EE194/290S: ML for Sequential Decision under Uncertainty **Fall 2018**

Lecture 4: Multiplicative Weights I

Lecturer: Anant Sahai, Vidya Muthukumar *Scribes: Brian Hung, Michael X. Xu*

In this lecture, we will introduce the difficulties of making predictions in unknown online environments. To that end, we explain why randomization plays an important role in such environments, provide a formal definition of the term "regret" in the setting of sequential prediction problems, and finally introduce the multiplicative weights algorithm.

4.1 Sequential Decisions

We start off first with a familiar problem: prediction. In the previous lectures, we reviewed classical machine learning methods and focused mainly on prediction problems where a dataset is provided to us in advance.

However, we will now explore the case where we don't get the entire dataset all at once, but instead we receive some data points one by one each round[1]; and in between these rounds, we choose to rely on the previously revealed information as input and make predictions. This variant, where we make best guesses of a sequence unknown to us, is called a **sequential prediction problem**:

1. Label arrives one piece at a time.
2. We make a prediction of the next label based on preceding labels.

In the context of classical ML, we can choose to formalize the "dataset" after the t-th round as $\mathcal{D}_t = \{(x_i, \hat{x}_i)\}_{i=0}^t$, where x_i denotes the label received at time i, and $\hat{x}_i$ denotes our prediction to the i-th signal.[2]

4.1.1 Warmup: Prediction with Bernoulli Variables

Here, let us start with a warm up illustration using the Bernoulli distribution for simplification.[3] Suppose we observe the i.i.d Bernoulli draws $X_1, X_2, \ldots, X_t, \ldots$ in a stream, where we have $X_t \in \{0, 1\}$ and $P[X_t = 1] = p \in [0, 1]$. At each time instance t, having observed X_1, X_2, ..., X_{t-1}, we are tasked with making the t-th prediction.

To formalize our goal of prediction:

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} \ell_{\text{HAMMING}}(X_t, \hat{X}_t) \\ \text{where} \quad & X_t \overset{\text{i.i.d}}{\sim} \text{Bernoulli}(p) \\ & \hat{X}_t = f_{\text{predict}}(X_1, X_2, ..., X_{t-1}) = f_{\text{predict}}(X^{(t-1)}) \end{aligned}$$

Here we incur the zero-one Hamming loss in the objective function:

$$\ell_{\text{HAMMING}}(X_t, \hat{X}_t) = \begin{cases} 0 & \text{if } X_t = \hat{X}_t \\ 1 & \text{if } X_t \neq \hat{X}_t \end{cases}$$

Before moving on and talking about how to find an optimal f which minimizes the loss function, we'll make an additional assumption that, as in most sequential prediction problems, this Bernoulli variable p will be unknown to us.

Note that if p is known to us, it is trivial to construct our universal prediction function f to be:

[1]As we will see later, these types of problems are solved or approximated via an online algorithm.

[2]In future lectures, we will explore the case of online supervised learning, where we receive both an input x_t and a label y_t.

[3]Remember that in most real online environments, one do not have a probabilistic model of the input sequence before hand.

$$f^*_{predict}(X^{(t)}_{(1)}) = \begin{cases} 0 & \text{if } p < 0.5 \\ 1 & \text{if } p \geq 0.5 \end{cases} \tag{1}$$

(In the case $p = 0.5$, we break ties in favor of 1.

4.2 Randomization

Now suppose we do not have prior information on the value of p, how can we approach picking f?

A direct intuition would be using the Maximum Likelihood Estimation (MLE) to estimate p and then rely on rule (1) for prediction. As it turns out, this would lead to picking the majority vote from the sequence (i.e., selecting the class that yields the least loss over all past rounds). This strategy in sequential prediction is called *follow-the-leader* (FTL), and in this case we could write our prediction function as:

$$f^*_{predict}(X^{(t-1)}_{(1)}) = \mathbf{I}\{\bar{X}^{(t-1)}_{(1)} \geq 0.5\} \tag{2}$$

where $\bar{X}^{(t-1)}_{(1)}$ is the running average of $X_1, X_2, \ldots, X_{t-1}$, that is,

$$\bar{X}^{(t-1)}_{(1)} := \frac{1}{t-1}\sum_{i=1}^{t-1} X_i.$$

Here we break ties by choosing $\hat{X}_t = 1$ when $\bar{X}^{(t-1)}_{(1)} = 0.5$.

