CS 683 — Learning, Games, and Electronic Markets **Spring 2007**

Notes from Week 1: Algorithms for sequential prediction

Instructor: Robert Kleinberg *22-26 Jan 2007*

1 Introduction

In this course we will be looking at online algorithms for learning and prediction. These algorithms are interesting in their own right — as a topic in theoretical computer science — but also because of their role in the design of electronic markets (e.g. as algorithms for sequential price experimentation, or for online recommendation systems) and their role in game theory (where online learning processes have been proposed as an explanation for how players learn to play an equilibrium of a game).

2 Online algorithms formalism

For general background on online algorithms, one can look at the book *Online Computation and Competitive Analysis* by Borodin and El-Yaniv, or read the notes from an online algorithms course taught by Michel Goemans at MIT, available by FTP at

`ftp://theory.csail.mit.edu/pub/classes/18.415/notes-online.ps`

In this section we give an abstract definition of online algorithms, suitable for the prediction problems we have studied in class.

Definition 1. An *online computation problem* is specified by:

1. A set of inputs $\mathscr{I} = \prod_{t=1}^{\infty} I_t$.
2. A set of outputs $\mathscr{O} = \prod_{t=1}^{\infty} O_t$.
3. A cost function $\mathsf{Cost} : \mathscr{I} \times \mathscr{O} \to \mathbb{R}$.

For a positive integer T, we will define

$$\mathscr{I}[T] = \prod_{t=1}^{T} I_t, \qquad \mathscr{O}[T] = \prod_{t=1}^{T} O_t.$$

One should interpret an element $i = (i_1, i_2, \ldots) \in \mathscr{I}$ as a sequence representing the inputs revealed to the algorithm over time, with i_t representing the part of the input revealed at time t. Similarly, one should interpret an element $o = (o_1, o_2, \ldots)$ as a sequence of outputs produced by the algorithm, with o_t being the output at time t.

Remark 1. The definition frames online computation problems in terms of an *infinite* sequence of inputs and outputs, but it is easy to incorporate problems with a finite time horizon T as a special case of the definition. Specifically, if $|I_t| = |O_t| = 1$ for all $t > T$ then this encodes an input-output sequence in which no information comes into or out of the algorithm after time T.

Definition 2. An *online algorithm* is a sequence of functions

$$F_t : \mathscr{I}[t] \to O_t.$$

An *adaptive adversary* (or, simply, *adversary*) is a sequence of functions

$$G_t : \mathscr{O}[t-1] \to I_t.$$

An adversary is called *oblivious* if each of the functions G_t is a constant function.

If F is an online algorithm and G is an adaptive adversary, the *transcript* of F and G is the unique pair $\mathsf{Trans}(F, G) = (i, o) \in \mathscr{I} \times \mathscr{O}$ such that for all $t \geq 1$,

$$\begin{aligned} i_t &= G_t(o_1, o_2, \ldots, o_{t-1}) \\ o_t &= F_t(i_1, i_2, \ldots, i_t). \end{aligned}$$

The *cost* of F and G is $\mathsf{Cost}(F, G) = \mathsf{Cost}(\mathsf{Trans}(F, G))$.

One should think of the algorithm and adversary as playing a game in which the adversary specifies a component of the input based on the algorithm's past outputs, and the algorithm responds by producing a new output. The transcript specifies the entire sequence of inputs and outputs produced when the algorithm and adversary play this game.

Remark 2. Designating an oblivious adversary is equivalent to designating a single input sequence $i = (i_1, i_2, \ldots) \in \mathscr{I}$.

Remark 3. Our definition of algorithm and adversary makes no mention of computational constraints (e.g. polynomial-time computation) for either party. In general we will want to design algorithms which are computationally efficient, but it is possible to ask meaningful and non-trivial questions about online computation without taking such constraints into account.

