Lecture 16. Bagging Random Forest and Boosting¶

CS 109A/AC 209A/STAT 121A Data Science:

Harvard University

Fall 2016

Instructors: P. Protopapas, K. Rader, W. Pan


Keyland Lawrence?

Announcements

- HW5 solutions are out
- HW6 is due tonight
- HW7 will be released today

More thinking less implementation

Quiz

Code: deepdarkwoods

Outline

Bagging

Random Forests

Boosting

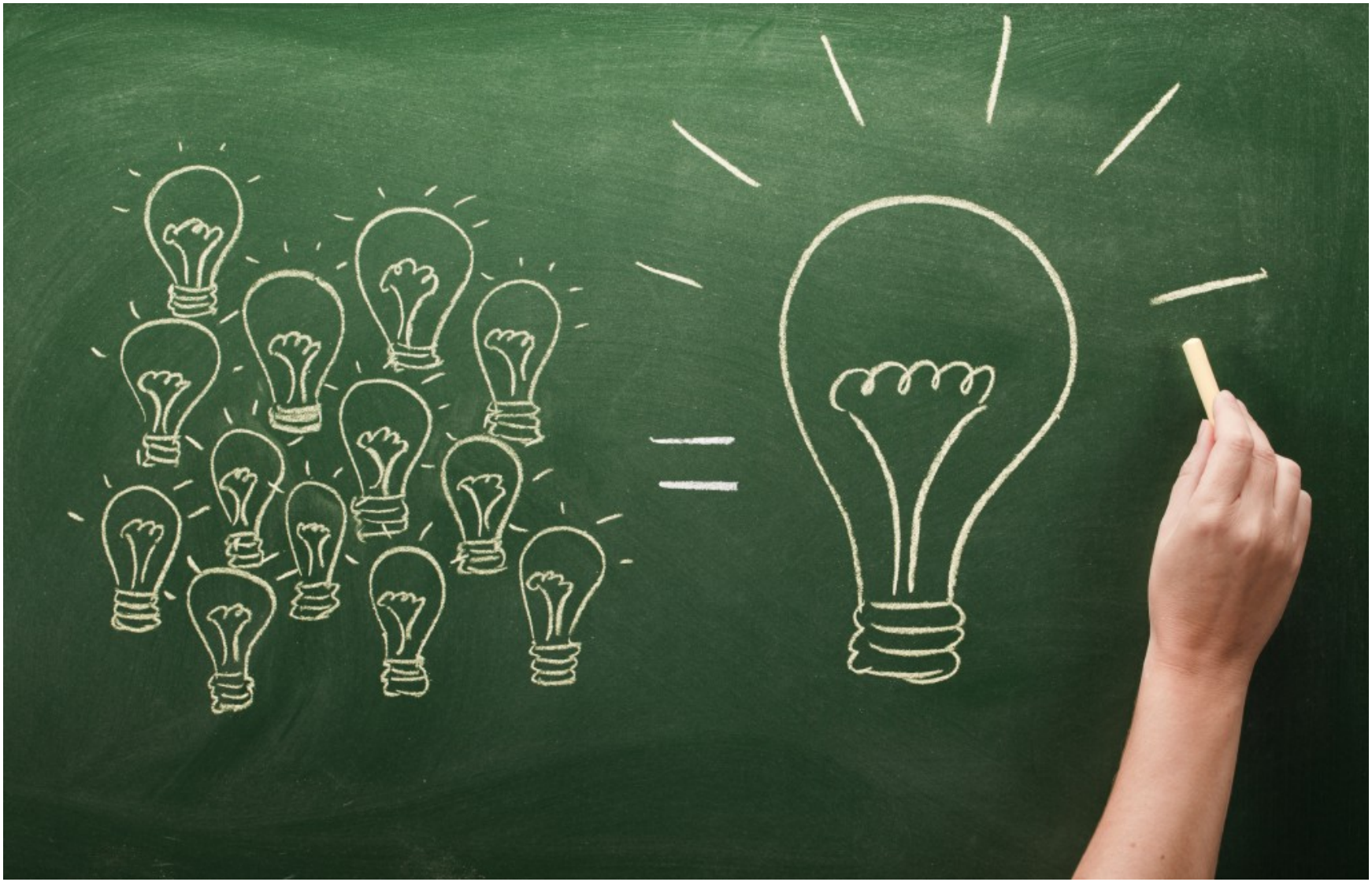
Outline

Bagging

Random Forests

Boosting

Power of the crowds



http://www.scaasymposium.org/portfolio/part-v-the-power-of-innovation-and-the-market/

Ensemble methods

- A single decision tree does not perform well
- But, it is super fast
- What if we learn multiple trees?

We need to make sure they do not all just learn the same

Bagging

If we split the data in random different ways, decision trees give different results, **high variance.**

Bagging: Bootstrap **agg**regat**ing** is a method that result in low variance.

If we had multiple realizations of the data (or multiple samples) we could calculate the predictions multiple times and take the average of the fact that averaging multiple onerous estimations produce less uncertain results

Bagging

Say for each sample *b*, we calculate $f^b(x)$, then:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

How?