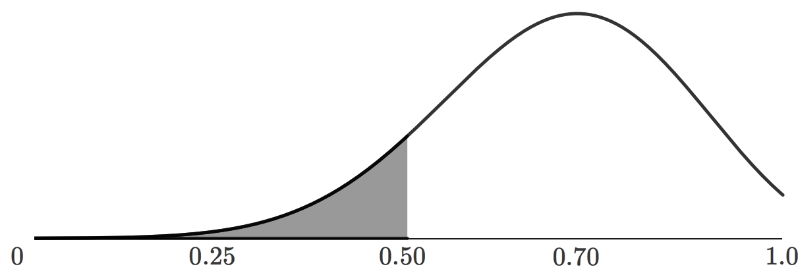
When the data is stochastic (i.e. the losses are independent and identically distributed), FTL is the natural approach to online prediction, and it works well. This is the setting familiar to us in statistical learning. Formally, for almost all sequences (under the probabilistic model), the average number of mistakes is (asymptotically) no more than the smallest between the proportion of zeros and proportion of ones in the sequence. We now show that this probability of deviation decays exponentially in proportion to T. Without loss of generality, assume the sequence is iid Bernoulli with $p \geq 0.5$. Then, we have

$$\begin{aligned} \Pr(\text{FTL is wrong at time } t) &= \Pr(\bar{x}^{(t)}_{(1)} < 0.5) & (3) \\ &= \Pr(\sum_{i=0}^{t} x_i - tp < 0.5 - tp) \\ &\leq e^{-\lambda(p-0.5)t}\mathbb{E}[e^{\lambda X(p-0.5)t}] & \text{for some } \lambda > 0 \\ &\leq e^{-\frac{(p-0.5)^2 t}{8}}, \end{aligned}$$

where the first inequality applies the Chernoff bound[4] and the second inequality applies Hoeffding's inequality on bounded random variables[5], then minimizes the bound over $\lambda > 0$.

As a more concrete example, let `p` $= 0.70$. The probability of FTL making a "wrong" prediction equals the shaded area[6].

[4]`https://en.wikipedia.org/wiki/Chernoff_bound#Example`.
[5]This was proved in HW 1!
[6]A further visualization in your browser: `https://www.desmos.com/calculator/ecaqw5mz9i`.



One problem that arises with FTL is that it breaks down badly when the sequence is antagonistic, that is generated by an adversarial environment. Considering the following example:

t	0	1	2	3	$\cdots$	$T-1$	T
x_t	0	1	0	1	$\cdots$	0	1
$\hat{x}_t$	1	0	1	0	$\cdots$	1	0

On this example, FTL will predict 0 in odd rounds *and* 1 in even rounds (by the tie-breaking rule). Thus, it incurs loss 1 in every round. The total loss incurred is $\sum_{t=1}^T \ell_{\text{HAMMING}}(X_t, \hat{X}_t) = T$.

It turns out this catastrophic loss is not limited to only FTL, but more generally to any strategy which is deterministic.

Lemma 4.1 *For any deterministic prediction strategy $\{\phi_t : \{0,1\}^t \to \{0,1\}\}_{t=0}^{T-1}$, there exists an adversarially generated input sequence $\{x_t\}_{t=1}^T$ such that the Hamming loss incurred by the prediction strategy is linear in T.*

Proof: By definition, the deterministic strategy will predict $\hat{x}_t = \phi_t(x_1, \ldots, x_{t-1})$ for all $t = 1, \ldots, T$. The adversary will then choose the sequence $\{x_t = 1 - \hat{x}_t\}_{t=1}^T$. This gives us

$$sum_{t=1}^T \ell_{\text{HAMMING}}(x_t, \phi_t(x_1, \ldots, x_{t-1})) = \sum_{t=1}^T \mathbf{I}\{\hat{x}_t \neq x_t\} = T.$$

■

A question then arises, *How can we find a strategy that does better than loss of T?* Here's a naive approach: We avoid deterministic play by flipping an unbiased (private) coin each round and base our prediction $\hat{X}_t$ on that. That is, $\hat{X}_t$ is Bernoulli with probability 1/2. The expected loss is then

$$\mathbb{E}[\sum_{t=1}^T \ell_{\text{HAMMING}}(x_t, \hat{X}_t)] = \sum_{t=1}^T \mathbb{E}[\ell_{\text{HAMMING}}(x_t, \hat{X}_t)] = \sum_{t=1}^T \mathbb{E}[1\{x_t \neq \hat{X}_t\}] = \frac{1}{2}T\ .$$

Observe that this strategy of *pure guessing* will incur loss $T/2$ regardless of the value of the true sequence $X_1, \ldots, X_T$. This is an attractive property because we have completely removed our dependence of the loss on the adversary!

