Adaptive Boosting (AdaBoost)

DATA2060 Final Project

Pranav Gundrala and Kate Gilbert

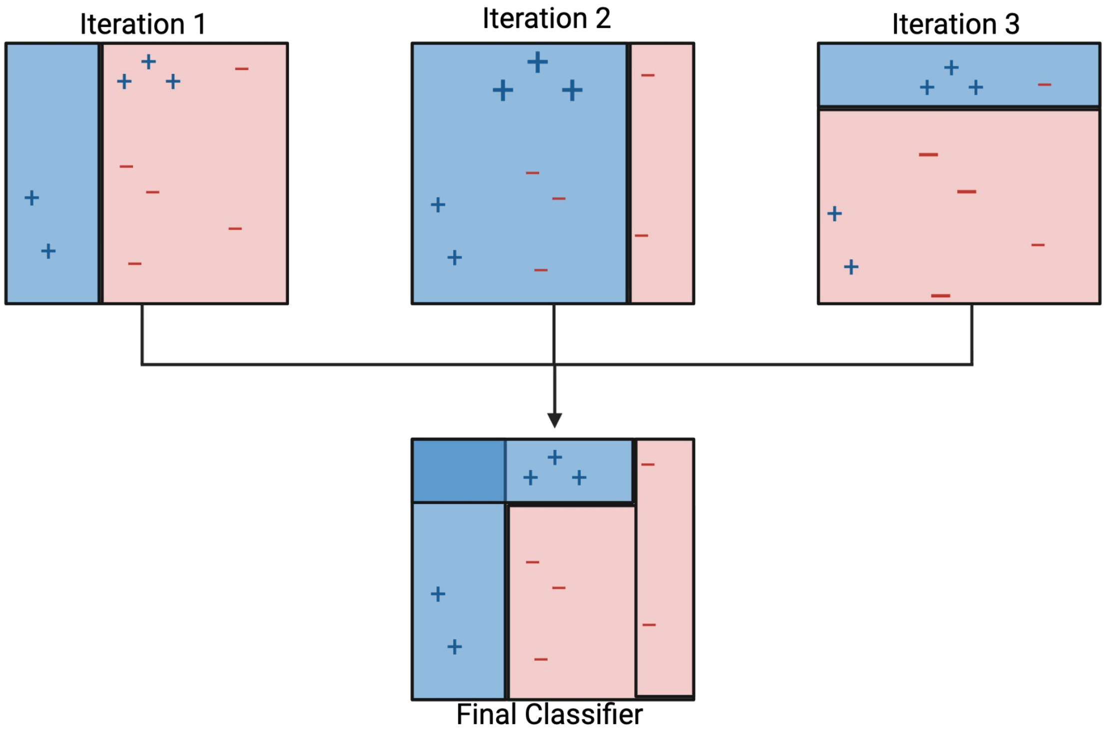
Math Background: Ensemble Technique

- Weighted combination of T weak learners, $h_t(x)$, with learner weights w_t: $F(x) = \sum_{t=1}^{T} w_t h_t(x)$
- Final prediction: $h_s(x) = \mathrm{sign}(F(x))$
- Weighted error of weak learner t, with example weights $D_i^{(t)}$: $\epsilon_t := L_{D^{(t)}}(h_t) := \sum_{i=1}^{m} D_i^{(t)} \mathbf{1}_{[h_s(x_i) \neq y_i]}$
- Update example weights: $D_i^{(t+1)} = \frac{D_i^{(t)} exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} exp(-w_t y_j h_t(x_j))} \; for\ all\ i = 1, \ldots, m$

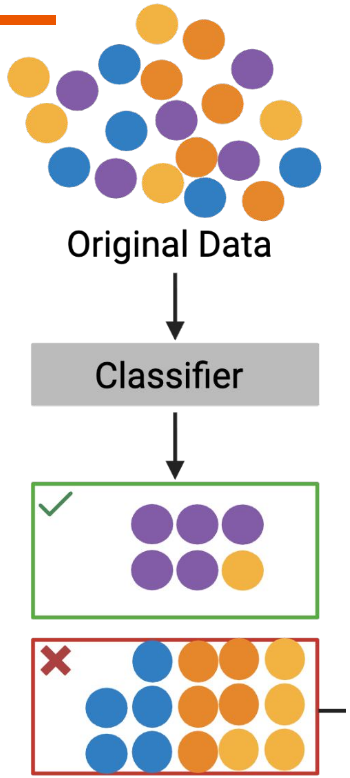
Math Background: Ensemble Technique

$$E(H,T) = \{x \mapsto \operatorname{sign}(\sum_{t=1}^{T} w_t h_t(x)) : w \in \mathbb{R}^T, \forall t\ h_t \in H\}$$

Weighted combination of weak learners:



Misclassified points are prioritized in subsequent iterations:



Numerical Technique: Greedy Optimization

Input:

training set S $(x_i, y_i), \ldots, (x_m, y_m)$

weak learner WL

number of rounds T

Initialize:

example weights: $D^{(1)} = (\frac{1}{m}, \ldots \frac{1}{m})$

Initial weights for all samples are the same

For $t = 1$ to T:

Train weak learner $h_t = WL(D^{(t)}, S)$

In the context of our project, a decision stump

Compute weighted error:

$\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbf{1}_{[y_i \neq h_t(x_i)]}$

Compute learner weight:

$w_t = \frac{1}{2} \log \left(\frac{1}{\epsilon_t} - 1 \right)$

Weight for the learner based on it's error

Update example weights:

$D_i^{(t+1)} = \frac{D_i^{(t)} exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} exp(-w_t y_j h_t(x_j))}$ for all $i = 1, \ldots, m$

Update sample weights using the ensemble's mistakes

Normalize weights so that:

$\sum_{i=1}^{m} D_i^{(t+1)} = 1$

Output the hypothesis $h_s(x) = \text{sign}(\sum_{t=1}^{T} w_t h_t(x))$

Sklearn results: implementation

An `AdaBoost` class for binary classification, using sklearn defaults

Sklearn results: implementation

An `AdaBoost` class for binary classification, using sklearn defaults

- **`Stump`**`(self, distribution, gain_function=gini):`
    - a decision stump class inspired by homework 5 (decision trees)
    - implements **sample weights** (distribution) and **continuous feature** support

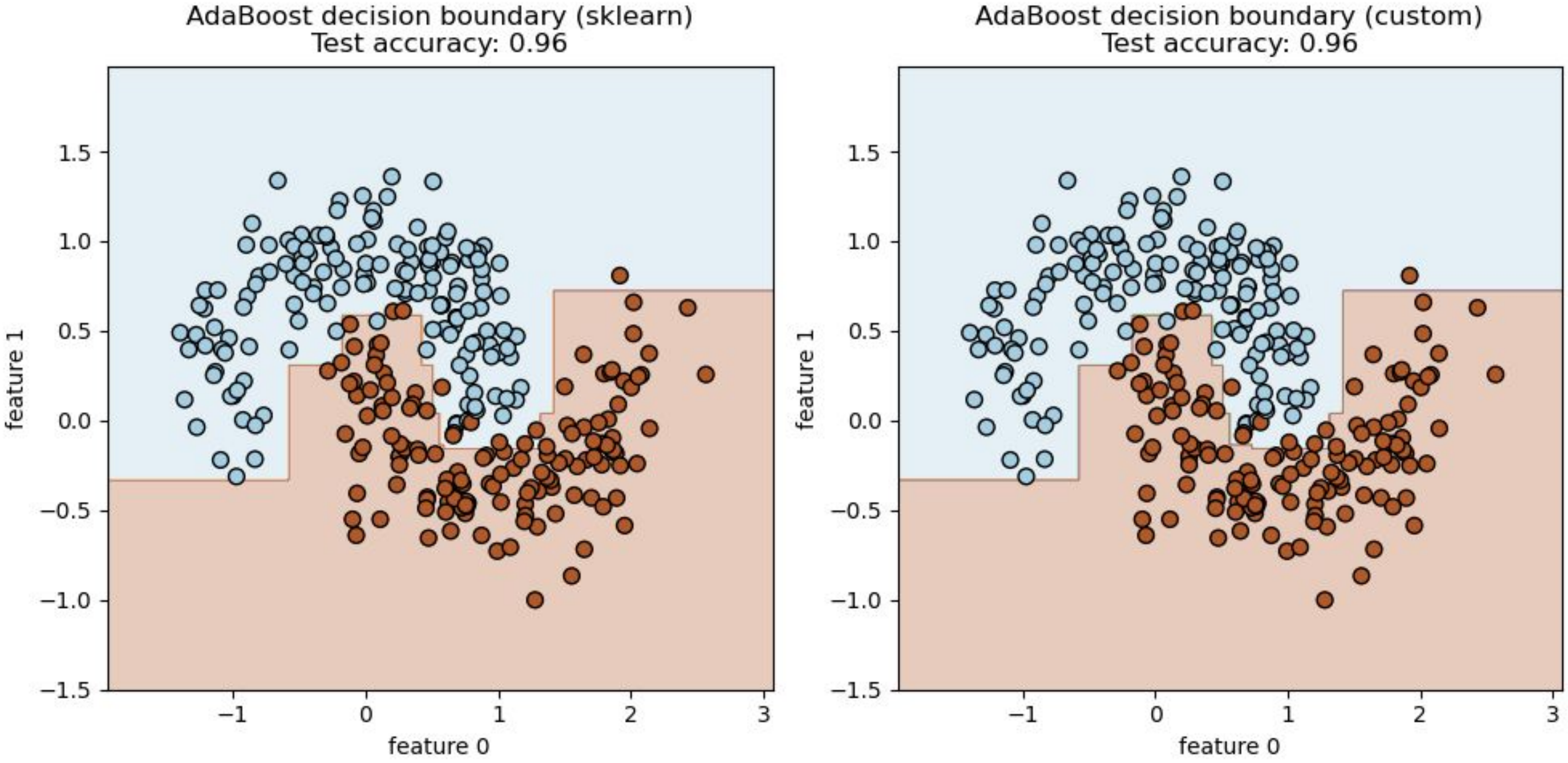
Sklearn results: implementation

An `AdaBoost` class for binary classification, using sklearn defaults

- **`Stump`**`(self, distribution, gain_function=gini):`
  - a decision stump class inspired by homework 5 (decision trees)
  - implements **sample weights** (distribution) and **continuous feature** support

- **`AdaBoost`**`(self, n_estimators, learner_class=`**`Stump`**`):`
  - ensemble class based on the pseudocode shown before
  - **n_estimators** is the number of learners (**T**) and **learner** is set to **Stump**
  - implements **greedy optimization** in training
  - returns a prediction as a weighted sum of weak learners

Sklearn results: synthetic data (`make_moons`)



Sklearn results: WI Breast Cancer dataset

Dataset (from Kaggle)

- **10** continuous features from cell images
- → radius, perimeter, area, texture, etc.
- 455 training points, 114 testing points
- **binary** classification – ‘malignant’ vs ‘benign’

Sklearn results: WI Breast Cancer dataset

Dataset (from Kaggle)

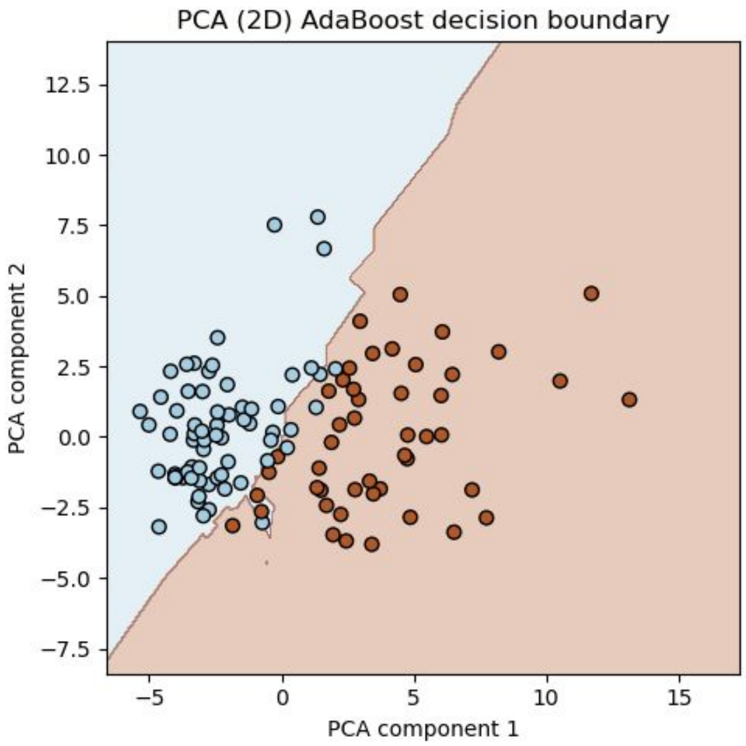
- **10** continuous features from cell images
- → radius, perimeter, area, texture, etc.
- 455 training points, 114 testing points
- **binary** classification – 'malignant' vs 'benign'

`AdaBoostClassifier() (sklearn):`

- **`.score():`** `0.973684`
- **`time:`** `0.0629457 seconds`

`AdaBoost() (custom):`

- **`accuracy_score():`** `0.973684`
- **`time:`** `11.5083223 seconds`



Summary Slide

AdaBoost generalizes well to different kinds of data and complex/non-linear relationships.

The method for including the weight distribution $D^{(t)}$ is not immediately obvious for a particular weak learner, so it was a challenge to figure out how to adjust the optimizer.

> In the case of a **Decision Stump**, this means sample weights are used in calculating impurity/gain for the ID3 algorithm (without recursion).
>
> **AdaBoost** predictions are -1/1 not 0/1, so this had to be adjusted for in the stump as well

The decision stump implemented here works differently than the tree we implemented in class since it works on continuous data → we had to loop over all possible mid points

The sklearn implementation was much faster, most likely because sklearn's DecisionTreeClassifier is highly optimized

References

Freund, Y. & Schapire, R.E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), pp.119–139. https://doi.org/10.1006/jcss.1997.1504

GeeksforGeeks, 2025. AdaBoost in Machine Learning. [online] Available at: https://www.sciencedirect.com/science/article/pii/S002200009791504X [Accessed 5 December 2025].

Schapire, R.E., 2013. Explaining AdaBoost. In: B. Schölkopf, Z. Luo and V. Vovk, eds. Empirical Inference: Festschrift in Honour of Vladimir N. Vapnik. Berlin: Springer, pp.37–52. DOI: 10.1007/978-3-642-41136-6_5.

de Giorgio, A., 2023. Systematic review of class imbalance problems in machine learning and deep learning solutions in manufacturing. Expert Systems with Applications, 229, p.120193. Available at: https://www.sciencedirect.com/science/article/pii/S0278612523002157 [Accessed 5 December 2025].