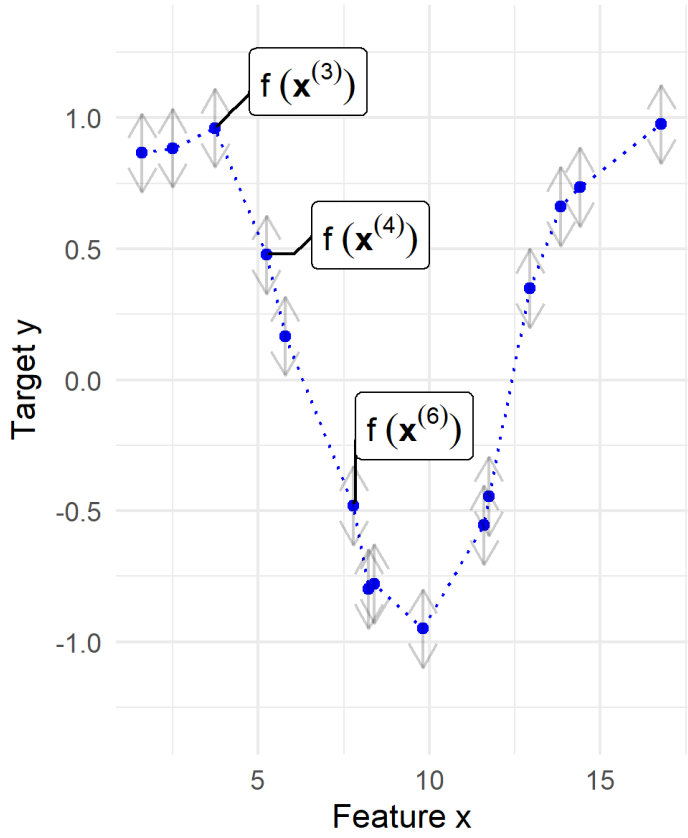
Introduction to Machine Learning

Gradient Boosting



Learning goals

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

FORWARD STAGEWISE ADDITIVE MODELING

Assume a regression problem for now (as this is simpler to explain); and assume a space of base learners $\mathcal{B}$.

We want to learn an additive model:

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

Hence, we minimize the empirical risk:

$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right) = \sum_{i=1}^{n} L\left(y^{(i)}, \sum_{m=1}^{M} \beta^{[m]} b(\mathbf{x}, \boldsymbol{\theta}^{[m]})\right)$$

FORWARD STAGEWISE ADDITIVE MODELING

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. $\left(\left(\beta^{[1]}, \boldsymbol{\theta}^{[1]}\right), \ldots, \left(\beta^{[M]}, \boldsymbol{\theta}^{[M]}\right)\right)$, 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

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

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

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

Algorithm 1 Forward Stagewise Additive Modeling.

