12 - Boosting

Data Mining

SYS 6018 | Fall 2019

12-boosting.pdf

Contents

1 Boosting

Boosting is a *sequential* ensemble method.

Boosting Sketch

- There are two main versions of boosting:
  - *Gradient Boosting*: fits the next model in the sequence $\hat{g}_k(x)$ to the (pseudo) residuals calculated from the predictions on the previous models $\sum_{l=0}^{k-1} \hat{w}_l \hat{g}_l(x)$.
  - *AdaBoost*: fits the next model to sequentially *weighted* observations. The weights are proportional to the how poorly the current models predict the observation.
- Boosting is primarily a *bias* reducer
  - The base models are often simple/weak (low variance, but high bias) models (like shallow trees)

2 AdaBoost

AdaBoost was motivated by the idea that many *weak* leaners can be combined to produce a *strong* aggregate model.

- AdaBoost is for binary classification problems
- Trees are a popular base learner
  - *Weak* learners are usually used. For trees, this means shallow depth.
- At each iteration, the current model is evaluated.
  - The *ensemble weight* of model m is based on its performance (on all the training data)
  - The *observation weight* of observation i is increased if it is mis-classified and decreased if it is correctly classified.

  - Thus, at each iteration, those observations that are mis-classified are weighted higher and get extra attention in the next iteration.

- Because Adaboost uses hard-classifiers, it is sensitive to unbalanced data and unequal misclassification costs.
  - Because the thresholds are set to $p > .50$
  - There are, of course, ways to account for unbalance and unequal costs in the algorithm
  - An improvement to AdaBoost, *LogitBoost* explicitly attempts to estimate the class probability during each iteration which will allow easier post-fitting adjustments for unequal costs

2.1 Adaboost Algorithm

Algorithm: AdaBoost

Inputs:

- $D = \{(x_i, y_i\}_{i=1}^n$, where $y_i \in \{-1, 1\}$
- Tuning parameters for base model $\hat{g}$

Algorithm:

1. Initialize *observation weights* $w_i = 1/n$ for all i
2. For $k = 1$ to M:
   a. Fit a *classifier* $\hat{g}_k(x)$ that maps (x_i, w_i) to $\{-1, 1\}$. In other words, the classifier must make a hard classification using weighted observations.

   b. Compute the weighted mis-classification rate

   $$e_k = \frac{\sum_{i=1}^n w_i \, \mathbb{1}(y_i \neq \hat{g}_k(x_i))}{\sum_{i=1}^n w_i}$$

   c. Calculate the *coefficient* for model k (*ensemble weight*)

   $$a_k = \log\left(\frac{1 - e_k}{e_k}\right)$$

   d. Update the *observations weights*. Increase weights for observations that are mis-classified by model $\hat{g}_k$ and decrease weights for the correctly classified observations.

   $$\tilde{w}_i = w_i \cdot \exp\left(a_k \cdot \mathbb{1}(y_i \neq \hat{g}_k(x_i))\right)$$

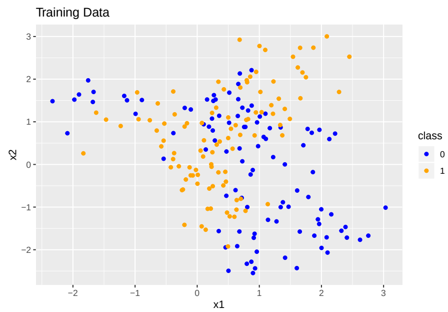
   $$w_i = \frac{\tilde{w}_i}{\sum_{j=1}^n \tilde{w}_j} \qquad \text{(\textit{re-normalize weights})}$$

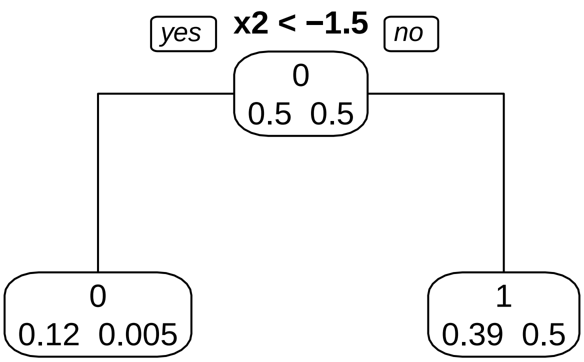
3. Output final ensemble $\hat{f}_M(x) \in [-1, 1]$