In defining randomized algorithms and adversaries, we think of each party as having access to infinitely many independent random bits (represented by the binary digits of a uniformly distributed element of $[0, 1]$) which are not revealed to the other party.

Definition 3. A *randomized online algorithm* is a sequence of functions

$$F_t : \mathscr{I}[t] \times [0,1] \rightarrow O_t.$$

A *randomized adaptive adversary* is a sequence of functions

$$G_t : \mathscr{O}[t-1] \times [0,1] \rightarrow I_t.$$

A randomized adversary is called *oblivious* if the output of each function $G_t(o, y)$ depends only on the parameter y.

If F and G are a randomized algorithm and randomized adaptive adversary, respectively, then the *transcript* of F and G is the function $\mathsf{Trans}(F, G) : [0,1] \times [0,1] \rightarrow \mathscr{I} \times \mathscr{O}$ which maps a pair (x, y) to the unique input-output pair (i, o) satisfying:

$$\begin{aligned} i_t &= G_t(o_1, o_2, \ldots, o_{t-1}, y) \\ o_t &= F_t(i_1, i_2, \ldots, i_t, x) \end{aligned}$$

for all $t \geq 1$. The *cost* of F and G is $\mathsf{Cost}(F, G) = \mathbf{E}[\mathsf{Cost}(\mathsf{Trans}(F, G)(x, y))]$, when the pair (x, y) is sampled from the uniform distribution on $[0,1]^2$.

Remark 4. A randomized oblivious adversary is equivalent to a probability distribution over input sequences $i = (i_1, i_2, \ldots) \in \mathscr{I}$.

Remark 5. In class I defined a randomized algorithm using an infinite sequence of independent random variables $(x_1, x_2, \ldots) \in [0,1]^\infty$, and similarly for a randomized adversary. Consequently the transcript $\mathsf{Trans}(F, G)$ was described as a function from $[0,1]^\infty \times [0,1]^\infty$ to $\mathscr{I} \times \mathscr{O}$. This was unnecessarily complicated: a single random number $x \in [0,1]$ contains infinitely many independent random binary digits, so it already contains as much randomness as the algorithm would need for an entire infinite sequence of input-output pairs. Accordingly, in these notes I have simplified the definition by assuming that the algorithm's and adversary's random bits are contained in a single pair of independent random real numbers (x, y), with x representing the algorithm's supply of random bits and y representing the adversary's supply.

3 Binary prediction with one perfect expert

As a warm-up for the algorithms to be presented below, let's consider the following "toy problem." The algorithm's goal is to predict the bits of an infinite binary sequence $\vec{B} = (B_1, B_2, \ldots)$, whose bits are revealed one at a time. Just before the t-th bit is revealed, a set of n experts make predictions $b_{1t}, b_{2t}, \ldots, b_{nt} \in \{0,1\}$. The algorithm is allowed to observe all of these predictions, then it makes a guess denoted by $a_t \in \{0,1\}$, and then the truth, B_t, is revealed. We are given a promise that there is at least one expert whose predictions are always accurate, i.e. we are promised that $\exists i \, \forall t \; b_{it} = B_t$.

This prediction problem is a special case of the framework described above. Here, $I_t = \{0,1\} \times \{0,1\}^n$ and $O_t = \{0,1\}$. The input i_t contains all the information revealed to the algorithm after it makes its $(t-1)$-th guess and before it makes its t-th guess: thus i_t consists of the value of B_{t-1} together with all the predictions $b_{1t}, \ldots, b_{nt}$. The output o_t is simply the algorithm's guess a_t. The cost $\mathsf{Cost}(i,o)$ is the number of times t such that $a_t \neq B_t$.

Consider the following algorithm, which we will call the "Majority algorithm": at each time t, it consults the predictions of all experts who did not make a mistake during one of the first $t-1$ steps. (In other words, it considers all experts i such that $b_{is} = B_s$ for all $s < t$.) If more of these experts predict 1 than 0, then $a_t = 1$; otherwise $a_t = 0$.