Pure guessing works well on examples like the sequence of alternating 0's and 1's (on which FTL performed poorly) – but is unsatisfactory on more predictable sequences where we would ideally like to exploit the predictable pattern. As an extreme example, let's pretend that the environment produces the sequence $x_t = 0$ for all $t = 1, \ldots, T$. A sample prediction $\hat{X}_1, \ldots, \hat{X}_T$ of the pure guessing strategy is given below:

t	0	1	2	3	$\cdots$	$T-1$	T
x_t	0	0	0	0	$\cdots$	0	0
$\hat{x}_t$	0	1	1	0	$\cdots$	0	1

$\mathbb{E}[\sum_{t=1}^T \ell_{\text{HAMMING}}(x_t, \hat{x}_t)] = \sum_{t=1}^T \mathbb{E}[\mathbf{I}\{x_t \neq \hat{x}_t\}] = \frac{1}{2}T$.

In hindsight, an algorithm like FTL, that would have incurred a total loss of 0, would have been a much better choice!

The key is that our environment is unknown: it may be unpredictable, but it also may be friendly/stochastic. We want an algorithm that minimizes the *worst-case* loss we could have encountered, where this worst case is itself a function of the approach we choose. We can formalize this below:

$$\min_{\hat{X}_1}\max_{X_1}\min_{\hat{X}_2}\max_{X_2}\ldots\min_{\hat{X}_T}\max_{X_T}\sum_{t=1}^{T}\ell_{\text{HAMMING}}(X_t,\hat{X}_t).$$

We cannot expect to get every single prediction right in a worst-case *and* online environment. We next look at reasonable benchmarks for what we can hope to achieve, based on what we could have done *in hindsight.*

4.3 Regret

In words, regret can best be described as what you wished you would've done in hindsight. The key phrase here is "in hindsight": we compare a candidate *sequential/online learning approach* with the best approach we might have taken in hindsight had we seen the entire sequence at once.

Definition (Regret) Fix the loss vectors $\ell(X_1,\cdot\,),\ldots,\ell(X_T,\cdot\,)$. The **regret** of a sequence of predictions $\hat{X}_1,\hat{X}_2,\ldots,\hat{X}_T$ is:

$$R(T)=\underbrace{\sum_{t=1}^{T}\ell(X_t,\hat{X}_t)}_{\text{our algorithm}}-\underbrace{\min_{X^\star\in\{0,1\}}\sum_{t=1}^{T}\ell(X_i,X^\star)}_{\text{best fixed prediction}}.$$

To formalize the definition of "regret", we require a benchmark, a reference class of predictors X as the sequence of actions our strategy could've taken in hindsight.

Note that our benchmark (for now) is the set of *constant predictors*, i.e. predicting $X^\star=1$ or $X^\star=0$ in every round. We'll discuss more ambitious benchmarks in later lectures.

Note that $\hat{X}_t$, the prediction made at round t, is a random quantity as it depends on the randomness in our algorithm. This also implies that the regret R(T) is also a random variable. So in general, we will talk about the expected regret $\mathbb{E}[R(T)]$.

Recall in a stochastic environment, FTL converge exponentially to the optimum strategy, and therefore incurred a *constant* regret. Formally, when X_t i.i.d Ber(p), FTL will only disagree with the best action in X for a constant number of times in the beginning and thus achieve an expected regret of constant:

$$\begin{aligned}\mathbb{E}[R(T)]&=\sum_{t=1}^{T}\mathbb{E}[\ell(x_t,\hat{x}_t)]-\min_{x^\star\in X}\mathbb{E}[\sum_{t=1}^{T}\ell(x_t,x^\star)]\\&=\min_{x^\star\in X}\sum_{t=1}^{T}\mathbb{E}[\ell(x_t,\hat{x}_t)-\ell(x_t,x^\star)]\\&\sim\mathcal{O}(1)\end{aligned}$$

For the example that we described as *adversarial* to FTL, i.e. the sequence of alternating 0's and 1's, the regret incurred by FTL is

$$\begin{aligned}R(T)&=\sum_{t=1}^{T}\ell(x_t,\hat{x}_t)-\min_{x^\star\in X}\sum_{t=1}^{T}\ell(x_i,x^\star)\\&=T-\frac{1}{2}T=\frac{1}{2}T\end{aligned}$$

With regret, the adversary can still force us to incur a loss of $\mathcal{O}(T)$, but not without negatively affecting the performance of the best constant predictor as well.

