COMBINING EXPERT ADVICE
NOTES FOR THE ADVANCED ALGORITHM CLASS

GIUSEPPE PERSIANO

CONTENTS

1. Model 2
2. Warm up 2
3. Greedy Algorithm 2
4. Randomized Greedy Algorithm 3
5. Weighted Majority Algorithm 3

1. Model

We have a set \( E = \{E_1, \ldots, E_N\} \) of \( N \) experts and, at each time step \( t \), the algorithm \( \text{Algo} \) has to choose a distribution \( p^t = (p_1^t, \ldots, p_N^t) \) over the set \( E \) of experts. \( p_i^t \) is the probability that \( \text{Algo} \) follows the recommendation of expert \( E_i \) at time step \( t \).

A deterministic algorithm \( \text{Algo} \) chooses, at each time step \( t \), a probability distribution \( p^t \) that assigns probability 1 to exactly one expert and probability 0 to all the remaining experts.

After the algorithm \( \text{Algo} \) has made his choice for step \( t \), adversary \( A \) chooses a loss vector \( \text{Loss}^t = (\text{Loss}_1^t, \ldots, \text{Loss}_N^t) \), where \( \text{Loss}_i^t \) is the loss incurred into by the \( i \)-th expert at time step \( t \). The loss of \( \text{Algo} \) at time step \( t \) is \( \text{Loss}_{\text{Algo}}(t) = \sum_{i=1}^N p_i^t \cdot \text{Loss}_i^t \). The cumulative loss of \( \text{Algo} \) up to time step \( T \) is the sum of all losses of \( \text{Algo} \),

\[
\text{cLoss}_{\text{Algo}}(T) = \sum_{t=1}^T \text{Loss}_{\text{Algo}}(t)
\]

and, similarly, the cumulative loss of \( E_i \) up to time step \( T \) is the sum of all of its losses,

\[
\text{cLoss}_i(T) = \sum_{t=1}^T \text{Loss}_i^t.
\]

We want to design algorithms that have small regret

\[
\text{Regret}_{\text{Algo}}(T) = \text{cLoss}_{\text{Algo}}(T) - \text{opt}(T).
\]

where

\[
\text{opt}(T) = \min_{i=1}^N \text{cLoss}_i(T).
\]

2. Warm up

Let us consider a very simple example: experts predict a simple binary value set by the adversary (for example, the stock market is going up or is going down) and thus losses are either 0 (the expert guessed right) or 1 (the expert did not guess right). In addition, we are guaranteed that there is an expert that never makes mistakes. Then the following very simple deterministic algorithm has regret at most \( \log N \).

(1) initially, set \( S = \{E_1, \ldots, E_N\} \);

(2) at each time step the algorithm predicts according to the majority of the experts in \( S \) and update \( S \) by removing experts that predicted incorrectly.

First of all, observe that the existence of an infallible expert guarantees that the set \( S \) is non-empty. Moreover, each time the algorithm makes a mistake (and thus incurs into a loss of 1) the set \( S \) is reduced by at least 1/2. Therefore the number of mistakes is at most \( \log N \).

3. Greedy Algorithm

Let us consider the greedy algorithm that selects at each time step the expert that currently has the minimum cumulative loss. If the minimum is attained by more than one action the greedy algorithm picks the one with the minimum index.

Theorem 1. For any sequence of loss vectors,

\[
\text{cLoss}_{\text{Greedy}}(T) \leq N \cdot \text{opt}(T) + (N - 1).
\]
Proof. We assume for simplicity that losses are either 0 or 1. Let us denote by $S^t$ the set of experts with minimum cumulative loss just before the decision at time step $t$ is taken and call a time step $t$ bad if $c\text{Loss}_{\text{Greedy}}(t)$ increases by 1 (because the algorithm has made a mistake) but $\text{opt}(t)$ does not increase. We observe that $S^t$ decreases by one for each bad step. Indeed, at a bad step $\text{Greedy}$ followed the (wrong) advice of an expert in $S^{t-1}$ which is then removed from $S^t$. Therefore $|S^t| < |S^{t-1}|$. Therefore, we can have at most $N$ bad step between successive increments of $\text{opt}(t)$ which implies the theorem. ■

4. Randomized Greedy Algorithm

We now modify the $\text{Greedy}$ algorithm and consider the randomized greedy algorithm $r\text{Greedy}$ that, at each time step, follows a randomly chosen expert with minimum cumulative loss. That is, at time step $t$, $p^t_i = 1/|S^t|$, if $i \in S^t$; $p^t_i = 0$, if $i \notin S^t$.

Theorem 2. For any sequence of loss vectors,
\[ c\text{Loss}_{r\text{Greedy}}(T) \leq (1 + \ln N) \cdot \text{opt}(T) + \ln N. \]

Proof. We assume for simplicity that losses are either 0 or 1. Consider a step $t$ in which $\text{opt}(t)$ does not increase and suppose that $S^{t-1} = n$ and $S^t = n - k$. Then the loss $\text{Loss}_{r\text{Greedy}}(t) = k/n$ since each expert of $S^{t-1}$ is selected with probability $1/n$ and exactly $k$ of them incur into a loss. Now observe that
\[ \text{Loss}_{r\text{Greedy}}(t) = \frac{k}{n} = \frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n-k+1} \]

Then between successive increments of $\text{opt}(t)$ the loss of $r\text{Greedy}$ is at most
\[ \frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \cdots + \frac{1}{1} \leq 1 + \ln N. \]

□

5. Weighted Majority Algorithm

In making a decision at time step $t$, the $\text{Greedy}$ and $r\text{Greedy}$ algorithm only take into account the experts that have the minimum cumulative losses and ignore the other experts. We next study a class of algorithms that take into account all experts with a weight that depends on the past performance of the experts. More precisely, for the case in which losses are in $\{0, 1\}$, the multiplicative weights update algorithm, MWU, proceeds as follows. For $i = 1, \ldots, N$, the initial weight, $w^0_i$, of $E_i$ is initialized to 1. At each time step $t$, predict according to the weighted majority of experts according to weights $w_i^{t-1}, \ldots, w_N^{t-1}$. Then set weight $w_i^t := w_i^{t-1}/2^t$. Notice that this update rule has the effect of halving the weights of all experts that incurred into a loss and in leaving unchanged the weights of the experts that did not incur into a loss.

