Boosting

Uri Shaham

May 31, 2026

In the previous lectures, the implicit assumption throughout was that we are working with a *fixed* class $\mathcal{H}$ that is rich enough to contain a good hypothesis.

Today we ask a different question: what if no single hypothesis in $\mathcal{H}$ performs well, but many hypotheses each perform *slightly* better than random guessing? Can we combine them into something powerful?

This is the central question of **boosting**. The answer, remarkably, is yes — and the combination can achieve arbitrarily small training error, and (with the right analysis) good generalization.

1 Weak and Strong Learning

PAC learnability is a strong requirement: it demands driving $L_{\mathcal{D}}(h)$ below any $\varepsilon > 0$. A strictly weaker notion turns out to be the right building block for boosting.

1.1 Weak Learning

Definition 1.1 (Weak Learner)**.** *An algorithm A is a **weak learner** for a hypothesis class $\mathcal{H}$ if there exists a fixed constant $\gamma > 0$ and a polynomial* $\mathrm{poly}(\cdot)$ *such that for every distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$ and every $\delta > 0$: given $n \geq \mathrm{poly}(1/\delta)$ samples, A outputs h satisfying*

$$L_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$$

with probability at least $1 - \delta$.

The parameter $\gamma > 0$ is the **edge** of the weak learner. A weak learner only needs to beat random guessing by a fixed margin γ — it makes no guarantee about how small the error can be made. In many applications, simple and computationally cheap hypothesis classes (e.g., decision stumps: threshold classifiers on a single feature) are weak learners.

1.2 The Boosting Question

The question Schapire (1990) and Freund (1995) asked and answered:

Is weak learnability equivalent to strong (i.e., PAC) learnability?

A priori this seems unlikely. Weak learning only guarantees a small advantage over random guessing; strong learning requires arbitrarily accurate classifiers. Yet the answer is **yes**, and the proof is constructive: boosting algorithms explicitly convert any weak learner into a strong learner.

Theorem 1.2 (Boosting Theorem, informal)**.** *If $\mathcal{H}$ is weakly learnable with edge $\gamma > 0$, then $\mathcal{H}$ is strongly learnable. Moreover, the strong learner makes $T = O\Big(\frac{\log(1/\varepsilon)}{\gamma^2}\Big)$ calls to the weak learner and produces a majority vote of T hypotheses.*

The proof of this theorem is the AdaBoost algorithm, to which we now turn.

Remark 1.3. *The equivalence is non-trivial. One direction is trivial: a strong learner is also a weak learner. The hard direction — weak implies strong — is what boosting establishes.*

2 The AdaBoost Algorithm

2.1 High-Level Idea

The key challenge in converting a weak learner into a strong one is that the weak learner's guarantee holds for *any* distribution. Boosting exploits this by constructing a sequence of distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots$ over the training sample, each designed to focus the weak learner on the examples that previous hypotheses got wrong.

Intuition. At each round t, run the weak learner on the current distribution $\mathcal{D}_t$ to get h_t. Increase the weight of examples that h_t misclassified (these are the "hard" examples) and decrease the weight of correctly classified ones. Repeat. The final classifier is a weighted majority vote of all $h_1, \ldots, h_T$.

2.2 The Algorithm

Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a training sample. AdaBoost maintains a weight vector $\mathbf{w}^{(t)} \in \mathbb{R}^m_{\geq 0}$ over examples, inducing a distribution $\mathcal{D}_t(i) = w_i^{(t)} / \sum_j w_j^{(t)}$.

Algorithm 1 AdaBoost

Require: Training sample $S = \{(x_i, y_i)\}_{i=1}^m$, weak learner $\mathcal{A}$, rounds T

1: Initialize weights $w_i^{(1)} = 1/m$ for all $i \in [m]$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Set distribution $\mathcal{D}_t(i) = w_i^{(t)} / \sum_j w_j^{(t)}$
4: Call weak learner: $h_t \leftarrow \mathcal{A}(S, \mathcal{D}_t)$
5: Compute weighted error: $\varepsilon_t = \sum_{i=1}^m \mathcal{D}_t(i) \cdot \mathbf{1}[h_t(x_i) \neq y_i]$
6: Compute voting weight: $\alpha_t = \dfrac{1}{2} \ln\!\left(\dfrac{1 - \varepsilon_t}{\varepsilon_t}\right)$
7: Update example weights:

$$w_i^{(t+1)} = w_i^{(t)} \cdot \exp\big(-\alpha_t \, y_i \, h_t(x_i)\big)$$

8: **end for**
9: **return** $H(x) = \operatorname{sign}\Big(\sum_{t=1}^T \alpha_t h_t(x)\Big)$

2.3 Dissecting the Update Rule

Several features of Algorithm 1 deserve comment.

The voting weight α_t. Since the weak learner guarantees $\varepsilon_t \le 1/2 - \gamma$, we have $\varepsilon_t < 1/2$, so $\alpha_t > 0$. More accurate hypotheses receive larger weights in the final vote. As $\varepsilon_t \to 0$, $\alpha_t \to \infty$; as $\varepsilon_t \to 1/2$, $\alpha_t \to 0$.

The weight update. For a correctly classified example ($y_i h_t(x_i) = +1$):

$$w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\alpha_t}.$$

Since $\alpha_t > 0$, the weight *decreases* — easy examples are down-weighted. For a misclassified example ($y_i h_t(x_i) = -1$):

$$w_i^{(t+1)} = w_i^{(t)} \cdot e^{+\alpha_t}.$$

The weight *increases* — the next weak learner is forced to pay attention to these.

The final classifier. $H(x) = \text{sign}(\sum_t \alpha_t h_t(x))$ is a weighted majority vote. The function $F(x) = \sum_t \alpha_t h_t(x)$ is the **margin**: larger $|F(x)|$ indicates more confident classification.

3 The Boosting Theorem

Theorem 3.1 (Adaboost drives the training error to zero)**.** *Assume that at each iteration of AdaBoost, the weak learner returns a hypothesis for which $\varepsilon_t \le 1/2 - \gamma$. Then After T calls to the weak learner,*

$$L_S(H) \le e^{-2\gamma^2 T}.$$

Proof. We proceed in three steps as follows.

Step 1: Reducing training error to a product of normalization factors. Let $W^{(t)} = \sum_{i=1}^m w_i^{(t)}$ denote the total weight at round t, so $W^{(1)} = 1$. Define the per-round normalization factor

$$Z_t \;=\; \sum_{i=1}^m D_t(i)\, e^{-\alpha_t y_i h_t(x_i)}.$$

Lemma 3.2. $W^{(T+1)} = \prod_{t=1}^T Z_t$.

Proof. By the weight update rule,

$$W^{(t+1)} = \sum_i w_i^{(t+1)} = \sum_i w_i^{(t)}\, e^{-\alpha_t y_i h_t(x_i)} = W^{(t)} \sum_i D_t(i)\, e^{-\alpha_t y_i h_t(x_i)} = W^{(t)} \cdot Z_t.$$

Telescoping from $t = 1$ to T and using $W^{(1)} = 1$ gives $W^{(T+1)} = \prod_{t=1}^T Z_t$. □

Unrolling all T update steps from the initialization $w_i^{(1)} = 1/m$,

$$w_i^{(T+1)} = \frac{1}{m}\, \exp\Big(-y_i\, F(x_i)\Big), \qquad \text{where } F(x) = \sum_{t=1}^T \alpha_t h_t(x).$$

If $H(x_i) \ne y_i$, then $\text{sign}(F(x_i)) \ne y_i$, so $y_i F(x_i) \le 0$, and therefore $e^{-y_i F(x_i)} \ge 1$. Hence

$$L_S(H) = \frac{1}{m}\sum_{i=1}^m \mathbf{1}[H(x_i) \ne y_i] \le \frac{1}{m}\sum_{i=1}^m e^{-y_i F(x_i)} = W^{(T+1)} = \prod_{t=1}^T Z_t.$$

Step 2: Closed-form expression for Z_t. Splitting the sum into correctly and incorrectly classified examples,

$$Z_t = (1-\varepsilon_t)\, e^{-\alpha_t} + \varepsilon_t\, e^{+\alpha_t}.$$

Substituting the chosen $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$, which gives $e^{-\alpha_t} = \sqrt{\varepsilon_t/(1-\varepsilon_t)}$ and $e^{+\alpha_t} = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$:

$$Z_t = (1-\varepsilon_t)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}.$$

Remark 3.3. *The choice of α_t is not arbitrary: it is precisely the minimizer of Z_t as a function of α, obtained by differentiating $(1-\varepsilon_t)e^{-\alpha} + \varepsilon_t e^{+\alpha}$ and setting it to zero. AdaBoost therefore performs a greedy coordinate descent on $\prod_t Z_t$, minimizing the training error bound one round at a time.*

Step 3: Applying the weak learning condition. The weak learner guarantees $\varepsilon_t \le \frac{1}{2} - \gamma$. Thus

$$\varepsilon_t(1-\varepsilon_t) \le \Big(\tfrac{1}{2}-\gamma\Big)\Big(\tfrac{1}{2}+\gamma\Big) = \tfrac{1}{4} - \gamma^2.$$

Therefore,

$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \le 2\sqrt{\tfrac{1}{4}-\gamma^2} = \sqrt{1-4\gamma^2}.$$

Using the inequality $\sqrt{1-x} \le e^{-x/2}$ (a consequence of $1-x \le e^{-x}$) with $x = 4\gamma^2$:

$$Z_t \le e^{-2\gamma^2}.$$

Conclusion of Part I.

$$L_S(H) \le \prod_{t=1}^{T} Z_t \le e^{-2\gamma^2 T}.$$

Setting $T \ge \frac{\ln(1/\varepsilon)}{2\gamma^2}$ drives the training error below ε.

□

Generalization

Driving training error to zero does not by itself imply learnability; we need to show that the class of functions corresponding to AdaBoost has the uniform convergence property.

To see that, we define the class of all classifiers that AdaBoost can produce after T rounds:

$$\mathcal{F}_T = \left\{ x \mapsto \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) : h_t \in \mathcal{H},\ \alpha_t \ge 0 \right\}.$$

Any $f \in \mathcal{F}_T$ is determined by a T-tuple $(h_1, \ldots, h_T) \in \mathcal{H}^T$. By Sauer's lemma, the number of distinct dichotomies any single $h \in \mathcal{H}$ can induce on m points is at most

$$\tau_{\mathcal{H}}(m) \le \Big(\frac{em}{d}\Big)^d,$$

where $d = \mathrm{VCdim}(\mathcal{H})$. AdaBoost selects T hypotheses, and there are at most $\big(\frac{em}{d}\big)^{dT}$ ways to do so. The sign of any fixed weighted sum $\sum_t \alpha_t h_t(x_i)$ is fully determined by the T-tuple of labelings $(h_1(x_i), \ldots, h_T(x_i)) \in \{-1,+1\}^T$. Hence, the growth function of $\mathcal{F}_T$ satisfies

$$\tau_{\mathcal{F}_T}(m) \le \tau_{\mathcal{H}}(m)^T \le \Big(\frac{em}{d}\Big)^{dT}.$$

This gives polynomial growth, which, in turn, implies uniform convergence.

4 Reading

- UML ch. 7
- https://www.cs.cmu.edu/~ninamf/ML11/lect1117.pdf