We saw previously that randomization was a viable strategy against an adversary. What is the expected regret of a *pure guessing* strategy? Intuitively, if our adversary knows that our strategy is random, say: $\hat{x}_t \sim Ber(\frac{1}{2})$, they would just fix the sequence to be constant, as in the example of all 0's. Now, our expected regret is

$$\begin{aligned}\mathbb{E}[R(T)] &= \sum_{t=1}^{T} \mathbb{E}[\ell(x_t, \hat{x}_t)] - \min_{x^\star \in X} \mathbb{E}[\sum_{t=1}^{T} \ell(x_i, x^\star)] \\ &= \frac{1}{2}T - 0 = \frac{1}{2}T\end{aligned}$$

Thus, we are interested in the following natural question: *Is there a way to combine the strategies of pure randomization and follow-the-leader to achieve a lower, worst-case regret?*

In the next section, we will briefly introduce an algorithm that incurs a *worst-case* regret of $\mathcal{O}(\sqrt{T})$. In the next lecture, we will prove this.

4.4 Multiplicative Weights Algorithm

Intuitively, instead of making a hard prediction (predicting either 0 or 1) each round, why not make a soft prediction (choosing 0 with probability p and 1 with $1-p$)?

We can do this by initially assigning an equal "weight" to each class, decreasing or increasing it proportionately when we see a new label. To then convert these weights into probabilities, we can use the familiar softmax function:

$$p_{t,0} = \frac{w_{t,0}}{w_{t,0}+w_{t,1}},\ p_{t,1} = \frac{w_{t,0}}{w_{t,0}} + w_{t,1}.$$

The softmax function has the following property: $\exp(\alpha_1 x) + \exp(\alpha_2 x) \approx \exp(\max(\alpha_1, \alpha_2))$.

Here is the multiplicative weights (MW) algorithm:

Algorithm 1: Multiplicative Weights

```
1 function MW ($a, b$);
  Input  : stream of labels
  Output: sequence of predictions
2 initialize $w_{t,0} = w_{t,1} = 1$
3 initialize $L_{t,0} = L_{t,1} = 0$
4 for each time step $t = 2, \dots, T$ do
5     $L_{t-1,j} = \sum_{i=1}^{t-1} \mathbb{I}[X_i \neq j] = L_{t-2,j} + \mathbb{I}[X_{t-1} \neq j]$        $j \in \{0,1\}$
      $w_{t,j} = \exp(-\eta L_{t-1,j}) = w_j^{(t-1)} \cdot \exp(-\eta \mathbf{I}(x_{t-1} \neq j))$
6     predict $\hat{X}_t = 0$ with probability $\frac{w_{t,0}}{w_{t,0}+w_{t,1}}$, $\hat{X}_t = 1$ with probability $\frac{w_{t,1}}{w_{t,0}+w_{t,1}}$.
7 end
```

Note that the weight at time t can be written as a product of the previous weight and the exponential of loss, and this is one of the reason for calling it "multiplicative weights".

Here, "eta" $\eta \in (0, \infty)$ is a hyper parameter of the multiplicative weights algorithm and can be interpreted as the learning rate. It can either be tuned as a function of T, or a function of $\{X_i\}_{i=1}^t$ (the latter being a form of adaptive learning, which we will discuss in future lectures).

Intuitively,

- When η is close to 0, the distributions will adhere close to the uniform distribution. Thus, small values of η encourage a more uniformly guessing strategy since all predictions are given equal weight.
- If $\eta = 1$, then $p_{t,k}$ can be viewed as the Bayesian posterior probability of class k, with $\sum_k w_{t,k}$ as the marginal likelihood of the labels.
- On the other hand, large values of η quickly lead to one prediction with a disproportionate amount of weight and will cause the algorithm to behave like follow-the-leader.

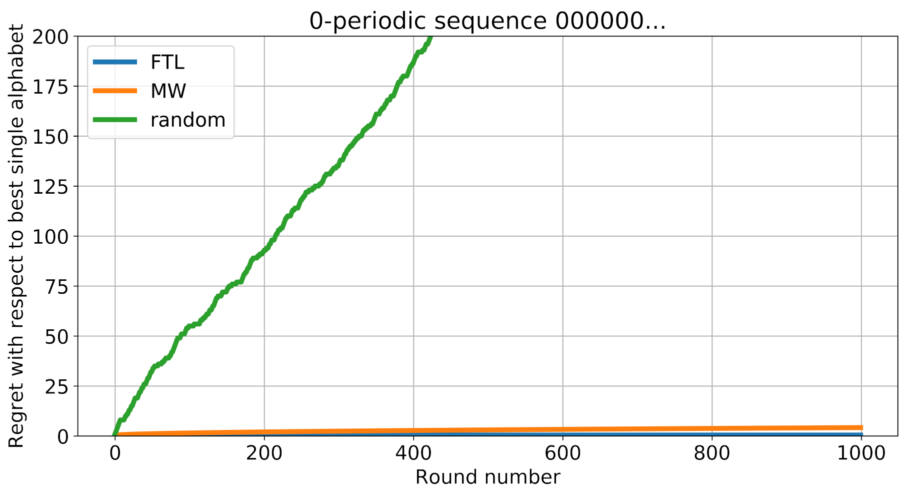
In this sense, η can be interpreted as knob between these two extremes – tuning it changes the extent to which the algorithm randomizes.

