BAGGING, RANDOM FORESTS, BOOSTING

Chapter 08 (part 02)

Outline

- Bagging
  - Bootstrapping
  - Bagging for Regression Trees
  - Bagging for Classification Trees
  - Out-of-Bag Error Estimation
  - Variable Importance: Relative Influence Plots
- Random Forests
- Boosting

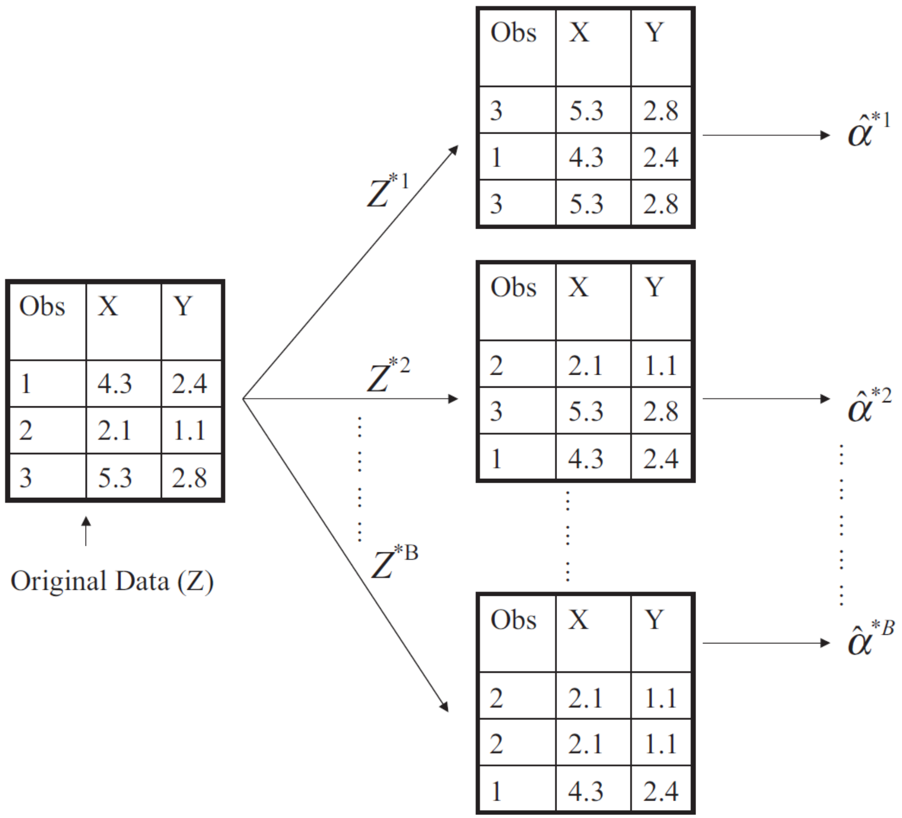
BAGGING
(Bootstrap AGGregatING)

Problem with Decision Trees

- (basic) Decision trees suffer from high variance
  - If we randomly split the training data into 2 parts, and fit separate decision trees on both parts, the results could be quite different
- We would like to have models with low variance
- To solve this problem, we can use bagging (***b***ootstrap ***agg***regat***ing***).
- Bagging is a general process for machine learning
  - It could be a preprocessing step for any model

Bootstrapping Revisited

- Resample the observed dataset to obtain a new set of data equal to the size to the observed dataset
- Each new observation is obtained by random sampling with replacement from the original dataset.



What is bagging?

- Bagging is an extremely powerful idea based on two things:
  - Averaging: reduce variance
  - Bootstrapping: generate many alternative training datasets
- Why does averaging reduce variance?
  - Averaging a set of observations reduces variance. Recall that given a set of n independent observations Z_1, …, Z_n, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n

How does bagging training work?

1. Generate B different bootstrapped training datasets
2. Train the model separately on each of the B training datasets, to obtain B different models.

