AdaBoost

Luu Minh Sao Khue

December 9, 2025

1. Introduction

AdaBoost (Adaptive Boosting) is a classical boosting algorithm for binary classification. It builds an ensemble of *weak learners*, typically very shallow decision trees (decision stumps), and combines them into a strong classifier.

Key characteristics:

- Models are trained **sequentially**, not independently.
- Each new model focuses more on examples that previous models misclassified.
- The final prediction is a **weighted vote** of all weak learners.

AdaBoost is one of the earliest and most influential boosting methods and is a useful contrast to bagging-based methods such as Random Forest.

2. Intuition

Suppose we have training data

$$\{(x_i, y_i)\}_{i=1}^{N}, \qquad y_i \in \{-1, +1\}.$$

The core idea of AdaBoost:

> *Train many simple classifiers one after another, each one paying more attention to the examples that previous classifiers got wrong, and then combine them with a weighted vote.*

2.1 From Weak Learners to a Strong Classifier

A *weak learner* is a classifier that performs only slightly better than random guessing (e.g. accuracy just above 50% for binary classification). A single decision stump (a depth-1 tree) is often weak but very fast.

AdaBoost turns many such weak learners into a strong classifier by:

- Training a weak learner on the data with a certain weight distribution over samples.
- Evaluating how well it performs using a **weighted** error.

- Giving more influence (larger weight) to learners with lower error.
- Increasing the weights of misclassified samples so that the next learner focuses on them.

Over many rounds, the ensemble learns to correct its own mistakes.

2.2 How AdaBoost Differs from Random Forest

Both AdaBoost and Random Forest are ensembles of decision trees, but they behave quite differently:

- **Training style:**
  - *Random Forest*: trees are trained **in parallel**, each on a different bootstrap sample; they do not interact.
  - *AdaBoost*: weak learners are trained **sequentially**; each learner depends on the mistakes of the previous ones.
- **Data sampling vs. reweighting:**
  - *Random Forest*: uses bootstrap sampling and random feature subsets; all samples in a tree have equal weight.
  - *AdaBoost*: keeps the whole dataset but **changes the weights** of samples: misclassified points become more important.
- **Goal:**
  - *Random Forest*: mainly reduces **variance** (stabilizes a high-variance model).
  - *AdaBoost*: can reduce both **bias and variance** by focusing on hard examples, but may become sensitive to noise and outliers.
- **Model complexity:**
  - *Random Forest*: often uses deeper trees (strong learners).
  - *AdaBoost*: usually uses very shallow trees (weak learners).

A short summary:

> Random Forest = many strong trees, trained independently, averaged. AdaBoost = many weak trees, trained sequentially, adaptively reweighted.

3. Weighted Error and Learner Weight

Let $w_i^{(m)}$ be the weight of sample i at iteration m. Initially, all samples have equal weight:

$$w_i^{(1)} = \frac{1}{N}, \quad i = 1, \ldots, N.$$

At iteration m, we train a weak learner $h_m(x)$ using these weights. The **weighted classification error** of h_m is

$$\varepsilon_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(x_i) \neq y_i\}}{\sum_{i=1}^{N} w_i^{(m)}}.$$

We then compute the importance (weight) of this learner:

$$\alpha_m = \frac{1}{2} \log\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right).$$

- If $\varepsilon_m < 0.5$, then $\alpha_m > 0$: the learner is better than random and contributes positively to the ensemble.
- If ε_m is close to 0, α_m is large: a very strong weak learner.
- If $\varepsilon_m = 0.5$, then $\alpha_m = 0$: the learner is equivalent to random guessing.

4. Updating Sample Weights

After we compute α_m, we update the sample weights as:

$$w_i^{(m+1)} = w_i^{(m)} \exp\big(-\alpha_m y_i h_m(x_i)\big).$$

Since $y_i \in \{-1, +1\}$ and $h_m(x_i) \in \{-1, +1\}$:

- If $h_m(x_i) = y_i$ (correct), then $y_i h_m(x_i) = +1$ and
$$w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m},$$
so the weight decreases.
- If $h_m(x_i) \neq y_i$ (wrong), then $y_i h_m(x_i) = -1$ and
$$w_i^{(m+1)} = w_i^{(m)} e^{+\alpha_m},$$
so the weight increases.

Finally, we normalize weights so that

$$\sum_{i=1}^{N} w_i^{(m+1)} = 1.$$

This mechanism ensures that misclassified examples become more influential in the next iteration.

5. Final Prediction

After M rounds, AdaBoost combines all weak learners using a weighted majority vote. We define the ensemble score

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x),$$

and the final prediction is

$$\hat{y}(x) = \operatorname{sign}\big(F(x)\big).$$

Learners with smaller error get larger α_m and have more impact on the final decision.

6. AdaBoost Algorithm

Algorithm 1 AdaBoost (Binary Classification)

Input: training data $\{(x_i, y_i)\}_{i=1}^N$ with $y_i \in \{-1, +1\}$, number of rounds M.

Initialize sample weights:

$$w_i^{(1)} = \frac{1}{N}, \quad i = 1, \ldots, N.$$

for $m = 1$ to M **do**

Train weak learner h_m using weights $w_i^{(m)}$.

Compute weighted error:

$$\varepsilon_m = \frac{\sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(x_i) \neq y_i\}}{\sum_{i=1}^N w_i^{(m)}}.$$

Compute learner weight:

$$\alpha_m = \frac{1}{2} \log\left(\frac{1 - \varepsilon_m}{\varepsilon_m}\right).$$

Update sample weights:

$$w_i^{(m+1)} = w_i^{(m)} \exp\big(-\alpha_m y_i h_m(x_i)\big), \quad i = 1, \ldots, N.$$

Normalize weights so that $\sum_{i=1}^N w_i^{(m+1)} = 1$.

end for

Output: final classifier

$$\hat{y}(x) = \operatorname{sign}\left(\sum_{m=1}^M \alpha_m h_m(x)\right).$$

7. Summary

AdaBoost is a boosting algorithm that:

- trains weak learners **sequentially** on reweighted data,
- emphasizes misclassified examples via sample weight updates,
- combines learners using a weighted majority vote.

Compared to Random Forest, which uses bagging and random feature selection to reduce variance with strong trees trained independently, AdaBoost uses many weak trees trained adaptively to focus on hard examples. This can yield a powerful classifier, but also makes AdaBoost more sensitive to outliers and noisy labels.