L_{T-1,1}}}_{\text{Looks like loss at time } T-1}) \cdot \frac{e^{-\eta L_{T,0}} + e^{-\eta L_{T,1}}}{e^{-\eta L_{T-1,0}} + e^{-\eta L_{T-1,1}}}) \tag{5.4}$$

But now, notice that the first term in Equation (5.4) looks like the loss at time $T-1$, and so similar to what we did before, we multiply and divide by the normalizing factor. As a result, this expression becomes a telescoping product:

$$L_{T,1} \approx -\frac{1}{\eta}\ln\left(\frac{e^{-\eta L_{1,0}}+e^{-\eta L_{1,1}}}{2}\cdot \ldots \frac{e^{-\eta L_{T-1,0}}+e^{-\eta L_{T-1,1}}}{e^{-\eta L_{T-2,0}}+e^{-\eta L_{T-2,1}}}\cdot \frac{e^{-\eta L_{T,0}}+e^{-\eta L_{T,1}}}{e^{-\eta L_{T-1,0}}+e^{-\eta L_{T-1,1}}}\right) \tag{5.5}$$

$$= \sum_{t=1}^{T} -\frac{1}{\eta}\ln\left(\frac{e^{-\eta L_{t,0}}+e^{-\eta L_{t,1}}}{e^{-\eta L_{t-1,0}}+e^{-\eta L_{t-1,1}}}\right) \tag{5.6}$$

$$= \sum_{t=1}^{T} \underbrace{-\frac{1}{\eta}\ln\left(w_{t,0}\cdot e^{-\eta \mathbb{1}\{X_t\neq 0\}} + w_{t,1}\cdot e^{-\eta \mathbb{1}\{X_t\neq 1\}}\right)}_{m_t} \tag{5.7}$$

Let's define a new loss term $m_t = -\frac{1}{\eta}\ln(w_{t,0}\cdot e^{-\eta\mathbb{1}\{X_t\neq 0\}} + w_{t,1}\cdot e^{-\eta\mathbb{1}\{X_t\neq 1\}})$. This means:

$$M_T = -\frac{1}{\eta}\ln\left(\frac{1}{2}(e^{-\eta L_{T,0}}+e^{-\eta L_{T,1}})\right) = \sum_{i=1}^{T} m_t \tag{5.8}$$

This new loss function M_T is called the Mix Loss.

5.2.2 Incorporating Mix Loss into the Regret

Recall that we defined regret, R_T, as the difference between our performance under Hamming Loss, H_T, and the best performance by a reference class in hindsight, L_T^*. That is:

$$R_T = H_T - L_T^* = (M_T - L_T^*) + (H_T - M_T) \tag{5.9}$$

where in the last equality, we simply added and subtract the Mix Loss term, M_T. This way, we can characterize how the regret behaves using this new loss function we defined.

5.2.3 Gap Between M_T and L_T^*

From Equation (5.8), we have:

$$M_T = -\frac{1}{\eta}\ln\left(\frac{1}{2}(e^{-\eta L_T^*}+e^{-\eta \bar{L_T^*}})\right) \tag{5.10}$$

Instead of considering the loss incurred by picking 0 and similarly for picking 1, we consider the loss incurred by picking the optimal choice in hindsight, hence replacing the $L_{t,k}$ with L_T^* and $\bar{L_T^*}$.

We want to bound M_T in terms of L_T^*, so we need to find the smallest upper bound for M_T. Notice that in Equation (5.10), we have the $e^{-\eta \bar{L_T^*}}$ term that we do not like (because it is not in terms of L_T^*). Since the exponential function is strictly positive and we are trying to minimize the bound, we can just drop that term. So, we have:

$$M_T \leq -\frac{1}{\eta}\ln\left(\frac{1}{2}(e^{-\eta L_T^*})\right) = L_T^* + \frac{\ln 2}{\eta} \tag{5.11}$$

$$\implies M_T - L_T^* \leq \frac{\ln 2}{\eta} \tag{5.12}$$

This tells us that for the Mix Loss to be close to the optimal loss, we should increase η. In fact, pushing $\eta \to \infty$ will make the Mix Loss exactly equal to the optimal loss. This also means that we are assigning a heavy weight to the loss function, meaning that incorrect predictions are penalized even more. Essentially, with $\eta \to \infty$, this algorithm will perform exactly like Follow the Leader (FTL).

