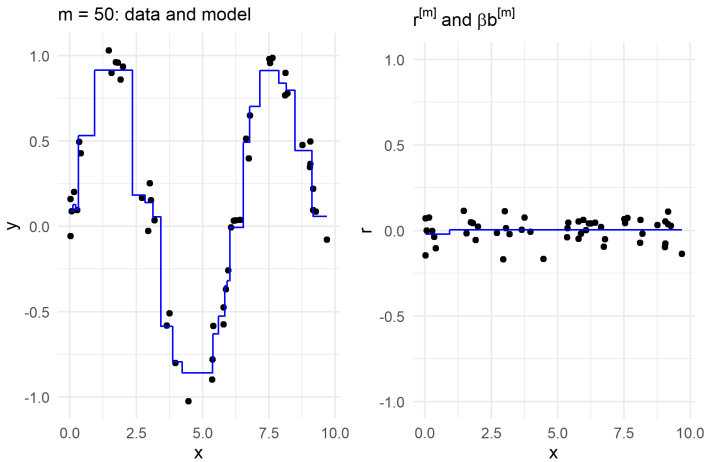
Introduction to Machine Learning

Gradient Boosting with Trees



Learning goals

- See how gradient boosting process is adapted for trees
- Understand relationship between model structure and interaction depth
- Understand multiclass extension for gradient boosting with trees

GRADIENT BOOSTING WITH TREES

Trees are mainly used as base learners for gradient boosting in ML. A great deal of research has been done on this combination so far, and it often provides the best results.

Reminder: advantages of trees

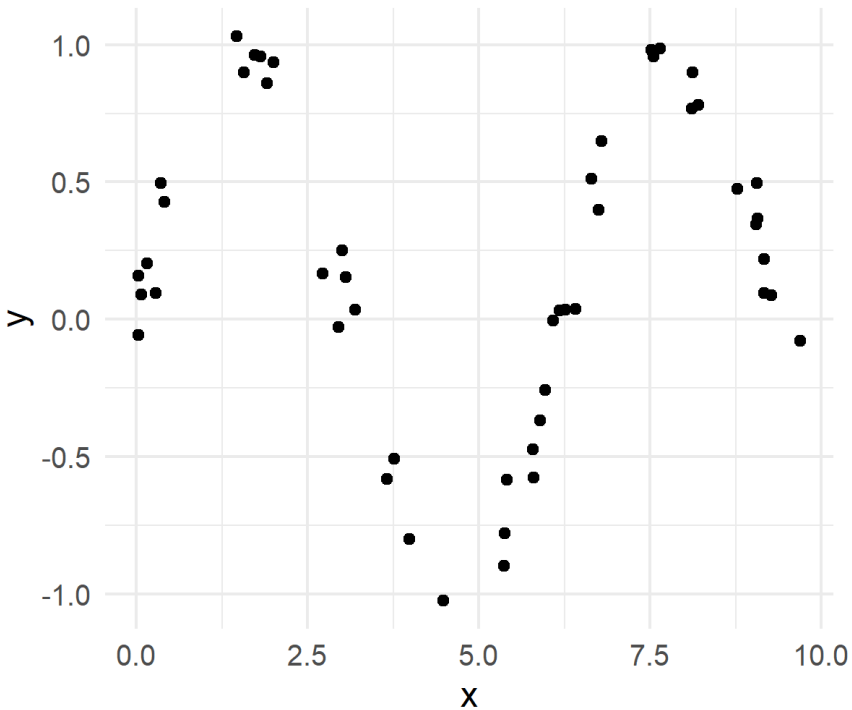
- No problems with categorical features.
- No problems with outliers in feature values.
- No problems with missing values.
- No problems with monotone transformations of features.
- Trees (and stumps!) can be fitted quickly, even for large n.
- Trees have a simple, built-in type of variable selection.

The gradient-boosted trees method retains all of them, and strongly improves the trees' predictive power. Furthermore, it is possible to adapt gradient boosting to tree learners in a targeted manner.

EXAMPLE 1

Simulation setting:

- Given: one feature x and one numeric target variable y of 50 observations.
- x is uniformly distributed between 0 and 10.
- y depends on x as follows: $y^{(i)} = \sin\left(x^{(i)}\right) + \epsilon^{(i)}$ with $\epsilon^{(i)} \sim \mathcal{N}(0, 0.01)$, $\forall i \in \{1, \ldots, 50\}$.



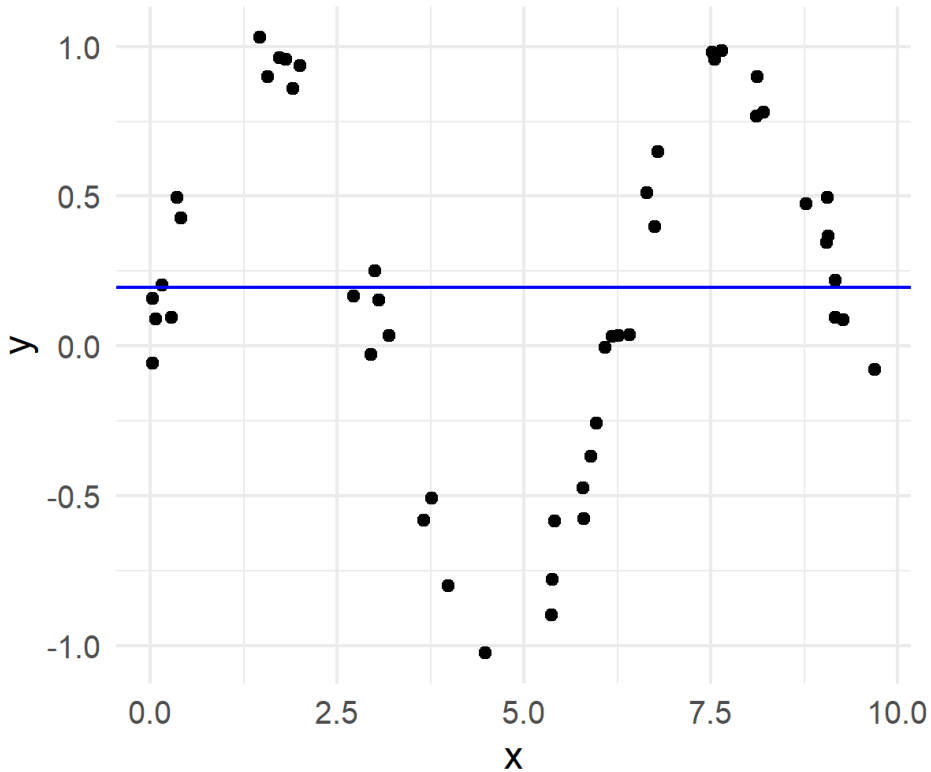
Aim: we want to fit a gradient boosting model to the data by using stumps as base learners.

