COMP6915 Machine Learning

Bagging and Boosting

Dr. Lourdes Peña-Castillo
Departments of Computer Science and Biology
Memorial University of Newfoundland

Bootstrap

- General tool for assessing statistical accuracy
  - Used to quantify uncertainty associated with a given learning method or estimator
  - Widely applicable
- The bootstrap method uses the raw data to generate new datasets by **sampling with replacement** from the training data
  - Also called nonparametric bootstrap
- Each bootstrap data set contains *n* observations randomly selected with replacement from the original data set
  - That is, the same observation can occur more than once in the bootstrap data set

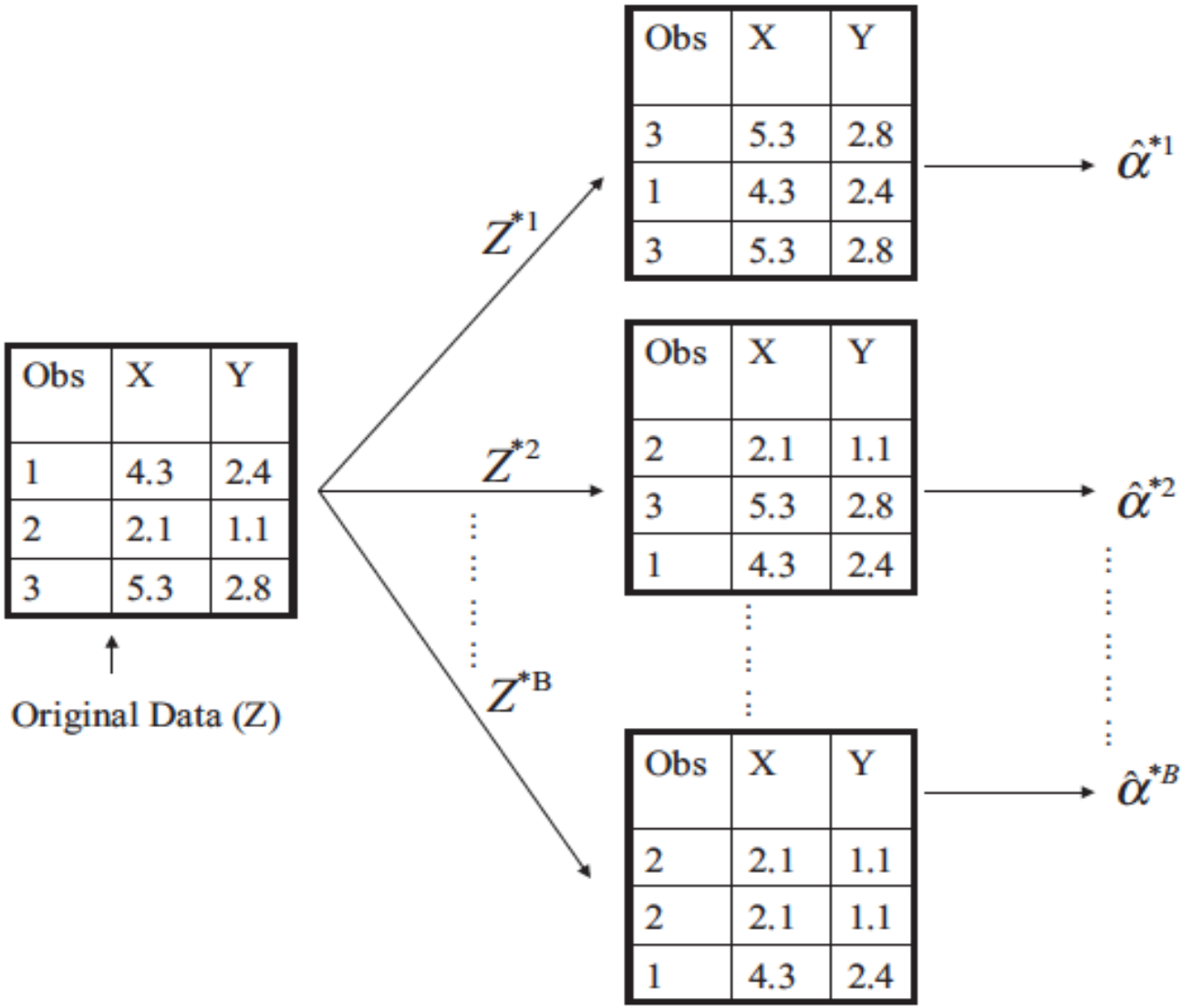


FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

Fig. from *An Introduction to Statistical Learning: with Applications in R* by James, Witten, Hastie, and Tibshirani, 2013.

