**Introduction to Computational Learning Theory** **Spring 2021**

# Lecture 7: Boosting

*Lecturer: Roi Livni* *Scribe:*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

We are now focusing on the computational aspects of learning, and we will already discussed several learning algorithms (focusing on learning half-spaces).

On the downside we also discussed hardness results – demonstrating that (statistical) learnability does not always imply efficient algorithm.

We next discuss Boosting techniques, which is a useful technique to turn 'finger rules" into a strong prediction rule. This algorithm can also be thought as a reduction from learning to what we will term *weak learning*.

## 7.1 The problem of Boosting

Next, we consider a fundamental property of learning in the PAC model so far considered: it is amenable to *boosting*. Roughly speaking, boosting refers to the process of taking a set of rough "rules of thumb" and combining them into a more accurate predictor.

Consider for example the problem of Optical Character Recognition (OCR) in its simplest form: given a set of bitmap images depicting hand-written postal-code digits, classify those that contain the digit "1" from those of "0".

Seemingly, discerning the two digits seems a formidable task taking into account the different styles of handwriting, errors, etc. However, an inaccurate rule of thumb is rather easy to produce: in the bottom-left area of the picture we'd expect many more dark bits for "1"s than if the image depicts a "0". This is, of course, a rather inaccurate statement. It does not consider the alignment of the digit, thickness of the handwriting etc. Nevertheless, as a rule of thumb - we'd expect better-than-random performance, or some correlation with the ground truth.

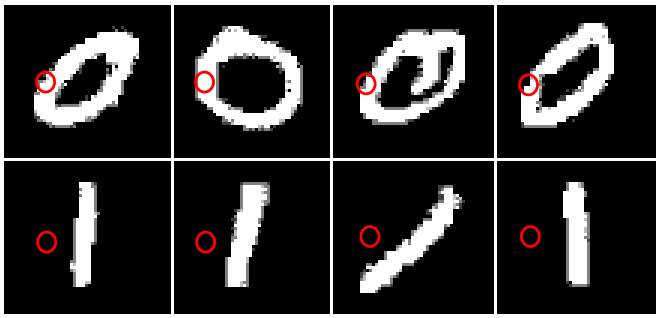

Figure 7.1: Distinguishing zero vs. one from a single pixle.

The inaccuracy of the simplistic single-bit predictor is compensated by its simplicity. It is a rather simple task to code up a classifier based upon this rule which is very efficient indeed. The natural and fundamental question which arises now is: can several of these rules of thumb be combined into a single, accurate and efficient classifier? From a computational view-point the problem of Boosting can be formalized as follows:

Suppose we have an efficient algorithm (or alternatively, an oracle) $O$ the can learn a class $\mathcal{H}$ to precision $\epsilon_0 = O(1)$, where $0 < \epsilon_0 < 1/2$. The $O(1)$ notation here means some positive constant, but arbitrary.

We require that the weak learner can do better than half, as otherwise the task is trivial. In other words, the learner does slightly better than random guessing. Can we construct (in the presence of such weak learner) a learning algorithm that learns to precision $\epsilon$? More formally, let us define weak learnability and what is a weak learner:

**Definition 7.1** (Weak learnability)**.** *A class $\mathcal{H}$ of hypotheses is said to be $\gamma$-weakly-learnable if the following holds. There exists a function $m : (0,1) \to \mathbb{N}$, and an algorithm $A$ that accepts a realizable (by $\mathcal{H}$) sample $S = \{(\mathbf{x}^{(i)}, y)\}_{i=1}^{m}$ and returns an hypothesis in $\mathcal{H}$, $h_S^A$, that satisfies:*
*if $m > m(\delta)$ and $S$ is an IID sequence from some arbitrary distribution, the with probability $1 - \delta$,*

$$err(h_S^A) \leq \frac{1}{2} - \gamma$$

This is an apparent weakening of the definition of statistical learnability. An algorithm that achieves weak

learning is referred to as a weak learner, and respectively we can refer to a strong learner as an algorithm that attains statistical learning for a certain concept class. We will show that weak and strong learnability are equivalent.

