Lecture 1

The Experts/Multiplicative Weights Algorithm and its Analysis

In this lecture we turn to the problem of *online learning*, and analyze a very powerful and versatile algorithm for online learning problems called the *multiplicative weights update* algorithm. First, let's see some example problems.

1.1 Some optimization problems

Machine learning: Learning a linear classifier. In machine learning, a basic but classic setting consists of a set of k labeled examples $(a_1, l_1), ..., (a_k, l_k)$ where the $a_j \in \mathbb{R}^N$ are "feature vectors" and the l_j are labels in $\{-1, 1\}$. The problem is to find a *linear classifier*: a unit vector $x \in \mathbb{R}^N_+$ such that $\|x\|_1 = 1$ and $\forall j \in \{1, \ldots, k\}$, $l_j(a_j \cdot x) \geq 0$ (think of x as a distribution over the coordinates of the a_j, the features, that "explains", or correlates with, the labeling indicated by the l_j).

Machine learning: Boosting. Suppose given a sequence of training points $x_1, \ldots, x_N$ taken from some universe. Each point has an (unknown) label $c(x_i) \in \{0, 1\}$, where the function c is taken from some restricted set of functions (the "concept class" $\mathcal{C}$). The goal is to find a hypothesis function $h \in \mathcal{C}$ that assigns labels to points, replicating the function c in the best way possible (on average over the samples). For instance, in the previous example the concept class is the class of all linear classifiers (or, "hyperplanes") and the hypothesis function is x.

Call a learning algorithm *strong* if it outputs with probability at least $1 - \delta$ a hypothesis h such that $\frac{1}{N}|\{i : h(x_i) \neq c(x_i)\}| \leq \varepsilon$, and *weak* if the error is at most $\frac{1}{2} - \gamma$. Suppose the only thing we have is a weak learning algorithm: for any distribution D, it returns a hypothesis h such that

$$\mathop{\mathbf{E}}_{i \sim D} \frac{1}{2}|h(x_i) - c(x_i)| \leq \frac{1}{2} + \gamma.$$

The goal is to use the weak learner in order to construct a strong learner.

Approximation algorithms: Vertex Cover Given a universe $U = \{1, \ldots, N\}$ and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of U, the goal is to find the smallest possible number of sets from $\mathcal{C}$ that covers all of U.

Linear programming. Given a set of N linear constraints $a_i^T x \geq b_i$, where $a_i \in \mathbb{R}^m$ and $b_i \in \mathbb{R}$, the goal is to decide if there is an (approximately) feasible solution: does there exist an $x^* \in \mathbb{R}_+^m$ such that $a_i \cdot x^* \geq b_i - \delta$ for all $i \in \{1, \ldots, N\}$?

Game theory: Zero-sum games. Consider a game between two players, the "row" and "column" players. Each player can choose an action $i, j \in \{1, \ldots, N\}$. The payoff to the first player if the actions are (i, j) is $M(i, j) \in \mathbb{R}$, and the payoff to the second player is $-M(i, j)$. A player can decide to randomize and choose a distribution over actions which she will sample from. In particular, if the row player plays according to a distribution $p \in \mathbb{R}_+^N$ over row actions and the column player plays according to a distribution $q \imath \mathbb{R}_+^N$ over column actions, then the expected payoff of the row player is $p^T M q$ and the expected payoff of the column player is $-p^T M q$. The goal is to find (approximately) optimal strategies for both players: (p^*, q^*) such that

$$(p^*)^T M q^* + \varepsilon \geq \lambda^* = \max_q \min_p p^T B q = \min_p \max_q p^T B q$$

What do these five problems have in common? For each there is a natural greedy approach. In a zero-sum game, we could pick a random strategy to start with, find the second player's best response, then the first player's best response to this, etc. For the linear programming problem we can again choose a random x, find a constraint that is violated, update x. Same for the linear classifier problem (which can itself be written as a linear program). In the set cover problem we can pick a first set that covers the most possible elements, then a second set covering the highest possible number of uncovered elements, etc. In boosting we can tweak the distribution over points and call a weak learner to improve it the performance of our current hypothesis with respect to misclassified points.

In each case it is not so easy to analyze such a greedy approach, as there is alwyas a risk to get stuck in a local minimum. Now we're going to develop an abstract "greedy-but-cautious" algorithm that can be applied to efficiently solve each of these problems.

1.2 Online learning

Consider the following situation. There are T steps, $t = 1, \ldots, T$. At each step t,

- The player chooses an action $x^{(t)} \in \mathcal{K} \subseteq \mathbb{R}^N$.
- She suffers a loss $f^{(t)}(x^{(t)})$, where $f^{(t)} : \mathcal{K} \to [-1, 1]$ is the *loss function.*

The player's goal is to devise a strategy for choosing her actions $x^{(1)}, \ldots, x^{(T)}$ which minimizes the *regret*,

$$R_T = \sum_{t=1}^{T} f^{(t)}(x^{(t)}) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f^{(t)}(x). \tag{1.1}$$

The regret measures how much worse the player is doing compared to a player who is told all loss functions in advance, but is restricted to play the same action at each step. Since the player may be adaptive regret can be positive or negative; however in practice it will almost always be the case that this is a positive quantity.

Exercise 1. What if regret were defined, not with respect to the best *fixed* action in hindsight, but by comparison with an offline player who is allowed to switch between experts at every step? Construct an example of an online convex optimization problem where the loss suffered by such an offline player is 0, but the expected loss of *any* online learning algorithm is at least $T \cdot (1 - 1/N)$.

In online learning people distinguish between the cases where the player is told what the function $f^{(t)}$ is, or only what her specific loss $f^{(t)}(x^{(t)})$ is. The former scenario is called the "full information model", and it is the model we'll focus on. The latter is usually referred to as the "multi-armed bandit" problem, but we won't discuss it further. In addition we are going to make the following assumptions:

- The set of allowed actions for the player is the probability simplex $\mathcal{K}_N = \{p \in \mathbb{R}^N_+, \sum_i p_i = 1\}$,
- Loss functions are *linear*, $f^{(t)}(p) = m^{(t)} \cdot p$ where $m^{(t)} \in \mathbb{R}^N$ is such that $|m_i^{(t)}| \leq 1$ for all $i \in \{1, \ldots, N\}$. Note that this ensures that $f^{(t)}(p) \in [-1, 1]$ for any $p \in \mathcal{K}_N$.

This setting has an interpretation in terms of "experts": think of each of the coordinates $i = 1, \ldots, N$ as an expert. At each time t, the experts make recommendations. The player has to bet on an expert, or more generally on a distribution over experts. Then she gets to see how well the expert's recommendations panned out: to each expert i is associated a loss $m_i^{(t)}$, and the player suffers an average loss $\sum_i p_i^{(t)} m_i^{(t)}$. She wishes to minimize her regret R_T, which in this case boils down to

$$\begin{aligned} R_T &= \sum_{t=1}^{T} f^{(t)}(p^{(t)}) - \min_{p \in \mathcal{K}_N} \sum_{t=1}^{T} f^{(t)}(p) \\ &= \sum_{t=1}^{T} m^{(t)} \cdot p^{(t)} - \min_{i \in \{1,\ldots,N\}} \sum_{t=1}^{T} m_i^{(t)}, \end{aligned}$$

i.e. she is comparing her loss to that of the best expert in hindsight.

How would you choose which expert to follow so as to minimize your total regret? A natural strategy is to always choose the expert who has performed best so far. Suppose for simplicity that experts are always either right or wrong, so losses $m_i^{(t)} \in \{0, 1\}$, 0 for wrong

and 1 for right. Suppose also there is at least one expert who always gets it right. Finally suppose at each step we "hedge" and choose an expert at random among those who have always been correct so far. Then whenever we suffer a loss $r_t \in [0,1]$ it precisely means that a fraction r_t of the remaining experts (those who were right so far) got it wrong and are thus eliminated. Since $N\prod_t(1-r_t) \geq 1$ (there are N experts to start with and there must be at least 1 remaining), taking logarithms we see $\sum_t r_t \leq \ln N$, which is not bad; in particular it doesn't grow with T.

But here's a worse scenario. Suppose at the first step all experts do equally well, so $m^{(1)} = (1/2, \ldots, 1/2)$. But now experts get it right or wrong on alterante days, $m^{(2t)} = (1, \ldots, 1, 0, \ldots, 0)$ and $m^{(2t+1)} = (0, \ldots, 0, 1, \ldots, 1)$ for $t \geq 1$. In this case, it turns out we're always going to pick the wrong expert! So our regret increases by $1/2$ per iteration on average, and we never improve! In this case a much better strategy would be to realize that no expert is consistently good, so that by picking a random expert at each step we suffer an expected loss of $1/2$, which is just as good as the best expert — our regret would be 0 instead of $T/2$.

