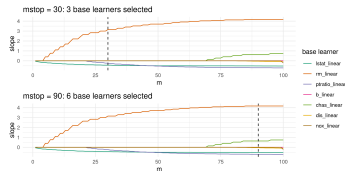
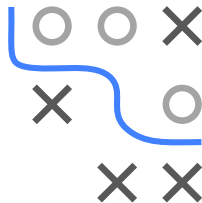


# Introduction to Machine Learning

## Boosting

### Gradient Boosting: CWB Basics 1



#### Learning goals

- Concept of CWB
- Which base learners do we use
- Built-in feature selection

# COMPONENTWISE GRADIENT BOOSTING

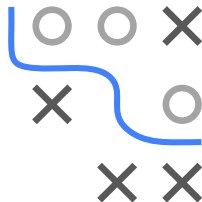
GB (with trees), has strong predictive performance but is difficult to interpret unless the base learners are stumps.

The aim of CWB is to find a model that exhibits:

- strong predictive performance,
- interpretable components,
- automatic selection of components,
- is sparser than a model fitted with maximum-likelihood estimation.

This is achieved by using “nice” base learners which yield familiar statistical models in the end.

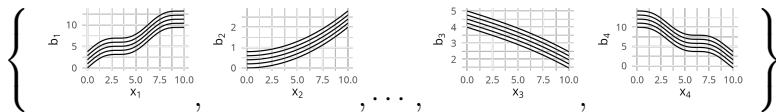
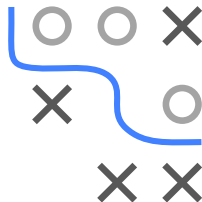
Because of this, CWB is also often referred to as **model-based boosting**.



# BASE LEARNERS

In GB only one kind of base learner  $\mathcal{B}$  is used, e.g., regression trees.

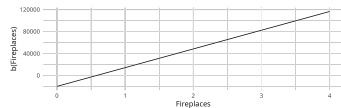
For CWB we generalize this to multiple base learner sets  $\{\mathcal{B}_1, \dots, \mathcal{B}_J\}$  with associated parameter spaces  $\{\Theta_1, \dots, \Theta_J\}$ , where  $j \in \{1, 2, \dots, J\}$  indexes the type of base learner.



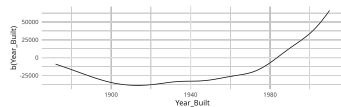
In each iteration, base learners are fitted to the **pseudo residuals**  $\tilde{r}^{[m]}$ .

# BASE LEARNERS / 2

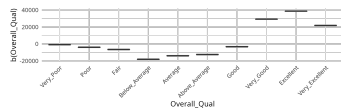
We restrict the base learners to additive model components, i.e.,



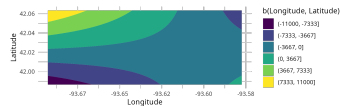
linear effect



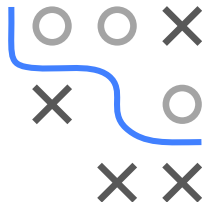
non-linear (spline) effect



dummy encoded linear model of a cat. feature



tensor product spline for interaction modelling (e.g. spatial effects)

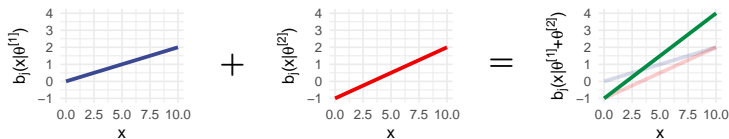


More advanced base learners could also be Markov random fields, random effects, or trees.

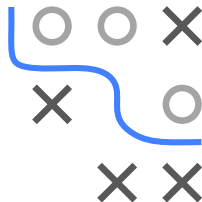
# BASE LEARNERS / 3

Two BLs of the same type can simply be added by adding up their parameter vectors:

$$b_j(\mathbf{x}, \theta^{[1]}) + b_j(\mathbf{x}, \theta^{[2]}) = b_j(\mathbf{x}, \theta^{[1]} + \theta^{[2]}).$$



Thus, if  $\{b_j(\mathbf{x}, \theta^{[1]}), b_j(\mathbf{x}, \theta^{[2]})\} \in \mathcal{B}_j$ , then  $b_j(\mathbf{x}, \theta^{[1]} + \theta^{[2]}) \in \mathcal{B}_j$ .



# COMPONENTWISE BOOSTING ALGORITHM

Different from GB, multiple base learners  $b_j \in \mathcal{B}_j, j = 1, \dots, J$ , are fitted and only best-fitting one is selected and updated.

- 1: Initialize  $f^{[0]}(\mathbf{x}) = \arg \min_{\theta_0 \in \mathbb{R}} \sum_{i=1}^n L(y^{(i)}, \theta_0)$
- 2: **for**  $m = 1 \rightarrow M$  **do**
- 3:     For all  $i$ :  $\tilde{r}^{[m](i)} = - \left[ \frac{\partial L(y, f)}{\partial f} \right]_{f=f^{[m-1]}(\mathbf{x}^{(i)}), y=y^{(i)}}$
- 4:     **for**  $j = 1 \rightarrow J$  **do**
- 5:         Fit regression base learner  $b_j \in \mathcal{B}_j$  to the vector of pseudo-residuals  $\tilde{r}^{[m]}$ :
- 6:          $\hat{\theta}_j^{[m]} = \arg \min_{\theta \in \Theta_j} \sum_{i=1}^n (\tilde{r}^{[m](i)} - b_j(\mathbf{x}^{(i)}, \theta))^2$
- 7:     **end for**
- 8:      $j^{[m]} = \arg \min_j \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\theta}_j^{[m]}))^2$
- 9:     Update  $f^{[m]}(\mathbf{x}) = f^{[m-1]}(\mathbf{x}) + \alpha \hat{b}_{j^{[m]}}(\mathbf{x}, \hat{\theta}_{j^{[m]}}^{[m]})$
- 10: **end for**
- 11: Output  $\hat{f}(\mathbf{x}) = f^{[M]}(\mathbf{x})$

# COMPONENTWISE BOOSTING ALGORITHM

---

**Algorithm** Componentwise Gradient Boosting (inner loop).

---

4: **for**  $j = 1 \rightarrow J$  **do**

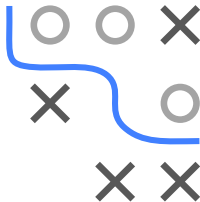
5:   Fit regression base learner  $b_j \in \mathcal{B}_j$  to the vector of pseudo-residuals  $\tilde{r}^{[m]}$ :

6:    $\hat{\theta}_j^{[m]} = \arg \min_{\theta \in \Theta_j} \sum_{i=1}^n (\tilde{r}^{[m](i)} - b_j(\mathbf{x}^{(i)}, \theta))^2$

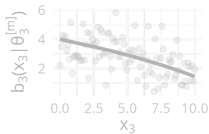
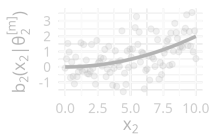
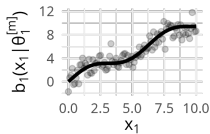
7: **end for**

8:  $j^{[m]} = \arg \min_j \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\theta}_j^{[m]}))^2$

---



Iteration  $m, j = 1, \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_1(x_1^{(i)}, \hat{\theta}_1^{[m]}))^2 = 24.4$ :



# COMPONENTWISE BOOSTING ALGORITHM

---

**Algorithm** Componentwise Gradient Boosting (inner loop).

---

4: **for**  $j = 1 \rightarrow J$  **do**

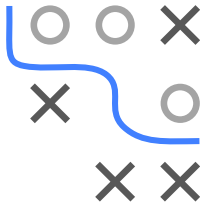
5:   Fit regression base learner  $b_j \in \mathcal{B}_j$  to the vector of pseudo-residuals  $\tilde{r}^{[m]}$ :

6:    $\hat{\theta}_j^{[m]} = \arg \min_{\theta \in \Theta_j} \sum_{i=1}^n (\tilde{r}^{[m](i)} - b_j(\mathbf{x}^{(i)}, \theta))^2$

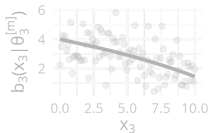
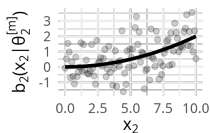
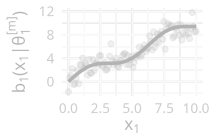
7: **end for**

8:  $j^{[m]} = \arg \min_j \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\theta}_j^{[m]}))^2$

---



Iteration  $m, j = 2, \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_2(x_2^{(i)}, \hat{\theta}_2^{[m]}))^2 = 43.2$ :





A 3x3 grid with a blue path starting at the top-left cell (0,0) and ending at the bottom-right cell (2,2). The path consists of the following cells: (0,0), (0,1), (1,1), (1,2), and (2,2). The cells (0,2), (1,0), and (2,0) are empty. The cells (0,1), (1,1), and (1,2) contain a grey 'X'. The cells (1,0) and (2,0) contain a grey circle.

$$8: j^{[m]} = \arg \min_j \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\theta}_j^{[m]}))^2$$

A scatter plot showing the relationship between  $x_3$  (x-axis, ranging from 0.0 to 10.0) and  $b_3(x_3 | \Theta_3^{ml})$  (y-axis, ranging from 2 to 6). The data points are represented by grey circles, and a solid black line indicates a negative linear regression fit.

# COMPONENTWISE BOOSTING ALGORITHM

---

**Algorithm** Componentwise Gradient Boosting (inner loop).

---

4: **for**  $j = 1 \rightarrow J$  **do**

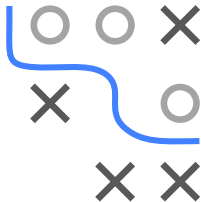
5:   Fit regression base learner  $b_j \in \mathcal{B}_j$  to the vector of pseudo-residuals  $\tilde{r}^{[m]}$ :

6:    $\hat{\theta}_j^{[m]} = \arg \min_{\theta \in \Theta_j} \sum_{i=1}^n (\tilde{r}^{[m](i)} - b_j(\mathbf{x}^{(i)}, \theta))^2$

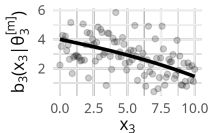
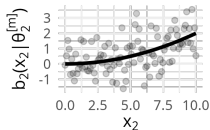
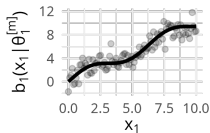
7: **end for**

8:  $j^{[m]} = \arg \min_j \sum_{i=1}^n (\tilde{r}^{[m](i)} - \hat{b}_j(\mathbf{x}^{(i)}, \hat{\theta}_j^{[m]}))^2$

---



Iteration  $m$ :  $\Rightarrow j^{[m]} = 1$



## FEATURE SELECTION IN CWB

In CWB, we often define BLs on a single feature

$$b_j(x_j, \theta) \quad \text{for } j = 1, 2, \dots, p.$$

Allows natural form of feature selection:

- When we select the best BL in one iter of CWB, we thereby also only select one (associated) feature
- Note that a feature (or rather a BL associated with it) can be selected in multiple iters, so  $\leq M$  features are selected