The AdaBoost algorithm Due to R.Schapire & Y.Freund [2], is probably one of the most useful and successful algorithms in Machine Learning, both from practical perspective as well as for several theoretical results.

We will present here a simplified version of the original algorithm. The proof we will give here follows the survey of Arora et al. on the *multiplicative weight algorithm* [1]. It relies on a framework called "prediction with expert advice". The advantage of the proof is that it will allow us to present the two problems in a unified manner. Further, it will allow us to make a first step in the field of what is known as *online learning*

### 7.1.1 Interlude– Prediction with Expert Advice and the Multiplicative Weight algorithm

We now forget, for a minute, the problem of Boosting and we focus on a seemingly different setting:

**Expert Advice** In the expert advice setting, we consider the following sequential game between a decision maker and some adversary.

The game proceeds for $T$ rounds, at each iteration $t \leq T$ the decision maker has to choose an advice from one of $n$ experts. The decision maker can choose the expert randomly. Specifically, at each iteration $t$ the decision maker chooses a distribution $p_t$ over the space of experts $E = \{e_1, \ldots, e_n\}$. Then, given $p_t$ the adversary chooses a loss function $\ell_t : E \to [0, 1]$ from the space of experts to the unit interval.

At the end of each round, the decision maker observes the loss function and suffers a loss according to her random choice of expert and the loss function chosen by the adversary. Specifically the expected loss of the decision maker at round $t$ is given by

$$\mathop{\mathbb{E}}_{e_i \sim p_t}[\ell_t(e_i)] = \sum_{i=1}^{n} p_t(e_i) \cdot \ell_t(e_i) = p_t \cdot \ell_t.$$

The aim of the learner is to minimize her *regret*, which is defined to be the difference between her expected

loss and the loss of the best performing expert *in hindsight*:

$$\text{REGRET} = \left(\sum_{t=1}^{T} p_t \cdot \ell_t\right) - \min_i \sum_{t=1}^{T} \ell_t(i).$$

We next present the Multiplicative Weight (also known as Hedge) algorithm that achieves regert $O(\sqrt{T \log n})$:

**Algorithm 1** Multiplicative Weight

**Inititalization** a Weight vector $W_1 = \mathbf{1} \in \mathbf{R}^n$ % $W_1 = (1, 1, \ldots, 1)$.

Set $p_1(i) = \frac{1}{n} W_1(i), \quad i = 1, \ldots, n.$

**for** $t = 1, 2 \ldots T$ **do**

Pick $i_t \propto p_t$. i.e. $p(i_t = i) = p_t(i)$.

incure loss $\ell_t(i_t)$.

Update Weights $W_{t+1}(i) = W_t(i) \cdot e^{-\epsilon \ell_t(i)} \quad i = 1, \ldots, n$ (i)

Set $p_{t+1}(i) = \frac{W_{t+1}(i)}{\sum_j W_{t+1}(j)} \quad i = 1, \ldots, n.$

**end for**

**return**

**Lemma 7.2.** *Let $\ell^2$ denote the $n$-dimensional vector of pointwise square losses (i.e $\ell^2(i) = (\ell(i))^2$), let $\epsilon > 0$ and assume all losses $\ell$ to be non-negative. The Hedge Algorithm satisfies for every expert $i^*$:*

$$\left(\sum_{t=1}^{T} p_t \cdot \ell_t\right) - \sum_{t=1}^{T} \ell_t(i^*) \leq \epsilon \sum_{t=1}^{T} p_t \cdot \ell_t^2 + \frac{\log n}{\epsilon} \tag{7.1}$$

*Proof.* Set $\Phi_t = \sum_{i=1}^{n} W_t(i)$ for all $t \leq T$, and note that $\Phi_1 = n$.

