Boosting

MATH 6312
Department of mathematics, UTA

ESL 10.1, 10.1.1, 10.2, 10.10.2,10.10.3,10.11, 10.13, 10.13.1

Optional reading: ESL 10.3, 10.4, 10.5, 10.7, 10.8

Boosting

- The motivation for boosting is a procedure that combines the outputs of many weak learners to produce a powerful committee.

- It is designed for classification, but it can be extended to regression as well.

- It is similar to bagging in that we combine the results of several base models. However, boosting does something fundamentally different, and can work a lot better.

- It could boost the performance (on the training set) of any weak (base) learner arbitrarily high, if the weak learner could always perform slightly better than random guessing.

- Boosting can can be applied to any weak learners, but here we mainly discuss boosting for (classification) trees.

- One of the key differences between boosting and bagging is how the individual classifiers are fit.

- Unlike in bagging, in boosting we fit the tree to the entire training set, but adaptively weight the observations to encourage better predictions for points that were previously misclassified.

  - Trees are grown sequentially: each tree is grown using information from previously grown trees.

Weighted observations

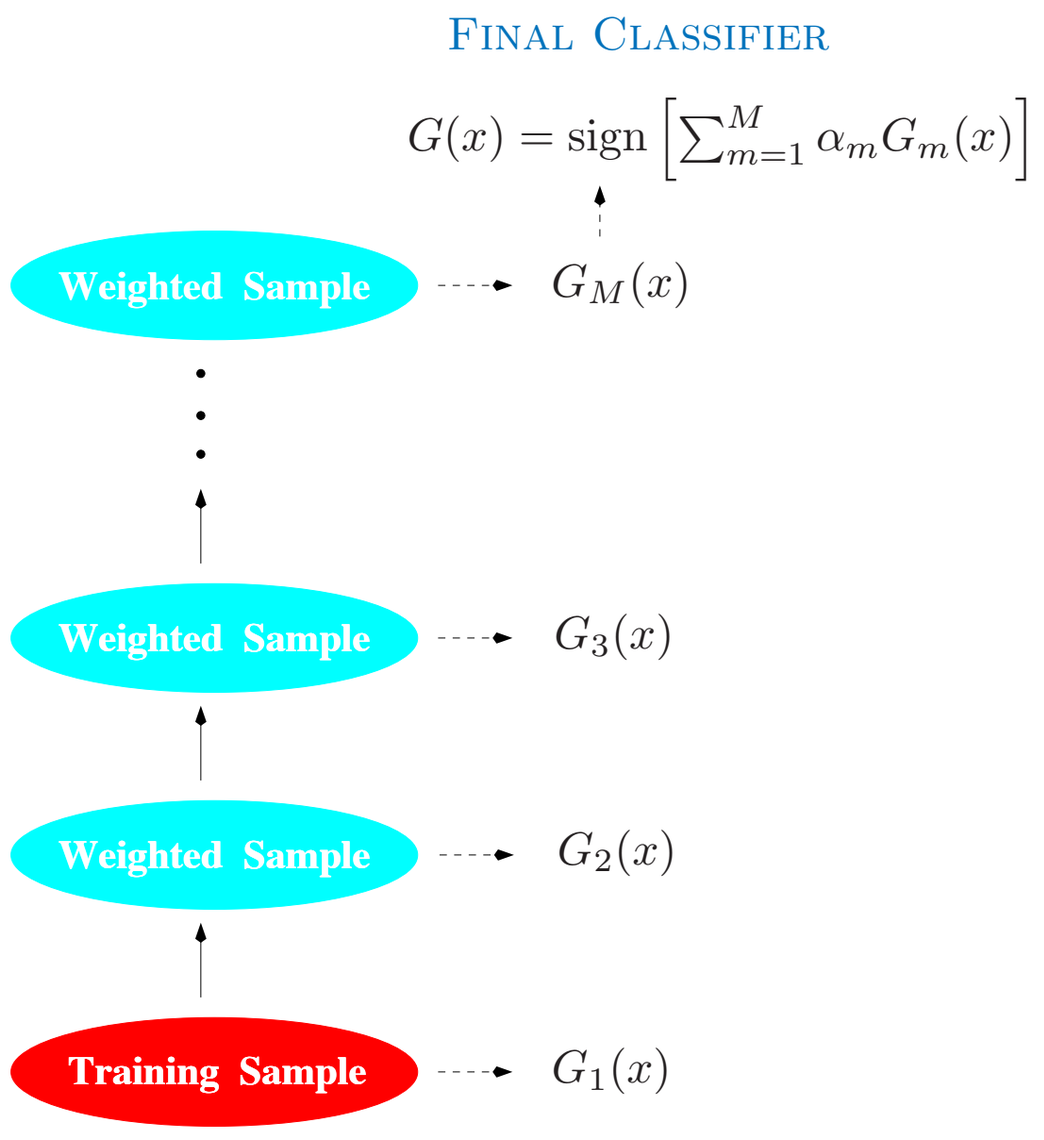
- Consider weight w_i for each observation (y_i, x_i) where $x_i \in \mathbb{R}^p$. A higher weight means that this observation has a higher importance to be correctly classified.
- The classification tree can be easily adapted to use the weights. The weighted proportion of class 1 is