4.5 Performance

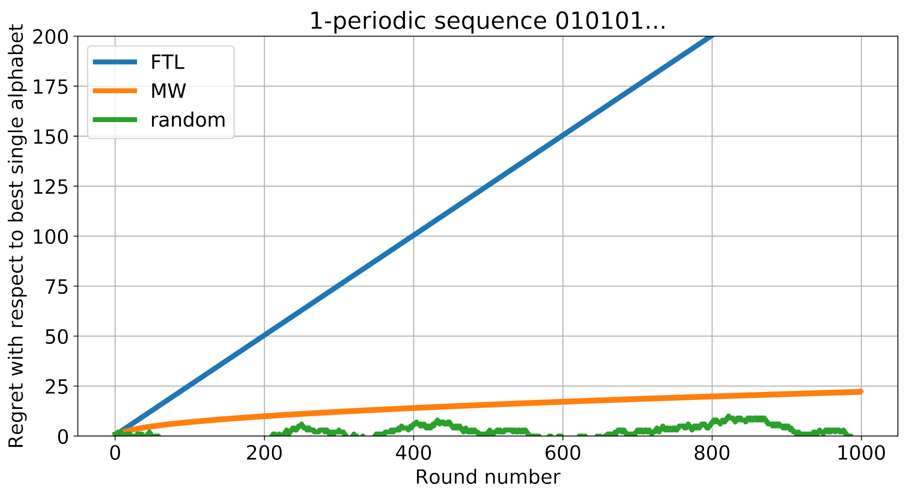
How well does the multiplicative weights algorithm perform against previously mentioned strategies? Here, we will only show empirical performance, and leave formal proofs for future lectures.

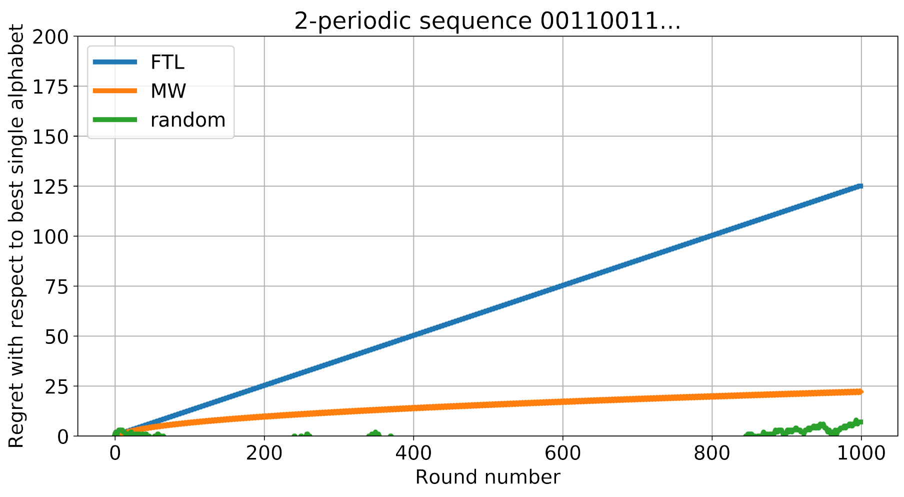
Recall our random strategy to be: flip a fair coin and choose 1 if heads and 0 otherwise.

As we can see below, that for a constant sequence, FTL performs the best with MW trailing. Both of them have a regret sublinear to T, while the random strategy incurs regret linear to it.



For the adversarial sequence like 1-periodic or 2-periodic, FTL yields the highest regret (linear). Multiplicative weights is better but is beaten by the random strategy. Among all of these three cases, multiplicative weights shows its versatility toward different environments.





4.6 References

1. Nicolo Cesa-Bianchi and Gabor Lugosi. Chapter 6.2 Label Efficient Prediction from *Prediction, Learning, and Games*, 2006.

2. Steven de Rooij, Tim van Erven, Peter D. Grunwald, and Wouter M. Koolen. *Follow the Leader If You Can, Hedge If You Must*, 2014.

3. Tim Roughgarden. CS261, *Lecture 11: Online Learning and the Multiplicative Weights Algorithm*, 2016.