5.2.4 Gap Between H_T and M_T (Mixability Gap)

$$H_T - M_T = \sum_{t=1}^{T}(w_{t,0}\cdot\mathbb{1}(X_t \neq 0) + w_{t,1}\cdot\mathbb{1}(X_t \neq 1)) - (-\frac{1}{\eta}\ln(w_{t,0}\cdot e^{-\eta\mathbb{1}\{X_t\neq 0\}} + w_{t,1}\cdot e^{-\eta\mathbb{1}\{X_t\neq 1\}})) \tag{5.13}$$

Equation (5.13) is also known as the **mixability gap**, which is the difference between the linear loss H_T and the η-mix loss M_T. Notice in Equation (5.13) that there are 2 terms where we have a sum of probability times some quantity. This should remind us of the concept of expectation of a random variable. So let's define a new random variable that will view these terms as expectations:

$$\ell_{Y,t} = \begin{cases} X_t & \text{if } Y_t = 0 \\ 1 - X_t & \text{if } Y_t = 1 \end{cases} \tag{5.14}$$

where Y_t is our prediction at time t according to $w_{t,k}$. In other words, Y_t is a Bernoulli random variable as follows:

$$Y_t \sim Bernoulli(w_{t,1})$$

$$Y_t = \begin{cases} 1 & \text{with probability } w_{t,1} \\ 0 & \text{with probability } w_{t,0} \end{cases} \tag{5.15}$$

This means that:

$$H_T - M_T = \sum_{t=1}^{T}\mathbb{E}[\ell_{Y,t}] + \frac{1}{\eta}\underbrace{\ln \underbrace{\mathbb{E}[e^{-\eta\ell_{Y,t}}]}_{\text{MGF for } \ell_Y}}_{\text{CGF for } \ell_Y} \tag{5.16}$$

Let's examine a part of the second term in Equation (5.16), $\ln\mathbb{E}[e^{-\eta\ell_{Y,t}}]$. Recall that this is a Cumulant Generating Function (CGF) for ℓ_Y. The CGF is essentially the log of the Moment Generating Function (MGF), where moments of the underlying probability distribution can be extracted by differentiating the MGF. Recall that we can write the CGF as a Taylor expansion:

$$\ln\mathbb{E}[e^{-\eta\ell_Y}] \approx -\eta\mathbb{E}[\ell_Y] + \frac{\eta^2}{2}Var(\ell_Y) + ... \tag{5.17}$$

To bound the gap $H_T - M_T$, we first need to bound this term. Since the CGF is the log of the MGF, we can apply Hoeffding's Lemma, which is an inequality that bounds the MGF of any bounded random variable.

Lemma 5.2 (Hoeffding's Lemma) *Suppose X is a bounded random variable $a \leq X \leq b$. Then for every real s, we know that:*

$$\ln\mathbb{E}[e^{sX}] \leq s\mathbb{E}[X] + \frac{s^2(b-a)^2}{8}$$

Proof: Let's first consider the zero-mean random variable $\tilde{X}$ such that $X = \tilde{X} + \mathbb{E}[X]$. So, we have:

$$\begin{aligned} \ln\mathbb{E}[e^{sX}] &= \ln\mathbb{E}[e^{s(\tilde{X}+\mathbb{E}[X])}] \\ &= \ln\mathbb{E}[e^{s\tilde{X}}e^{s\mathbb{E}[X]}] \\ &= \ln\mathbb{E}[e^{s\tilde{X}}]e^{s\mathbb{E}[X]} \\ &= \ln\mathbb{E}[e^{s\tilde{X}}] + \ln e^{s\mathbb{E}[X]} \\ &= \ln\mathbb{E}[e^{s\tilde{X}}] + s\mathbb{E}[X] \end{aligned}$$