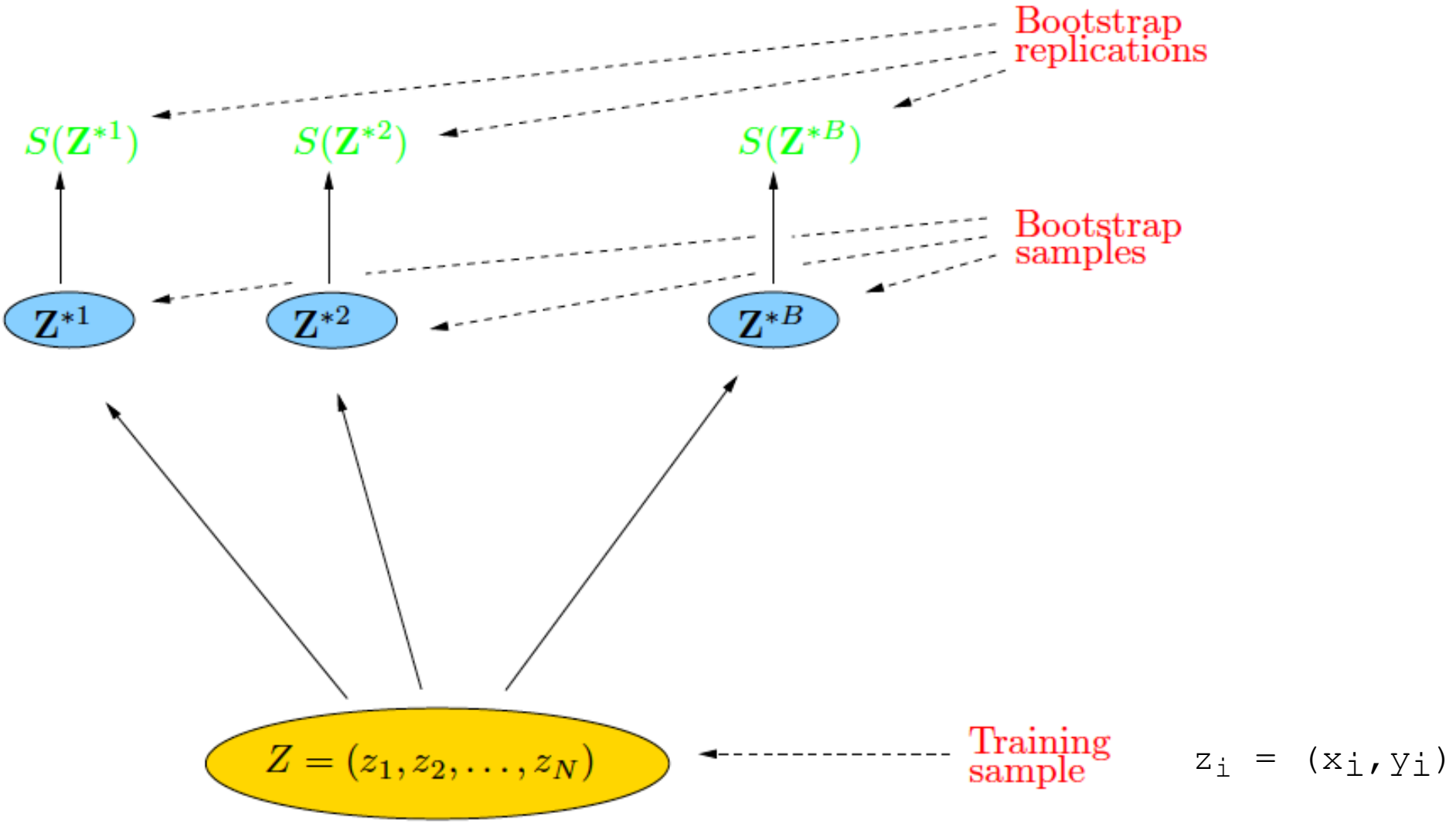


FIGURE 7.12. *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\mathbf{Z})$ computed from our dataset. B training sets $\mathbf{Z}^{*b}$, $b = 1, \ldots, B$ each of size N are drawn with replacement from the original dataset. The quantity of interest $S(\mathbf{Z})$ is computed from each bootstrap training set, and the values $S(\mathbf{Z}^{*1}), \ldots, S(\mathbf{Z}^{*B})$ are used to assess the statistical accuracy of $S(\mathbf{Z})$.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Bagging

- A general-purpose procedure for reducing the variance of a statistical learning method
  - The decision trees we discussed in lecture 6 suffer from high variance, thus bagging is frequently used with decision trees
- Idea:
  - Averaging a set of observations reduces variance
- Process:
  - Generate *B* bootstrapped training sets
  - Build a separate model using each bootstrapped training set *b*, and
  - Average the resulting predictions

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

Applying bagging to trees

- Construct *B* trees using *B* bootstrapped training data sets
  - between 50 to thousands of bootstrapped trees are built
  - *B* is not critical because bagging doesn't overfit
- Trees are grown deep, and not pruned
  - Each tree has high variance but low bias
  - Combining the trees predictions reduce the variance
- For regression, average trees predictions
- For classification, use majority vote

Suppose we have a tree that produces a classifier $\hat{G}(x)$ for a K-class response. For simplicity, let's consider an underlying indicator-vector function $\hat{f}(x)$, with value a single one and $K-1$ zeroes, such that $\hat{G}(x) = \arg\max_k \hat{f}(x)$.