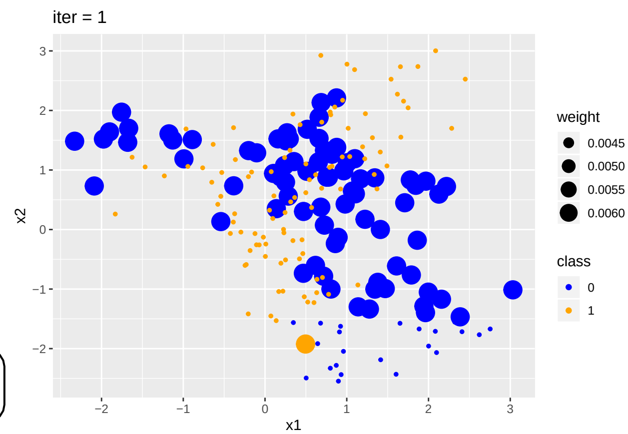
   $$\hat{f}_M(x) = \sum_{k=1}^M a_k \hat{g}_k(x)$$

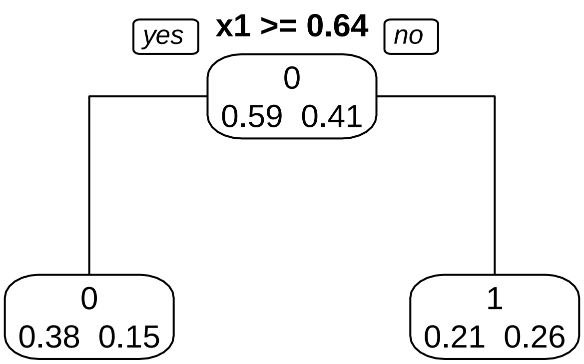
   - Or remap to a probability $\hat{p}(x) = \frac{e^{2f}}{1+e^{2f}}$

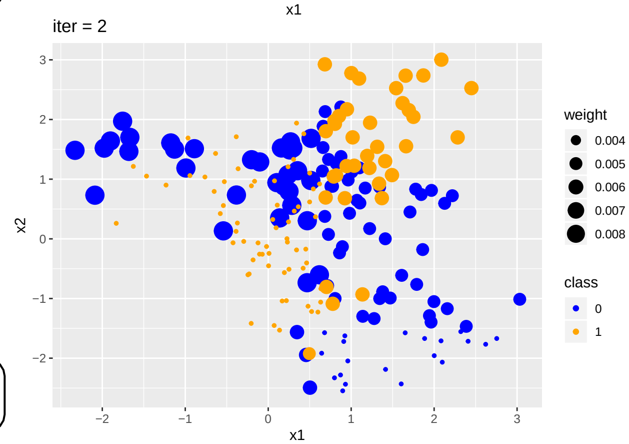
2.1.1 Illustration with Stumps (depth = 1, n.nodes=2)

