AdaBoost: From Weak Learners to Strong Classifiers

Minimizing Exponential Error via Sequential Learning

Nikhil, Jannen

January 9, 2026

The Power of the Committee

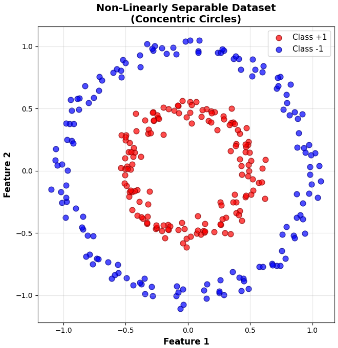

Figure: Linear Separability Challenge

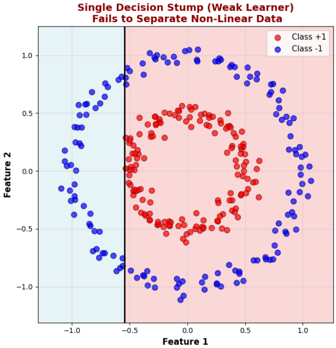

Figure: Decision Stump fails

- **The Problem:** Complex data is rarely linearly separable.
- **The Naive Solution:** Build a massive, complex model.
- **The AdaBoost Solution:** Combine simple "Decision boundaries."

$$H(x) = \text{sign}\left(\sum \alpha_k h_k(x)\right)$$

Why Not Squared Error?

- **Regression:** Minimizes Squared Error (MSE).
- **Classification:** MSE is unreliable.
  - Penalizes "too correct" predictions.
  - Sensitive to outliers in the wrong way.
- **AdaBoost:** Minimizes Exponential Error.

$$E = \sum e^{-t_n f(x_n)}$$

Focuses heavily on misclassified points ($t \neq f(x)$).

Sequential Minimization

$$E = \sum_{n=1}^{N} \exp\left(-t_n\left[H_{k-1}(x_n) + \alpha_k h_k(x_n)\right]\right)$$

- Cannot optimize all α and h at once.
- **Greedy Approach:** Freeze past, optimize current step.
- **The Weight Trick:**

$$w_n^{(k)} = \exp(-t_n H_{k-1}(x_n))$$

- Previous errors become current weights!
- Misclassified points get higher weights.

Finding the Optimal Weight (α_k)

- **Goal:** Find α_k to minimize E.
- Taking the derivative: $\frac{\partial E}{\partial \alpha_k} = 0$
- **Result:**

$$\alpha_k = \frac{1}{2} \ln \left(\frac{1 - \epsilon_k}{\epsilon_k} \right)$$

- Low Error ($\epsilon_k \to 0$) $\Rightarrow$ High Alpha
- Random Guess ($\epsilon_k = 0.5$) $\Rightarrow$ Zero Alpha
- Worse than random ($\epsilon_k > 0.5$) $\Rightarrow$ Negative Alpha

Updating Sample Weights

- After computing α_k and training h_k, update weights:
$$w_n^{(k+1)} = w_n^{(k)} \cdot \exp\left(-\alpha_k \cdot t_n \cdot h_k(x_n)\right)$$

- **If correct:** $t_n \cdot h_k(x_n) = 1 \Rightarrow$ weight decreases
- **If wrong:** $t_n \cdot h_k(x_n) = -1 \Rightarrow$ weight increases
- **Normalize:** $w_n^{(k+1)} \leftarrow \frac{w_n^{(k+1)}}{\sum_i w_i^{(k+1)}}$

Hard examples get harder weights: forces next learner to focus.

The AdaBoost Algorithm

1. **Initialize:** $w_n^{(1)} = \frac{1}{N}$
2. **For** $k = 1, 2, \dots, K$:
    1. Train weak learner h_k on weighted data
    2. Calculate error: $\epsilon_k = \sum_n w_n^{(k)} \cdot I(h_k(x_n) \neq t_n)$
    3. Compute weight: $\alpha_k = \frac{1}{2} \ln \left(\frac{1-\epsilon_k}{\epsilon_k} \right)$
    4. Update: $w_n^{(k+1)} = w_n^{(k)} \cdot \exp(-\alpha_k \cdot t_n \cdot h_k(x_n))$
    5. Normalize: $w_n^{(k+1)} \leftarrow \frac{w_n^{(k+1)}}{\sum_i w_i^{(k+1)}}$
3. **Output:** $H(x) = \text{sign} \left(\sum_{k=1}^{K} \alpha_k h_k(x) \right)$

The Dataset: Breast Cancer Wisconsin (Diagnostic)

What are we classifying?

- **Source:** FNA biopsy images
- **Samples:** 569 patients
- **Task:** Binary classification

- Malignant (M): $\sim 37\%$
- Benign (B): $\sim 63\%$

Why it matters:

- Early cancer detection
- Non-invasive procedure
- Real-world ML

Feature Engineering:

- 10 cell nucleus characteristics
- 3 measurements each
- $\Rightarrow$ 30 total features

1. Radius
2. Texture
3. Perimeter
4. Area
5. Smoothness
6. Compactness
7. Concavity
8. Concave Points
9. Symmetry
10. Fractal Dimension

Visualization Strategy: 30D to 2D Projection

Figure: Breast Cancer Wisconsin (Diagnostic) dataset: 569 FNA-based samples with 30 cell-nucleus features used to classify malignant vs. benign tumors and demonstrate AdaBoost on high-dimensional medical data.

- **Challenge:** 30 features impossible to visualize
- **Solution:** Principal Component Analysis (PCA)
  - Projects 30D to 2D while preserving variance
  - Classes remain **non-linearly separable**
  - Perfect for demo
- **Key Insight:** Single stump (line) still fails in 2D

Live Demo: AdaBoost on Breast Cancer (2D PCA)

- Switch to Python console
- Run: `python visualize_adaboost_pca.py`
- Evolution of decision boundary:
  - Panel 1: Single stump (fails)
  - Panels 2-5: Ensemble grows
  - Right: Error curves decreasing
- Compare to data (red = malignant, blue = benign)

Key: Exponential error decreases each iteration!

Why AdaBoost Works: The Exponential Error

Figure: Decision boundary evolution: 1, 2, 3, 5 weak learners

- Single stump: Misclassifies $\sim 50\%$
- 2 stumps: Better separation
- 5 stumps: $> 95\%$ accuracy

Summary

- **AdaBoost minimizes Exponential Error**
- Turns complex problem into simple sequence (greedy)
- **Weight update:** Focuses on misclassified examples
- **Key Takeaway:** Smart combination of simple models
- **Rigorous:** Every formula from exponential loss

AdaBoost: Mathematically principled and empirically powerful

References & Tools

- **Literature:**

  - Bishop, C. M. (2006). *Pattern Recognition* (Ch. 14).
  - Friedman, Hastie, Tibshirani (2000). *Additive Logistic Regression*.
  - Viola, Jones (2004). *Robust Real-Time Face Detection*.

- **Dataset:**

  - Wolberg et al. (1995). *Breast Cancer Wisconsin*. UCI ML. DOI: 10.24432/C5DW2B

- **Tools:**
  - scikit-learn, ucimlrepo, matplotlib