Theorem 3. For any sequence of loss vectors,
\[ c\text{Loss}_{\text{MWU}}(T) \leq 2.4(\text{opt}(T) + \log N). \]
Proof. If MWU makes a mistake at time step $t$ this must be because the total weight of all the experts that were wrong at time step $t$ is at least one half; after time step $t$ their total weight is halved. In other words, the total weight after time step $t$ is reduced by at least $3/4$. Since the initial total weight $W^0 = N$, if $c\text{Loss}_{\text{MWU}}(T) = M$ (that is, up to time $T$, MWU has made $M$ mistakes) then

$$W^t \leq N \cdot \left(\frac{3}{4}\right)^M.$$  

On the other hand, if $\text{opt}(T) = m$ then the weight of the best expert (and thus the total weight) is at least $1/2m$. Thus we can write

$$\left(\frac{1}{2}\right)^m \leq W^t \leq N \cdot \left(\frac{3}{4}\right)^M$$

whence

$$c\text{Loss}_{\text{MWU}}(T) \leq \frac{\text{opt}(T) + \log N}{\log(4/3)} \leq 2.4(\text{opt}(T) + \log N).$$

Of course, the choice of halving the weight of wrong experts is arbitrary. Algorithm $\text{MWU}(\epsilon)$ multiplies the weight of wrong experts by $1 - \epsilon$ and has the following result.

**Theorem 4.** For any sequence of loss vectors,

$$c\text{Loss}_{\text{MWU}(\epsilon)}(T) \leq 2 \cdot (1 + \epsilon) \cdot \text{opt}(T) + \frac{2 \ln N}{\epsilon}.$$  

Algorithm $\text{MWU}(\epsilon)$ guarantees a performance that is about twice that of the best expert. We next describe the randomized multiplicative weights update algorithm, $\text{rMWU}(\epsilon)$ that achieves a cumulative loss that is very close to that of the best expert. $\text{rMWU}(\epsilon)$ associates weight $w_i$ to expert $E_i$ and initializes the weights by setting $w_1^i = 1$, for $i = 1, \ldots, N$. At each time step $t$, $\text{rMWU}(\epsilon)$ sets $p_t^i := w_t^i / W^t$ where $W^t = \sum_i w_t^i$ is the total weight. Once the loss vector is announced, weights are updated by setting $w_{t+1}^i := w_t^i \cdot (1 - \epsilon)^{\text{Loss}_t^i}$. We have the following theorem.

**Theorem 5.** For any sequence of loss vectors and $\epsilon \leq 1/2$,

$$c\text{Loss}_{\text{rMWU}(\epsilon)}(T) \leq (1 + \epsilon) \cdot \text{opt}(T) + \frac{\ln N}{\epsilon}.$$  

Proof. We prove the theorem for losses in $\{0,1\}$. Essentially the same proof works for general loss vectors.

Denote by $L^T$ the experts $E_i$ for which $\text{Loss}_i^T = 1$ and let $F^T := \sum_{i=1}^{\text{Loss}_i^T} w^T_i / W^T$ be the fraction of the total weight corresponding to $L^T$; notice that $F^T = \text{Loss}_i^T$ as it coincides with the loss of the algorithm at time step $T$. The weight of each of the experts in $L^T$ is multiplied by $(1 - \epsilon)$ and thus the total weight at time step $T + 1$ is $W^{T+1} = (1 - \epsilon F^T) \cdot W^T$. On the other hand, $W^{T+1} \geq \max_i w_i^{T+1} = (1 - \epsilon)^{\text{opt}(T)}$ and thus we can write

$$(1 - \epsilon)^{\text{opt}(T)} \leq W^{T+1} = W^1 \cdot \prod_{t=1}^{T} (1 - \epsilon F^t) = N \cdot \prod_{t=1}^{T} (1 - \epsilon F^t).$$
Taking logarithms and using the inequality $\ln(1 - z) \leq -z$, we write

$$\text{opt}(T) \ln(1 - \epsilon) \leq \ln N + \sum_{t=1}^{T} \ln(1 - \epsilon F_t)$$

$$\leq \ln N + \sum_{t=1}^{T} \epsilon F_t$$

$$\leq \ln N - \epsilon c_{\text{Loss}_{r\text{MWU}}(\epsilon)}(T)$$

Using the fact that $\ln(1 - z) \geq -z - z^2$, for $0 \leq z \leq 1/2$, we have

$$c_{\text{Loss}_{r\text{MWU}}(\epsilon)}(T) \leq - \frac{\text{opt}(T) \ln(1 - \epsilon)}{\epsilon} + \frac{\ln(N)}{\epsilon}$$

$$\leq (1 + \epsilon) \cdot \text{opt}(T) + \frac{\ln N}{\epsilon}.$$

By setting $\epsilon = \min(\sqrt{(\ln N)/T}, 1/2)$ gives $c_{\text{Loss}_{r\text{MWU}}(\epsilon)}(T) \leq \text{opt}(T) + 2\sqrt{T \ln N}$. 