Thus, we just need to show that $\ln\mathbb{E}[e^{s\tilde{X}}] \leq \frac{s^2(b-a)^2}{8}$. Notice that even though we demeaned X, the width of the bounds on $\tilde{X}$, $(b-a)$, is still the same. So, we can write $\tilde{X}$ as a convex combination of the edges of its bounds: $\tilde{X} = tb + (1-t)a$ where $t = \frac{\tilde{X}-a}{b-a}$. Since the exponential function is convex, we can write:

$$e^{t\tilde{X}} \leq te^{tb} + (1-t)e^{ta} = \frac{\tilde{X}-a}{b-a}e^{tb} + \frac{b-\tilde{X}}{b-a}e^{ta}$$

Taking the expectation on both sides of the equation, and knowing that $\tilde{X}$ is a zero-mean random variable ($\implies \mathbb{E}[\tilde{X}] = 0$):

$$\mathbb{E}[e^{t\tilde{X}}] \leq -\frac{a}{b-a}e^{tb} + \frac{b}{b-a}e^{ta} = e^{f(z)}$$

where $z = t(b-a)$, $f(z) = -\gamma z + \log(1-\gamma+\gamma e^z)$, and $\gamma = -\frac{a}{b-a}$. We know that:

$$f(0) = \frac{df}{dz}|_{z=0} = f'(0) = 0$$

$$\frac{d^2f}{dz^2} \leq \frac{1}{4}, \forall z > 0$$

Using Taylor's theorem, we know that there exists an $\alpha \in (0, z)$ such that:

$$f(z) = f(0) + zf'(0) + \frac{z^2}{2}f''(\alpha) = \frac{z^2}{2}f''(\alpha) \leq \frac{z^2}{8} = \frac{t^2(b-a)^2}{8}$$

Therefore, we conclude that:

$$\ln \mathbb{E}[e^{t\tilde{X}}] \leq f(z) \leq \frac{t^2(b-a)^2}{8}$$

■

Thus, if the random variable $\ell_Y \in [a, b]$, then from Hoeffding's Lemma, we have:

$$\ln \mathbb{E}[e^{-\eta\ell_Y}] \leq -\eta\mathbb{E}[\ell_Y] + \frac{\eta^2(b-a)^2}{8} \tag{5.18}$$

Now, let's revisit Equation (5.16). Applying the bound we found in Equation (5.18), we have that:

$$H_T - M_T = \sum_{t=1}^{T} \mathbb{E}[\ell_{Y,t}] + \frac{1}{\eta}\ln \mathbb{E}[e^{-\eta\ell_{Y,t}}]$$

$$\leq \sum_{t=1}^{T} \mathbb{E}[\ell_{Y,t}] + \frac{1}{\eta}(-\eta\mathbb{E}[\ell_{Y,t}] + \frac{\eta^2(1-0)^2}{8}) \tag{5.19}$$

$$= \sum_{t=1}^{T} \frac{\eta}{8} = \frac{T\eta}{8} \tag{5.20}$$

So, we conclude that:

$$H_T - M_T \leq \frac{T\eta}{8} \tag{5.21}$$

5.2.5 Choosing an appropriate η

Looking at Equation (5.9), we have that:

$$R_T = H_T - L_T^* = (M_T - L_T^*) + (H_T - M_T)$$

Incorporating the bounds we found in Equations (5.12) and (5.21), we have that:

$$R_T \leq \frac{\ln 2}{\eta} + \frac{T\eta}{8} \tag{5.22}$$

Recall our goal was to get our regret to grow sublinearly in T by properly tuning the learning rate η. From Equation (5.22), we can see that a suitable choice for η would be $\eta = \frac{1}{\sqrt{T}}$. By using this learning rate, we have that:

$$R_T \leq \frac{\ln 2}{\frac{1}{\sqrt{T}}} + \frac{T \cdot \frac{1}{\sqrt{T}}}{8} = \sqrt{T} \cdot (\ln 2 + \frac{1}{8}) \tag{5.23}$$

■

Let's see some examples of the effect of various learning rates η on different sequences.