Theorem 1. *Assuming there is at least one expert i such that $b_{it} = B_t$ for all t, the Majority algorithm makes at most $\lfloor \log_2(n) \rfloor$ mistakes.*

Proof. Let S_t denote the set of experts who make no mistakes before time t. Let $W_t = |S_t|$. If the Majority algorithm makes a mistake at time t, it means that at least half of the experts in S_t made a mistake at that time, so $W_{t+1} \leq \lfloor W_t/2 \rfloor$. On the other hand, by assumption we have $|W_t| \geq 1$ for all t. Thus the number of mistakes made by the algorithm is bounded above by the number of iterations of the function $x \mapsto \lfloor x/2 \rfloor$ required to get from n down to 1. This is $\lfloor \log_2(n) \rfloor$. $\square$

Remark 6. The bound of $\lfloor \log_2(n) \rfloor$ in Theorem 1 is information-theoretically optimal, i.e. one can prove that no deterministic algorithm makes strictly fewer than $\lfloor \log_2(n) \rfloor$ mistakes on every input.

Remark 7. Although the proof of Theorem 1 is very easy, it contains the two essential ingredients which will reappear in the analysis of the Weighted Majority and Hedge algorithms below. Namely, we define a number W_t which measures the "remaining amount of credibility" of the set of experts at time t, and we exploit two key properties of W_t:

- When the algorithm makes a mistake, there is a corresponding multiplicative decrease in W_t.
- The assumption that there is an expert whose predictions are close to the truth implies a lower bound on the value of W_t for all t.

The second property says that W_t can't shrink too much starting from its initial value of n; the first property says that if W_t doesn't shrink too much then the algorithm can't make too many mistakes. Putting these two observations together results in the stated mistake bound. Each of the remaining proofs in these notes also hinges on these two observations, although the manipulations required to justify the two observations become more sophisticated as the algorithms we are analyzing become more sophisticated.

```
Algorithm WMA($\varepsilon$)

/* Initialization */
$w_i \leftarrow 1$ for $i = 1, 2, \ldots, n$.

/* Main loop */
for $t = 1, 2, \ldots$
    /* Make prediction by taking weighted majority vote */
    if $\sum_{i\,:\,b_{it}=0} w_i > \sum_{i\,:\,b_{it}=1} w_i$
        output $a_t = 0$;
    else
        output $a_t = 1$.

    Observe the value of $B_t$.

    /* Update weights multiplicatively */
    $E_t \leftarrow$ {experts who predicted incorrectly}
    $w_i \leftarrow (1-\varepsilon) \cdot w_i$ for all $i \in E_t$.
end
```

Figure 1: The weighted majority algorithm

4 Deterministic binary prediction: the Weighted Majority Algorithm

We now present an algorithm for the same binary prediction problem discussed in Section 3. This new algorithm, the Weighted Majority algorithm, satisfies a provable mistake bound even when we don't assume that there is an expert who never makes a mistake. The algorithm is shown in Figure 1. It is actually a one-parameter family of algorithms WMA(ε), each with a preconfigured parameter $\varepsilon \in (0, 1)$.

Theorem 2. *Let M denote the number of mistakes made by the algorithm* WMA(ε)*. For every integer m, if there exists an expert i which makes at most m mistakes, then*

$$M < \left(\frac{2}{1-\varepsilon}\right) m + \left(\frac{2}{\varepsilon}\right) \ln(n).$$

Proof. Let w_{it} denote the value of w_i at the beginning of the t-th iteration of the main loop, and let $W_t = \sum_{i=1}^{n} w_{it}$. The hypothesis implies that there is an expert i such that $w_{iT} \geq (1-\varepsilon)^m$ for all T, so

$$W_T > w_{iT} \geq (1-\varepsilon)^m \tag{1}$$

for all T. On the other hand, if the algorithm makes a mistake at time t, it implies that

$$\sum_{i \in E_t} w_{it} \geq \frac{W_t}{2},$$

hence

$$\begin{aligned} W_{t+1} &= \sum_{i \in E_t} (1-\varepsilon) \cdot w_{it} + \sum_{i \notin E_t} w_{it} \\ &= \sum_{i=1}^{n} w_{it} - \varepsilon \sum_{i \in E_t} w_{it} \\ &\leq W_t \left(1 - \frac{\varepsilon}{2}\right). \end{aligned}$$

