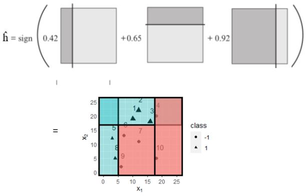
Introduction to Machine Learning

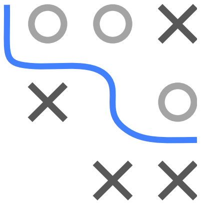
Boosting
Gradient Boosting: Introduction and AdaBoost



Learning goals

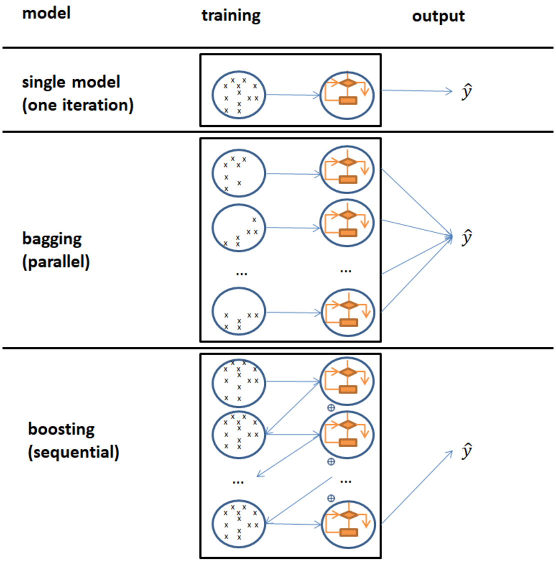
- Understand general idea of boosting
- Learn AdaBoost algorithm
- Understand difference between bagging and boosting

INTRODUCTION TO BOOSTING

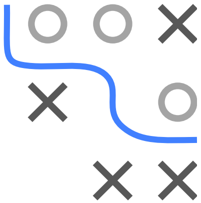


- Boosting is considered to be one of the most powerful learning ideas within the last twenty years.
- Originally designed for classification, (especially gradient) boosting handles regression (and many other supervised tasks) naturally nowadays.
- Homogeneous ensemble method (like bagging), but fundamentally 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



THE BOOSTING QUESTION



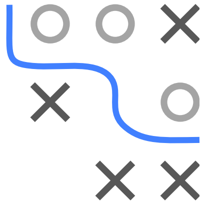
The first boosting algorithm ever was in fact no algorithm for practical purposes, but the solution for a theoretical problem:

“Does the existence of a weak learner for a certain problem imply the existence of a strong learner?” ▸ Kearns n.d.

- **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 build a strong one.

THE BOOSTING ANSWER - ADABOOST

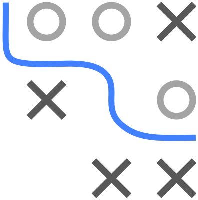


Any weak (base) learner can be iteratively boosted to become a strong learner. The proof of this ground-breaking idea generated the first boosting algorithm.

- The **AdaBoost** (Adaptive Boosting) algorithm is a **boosting** method for binary classification by ▸ Freund, Schapire, et al. 1996.
- 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.

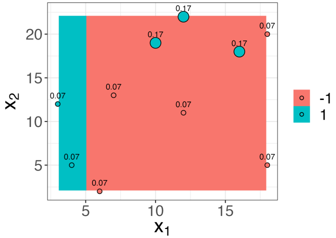
Leo Breiman (referring to the success of AdaBoost):
"Boosting is the best off-the-shelf classifier in the world."

ADABOOST ILLUSTRATION



Example description

- $n = 10$ 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: $w^{[2](i)} \approx 0.17$

BAGGING VS BOOSTING

Random forest

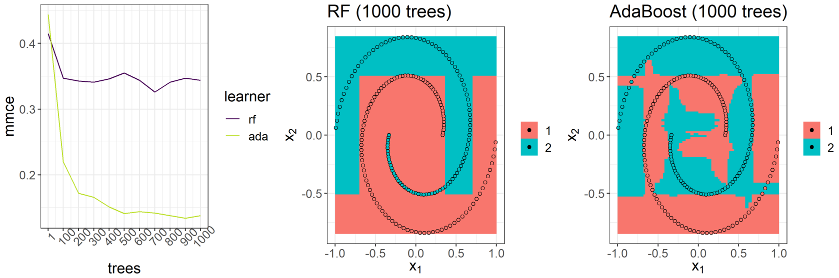
- Base learners are typically deeper decision trees (not only stumps!)
- Equal weights for base learners
- Base learners independent of each other
- Aim: variance reduction
- Tends **not** to overfit

AdaBoost

- Base learners are weak learners, e.g., only stumps
- Base learners have different weights depending on their predictive accuracy
- Sequential algorithm, hence order matters
- Aim: bias and variance reduction
- Tends to overfit

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. Increasing standard deviation 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.