Follow the Regularized Leader. The solution is to consider a strategy which forces itself to not change its mind too quickly. Instead of jumping on the best expert at every step, we're going to add a "risk penalty" which will penalize distributions that concentrate too much on a small number of experts — in a way, we'll only allow such high concentration if there is really a high confidence that the experts are doing much better than the others. Specifically, instead of playing the distribution p such that

$$p^{(t+1)} = \arg\min_{p \in \mathcal{K}_N} \sum_{j \leq t} m^{(j)} \cdot p,$$

we'll choose

$$p^{(t+1)} = \arg\min_{p \in \mathcal{K}_N} \Big(\eta \sum_{j \leq t} m^{(j)} \cdot p + R(p) \Big), \tag{1.2}$$

where R is a "regularizer" and $\eta > 0$ a small weight of our choosing used to balance the two terms.

1.3 The Multiplicative Weights Update algorithm

The multiplicative weights update (MWU for short) algorithm is a special case of follow the regularized leader where the regularizer R is chosen to be the (negative of the) entropy function.

Regularizing via entropy. Let $R(p) = -H(p)$, where $H(p) = -\sum_i p_i \ln p_i$ is the *entropy* of p. (In thi lecture we'll use the unusual convention that logarithms are measured in "nats". This is only for convenience, and in later lectures we'll revert to the binary logarithm log.) We'll use the following properties of entropy.

Lemma 1.1. *Let p, q be distributions in $\mathcal{K}_N$, $H(p) = \sum_i p_i \ln \frac{1}{p_i}$ the entropy of p and $D(p\|q) = \sum_i p_i \ln \frac{p_i}{q_i}$ the relative entropy (sometimes also called KL divergence) between p and q. Then the following hold:*

1. *For all p, $0 \le H(p) \le \ln N$.*
2. *For all p and q, $D(p\|q) \ge 0$.*

Exercise 2. Prove the lemma.

The only way to have zero entropy is to have $\ln p_i = 0$ for each i such that $p_i \neq 0$, i.e. $p_i = 1$ for some i and zero elsewhere: low-entropy distributions are highly concentrated. At the opposite extreme, the only distribution with maximal entropy is the uniform distribution (as it is the only way to saturate Jensen's inequality used in the proof of the lemma). Thus the effect of using the negative entropy as a regularizer is to penalize distributions that are too highly concentrated.

The multiplicative weights update rule. With this choice of regularizer it is possible to solve for (1.2) explicitly. Specifically, note that the gradient at point q is simply $\nabla q_i = \eta \sum_{j \le t} m_i^{(j)} + \ln(q_i) + 1$, and setting this to zero gives $q_i = e^{-1-\eta \sum_{t \le j} m_i^{(t)}}$. Taking into account the normalization to a probability distribution, we find that the unique minimizer to (1.2) when $R(p) = -H(p)$ is

$$\begin{aligned} p_i^{(t+1)} &= \frac{e^{-\eta \sum_{j \le t} m_i^{(j)}}}{\sum_i e^{-\eta \sum_{j \le t} m_i^{(j)}}} \\ &= \frac{e^{-\eta m_i^{(t)}} p_i^{(t)}}{\sum_i e^{-\eta m_i^{(t)}} p_i^{(t)}}. \end{aligned}$$

This suggests the following *multiplicative weights update algorithm*:

Algorithm 1: Multiplicative Weights Algorithm (a.k.a MW or MWA)

Choose $\eta \in (0, 1/2]$;
Initialize weights $w_i^{(1)} := 1$ for each $i \in \{1, \dots, N\}$;
for $t = 1, 2, \dots, T$ **do**
  Choose decisions proportional to the weights $w_i^{(t)}$, i.e., use the distribution $p^{(t)} = \{\frac{w_1^{(t)}}{\Phi^{(t)}}, \dots, \frac{w_N^{(t)}}{\Phi^{(t)}}\}$, where $\Phi^{(t)} = \sum_i w_i^{(t)}$;
  Observe the costs $m^{(t)}$ of the decisions;
  Penalize costly decisions by updating their weights: $w_i^{(t+1)} \leftarrow e^{-\eta m_i^{(t)}} w_i^{(t)}$;
end

Thus at every step we update our weights $w^{(t)} \leftarrow w^{(t+1)}$ by re-weighting expert i by a multiplicative factor $e^{-\eta m_i^{(t)}}$ that is directly related to its performance in the previous step. However, we do so smoothly, a small step at a time; how small the step is governed by η, a parameter that we'll choose for best performance later.