For any $T > 0$, we find that

$$\frac{W_T}{W_0} = \prod_{t=0}^{T-1} \frac{W_{t+1}}{W_t} \leq \left(1 - \frac{\varepsilon}{2}\right)^M \tag{2}$$

where M is the total number of mistakes made by the algorithm $\mathsf{WMA}(\varepsilon)$. Combining (1) with (2) and recalling that $W_0 = \sum_{i=1}^n w_{i0} = \sum_{i=1}^n 1 = n$, we obtain

$$\frac{(1-\varepsilon)^m}{n} < \frac{W_T}{W_0} \leq \left(1 - \frac{\varepsilon}{2}\right)^M.$$

Now we take the natural logarithm of both sides.

$$\begin{aligned} \ln(1-\varepsilon)m - \ln(n) &< \ln\left(1 - \frac{\varepsilon}{2}\right) M & (3) \\ \ln(1-\varepsilon)m - \ln(n) &< -(\varepsilon/2)M & (4) \\ \ln\left(\frac{1}{1-\varepsilon}\right) m + \ln(n) &> (\varepsilon/2)M & (5) \\ \left(\frac{2}{\varepsilon}\right) \ln\left(\frac{1}{1-\varepsilon}\right) m + \left(\frac{2}{\varepsilon}\right) \ln(n) &> M & (6) \\ \left(\frac{2}{1-\varepsilon}\right) m + \left(\frac{2}{\varepsilon}\right) \ln(n) &> M & (7) \end{aligned}$$

where (4) was derived from (3) using identity (21) from the appendix of these notes, and (7) was derived from (6) using identity (22) from the appendix. □

5 Randomized prediction: the Hedge Algorithm

We now turn to a generalization of the binary prediction problem: the "best expert" problem. In this problem, there is again a set of n experts, which we will identify

```
Algorithm Hedge($\varepsilon$)

/* Initialization */
$w_x \leftarrow 1$ for $x \in [n]$

/* Main loop */
for $t = 1, 2, \ldots$
    /* Define distribution for sampling random strategy */
    for $x \in [n]$
        $p_t(x) \leftarrow w_x \Big/ \left(\sum_{y=1}^{n} w_y\right)$
    end
    Choose $x_t \in [n]$ at random according to distribution $p_t$.
    Observe cost function $c_t$.

    /* Update score for each strategy */
    for $x \in [n]$
        $w_x \leftarrow w_x \cdot (1-\varepsilon)^{c_t(x)}$
    end
end
```

Figure 2: The algorithm Hedge(ε).

with the set $[n] = \{1, 2, \ldots, n\}$. In each time step t, the adversary designates a cost function c_t from $[n]$ to $[0, 1]$, and the algorithm chooses an expert $x_t \in [n]$. The cost function C_t is revealed to the algorithm only after it has chosen x_t. The algorithm's objective is to minimize the sum of the costs of the chosen experts, i.e. to minimize $\sum_{t=1}^{\infty} c_t(x_t)$.

Observe that this problem formulation fits into the formalism specified in Section 2; the input sequence $(i_1, i_2, \ldots)$ is given by $i_t = c_{t-1}$, the output sequence $(o_1, o_2, \ldots)$ is given by $o_t = x_t$, and the cost function is

$$\mathsf{Cost}(i, o) = \sum_{t=1}^{\infty} i_{t+1}(o_t) = \sum_{t=1}^{\infty} c_t(x_t).$$

Also observe that the binary prediction problem is a special case of the best expert problem, in which we define $c_t(x) = 1$ if $b_{xt} \neq B_t$, 0 otherwise.