Inspecting the sum of weights

$$\begin{aligned}
\Phi_{t+1} &= \sum W_{t+1} \\
&= \sum_i W_t(i) e^{-\epsilon \ell_t(i)} \\
&= \Phi_t \sum_i p_t(i) e^{-\epsilon \ell_t(i)} && p_t(i) = \frac{W_t(i)}{\sum W_t(j)} \\
&\le \Phi_t \sum_i p_t(i)(1 - \epsilon \ell_t(i) + \epsilon^2 \ell_t^2(i)) && \forall x \ge 0,\ e^{-x} \le 1 - x + x^2 \\
&= \Phi_t (1 - \epsilon p_t \cdot \ell_t + \epsilon^2 p_t \cdot \ell_t^2) \\
&\le \Phi_t e^{-\epsilon p_t \cdot \ell_t + \epsilon^2 p_t \cdot \ell_t^2} && 1 + x \le e^x
\end{aligned}$$

And by definition of expert $i^*$ we have that: $W_{T+1}(i^*) = e^{-\epsilon \sum_{t=1}^{T} \ell_t(i^*)}$. Given that $W_{T+1}(i^*)$ is less then the sum of weights $\Phi_{T+1}$, we have that

$$W_{T+1}(i^*) \le \Phi_{T+1} \le n e^{-\epsilon \sum p_t \cdot \ell_t + \epsilon^2 \sum p_t \cdot \ell_t^2}$$

Taking logarithm of both sides we get:

$$-\epsilon \sum_{t=1}^{T} \ell_t(i^*) \le \log n - \epsilon \sum_{t=1}^{T} p_t \cdot \ell_t + \epsilon^2 \sum_{t=1}^{T} p_t \cdot \ell_t^2.$$

And the result follows be rearranging terms. □

**Theorem 7.3.** *Apply Alg. 1 to the Online Expert problem, with* $\epsilon = \sqrt{\frac{\log n}{T}}$ *then*

$$\mathit{Regret}_T = O(\sqrt{T \log n})$$

*Proof.* First observe that $\ell^2 \le 1$ hence $p_t \cdot \ell_t^2 \le 1$. Plugging this into Eq. 7.1 we obtain that for every $i^*$:

$$\sum p_t \cdot \ell_t - \ell_t(i^*) \le T\epsilon + \frac{\log n}{\epsilon}$$

The algorithm picks the action of expert $i$ at iteration $i$ according to $p_t$ hence incurs expected loss of $p_t \cdot \ell_t$

overall we have that for our choice of $\epsilon$:

$$\text{Regret}_T \leq T\epsilon + \frac{\log n}{\epsilon} \leq 2\sqrt{T \log n}$$

□

We are now ready to present a simplified version of the ada–Boost algorithm. We will require the following notation: Given a sample $S = \{(x_i, y_i)\}_{i=1}^m$ and a distribution vector $p \in [0,1]^m$ (i.e. $\sum_{i=1}^m p(i) = 1$), we will denote $p(S)$ the distribution that shows example $i$ with probability $p(i)$.

**Algorithm 2** ada-Boost

**Input**: A sample $S = \{(x_i, y_i)\}_{i=1}^m$ access to a $\gamma$-weak learner algorithm for the class $H$, $\mathcal{O}$.

**Inititalization** a Weight vector $W_1 = 1 \in \mathrm{R}^n$ % $W_1 = (1, 1, \ldots, 1)$

SET $T = 4 \log |S| / \gamma^2$

SET $\epsilon = \sqrt{\frac{\log m}{T}}$

SET $p_1 = \frac{1}{n} W_1$.

**for** $t = 1, 2 \ldots T$ **do**

Receive $h_t = \mathcal{O}(p_t(S))$.

Set $\ell_t(i) = 1[h_t(x_i) = y_i]$.

Update Weights $W_{t+1}(i) = W_t(i) \cdot e^{-\epsilon \ell_t(i)}$

Set $p_{t+1}(i) = \frac{W_{t+1}}{\sum_j W_t(j)}$.

**end for**

**return** **MAJ**$(h_1, \ldots, h_T)$

**Theorem 7.4.** *Let $\mathcal{H}$ be a class, and consider algorithm 2. Run it on any realizable sample $S$ (i.e. there exists $h \in \mathcal{H}$ such that $h(x_i) = y_i$).*

*Let $h_1, \ldots, h_T$ be the different hypotheses outputted by ada-Boost, and set $h_S^{ada} =$ **MAJ**$(h_1, \ldots, h_T)$ to be the predictor that outputs*

$$\boldsymbol{MAJ}(h_{1,\ldots,T})[x] = \begin{cases} 1 & \sum 1[h_t(x) = 1] \geq T/2 \\ 0 & \text{else} \end{cases}.$$

*Suppose that at each iteration the output of $\mathcal{O}$ is a $\gamma$-weak hypothesis for the distribution $p_t(S)$. i.e.*

$$\mathop{\mathbb{E}}_{(x,y)\sim p_t(S)}[\mathbf{1}[h_t(x) = y] \geq \frac{1}{2} + \gamma.$$

*Then*

$$err_S(h_S^{ada}) = 0$$

*Proof.* We basically interpret adaBoost as a sequential game between a decision maker and the weak learner as follows: The deicision maker is searching for a distribution $p_t$ that would make the weak learner "fail": Specifically, the decision maker chooses at each iteration a distribution $p_t$ over the sample $S$ and an adversary chooses a hypothesis $h_t$: the decision maker suffers a loss 1 on any point that the adversary labels correctly.

In terms of expert advice, we think of the sample points $(x_1, y_1), \ldots, (x_T, y_T)$ as experts. Each expert succeed, if the learner misclassified it (you can think of it as if the decision maker is searching the "hard examples").

The decision maker chooses a distribution $p_t$ and each "expert" suffers loss

$$\ell_t(i) = \mathbf{1}[h_t(x_i) = y_i].$$

With this interpretation, one can observe that the strategy chosen by the decision maker in adaboost is exactly the MW algorithm. It thus follows that

$$\sum_{t=1}^{T} p_t \cdot \ell_t - \min_i \sum_{t=1}^{T} \ell_t(i) \leq 2\sqrt{T \log m}$$

On the other hand, we assume that for every $t$:

$$p_t \cdot \ell_t \geq \frac{1}{2} + \gamma.$$

In particular $\sum_{t=1}^{T} p_t \cdot \ell_t \geq T(\frac{1}{2} + \gamma)$.

Finally, we want to show that for any sample $x_i, y_i$ we have that $h_S^{ada}(x_i) = y_i$, by the definition of $\ell_t$ this would be true if for every $i$ $\sum_{t=1}^{T} \ell_t(i) \geq \frac{T}{2}$.

Assume to the contrary, namely that for some $i$ we have that $\sum_{t=1}^{T} \ell_t(i) < \frac{T}{2}$

$$\begin{aligned}(\frac{1}{2}+\gamma)T &\leq \sum p_t \cdot \ell_t \\ &\leq \min_i \sum_{t=1}^{T} \ell_t(i) + 2\sqrt{T \log m} \\ &\leq \frac{1}{2}T + 2\sqrt{T \log m}\end{aligned}$$

Rearranging we obtain the

$$\gamma T \leq 2\sqrt{T \log m}$$

Equivalently

$$T \leq 4\frac{\log m}{\gamma^2}$$

□

### 7.1.2 Generalization Bound

We so far focused on the training error of ada–boost. In particular we showed that given a realizable sample, the algorithm outputs a hypothesis that has 0 training error. Note that ada–boost is *improper*: It returns a function of the form $\text{MAJ}(h_1, \ldots, h_T)$. Thus, if we want to show that adaboost generalizes, we need to give generalization guarantee for a new class. You will prove the following statement in your exercise

**Theorem 7.5.** *Let $\mathcal{H}$ be a class with VC dimension $d$, and let $H_T$ be a class of the form*

$$\mathcal{H}_T = \{\text{MAJ}(h_1, \ldots, h_T)\} : h_t \in \mathcal{H} \ t = 1, \ldots, T\}.$$

*Then* $\text{VCdim}\mathcal{H}_T \leq dT \log T$.

In the exercise you will also be required to compute a sample complexity bound for adaboost

**Corollary 7.6.** *Suppose we run ada–boost on a realizable sample $S$ drawn IID according to somae unknown distribution with access to a $\gamma$-weak learner. There exists a sample complexity bound $m(\epsilon, \delta, \gamma)$ such that if $|S| > m(\epsilon, \delta, \gamma)$, then w.p. at least $(1-\delta)$ the output hypothesis of $h_S^{ada} = \text{MAJ}(h_1, \ldots, h_T)$ has $err(h_S^{ada}) < \epsilon$.*

## References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[2] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.