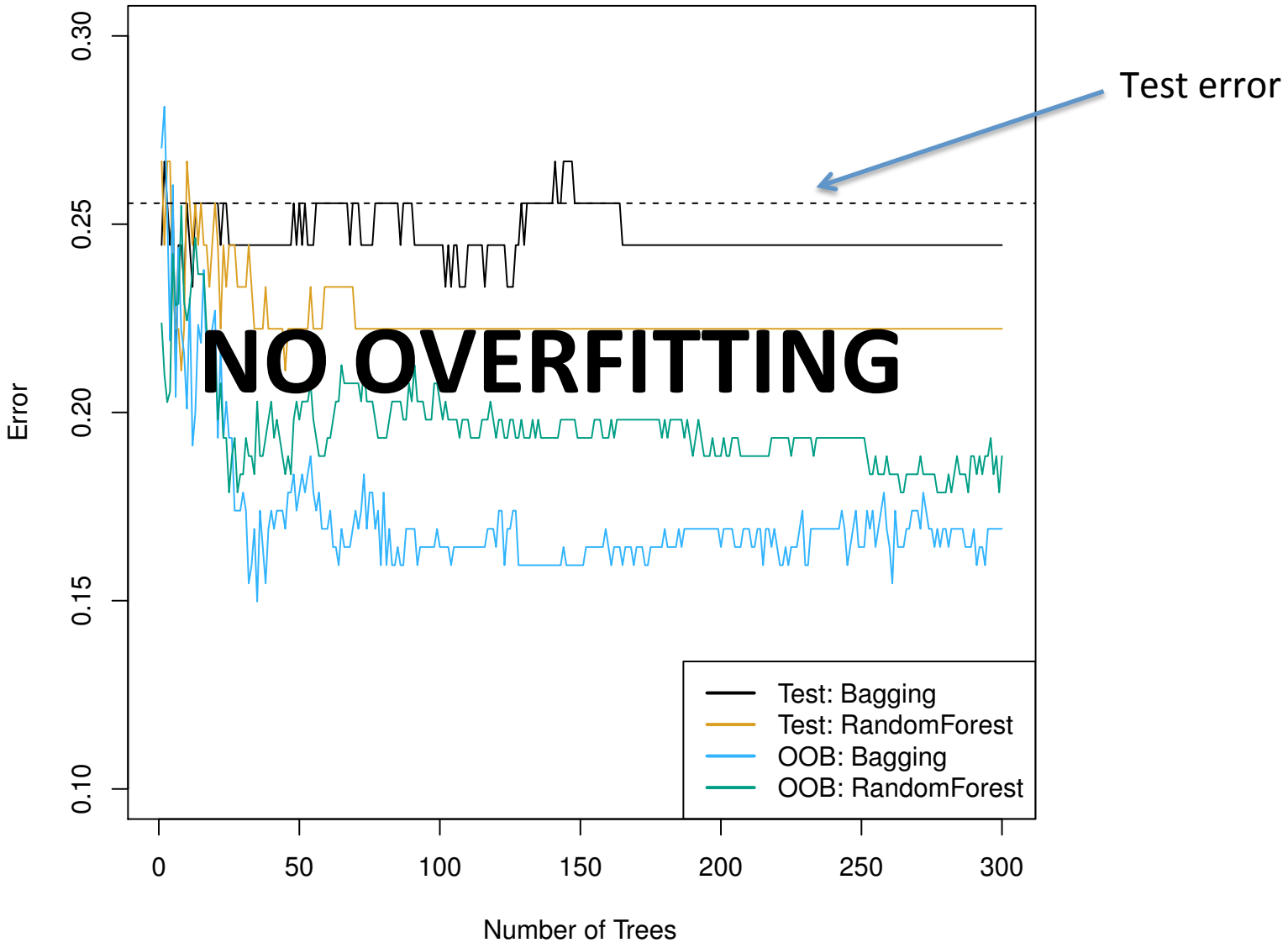
Bootstrap

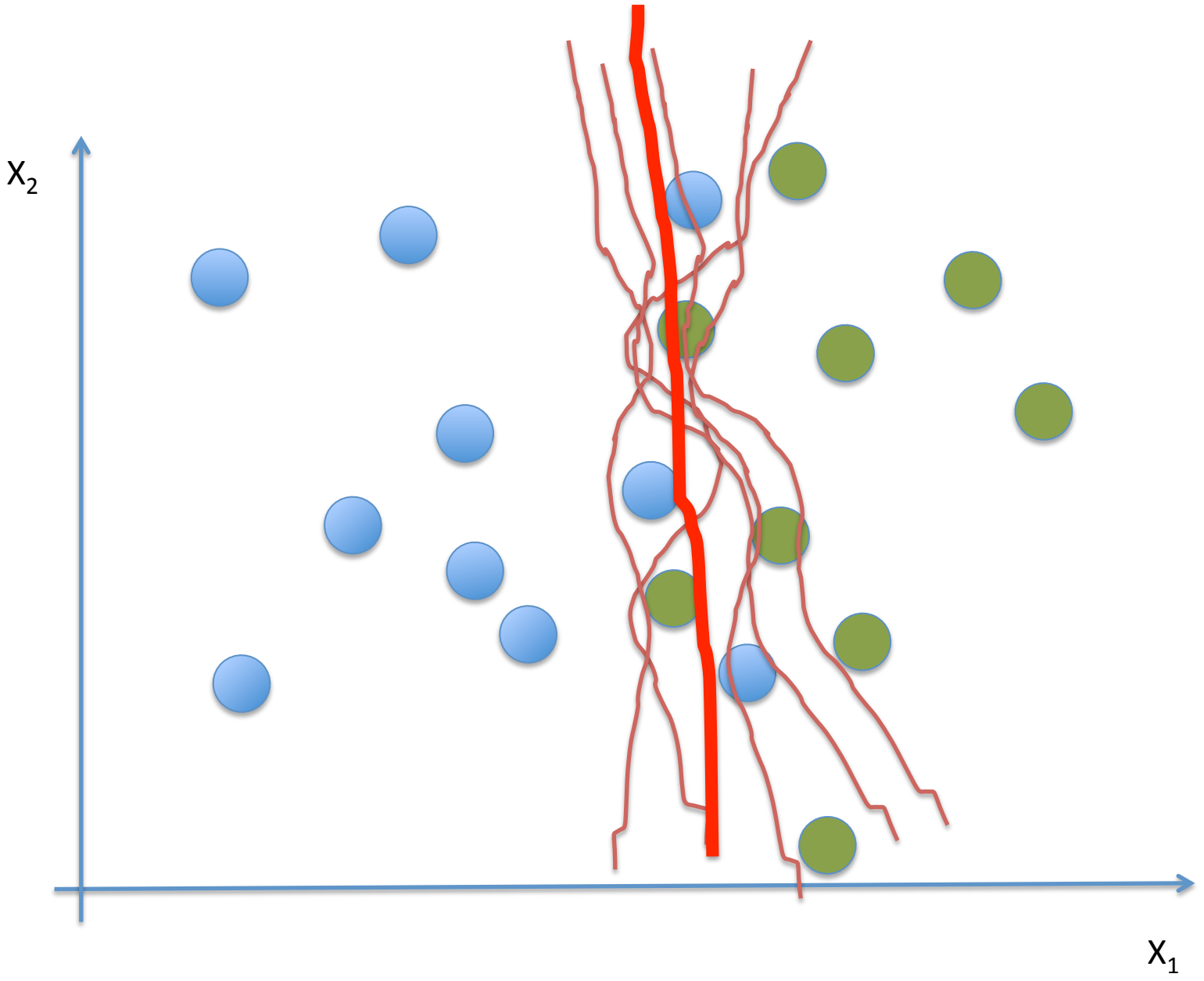
Construct B (hundreds) of trees (no pruning)