Figure 2 presents a randomized online algorithm for the best expert problem. As before, it is actually a one-parameter family of algorithms Hedge(ε) with a preconfigured parameter $\varepsilon \in (0, 1)$. Note the algorithm's similarity to WMA(ε): it maintains a vector of weights, one for each expert, and it updates these weights multiplicatively using a straightforward generalization of the multiplicative update rule in WMA. The main difference is that WMA makes its decisions by taking a weighted majority vote of the experts, while Hedge makes its decisions by performing a weighted random selection of a single expert.

Theorem 3. *For every randomized adaptive adversary, for every $T > 0$, the expected cost suffered by* Hedge(ε) *satisfies*

$$\mathbf{E}\left[\sum_{t=1}^{T} c_t(x_t)\right] < \left(\frac{1}{1-\varepsilon}\right)\mathbf{E}\left[\min_{x\in[n]}\sum_{t=1}^{T} c_t(x)\right] + \left(\frac{1}{\varepsilon}\right)\ln(n). \tag{8}$$

Proof. Let w_{xt} denote the value of w_x at the beginning of the t-th iteration of the main loop, and let $W_t = \sum_{x=1}^{n} w_{xt}$. Note that w_{xt}, W_t are random variables, since they depend on the adversary's choices which in turn depend on the algorithm's random choices in previous steps. For an expert $x \in [n]$, let $c_{1..T}(x)$ denote the total cost

$$c_{1..T}(x) = \sum_{t=1}^{T} c_t(x).$$

Let $x^* = \arg\min_{x\in[n]} c_{1..T}(x)$. We have

$$W_T > w_{x^*t} = (1-\varepsilon)^{c_{1..T}(x^*)}$$

and after taking logarithms of both sides this becomes

$$\ln(W_T) > \ln(1-\varepsilon)c_{1..T}(x^*) \tag{9}$$

On the other hand, we can bound the expected value of $\ln(W_T)$ from above, using an inductive argument. Let w_{*t} denote the vector of weights $(w_{1t}, \ldots, w_{nt})$.

$$\begin{aligned}
\mathbf{E}(W_{t+1}\,|\,w_{*t}) &= \sum_{x=1}^{n} \mathbf{E}\left((1-\varepsilon)^{c_t(x)} w_{xt}\,|\,w_{*t}\right) && (10)\\
&\le \sum_{x=1}^{n} \mathbf{E}\left((1-\varepsilon c_t(x)) w_{xt}\,|\,w_{*t}\right) && (11)\\
&= \sum_{x=1}^{n} w_{xt} - \varepsilon\mathbf{E}\left(\sum_{x=1}^{n} c_t(x) w_{xt}\,|\,w_{*t}\right) && (12)\\
&= W_t \cdot \left(1 - \varepsilon\mathbf{E}\left(\sum_{x=1}^{n} c_t(x) p_t(x)\,|\,w_{*t}\right)\right) && (13)\\
&= W_t \cdot (1 - \varepsilon\mathbf{E}(c_t(x_t)\,|\,w_{*t})) && (14)\\
\mathbf{E}(\ln(W_{t+1})\,|\,w_{*t}) &\le \ln(W_t) + \ln(1 - \varepsilon\mathbf{E}(c_t(x_t)\,|\,w_{*t})) && (15)\\
&\le \ln(W_t) - \varepsilon\mathbf{E}(c_t(x_t)\,|\,w_{*t}) && (16)
\end{aligned}$$

$$\begin{aligned}
\varepsilon \mathbf{E}(c_t(x_t) \,|\, w_{*t}) &\leq \ln(W_t) - \mathbf{E}(\ln(W_{t+1}) \,|\, w_{*t}) && (17)\\
\varepsilon \mathbf{E}(c_t(x_t)) &\leq \mathbf{E}(\ln(W_t)) - \mathbf{E}(\ln(W_{t+1})) && (18)\\
\varepsilon \mathbf{E}\left(\sum_{t=1}^{T} c_t(x_t)\right) &\leq \ln(n) - \mathbf{E}(\ln(W_T)). && (19)
\end{aligned}$$