The bagged estimate $\hat{f}_{bag}(x)$ is a K-vector $[p_1(x), p_2(x), \ldots, p_K(x)]$, with $p_k(x)$ equal to the proportion of trees predicting class k at x. The bagged classifier selects the class with the most "votes" from the B trees, $\hat{G}_{bag}(x) = \arg\max_k \hat{f}_{bag}(x)$.

Estimating the class-probability

- To estimate the bagged class probability, average the class proportions in the terminal nodes for the trees predicting class *k* for *x*
  - Do not use the voting proportions $p_k(x)$

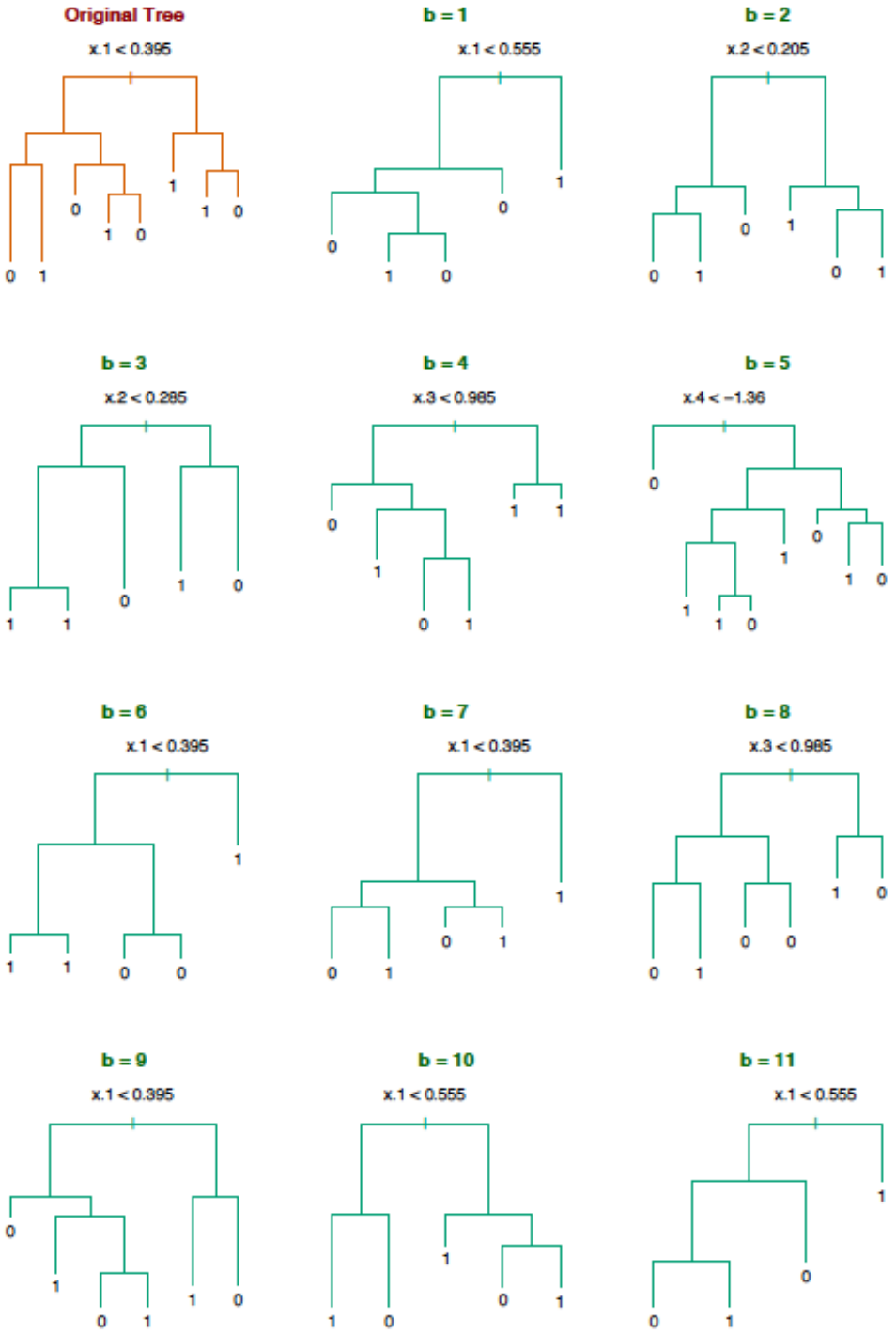


FIGURE 8.9. *Bagging trees on simulated dataset. The top left panel shows the original tree. Eleven trees grown on bootstrap samples are shown. For each tree, the top split is annotated.*

