Week 6: Weak Learning, Boosting, and Compression

AdaBoost, sample compression, and the road to scale-sensitive complexity

Tianhao Wang
tianhaowang@ucsd.edu

UCSD · Spring 2026
DSC 190/291 Topics: Learning Theory

Contents

Bridge from Week 5

Where we left off

Classical neural-net theory

- polynomial-size networks express any **polynomial-time computation**
- VC dimension scales with **parameter count**
- worst-case learning is **NP-hard** and **crypto-hard**, even in the realizable case.

What classical theory does not explain

- gradient-based methods often work on real data;
- models generalize even when **capacity bounds** are vacuous;
- regularization depends on **scale**, **norm**, and **margin**.

- For some natural classes, strong PAC accuracy is unreachable; we need a weaker target.
- Today's move: relax the accuracy demand on a learner.

The new question

Two questions structure today's lecture:

1. **Weak learning:** what's a useful relaxation of the accuracy demand?
   - Replace "error $\leq \epsilon$ for every ϵ" with "error $\leq \frac{1}{2} - \gamma$ for one fixed $\gamma > 0$".
2. **Boosting:** if a learner meets that weaker demand, can we amplify it?
   - Equivalently, are weak and strong learning the same problem?

> Today's central question: can a fixed edge over chance be amplified to arbitrary accuracy?

Plan for today

1. Weak learning

Formalizing learners that are only slightly better than random.

2. AdaBoost

Reweighting hard examples, weighted votes, and exponential training-error decay.

3. Optimization view

Boosting as coordinate descent on the exponential loss.

4. Compression

Generalization from compact descriptions of the learner.

Looking ahead next week: real-valued losses and scale-sensitive complexity.

Weak Learning and the Boosting Problem

Strong PAC vs weak PAC

Realizable assumption. Some $h^* \in \mathcal{H}$ achieves $L_{\mathcal{D}}(h^*) = 0$.

Strong PAC learning. A rule A strongly learns $\mathcal{H}$ if for every $\epsilon, \delta > 0$ there is $n(\epsilon, \delta)$ such that, for every realizable $\mathcal{D}$,

$$\mathbb{P}_{S \sim \mathcal{D}^{n(\epsilon,\delta)}}[L_{\mathcal{D}}(A(S)) \leq \epsilon] \geq 1 - \delta.$$

Weak PAC learning. A rule A weakly learns $\mathcal{H}$ if there exist fixed constants $\gamma > 0$, $\delta_0 < 1$, and n_0 such that, for every realizable $\mathcal{D}$,

$$\mathbb{P}_{S \sim \mathcal{D}^{n_0}}\left[L_{\mathcal{D}}(A(S)) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta_0.$$

Strong learning must work for every ϵ, δ; weak learning fixes one γ, δ_0, n_0.

Setup: stumps and rectangles

Target class. Axis-aligned rectangles in $\mathbb{R}^2$ (positive inside, negative outside).

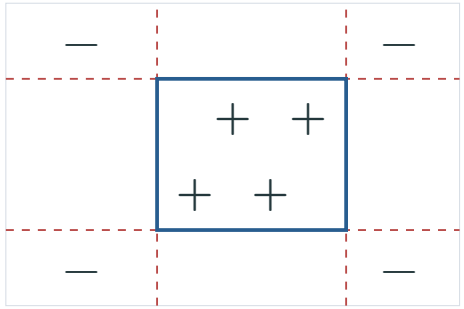
Base class $\mathcal{B}$ (the weak learner's hypothesis class).

- **Stump:** one axis-aligned cut, $\mathsf{sign}\big(s(x_j - \theta)\big)$.
- $j \in \{1, 2\}$: which axis; θ: cut location; s: which side is $+1$.
- Plus the constant negative rule.

$$\mathcal{B} = \big\{x \to \mathsf{sign}\big(s \cdot (x_j - \theta)\big) : j \in \{1, 2\}, s \in \{-1, +1\}, \theta \in \mathbb{R}\big\} \cup \{-1\}.$$

Geometric fact.

- Each rectangle side = boundary of one **bounding stump**.
- That stump puts the rectangle on its positive side.
- 4 sides $\to$ 4 bounding stumps.
- Rectangle = **intersection** of 4 positive halfplanes.



Blue rectangle = intersection of 4 red bounding stumps.

$\mathcal{B}$ is much simpler than rectangles, yet rich enough to cover any rectangle's geometry.

Decision stumps weakly learn rectangles

Let $p = \mathbb{P}_{\mathcal{D}}[y = +1]$.

Case 1 ($p \leq \frac{3}{7}$).
- Predict -1 everywhere.
- Every negative right, every positive wrong $\rightarrow$ error $= p \leq \frac{3}{7}$.

Case 2 ($p > \frac{3}{7}$). Take the 4 bounding stumps.
- Every positive: inside $\rightarrow$ correct on all 4.
- Every negative: outside $\rightarrow$ misclassified by at most 3 of 4.
- Sum of **population** error rates over 4 stumps $\leq 3(1-p) \rightarrow$ best has error $\leq 3\frac{1-p}{4} < \frac{3}{7}$.

Weak-learning guarantee. Take ERM A over $\mathcal{B}$. For some constants n_0, δ_0 and every realizable rectangle-$\mathcal{D}$:

$$\mathbb{P}_{S \sim \mathcal{D}^{n_0}}\left[L_{\mathcal{D}}(A(S)) \leq \frac{1}{2} - \frac{1}{14}\right] \geq 1 - \delta_0.$$

So $\mathcal{B}$ weakly learns rectangles with $\gamma = \frac{1}{14}$.

