shipout/lastpageshipout/foregroundshipout/background
shipout/lastpage

FCIM / Predictive Modeling

Chapter 11: Introduction to Boosting / AdaBoost

Bernd Bischl

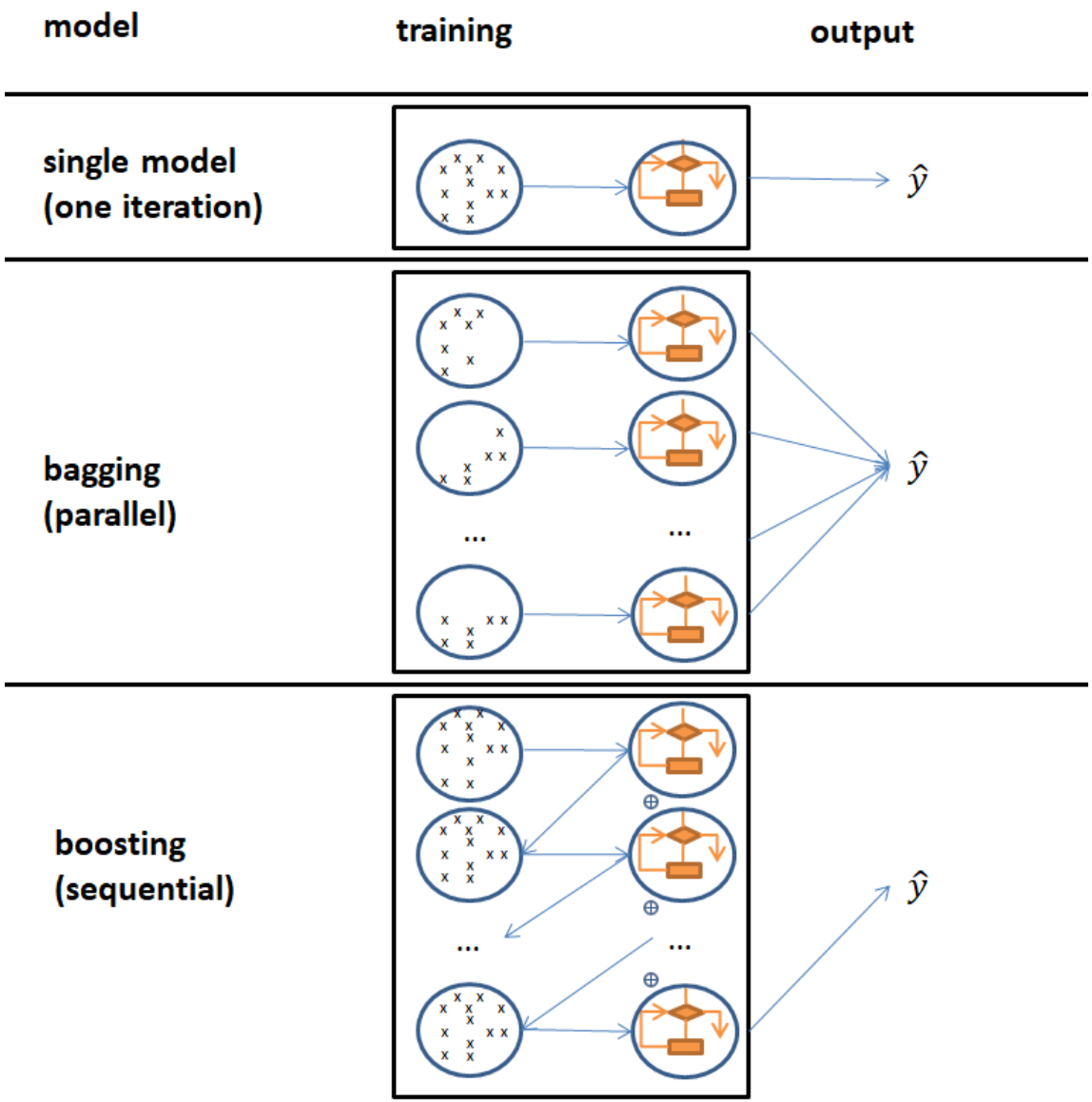
Department of Statistics - LMU Munich

Summer term 2020

INTRODUCTION TO BOOSTING

- Boosting is considered to be one of the most powerful learning ideas within the last twenty years.
- Originally designed for classification, (especially gradient) boosting handles regression (and many others 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



THE BOOSTING QUESTION

The first boosting algorithm ever was in fact no algorithm for practical purposes, but the solution for 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).

In practice it is typically easy to construct weak learners, but difficult to get a strong one.

THE BOOSTING ANSWER - ADABOOST

Any weak (base) learner can be iteratively boosted to become also a strong learner (Schapire and Freund, 1990). The proof of this ground-breaking idea generated the first boosting algorithm.

- The **AdaBoost** (Adaptive Boosting) algorithm is a **boosting** method for binary classification by Freund and Schapire (1997).
- 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 the „harder“ instances.

Leo Breiman (referring to the success of AdaBoost):
„Boosting is the best off-the-shelf classifier in the world.“

THE BOOSTING ANSWER - ADABOOST

- Assume a target variable y encoded as $\{-1, 1\}$, and weak base learners (e.g. tree stumps) from a hypothesis space $\mathcal{B}$.
- Base learner models $b^{[m]}$ are binary classifiers that map to $\mathcal{Y} = \{-1, +1\}$. We might sometimes write $b(\mathbf{x}, \theta^{[m]})$ instead.
- Predictions from all base models $b^{[m]}$ are combined to form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta^{[m]} b^{[m]}(\mathbf{x}).$$

- Weights $\beta^{[m]}$ are computed by the boosting algorithm. Their effect is to give higher values to base learners with higher predictive accuracy.
- The number of iterations M is the main tuning parameter.
- The discrete prediction function is $h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \in \{-1, 1\}$.

THE BOOSTING ANSWER - ADABOOST

Algorithm 1 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

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

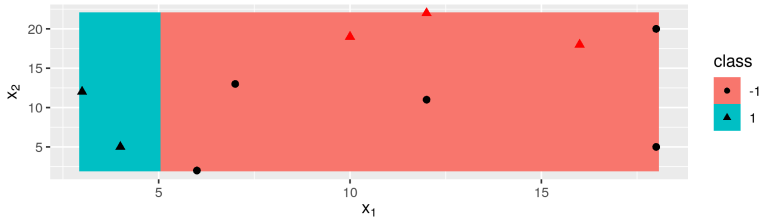
5: Compute: $\hat{\beta}^{[m]} = \frac{1}{2} \log\left(\frac{1-\text{err}^{[m]}}{\text{err}^{[m]}}\right)$

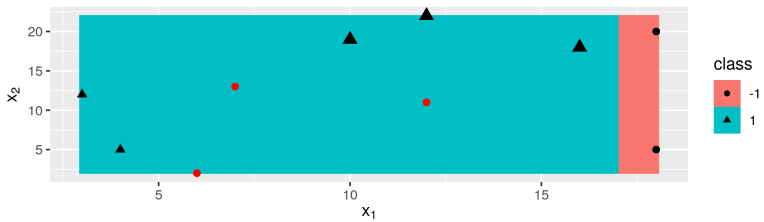
6: Set: $w^{[m+1](i)} = w^{[m](i)} \cdot \exp\left(\hat{\beta}^{[m]} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}}\right)$

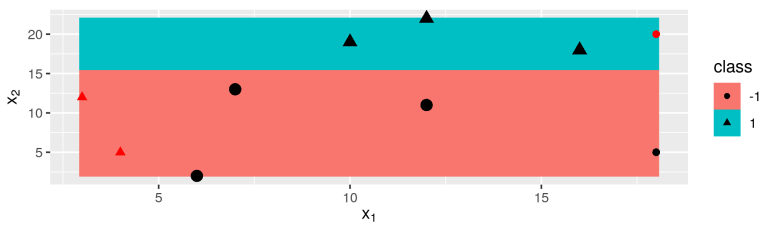
7: **end for**

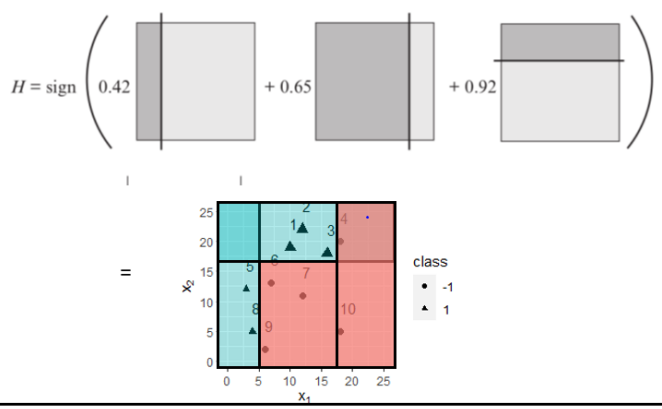
8: Output: $\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} \hat{\beta}^{[m]} \hat{b}^{[m]}(\mathbf{x})$

ADABOOST ILLUSTRATION









- $n = 10$ observations and $M = 3$ iterations
- Tree stumps as base learners.
- Size of label = $w^{[m](i)}$.
- First base learner $\text{err}^{[1]} = 0.3$
- Weight of the first base learner:
  $(1 - \text{err}^{[m]})/\text{err}^{[m]} = 2.33$;
  $0.5 \log(2.33) \approx 0.42$

New observation weights:

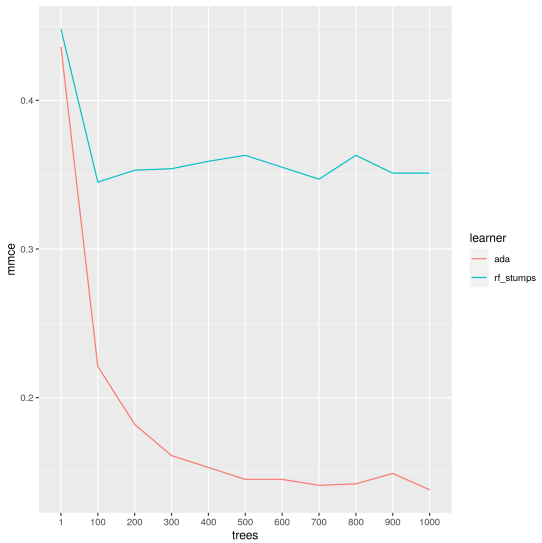
- $w^{[m](i)} \cdot \exp\left(\hat{\beta}^{[m]} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}}\right)$
- Prediction correct = no change.
- For 3 misclassified observations:

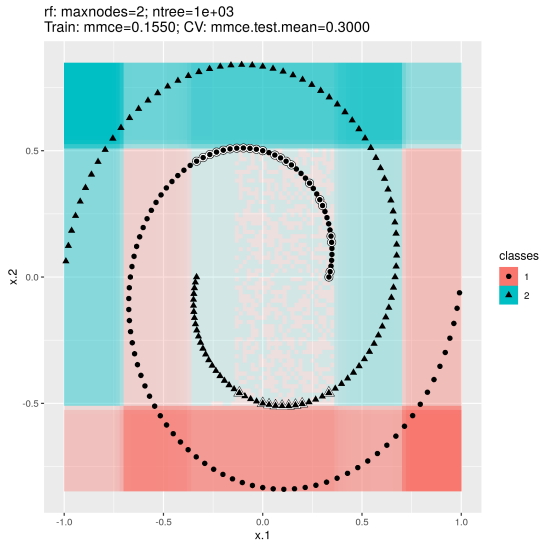
$$w^{[2](i)} = w^{[1](i)} \cdot \exp\left(\hat{\beta}^{[1]} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}}\right)$$

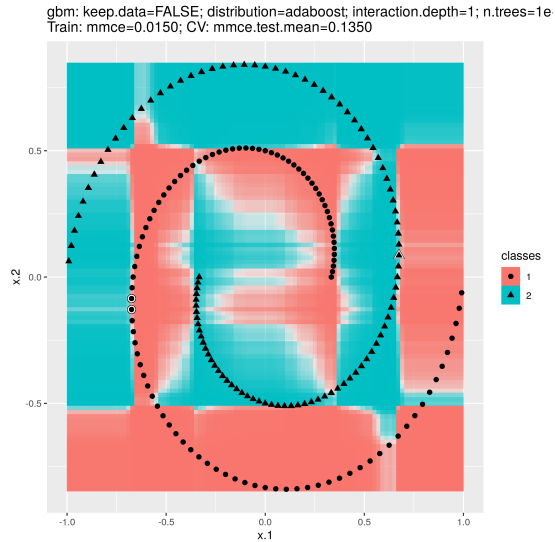
$$= \frac{1}{10} \cdot \exp\left(0.42 \cdot 1\right) \approx 0.15.$$

BAGGING VS BOOSTING STUMPS

Random Forest versus AdaBoost (both with stumps) on `Spirals` data from `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. Random Forest versus AdaBoost on `Spirals` data from `mlbench` ($n = 200$, $sd = 0.3$). With 5×5 repeated CV. AdaBoost overfits with increasing number of trees while the RF only saturates. The overfitting of AdaBoost here is quite typical as the data is very noisy.