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

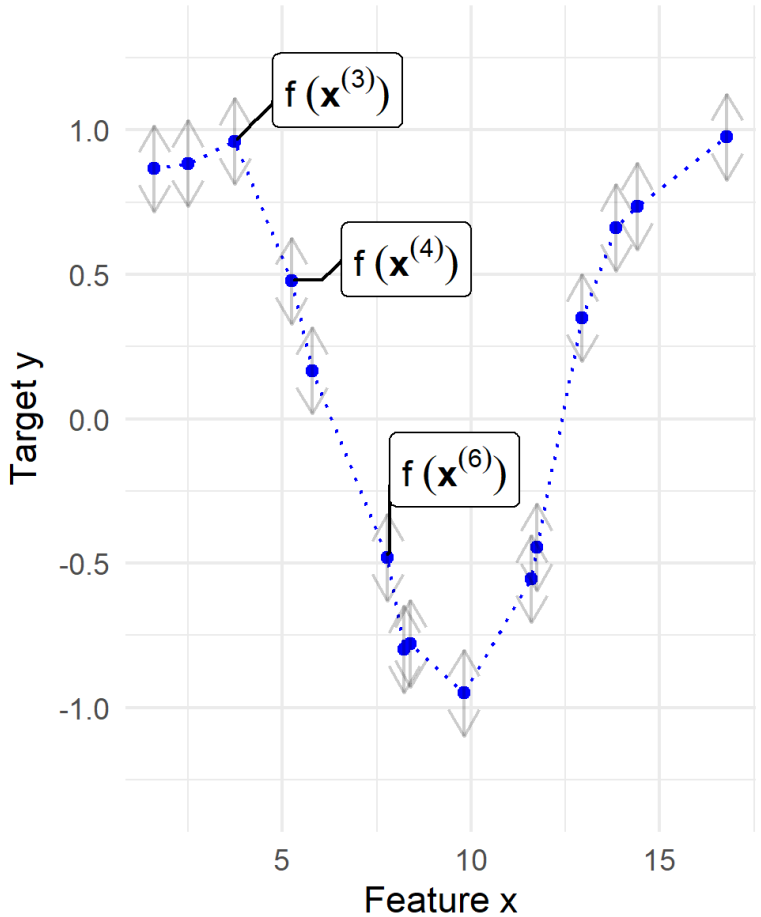
GRADIENT BOOSTING

This is not really an algorithm, but an abstract principle. To find $b\left(\mathbf{x}, \boldsymbol{\theta}^{[m]}\right)$ and $\beta^{[m]}$, we use gradient descent, but in function space!

Consider a model f whose predictions we can arbitrarily define for each training $\mathbf{x}^{(i)}$, i.e., f is a finite vector

$$\left(f\left(\mathbf{x}^{(1)}\right), \ldots, f\left(\mathbf{x}^{(n)}\right)\right)^{\top}$$

This implies n parameters $f\left(\mathbf{x}^{(i)}\right)$ (and the model would provide no generalization...). Also, we let's assume L to be differentiable.

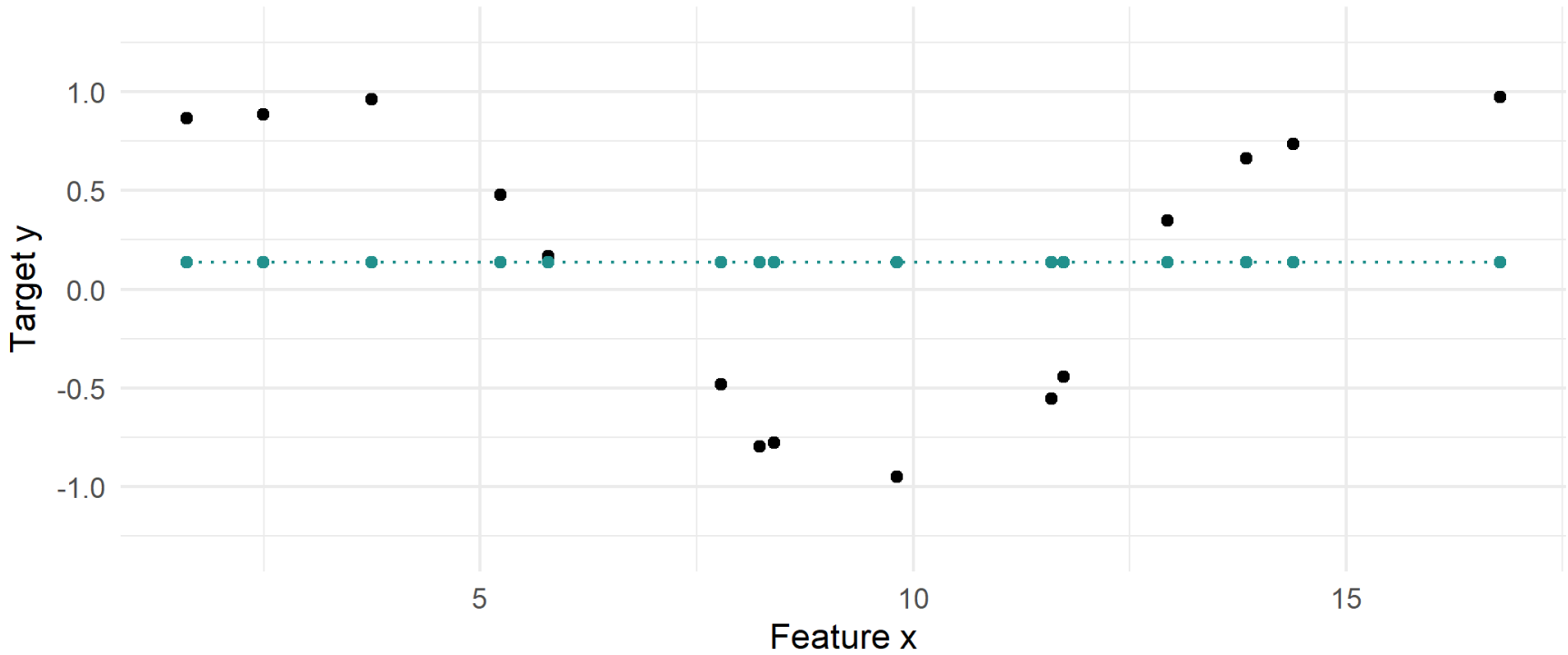


GRADIENT BOOSTING

Aim: Define a movement in function space so we can push our current function towards the data points.