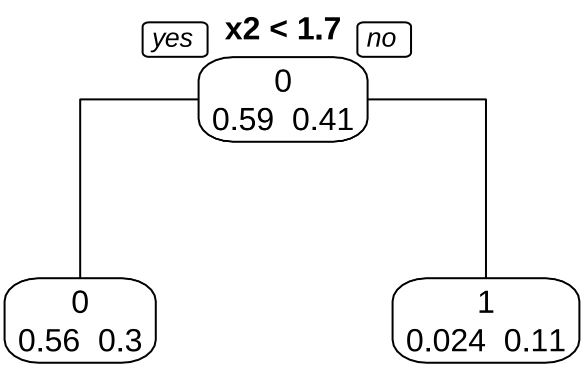


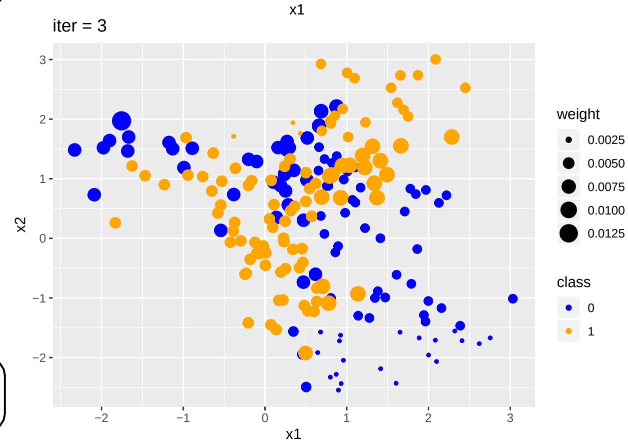




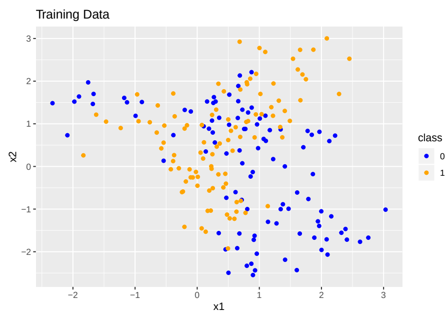


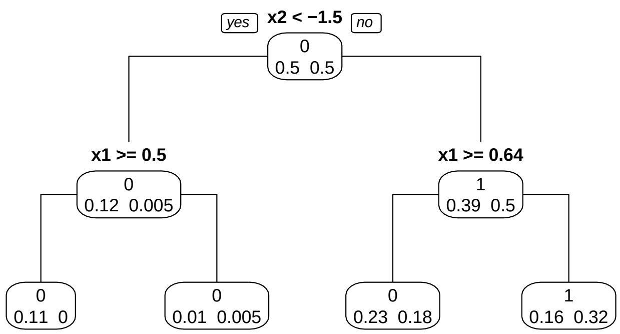


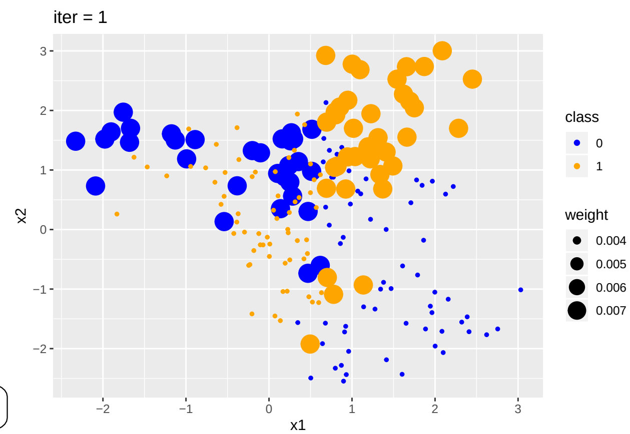


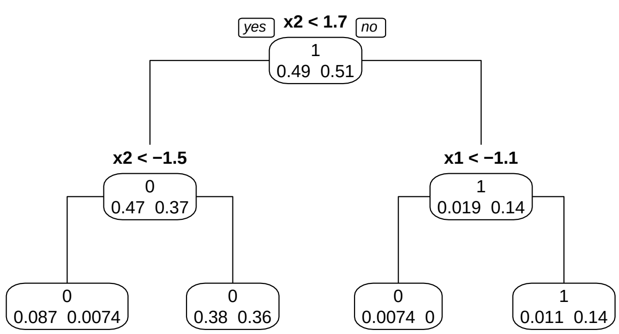


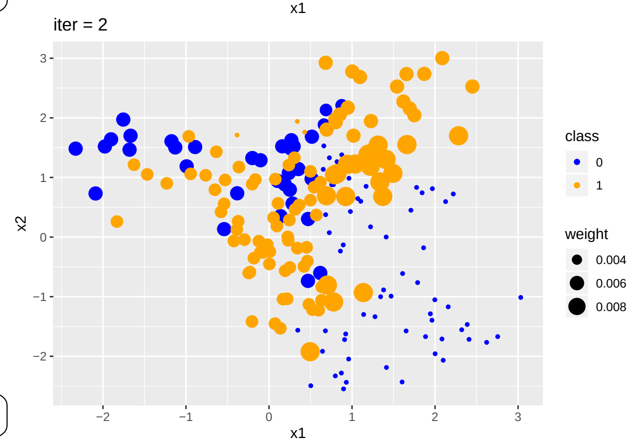
2.1.2 Illustration with depth = 2, n.nodes=4