Here, (11) is derived using identity (23) from the appendix, (13) is derived using the fact that $p_t(x) = w_{xt}/W_t$, (14) is derived using the observation that x_t is a random element sampled from the probability distribution $p_t(\cdot)$ on $[n]$, (15) and (16) are derived using the identities (24) and (21) respectively, (18) is derived by taking the unconditional expectation of both sides of the inequality, and (19) is derived by summing over t and recalling that $W_0 = n$.

Combining (9) and (19) we obtain

$$\begin{aligned}
\varepsilon \mathbf{E}\left(\sum_{t=1}^{T} c_t(x_t)\right) &< \ln(n) - \ln(1-\varepsilon)\mathbf{E}(c_{1..T}(x^*))\\
\mathbf{E}\left(\sum_{t=1}^{T} c_t(x_t)\right) &< \left(\frac{1}{\varepsilon}\right)\ln(n) + \frac{1}{\varepsilon}\ln\left(\frac{1}{1-\varepsilon}\right)\mathbf{E}(c_{1..T}(x^*))\\
\mathbf{E}\left(\sum_{t=1}^{T} c_t(x_t)\right) &< \left(\frac{1}{\varepsilon}\right)\ln(n) + \left(\frac{1}{1-\varepsilon}\right)\mathbf{E}(c_{1..T}(x^*))
\end{aligned}$$

where the last line is derived using identity (22) from the appendix. □

6 Appendix: Some useful inequalities for logarithms and exponential functions

In various steps of the proofs given above, we applied some useful inequalities that follow from the convexity of exponential functions or the concavity of logarithms. In this section we collect together all of these inequalities and indicate their proofs.

Lemma 4. *For all real numbers x,*

$$1 + x \leq e^x \tag{20}$$

with equality if and only if $x = 0$.

Proof. The function e^x is strictly convex, and $y = 1 + x$ is the tangent line to $y = e^x$ at $(0, 1)$. □

Lemma 5. *For all real numbers $x > -1$,*

$$\ln(1 + x) \leq x \tag{21}$$

with equality if and only if $x = 0$.

Proof. Take the natural logarithm of both sides of (20). ☐

Lemma 6. *For all real numbers $y \in (0,1)$,*

$$\frac{1}{y}\ln\left(\frac{1}{1-y}\right) < \frac{1}{1-y}. \tag{22}$$

Proof. Apply (21) with $x = \frac{y}{1-y}$, then divide both sides by y. ☐

Lemma 7. *For every pair of real numbers $x \in [0,1], \varepsilon \in (0,1)$,*

$$(1-\varepsilon)^x \le 1 - \varepsilon x \tag{23}$$

with equality if and only if $x = 0$ or $x = 1$.

Proof. The function $y = (1-\varepsilon)^x$ is strictly convex and the line $y = 1 - \varepsilon x$ intersects it at the points $(0,1)$ and $(1, 1-\varepsilon)$. ☐

Lemma 8. *For every random variable X, we have*

$$\mathbf{E}(\ln(X)) \le \ln(\mathbf{E}(X)) \tag{24}$$

with equality if and only if there is a constant c such that $\Pr(X = c) = 1$.

Proof. Jensen's inequality for convex functions says that if f is a convex function and X is a random variable,

$$\mathbf{E}(f(X)) \ge f(\mathbf{E}(X)),$$

and that if f is strictly convex, then equality holds if and only if there is a constant c such that $\Pr(X = c) = 1$. The lemma follows by applying Jensen's inequality to the strictly convex function $f(x) = -\ln(x)$. ☐