Since we are facing a regression problem, we use *L2* loss.

EXAMPLE 1

Iteration 0: initialization by optimal constant (mean) prediction $\hat{f}^{[0](i)}(x) = \bar{y} \approx 0.2$.

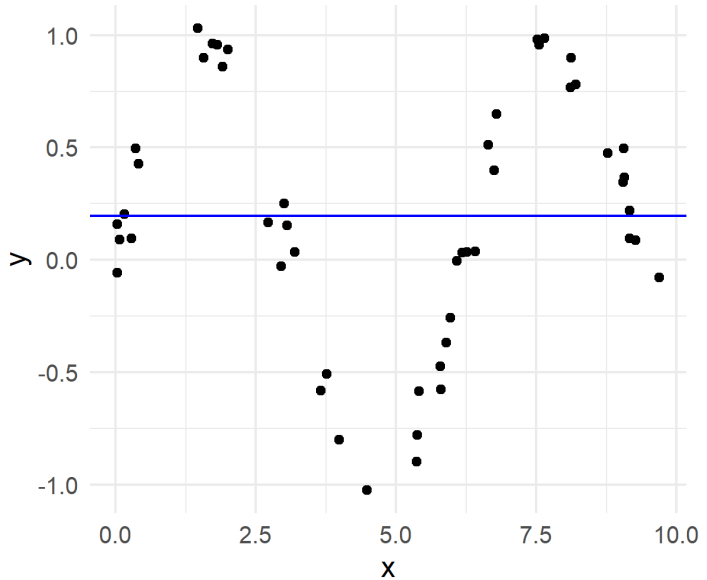
i	$x^{(i)}$	$y^{(i)}$	$\hat{f}^{[0]}$
1	0.03	0.16	0.20
2	0.03	-0.06	0.20
3	0.07	0.09	0.20
⋮	⋮	⋮	⋮
50	9.69	-0.08	0.20



EXAMPLE 1

Iteration 1: (1) Calculate pseudo-residuals $\tilde{r}^{[m](i)}$ and (2) fit a regression stump $b^{[m]}$.

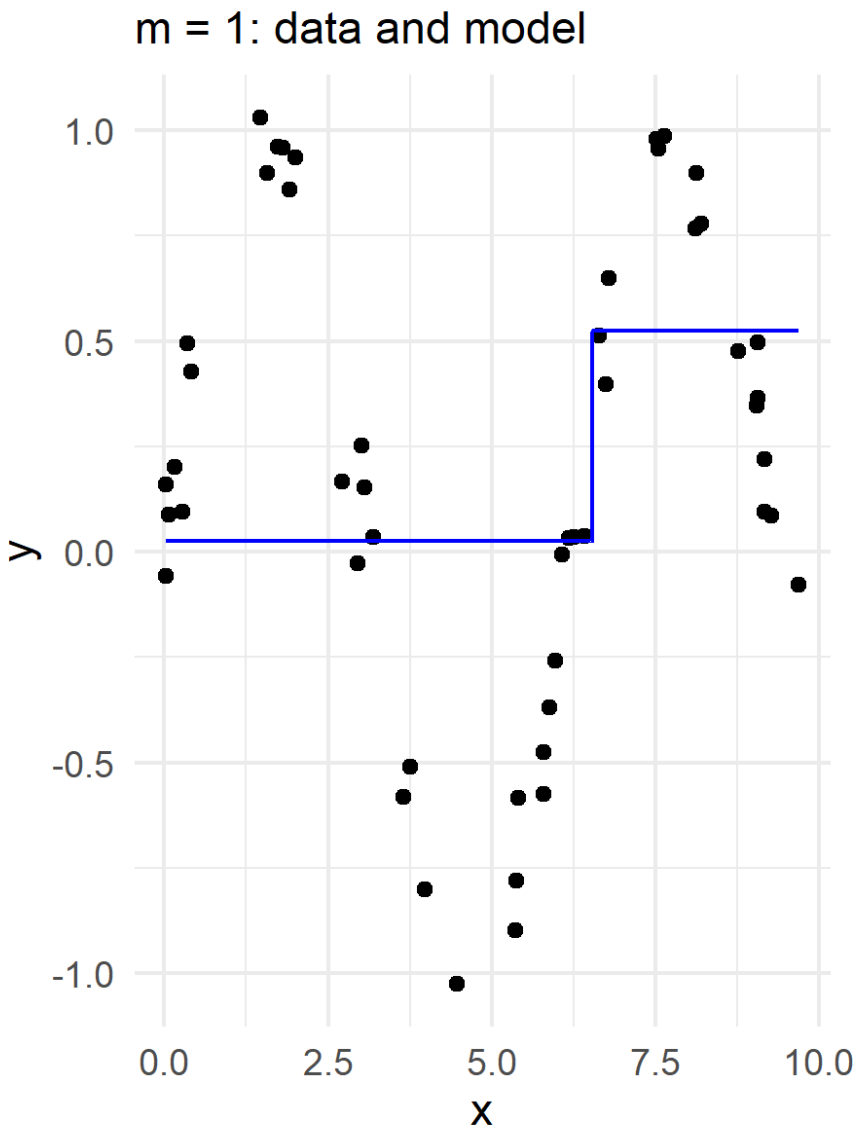
i	$x^{(i)}$	$y^{(i)}$	$\hat{f}^{[0]}$	$\tilde{r}^{[1](i)}$	$\hat{b}^{[1](i)}$
1	0.03	0.16	0.20	-0.04	-0.17
2	0.03	-0.06	0.20	-0.25	-0.17
3	0.07	0.09	0.20	-0.11	-0.17
⋮	⋮	⋮	⋮	⋮	⋮
50	9.69	-0.08	0.20	-0.27	0.33