$$\hat{p}(R_j) = \frac{\sum_{i \in R_j} w_i \cdot \mathbb{I}(y_i = 1)}{\sum_{i \in R_j} w_i}$$

- Then, we classify the rectangle R_j as class 1 if the weighted proportion is > 0.5

Schematic of AdaBoost

- Outcome is coded as y ∈{-1,1}
- We sequentially apply the weak classification method on adaptively reweighed samples.
- $G_m(x)$'s are weak classifiers, and G(x) is a weighted average of these.
- Higher weight α_m for more accurate $G_m(x)$.



❖ AdaBoost.M1 with exponential loss

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute

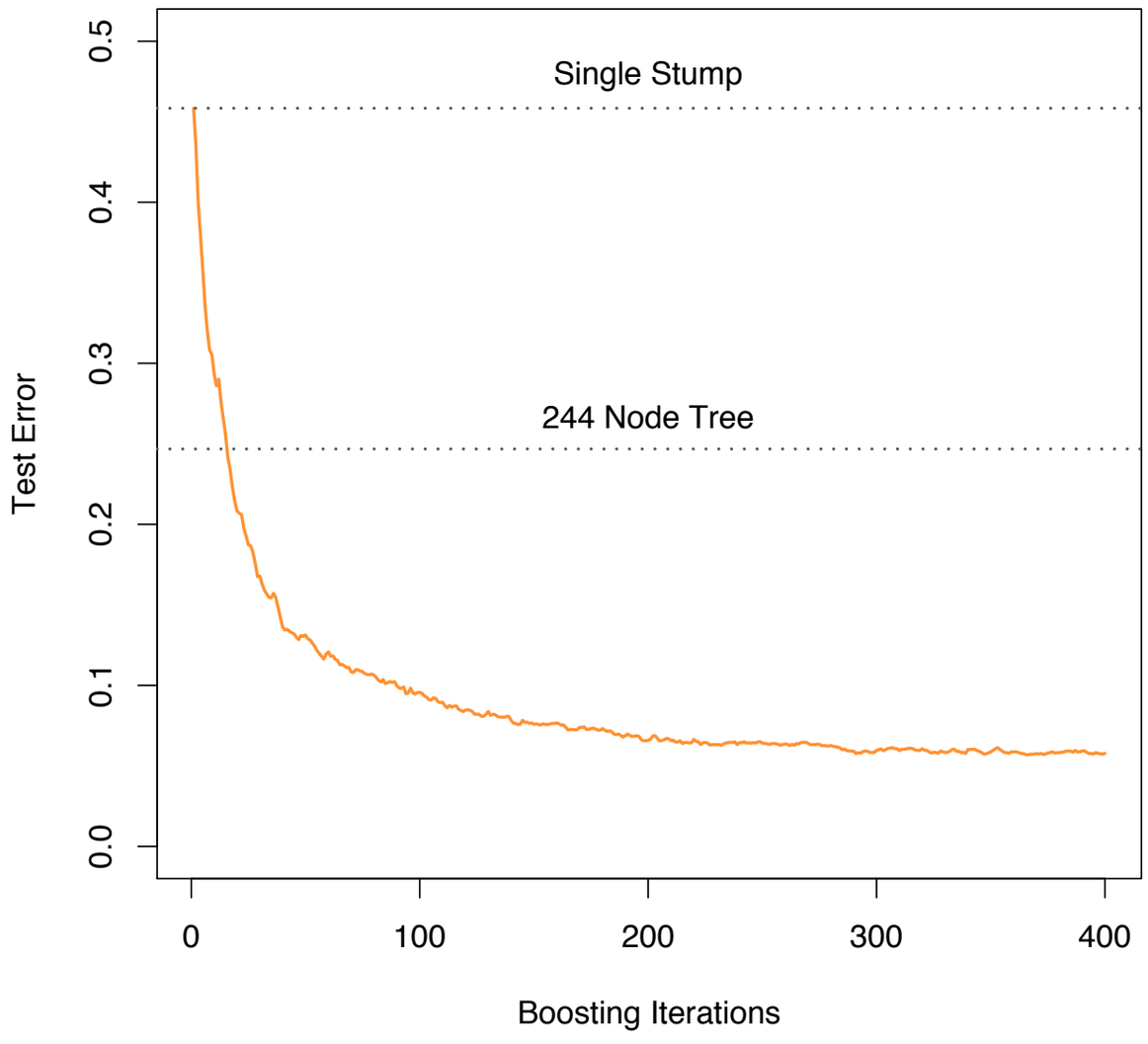
weighted training error: $\text{err}_m = \dfrac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$.