Bagging for Regression Trees

- Construct B regression trees using B bootstrapped training datasets. Do not prune the trees
- Average (or vote) to determine the resulting predictions

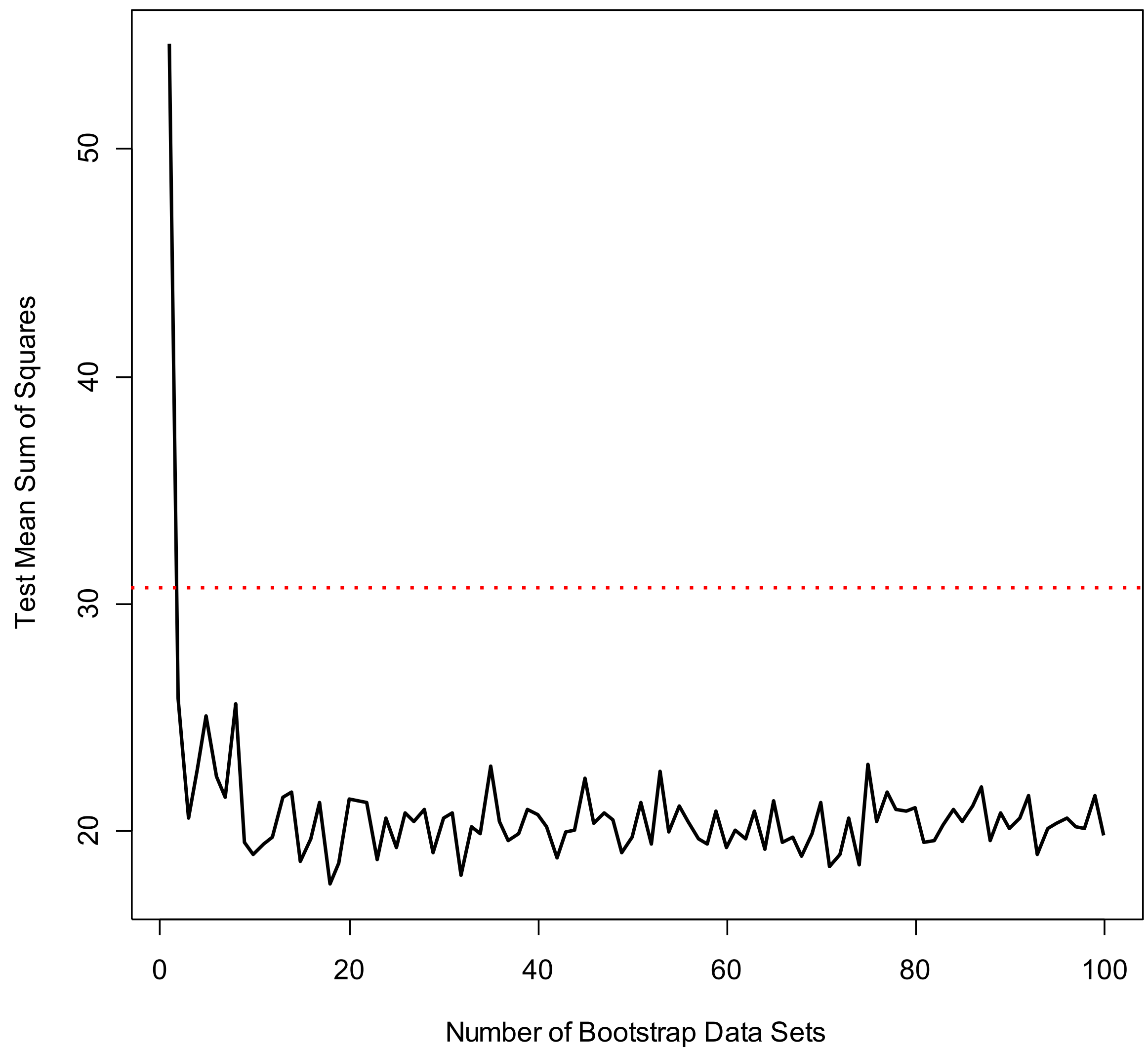
Bagging for Classification Trees

- Construct B regression trees using B bootstrapped training datasets
- For prediction, there are two approaches:
    1. Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote).
    2. If our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability.
- Both methods work well

