DS 320 Algorithms for Data Science **Multiplicative Weight Update**
Spring 2024 Prof. Kira Goldner

Multiplicative Weight Update Key Points

- What you should remember about this problem:
  - There are a bunch of *experts* or *actions* that we want to choose from each time-step. Think of them as weather apps or financial advisors who you'll take advice from each day.
  - You come up with an algorithm to choose an expert/action each time step (you must use randomness to be competitive, as we saw—think back to rock paper scissors for intuition), then the adversary chooses losses/rewards for each expert, e.g., "how wrong was weather.com's prediction of the weather today?"
  - You want to update how you pick experts so that in hindsight you've done pretty okay. If one expert isn't doing a good job, you should stop giving them so much of a chance—so we *multiply* our weights to decrease them based on the *loss* ℓ that the expert suffered, parameterized by some update step size (η), i.e. for expert k at time $t+1$:

    $$w_k^{t+1} = w_k^t \cdot (1 - \eta\, \ell_k^t)$$

    and then we choose expert k with probability $w_k^t / \sum_i w_i^t$.
  - This is an *online* problem: we don't know all the information in advance and plan ahead, but rather, we learn a little bit at each time step (what the losses were for this last period) and have to *update* accordingly.
- In *online decision problems*, we compare to the *best fixed action* rather than the *best action sequence* because it's not actually realistic to get anywhere close to the best action sequence with an adversary.
- The *additive difference* (rather than multiplicative) between how the algorithm does and the best fixed action is called *regret*. For loss[1] vectors $\ell^1, \ldots, \ell^T$, the regret of the action sequence $a^1, \ldots, a^T$ is

$$\underbrace{\sum_{t=1}^{T} \ell^t(a^t)}_{\text{our algorithm}} - \underbrace{\min_{i=1}^{N} \sum_{t=1}^{T} \ell_i^t}_{\text{best fixed action}} . \tag{1}$$

- Regret $\Theta(T)$ is *really bad*. That means you made a mistake and the best fixed action didn't in effectively every time step. We want sublinear $o(T)$ regret, like $O(\sqrt{T})$.

[1]Note that for the rewards setting, the definition of regret would instead be $\max_{i=1}^{N} \sum_{t=1}^{T} r_i^t - \sum_{t=1}^{T} r^t(a^t)$, still minimizing the difference between the algorithm and the best fixed action, but now the maximum reward for the best fixed action will be larger than the algorithm instead of the minimum loss being smaller.