   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Example: AdaBoost for decision stump



ESL 10.2: simulated data

- AdaBoost increase the performance of even a very weak classifier.
- Feature variables $X_1,\ldots,X_{10}$ are simulated (N=2,000) from standard Gaussian. Outcome Y is defined by

$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5), \\ -1 & \text{otherwise.} \end{cases}$$

- Single stump (45.8%) performs slightly better than random guessing. AdaBoost at 400 iterations has 5.8% test error.

Tuning parameters for boosting

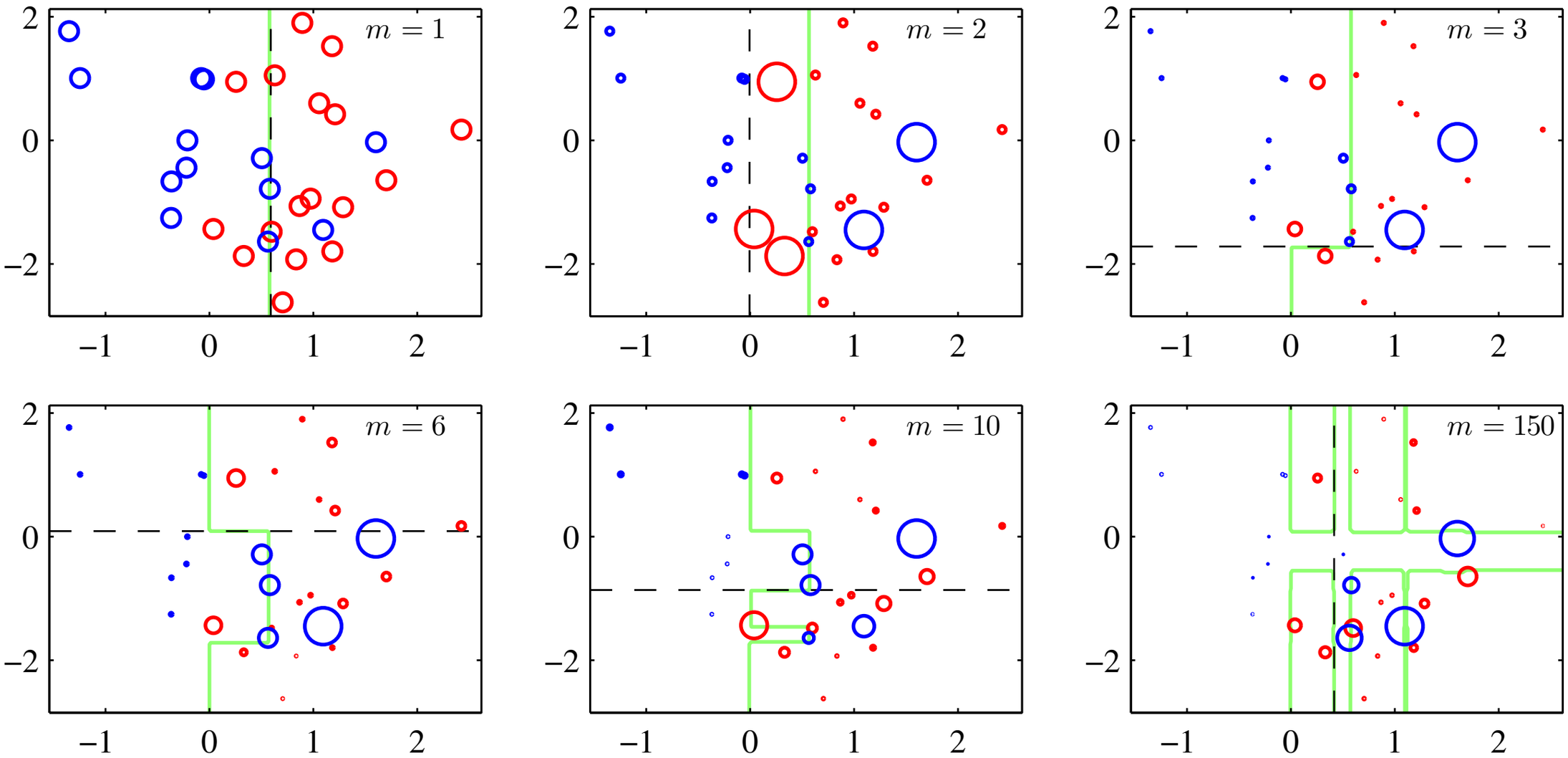
- Boosting iteration M: unlike bagging and random forests, boosting can overfit if M is too large, although this overfitting tends to occur slowly if at all. We use cross validation to select M.