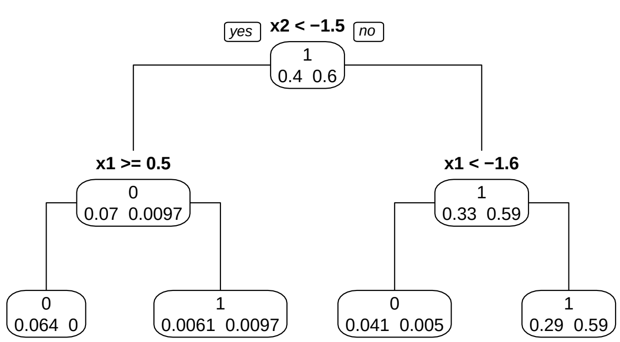


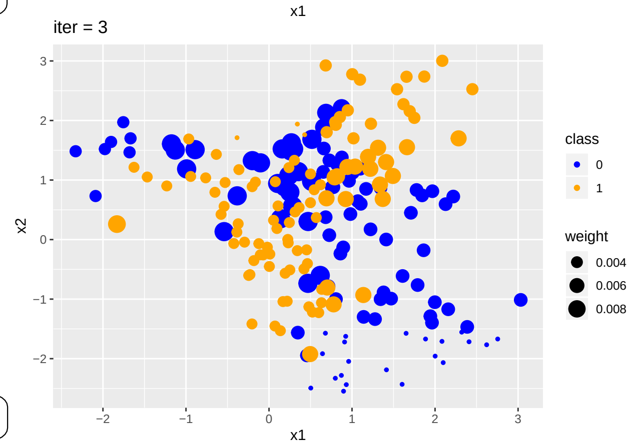


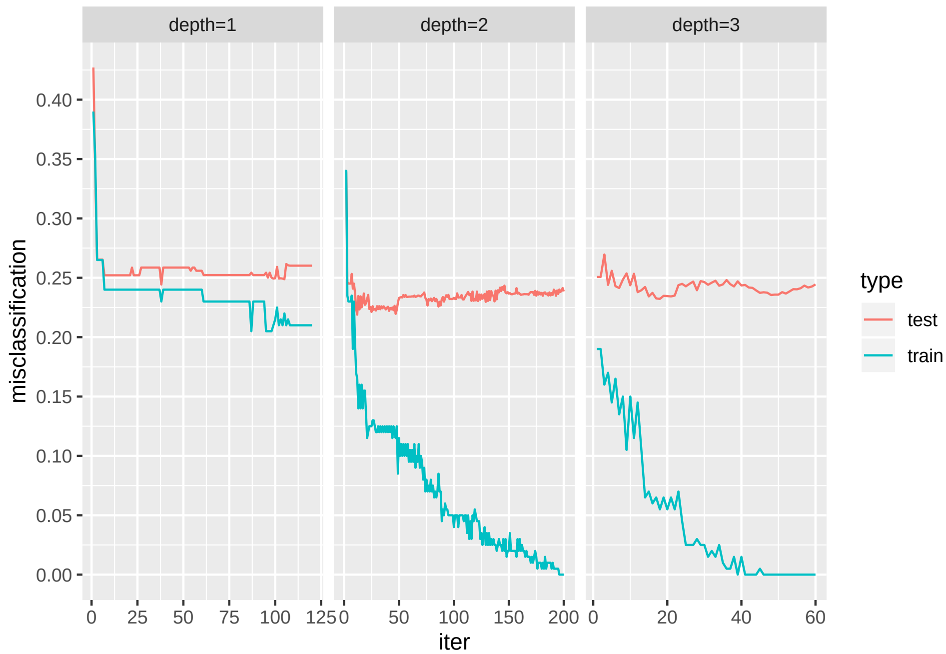








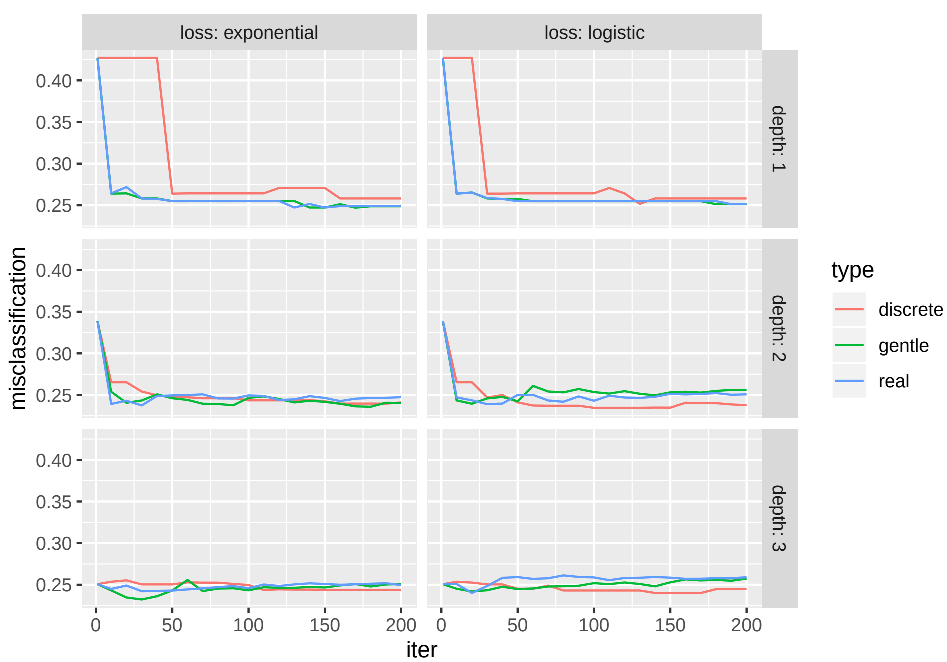




2.2 R package `ada`

The R package `ada` provides an implementation of AdaBoost (and related methods).

- See Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: A statistical view of boosting. Annals of Statistics, 28(2), 337-374. for the details of model variations
  - {Discrete, Real, Gentle} AdaBoost
  - Logitboost



3 Gradient Boosting

In gradient boosting, instead of re-weighting the observations, each new model is fit to the functional gradients (i.e., a type of residuals)