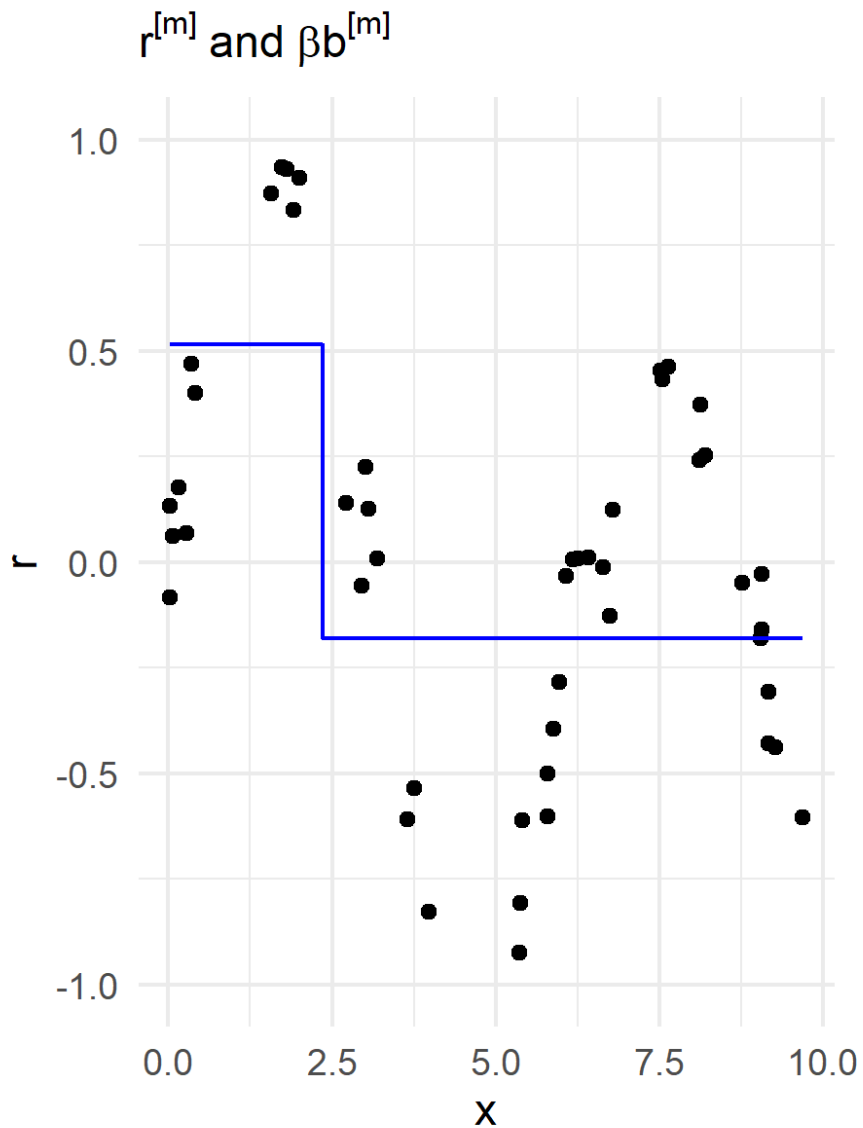


(3) Update model by $\hat{f}^{[1]}(x) = \hat{f}^{[0]}(x) + \hat{b}^{[1]}$.

EXAMPLE 1

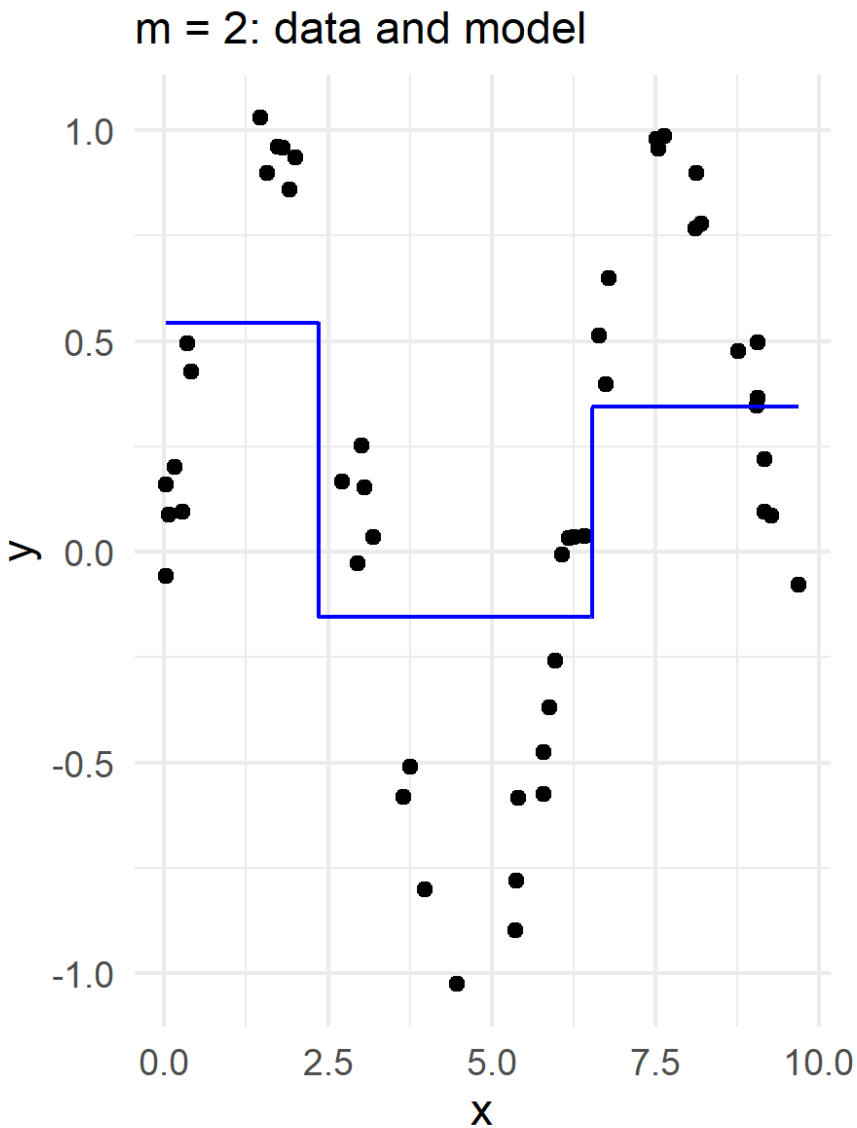
Repeat step (1) to (3):

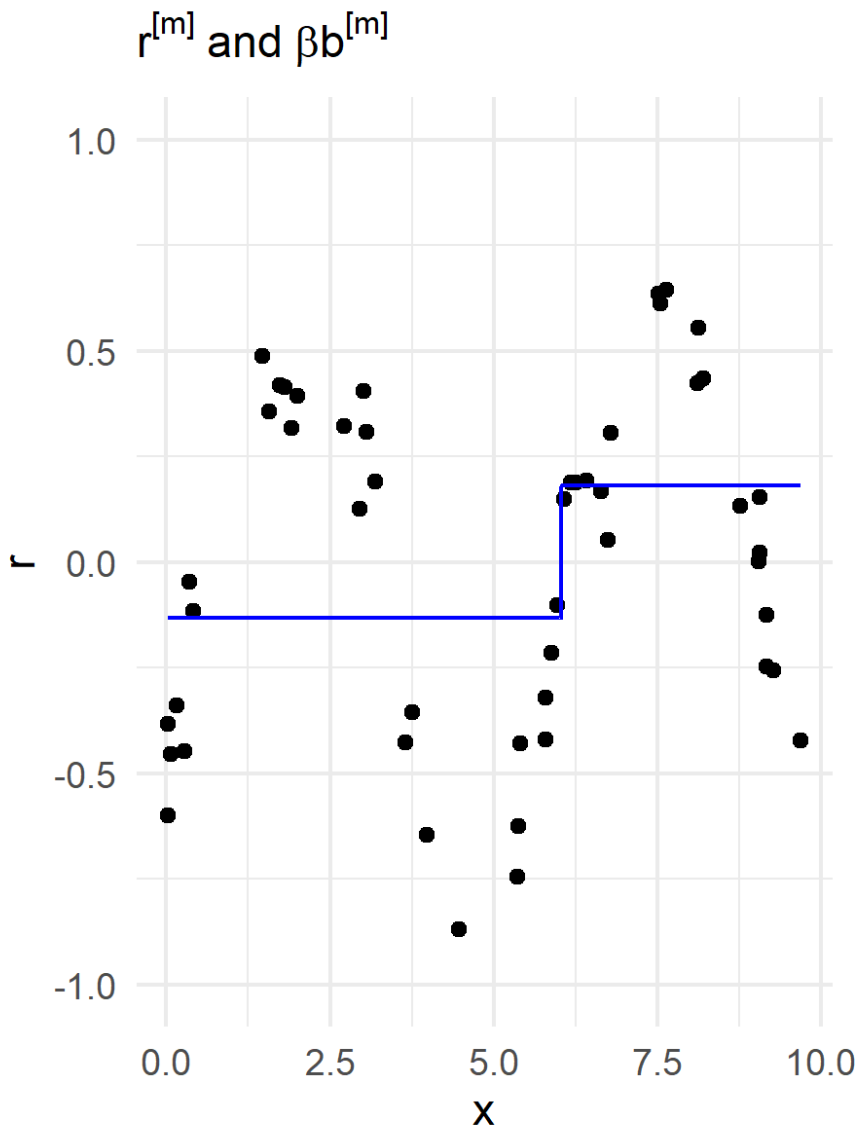




EXAMPLE 1

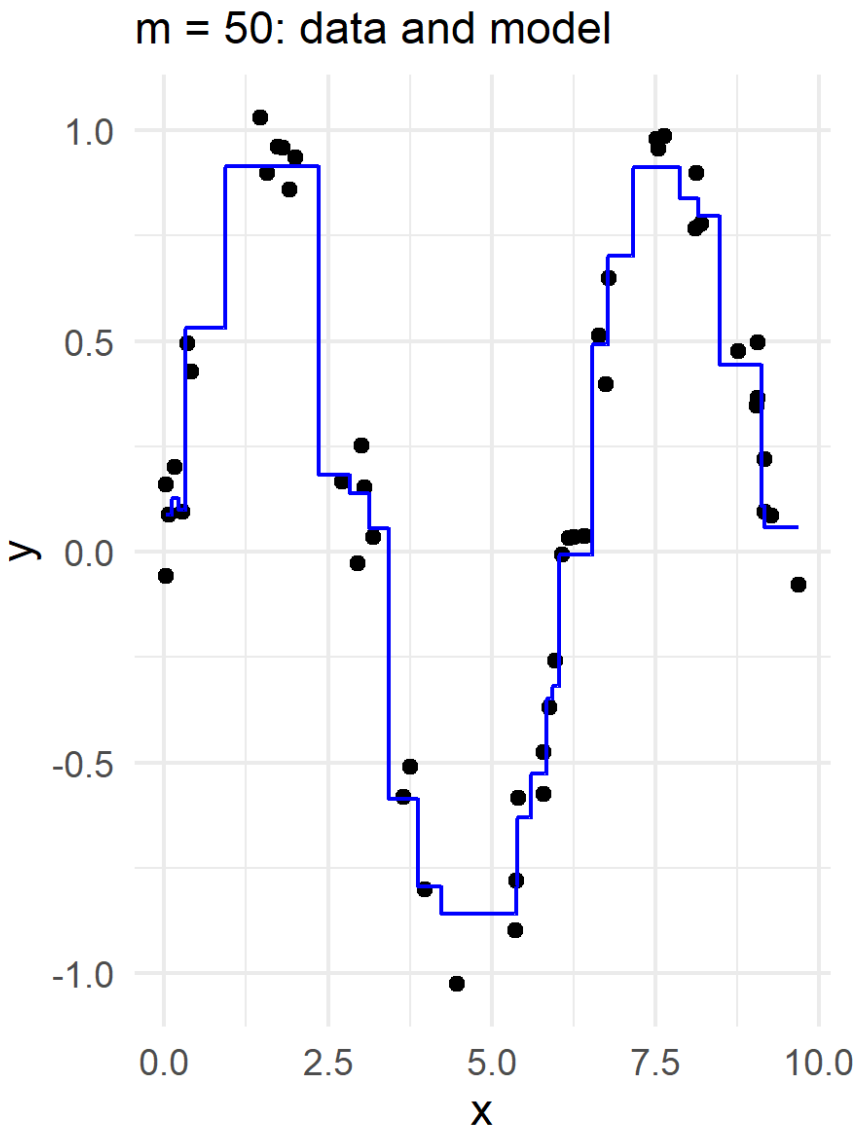
Repeat step (1) to (3):

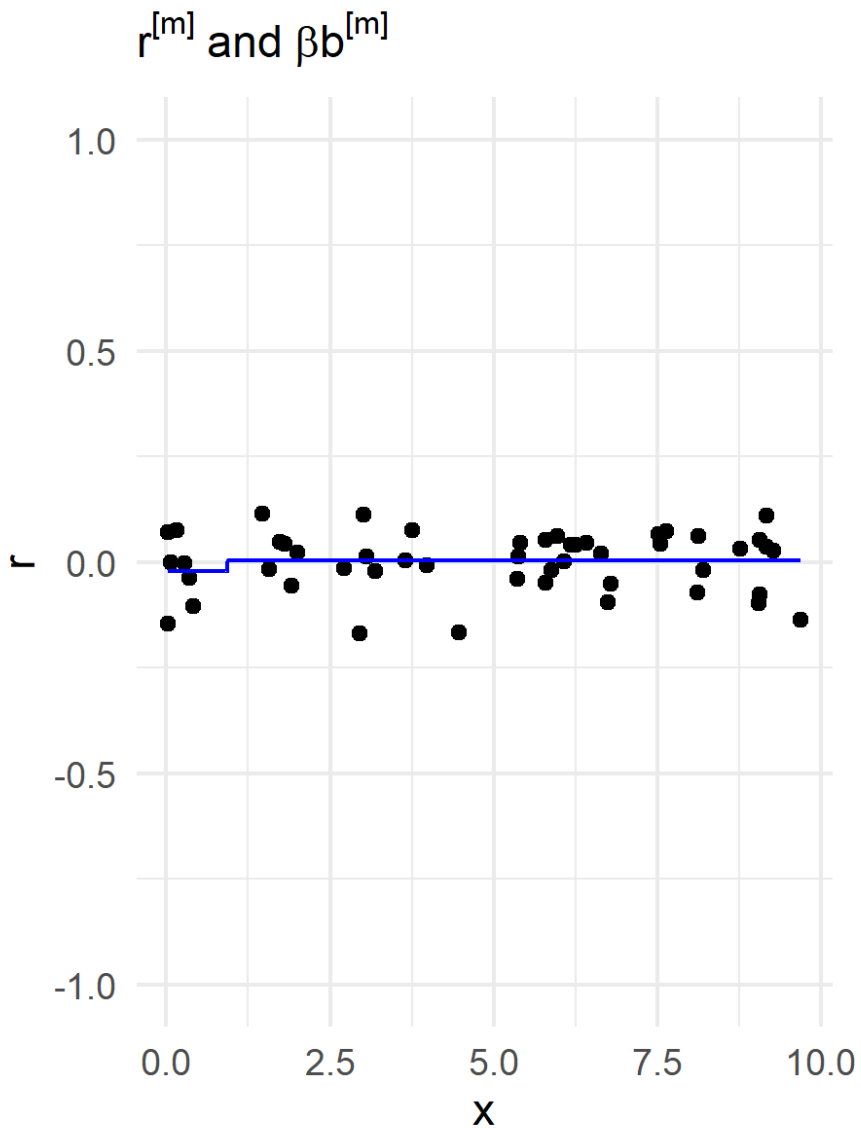




EXAMPLE 1

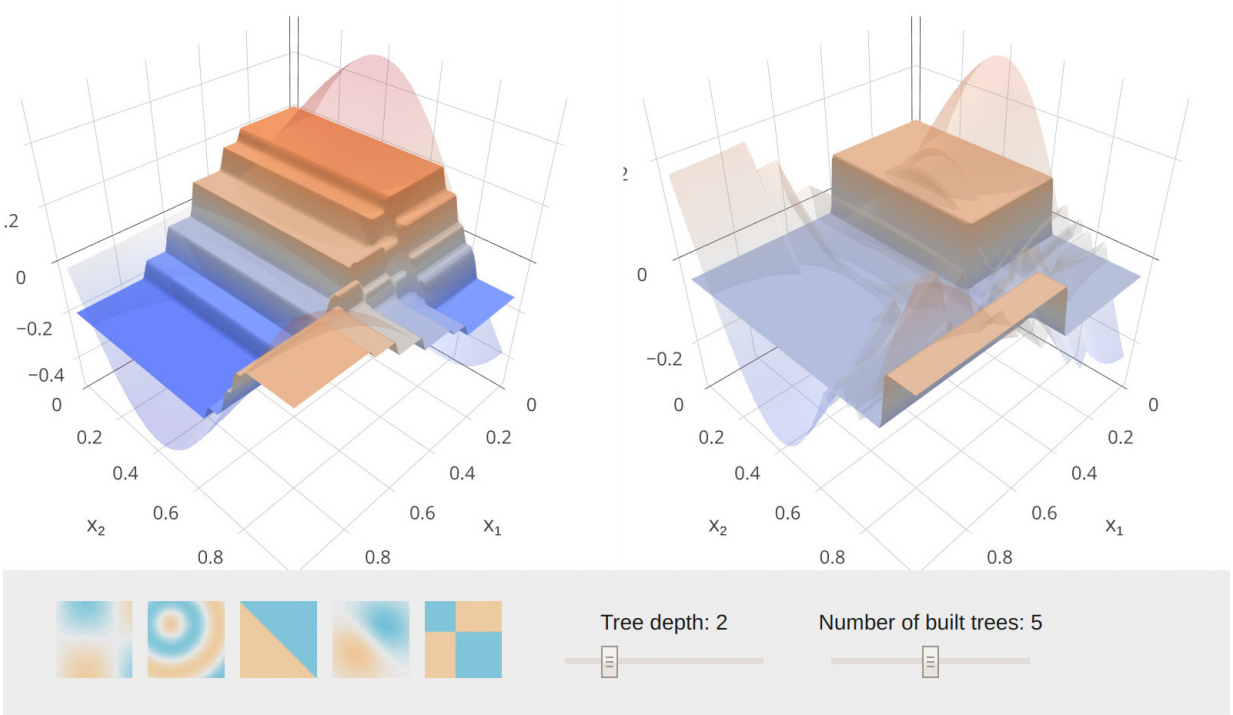
Repeat step (1) to (3):





EXAMPLE 2

This website shows on various 3D examples how tree depth and number of iterations influence the model fit of a GBM with trees.



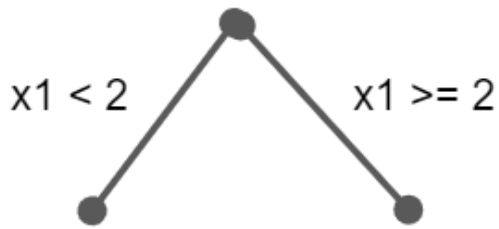
MODEL STRUCTURE AND INTERACTION DEPTH

The model structure of a gradient boosting model with trees is influenced by the chosen tree / interaction depth of $b^{[m]}(\mathbf{x})$.

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

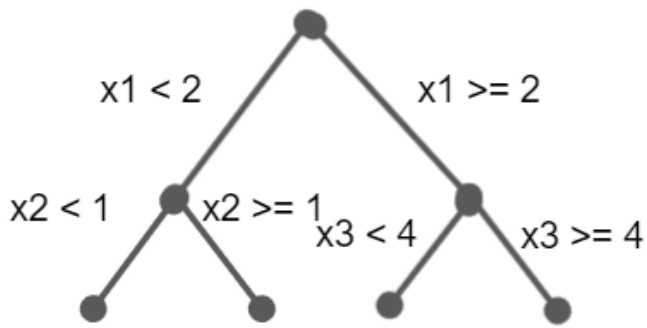
When using stumps (depth = 1), the resulting model is an additive model (GAM) without any interactions:

$$f(\mathbf{x}) = f_0 + \sum_{j=1}^{p} f_j(x_j)$$



When also including trees with a depth of 2, 2-way interactions are included and we get:

$$f(\mathbf{x}) = f_0 + \sum_{j=1}^{p} f_j(x_j) + \sum_{j \neq k} f_{j,k}(x_j, x_k)$$



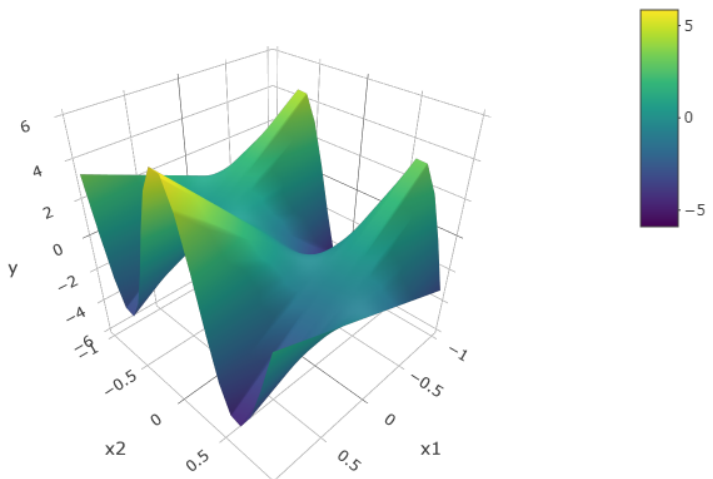
with f_0 being a constant intercept.

MODEL STRUCTURE AND INTERACTION DEPTH

Simulation setting:

- Given: two features x_1 and x_2 and one numeric target variable y of 500 observations.
- x_1 and x_2 are uniformly distributed between -1 and 1.
- Target function: $y^{(i)} = x_1^{(i)} - x_2^{(i)} + 5\cos(5x_2^{(i)}) \cdot x_1^{(i)} + \epsilon^{(i)}$ with $\epsilon^{(i)} \sim \mathcal{N}(0, 1)$, $\forall i \in \{1, \ldots, 500\}$.

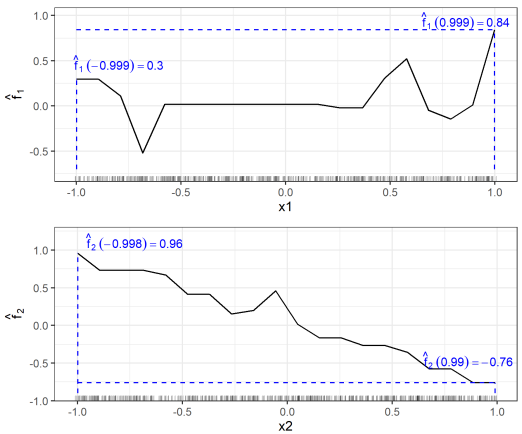
We fit two tree-based GBMs, one with an interaction depth (ID) of 1 (GAM) and one with an interaction depth of 2 (all possible interactions included).

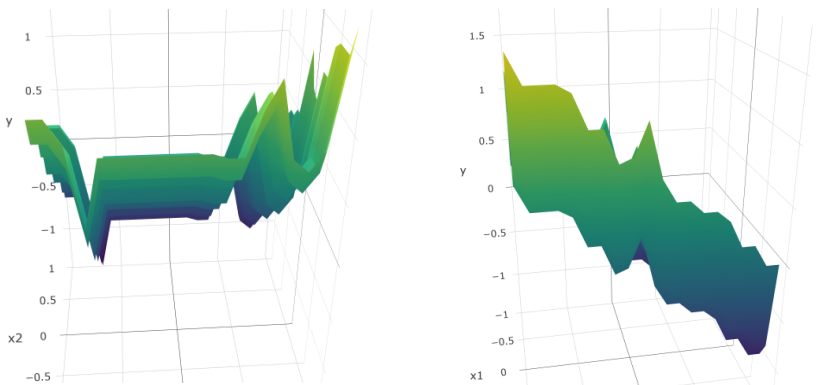


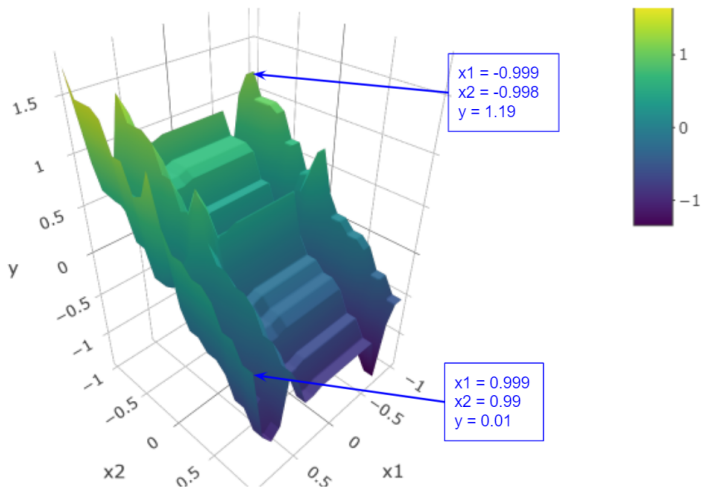
MODEL STRUCTURE AND INTERACTION DEPTH

GBM with interaction depth of 1 (GAM)

No interactions are modelled: Marginal effects of x_1 and x_2 add up to joint effect (plus the constant intercept $\hat{f}_0 = -0.07$).





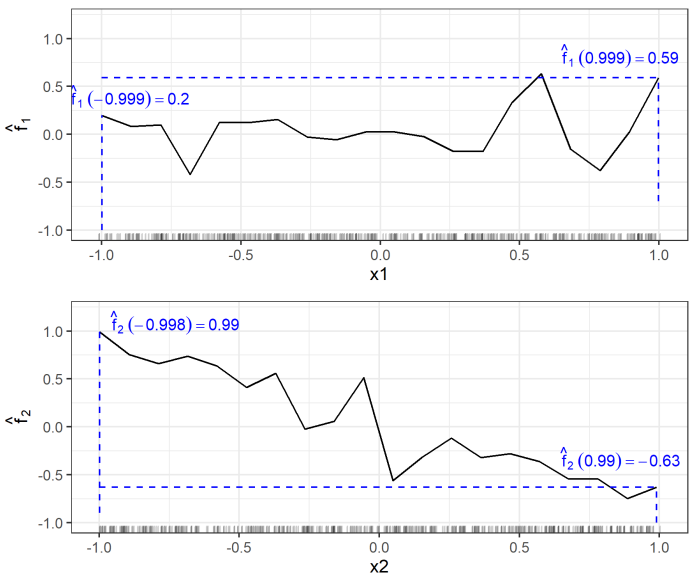


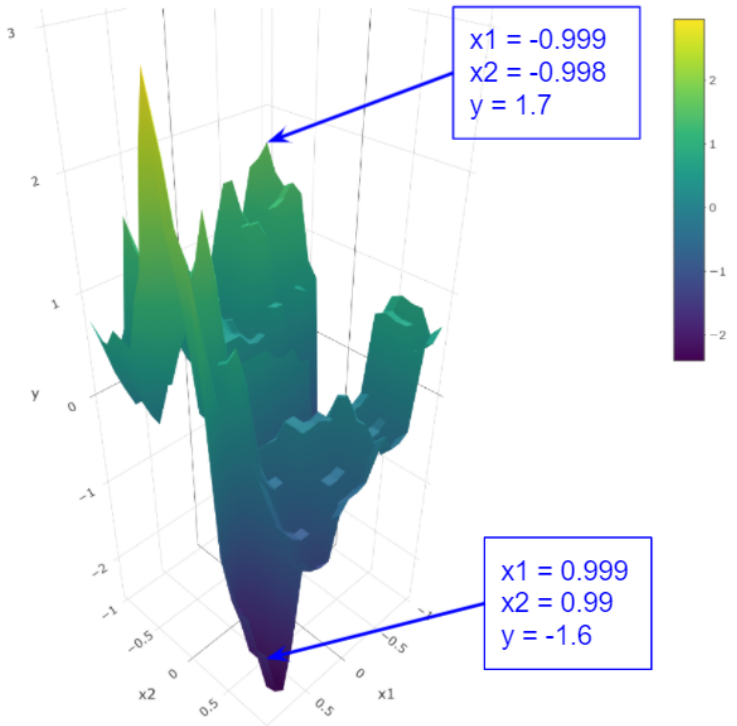
$$\hat{f}(-0.999, -0.998)$$
$$= \hat{f}_0 + \hat{f}_1(-0.999) + \hat{f}_2(-0.998)$$
$$= -0.07 + 0.3 + 0.96 = 1.19$$

MODEL STRUCTURE AND INTERACTION DEPTH

GBM with interaction depth of 2

Interactions between x_1 and x_2 are modelled: Marginal effects of x_1 and x_2 do NOT add up to joint effect due to interaction effects.





THEORETICAL BACKGROUND

One can write a tree as: $b(\mathbf{x}) = \sum_{t=1}^{T} c_t \mathbb{1}_{\{\mathbf{x} \in R_t\}}$, where R_t are the terminal regions and c_t the corresponding constant parameters.

For a fitted tree with regions R_t, the special additive structure can be exploited in boosting:

$$\begin{aligned} f^{[m]}(\mathbf{x}) &= f^{[m-1]}(\mathbf{x}) + \beta^{[m]} b^{[m]}(\mathbf{x}) \\ &= f^{[m-1]}(\mathbf{x}) + \beta^{[m]} \sum_{t=1}^{T^{[m]}} c_t^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_t^{[m]}\}} \\ &= f^{[m-1]}(\mathbf{x}) + \sum_{t=1}^{T^{[m]}} \tilde{c}_t^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_t^{[m]}\}}. \end{aligned}$$

With $\tilde{c}_t^{[m]} = \beta^{[m]} \cdot c_t^{[m]}$ in the case that $\beta^{[m]}$ is a constant learning rate

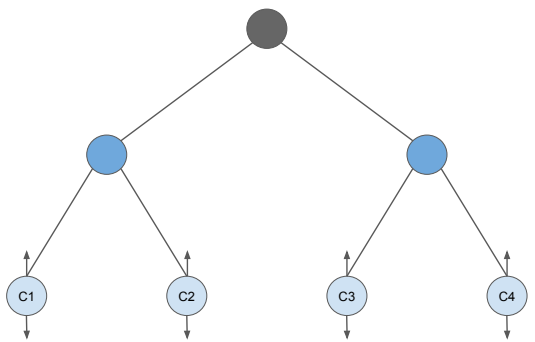
THEORETICAL BACKGROUND

We do the same steps as before: (1) calculate the pseudo-residuals, (2) fit a tree against pseudo-residuals, **but now** we keep only the structure of the tree and optimize the c parameter in a (further) post-hoc step.

$$f^{[m]}(\mathbf{x}) = f^{[m-1]}(\mathbf{x}) + \sum_{t=1}^{T^{[m]}} \tilde{c}_t^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_t^{[m]}\}}.$$

We can determine/change all $\tilde{c}_t^{[m]}$ individually and directly L-optimally:

$$\tilde{c}_t^{[m]} = \arg\min_c \sum_{\mathbf{x}^{(i)} \in R_t^{[m]}} L(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + c).$$



THEORETICAL BACKGROUND

An alternative approach ist to directly fit a loss-optimal tree. The risk function is then defined by:

$$\mathcal{R}(\mathcal{N}') = \sum_{i \in \mathcal{N}'} L(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + c)$$

with $\mathcal{N}'$ being the index set of a specific (left or right) node after splitting and c being a constant value added to the current model for this node. Thus, instead of having a two-step approach of first fitting a tree to the pseudo-residuals of the current model and then finding the optimal value for c, we now directly build a tree that finds c loss-optimally. Since c is unknown, it needs to be determined, which can either be done by a line search or by taking the derivative:

$$\frac{\partial \mathcal{R}(\mathcal{N}')}{\partial c} = \sum_{i \in \mathcal{N}'} \frac{\partial L(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + c)}{\partial f|_{f = f^{[m-1]} + c}} = 0$$

THEORETICAL BACKGROUND

Algorithm Tree Algorithm for Gradient Boosting.

Input: All observations $\mathcal{N}$ and risk function $\mathcal{R}$
Output: $\mathcal{N}_l^{j^*,s^*}$ and $\mathcal{N}_r^{j^*,s^*}$
for $j = x_1 \ldots x_p$ **do**
    for every split s on feature j **do**
        $\mathcal{N}_l^{j,s} = \{i \in \mathcal{N}\}_{j^{(i)} \leq s}$
        $\mathcal{N}_r^{j,s} = \{i \in \mathcal{N}\}_{j^{(i)} > s}$
        Find c which minimizes $\mathcal{R}$ for each node
        $\mathcal{I}(j,s) = \mathcal{R}(\mathcal{N}_l^{j,s}) + \mathcal{R}(\mathcal{N}_r^{j,s})$
    end for
end for
$(j^*, s^*) \in \arg\min_{j,s} \mathcal{I}(j,s)$

The tree algorithm based on the CART algorithm of Breiman shows one partitioning step based on the risk function we introduced before.

GB MULTICLASS WITH TREES

- From Friedman, J. H. - Greedy Function Approximation: A Gradient Boosting Machine (1999)
- Determining the tree structure for each $\hat{b}_k^{[m]}$ by *L2* loss works just like before in the 2-class problem.
- In the estimation of the *c* values, i.e., the heights of the terminal regions, however, all models depend on each other because of the definition of *L*. Optimizing this is more difficult, so we will skip some details and present the main idea and results.

GB MULTICLASS WITH TREES

- The post-hoc, loss-optimal heights of the terminals $\hat{c}_{tk}^{[m]}$ are:

$$\hat{c}_{tk}^{[m]} = - \arg\min_{c_{tk}^{[m]}} \sum_{i=1}^{n} \sum_{k=1}^{g} \mathbb{1}_{\{y=k\}} \ln \pi_k^{[m]}(\mathbf{x}^{(i)}) \,.$$

- Softmax trafo: $\pi_k^{[m]}(\mathbf{x}) = \frac{\exp(f_k^{[m]}(\mathbf{x}))}{\sum_j \exp(f_j^{[m]}(\mathbf{x}))}$, with
- The k-th model: $\hat{f}_k^{[m]}(\mathbf{x}^{(i)})) = \hat{f}_k^{[m-1]}(\mathbf{x}^{(i)}) + \sum_{t=1}^{T_k^{[m]}} \hat{c}_{tk}^{[m]} \mathbb{1}_{\{\mathbf{x}^{(i)} \in R_{tk}^{[m]}\}}$.

GB MULTICLASS WITH TREES

- There is no closed-form solution for finding the optimal $\hat{c}_{tk}^{[m]}$ values. Additionally, the regions corresponding to the different class trees overlap, so that the solution does not reduce to a separate calculation within each region of each tree.
- Hence, we approximate the solution with a single Newton-Raphson step, using a diagonal approximation to the Hessian (we leave out the details here).
- This decomposes the problem into a separate calculation for each terminal node of each tree.
- The result is

$$\hat{c}_{tk}^{[m]} = \frac{g-1}{g} \frac{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} \tilde{r}_k^{[m](i)}}{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} \left| \tilde{r}_k^{[m](i)} \right| \left(1 - \left| \tilde{r}_k^{[m](i)} \right| \right)}.$$

GB MULTICLASS WITH TREES

Algorithm Gradient Boosting for g-class Classification.

Initialize $f_k^{[0]}(\mathbf{x}) = 0,\ k = 1, \ldots, g$
for $m = 1 \to M$ **do**
  Set $\pi_k(\mathbf{x}) = \frac{\exp(f_k^{[m]}(\mathbf{x}))}{\sum_j \exp(f_j^{[m]}(\mathbf{x}))}, k = 1, \ldots, g$
  for $k = 1 \to g$ **do**
    For all i: Compute $\tilde{r}_k^{[m](i)} = \mathbb{1}_{\{y^{(i)}=k\}} - \pi_k(\mathbf{x}^{(i)})$
    Fit regr. tree to the $\tilde{r}_k^{[m](i)}$ giving terminal regions $R_{tk}^{[m]}$
    Compute

$$\hat{c}_{tk}^{[m]} = \frac{g-1}{g} \frac{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} \tilde{r}_k^{[m](i)}}{\sum_{\mathbf{x}^{(i)} \in R_{tk}^{[m]}} \left|\tilde{r}_k^{[m](i)}\right| \left(1 - \left|\tilde{r}_k^{[m](i)}\right|\right)}$$

    Update $\hat{f}_k^{[m]}(\mathbf{x}) = \hat{f}_k^{[m-1]}(\mathbf{x}) + \sum_t \hat{c}_{tk}^{[m]} \mathbb{1}_{\{\mathbf{x} \in R_{tk}^{[m]}\}}$
  end for
end for
Output $\hat{f}_1^{[M]}, \ldots, \hat{f}_g^{[M]}$