From weak to strong: the central question

Question. Are weak and strong PAC learnability the same notion?

Statistically: yes, trivially.

- strong $\to$ weak: any $\epsilon < \frac{1}{2}$ gives an edge γ;
- weak $\to$ strong: weak learnability forces $\mathrm{VCdim}(\mathcal{H}) < \infty$, then ERM strongly learns;
- so {weakly learnable} = {strongly learnable} = {finite VC}.

Computationally: not obvious.

- given an **efficient** weak learner A, can we build an **efficient** strong learner that uses A as a black box?
- "efficient" = polynomial in n, $\frac{1}{\epsilon}$, $\log\left(\frac{1}{\delta}\right)$, and the cost of one call to A.

As classes they coincide; whether the algorithms can be efficiently upgraded is the real question.

Why study weak learning?

If weak and strong learning define the same classes, why bother with weak learning?

Equivalent as classes, different as engineering objects.

Strong learner

- drives error to arbitrary ϵ;
- often a sophisticated algorithm;
- needs its own design per class.

Weak learner

- just beats chance by a fixed γ;
- often a one-liner (e.g., a decision stump);
- one recipe upgrades it for every class.

Three payoffs:

- **easier construction:** one stump shows rectangles are weak-learnable; a strong learner is harder to design directly.
- **sharper hardness:** prove weak hardness, get strong hardness for free.
- **modular ensembles:** this design pattern underlies many widely-used ML algorithms.

Weak learning is the right unit of *design* the upgrade to a strong learner is mechanical.

The boosting problem

Now make the algorithmic question precise, in the realizable PAC model.

Setup. An efficient weak learner A for $\mathcal{H}$:

- constants $\gamma > 0$, $\delta_0 < 1$, n_0;
- on every realizable $\mathcal{D}$, with n_0 samples, A returns h with $L_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$ w.p. $\geq 1 - \delta_0$.

Goal. A strong learner A^* that, on every realizable $\mathcal{D}$ and every $\epsilon, \delta > 0$:

- calls A as a black box;
- achieves $L_{\mathcal{D}}(A^*(S)) \leq \epsilon$ w.p. $\geq 1 - \delta$;
- uses sample size and runtime polynomial in $\frac{1}{\epsilon}$, $\log\left(\frac{1}{\delta}\right)$, n_0, $\frac{1}{\gamma}$, $\log\left(\frac{1}{\delta_0}\right)$, and the cost of one call to A.

Two sub-problems:

- **confidence boosting**: amplify $\delta_0 < 1$ to any $\delta > 0$;
- **error boosting**: amplify the edge γ to arbitrary accuracy ϵ.

Today: solve both, with only polynomial overhead.

Confidence boosting: step 1 (multiple runs)

Goal. Amplify success probability from $1 - \delta_0$ to $1 - \delta$, for any $\delta > 0$.

Setup. A achieves error $\leq \frac{1}{2} - \gamma$ but only with probability $1 - \delta_0$.

S_1	S_2	$\cdots$	**validation**
run A; get h_1	run A; get h_2	k independent runs	choose best h_i

Step 1. Run A on k independent samples $S_1, \ldots, S_k$ to get $h_1, \ldots, h_k$.

By independence, all k runs fail (every h_i bad) with probability at most δ_0^k.

Force this $\leq \delta/2$:

$$\delta_0^k \leq \delta/2 \quad \Rightarrow \quad k \geq \log(2/\delta)/\log(1/\delta_0) = O(\log(1/\delta)/\log(1/\delta_0)).$$

With probability $\geq 1 - \delta/2$, at least one h_i has true error $\leq \frac{1}{2} - \gamma$.

Confidence boosting: step 2 (validation)

S_1	S_2	$\cdots$	**validation**
run A; get h_1	run A; get h_2	k independent runs	choose best h_i

From step 1. With probability $\geq 1 - \delta/2$, at least one h_i has true error $\leq \frac{1}{2} - \gamma$.

Step 2. Draw a fresh validation sample S_v, compute $\hat{L}(h_i)$ on S_v, return $\hat{h}$ with smallest $\hat{L}$.

Hoeffding at deviation $\gamma/4$: $\mathbb{P}\left[|\hat{L}(h_i) - L_{\mathcal{D}}(h_i)| > \gamma/4\right] \leq 2\exp(-|S_v|\ \gamma^2/8)$.

Union bound over k and force total failure $\leq \delta/2$ gives $|S_v| = O(\log(k/\delta)/\gamma^2)$.

Combining. Validation estimates each h_i's true error to within $\pm\gamma/4$. Picking the empirical winner can be worse than the truly best h^* by at most $\gamma/4 + \gamma/4 = \gamma/2$ in true error, giving $L_{\mathcal{D}}\left(\hat{h}\right) \leq \frac{1}{2} - \gamma/2$.

Total samples. $kn_0 + |S_v| = O((n_0 + 1/\gamma^2)\log(1/\delta))$ (with δ_0 fixed).

Confidence boosting is easy: with prob $\geq 1 - \delta$, $\hat{h}$ has error $\leq \frac{1}{2} - \gamma/2$.

Error boosting is the real problem

- **Confidence boosting** was just selection over k runs.
- **Error boosting** needs a new mechanism: the deep half of the problem.

Setup. Weak learner A: error $\leq \frac{1}{2} - \gamma$ on every realizable $\mathcal{D}$.