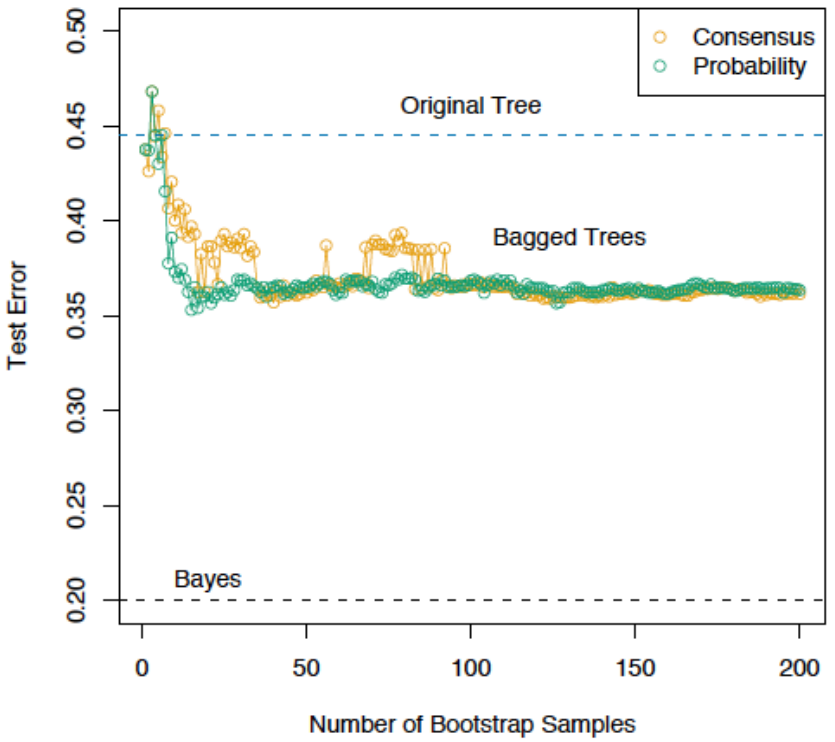


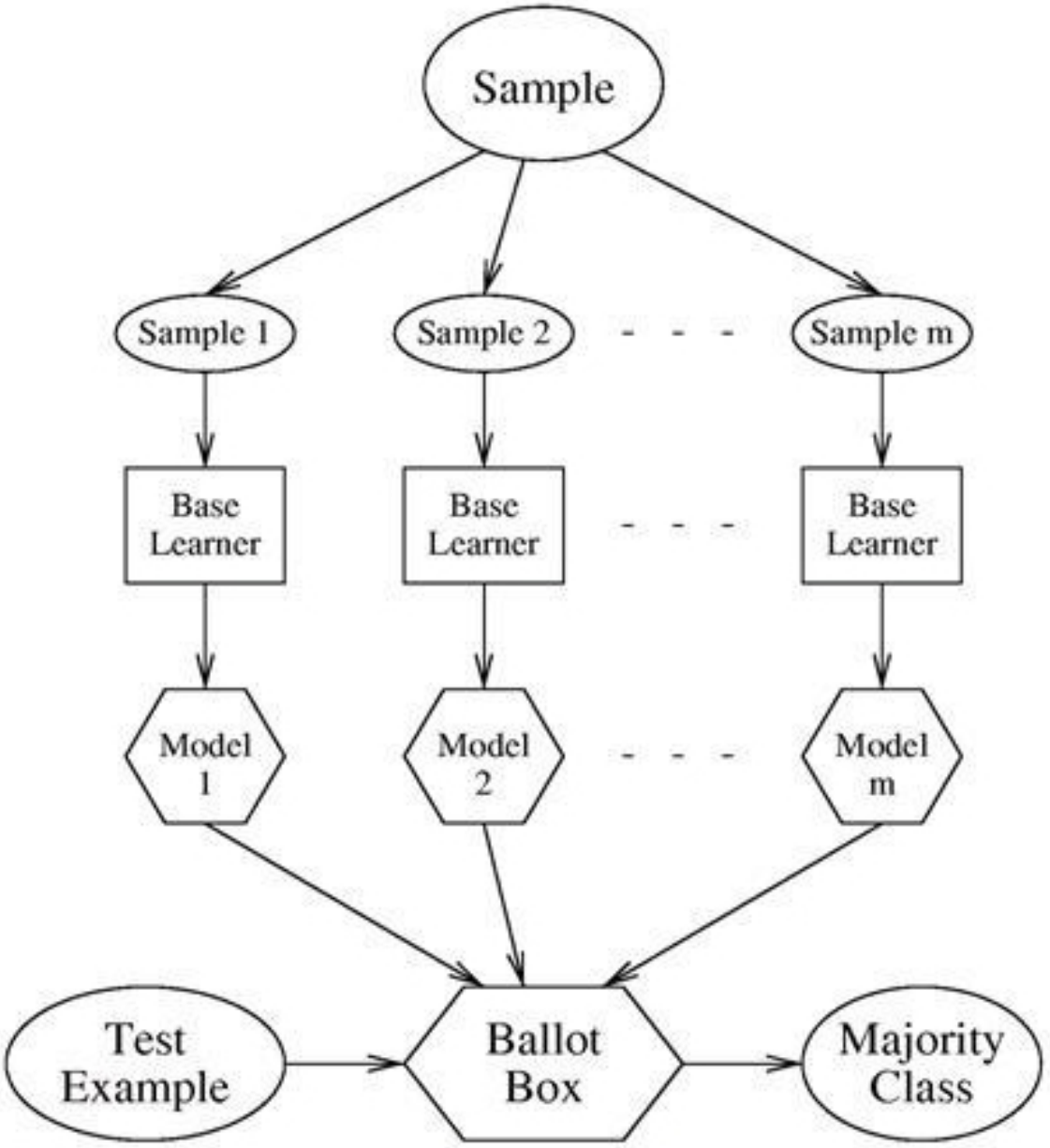
FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

Figs. from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Bagging

- Bagging can dramatically reduce the variance of unstable procedures (like trees) and improve prediction performance
- This effect can be seen in terms of a consensus of **independent** *weak learners* (Dietterich, 2000)
  - Bagged trees are not independent. Random forests improve on bagging by reducing the correlation between the sampled trees.
- Note that when we bag a model, any simple structure in the model is lost. That is, a bagged tree is no longer a tree, and thus, not so easily interpretable

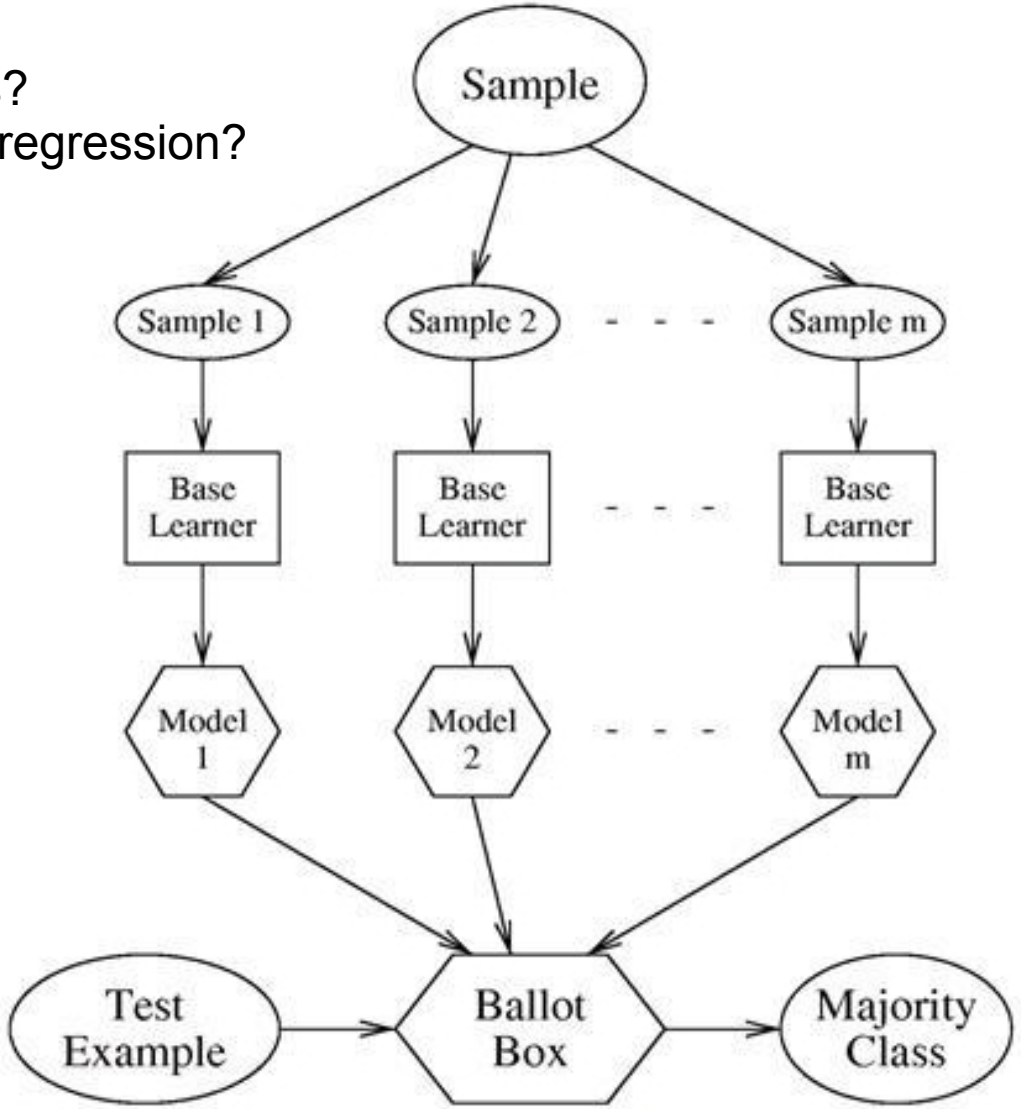
Bagging is a general procedure



Slides by Pedro Domingos, Tom Mitchell and Thomas Dietterich

What should the base learners produce for bagging to work?

Why does bagging work for trees?
Could it work for LDA or Logistic regression?



Slides by Pedro Domingos, Tom Mitchell and Thomas Dietterich

Out-of-Bag Error Estimation

- To estimate the test error of a bagged model one uses the instances not used to fit a given bagged tree
  - The out-of-bag (OOB) instances
- For each instance, we predict its response using each of the trees for which that instances was an OOB instance (~ $B/3$ predictions) and take the average of the predictions (for regression) or majority vote (for classification)
- This obtains a single OOB prediction for all the instances

Out-of-Bag Error Estimation

- These OOB predictions can be used to calculate the overall OOB MSE or classification error.
- The resulting OOB error is a valid estimate of the test error for the bagged model as the response for each instance is predicted using only the trees that were not fit using that observation
  - Not need for cross-validation nor a validation set
- With *B* sufficiently large, OOB error is equivalent to leave-one-out cross-validation error

Measuring variable importance

- To obtain an overall estimate of the importance of each feature, record the total amount that the RSS (for regression) or the Gini index (for classification) decrease due to splits over a given feature and average over all *B* trees
- A large value indicates an important feature

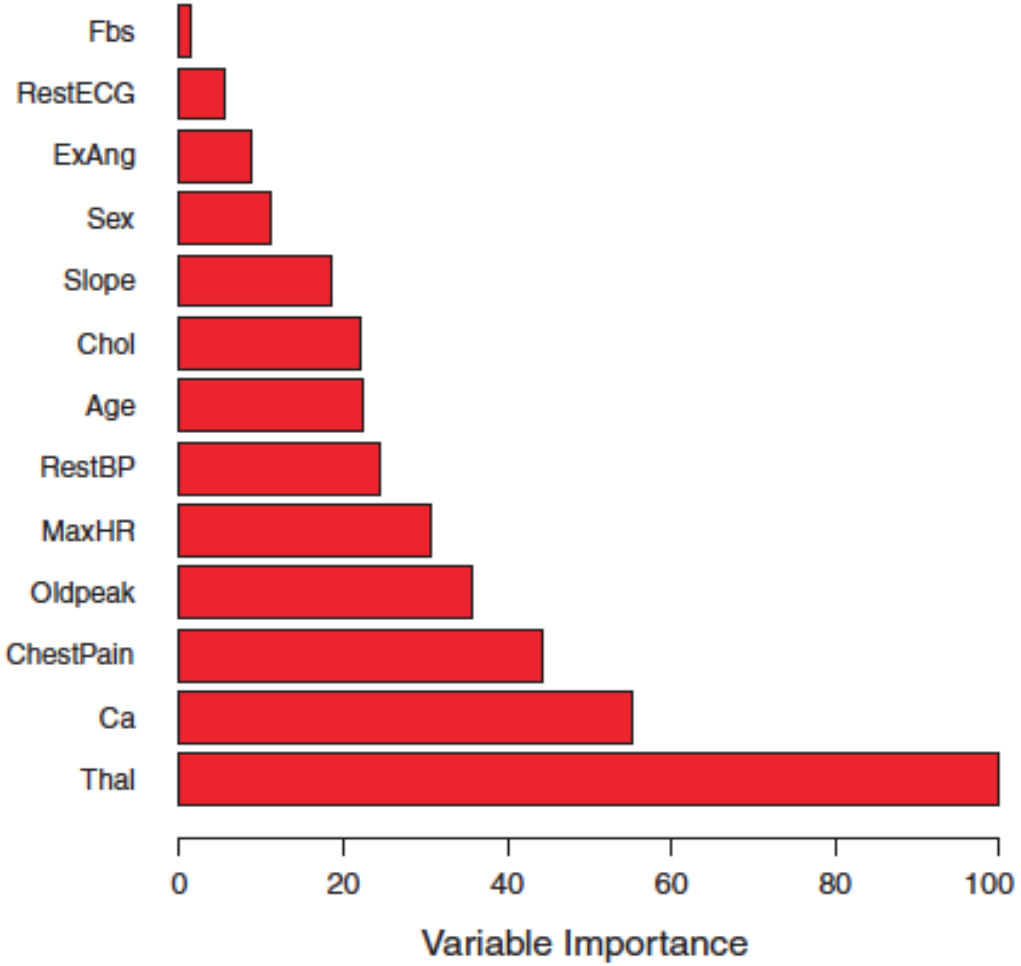


FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

Fig. from *An Introduction to Statistical Learning: with Applications in R* by James, Witten, Hastie, and Tibshirani, 2013.

Boosting

- General method applied to many learning approaches
  - Originally designed for classification problems
- Idea:
  - Design training data sets (instead of random selection done in bagging) so that the next learner improves on the examples were the previous learner failed
- Process:
  - Sequentially apply a weak learning algorithm to repeatedly *B* modified version of the data (*B* shouldn't be too large as boosting does overfit)
- Resulting classifier:
  - A sequence of weak learners combined by a weighted majority vote

AdaBoost

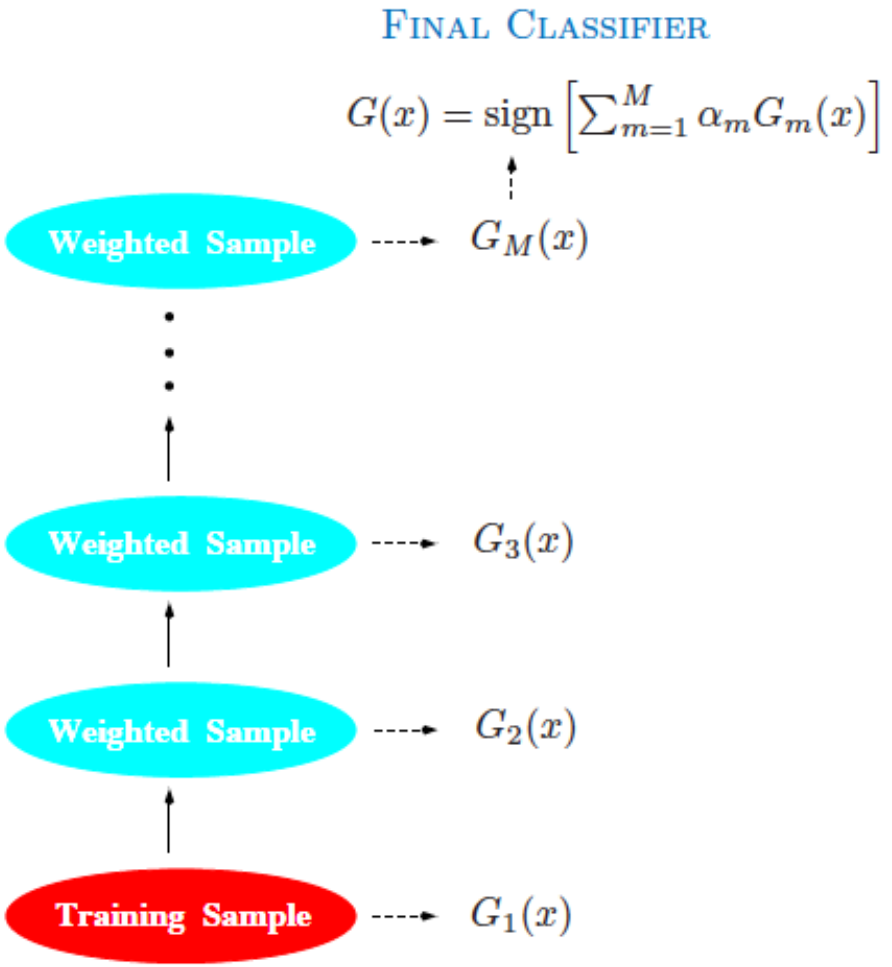


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

- Each G_i is a weak learner
  - What is a weak learner?
- α_i are computed by the boosting algorithm and weight the contributions of each respective G_i
- The data modifications at each boosting step consist of applying weights to each of the training observations (x_i, y_i)

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

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
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (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 $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

AdaBoost

- In the first step, a classifier is trained on the original data
- At each iteration *m*, the observations that were misclassified by the classifier constructed in step *m-1* have their weights increased by a factor exp(α_m)
- As iterations proceed, hard-to-correctly-classify observations receive higher weight
  - Each successive classifier is forced to concentrate on those training observations that are misclassified by previous classifiers in the sequence
- The final classifier returns a discrete class label (+ or -)

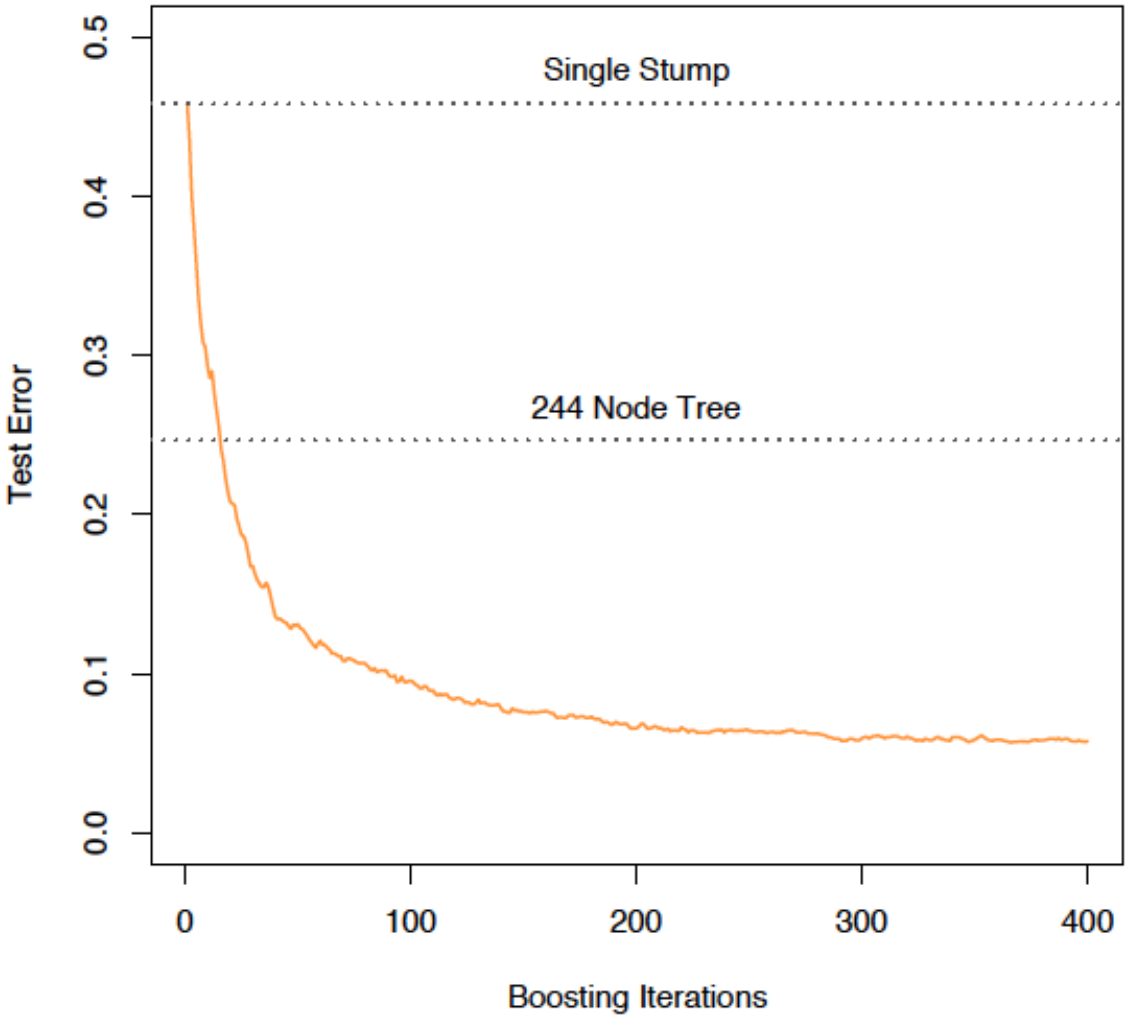


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Boosting regression trees

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:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Parameters
* *B* number of trees (use CV to select *B*)
* λ shrinkage parameter (a small positive number ~0.01 to 0.001)
* *d* number of splits in each tree (often *d* = 1 and the tree is a stump using one feature)

Fig. from *An Introduction to Statistical Learning: with Applications in R* by James, Witten, Hastie, and Tibshirani, 2013.

Combining Bagging and Boosting

- Multiple Boosting – boosting is used on several bootstrapped samples of the training data
- Found to be more accurate than bagging and more stable than boosting
- Suitable for parallel processing

Error rates (top) and error rates ratios (bottom) on 40 domains from the UCI machine learning repository

	C4.5	Boost	Bag	MB
average	19.18	15.97	16.35	15.42

	Boost	Bag	MB	MB vs	
	vs C4.5			Boost	Bag
average	.80	.86	.78	1.00	.89
w/t/l	33/0/7	34/1/5	35/0/5	24/1/15	33/1/6
p. of wtl	< .0001	< .0001	< .0001	.0998	< .0001
significant w/t/l	27/8/5	23/17/0	29/10/1	12/22/6	13/27/0
p. of sign. wtl	< .0001	< .0001	< .0001	.1189	.0001

Results from Zheng and Webb, International conference on parallel and distributed processing techniques and applications (PDPTA) 1998.

By now you should be able to...

- explain bootstrap and generate bootstrapped training sets
- explain how bagging works, apply bagging, and estimate the bagged class-probability
- know which learning methods are suitable for bagging
- define the out-of-bag error
- explain how variable importance is measured in bagging
- define a weak learner
- describe how boosting works and apply boosting to classification and regression trees
- be aware that boosting and bagging can be combined