Given: Regression problem with one feature x and target variable y.

Initialization: Set all parameters to the optimal constant value (e.g., the mean of y for $L2$).



PSEUDO RESIDUALS

How do we distort our f to move it towards the labels and reduce risk? Let's minimize risk with GD.

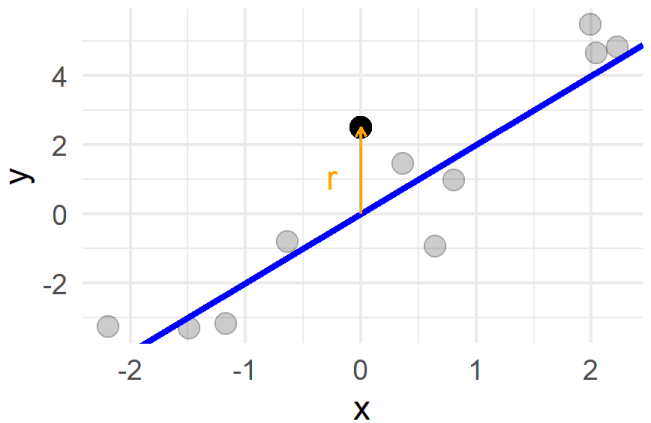
So, we calculate the (negative) gradient of the risk for each parameter, which (weirdly) here are the outputs $f\left(\mathbf{x}^{(i)}\right)$ (0 if $i \neq j$):

$$\tilde{r}^{(i)} = -\frac{\partial \mathcal{R}_{\text{emp}}}{\partial f\left(\mathbf{x}^{(i)}\right)} = -\frac{\partial \sum_j L(y^{(j)}, f(\mathbf{x}^{(j)}))}{\partial f\left(\mathbf{x}^{(i)}\right)} = -\frac{\partial L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)}{\partial f\left(\mathbf{x}^{(i)}\right)}.$$

At each point, we would like to change the output via: $\tilde{r}(f) = -\frac{\partial L(y,f)}{\partial f}$.

L2 Example: The PRs match the usual residuals:

$$\tilde{r}(f) = -\frac{\partial 0.5(y-f)^2}{\partial f} = y - f$$



BOOSTING AS GRADIENT DESCENT

Combining this with "forward stagewise modeling", we are at $f^{[m-1]}$ during minimization. Here, we calculate the direction of the negative gradient or vector of PRs:

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

The gradient descent update for each vector component of f is:

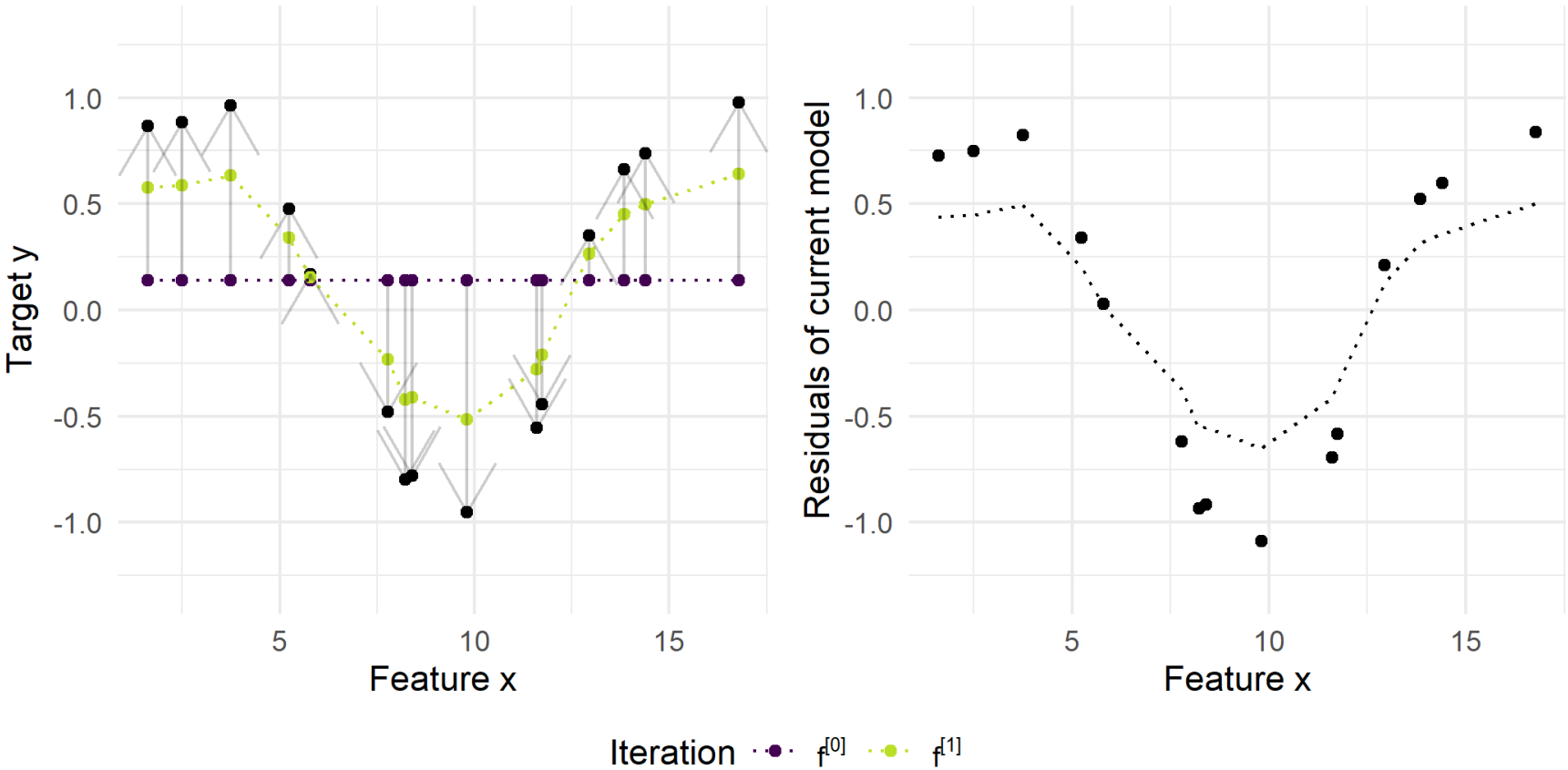
$$f^{[m]}(\mathbf{x}^{(i)}) = f^{[m-1]}(\mathbf{x}^{(i)}) + \beta \tilde{r}^{[m](i)}.$$

Like this, we should "nudge" f in the direction best risk reduction.

GRADIENT BOOSTING

Iteration 1:

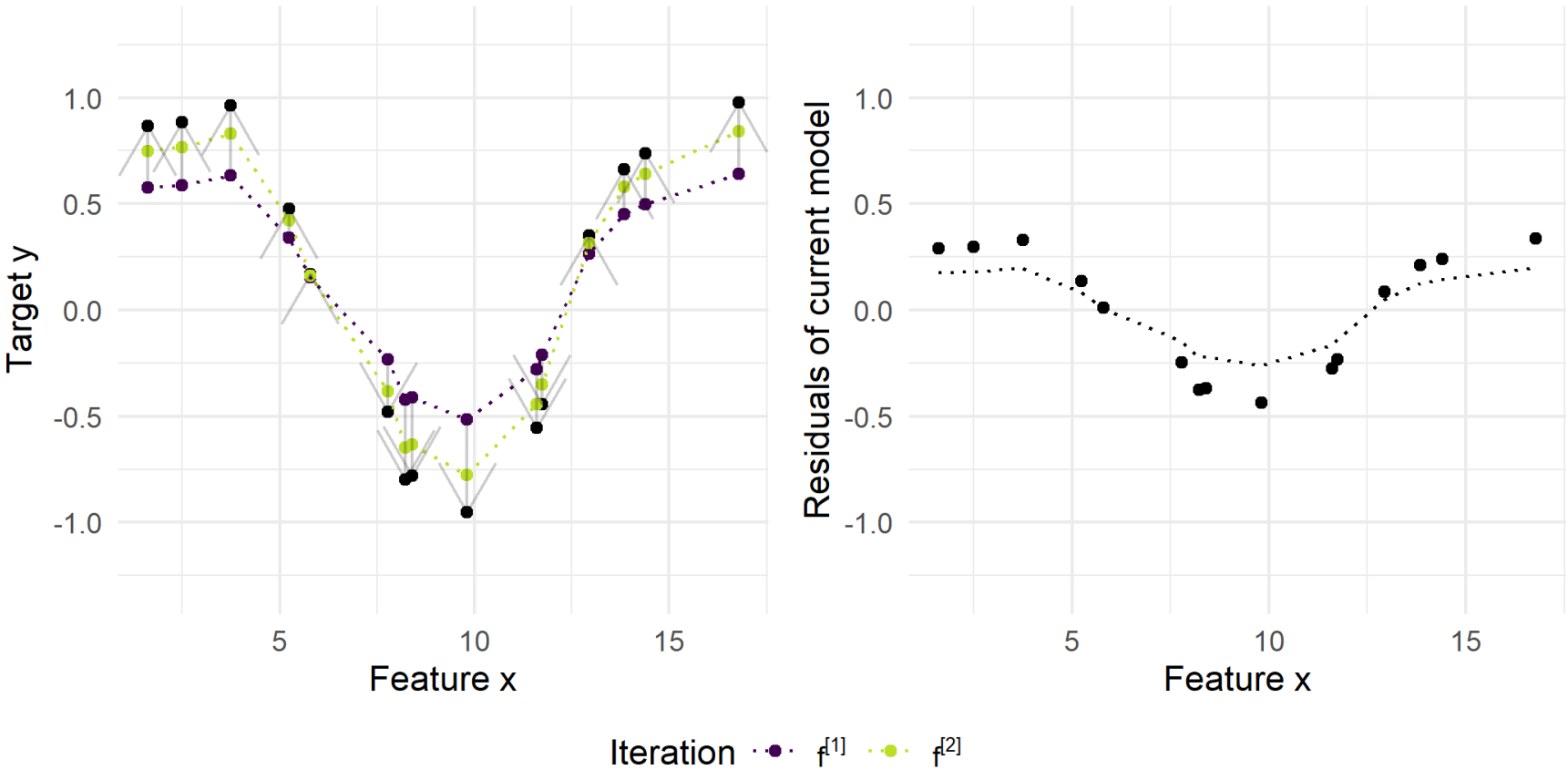
Let's move our function $f\left(\mathbf{x}^{(i)}\right)$ a fraction towards the pseudo-residuals with a learning rate of $\beta = 0.6$.



GRADIENT BOOSTING

Iteration 2:

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



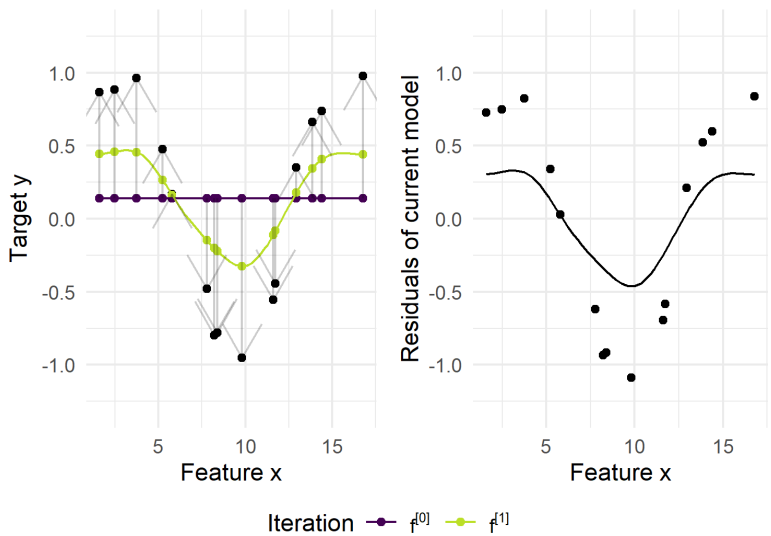
GRADIENT BOOSTING

As said, such a model parameterization is pointless.

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:

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

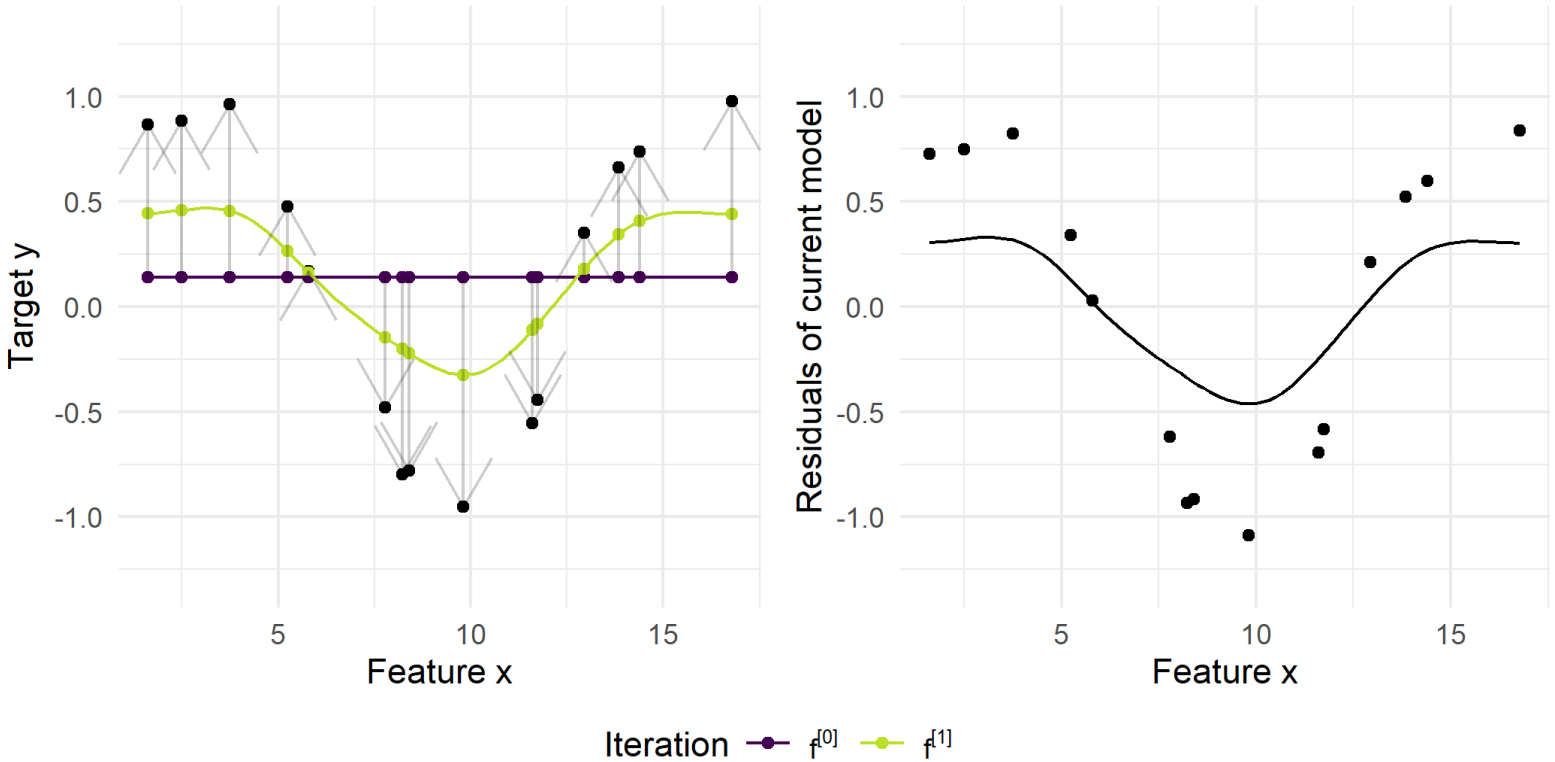
So, evaluated on the training data, $b(\mathbf{x}, \boldsymbol{\theta}^{[m]})$ corresponds as closely as possible to the negative risk gradient and generalizes over $\mathcal{X}$.



GRADIENT BOOSTING

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.

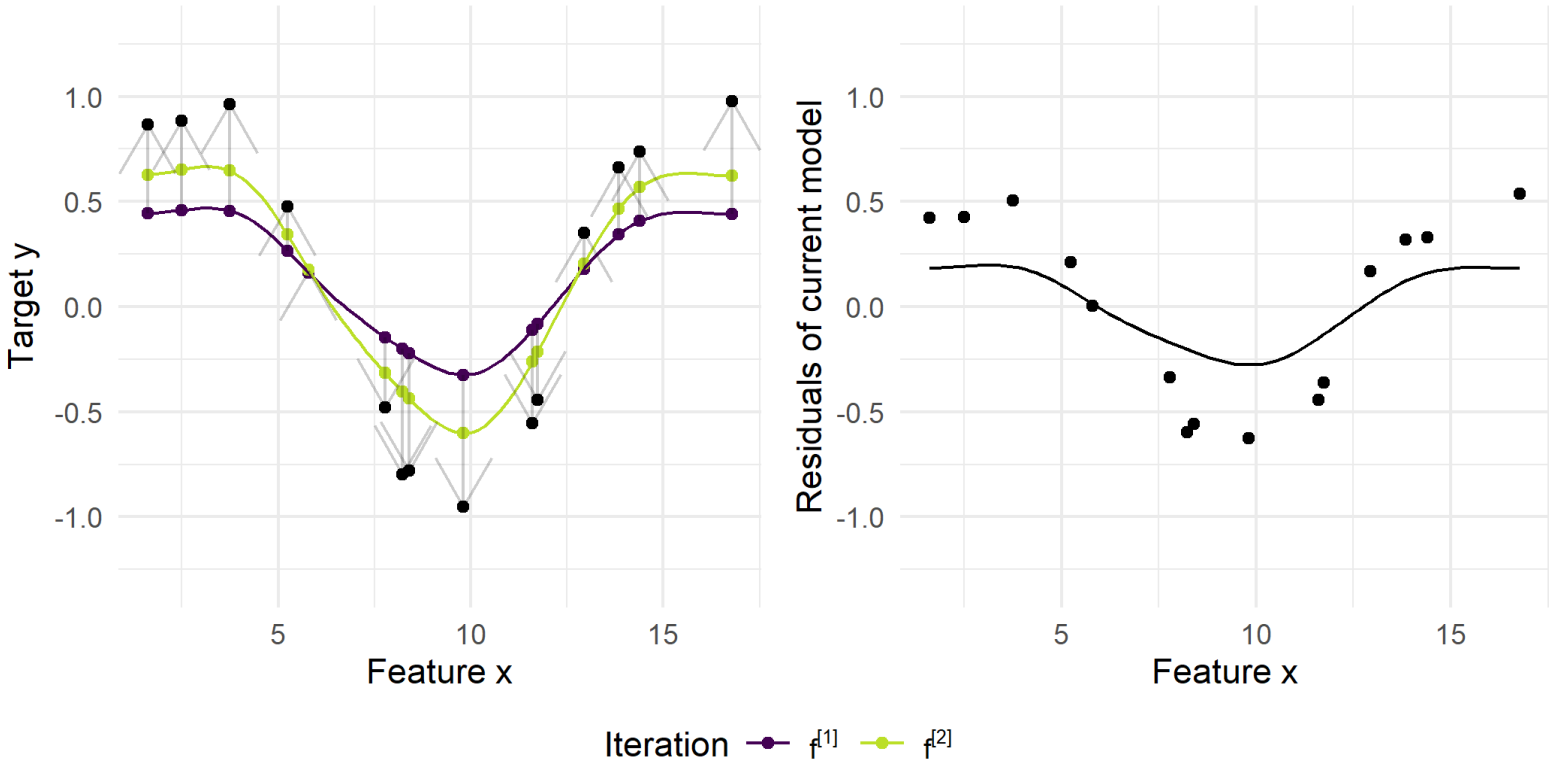
Iteration 1:



GRADIENT BOOSTING

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).

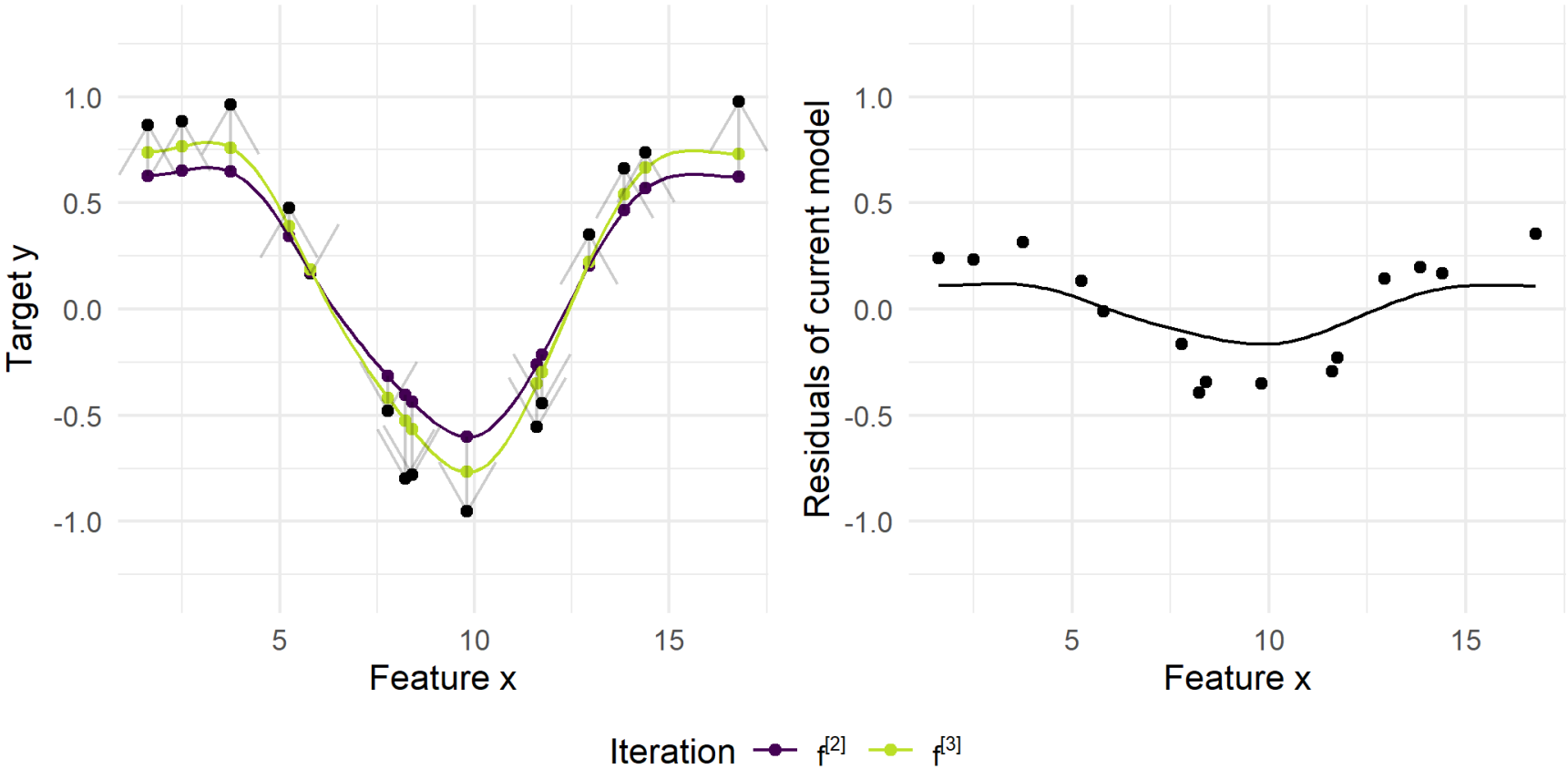
Iteration 2:



GRADIENT BOOSTING

This BL is added to the of the ensemble, weighted by a learning rate (here: $\beta = 0.4$). Then we iterate.

Iteration 3:



GRADIENT BOOSTING ALGORITHM

Algorithm 2 Gradient Boosting Algorithm.

1: Initialize $\hat{f}^{[0]}(\mathbf{x}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} L(y^{(i)}, b(\mathbf{x}^{(i)}, \boldsymbol{\theta}))$
2: **for** $m = 1 \rightarrow M$ **do**
3: For all i: $\tilde{r}^{[m](i)} = - \left[\frac{\partial L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)}{\partial f\left(\mathbf{x}^{(i)}\right)} \right]_{f=\hat{f}^{[m-1]}}$
4: Fit a regression base learner to the pseudo-residuals $\tilde{r}^{[m](i)}$:
5: $\hat{\boldsymbol{\theta}}^{[m]} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (\tilde{r}^{[m](i)} - b(\mathbf{x}^{(i)}, \boldsymbol{\theta}))^2$
6: Set $\beta^{[m]}$ to β being a small constant value or via line search
7: Update $\hat{f}^{[m]}(\mathbf{x}) = \hat{f}^{[m-1]}(\mathbf{x}) + \beta^{[m]} b(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[m]})$
8: **end for**
9: 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, as always, influences how we converge. Although a small constant LR is commonly used, we can also run line search,

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

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