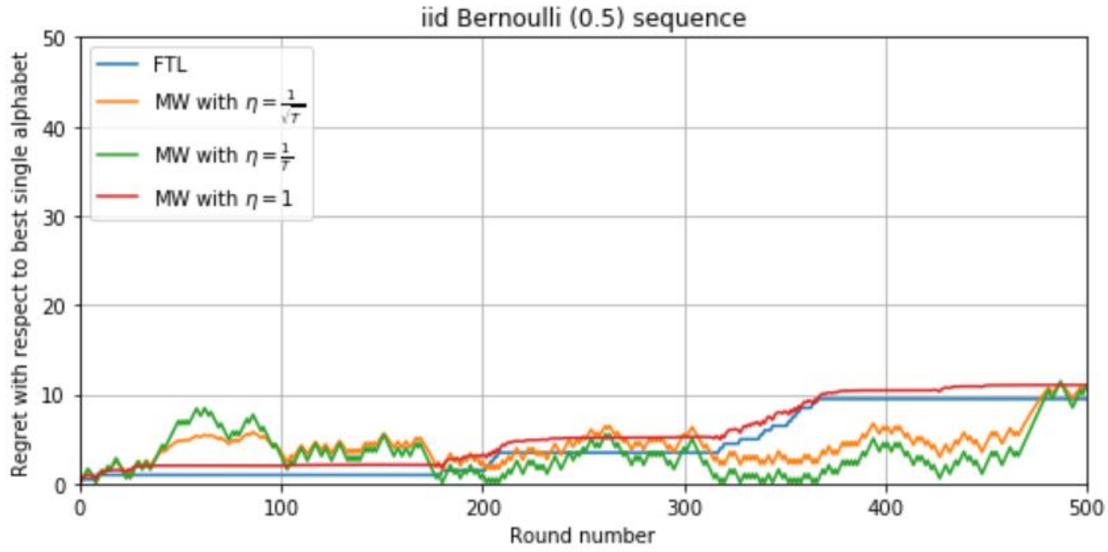

Figure 5.1: Regret using FTL and MW using learning rates $\eta = \frac{1}{\sqrt{T}}$, $\frac{1}{T}$, and 1 on a sequence generated from a $Bernoulli(0.5)$ distribution.

The sequence in this example is generated by flipping a fair coin. Notice that in this example, MW is being misled by the sequence, as depicted by the regret growing and shrinking over time. This is because if the environment is flipping a fair coin independently over time, then there is no "pattern" to learn. As a result, the probabilities that MW updates in each iteration aren't really modelling the data distribution all that well. Another thing to note is that FTL and MW with a constant learning rate seem to behave similar to each other, and MW with adaptive learning rates seem to behave similar to each other as well.

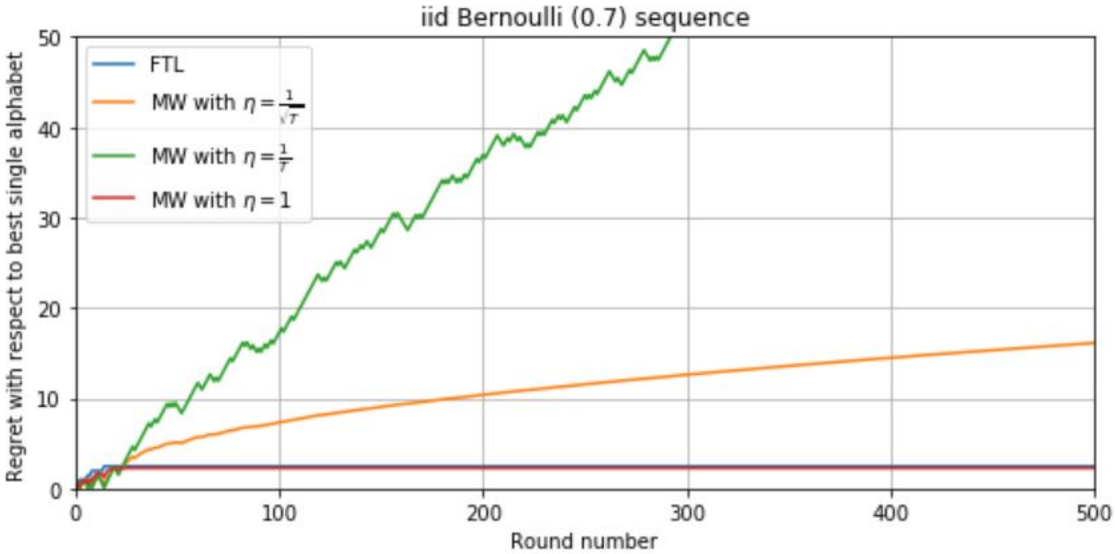

Figure 5.2: Regret using FTL and MW using learning rates $\eta = \frac{1}{\sqrt{T}}$, $\frac{1}{T}$, and 1 on a sequence generated from a $Bernoulli(0.7)$ distribution.