- Gradient Boosting can fit to a variety of loss functions by simply changing how the *residuals* are calculated.
- Again, trees are often used as the base learner
  - For gradient boosting, *regression* trees are used

3.1 L_2 Boosting

L_2 boosting is based on the the squared error loss function

$$L(y_i, \hat{f}(x_i)) = \frac{1}{2}(y_i - \hat{f}(x_i))^2$$

- The *residuals* are

$$\begin{aligned} r_i &= \left[\frac{\partial L(y_i, f_i)}{\partial f_i}\right]_{f_i = \hat{f}(x_i)} \\ &= y_i - \hat{f}(x_i) \end{aligned}$$

- This is basically just re-fitting to the residuals.

Algorithm: L_2 Boosting

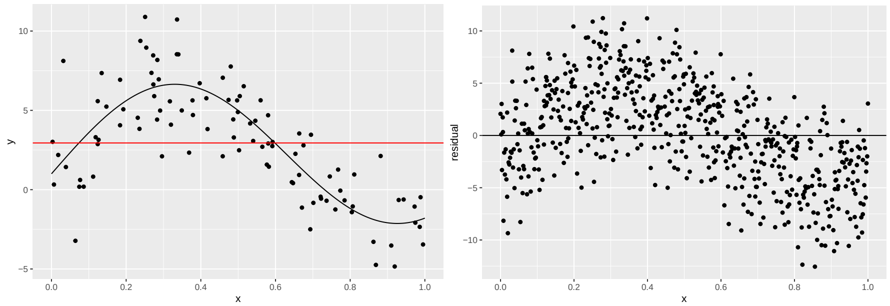
1. Initialize $\hat{f}_0(x) = \bar{y}$
2. For $k = 1$ to M:
   a. Calculate residuals $r_i = y_i - \hat{f}_{m-1}(x_i)$ for all i