Example 1: Housing Data (Regression)

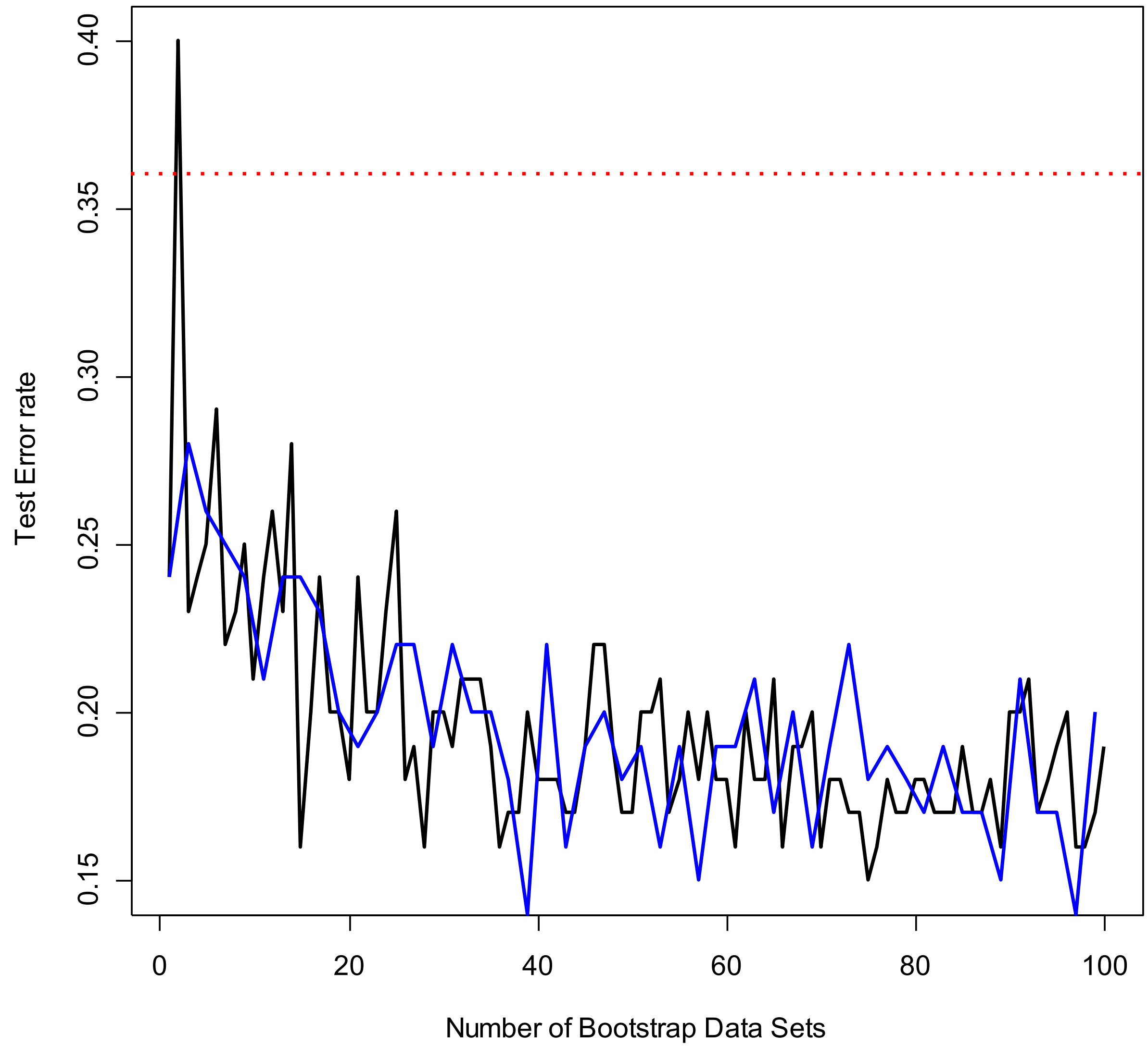
- Red line:
  test mean sum of squares using a **single tree**.

