AI-3001: Advanced Topics in Machine Learning **Aug-Nov 2022**

Lecture 1: Introduction to OPL

Lecturer: Ganesh Ghalme *Scribes: Ganesh Ghalme*

1.1 Introduction

1.1.1 Online learning framework

Algorithm 1: Online learning Framework

for $t = 1, 2, \cdots$ **do**
- ALG **selects action** $a_t \in \mathcal{A}$;
- **Environment/adversary selects an outcome** $y_t \in \mathcal{Y}$;
- ALG **incurs loss** $\ell(a_t, y_t)$;

end

Algorithm's Goal

- Minimize the cumulative loss i.e. $\sum_{t=1}^{T} \ell_t(a_t, y_t)$ (NOT VERY POPULAR)
- Minimize the regret i.e. $\sum_{t=1}^{T} \ell_t(a_t, y_t) - \min_{a \in \mathcal{A}} \sum_{t=1}^{T} \ell_t(a, y_t,)$
- We will consider $\ell_t = \ell$ throughout the course

Stopping time T

- Algorithm does not have an a-priori knowledge of T; must do well for every T
- Good algorithms have strictly decreasing per-round loss (Hannan Consistency)
- Also called as time horizon

Comparison with Classical ML

- Data comes online and algorithm must make decisions based on available data (no separation between test and training data) in contrast with standard ML models whose performance is evaluated on test data
- Regret vs accuracy: while regret is relative performance measure, accuracy is absolute
- Loss function is fixed in classical ML (and algorithms adapt to these losses) models whereas in online learning (typically) an algorithm must do well against a family of loss functions

- While most of classical ML considers a distributional assumption on the data this does not hold for many online learning models; data is not considered as iid samples from a fixed distribution (in fact, most of the models we will see do not attempt to model the environment)

In this course we study algorithm design under following different models

- Full-feedback vs Partial feedback model
  - At each time the history corresponding to all the actions (even the ones algorithms didnt take) is accessible to the algorithm
  - Example: Stock market investments, weather prediction
  - Bandit feedback: Only the history corresponding to actions taken by the algorithm is accessible (Reinforcement learning framework)
  - Example: Stochastic shortest path
- Expert advice model vs Actions model
  - In adversarial model experts may provide the advice to mislead it
  - While regret in experts model is defined over best expert, the actions model defines regret in terms of best action in "hindsight"
  - Experts problem typically considers finite experts, the actions set could be infinite
- Adversarial vs Stochastic setting
  - The regret in stochastic stochastic environment is defined as the expected regret incurred due to making suboptimal choices. Expectation is computed wrt the randomness of the environment and algorithm (if any)
  - In adversarial setting adversary selects the loss from a feasible set with a view to maximize algorithms' regret
  - Types of adversaries
    - **Oblivious adversary:** Adversary's choice of y_t does not depend on algorithms' past actions. Equivalently, the adversary only knows the randomness in the algorithm and not any of the realizations. One can think that the adversary fixes the sequence y_t beforehand and only reveals it to the algorithm at a given time instance.
    - **Non-oblivious adversary** Has access to entire history of realizations of the algorithms choice decisions.
    - **Offline adversary**: Also has access to future realizations. That is the adversary decides the sequence of outcomes $\{y_t\}$ after the algorithm has made its choices. Most powerful adversary.

1.1.2 Examples/Applications

1. **Shortest Path:** Given a graph $G = (V, E)$ and two vertices $u, v \in V$. The problem is to find the shortest path from u to v. At each time t, the algorithm chooses the path $p_t \in \mathcal{P}_{u,v}$ and adversary assigns weights $w_t \in \mathbb{R}^{|E|}$ to each edge. The loss function is given as $\ell(p_t, w_t) = \sum_{e \in p_t} w_t(e)$.
2. **Weather Forecast:** Let there are N weather forecasting channels (experts) and each gives a forecast about next days weather (rain/no rain). The algorithm wants to predict, as accurately as possible, whether it will rain or not each day. Every day algorithm receives predictions $f_{i,t} \in \{0, 1\}$ from each expert i, makes prediction $p_t \in \{0, 1\}$, and then the environment reveals the weather (rain/not rain) y_t. The algorithm incurs a loss of $\ell(p_t, y_t) = \mathbb{1}(p_t \neq y_t)$.

3. **Portfolio Optimization** There are N stocks. The algorithm must decide how to split a single rupee among these N stocks to maximize the returns. At each time t, algorithm selects a portfolio $p_t \in \Delta_N$, environment selects the returns (given in terms of ratio of stock value at the end of the day and at the beginning) r_t. The expected return from the portfolio is given by $p_t^T r_t$. The loss in this setting is given by $\ell(p_t, r_t) = -p_t^T r_t$.

