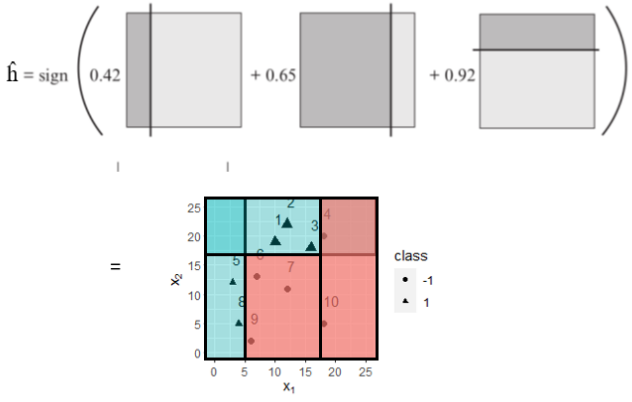
Introduction to Machine Learning

Introduction to Boosting / AdaBoost



Learning goals

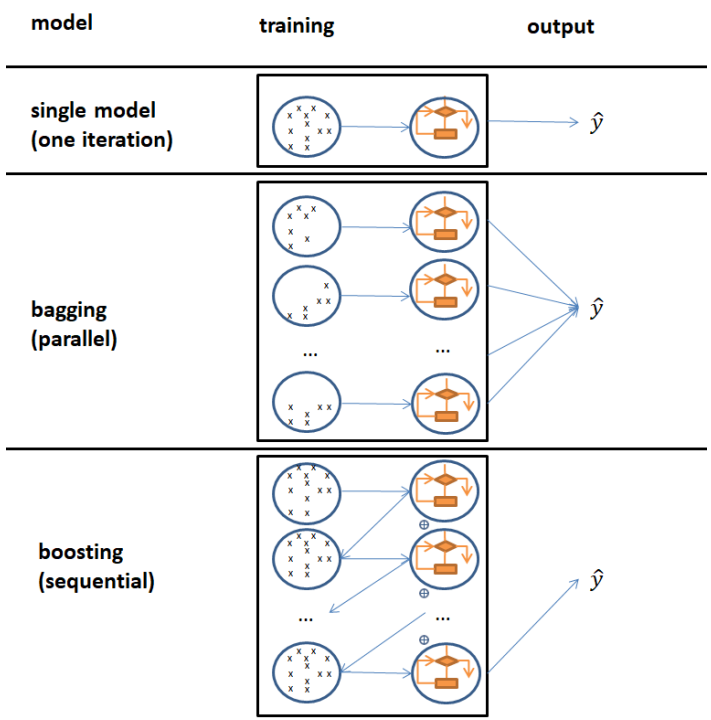
- Briefly cover the older AdaBoost and its general idea
- Understand difference between bagging and boosting

INTRODUCTION TO BOOSTING

- Boosting is one of the most powerful learning ideas since 1990.
- Originally designed for classification, (especially gradient) boosting handles regression (and many other supervised tasks) naturally nowadays.
- Homogeneous ensemble method (like bagging), but fundamentally different approach.
- **Idea:** Take a weak classifier and sequentially apply it to modified versions of the training data.
- We will begin by describing an older, simpler boosting algorithm designed for binary classification, the popular “AdaBoost”.

BOOSTING VS. BAGGING

- Homogeneous ensemble method (like bagging).
- **Idea:** Take a weak classifier and sequentially apply it to modified versions of the training data.
- Sequential not parallel model building, to minimize loss instead of variance reduction.



BOOSTING AS A THEORETICAL PROBLEM

Boosting was developed as the answer to a theoretical problem:

"Does the existence of a weak learner for a certain problem imply the existence of a strong learner?" (Kearns, 1988)

- **Weak learners** are defined as a prediction rule with a correct classification rate that is at least slightly better than random guessing (> 50% accuracy on a balanced binary problem).
- We call a learner a **strong learner** "if there exists a polynomial-time algorithm that achieves low error with high confidence for all concepts in the class" (Schapire, 1990).
- Any weak (base) learner can be iteratively boosted to become a strong learner (Schapire and Freund, 1990).
- Idea was refined into **AdaBoost** (Adaptive Boosting).

ADABOOST

- Assume binary classification with y encoded as $\{-1, +1\}$.
- We use binary classifiers as weak base learners (e.g., tree stumps) from a hypothesis space $\mathcal{B}$, denoted $b^{[m]}$ (or $b^{[m]}(\mathbf{x})$ or $b(\mathbf{x}, \boldsymbol{\theta}^{[m]})$), which are hard labelers and output from $\{-1, +1\}$.
- Decision score of the ensemble of size M is weighted average:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta^{[m]} b(\mathbf{x}, \boldsymbol{\theta}^{[m]}) \in \mathbb{R}$$

- The base learner is sequentially applied to weighted training observations. After each base learner fit, currently misclassified observations receive a higher weight for the next iteration, so we focus more on instances that are harder to classify.
- BLs with higher predictive accuracy receive higher weights $\beta^{[m]}$.