The sequence in this example is generated by flipping a biased coin with $\mathbb{P}(Heads) = 0.7$. We see here that both FTL and MW in general yields sublinear regret, except in the case where $\eta = \frac{1}{T}$ (the green curve). In fact, it is evident here that for this specific sequence, MW with learning rate $\eta = \frac{1}{T}$ is randomizing too much, resulting in its poor performance.

The sequences in Figures 5.3 and 5.4 are 1-periodic and 2-periodic sequences, respectively. We can think of these as adversarial sequences that are trying to fool our algorithm into thinking one choice is better than the other. Let's examine the 1-periodic sequence; the 2-periodic sequence explanation will be similar.

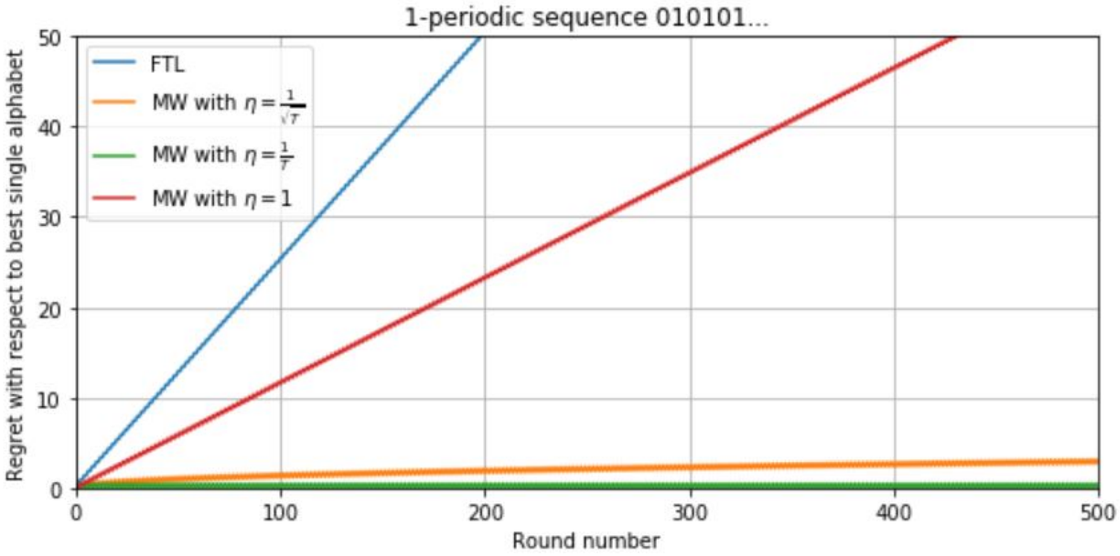

Figure 5.3: Regret using FTL and MW using learning rates $\eta = \frac{1}{\sqrt{T}}$, $\frac{1}{T}$, and 1 on a 1-periodic (alternating) sequence.