4. **Sponsored Search** N advertisers seek to advertise their product on search platform such as Google against a keyword search by the user. The platform works on "pay-per-click" based model which means that the advertiser only pays for the ad if user clicks on it. This parameter is specified by 'click-through-rate' (probability that an user will click if the product is shown to her). However, the click through rate is unknown to the search engine which tries to learn it with time. The platforms goal is to minimize the expected regret given by

$$\sum_{t=1}^{T} [\max_i \mathbb{E}[r_i] - \mathbb{E}[r_{i_t}]]$$

5. **Spam filtering** Assume that an e-mail is reprented by using a "bag-of-words" representation. That is, a vector representing number of words from the dictionary with d words. At each time we get an email x_t and our algorithm selects a classifier (say linear) represented by a vector p_t and predicts spam/not spam based on $x_t^T p_t$. The environment reveals the true nature of an email and the loss function is either 0-1 loss or the squared loss.

1.1.3 Learning With Expert Advice

Algorithm 2: Learning with Expert Advice framework

Input: N experts
for $t = 1, 2, \cdots$ **do**
- **Each expert provide advice:** $f_{i,t} \in \mathcal{D}$;
- ALG **selects action** $a_t \in \mathcal{D}$;
- **Environment/adversary selects an action** $y_t \in \mathcal{Y}$;
- ALG **incurs loss** $\ell(a_t, y_t)$;

end

Today we discuss binary prediction with expert advice problem.

- $\mathcal{D} = \{0, 1\}$ and ALG makes a binary prediction $p_t \in \{0, 1\}$
- Outcome $y_t \in \{0, 1\}$ is also binary.
- The loss function is 0-1 loss i.e. $\ell(p_y, y_t) = \mathbb{1}(p_t \neq y_t)$.
- Decision makers goal is to minimize the cumulative *mistakes* i.e. $\sum_{t=1}^{T} \mathbb{1}(p_t \neq y_t)$

Assumption: There exists a perfect expert; i.e. $\exists i$ such that $f_{i,t} = y_t$ for every $t = 1, 2, \cdots$

1.1.3.1 Naive Approach

Try each expert one by one till they make a mistake. Once a chosen expert makes a mistake move on to the next one.

Algorithm 3: NAIVE

Data: # experts N
Initialize: $i = 1$;
for $t = 1, 2, \cdots$ **do**
  - **Input**: Expert recommendations $(f_{i,t})_{i=1}^{N}$;
  - **Predict** $p_t = f_{i,t}$;
  - **Observe** y_t ;
  if $p_t \neq y_t$ **then**
    $i \leftarrow i + 1$
end

Observation 1. *Each mistake eliminates exactly one expert.*

Claim 1.1. *The* NAIVE *algorithm makes at-most* $N - 1$ *mistakes.*

Proof of the claim is left as an exercise. The question is, can we do better than a linear (in terms of number of experts) number of mistakes. Towards this, we give another algorithm, known in literature as MAJORITY algorithm.

1.1.3.2 Majority Algorithm

Algorithm 4: MAJORITY

Input: # experts: N
Initialize: $\mathcal{A} = N$;
for $t = 1, 2, \cdots$ **do**
  - **Input:** expert recommendations $(f_{i,t})_{i=1}^{N}$;
  - **Predict:** $p_t = \mathbb{1}\{\sum_{i \in \mathcal{A}} f_{i,t} \geq \frac{|\mathcal{A}|}{2}\}$;
  - **Observe:** y_t ;
  if $p_t \neq y_t$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i \in \mathcal{A} : f_{i,t} = p_t\}$;
  end
end

MAJORITY algorithm selects the choice made by majority of experts at each time. If the majority makes a wrong choice we eliminate all those agents.

Observation 2. *The* MAJORITY *algorithm eliminates at-least half of the currently active experts after every mistake.*

Proof is left as an exercise. Since we eliminate more than one agents with every mistake we can arrive at the perfect expert with significantly less number of mistakes. Below results substantiates this intuition.

Claim 1.2. *Weighted Majority algorithm finds the perfect expert in at-most* $\log_2(N)$ *mistakes.*

Proof. Let $a_t = |\mathcal{A}_t|$ be experts active (not eliminated) till time t. We have from above observation that $a_t \leq a_{t-1}/2$. Which implies $a_t \leq a_0/2^t = N/2^t$ for all t. The algorithm finds the perfect expert when $a_t = 1$ for the first time i.e. when $1 \leq N/2^t$. This gives $t = \log_2(N)$. □

Relaxing the assumption about perfect expert We considered above that there exists a perfect expert; an expert who does not make any mistake. We use this assumption while deriving the mistake bound for the majority algorithm. Our next question is can we modify the majority algorithm to make it work without the perfect-expert assumption.

1.1.3.3 Weighted Majority Algorithm

When we relax the perfect expert assumption, we allow for the best expert to make mistakes. We cannot rule out experts completely even after they make mistakes (why?). On the other hand, one should put less confidence on the recommendation by experts that have made more mistakes in the past. The weighted majority algorithm given next, captures this intuition.

1.1.4 Next time

Next time we will see weighted majority and randomized weighted majority algorithms.