Learn a classifier for each bootstrap sample and average them

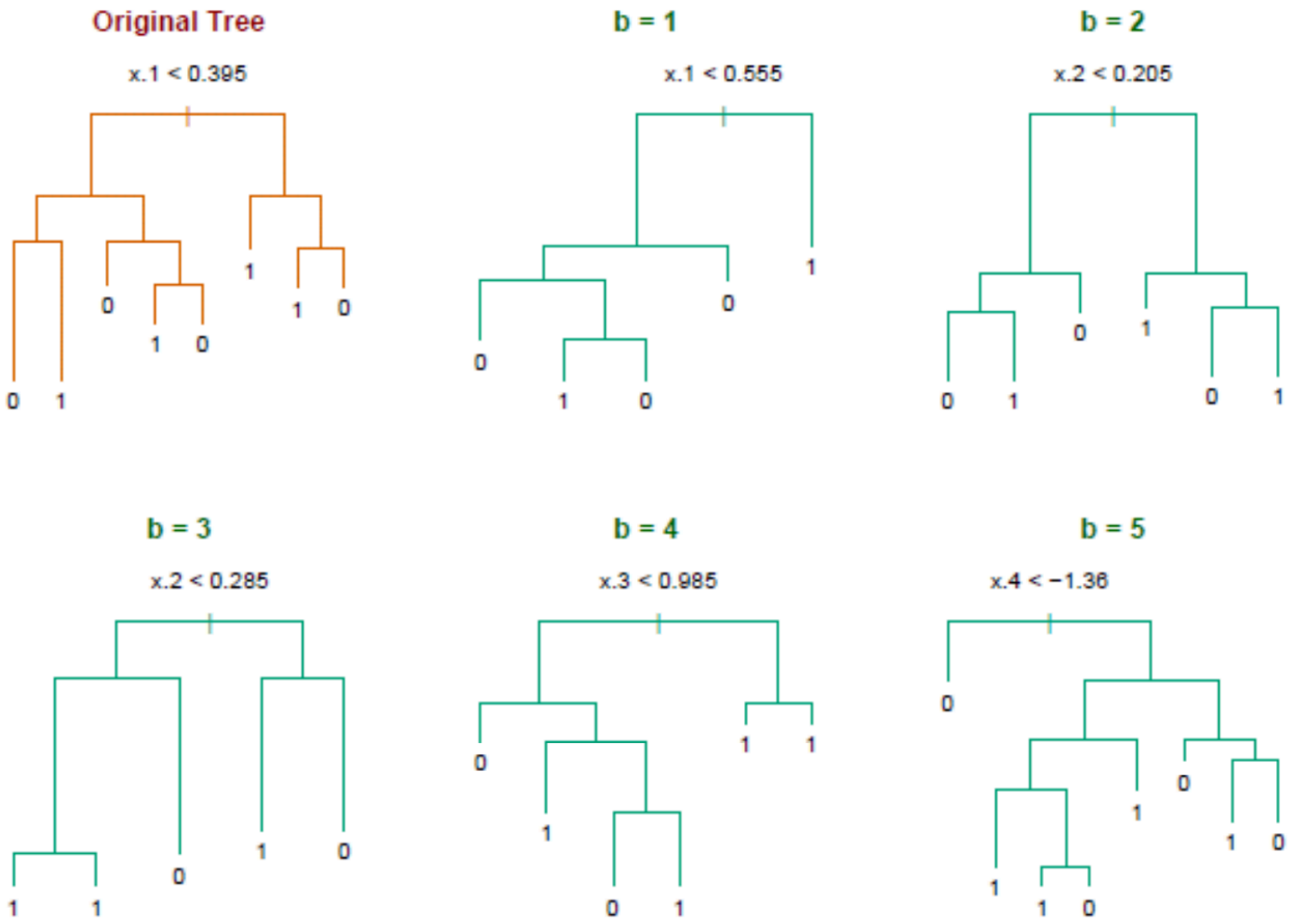
Very effective

Bagging for classification: Majority vote




X_2
X_1

Bagging decision trees



Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

Out-of-Bag Error Estimation

- No cross validation?
- Remember, in bootstrapping we sample with replacement, and therefore **not all observations are used for each bootstrap sample**. On average 1/3 of them are not used!
- We call them out-of-bag samples (OOB)
- We can predict the response for the *i-th* observation using each of the trees in which that observation was OOB and do this for *n* observations
- Calculate overall OOB MSE or classification error