Baseline. The constant predictor already achieves error $\leq \frac{1}{2}$, so weak learning improves on chance only by the edge γ.

Question. Can we use A as a black box to drive error to arbitrary ϵ?

Today's answer. Yes, via **AdaBoost**: reweight the input distribution each round to push A toward what previous rounds missed.

The mechanism is reweighting.

AdaBoost

AdaBoost: setup and output

Input: sample $S = \{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{-1, +1\}$; weak learner A; rounds T.

AdaBoost maintains a **distribution** D^t over the n training examples; initialize $D_i^1 = 1/n$. We write $A(D^t)$ for the weak learner's output when called on D^t (next slide unpacks how).

Three objects per round.

- $h_t = A(D^t) \in \mathcal{H}$: the **weak rule** at round t;
- $f_t(x) = \sum_{s \le t} \alpha_s h_s(x)$: the unthresholded running sum (real-valued);
- $H_t(x) = \mathsf{sign}(f_t(x))$: the **ensemble classifier** through round t.

Each round produces h_t and a vote weight α_t, then updates $D^t \to D^{t+1}$ (next slide).

Output: H_T.

- **lowercase h_t is one weak rule; uppercase $H_t = \mathsf{sign}(f_t)$ is the running ensemble;**
- **only h_t requires a call to A; weights and votes are deterministic.**

Why can we call A on D^t?

- weak-PAC hypothesis: stated for samples from a realizable **population**;
- AdaBoost: feeds A a distribution D^t on the **training sample** S;
- does the hypothesis still apply?

Realizability transfers from $\mathcal{D}$ to D^t.

- $\mathcal{D}$ realizable: $L_{\mathcal{D}}(h^*) = 0$ for some $h^* \in \mathcal{H}$;
- D^t is supported on S, where h^* is correct, so $L_{D^t}(h^*) = 0$ and D^t is realizable;
- **idealized weighted ERM:** an oracle returning $\arg\min_{h \in \mathcal{H}} \sum_i D_i^t \mathbf{1}[h(x_i) \neq y_i]$ gets error $0 \leq 1/2 - \gamma$ on D^t. But our primitive is the sample-based weak learner A, not an ERM oracle.

How AdaBoost actually calls A on D^t.

- draw resamples from D^t (staying inside S), run A, validate, pick;
- call n_1 the sample budget of this high-confidence weak call;
- n_1 includes several n_0-sized weak runs plus validation resamples;
- weak-PAC on D^t gives $\varepsilon_t \leq 1/2 - \gamma$ w.p. $\geq 1 - \delta/(2T)$ per round;
- union bound: all T rounds succeed w.p. $\geq 1 - \delta/2$.

Resamples stay inside S: confidence boosting costs extra runs of A, not extra training data.

AdaBoost: per-round update

At round t with current distribution D^t:

h_t	$= A(D^t)$	**weak-PAC call** on D^t: $L_{D^t}(h_t) \le \frac{1}{2} - \gamma$ $A(D^t) \in \arg\min_{h \in \mathcal{H}} \sum_{i=1}^n D_i^t \mathbf{1}[h(x_i) \neq y_i]$ [idealized form]
ε_t	$= \sum_{i=1}^n D_i^t \mathbf{1}[h_t(x_i) \neq y_i]$	weighted error of h_t under D^t
α_t	$= \frac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$	**vote weight**: larger when ε_t is smaller
Z_t	$= \sum_{j=1}^n D_j^t \exp(-\alpha_t y_j h_t(x_j))$	normalizer; makes D^{t+1} a distribution
D_i^{t+1}	$= D_i^t \exp(-\alpha_t y_i h_t(x_i))/Z_t$	**reweight**: mistakes get more mass, correct points less

Edge cases. If $\varepsilon_t = 0$, return h_t; if $\varepsilon_t \ge \frac{1}{2}$, replace h_t with $-h_t$.

Weight update intuition

Update factor: $\exp(-\alpha_t y_i h_t(x_i))$. The sign $y_i h_t(x_i) \in \{-1, +1\}$ flags correct vs. mistake.

Correct point

$y_i h_t(x_i) = +1$

$D_i^{t+1} \propto D_i^t \cdot e^{-\alpha_t}$

weight decreases

Mistake

$y_i h_t(x_i) = -1$

$D_i^{t+1} \propto D_i^t \cdot e^{+\alpha_t}$

weight increases

Since $\alpha_t > 0$, mistakes are **upweighted** and correct points **downweighted**.

D^{t+1} concentrates on what h_t got wrong.

After reweighting: h_t has error $\frac{1}{2}$

Recall the weighted ERM:

$$h_t = A(D^t) \in \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} D_i^t \mathbf{1}[h(x_i) \neq y_i], \quad L_{D^t}(h_t) = \varepsilon_t.$$

Reweighting upweights its mistakes. What is $L_{D^{t+1}}(h_t)$?

Step 1. From $\alpha_t = \frac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$, we get $e^{\alpha_t} = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$ and $e^{-\alpha_t} = \sqrt{\varepsilon_t/(1-\varepsilon_t)}$.

Step 2. Split Z_t over mistakes (D^t-mass ε_t, factor e^{α_t}) and correct points ($1-\varepsilon_t$, factor $e^{-\alpha_t}$):

$$Z_t = \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t)e^{-\alpha_t} = \mathbf{2\sqrt{\varepsilon_t(1-\varepsilon_t)}}.$$

Step 3. Each mistake's weight scales by e^{α_t}/Z_t, so

$$L_{D^{t+1}}(h_t) = \frac{\varepsilon_t e^{\alpha_t}}{Z_t} = \mathbf{\frac{1}{2}}.$$

The choice of α_t forces h_t to error $\frac{1}{2}$ on D^{t+1}; round $t+1$ must find a different rule.

