1.1 Course Structure

- 3 or 4 Problem sets (35%).
- Final project with final report or literature review (50%).
- Participation, such as scribing, answering questions proposed during lectures, etc (15%).

1.2 Overview of Core Materials

1.2.1 Time Line

<table>
<thead>
<tr>
<th>Early Online Learning</th>
<th>Modern Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hannan 1956</td>
<td>Littlestone/Warmuth 1992</td>
</tr>
<tr>
<td>Blackwell 1957</td>
<td>Zinkevich 2003</td>
</tr>
<tr>
<td>Rosenblatt (1957)</td>
<td>Kalai/Vempala 2006</td>
</tr>
<tr>
<td>... (Big time gap)</td>
<td>etc. (lots more)</td>
</tr>
<tr>
<td>Sequential “Learning” without Stochastic Process</td>
<td>Weighted Majority Algorithm</td>
</tr>
<tr>
<td>Generalization of Minimax Theorem: “reduction” from Hannan’s result.</td>
<td>Online Convex Optimization</td>
</tr>
<tr>
<td>The Perceptron</td>
<td>Follow the Perturbed Leader</td>
</tr>
</tbody>
</table>

1.2.2 Connections

Here’s a list of some major areas of research where strong connections have been found to the online no-regret framework that we’ll be discussing in this course:

- Boosting/Adaboost: originated from the Weighted Majority Algorithm
- Computing Nash Equilibrium
  - Zero Sum Games
- Comparing Correlation Equation
- Optimization
  - Convergence rates for Stochastic Gradient Descent
- Finance
“Universal” Portfolio
Black-Scholes Option Pricing

“Calibrated” Forecasting
Prediction Market Mechanism
Differential Privacy Mechanics

1.3 “Prediction with Expert Advice”

1.3.1 Problem Statement

1. There is a sequence of rounds. For each round, there is an “outcome”.
   For simplicity, our “outcome” is either 1 or 0.
   For example: \( y_1 = 0, \ y_2 = 1, \ y_3 = 1, \ldots \), and for any positive integer \( t, \ y_t \in \{0,1\} \).
   Such binary representation may reflect real life events, for example: rain/shine, price going up/price going down.

2. In addition to the rounds, we also have a pool of \( N \) “experts”.
   On round \( t \), the “advice” of the \( i \)th expert is denoted as \( f_{i,t} \in \{0,1\} \).
   For example, suppose that the 7th expert claims that the 5th round outcome is 1, then we denote it as \( f_{7,5} = 1 \).

3. We want to design a Master Algorithm, combining the experts’ advice, to predict the outcomes. We denote the prediction for the \( t \)th round outcome as \( \hat{y}_t \in \{0,1\} \).

4. Further, we have a formula to express the number of mistakes of the Master Algorithm up to round \( T \):

\[
\text{Mistake}_T(\text{MA}) = \sum_{t=1}^{T} 1[\hat{y}_t \neq y_t]
\]

where \( 1[\hat{y}_t \neq y_t] \) is the indicator function\(^1\).

1.3.2 Design a Good Master Algorithm

Our next question is how to design a good master algorithm, especially given historical performances of the experts. How should we combine the advice of the experts to make our own prediction?

We will assume we are in the “realizable” setting. That is, we will assume that there exist a perfect expert who guesses every round’s outcome correctly. Then we can use Halving Algorithm, which we now present.

1.3.2.1 Introduction to Halving Algorithm

Heuristically, suppose we are at the \( t \)th round, :

1. Maintain the “good” experts, that is, who predict the outcomes correctly up to the \( t \)th round.

2. To form a prediction on round \( t \), take a majority vote of the “good” experts!

\(^1\)The indicator function is defined as \( 1[\text{statement}] = \begin{cases} 1 & \text{if statement true;} \\ 0 & \text{if statement false}. \end{cases} \)
Symbolically, denote $C_t$ as the set of the consistently good experts up till the $t$th round. We start with 1st round and we have $C_1 := [N]$. At the $t$th round, we can inductively define

$$C_{t+1} = C_t \setminus \{i \in C_t : f_{i,t} \neq y_t\}.$$ 

Then we use master algorithm to predict the $t + 1$ round outcome

$$\hat{y}_t = \begin{cases} 1 & \text{if } |\{i \in C_t : f_{i,t} = 1\}| \geq \frac{|C_t|}{2} \\ 0 & \text{if } |\{i \in C_t : f_{i,t} = 0\}| > \frac{|C_t|}{2} \end{cases}$$

1.3.2.2 Upper Bound for $\text{Mistake}_T(\text{Halving})$

**Theorem 1.1** Suppose among the $N$ experts, there exists a perfect expert. For a sequence of $T$ rounds, the total number of mistakes made during Halving Algorithm is bounded by

$$\text{Mistake}_T(\text{Halving}) \leq \log_2 N$$

**Proof:** We have $|C_1| = N$. And by the assumption that there is always at least one perfect expert, $|C_T| \geq 1$. By the nature of Halving Algorithm, if $\hat{y}_t \neq y_t$, then $|C_{t+1}| \leq \frac{1}{2}|C_t|$. Together we have

$$1 \leq |C_T| \leq |C_1| \left(\frac{1}{2}\right)^{\text{Mistake}_T(\text{Halving})} = N \left(\frac{1}{2}\right)^{\text{Mistake}_T(\text{Halving})}$$

$$\implies N \geq 2^{\text{Mistake}_T(\text{Halving})}$$

$$\implies \text{Mistake}_T(\text{Halving}) \leq \log_2 N.$$ 

In addition, the upper bound is sharp. To see this, imagine we have $N = 2^n$ experts, only one of them is perfect, and for each round $t$, half of them vote 1, the other half 0, till the only one left is the perfect expert. And the correct outcome is always 0, then Halving algorithm makes exactly $\log_2 N$ mistakes.

Here is an exercise:

**Exercise:** Suppose the all assumptions in Theorem 1.1 still hold except that now there exists at least $K$ perfect experts. Now show that $\text{Mistake}_T(\text{Halving}) \leq \log_2 \left(\frac{N}{K}\right)$.

1.3.2.3 Predict Sorting $K$ teams

Here is another example where we can apply Halving Algorithm.

**Problem Statement:** There are $K$ teams, we denote them as 1, 2, 3, · · · $K$. We observe pairs $(i_1, j_1), (i_2, j_2), \cdots (i_T, j_T)$. Suppose that $\exists$ an unknown ranking of the teams, represented by $\pi^* \in S_k$.

The algorithm must predict the outcome, where

$$y_t = \begin{cases} 1 & \text{if } \pi^*(i_t) > \pi^*(j_t) \\ 0 & \text{if otherwise} \end{cases}$$

We may reduce this problem to a Halving problem. The “experts” are all the elements, $\pi$’s, in $S_k$, that is, all the possible permutations. Thus there are $|S_k| = K!$ of them. And

$$C_T = \{\pi \in S_k : \pi \text{ and } \pi^* \text{ agree on } i_t \text{ VS } j_t, \forall t \in [T - 1]\}.$$ 

$^2[N] = \{1, 2, \cdots N\}.$

$^3$For a set $A$, $|A|$ is the cardinality of $A$.

$^4S_k$ is the symmetric group on $[k]$, which consists of all the possible permutations of $[k]$. 


By Theorem 1.1, we know that $\text{Mistake}_T \leq \log_2 K! = O(K \log_2 K)$. Note that the complexity is the same as the binary tree sorting algorithm.

Here is a challenge:

**Challenge:** The naive way to implement the above algorithm is to list out all permutations and, after observing the outcome on each round, to simply delete the permutations that are inconsistent with this outcome. This would require $O(K!)$ computation on every round. But perhaps there’s an easier way? I am not entirely sure about the answer to this, so I pose the following question: Is there an efficient method to apply the halving algorithm to this predictive sorting problem? Ideally the algorithm would run in time polynomial in $K$ and $T$. 