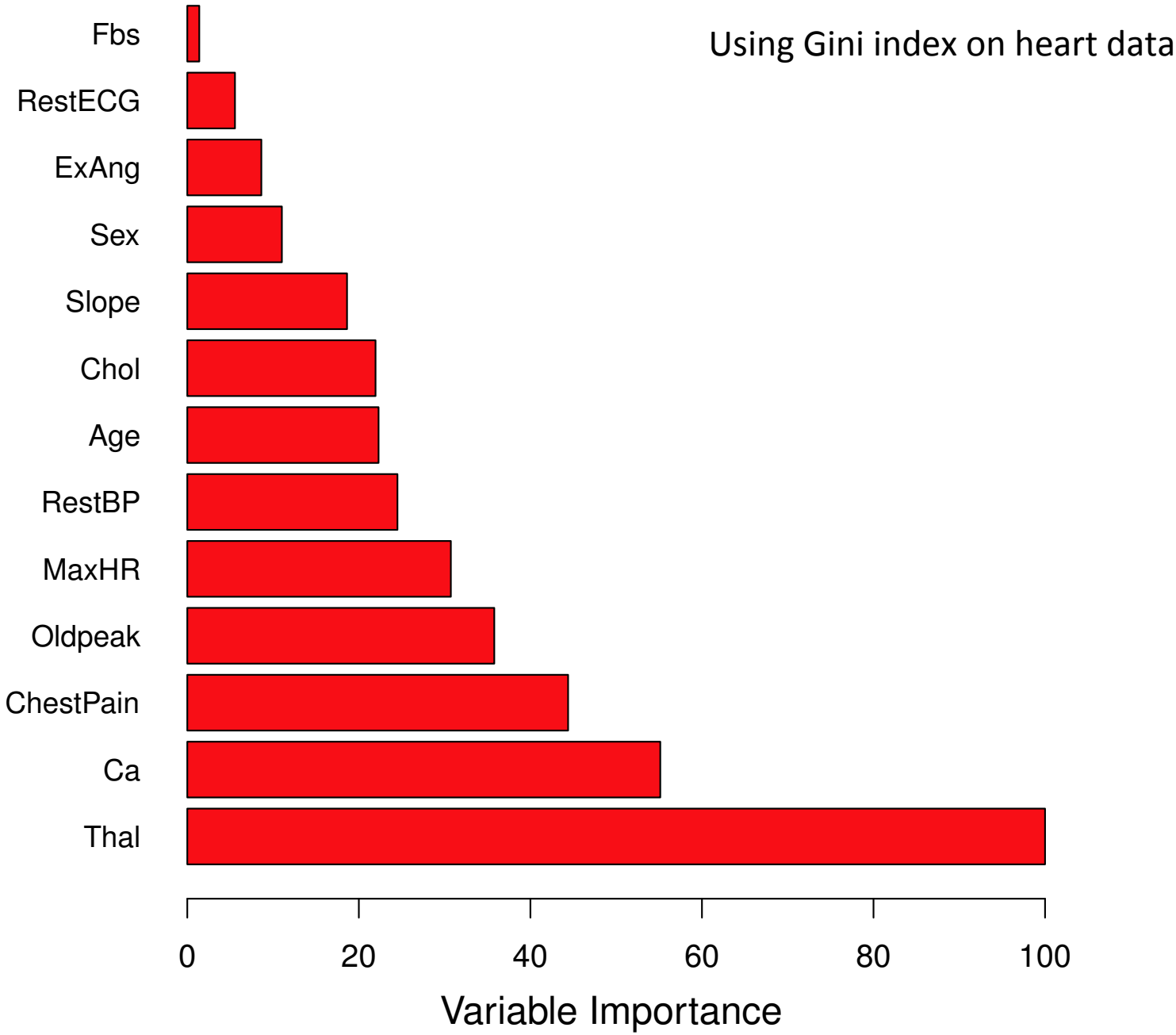
Bagging

- Reduces overfitting (variance)
- Normally uses one type of classifier
- Decision trees are popular
- Easy to parallelize

Variable Importance Measures

- Bagging results in improved accuracy over prediction using a single tree
- Unfortunately, difficult to interpret the resulting model. Bagging improves prediction accuracy at the expense of interpretability.

Calculate the total amount that the RSS or Gini index is decreased due to splits over a given predictor, averaged over all B trees.