AdaBoost analysis: Exponential potential

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

- **Observation.** Each round forces h_t to neutrality; the next round attacks different mistakes.
- **Question.** Does this compound into low training error?
- **Tool.** Track $L_S^{0/1}(H_T)$ via a smooth surrogate, the **exponential potential**:

$$L_S^{\exp}(f) = \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i f(x_i)).$$

Two facts to establish:

- **bound**: $L_S^{0/1}(H_T) \le L_S^{\exp}(f_T)$ (Step 1, next);
- **multiplicativity**: $L^{\exp}$ shrinks by Z_t per round (One-step identity).

Bound $L^{0/1}$ by $L^{\exp}$ next; later, multiplicative shrinkage gives the rate.

Step 1: $L^{0/1}(H_T) \le L^{\exp}(f_T)$

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

Verify the **bound**: $L^{0/1}(H_T) \le L^{\exp}(f_T)$. Recall $H_T(x) = \mathsf{sign}(f_T(x))$.

Key observation. If H_T misclassifies point i (so $H_T(x_i) \neq y_i$), the signs of y_i and $f_T(x_i)$ disagree, so $y_i f_T(x_i) \le 0$ and $\mathbf{\exp(-y_i f_T(x_i)) \ge 1}$.

So pointwise, $\mathbf{1}[H_T(x_i) \neq y_i] \le \exp(-y_i f_T(x_i))$ for every i (trivially ≤ 1 when correct, and $\le \exp(\dots)$ when wrong).

Averaging over i:

$$L_S^{0/1}(H_T) = \frac{1}{n}\sum_i \mathbf{1}[H_T(x_i) \neq y_i] \le \frac{1}{n}\sum_i \exp(-y_i f_T(x_i)) = L_S^{\exp}(f_T).$$

Training error hits zero once $L_S^{\exp}(f_T) < 1/n$.

One-step identity: $L^{\exp}(f_t) = L^{\exp}(f_{t-1}) \cdot Z_t$

Recursion. From $f_t = f_{t-1} + \alpha_t h_t$ (box):

$$L_S^{\exp}(f_t) = \frac{1}{n} \sum_i \exp(-y_i f_{t-1}(x_i)) \cdot \exp(-\alpha_t y_i h_t(x_i)).$$

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

Use D^t from the box.

- normalization $\sum_i D_i^t = 1$ pins the box's proportionality constant to $nL^{\exp}(f_{t-1})$;
- substituting, $L_S^{\exp}(f_t) = L_S^{\exp}(f_{t-1}) \cdot Z_t$, with $Z_t = \sum_i D_i^t \exp(-\alpha_t y_i h_t(x_i))$.

Closed form for Z_t.

- per-point: $\exp(-\alpha_t y_i h_t(x_i)) = e^{\alpha_t}$ if h_t wrong on i, $e^{-\alpha_t}$ if correct;
- group by case: $Z_t = \left(\sum_{i:\text{ wrong}} D_i^t\right) e^{\alpha_t} + \left(\sum_{i:\text{ correct}} D_i^t\right) e^{-\alpha_t} = \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t}$;
- $L^{\exp}(f_{t-1})$ is fixed at round t, so $\min_{\alpha_t} L^{\exp}(f_t) = L^{\exp}(f_{t-1}) \cdot \min_{\alpha_t} Z_t$;
- minimum at $\alpha_t = \frac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$ (matches box), giving $Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$.

AdaBoost greedily minimizes Z_t: each round picks α_t to drive the potential down maximally.

From error to edge

- **Iterate one-step identity**: $L_S^{\exp}(f_T) = \prod_{t=1}^{T} Z_t$ (with $L^{\exp}(f_0) = 1$ since $f_0 = 0$).
- **Goal**: bound $\prod_{t=1}^{T} Z_t$, so we need a uniform bound on Z_t.

Apply the weak-learning hypothesis. Since D^t is a (realizable) distribution on S, the hypothesis applies at every round:

- $\varepsilon_t = L_{D^t}(h_t) \leq \frac{1}{2} - \gamma$;
- equivalently, the per-round **edge** $\gamma_t = \frac{1}{2} - \varepsilon_t \geq \gamma$.

In edge form, $Z_t = \sqrt{1 - 4\gamma_t^2}$.

Larger **edge** $\Rightarrow$ smaller Z_t (faster decay).

$Z_t = \sqrt{1 - 4\gamma_t^2}$ **— Step 2 (next) bounds this by** $\exp(-2\gamma^2)$**.**

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

Step 2: $L^{\exp}(f_T) \le \exp(-2\gamma^2 T)$

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

Telescope. From $f_0 = 0$ and $L^{\exp}(f_0) = 1$, iterating $L^{\exp}(f_t) = L^{\exp}(f_{t-1}) \cdot Z_t$:

$$L_S^{\exp}(f_T) = \prod_{t=1}^{T} Z_t.$$

Per-round bound. From the edge form $Z_t = \sqrt{1-4\gamma_t^2}$ and $\gamma_t \ge \gamma$:

$$Z_t = \sqrt{1-4\gamma_t^2} \le \sqrt{1-4\gamma^2} \le \mathbf{\exp(-2\gamma^2)}.$$

(Last step: $1 + x \le e^x$ with $x = -4\gamma^2$, then square root.)

Combine. $L_S^{\exp}(f_T) \le \prod_{t=1}^{T} \exp(-2\gamma^2) = \exp(-2\gamma^2 T)$.

Training-error theorem

Chain Step 1 ($L^{0/1}(H_T) \le L^{\exp}(f_T)$) and Step 2 ($L^{\exp}(f_T) \le \exp(-2\gamma^2 T)$):

AdaBoost training-error decay

Suppose $\varepsilon_t \le \frac{1}{2} - \gamma$ for all t. Then

$$L_S^{0/1}(H_T) \le L_S^{\exp}(f_T) \le \mathbf{\exp(-2\gamma^2 T)}.$$

This is error boosting on the training set.

- each h_t has only edge γ over chance ($\varepsilon_t \le \frac{1}{2} - \gamma$);
- chaining T such rules compounds the edge into exponential decay;
- a fixed edge $\gamma > 0$ drives training error to 0 as $T \to \infty$;
- rate γ^2: smaller edge $\Rightarrow$ more rounds for a target accuracy.

Choosing the number of rounds

Two corollaries of the training-error theorem:

Training error $\leq \epsilon$

Set $\exp(-2\gamma^2 T) \leq \epsilon$ and solve for T:

$T = O(\log(1/\epsilon)/\gamma^2)$.

Training error $= 0$

Set $\exp(-2\gamma^2 T) < 1/n$ and solve for T:

$T = O(\log n/\gamma^2)$.

For zero error: $L_S^{0/1}(H_T)$ is a multiple of $1/n$, so once it is $< 1/n$ it must be 0.

More rounds give better empirical fit, but also a more complex ensemble; T becomes a **complexity parameter**.

All of this is **training-set** analysis. We still need a generalization argument — next.

Generalization with a finite-VC base class

Goal. Promote $L_S^{0/1}(H_T) = 0$ to small population error.

Function class. With $h_t \in \mathcal{B}$, the ensemble $H_T = \mathsf{sign}(f_T)$ lies in $\mathrm{Lin}(\mathcal{B}, T)$, the **sign-thresholded T-term linear combinations** over $\mathcal{B}$:

$$\mathrm{Lin}(\mathcal{B}, T) = \left\{ x \to \mathsf{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) : h_t \in \mathcal{B} \right\}.$$

Capacity. $\mathrm{VCdim}(\mathrm{Lin}(\mathcal{B}, T)) \leq \tilde{O}(T \cdot \mathrm{VCdim}(\mathcal{B}))$ (sparse-linear-combination bound; proof omitted).

Consistency. Training-error theorem with $T = O(\log n/\gamma^2)$ gives $L_S^{0/1}(H_T) = 0$

Realizable PAC. For $d = \mathrm{VCdim}(\mathrm{Lin}(\mathcal{B}, T))$, $n = \tilde{O}(d/\epsilon)$ samples suffice. Substituting:

$$n(\epsilon, \delta) = \tilde{O}(T \cdot \mathrm{VCdim}(\mathcal{B})/\epsilon) = \boldsymbol{\tilde{O}(\mathrm{VCdim}(\mathcal{B})/(\gamma^2 \epsilon))}.$$

Boosting reduces strong learning to weak learning plus sparse-ensemble complexity.

Weak-to-strong reduction (theorem)

Weak ⇒ strong, realizable PAC

Suppose $\mathcal{B}$ has finite VC dimension and A is a $\left(\frac{1}{2} - \gamma, \delta_0\right)$-weak learner for the realizable case using n_0 samples per call. Then for every $\epsilon, \delta > 0$, confidence-boosted AdaBoost run for $T = O(\log n/\gamma^2)$ rounds on a sample of size

$$n = \tilde{O}(\mathrm{VCdim}(\mathcal{B})/(\gamma^2\epsilon) + \log(1/\delta)/\epsilon)$$

returns H_T satisfying $\mathbb{P}_S[L_{\mathcal{D}}(H_T) \leq \epsilon] \geq 1 - \delta$.

- **capacity + realizable PAC** on $\mathrm{Lin}(\mathcal{B}, T)$: contributes $\mathrm{VCdim}(\mathcal{B})/(\gamma^2\epsilon)$ (capacity term) and $\log(1/\delta)/\epsilon$ (failure-budget term).

Efficient weak learning ⇔ Efficient strong learning, in the realizable PAC model.

What the guarantee means

Three quantities govern the bound:

Weak-learner complexity	**Edge**	**Rounds**
$\mathrm{VCdim}(\mathcal{B})$	γ	$T \approx \log n/\gamma^2$

What's surprising:

- the strong learner's sample complexity scales with $\mathrm{VCdim}(\mathcal{B})$ (the **base** class), not with the richer ensemble class $\mathrm{Lin}(\mathcal{B}, T)$ (whose VC dim can be up to $\tilde{O}(T \cdot \mathrm{VCdim}(\mathcal{B}))$);
- T enters n only as $\log n$, absorbed into $\tilde{O}$: no polynomial cost in T;
- the edge γ enters quadratically: half the edge needs $4\times$ the samples.

Why this works: $H_T \in \mathrm{Lin}(\mathcal{B}, T)$ is **T-sparse** only T base rules get nonzero weight, out of (possibly infinitely many).

Boosting inherits the statistical complexity of the base class, not of the bigger ensemble class.

Two more perspectives on AdaBoost

The theorem gives a clean realizable-PAC bound. What does it leave open?

- AdaBoost often keeps improving past $L_S^{0/1} = 0$ — **why?**
- the bound used the unproved VC bound on $\mathrm{Lin}(\mathcal{B}, T)$ — **can we avoid it?**
- in noisy settings, exponential loss over-focuses on mislabels — **what controls this?**

Two perspectives ahead:

- **optimization view**: AdaBoost as coordinate descent on $L^{\exp}$ — explains margin growth past zero training error;
- **compression view**: an algorithm-specific generalization tool that sidesteps the VC bound on $\mathrm{Lin}(\mathcal{B}, T)$.

Optimization clarifies what AdaBoost is doing; compression gives a parallel generalization bound.

AdaBoost as Optimization

Ensembles as linear predictors

One coordinate per hypothesis. Index coordinates of the feature space by elements of the base class $\mathcal{B}$. Each $h \in \mathcal{B}$ gives one feature

$$\varphi(x)_h = h(x) \in \{-1, +1\}, \quad \text{for each } \boldsymbol{h \in \mathcal{B}}.$$

An ensemble is a **linear predictor** in this feature space:

$$f_w(x) = \langle w, \varphi(x) \rangle = \sum_{h \in \mathcal{B}} w_h \cdot h(x),$$

where the weight w_h is the **vote given to hypothesis** h.

- $\mathcal{B}$ may be infinite (e.g. all decision stumps), so the feature space is infinite-dimensional;
- AdaBoost only ever activates T coordinates:

$$w = \sum_{t=1}^{T} \alpha_t e_{h_t}, \quad \boldsymbol{\|w\|_0 \le T}, \quad \|w\|_1 = \sum_{t=1}^{T} \alpha_t.$$

Boosting = sparse linear prediction over a dictionary indexed by weak rules.

Exponential loss objective

Empirical exponential loss (from the analysis section):

$$L_S^{\exp}(f) = \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i f(x_i)).$$

Linear-predictor view: $f(x) = \langle w, \varphi(x)\rangle = \sum_{h\in\mathcal{B}} w_h h(x)$. Substituting:

$$L_S^{\exp}(w) = \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i \langle w, \varphi(x_i)\rangle).$$

Coordinate h derivative (chain rule, $\frac{\partial}{\partial w_h}\langle w, \varphi(x_i)\rangle = h(x_i)$):

$$\boldsymbol{\frac{\partial L_S^{\exp}}{\partial w_h}(w) = -\frac{1}{n}\sum_{i=1}^{n} y_i h(x_i) \exp(-y_i \langle w, \varphi(x_i)\rangle).}$$

A convex objective on $w \in \mathbb{R}^{|\mathcal{B}|}$. Next: AdaBoost as coordinate descent on $L_S^{\exp}$.

AdaBoost as coordinate descent

Specialize to AdaBoost. At round t, $w_{t-1} := \sum_{s<t} \alpha_s e_{h_s}$, so $\langle w_{t-1}, \varphi(x_i)\rangle = f_{t-1}(x_i)$. Then (using D_i^t from the box):

$$\begin{aligned}\frac{\partial L_S^{\exp}}{\partial w_h}(w_{t-1}) &= -\frac{1}{n}\sum_{i=1}^{n} y_i h(x_i) \exp(-y_i \langle w_{t-1}, \varphi(x_i)\rangle) \\ &= -\frac{1}{n}\sum_{i=1}^{n} y_i h(x_i) \exp(-y_i f_{t-1}(x_i)) \\ &\propto -\mathbb{E}_{i\sim D^t}[y_i h(x_i)] = \mathbf{2L_{D^t}(h) - 1}.\end{aligned}$$

AdaBoost (round t)

$$D_i^t \propto \exp(-y_i f_{t-1}(x_i))$$
$$h_t = A(D^t)$$
$$\alpha_t = \tfrac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$$
$$f_t = f_{t-1} + \alpha_t h_t$$
$$H_t(x) = \mathsf{sign}(f_t(x))$$

Each round = one coordinate-descent step on $L_S^{\exp}(w)$:

- **Pick h_t:** most negative derivative $\Leftrightarrow$ smallest $L_{D^t}(h) \Leftrightarrow$ weak learner's output $h_t = A(D^t)$;
- **Pick α_t:** exact line search $\arg\min_\alpha L_S^{\exp}\left(w_{t-1} + \alpha e_{h_t}\right) = \frac{1}{2}\log((1-\varepsilon_t)/\varepsilon_t)$;
- **Update:** $w_t = w_{t-1} + \alpha_t e_{h_t}$, equivalently $f_t = f_{t-1} + \alpha_t h_t$.

AdaBoost = greedy coordinate descent on $L^{\exp}$

Past zero training error: margins keep growing

Around $T \approx \log(m)/(2\gamma^2)$:

- $L_S^{0/1} = 0$, so classification training error stops moving;
- coordinate descent on $L^{\exp}$ keeps going.

Coord descent keeps shrinking $L^{\exp}$. From the training-error analysis,

$$L_S^{\exp}(w_T) \leq \exp(-2\gamma^2 T).$$

Margins grow linearly in T. Since $\exp(-y_i f_T(x_i)) \leq n L_S^{\exp}(w_T)$, taking logs gives, for every example $i \in [n]$,

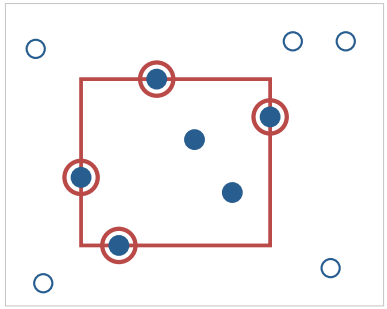
$$\boldsymbol{y_i f_T(x_i) \geq 2\gamma^2 T - \log n.}$$

Each example is classified more and more confidently as $T \to \infty$ (Schapire–Singer 1999).

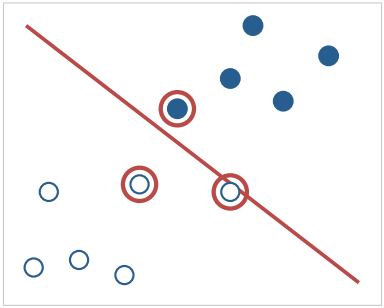
Past zero training error, classification loss flattens but margins keep growing. The right yardstick is scale-sensitive (next week).

Compression Schemes

Examples: hypotheses pinned by few sample points



Rectangles in $\mathbb{R}^2$. 4 extreme positive points (leftmost / rightmost / topmost / bottommost) pin the minimal consistent rectangle. The other $n - 4$ examples are ignored.



Halfspaces in $\mathbb{R}^d$. At most $d + 1$ "support" points pin a separating hyperplane. In $\mathbb{R}^2$: at most 3 (e.g. 1 positive, 2 negatives).

The learner ran on all n examples, yet the output is reconstructible from a small subset $\kappa(S) \subseteq S$.

Why another generalization tool?

Two ways to bound population error.

- **uniform (VC):** control every hypothesis in some class $\mathcal{H}$ simultaneously, via $\mathrm{VCdim}(\mathcal{H})$;
- **compression:** bound by **how few sample points reconstruct the output** $A(S)$.

Compression bound (informal). If $A(S)$ is reconstructible from a subset $\kappa(S) \subseteq S$ of size r, then

$$L_{\mathcal{D}}(A(S)) \leq L_S(A(S)) + \tilde{O}(r/n).$$

Applied to the previous slide:

- rectangles in $\mathbb{R}^2$: $r = 4 \to$ rate $\tilde{O}(4/n)$;
- halfspaces in $\mathbb{R}^d$: $r \leq d + 1 \to$ rate $\tilde{O}((d+1)/n)$.

The bound depends on r (samples needed to pin $A(S)$), not on $|\mathcal{H}|$ or $\mathrm{VCdim}(\mathcal{H})$.

Compression scheme: definition

A learning rule A is **r-compressing** if there exists a reconstruction map $\tilde{A}$ and a selection function $I(S) \subset \{1, \ldots, n\}$, $|I(S)| \leq r$, such that

$$A(S) = \tilde{A}\big(S_{I(S)}\big).$$

Here $\tilde{A}$ takes the r selected labeled examples and outputs a hypothesis on all of $\mathcal{X}$.

full sample

S, size n

$\rightarrow$

choose indices

$I(S)$, size r

$\rightarrow$

reconstruct

$A(S)$

The key is not $|\mathcal{H}|$; it is the number of sample points needed to describe the output.

Compression generalization bound

Compression bound (Littlestone-Warmuth)

If A is r-compressing and **consistent on S** ($L_S(A(S)) = 0$), then with probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$,

$$L_{\mathcal{D}}(A(S)) \leq \frac{r \log n + \log(1/\delta)}{n - r} = \tilde{O}\left(\frac{r}{n}\right).$$

Examples (from previous slide).

- rectangles in $\mathbb{R}^2$: $r = 4 \to$ rate $\tilde{O}(4/n)$;
- halfspaces in $\mathbb{R}^d$: $r \leq d + 1 \to$ rate $\tilde{O}((d+1)/n)$.

Notes.

- same form as the **realizable** VC bound, with $\mathrm{VCdim}(\mathcal{H})$ replaced by r;
- consistency is essential: the proof requires $A(S)$ to be perfect on S (see next slide).

Proof idea: union bound over descriptions

Setup. A is r-compressing, so $A(S) = \tilde{A}\left(S_{I(S)}\right)$ for some adaptively chosen $I(S)$ of size r, and $A(S)$ is consistent on S.

Fix any subset I with $|I| = r$. The reconstruction $h_I := \tilde{A}(S_I)$ depends only on S_I, hence is **independent of the remaining $n - r$ samples** $S \setminus S_I$.

Suppose $L_{\mathcal{D}}(h_I) > \epsilon$. The remaining samples are iid from $\mathcal{D}$ and independent of h_I, so

$$\mathbb{P}[h_I \text{ consistent on } S \setminus S_I] \leq (1-\epsilon)^{n-r} \leq \exp(-\epsilon(n-r)).$$

Union bound over the $\binom{n}{r} \leq n^r$ subsets:

$$\mathbb{P}[\exists I : L_{\mathcal{D}}(h_I) > \epsilon \text{ and } h_I \text{ consistent on } S \setminus S_I] \leq n^r \exp(-\epsilon(n-r)).$$

Apply to $A(S)$. Since $A(S) = h_{I(S)}$ is consistent on S, it is consistent on $S \setminus S_{I(S)}$. Setting the union bound $\leq \delta$:

$$L_{\mathcal{D}}(A(S)) \leq (r \log n + \log(1/\delta))/(n-r) \quad \text{w.p.} \geq 1 - \delta.$$

Specializing to AdaBoost

Outer sample size n.

- Draw the full training sample $S = \{(x_i, y_i)\}_{i=1}^{n}$ from $\mathcal{D}$.

High-confidence weak-call sample size n_1.

- Round t has a distribution D^t over the same n examples.
- To call A with high confidence, draw n_1 resampled examples from S according to D^t.
- n_1 includes repeated n_0-sample weak runs and validation resamples.

Compression size.

- Let r be the number of selected examples needed to describe $h_1, \ldots, h_T$.
- Since each high-confidence weak call uses at most n_1 examples, $r \leq Tn_1$.
- Training-error zero uses $T = O(\log n/\gamma^2)$ rounds.
- With per-round failure $\delta_1 = \delta/(2T)$, $n_1 = O((n_0 + 1/\gamma^2)\log(T/\delta))$.
- Hence $r \leq Tn_1 = O(T(n_0 + 1/\gamma^2)\log(T/\delta))$.

n is the full sample used for certification; $r \leq Tn_1$ describes the weak rules AdaBoost selected.

Specializing to AdaBoost: partial compression

Votes require partial compression.

- The selected examples reconstruct $h_1, \ldots, h_T$.
- They do not encode the real weights $\alpha_1, \ldots, \alpha_T$.
- Once the weak rules are fixed, the final vote ranges over $\mathrm{Lin}(\{h_1, \ldots, h_T\}, T)$.

Residual class.

- $\mathrm{Lin}(\{h_1, \ldots, h_T\}, T)$ is just linear thresholding over T fixed features.
- Its VC dimension is $O(T)$.

Partial-compression guarantee.

- Compression part: $r \leq Tn_1$ selected examples.
- Residual part: VC dimension $O(T)$ for the votes.
- If AdaBoost is consistent on the full sample S, then roughly

$$L_{\mathcal{D}}(H_T) \lesssim O((r + T + \log(1/\delta))/n).$$

Compressed resamples specify the weak rules; a low-dimensional residual class handles the votes.

Comparing the two generalization routes

Pure VC route.

- Bound the entire boosted class $\mathrm{Lin}(\mathcal{B}, T)$.
- If $\mathrm{VCdim}(\mathcal{B}) = d$, then $\mathrm{VCdim}(\mathrm{Lin}(\mathcal{B}, T)) \lesssim O(Td)$.
- Realizable VC bound gives roughly $L_{\mathcal{D}}(H_T) \lesssim O(Td/n)$.

Partial-compression route.

- Describe only the weak rules AdaBoost selected.
- Complexity is r for the weak-rule descriptions plus $O(T)$ for the votes.
- Bound gives roughly $L_{\mathcal{D}}(H_T) \lesssim O((r+T)/n)$, with $r \leq Tn_1$.

When is compression useful?

- Especially when $\mathcal{B}$ is huge, implicit, or improper, but selected weak rules have short descriptions.

VC controls the whole search space; compression controls the algorithm's selected description.

From Binary Classification to Generalized Learning

Beyond 0/1 loss: AdaBoost as a hint

- AdaBoost did not minimize 0/1 loss directly; it minimized the empirical **exponential** loss.
- The inequality $L_S^{0/1}(H_T) \le L_S^{\exp}(f_T)$ transferred the control.
- The training objective was already real-valued.

Real-valued losses common in ML:

hinge	**logistic**	**squared**	**exponential**
$[1 - yf(x)]_+$	$\log\big(1 + e^{-yf(x)}\big)$	$(y - f(x))^2$	$e^{-yf(x)}$

General problem. A loss $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$, sample $S = (z_1, \ldots, z_n) \sim \mathcal{D}^n$, compete with

$$\inf_{h \in \mathcal{H}} L(h), \quad L(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)], \quad L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, z_i).$$

Binary classification is the special case $z = (x, y)$, $\ell(h, z) = \mathbf{1}[h(x) \neq y]$.

AdaBoost was already a real-valued loss analysis; next we generalize.

Where VC stops, scale begins

Why VC alone is not enough. For real-valued $f : \mathcal{X} \to \mathbb{R}$, a finite sample produces a continuum of output values; counting label patterns no longer measures complexity.

Hint from today.

- training-error decay $\exp(-2\gamma^2 T)$ depends on the **edge** γ, a scale, not a combinatorial dimension;
- past zero training error, the margin $y_i f_T(x_i) \geq 2\gamma^2 T - \log n$ keeps growing;
- generalization in this regime is controlled by **margin** and **norm**, not parameter count.

Next: the scale-sensitive toolbox.

Covering numbers

count behaviors up to resolution α

Rademacher complexity

fit random sign patterns

Norm and margin

$\|w\|$, $y\langle w, \varphi(x)\rangle$, Lipschitz losses

The new question: not how many parameters? but how large is the predictor relative to the margin?

Summary

Week 6 summary

Week 5	→	Week 6	→	Next
hardness + neural nets		boosting + compression		real-valued losses + scale

- **Weak learning:** fixed edge γ over random guessing.
- **Boosting:** reweight examples to convert weak learners into a strong ensemble.
- **AdaBoost:** training error decays as $\exp(-2\gamma^2 T)$; coordinate descent on $L_S^{\exp}$.
- **Generalization:** sparse ensembles ($\tilde{O}(T \cdot \mathrm{VCdim}(\mathcal{B}))$) or compression descriptions ($\tilde{O}(r/n)$).
- **Implicit bias:** margins keep growing past zero training error.

AdaBoost is simultaneously a statistical reduction, an algorithm, and an implicit regularizer.