Remark 1.2. Note that for small η, $e^{-\eta m_i^{(t)}} \approx 1 - \eta m_i^{(t)}$. In the homework you will analyze the modified update rule, $w_i^{(t+1)} = (1 - \eta m_i^{(t)}) w_i^{(t)}$, directly and show that it can sometimes lead to a better regret bound.

Analysis. The following theorem quantifies the performance of the Multiplicative Weights Algorithm.

Theorem 1.3. *For all $N \in \mathbb{N}$, $0 < \eta < 1$, $T \in \mathbb{N}$, losses $m_i^{(t)} \in [-1, 1]$ for $i \in \{1, \dots, N\}$ and $t \in \{1, \dots, T\}$ the above procedure starting at some $p^{(1)} \in \mathcal{K}_N$ suffers a total loss such that*

$$\sum_{t=1}^{T} m^{(t)} \cdot p^{(t)} \leq \min_{p \in \mathcal{K}_N} \Big(\sum_{t=1}^{T} m^{(t)} \cdot p + \frac{1}{\eta} D\big(p \| p^{(1)}\big) \Big) + \eta \sum_{t=1}^{T} \sum_{i=1}^{N} \big(m_i^{(t)}\big)^2 p_i^{(t)}$$

where $D(p\|q) = \sum_i p_i \ln(p_i/q_i)$ is the relative entropy.

Before proving the theorem we can make a number of important comments. First let's make some observations that will give us a simpler formulation. By choosing $p^{(1)}$ to be the uniform distribution we can ensure that $D(p\|p^{(1)}) = \ln N - H(p)$ is always at most $\ln N$, whatever the optimal p is. Moreover, using $|m_i^{(t)}| \leq 1$ the last term is at most ηT. Using the definition of the regret (1.1), we get

$$R_T \leq \frac{1}{\eta} \ln(n) + \eta T.$$

The best choice of η in this equation is $\eta = \sqrt{\ln(n)/T}$, in which case we get the bound

$$\frac{1}{T} R_T \leq 2\sqrt{\frac{\ln N}{T}}.$$

What this means is that, if we want to have an average regret, over the T rounds, that is at most ε, then it will suffice to run the procedure for a number of iterations $T = 4 \ln N/\varepsilon^2$. The most remarkable feature of this bound is the logarithmic dependence on N: in only a logarithmic number of iterations we are able to narrow down on a way to select experts that ensures our average loss is small. This matches the guarantee of our earlier naive procedure of choosing the best expert so far, except that now it works in *all* cases, whether the experts are consistent or not! The dependence on ε, however, is not so good. This is an important drawback of the MWA; in many cases it is not a serious limitation but it is important to keep it in mind, and we'll return to this issue when we look at some examples. First let's prove the theorem.

Proof of Theorem 1.3. The proof is based on the use of the relative entropy as a potential

function: for any q we measure the decrease

$$\begin{aligned} D(q\|p^{(t+1)}) - D(q\|p^{(t)}) &= \sum_i q_i \ln \frac{p_i^{(t)}}{p_i^{(t+1)}} \\ &= \eta \sum_i q_i m_i^{(t)} + \Big(\sum_i q_i\Big) \ln \Big(\sum_i e^{-\eta m_i^{(t)}} p_i^{(t)}\Big) \\ &\leq \eta m^{(t)} \cdot q - \eta m^{(t)} \cdot p^{(t)} + \eta^2 \sum_i \big(m_i^{(t)}\big)^2 p_i^{(t)}, \end{aligned}$$

where for the last line we used $e^{-x} \leq 1 - x + x^2$ and $\ln(1-x) \leq -x$ for all $|x| \leq 1$. Summing up over t and using that the relative entropy is always positive proves the theorem. $\square$