- Number of splits d in each tree, which controls the complexity of the boosted ensemble.

  - Often d = 1 (single stump) works well.

  - More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Toy example: AdaBoost

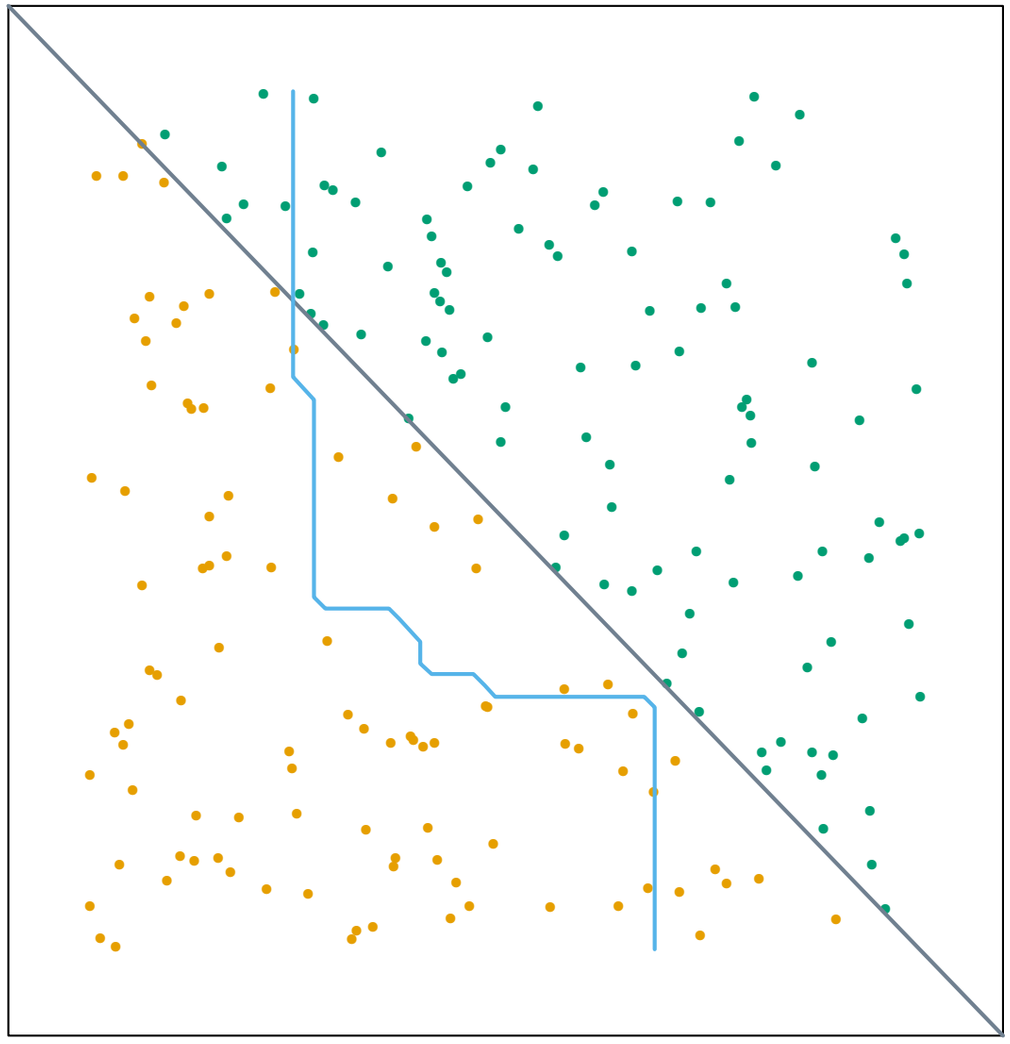


PRML 14.2: Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line).

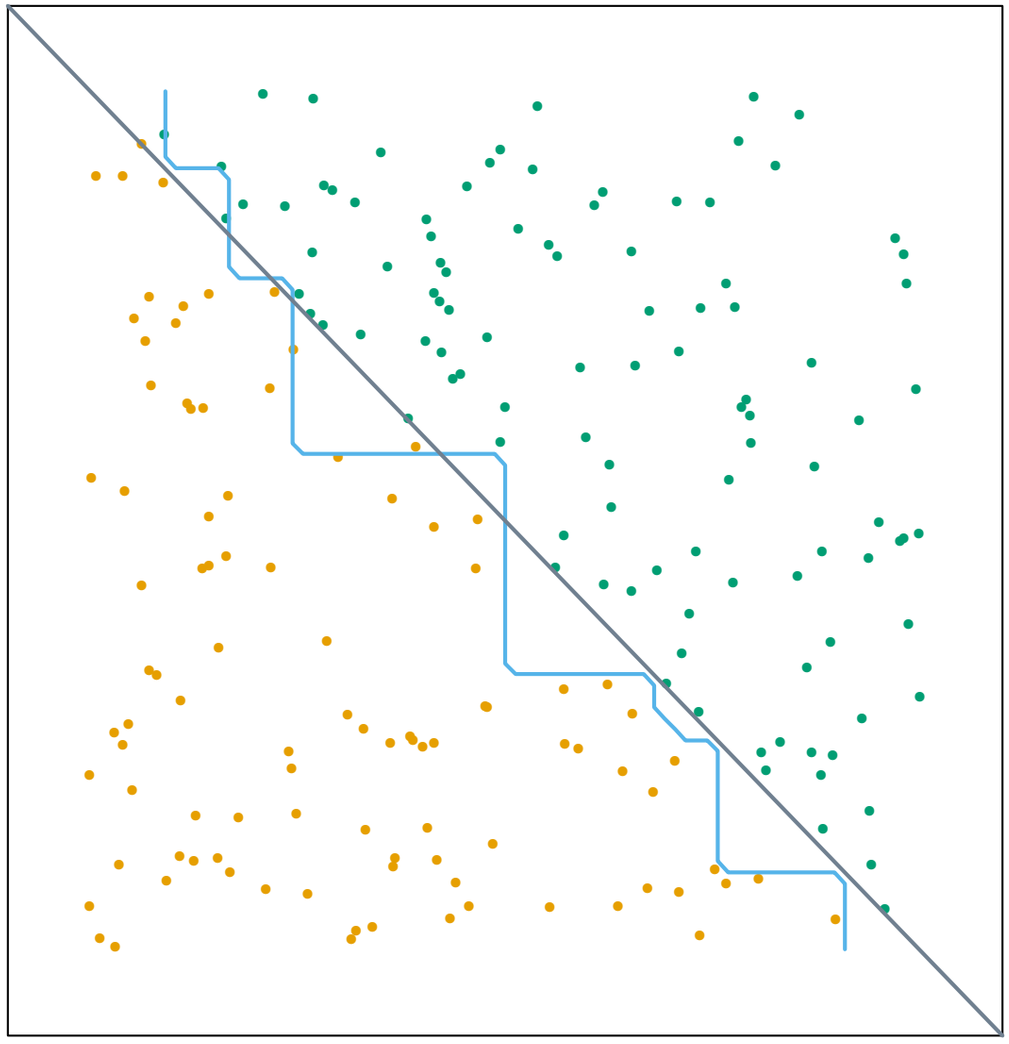
Boosting vs bagging

- What's common?
  - Bagging and boosting aggregate weak learners that are barely better than random guessing to create a strong learner that can make accurate predictions.
- What's different?
  - In bagging, you take bootstrap samples (with replacement) of your data set and each sample trains a (potentially) weak learner.
  - Boosting uses all data to train each learner, but instances that were misclassified by the previous learners are given more weight so that subsequent learners give more focus to them during training.

Bagged Decision Rule



Boosted Decision Rule



- ESL 8.12. Boosting roughly captures the diagonal boundary, while bagging does not. Both methods are based on the decision stump (50 trees). In this simulation, boosting performs a lot better than bagging.

Gradient boosting

- So far, we discussed AdaBoost. There are many boosting procedures in the literature.
- In AdaBoost, shortcomings are identified by high-weight data points.
- In gradient boosting, shortcomings (of existing weak learners) are identified by gradients.
- Both high-weight data points and gradients tell us how to improve our model.

Gradients of loss functions

TABLE 10.2. *Gradients for commonly used loss functions.*

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \le \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

- Negative gradients can be thought as (generalized or pseudo) residuals. These are the parts that existing weak leaners cannot do well. ($f(x_i)$ is prediction for y_i).

- We add a weak learner to compensate the shortcoming of existing weak learners.

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.

2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

   pseudo residuals $r_{im} = -\left[\dfrac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$.

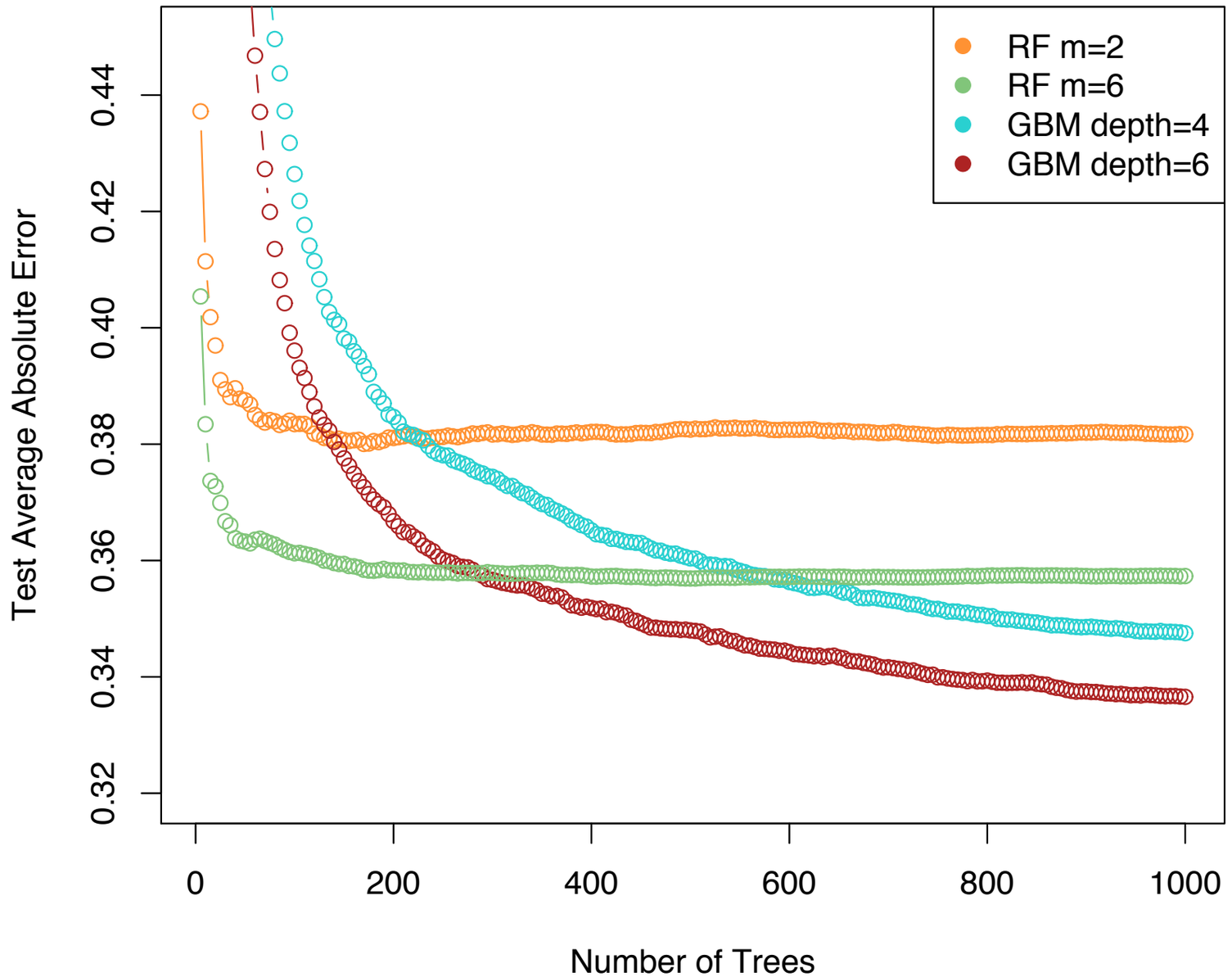
   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.

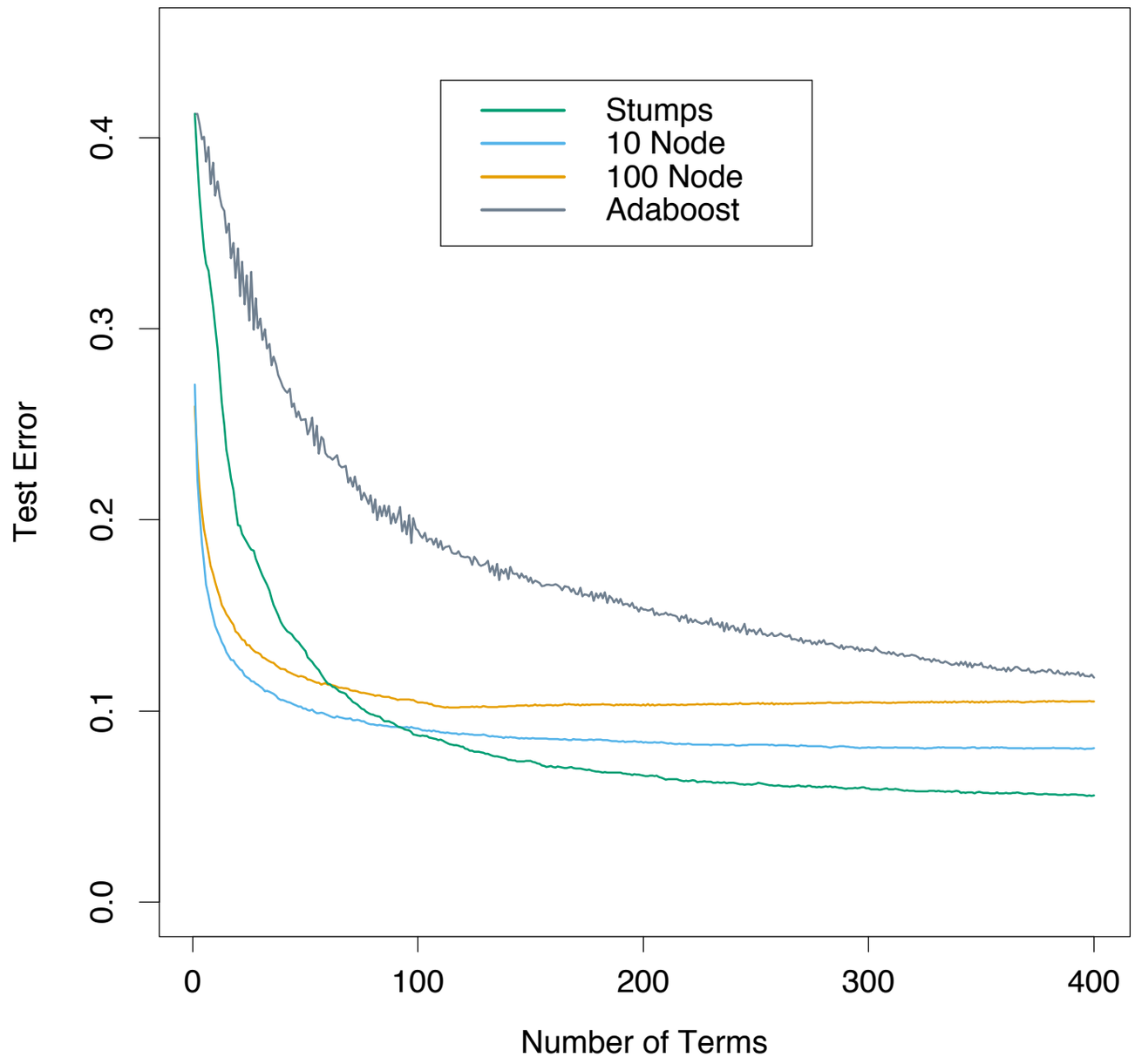


- ESL 15.3: random forests compared to gradient boosting on the California housing data.
- GBM depth=d: gradient boosting method with tree depth d (# of splits).
- RF m: random forests with m candidates for splitting.

Right-Sized Trees for boosting

- In boosting, we grow small trees with no pruning. In bagging, pruning large trees can be considered.
- Boosting with large trees incurs unnecessary variance, and it increases computation.
- Typically, trees with J=2~8 leaves work well. Right size for tree depends on the level of the interactions between the predictor variables.
  - J=2: main effect only
  - J=3: two-way interactions
  - J>4: (J-1)-way interactions

Gradient boosting vs Adaboost



ESL 10.9: Gradient boosting with different tree sizes

Disadvantages of boosting

- Loss of interpretability (black box):
  - The final leaner is a weighted sum of trees, which cannot necessarily be represented by a single tree.
- Computational expensive:
  - We grow small trees, so each step can be done relatively quickly compared to bagging.
- To overcome the first downside, a measure of variable importance has been developed for boosting (also for boosting).

Variable importance

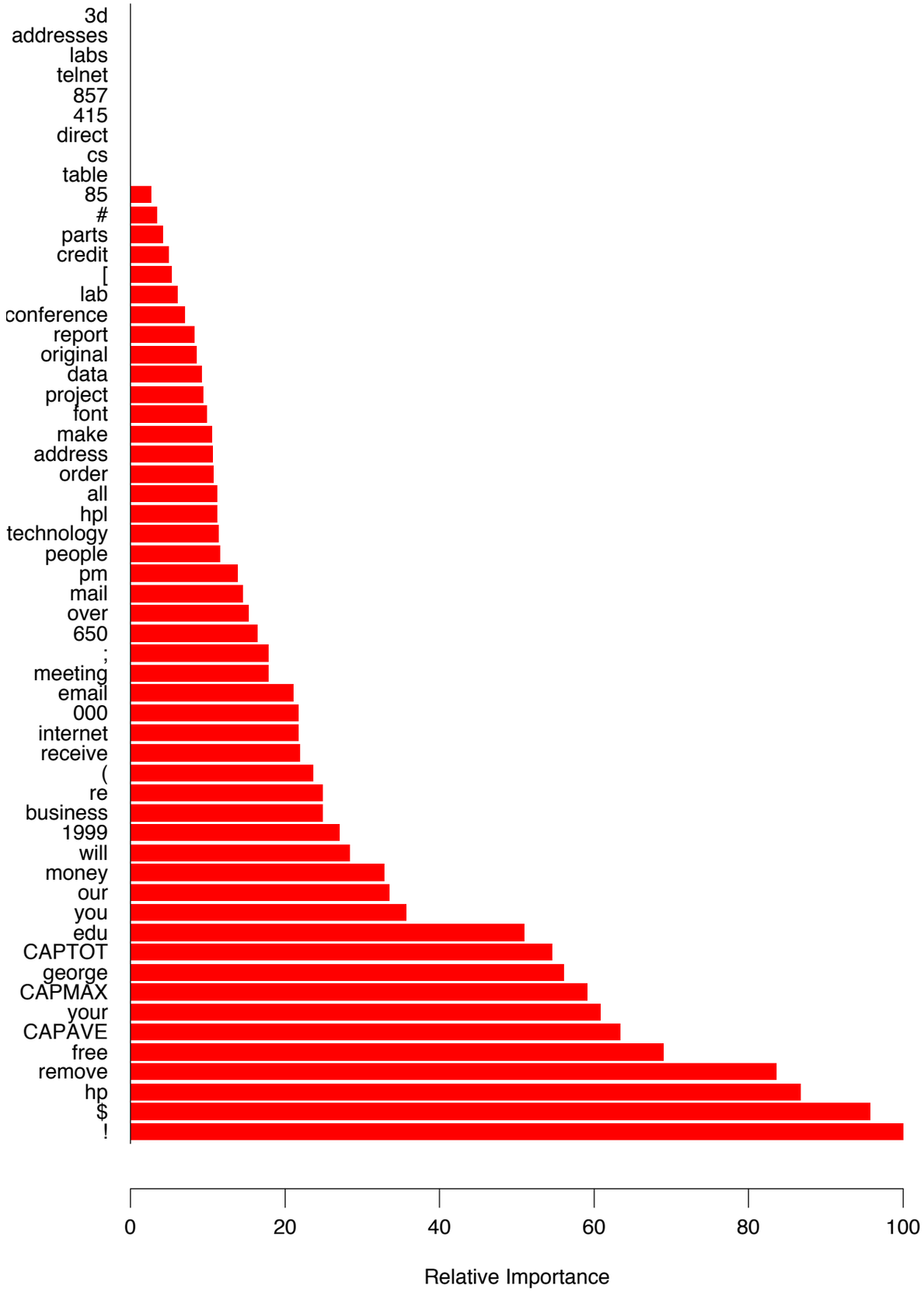
- Single tree: variable importance of l*th* predictor:
  - J: # of internal nodes

split at node t is on n*th* predictor

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 \, I(v(t) = \ell)$$

squared improvement in training error from making this split

- Boosting: averaging the importance of l*th* predictor over all of trees (T_m)
- $$\mathcal{I}_\ell^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_\ell^2(T_m)$$



ESL 10.6: relative predictor variable importance for the spam data (the largest importance is set to100).

Take home messages

- Boosting is a very general and powerful tool to boost performance of a weak learner.

- Boosting procedure sequentially combine weak learners that are trained adaptively using weights or gradients.

- The final learner is not as simple as a single tree. Relative variable importances can be computed by looking at how many times a given variable contributes to a split, across all trees.