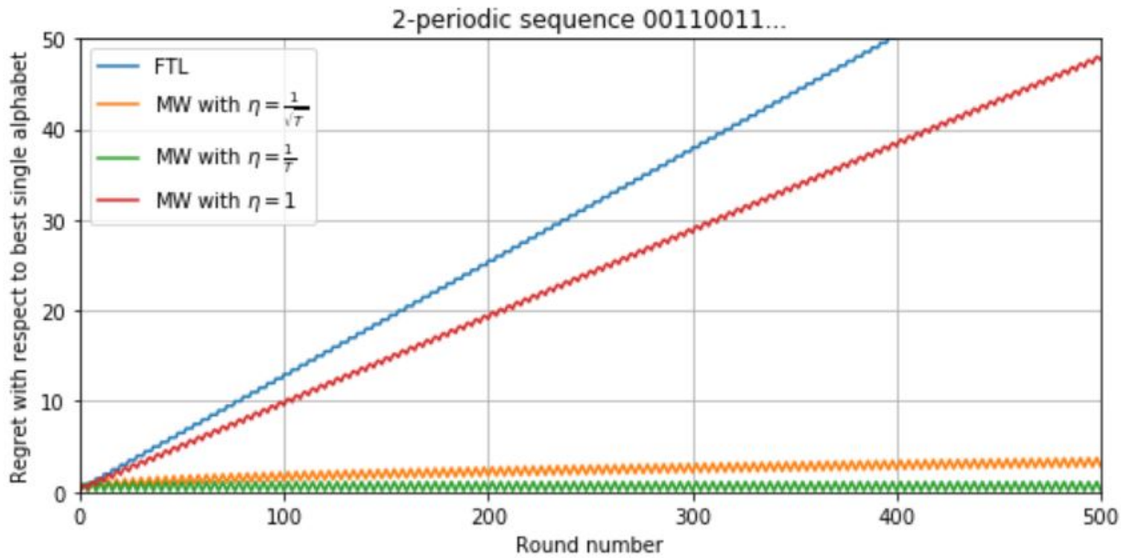

Figure 5.4: Regret using FTL and MW using learning rates $\eta = \frac{1}{\sqrt{T}}$, $\frac{1}{T}$, and 1 on a 2-periodic (twice alternating) sequence.

In the case of FTL, it actually performs the worst because it initially sees a 0, so the majority votes for 0 in the next time step. However, the sequence then outputs a 1, in which case FTL will predict 1 in the next time step. As this process goes on, we incur full loss and get no predictions correct (as explained previously where deterministic algorithms are prone to exploitation by adversarial environments). However, had we only stuck with one option, we notice that FTL could've predicted half of the sequence correctly. Thus, the regret associated with using FTL for this sequence grows linearly over time.

However, incorporating a randomized component into our algorithm (like MW) actually still gives us sublinear regret. A key point to note is that even though MW introduces randomization, using a fixed learning rate still doesn't necessarily give us sublinear regret (see the red curve). This is because with a fixed learning rate, we are penalizing the same amount for short and long sequences (essentially placing an equal weight on each time step). In fact, MW with a constant learning rate behaves like FTL in the long run. However, an adaptive learning rate that is a function of the sequence length does give us sublinear regret because it induces recency bias, giving more importance to recent samples than samples that were drawn a long time ago.

To summarize our observations from the four examples above, deterministic algorithms like FTL certainly perform better than MW on specific sequences like purely random sequences, but when the sequence is generated by an adversary, we find that MW performs better than FTL. In fact, from the proof above, we showed that MW actually yields sublinear regret on universal sequences (whether they are random or generated by an adversary) because the proof made no assumptions about the sequence.

5.2.6 Can we do better?

At this point, a natural question to ask is: can we do better than $\mathcal{O}(\sqrt{T})$ on this type of sequence by picking a another η or even using a different algorithm? First notice that this bound holds universally for every sequence $\{X\}_{t=1}^{T}$ because we never used anything about the sequence in our proof. In addition, notice in Equation (5.22) that there is a sort of push-pull effect happening. If we increase η too much, sure the first term will decrease, but the second term will blow up. Conversely, if we decrease η too much, the first term will blow up while the second term decreases. A general rule of thumb to balance two terms where one is increasing and the other is decreasing with respect to some parameter is to pick a parameter that brings the two terms to the same order of magnitude. Certainly, we can do better than $\mathcal{O}(\sqrt{T})$ by using calculus to optimize to find the optimal η.

However, now we ask the question: is it possible for a single algorithm to do better than $\mathcal{O}(\sqrt{T})$ regret on arbitrary sequences? The answer is NO. This doesn't mean that for a specific sequence, an algorithm cannot do better than $\mathcal{O}(\sqrt{T})$ regret. Consider the sequence $\{0\}_{t=1}^{T}$. An algorithm that always picks 0 will achieve sublinear regret; in fact, it will have no regret because it makes the same prediction as the best choice in hindsight.

Claim 5.3 *A single algorithm cannot universally do better than $\mathcal{O}(\sqrt{T})$ regret on arbitrary sequences.*

Proof: Suppose for $i \in \{1, 2, ..., T\}$, $X_i \sim Bernoulli(\frac{1}{2})$ i.i.d. This is essentially saying that we have a uniform distribution over all length T sequences. We know that if the environment generates a sequence of length T according to a $Bernoulli(\frac{1}{2})$ distribution, it can force us to incur an expected total loss of $\frac{T}{2}$, regardless of which algorithm we use.

Now we ask ourselves how can the best choice in hindsight be better than $\frac{T}{2}$? If we have a sequence of fair coin tosses, our first intuition is that if the probability of getting 0 and 1 are the same, then about half of the outcomes will be 0 and about half will be 1. So at first glance, it seems as though it doesn't matter if we pick 0 or 1. Either way, our expected loss will be $\frac{T}{2}$. Another way of looking at the best predictor in hindsight is to see which class has more net occurrences, and just pick that class all the time. If there were more 0s than 1s that occurred, the best predictor in hindsight should pick 0. So it may be the case that the dominant term in our loss is $\frac{T}{2}$, but how about the next term? What is the fluctuation like?

To understand this, let's assume we have a sequence:

$$X_1, X_2, ..., X_T$$

We want to see whether there were more 0s or 1s (regardless of which was more frequent) to determine the advantage that hindsight has, so we care about the quantity |# of 1s - # of 0s|:

$$\left| \sum_{t=1}^{T} [\mathbb{1}(X_t = 1) - \mathbb{1}(X_t = 0)] \right| = \left| \sum_{t=1}^{T} R_t \right|$$

where R_t are i.i.d. Rademacher random variables:

$$R_t = \begin{cases} +1 & \text{if } X_t = 1 \\ -1 & \text{if } X_t = 0 \end{cases}$$

The size of this quantity tells us how much of an advantage hindsight will have. As T tends to infinity, we can invoke the Central Limit Theorem and claim:

$$\lim_{T \to \infty} \frac{1}{\sqrt{T}} \sum_{t=1}^{T} R_t \sim \mathcal{N}(0, 1)$$

Lemma 5.4 *If $S \sim \mathcal{N}(0, \sigma^2)$, then $Y = |S|$ follows a half-normal distribution. The PDF of Y is:*

$$f_Y(y; \sigma) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} e^{-\frac{y^2}{2\sigma^2}}$$

and the expectation is:

$$\mathbb{E}[Y] = \frac{\sigma\sqrt{2}}{\sqrt{\pi}}$$

So, the expected advantage that hindsight will have (net difference in the number of 0s and 1s) will be like $\sqrt{T} \cdot \sqrt{\frac{2}{\pi}}$. Recall that we can write the expected regret as:

$$\mathbb{E}[Regret] = E[Loss] - E[Best\_loss\_in\_hindsight]$$

where $\mathbb{E}[Loss] = \frac{T}{2}$ by virtue of the environment forcing us to incur this loss by flipping a fair coin to generate the sequence and $E[Best\_loss\_in\_hindsight]$ is the net advantage that the best predictor (from the reference class of constant predictors) in hindsight has. Thus, we have:

$$\mathbb{E}[Regret] = \underbrace{\frac{T}{2}}_{\text{Expected Loss}} - \underbrace{\sqrt{T} \cdot \sqrt{\frac{2}{\pi}}}_{\text{Net advantage in hindsight}}$$

The reason why we cannot do better than $\mathcal{O}(\sqrt{T})$ regret because the best in hindsight has a $\sqrt{T}$ advantage over what we can reasonably expect to get. ■

We can actually think of the $\sqrt{T} \cdot \sqrt{\frac{2}{\pi}}$ term as an overfitting term or unfair advantage because it is essentially looking at information that isn't supposed to be available to it. No algorithm can ever attain this advantage and there is no way we can avoid it. It is in fact our definition of regret is allowing this advantage to happen.

5.3 References

1. De Rooij, Steven, et al. "Follow the Leader If You Can, Hedge If You Must." Journal of Machine Learning Research, vol. 15, no. 1, Jan. 2014, pp. 1281-1316.

2. Freund, Yoav, and Robert E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." Journal of Computer and System Sciences, vol. 55, no. 1, Aug. 1997, pp. 119-139., doi:10.1006/jcss.1997.1504.

3. "Tight Bounds for Specific Losses." Prediction, Learning, and Games, by Nicolo Cesa-Bianchi and Lugosi Gabor, Cambridge University Press, 2006, pp. 40-66.