ADABOOST

Algorithm AdaBoost

1: Initialize observation weights: $w^{[1](i)} = \frac{1}{n} \quad \forall i \in \{1, \ldots, n\}$

2: **for** $m = 1 \rightarrow M$ **do**

3: Fit classifier to training data with weights $w^{[m]}$ and get $\hat{b}^{[m]}$

4: Calculate weighted in-sample misclassification rate

$$\text{err}^{[m]} = \sum_{i=1}^{n} w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}}$$

5: Compute: $\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1-\text{err}^{[m]}}{\text{err}^{[m]}}\right)$

6: Set: $w^{[m+1](i)} = w^{[m](i)} \cdot \exp\left(-\hat{\beta}^{[m]} \cdot y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})\right)$

7: Normalize $w^{[m+1](i)}$ such that $\sum_{i=1}^{n} w^{[m+1](i)} = 1$

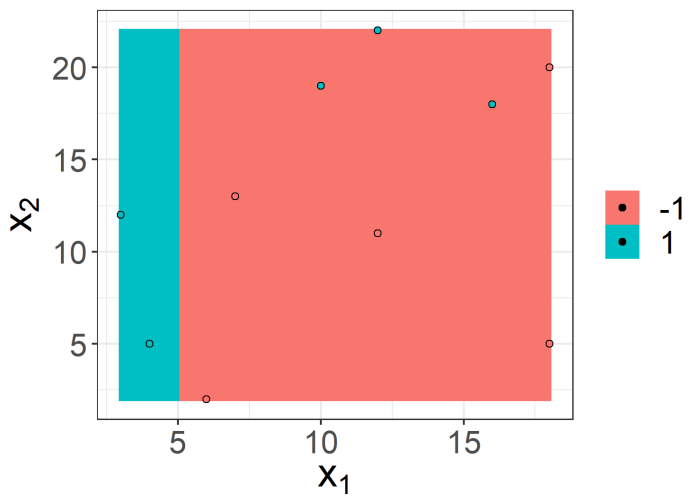
8: **end for**

9: Output: $\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \hat{\beta}^{[m]} \hat{b}^{[m]}(\mathbf{x})$

ADABOOST ILLUSTRATION

Example

- $n = 10$ observations and two features x_1 and x_2
- Tree stumps as base learners $b^{[m]}(\mathbf{x})$
- Balanced classification task with y encoded as $\{-1, +1\}$
- $M = 3$ iterations $\Rightarrow$ initial weights $w^{[1](i)} = \frac{1}{10} \quad \forall i \in 1, \ldots, 10.$



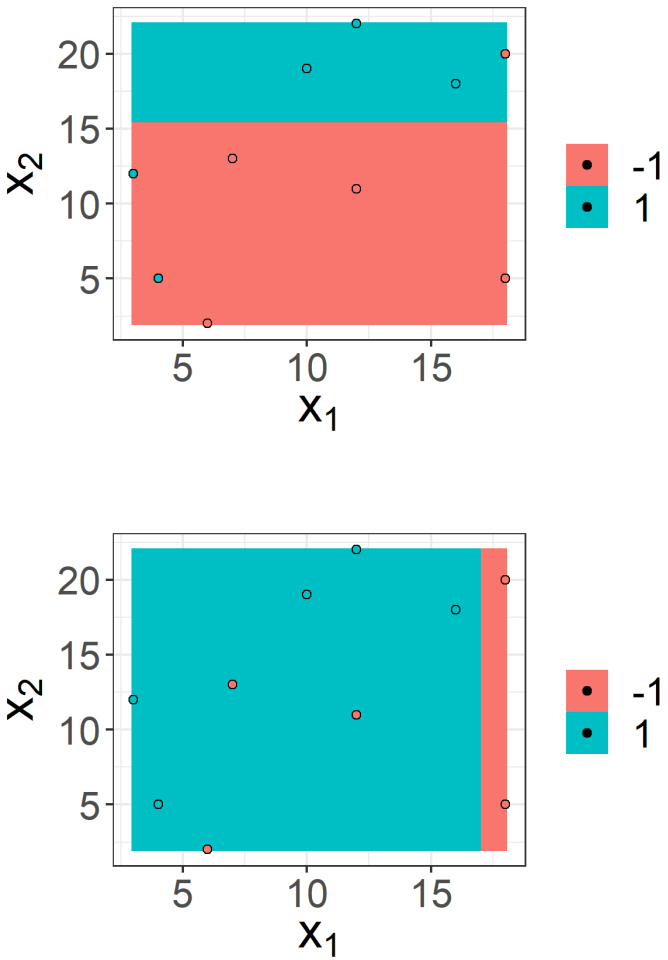
Iteration $m = 1$:

- $\text{err}^{[1]} = 0.3$
- $\hat{\beta}^{[1]} = \frac{1}{2} \log\left(\frac{1-0.3}{0.3}\right) \approx 0.42$

New observation weights:

- Prediction correct:
  $w^{[2](i)} = w^{[1](i)} \cdot \exp\left(-\hat{\beta}^{[1]} \cdot 1\right)$
  $\approx 0.065.$
- For 3 misclassified observations:
  $w^{[2](i)} = w^{[1](i)} \cdot \exp\left(-\hat{\beta}^{[1]} \cdot (-1)\right)$
  $\approx 0.15.$
- After normalization:
  - correctly classified: $w^{[2](i)} \approx 0.07$
  - misclassified: $w^{[2](i)} \approx 0.17$

ADABOOST ILLUSTRATION



Iteration $m = 2$:

- $\text{err}^{[2]} = 3 \cdot 0.07 = 0.21$
- $\hat{\beta}^{[2]} \approx 0.65$

New observation weights:

- E.g., for 3 misclassified observations: $w^{[3](i)} = w^{[2](i)} \cdot \exp\left(-\hat{\beta}^{[1]} \cdot (-1)\right) \approx 0.14.$
- After normalization: $w^{[3](i)} \approx 0.17$ (misclassified)

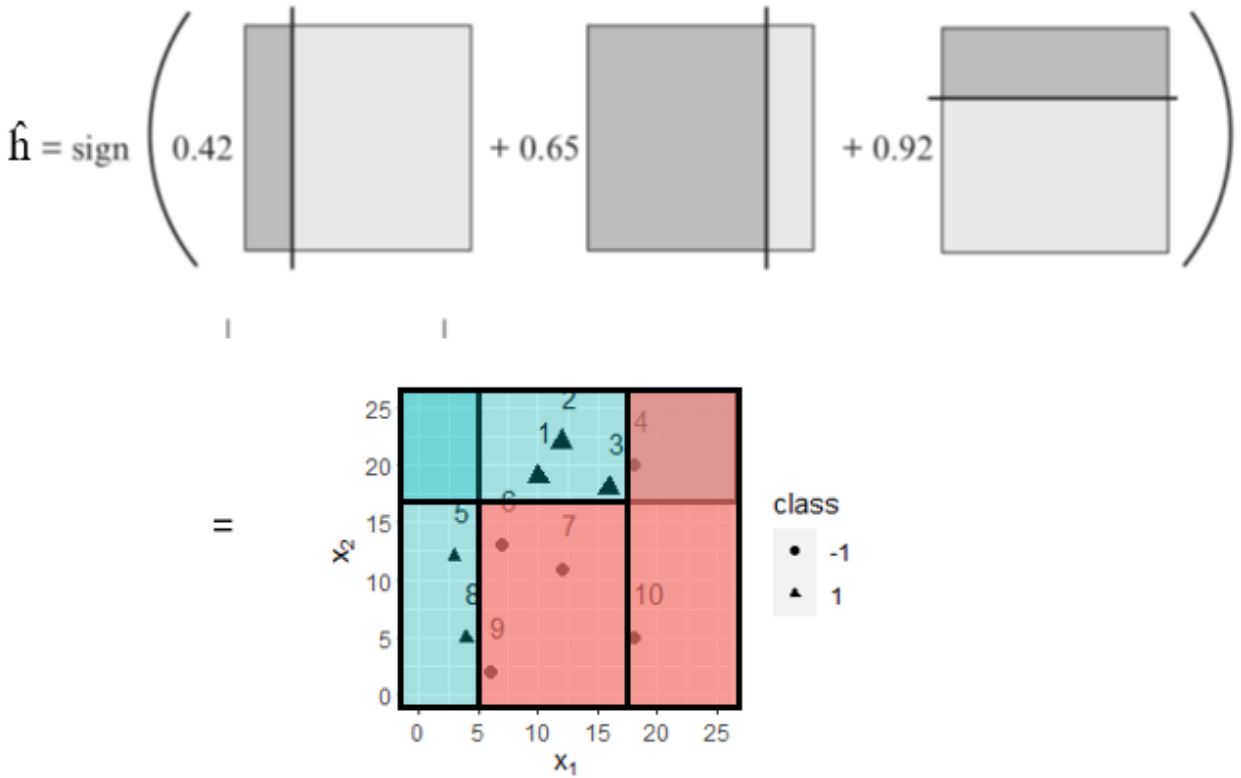
Iteration $m = 3$:

- $\text{err}^{[3]} = 3 \cdot 0.045 \approx 0.14$
- $\hat{\beta}^{[3]} \approx 0.92$

Note: the smaller the error rate of a base learner, the larger the weight, e.g., $\text{err}^{[3]} \approx 0.14 < \text{err}^{[1]} \approx 0.3$ and $\hat{\beta}^{[3]} \approx 0.92 > \hat{\beta}^{[1]} \approx 0.42$.

ADABOOST ILLUSTRATION

With $\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \hat{\beta}^{[m]} \hat{b}^{[m]}(\mathbf{x})$ and $h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \in \{-1, +1\}$, we get:



Hence, when all three base classifiers are combined, all samples are classified correctly.

BAGGING VS BOOSTING

Random forest

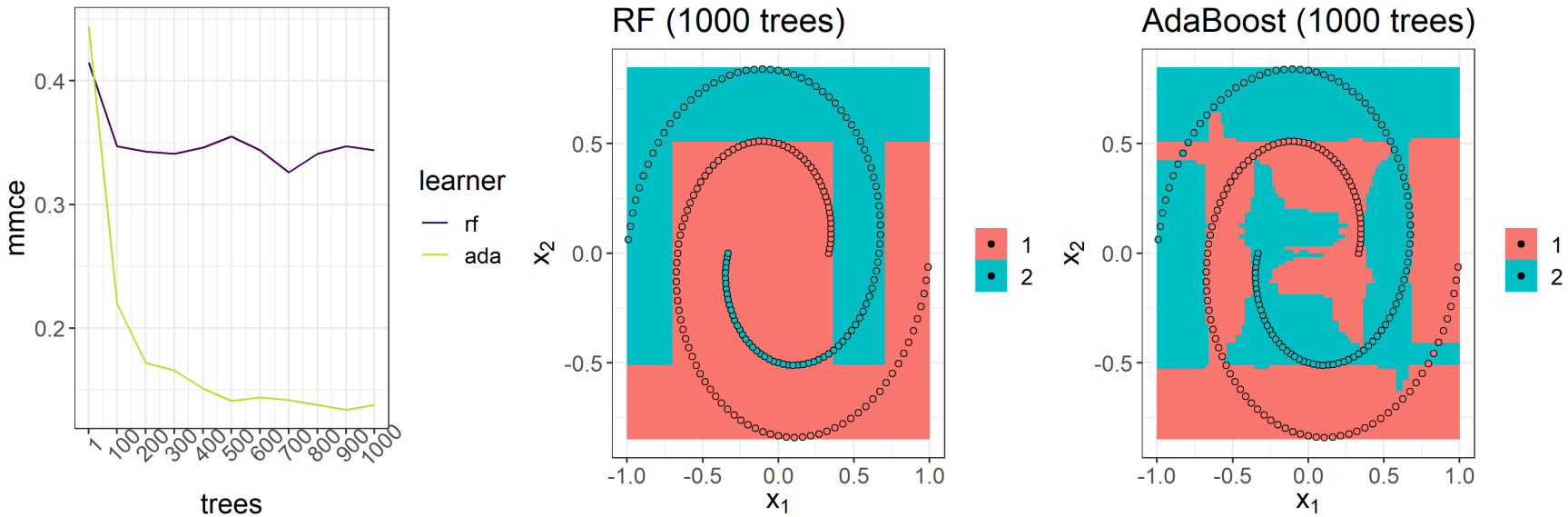
- Base learners are typically deeper decision trees (not only stumps!)
- Equal weights for BL
- BLs fitted independently
- Aim: variance reduction
- Tends **not** to overfit if ensemble size grows

AdaBoost

- BLs are weak learners, e.g., only stumps
- BL are weighted
- Sequential fitting of BLs
- Aim: loss reduction
- Tends to overfit with more iterations

BAGGING VS BOOSTING STUMPS

Random forest versus AdaBoost (both with stumps) on $\mathrm{spirals}$ data from $\mathrm{mlbench}$ ($n = 200$, $sd = 0$), with 5×5 repeated CV.



Weak learners do not work well with bagging as only variance, but no bias reduction happens.

OVERFITTING BEHAVIOR

Historically, the overfitting behavior of AdaBoost was often discussed. Increasing standard deviation in $\mathrm{spirals}$ to $sd = 0.3$ and allowing for more flexibility in the base learners, AdaBoost overfits with increasing number of trees while the RF only saturates. The overfitting of AdaBoost here is quite typical as data is very noisy.