- Black line:
  bagging test error rate



Example 2: Car Seat Data (Classification)

- The red line represents the test error rate using a single tree.
- The black line corresponds to the bagging error rate using majority vote
- The blue line averages the class probabilities before deciding class.



Out-of-Bag Error Estimation

- Bootstrapping randomly selects a subset of observations to train each tree
- The remaining non-selected subset could be used to estimate performance on unseen data
- On average, each bagged tree makes use of around 2/3 of the observations, so we end up having 1/3 of the observations used for estimating performance

Variable Importance Measure

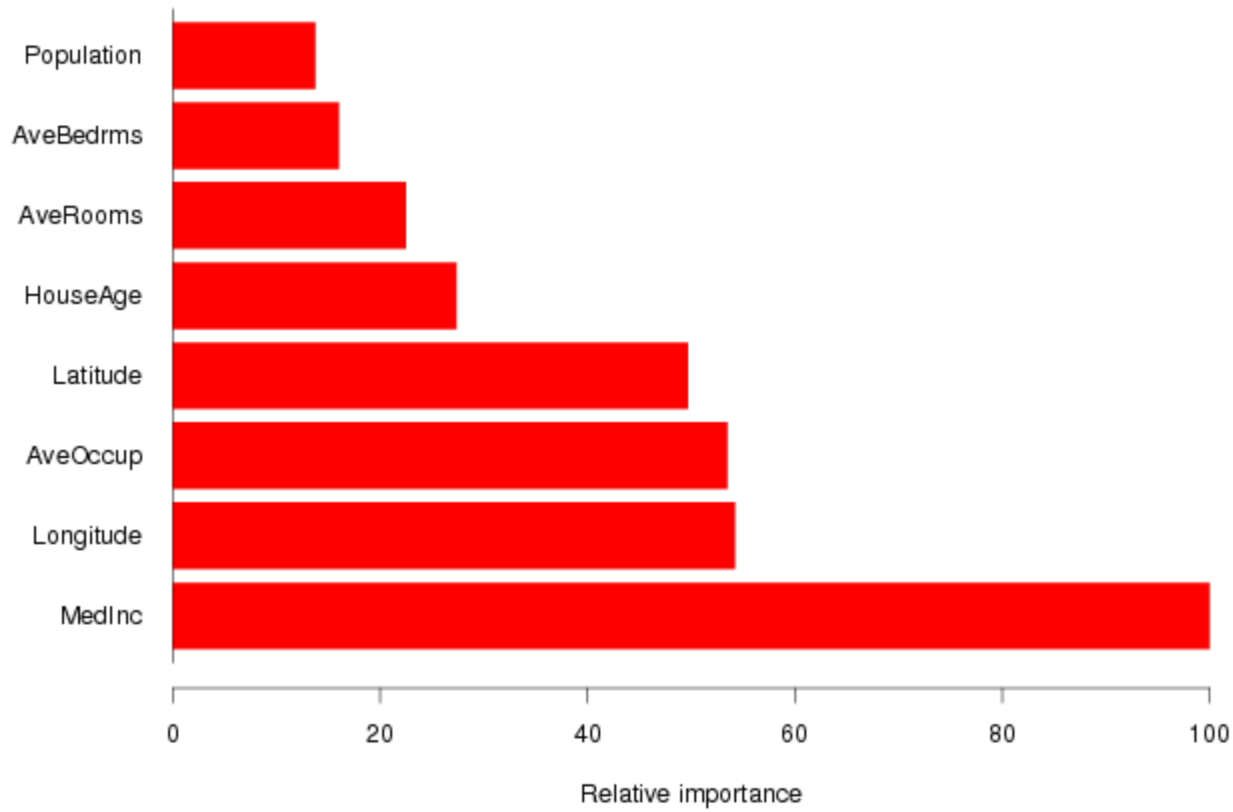
- Bagging typically improves the accuracy over prediction using a single tree, but it is harder to interpret the model
- We have hundreds of trees, and it is no longer clear which variables are most important to the procedure
- Thus bagging improves prediction accuracy at the expense of interpretability
- But, we can still get an overall summary of the importance of each predictor using Relative Influence Plots

Inference with multiple trees

- Which variables are most useful in predicting the response?
  - Relative influence plots give a score for each variable.
  - These scores represent the decrease in MSE (or Gini Index for classification) when splitting on a particular variable
  - The most influential variable is given a value of 100 and other variables are shown as a relative fraction of the influence of the most influential variable
  - The larger the score the more influence the variable has
  - A number close to zero indicates the variable is not important and could be dropped

Example: Housing Data

- Median Income is by far the most important variable.
- Longitude, Latitude and Average occupancy are the next most important.



Problems with Bagging?

- If there is a very strong predictor in the data set along with a number of other moderately strong predictors, then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split
- All bagged trees will look similar. Hence all the predictions from the bagged trees will be **highly correlated**
- Averaging many highly correlated quantities does not lead to a large variance reduction
- If we want to reduce the variance, we need something to do something to break the correlation of outputs

RANDOM FORESTS

Random Forests

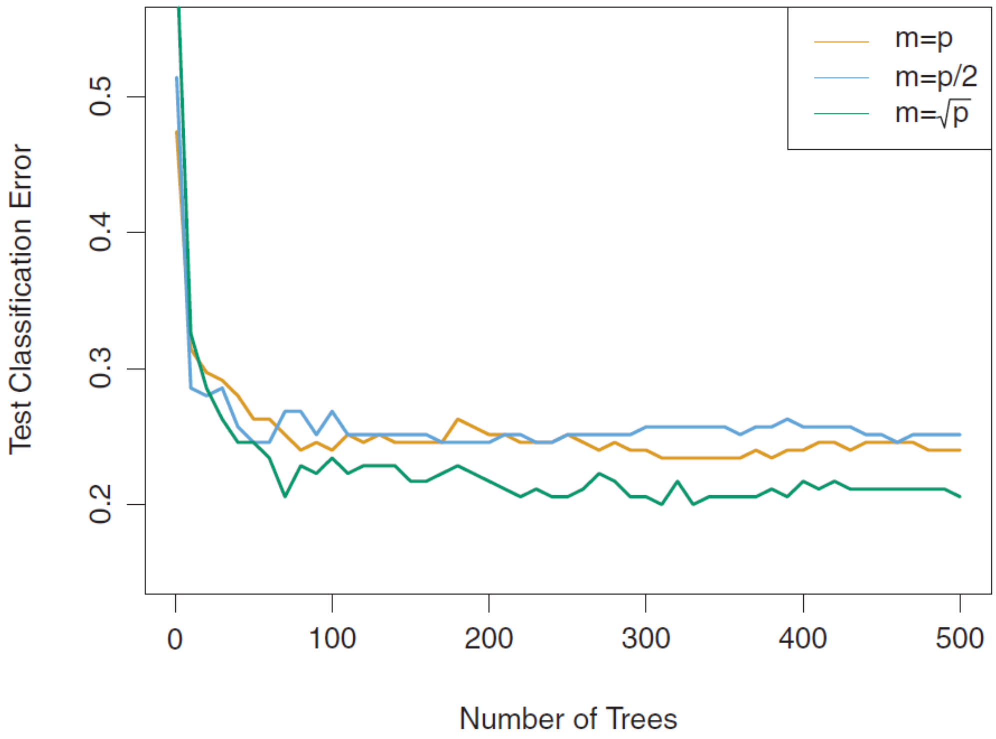
- A very efficient statistical learning method
- Builds on the idea of bagging, but it provides an improvement because it *de-correlates* the trees
- How does it work?
  - Build a number of decision trees on bootstrapped training sample, but when building these trees, each time a split in a tree is considered, a random sample of *m* predictors is chosen as split candidates from the full set of *p* predictors (Usually $m \approx \sqrt{p}$)

Why are we considering a random sample of m predictors instead of all *p* predictors for splitting?

- If each tree has a different subset of *p* predictors, then the trees wont get stuck always picking the same best predictor if one of the *p* predictors was stronger than most others
- This means each tree will give grow differently and a set of trees grown this way will have less correlation in their outputs
- Random forests "de-correlate" the outputs that bagged trees would have generated… leading to more reduction in variance

Random Forest Hyperparameter "m"

- *m* is the number of features randomly selected for use in each tree
- If random forests are set *m* = *p*, this is equivalent to bagging
- Empirically, sqrt(*p*) often works well



Boosting – Random Forest mod

- Idea: build a forest of B trees *incrementally*, but when building the next tree, instead of fitting a model to best predict Y, attempt to best predict the *residual* error remaining in the forest
- Use crossval to determine a good size for the forest (B)

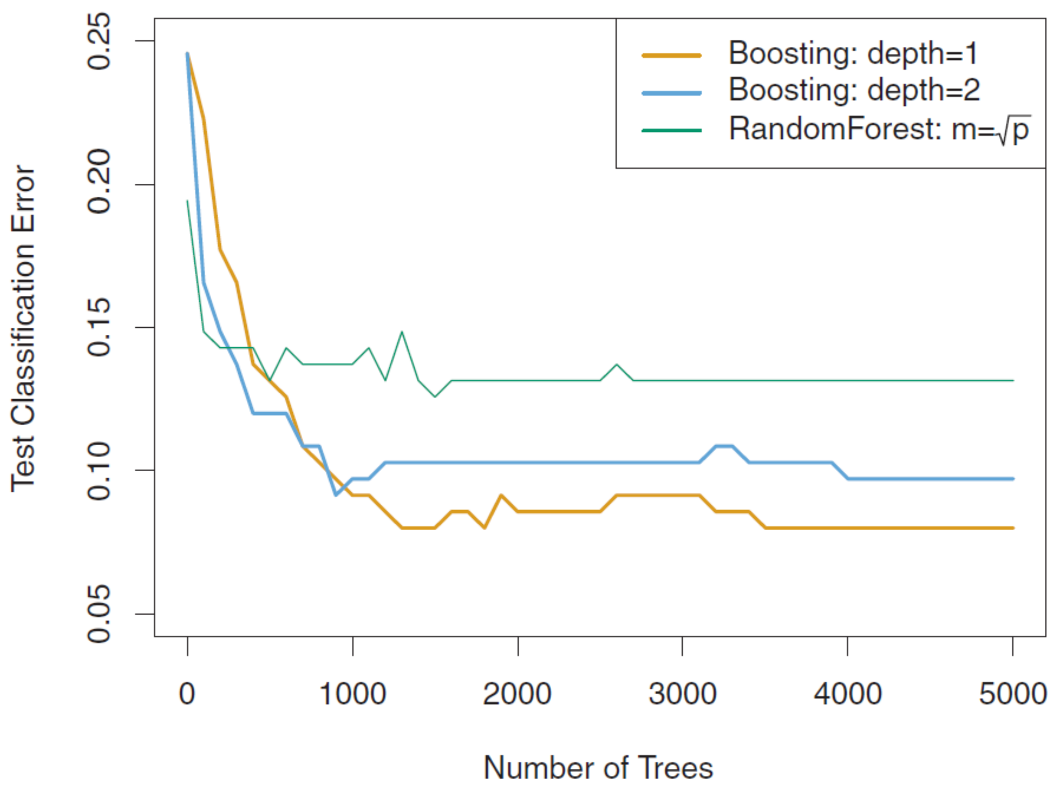
Algorithm 8.2 *Boosting for Regression Trees*

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

This algorithm learns *slowly*
Smaller lambda leads to slower learning… which means lower variance

Boosting

- *d* is the interaction depth
- *d* = 1 gives us a *stump* with two leaf nodes
- Boosted stumps tend to perform well



Trees & Forest Summary

- Trees and Forests offer non-linear machine learning models which can answer both prediction and inference questions
- There are *many* alternative tree & forest model architectures
- Each architecture has a set of hyperparameters to determine
- Cross-validation should be used to make model and hyperparameter decisions… but this increases the computational cost of finding a good model