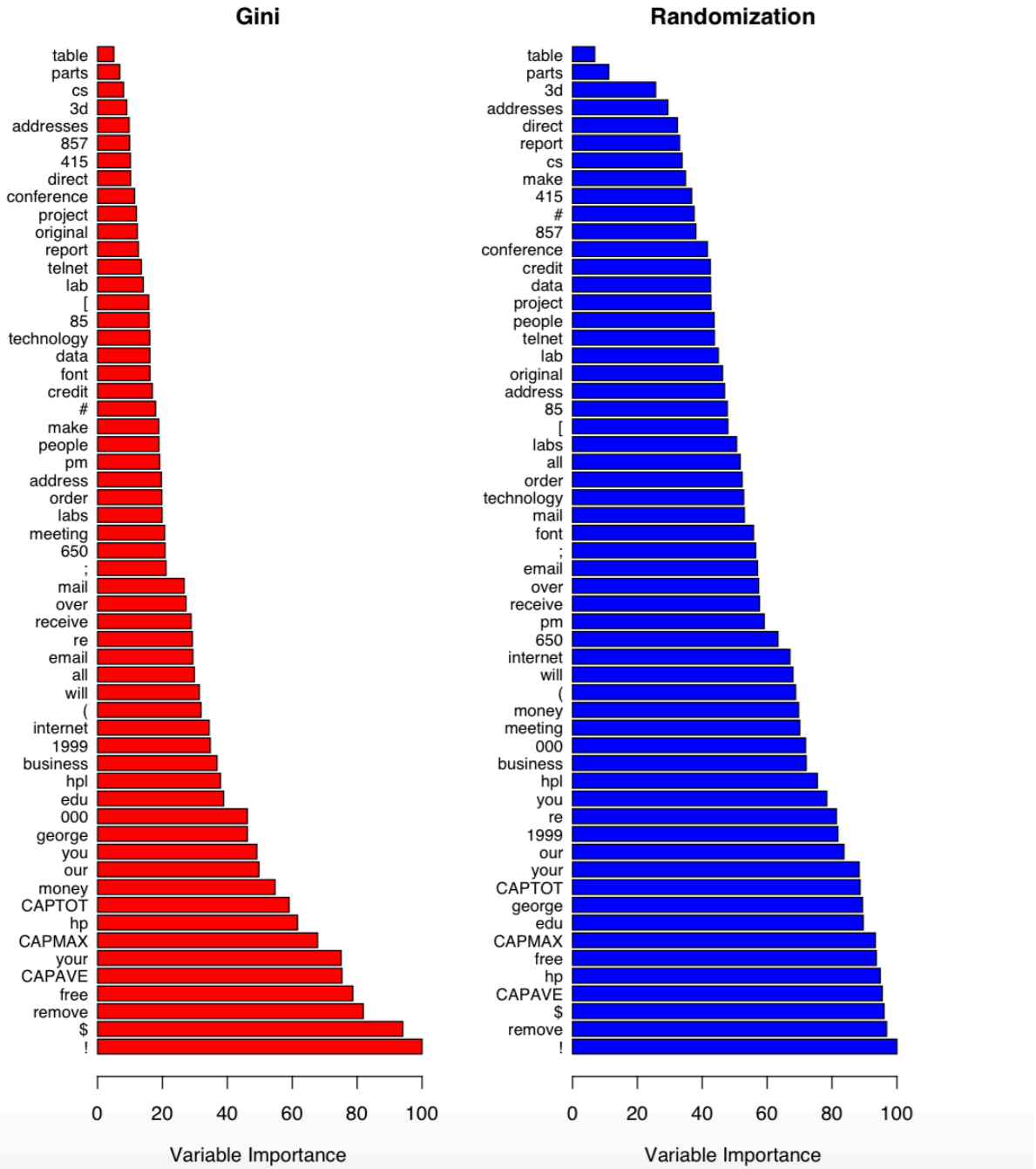

Using Gini index on heart data
Fbs
RestECG
ExAng
Sex
Slope
Chol
Age
RestBP
MaxHR
Oldpeak
ChestPain
Ca
Thal
0
20
40
60
80
100
Variable Importance

RF: Variable Importance Measures

Record the prediction accuracy on the oob samples for each tree

Randomly permute the data for column *j* in the oob samples the record the accuracy again.

The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.


Gini
table
parts
cs
3d
addresses
857
415
direct
conference
project
original
report
telnet
lab
[
85
technology
data
font
credit
#
make
people
pm
address
order
labs
meeting
650
;
mail
over
receive
re
email
all
will
(
internet
1999
business
hpl
edu
000
george
you
our
money
CAPTOT
hp
CAPMAX
your
CAPAVE
free
remove
$
!
0
20
40
60
80
100
Variable Importance
Randomization
table
parts
3d
addresses
direct
report
cs
make
415
#
857
conference
credit
data
project
people
telnet
lab
original
address
85
[
labs
all
order
technology
mail
font
;
email
over
receive
pm
650
internet
will
(
money
meeting
000
business
hpl
you
re
1999
our
your
CAPTOT
george
edu
CAPMAX
free
hp
CAPAVE
$
remove
!
0
20
40
60
80
100
Variable Importance

Bagging - issues

Each tree is identically distributed (i.d.)

➔ the expectation of the average of *B* such trees is the same as the expectation of any one of them

➔the bias of bagged trees is the same as that of the individual trees

i.d. and not i.i.d

Bagging - issues

An average of *B* i.i.d. random variables, each with variance σ^2, has variance*:* σ^2/B

If i.d. (identical but not independent) and pair correlation ρ is present, then the variance is:

$$\rho\,\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As *B* increases the second term disappears but the first term remains

Why does bagging generate correlated trees?

Bagging - issues

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

Then all bagged trees will select the strong predictor at the top of the tree and therefore all trees will look similar.

How do we avoid this?

Bagging - issues

We can penalize the splitting (like in pruning) with a pena**NO THE SAME BIAS**n the number of times a predictor is selected at a given length

We can res**NO THE SAME BIAS**redictor can be used

We only allo**NO THE SAME BIAS**edictors

Bagging - issues

Remember we want i.i.d such as the bias to be the same and variance to be less?

Other ideas?

What if we consider only a subset of the predictors at each split?

We will still get correlated trees unless ….

we **randomly** select the subset !


Random Forests

Outline

Bagging

Random Forests

Boosting

Random Forests

As in bagging, we build a number of decision trees on bootstrapped training samples each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates from the full set of p predictors.

Note that if $m = p$, then this is bagging.

Random Forests

Random forests are popular. Leo Breiman's and Adele Cutler maintains a random forest website where the software is freely available, and of course it is included in every ML/STAT package

http://www.stat.berkeley.edu/~breiman/RandomForests/

Random Forests Algorithm

For b = 1 to B:

(a) Draw a bootstrap sample Z* of size *N* from the training data.

(b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $\boldsymbol{n_{min}}$ is reached.

i. Select ***m*** variables at random from the *p* variables.

ii. Pick the best variable/split-point among the *m*.

iii. Split the node into two daughter nodes.

Output the ensemble of trees.

To make a prediction at a new point *x* we do:

For regression: average the results

For classification: majority vote

Random Forests Tuning

The inventors make the following recommendations:

- For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

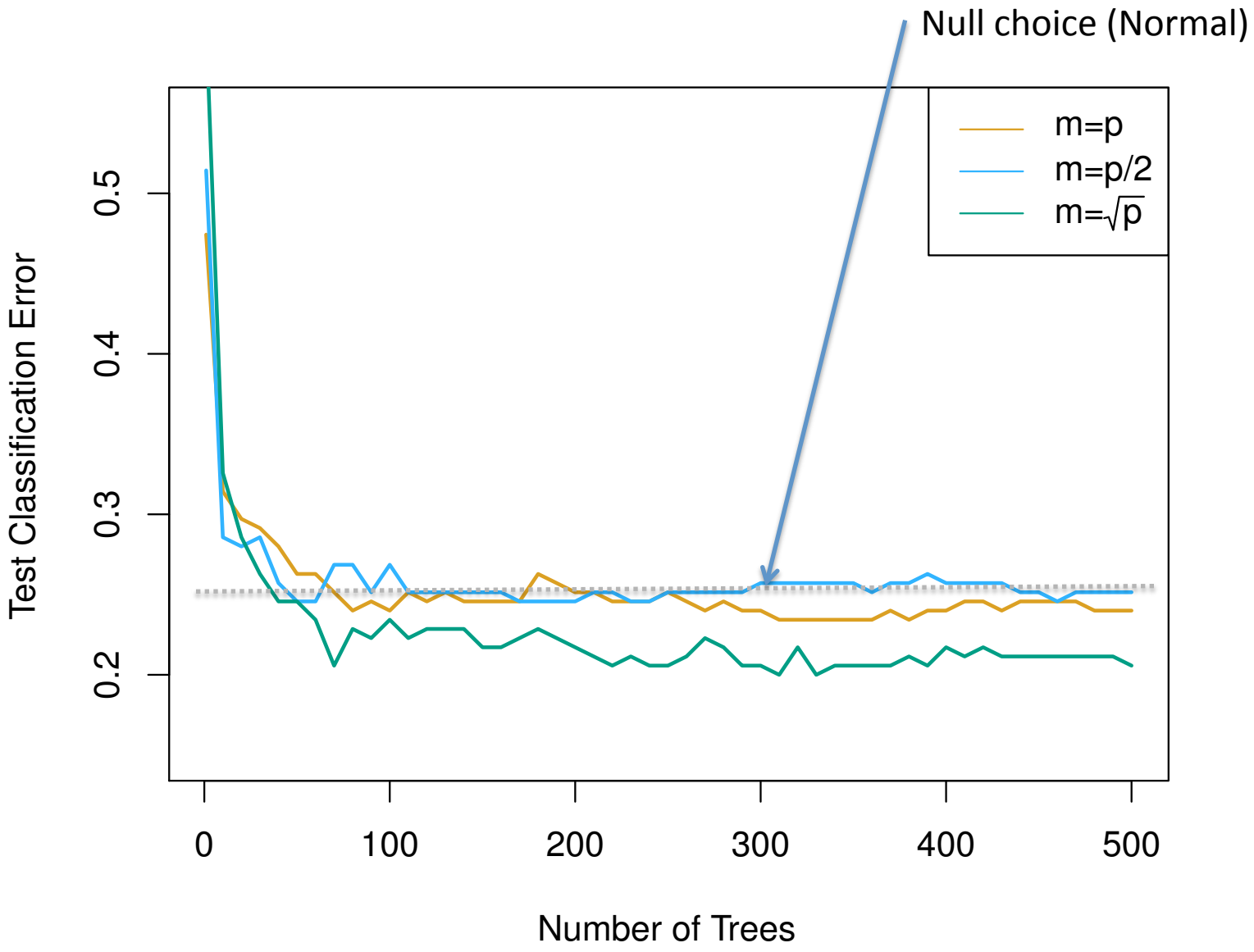
In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Like with Bagging, we can use OOB and therefore RF can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Example

- 4,718 genes measured on tissue samples from 349 patients.
- Each gene has different expression
- Each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.


Null choice (Normal)
m=p
m=p/2
m=√p
Test Classification Error
0.2
0.3
0.4
0.5
0
100
200
300
400
500
Number of Trees

Random Forests Issues

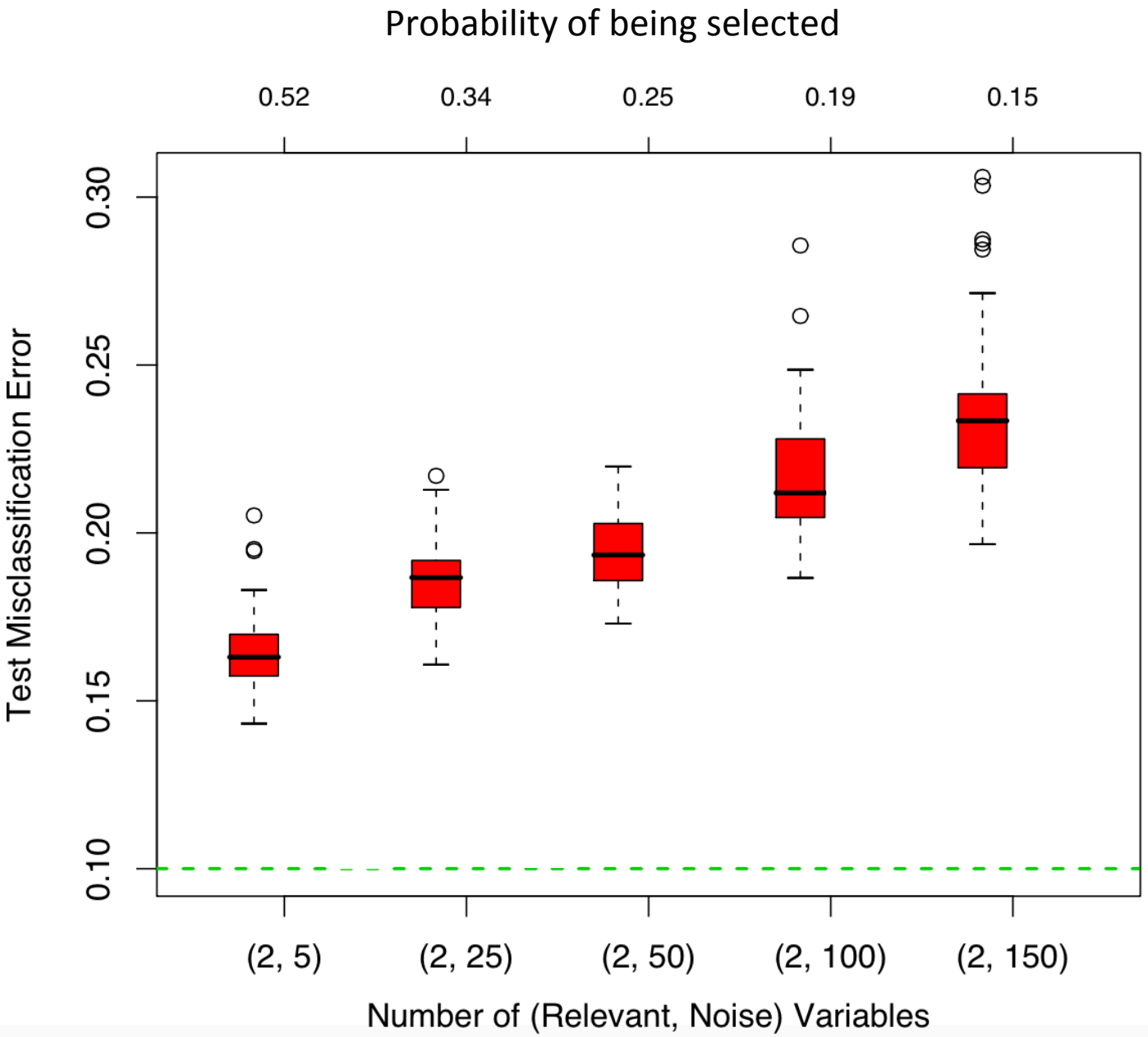
When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when *m* is small

Why?

Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25


Probability of being selected
0.52
0.34
0.25
0.19
0.15
Test Misclassification Error
0.10
0.15
0.20
0.25
0.30
(2, 5)
(2, 25)
(2, 50)
(2, 100)
(2, 150)
Number of (Relevant, Noise) Variables

Can RF overfit?

Random forests "cannot overfit" the data wrt to number of trees.

Why?

The number of trees, *B* does not mean increase in the flexibility of the model

I have seen discussion about gains in performance by controlling the depths of the individual trees grown in random forests. I usually use full-grown trees and seldom it costs much (in the classification error) and results in one less tuning parameter.

Outline

Bagging

Random Forests

Boosting

Boosting

Boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

Bagging: Generate multiple trees from bootstrapped data and average the trees.

Recall bagging results in i.d. trees and not i.i.d.

RF produces i.i.d (or more independent) trees by randomly selecting a subset of predictors at each step

Boosting

Boosting works very differently.

1. Boosting does not involve bootstrap sampling
2. Trees are grown sequentially: each tree is grown using information from previously grown trees
3. Like bagging, boosting involves combining a large number of decision trees, $f^1, \ldots, f^B$

Sequential fitting

Given the current model,

- we fit a decision tree to the **residuals** from the model. Response variable now is the residuals and not Y
- We then add this new decision tree into the fitted function in order to update the residuals
- The learning rate has to be controlled

Boosting for regression

1. Set *f(x)=0* and $r_i = y_i$ for all *i* in the training set.

2. For *b=1,2,...,B*, repeat:

   a. Fit a tree with d splits(+1 terminal nodes) to the training data (X, r).

   b. Update the tree by adding in a shrunken version of the new tree:

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

   c. Update the residuals,

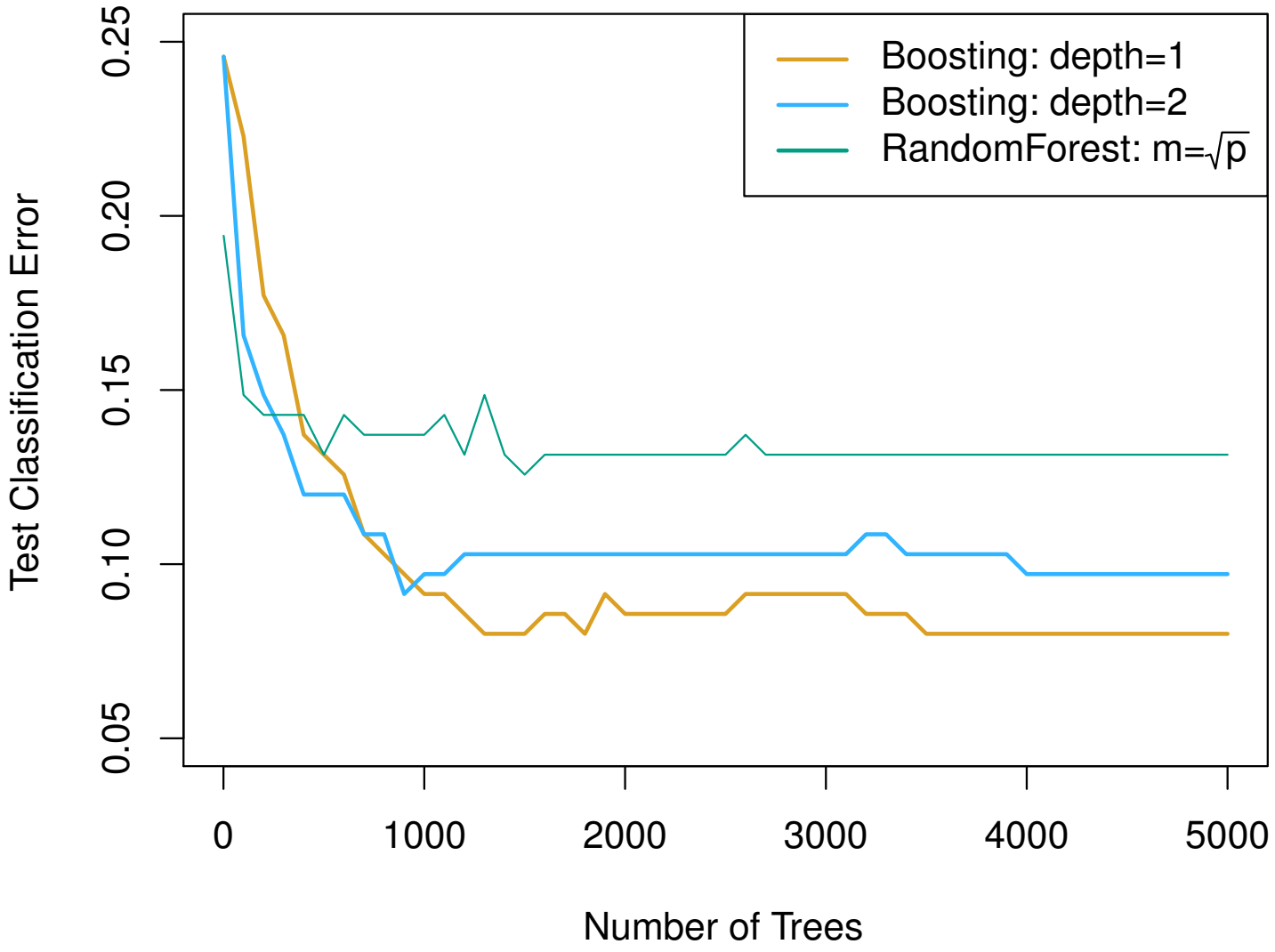
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Output the boosted model,

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

Boosting tuning parameters

- The number of trees B. RF and Bagging do not overfit as B increases. Boosting can overfit! **Cross Validation**
- The shrinkage parameter λ, a small positive number. Typical values are 0.01 or 0.001 but it depends on the problem. λ only controls the learning rate
- The number d of splits in each tree, which controls the complexity of the boosted ensemble. Stumpy trees, d = 1 works well.


Boosting: depth=1
Boosting: depth=2
RandomForest: m=√p
Test Classification Error
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Number of Trees

Boosting for classification

Challenge question for HW7

Different flavors

- ID3, or alternative Dichotomizer, was the first of three Decision Tree implementations developed by Ross Quinlan (Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.) Only categorical predictors and no pruning.
- C4.5, Quinlan's next iteration. The new features (versus ID3) are: (i) accepts both continuous and discrete features; (ii) handles incomplete data points; (iii) solves over-fitting problem by (very clever) bottom-up technique usually known as "pruning"; and (iv) different weights can be applied the features that comprise the training data.

Used in orange http://orange.biolab.si/

Different flavors

- C5.0, The most significant feature unique to C5.0 is a scheme for deriving rule sets. After a tree is grown, the splitting rules that define the terminal nodes can sometimes be simplified: that is, one or more condition can be dropped without changing the subset of observations that fall in the node.

- CART or Classification And Regression Trees is often used as a generic acronym for the term Decision Tree, though it apparently has a more specific meaning. In sum, the CART implementation is very similar to C4.5. **Used in sklearn**

Missing data

- What if we miss predictor values?
  - Remove those examples => depletion of the training set
  - Impute the values either with mean, knn, from the marginal or joint distributions
- Trees have a nicer way of doing this
  - Categorical

Further reading

- Pattern Recognition and Machine Learning,
  Christopher M. Bishop
- The Elements of Statistical Learning

  Trevor Hastie, Robert Tibshirani, Jerome Friedman

  http://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf