Introduction to Machine Learning

Lecture 9
Tree-based models II
Ensemble learning

Goker Erdogan
26 – 30 November 2018
Pontificia Universidad Javeriana

Ensemble learning

- Combining multiple models to produce a better prediction
  - Can make good predictions with weak models
  - Increase in performance at the expense of more computation

- Works especially well with decision trees

- We’ll look at
  - Bagging
  - Random forests
  - Boosting

Bootstrap

- General technique to estimate the distribution of any value of interest

Bootstrap

- Given a set of N samples
- Repeat M times
  - Randomly draw N samples with replacement from data
  - Calculate the value of interest
- Output: M samples of the value of interest

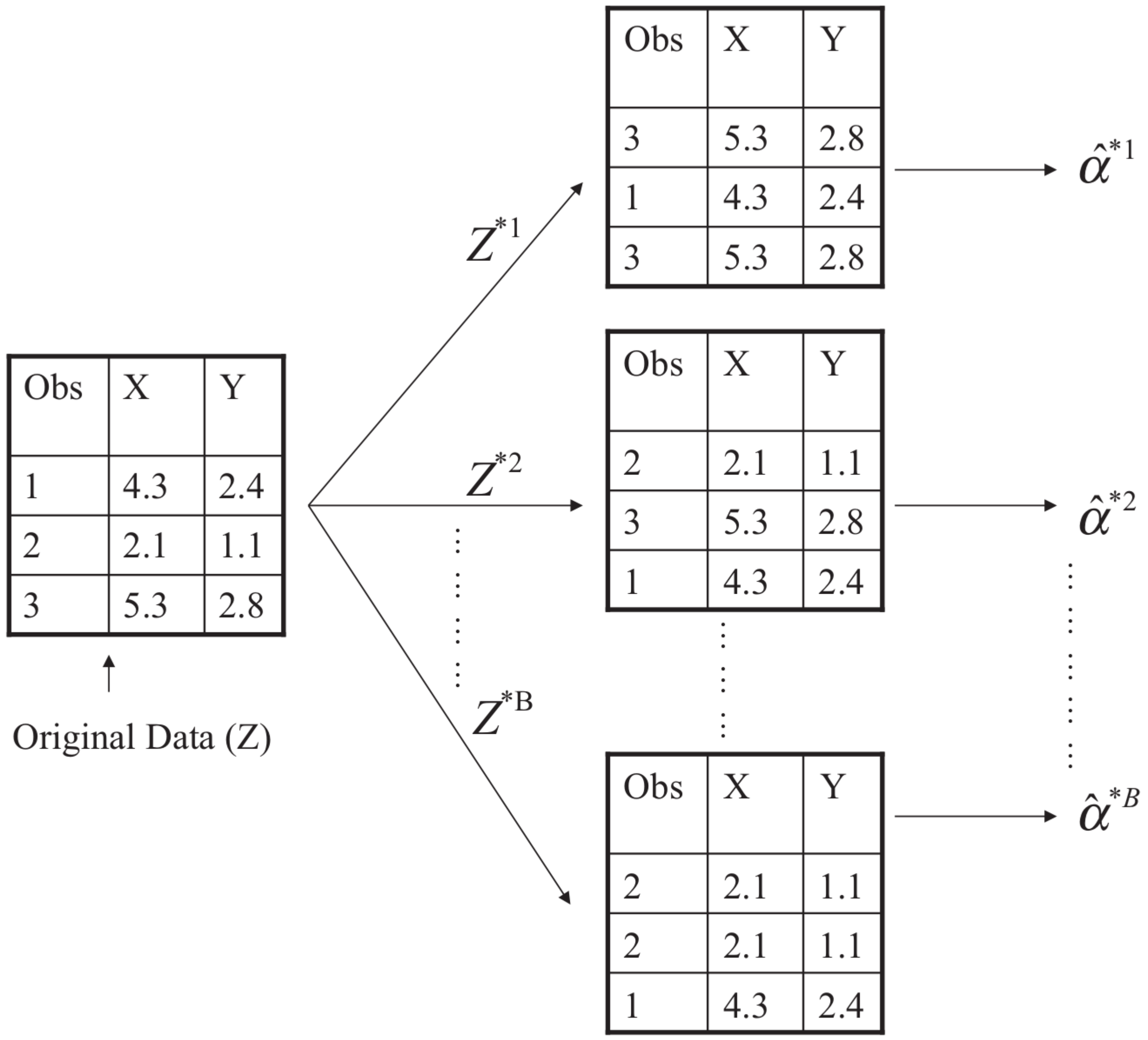


FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

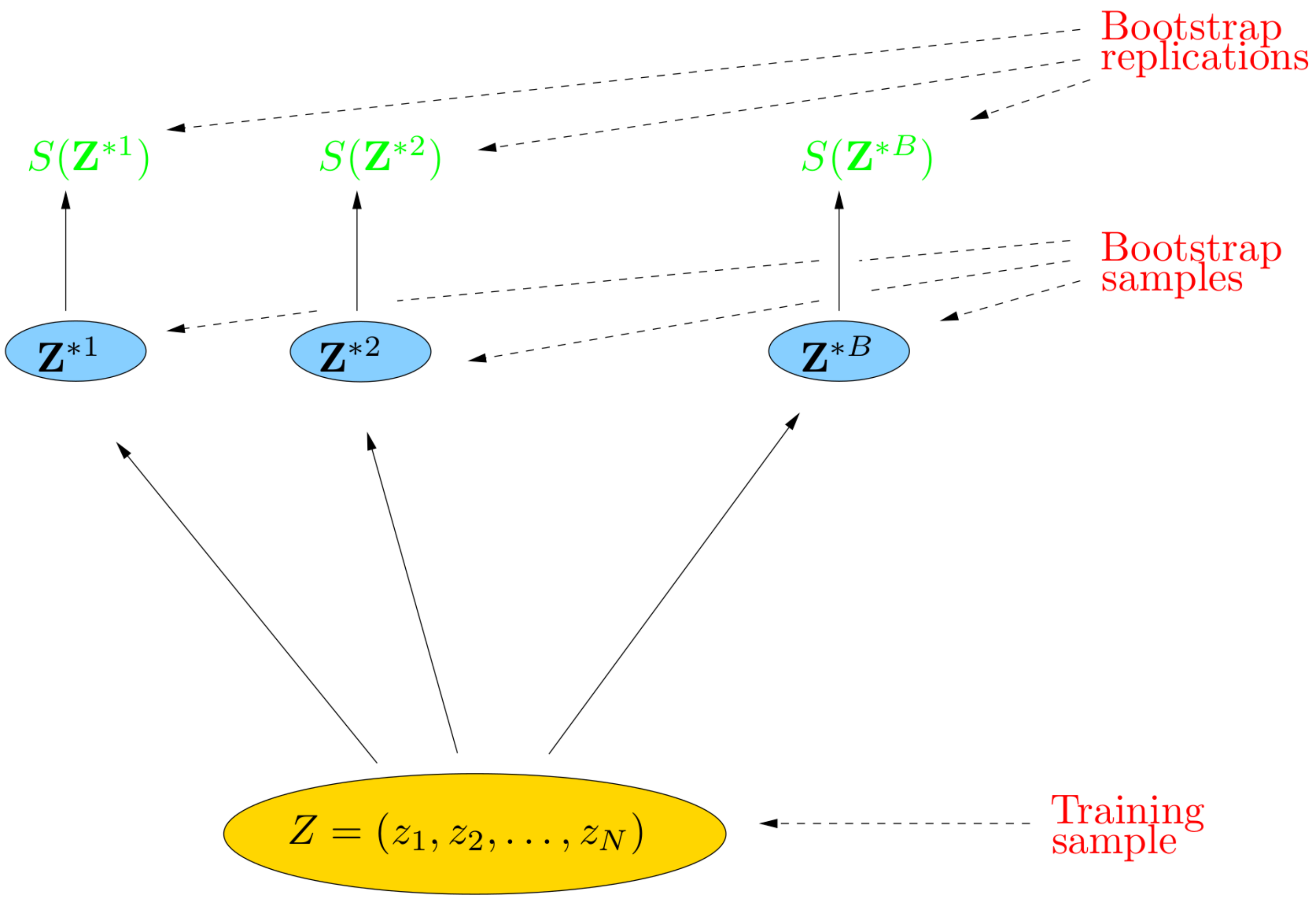
Bootstrap

- Example I
  - Given a set of 100 samples, we want to estimate the mean
  - What should be our confidence in our estimate?
- Example II
  - In a (linear regression) model
    - estimate the standard deviation for a coefficient

- Q: What is the number of distinct examples in a resampled dataset?
  - Given N samples,
    - Resampled dataset will have 0.632*N distinct examples
    - Around 2/3 of data will be in the resampled dataset
    - With ~2/3 probability, a sample will appear in the resampled dataset

Bagging

- Bagging: bootstrap aggregation
  - Use bootstrap to train multiple models
  - Combine these models to make a prediction



Bagging

- How do you combine multiple models?
  - Regression: take the average

$$y_{\mathrm{BAG}}(x) = \frac{1}{M}\sum_m y_m(x)$$

  - Classification: majority voting

$$y_{\mathrm{BAG}}(x) = \mathrm{argmax}_k \sum_m y_{mk}(x)$$

    - If your model outputs probabilities, just average them

Why does it help?

- Assume that our model has variance σ^2
  - Var(y_m) = σ^2
- Now look at the variance of y_{BAG}

$$\mathrm{Var}\left(\frac{1}{M}\sum_m y_m(x)\right) = \frac{1}{M}\mathrm{Var}(y)$$
$$= \frac{1}{M}\sigma^2$$

- Averaging reduces variance
- NOTE
  - This is only true if y_m are uncorrelated

$$\mathbb{E}[y_i y_j] = 0$$

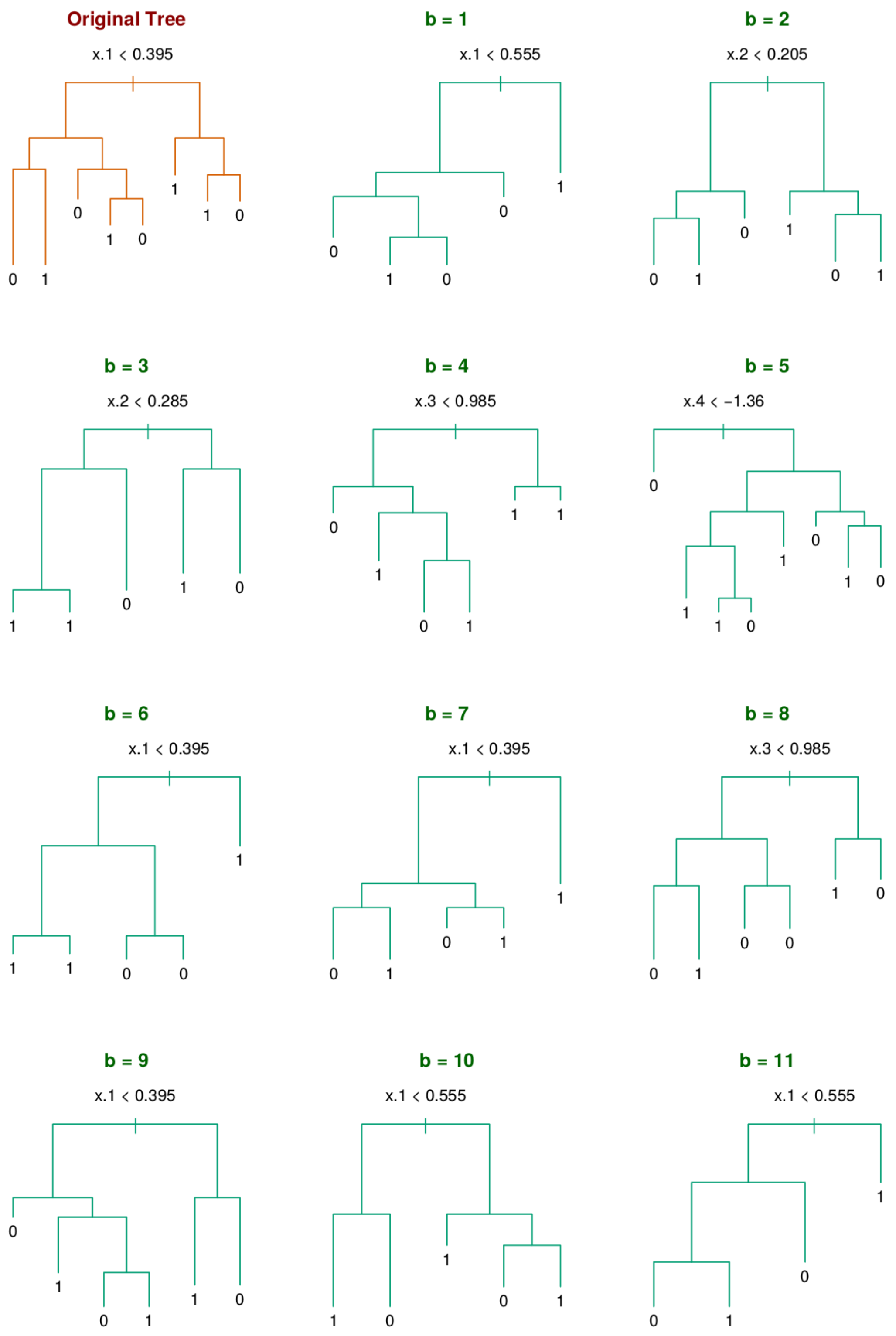
Bagging on decision trees

- Decision trees have high variance
  - Bagging improves performance significantly

Bagging (decision trees)

- Repeat M times
  - Resample a new dataset (bootstrap)
  - Fit a decision tree
    - Do not prune
- Make prediction using all M trees

- Number of models (M) is not critical, as long as it is large
- Less likely to overfit
  - Because of averaging
  - No need to prune the trees


Original Tree
x.1 < 0.395
b = 1
x.1 < 0.555
b = 2
x.2 < 0.205
b = 3
x.2 < 0.285
b = 4
x.3 < 0.985
b = 5
x.4 < −1.36
b = 6
x.1 < 0.395
b = 7
x.1 < 0.395
b = 8
x.3 < 0.985
b = 9
x.1 < 0.395
b = 10
x.1 < 0.555
b = 11
x.1 < 0.555

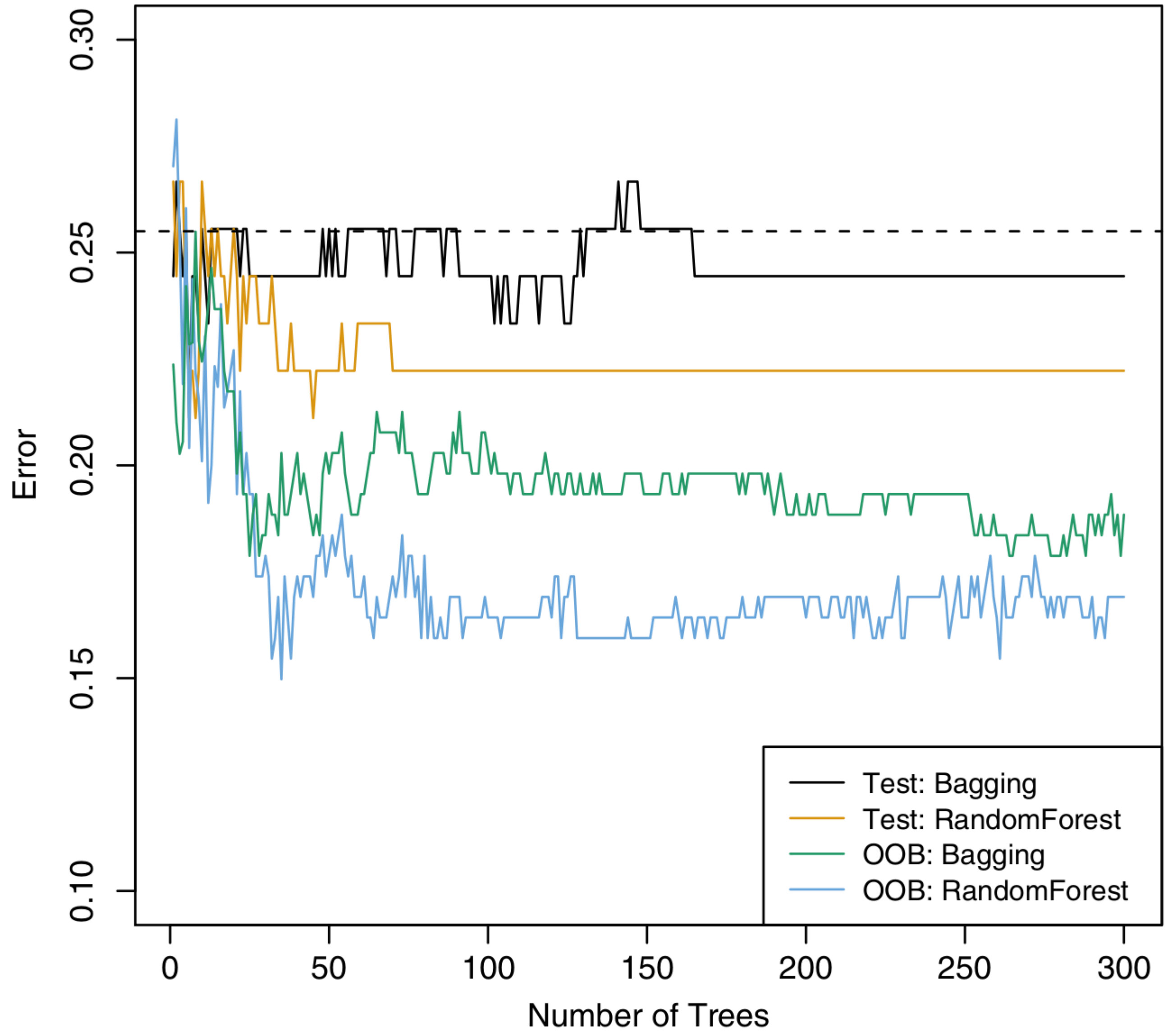


FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

Out-of-bag error estimation

- Can use cross-validation to estimate error

- Bagging allows a simpler error estimate
  - Out-of-bag error estimation
  - Remember each tree is trained on a subset of all training data
  - For each sample n
    - There is a set of trees T_{-n} that did not have that sample in training set
    - Make prediction for sample n using trees in T_{-n}

$$y_{\mathrm{OOB}}(x_n) = \frac{1}{|T_{-n}|} \sum_{m \in T_{-n}} y_m(x_n)$$

    - Calculate out-of-bag error

$$E_{\mathrm{OOB}} = \sum_n (y_{\mathrm{OOB}}(x_n) - t_n)^2$$

Out-of-bag error estimation contd.

- No need for a separate validation set
- Can be used to pick M
  - Stop when OOB error does not improve anymore
  - Early stopping
- In the limit $M \to \infty$
  - OOB error = leave-one-out CV error
- However, still useful to validate model on a separate set
  - OOB error can underestimate/overestimate true test error

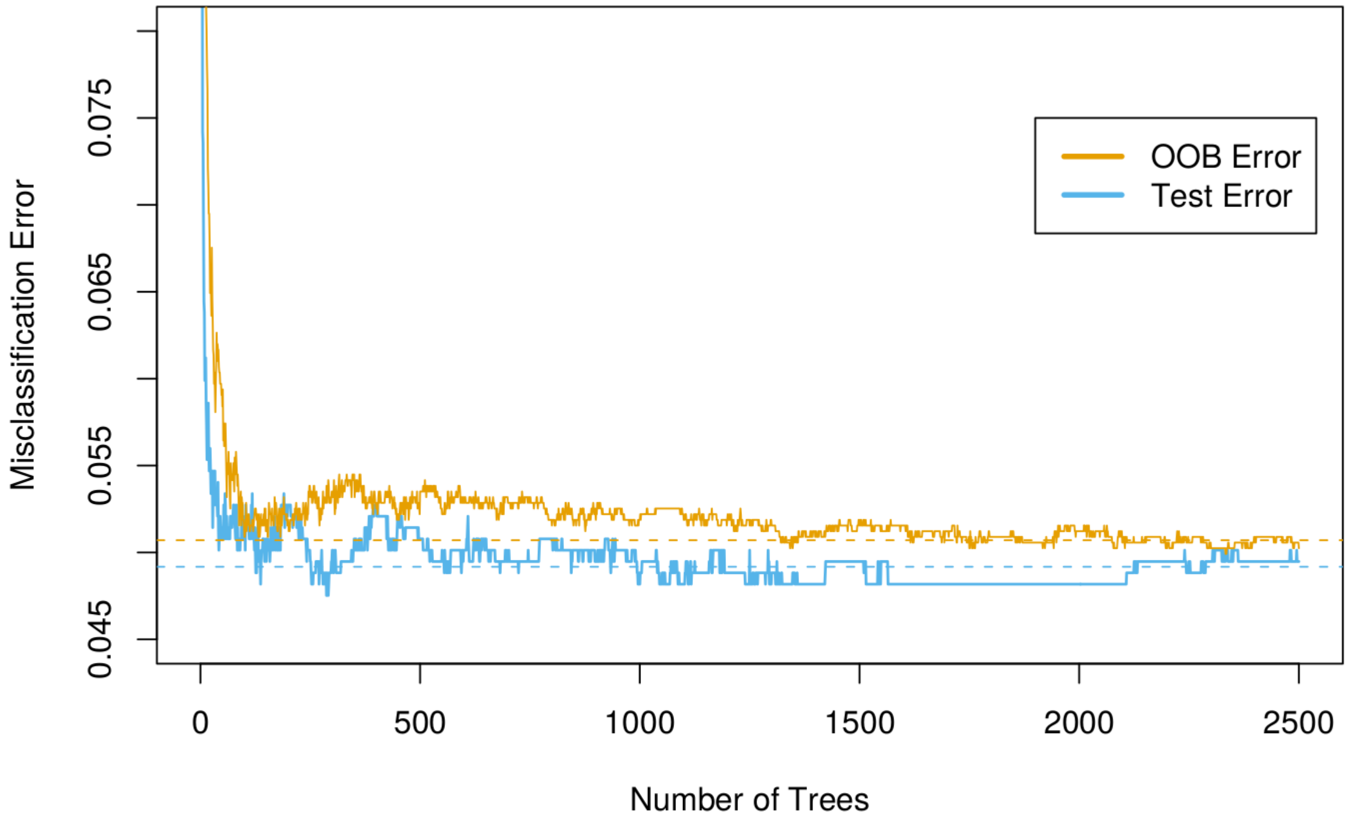


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Bagging

- Good general technique
  - Reduces mean squared error in the limit $M \rightarrow \infty$
  - Not guaranteed to reduce 0-1 misclassification error
    - But often does
  - It can be shown
    - A committee of M weak classifiers will reduce error
    - Weak classifier: error < 0.5
- For decision trees, loses interpretability
  - Still can measure variable (feature) importance
- Bagging assumes uncorrelated models

What if models are correlated?

- Remember the variance of a bagging estimator

$$\mathrm{Var}\left(\frac{1}{M}\sum_m y_m(x)\right) = \frac{1}{M}\sigma^2$$

$$\mathbb{E}[y_i y_j] = 0$$

- Assume models are correlated

$$\mathbb{E}[y_i y_j] = \rho\sigma^2$$

- Then

$$\mathrm{Var}\left(\frac{1}{M}\sum_m y_m(x)\right) = \frac{1-\rho}{M}\sigma^2 + \rho\sigma^2$$

Random forests

- Averaging doesn’t help as much if models are correlated
- In bagging, y_m can be highly correlated
  - Imagine one feature is strongly related to output
  - All trees will pick this as their first node
  - Increased correlation between trees

- Random forests
  - De-correlate trees by using subsets of features at each split
  - Each time a split is considered,
    - Pick $s < D$ features randomly and consider only these
    - Usually $s \sim \sqrt{D}$
  - $s=D$ is just bagging!

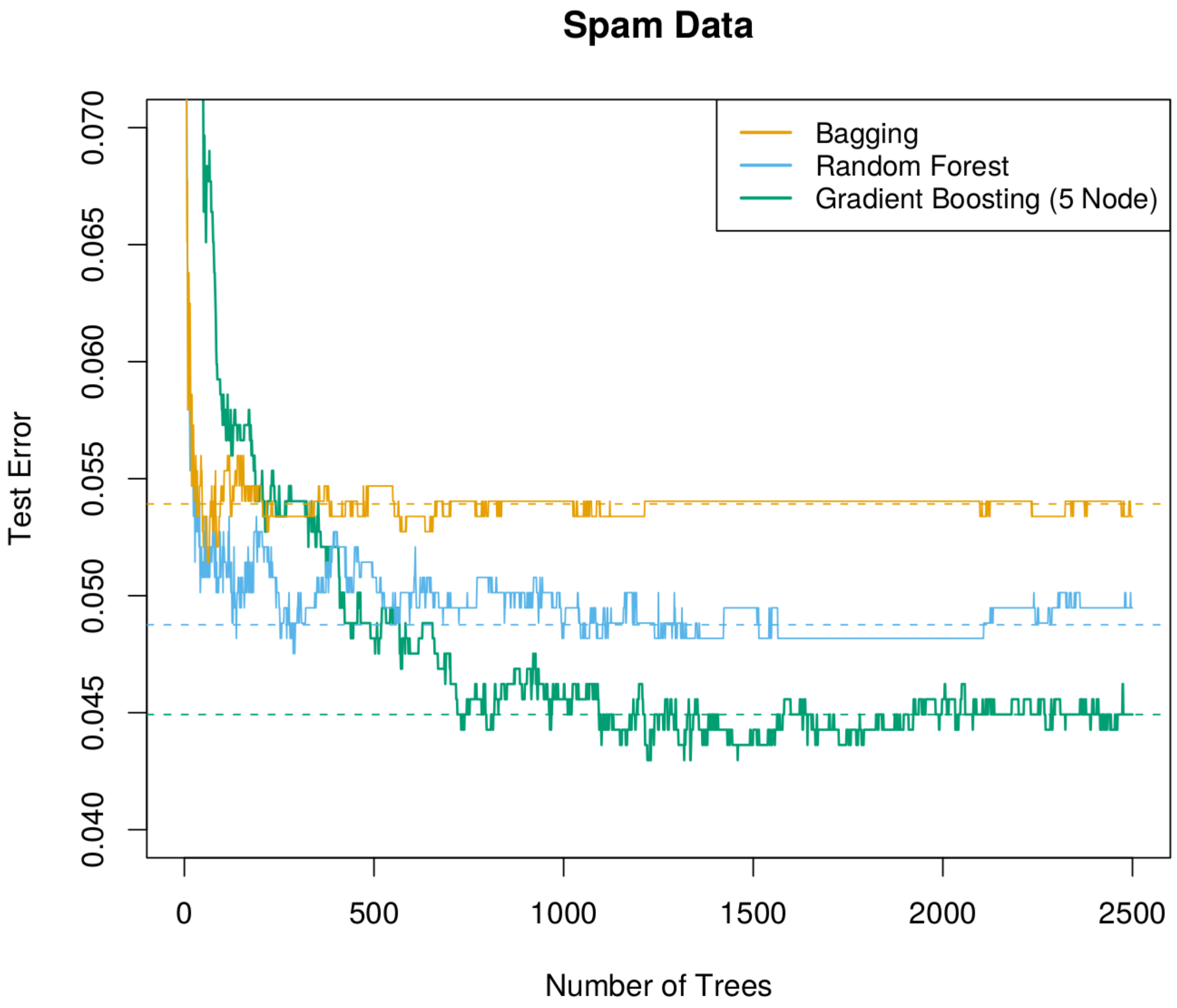


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Boosting

- Another general approach to build an ensemble of model
  - Learn models sequentially instead of independently
  - Popular with decision trees
    - One of the best off-the-shelf techniques

- General idea
  - Fit model on (full) training set
    - No resampling
  - Look at errors of the model
  - Fit the next model so it focuses more on samples with high error
  - Repeat

Algorithm 8.2 *Boosting for Regression Trees*

```
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
    (a) Fit a tree $\hat{f}^b$ with $d$ splits ($d + 1$ terminal nodes) to the training
        data $(X, r)$.
    (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
            $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$        (8.10)
    (c) Update the residuals,
            $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$        (8.11)
3. Output the boosted model,
            $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$        (8.12)
```

Boosting regression trees

- Key idea: fit to residuals
- Risk of overfitting
  - Pick M (number of models) with cross-validation
    - Stop when error stops decreasing
  - Don’t fit large trees
    - Set a maximum size (d: number of splits)
    - Can pick even d=1 (stumps)
    - d controls interaction level
  - Pick λ small (e.g., $\lambda = 0.01, 0.001$)

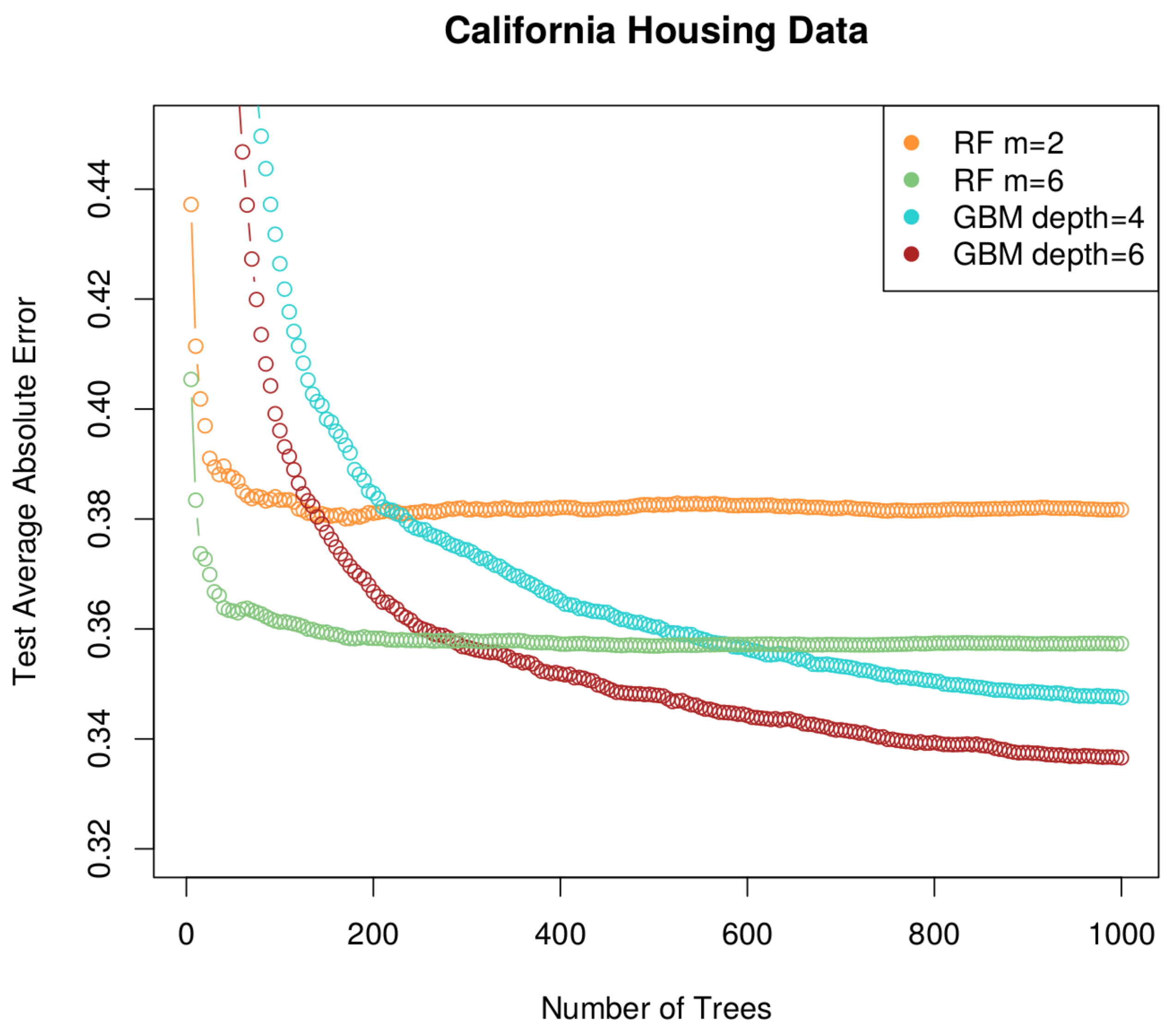


FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with* $m = 2$ *and* $m = 6$*. The two gradient boosted models use a shrinkage parameter* $\nu = 0.05$ *in (10.41), and have interaction depths of* 4 *and* 6*. The boosted models outperform random forests.*

AdaBoost

- Boosting for classification
  - There is no residual
- Idea
  - Keep weights for each sample
  - Increase/decrease weight of a sample depending on whether it is correctly classified or not
- How to increase/decrease weights?
  - AdaBoost
- Note we use -1, +1 to represent class labels
  - Binary classification
  - Class 1=-1, Class 2=+1

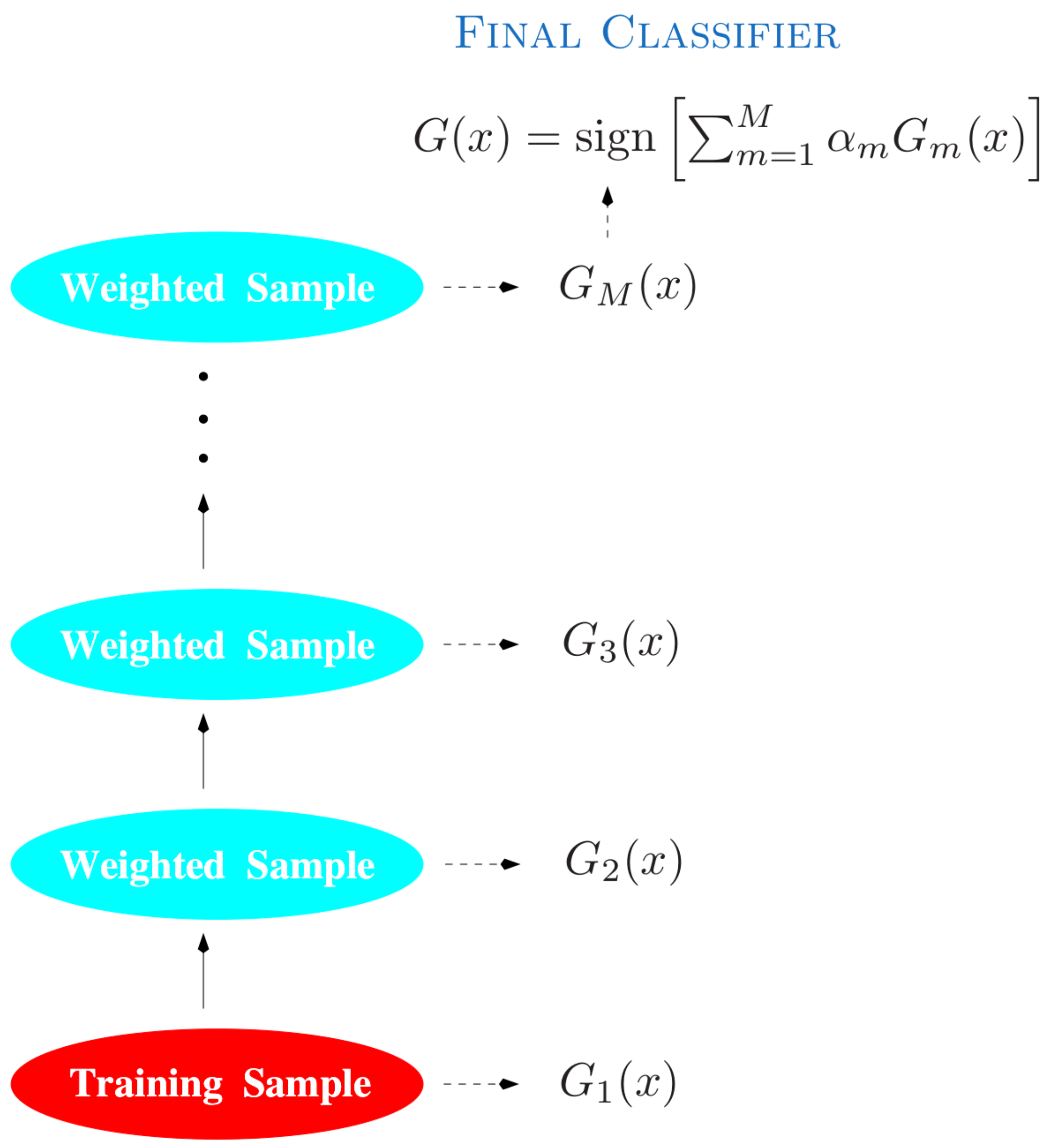


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

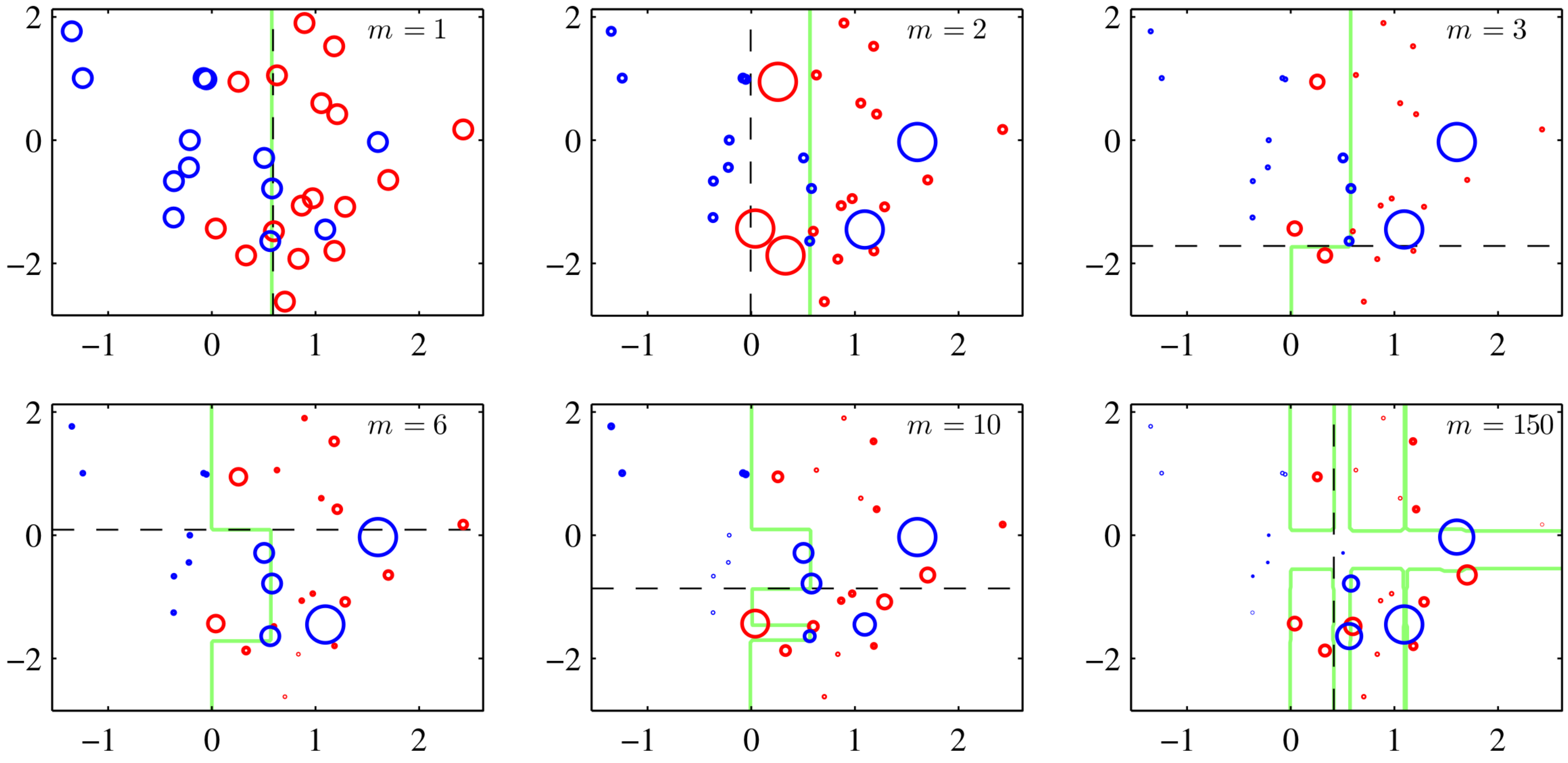


Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \dots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \dots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

AdaBoost

- AdaBoost is not minimizing 0-1 classification loss
  - Minimizes an exponential loss

$$E = \sum_n \exp(-t_n f(x_n))$$

$$f(x) = \sum_m \alpha_m f_m(x)$$

  - The optimal f is the log-odds

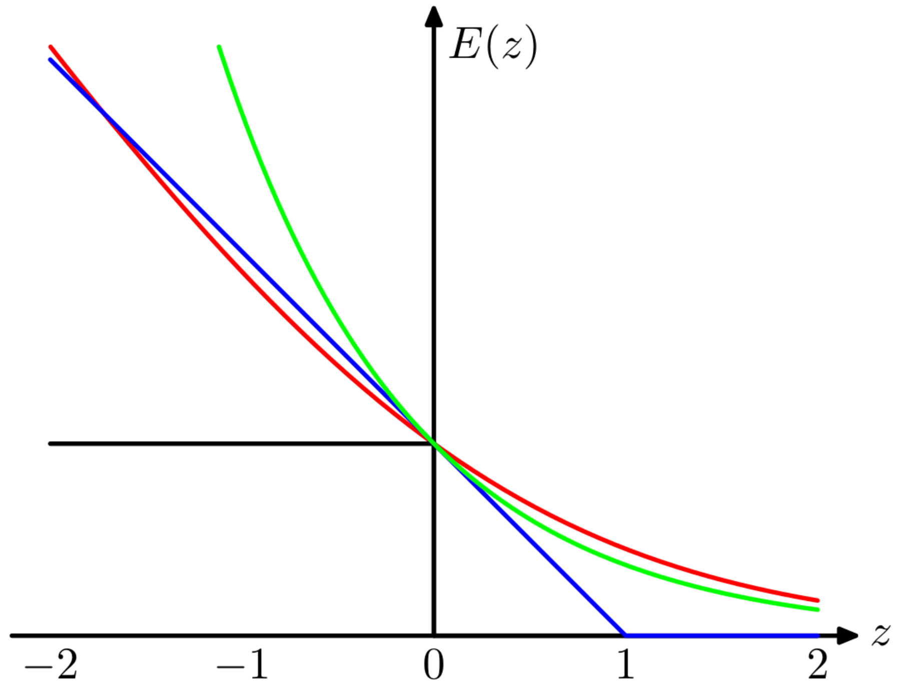
$$f^*(x) = \frac{1}{2} \ln \frac{p(t=1|x)}{p(t=-1|x)}$$

  - AdaBoost is approximating log-odds

AdaBoost and beyond

- Exponential loss is not robust
  - Outliers and mislabeled samples can hurt performance
- Robust alternatives to exponential loss
  - Cross-entropy
  - Hinge-loss

Figure 14.3 Plot of the exponential (green) and rescaled cross-entropy (red) error functions along with the hinge error (blue) used in support vector machines, and the misclassification error (black). Note that for large negative values of $z = ty(\mathbf{x})$, the cross-entropy gives a linearly increasing penalty, whereas the exponential loss gives an exponentially increasing penalty.



AdaBoost and beyond

- With different losses (like cross-entropy)
  - No longer a simple weighting scheme

- However a general boosting technique gradient boosting
  - Applicable to any loss function
  - Similar to boosting regression trees
    - Fit to some residual-like quantities

Boosting vs. bagging

- Bagging: combine low bias-high variance models
  - Average to reduce variance
  - Little risk of overfitting
    - Don’t trust any individual model
    - Always average
    - So each individual model is safe to overfit
  - By itself not very good
    - Random forests
- Boosting: combine high bias-low variance models
  - Build sequentially to reduce bias
  - Risk of overfitting
    - If a model does well on a sample, we trust it
    - So if an individual model overfits, the ensemble overfits
  - Good general technique

Summary

- Bootstrap
- Bagging
  - Bagging for decision trees
- Random forests
- Boosting
  - Boosting regression trees
  - AdaBoost

- Exercises
  - In bootstrap, show that the probability of a sample being picked is around 0.632.
  - Do the labs in Section 8.3.3 and 8.3.4 in ISLR

References

[1] James, Witten, Hastie, and Tibshirani. An Introduction to Statistical Learning with Applications in R. Chapter 5 and Chapter 8.

[2] Hastie, Tibshirani, and Friedman. The Elements of Statistical Learning. Chapter 8, Chapter 10, and Chapter 15.

[3] Bishop. Pattern Recognition and Machine Learning. Chapter 14.