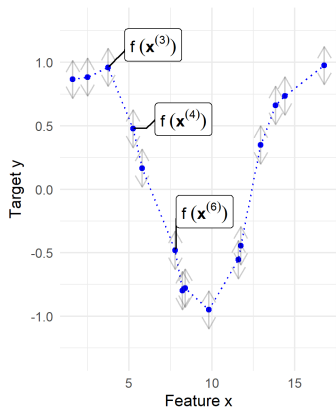


Introduction to Machine Learning

Boosting

Gradient Boosting: Concept



Learning goals

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

FORWARD STAGEWISE ADDITIVE MODELING

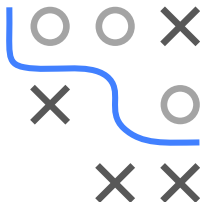
Assume a regression problem for now (as this is simpler to explain); and assume a space of base learners \mathcal{B} .

We want to learn an additive model:

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

Hence, we minimize the empirical risk:

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



GRADIENT BOOSTING

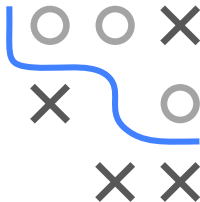
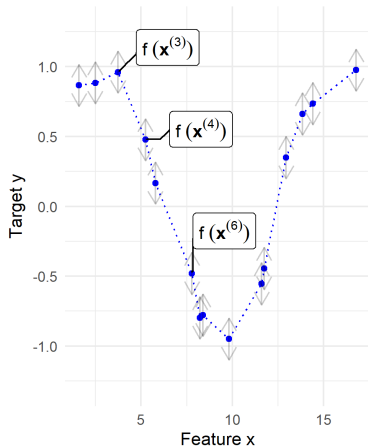
The algorithm we just introduced is not really an algorithm, but rather an abstract principle. We need to find the new additive component $b(\mathbf{x}, \theta^{[m]})$ and its weight coefficient $\alpha^{[m]}$ in each iteration m . This can be done by gradient descent, but in function space.

Thought experiment: Consider a completely non-parametric model f whose predictions we can arbitrarily define on every point of the training data $\mathbf{x}^{(i)}$. So we basically specify f as a discrete, finite vector.

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

This implies n parameters $f(\mathbf{x}^{(i)})$ (and the model would provide no generalization...).

Furthermore, we assume our loss function $L(\cdot)$ to be differentiable.

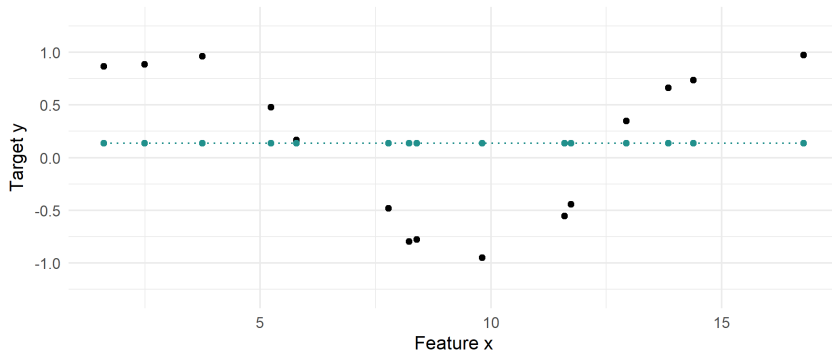
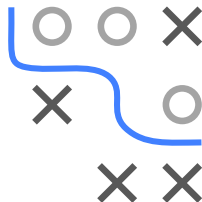


GRADIENT BOOSTING

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 have to distort this function to move it towards the observations and drive loss down?

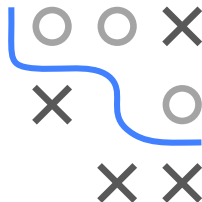
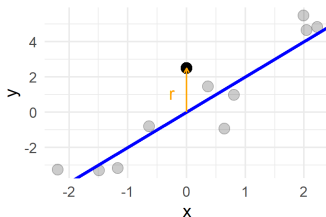
We minimize the risk of such a model with gradient descent (yes, this makes no sense, suspend all doubts for a few seconds).

So, we calculate the gradient at a point of the parameter space, that is, the derivative w.r.t. each component of the parameter vector (which is 0 for all terms with $i \neq j$):

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

Reminder: The pseudo-residuals $\tilde{r}(f)$ match the usual residuals for the squared loss:

$$\begin{aligned} -\frac{\partial L(y, f(\mathbf{x}))}{\partial f(\mathbf{x})} &= -\frac{\partial 0.5(y - f(\mathbf{x}))^2}{\partial f(\mathbf{x})} \\ &= y - f(\mathbf{x}) \end{aligned}$$



BOOSTING AS GRADIENT DESCENT

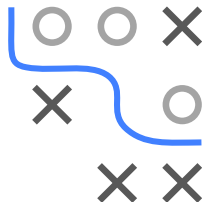
Combining this with the iterative additive procedure of “forward stagewise modeling”, we are at the spot $f^{[m-1]}$ during minimization. At this point, we now calculate the direction of the negative gradient or also called pseudo-residuals $\tilde{r}^{[m](i)}$:

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

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

$$f^{[m]}(\mathbf{x}^{(i)}) = f^{[m-1]}(\mathbf{x}^{(i)}) - \alpha \frac{\partial L(y^{(i)}, f(\mathbf{x}^{(i)}))}{\partial f^{[m-1]}(\mathbf{x}^{(i)})}.$$

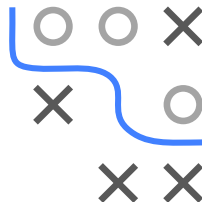
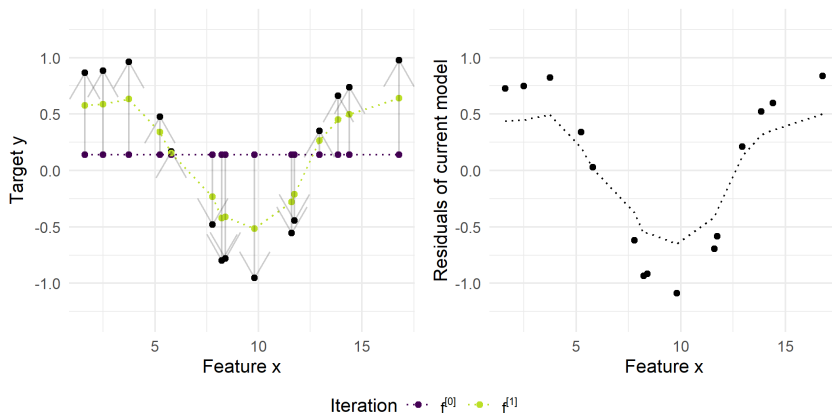
This tells us how we could “nudge” our whole function f in the direction of the data to reduce its empirical risk.



GRADIENT BOOSTING

Iteration 1:

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



GRADIENT BOOSTING ALGORITHM

Algorithm Gradient Boosting Algorithm.

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

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

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

8: end for

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:

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

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