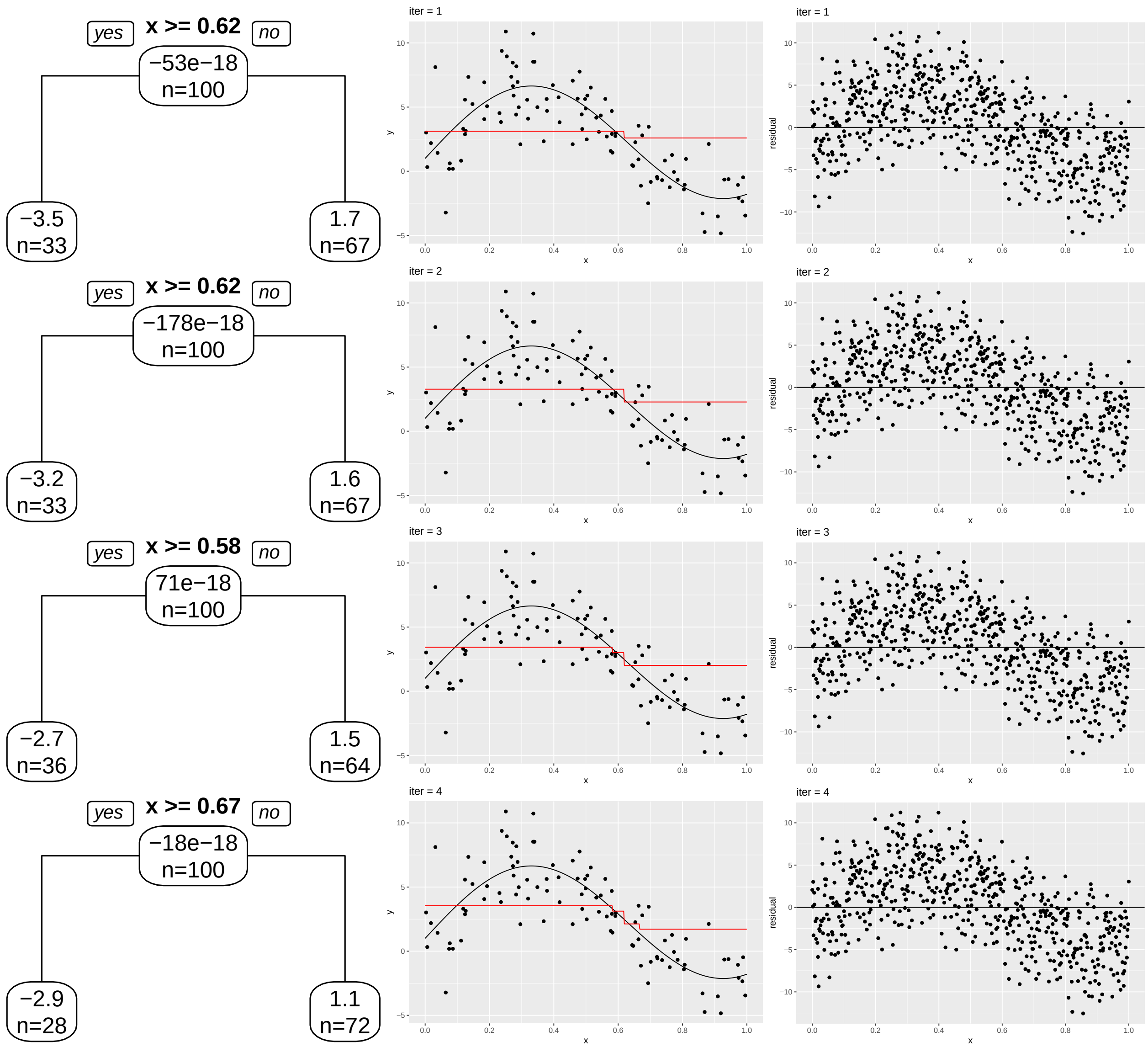
   b. Fit a base learner (e.g., regression trees) to the residuals $\{(x_i, r_i)\}_{i=1}^n$ to get the model $\hat{g}_m(x)$

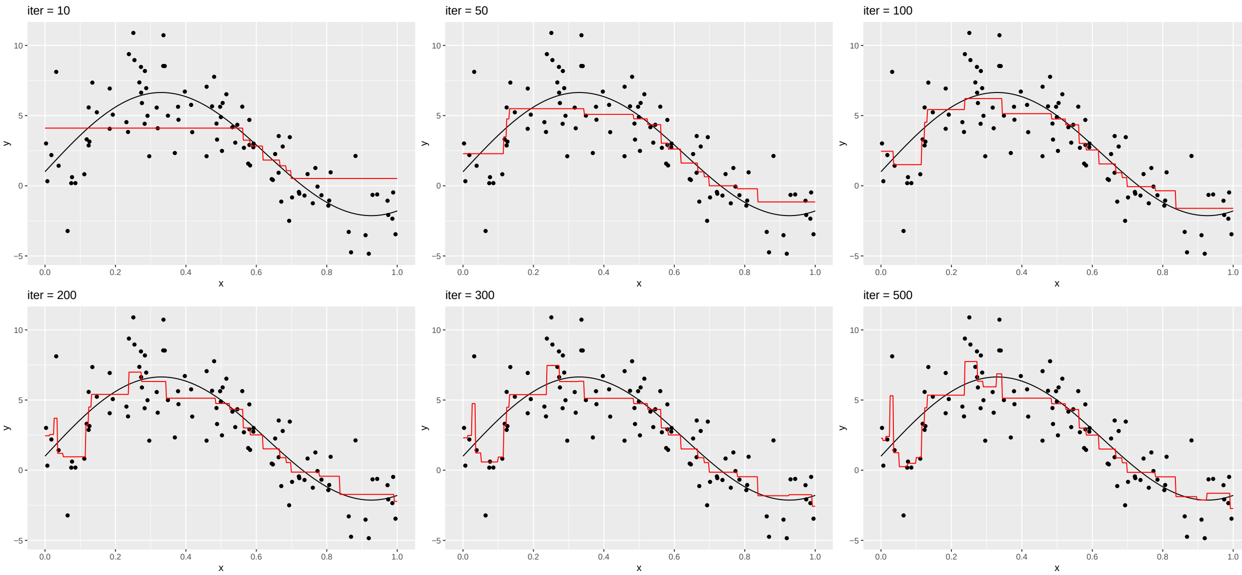
   c. Update the overall model $\hat{f}_m(x) = f_{m-1}(x) + \nu\hat{g}_m(x)$
      - $0 \le \nu \le 1$ is the step-size (*shrinkage*)
3. Final model is $\hat{f}_M(x) = \bar{y} + \sum_{k=1}^M \nu\hat{g}_k(x)$

- Like AdaBoost, emphasis is given to observations that are predicted poorly (large residuals)

3.1.1 Illustration using stumps (depth=1, n.nodes=2)







3.2 GBM (Gradient Boosting Machine)

- R package `gbm`
- GBM Documentation
- Model/Tree Tuning Parameters:

- Boosting Tuning Parameters:

3.3 xgboost (Extreme Gradient Boosting)

- R package `xgboost`
- xgboost Documentation
- Model/Tree Tuning Parameters:

- Boosting Tuning Parameters:

3.4 CatBoost

- R package: (https://github.com/catboost/catboost/tree/master/catboost/R-package)
- CatBoost Documentation
- Model/Tree Tuning Parameters:

- Boosting Tuning Parameters:

3.5 LightGBM

- R Package: https://github.com/microsoft/LightGBM/tree/master/R-package
- LightGBM Documentation
- Model/Tree Tuning Parameters:

- Boosting Tuning Parameters: