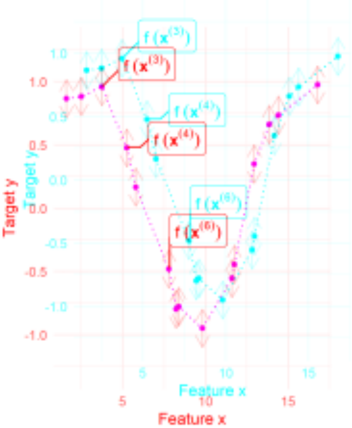
Introduction to Machine Learning

Boosting

Gradient Boosting: Concept

Gradient Boosting: Concept



Learning goals

Learning goals

- Understand idea of forward stagewise modelling
- Understand fitting process of gradient boosting for regression problems

FORWARD STAGEWISE ADDITIVE MODELING / 2

Why is gradient boosting a good choice for this problem?

- Because of the additive structure it is difficult to jointly minimize $\mathcal{R}_{\text{emp}}(f)$ w.r.t. $\left(\left(\alpha^{[1]}, \boldsymbol{\theta}^{[1]}\right), \dots, \left(\alpha^{[M]}, \boldsymbol{\theta}^{[M]}\right)\right)$, which is a very high-dimensional parameter space (though this is less of a problem nowadays, especially in the case of numeric parameter spaces).
- Considering trees as base learners is worse as we would have to grow M trees in parallel so they work optimally together as an ensemble.
- Stagewise additive modeling has nice properties, which we want to make use of, e.g. for regularization, early stopping, ...

FORWARD STAGEWISE ADDITIVE MODELING / 3

Hence, we add additive components in a greedy fashion by sequentially minimizing the risk only w.r.t. the next additive component:

$$\min_{\alpha, \boldsymbol{\theta}} \sum_{i=1}^{n} L\left(y^{(i)}, \hat{f}^{[m-1]}\left(\mathbf{x}^{(i)}\right) + \alpha b\left(\mathbf{x}^{(i)}, \boldsymbol{\theta}\right)\right)$$

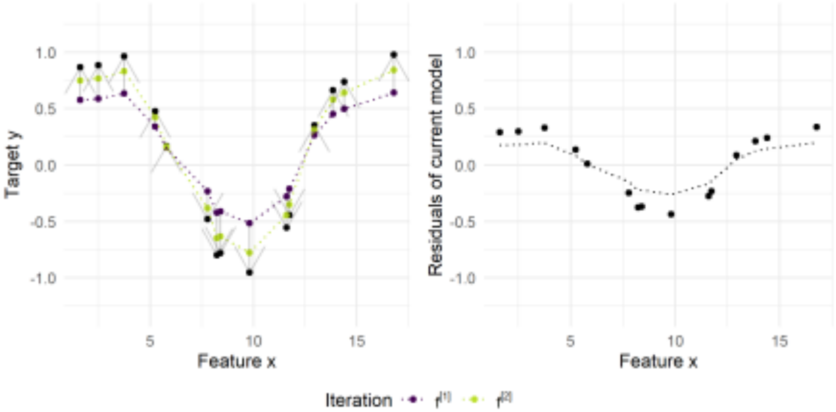
Doing this iteratively is called **forward stagewise additive modeling**.

Algorithm Forward Stagewise Additive Modeling.

Initialize $\hat{f}^{[0]}(\mathbf{x})$ with loss optimal constant model
for $m = 1 \to M$ **do**
$\quad (\hat{\alpha}^{[m]}, \hat{\boldsymbol{\theta}}^{[m]}) = \arg\min_{\alpha, \boldsymbol{\theta}} \sum_{i=1}^{n} L\left(y^{(i)}, \hat{f}^{[m-1]}\left(\mathbf{x}^{(i)}\right) + \alpha b\left(\mathbf{x}^{(i)}, \boldsymbol{\theta}\right)\right)$
$\quad$ Update $\hat{f}^{[m]}(\mathbf{x}) \leftarrow \hat{f}^{[m-1]}(\mathbf{x}) + \hat{\alpha}^{[m]} b\left(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[m]}\right)$
end for

GRADIENT BOOSTING / 2

Iteration 2:
Let's move our function $f\left(\mathbf{x}^{(i)}\right)$ a fraction towards the pseudo-residuals with a learning rate of $\alpha = 0.6$.



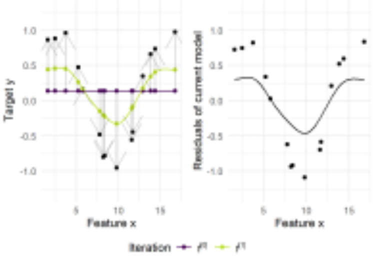
GRADIENT BOOSTING / 3

To parameterize a model in this way is pointless, as it just memorizes the instances of the training data.

So, we restrict our additive components to $b\left(\mathbf{x}, \boldsymbol{\theta}^{[m]}\right) \in \mathcal{B}$.
The pseudo-residuals are calculated exactly as stated above, then we fit a simple model $b(\mathbf{x}, \boldsymbol{\theta}^{[m]})$ to them:

$$\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(\tilde{r}^{[m](i)} - b(\mathbf{x}^{(i)}, \boldsymbol{\theta})\right)^2 .$$

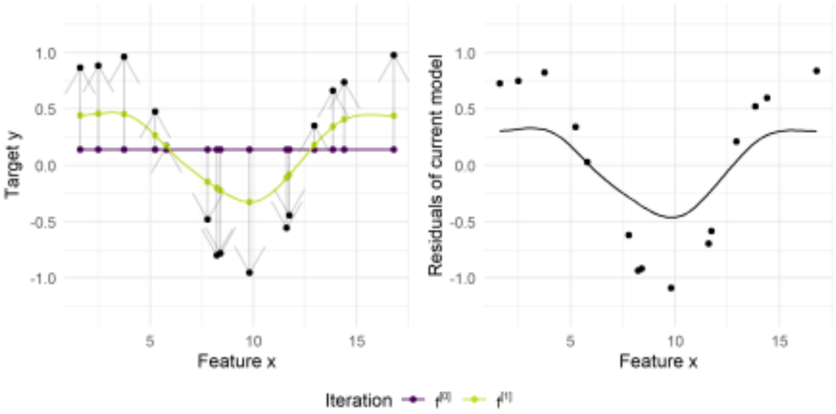
So, evaluated on the training data, our $b(\mathbf{x}, \boldsymbol{\theta}^{[m]})$ corresponds as closely as possible to the negative loss function gradient and generalizes over the whole space.



GRADIENT BOOSTING / 4

In a nutshell: One boosting iteration is exactly one approximated gradient descent step in function space, which minimizes the empirical risk as much as possible.

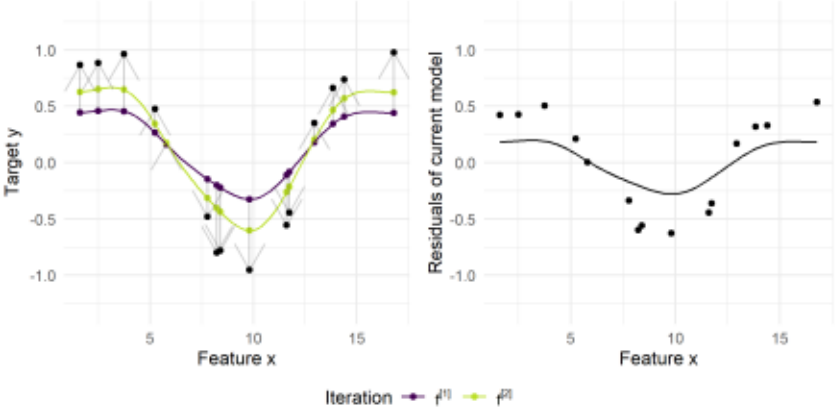
Iteration 1:



GRADIENT BOOSTING / 5

Instead of moving the function values for each observation by a fraction closer to the observed data, we fit a regression base learner to the pseudo-residuals (right plot).

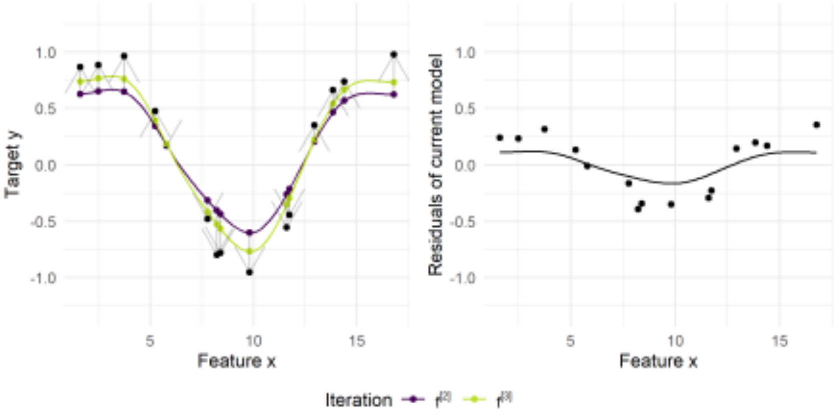
Iteration 2:



GRADIENT BOOSTING / 6

This base learner is then added to the current state of the ensemble weighted by the learning rate (here: $\alpha = 0.4$) and for the next iteration again the pseudo-residuals of the adapted ensemble are calculated and a base learner is fitted to them.

Iteration 3:



GRADIENT BOOSTING ALGORITHM

Algorithm Gradient Boosting Algorithm.

Initialize $\hat{f}^{[0]}(\mathbf{x}) = \arg\min_{\theta_0 \in \mathbb{R}} \sum_{i=1}^{n} L(y^{(i)}, \theta_0)$

for $m = 1 \to M$ **do**

  For all i: $\tilde{r}^{[m](i)} = -\left[\frac{\partial L(y,f)}{\partial f}\right]_{f=\hat{f}^{[m-1]}(\mathbf{x}^{(i)}), y=y^{(i)}}$

  Fit a regression base learner to the vector of pseudo-residuals $\tilde{r}^{[m]}$:

  $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (\tilde{r}^{[m](i)} - b(\mathbf{x}^{(i)}, \boldsymbol{\theta}))^2$

  Set $\alpha^{[m]}$ to α being a small constant value or via line search

  Update $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \alpha^{[m]} b(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[m]})$

end for

Output $\hat{f}(\mathbf{x}) = \hat{f}^{[M]}(\mathbf{x})$

Note that we also initialize the model in a loss-optimal manner.

LINE SEARCH

The learning rate in gradient boosting influences how fast the algorithm converges. Although a small constant learning rate is commonly used in practice, it can also be replaced by a line search.

Line search is an iterative approach to find a local minimum. In the case of setting the learning rate, the following one-dimensional optimization problem has to be solved:

$$\hat{\alpha}^{[m]} = \arg\min_{\alpha} \sum_{i=1}^{n} L\left(y^{(i)}, f^{[m-1]}(\mathbf{x}) + \alpha b(\mathbf{x}, \hat{\theta}^{[m]})\right)$$

Optionally, an (inexact) backtracking line search can be used to find the $\alpha^{[m]}$ that minimizes the above equation.