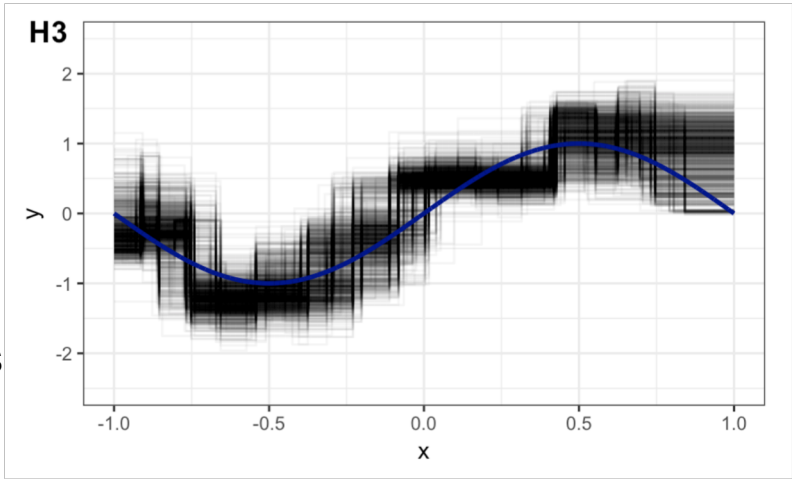
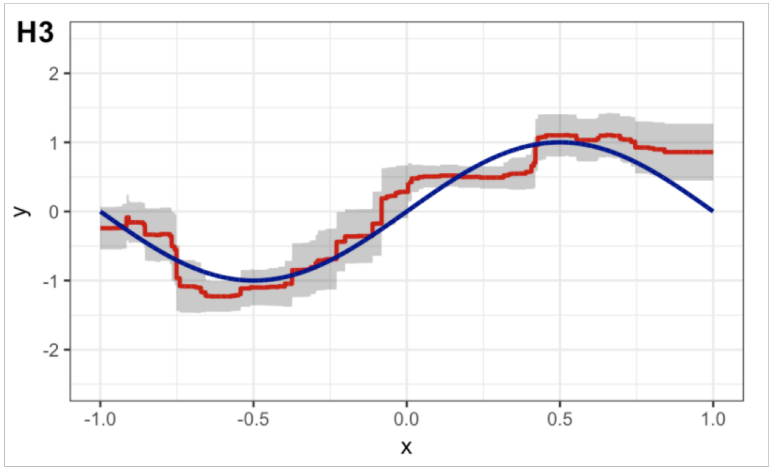
DSC5103 Statistics

Session 9. Boosting

Last time

- The Bootstrap
  - generate bootstrap samples
  - estimate variabilities
- Bagging and Random Forest
  - Grow trees in parallel on bootstrap samples
  - Aggregate predictions
  - Randomize predictors
- Ensemble methods in general

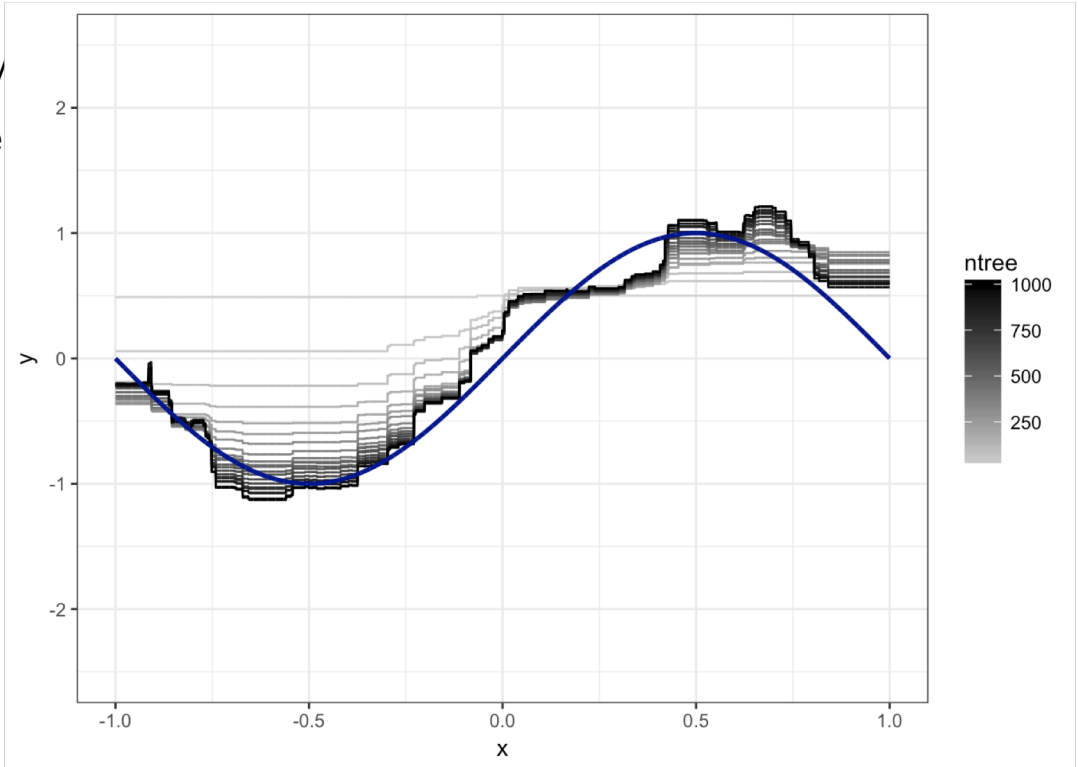




Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods
  - *Iteratively* build weak learners and aggregate them into a final strong learner
  - Weighted in some way that is usually related to the weak learners' accuracy

- Gradient Boosting Machines/M
  - Grow the trees ***sequentially***, e
    previously grown trees



GBM algorithm for regression trees

1. Fit the first tree $T^1(X)$ with d splits to the data (X, Y)
   - use $T^1(X)$ as partial prediction: $f^1(X) = \lambda * T^1(X)$
   - The residual is $R^1 = Y - f^1(X)$

2. For b = 2, …, B:

   1. Fit the b-th tree $T^b(X)$ with d splits to the data (X, R^{b-1})

   2. Update prediction: $f^b(X) = f^{b-1}(X) + \lambda * T^b(X)$

   3. Update residual $R^b = R^{b-1} - \lambda * T^b(X)$

3. Final prediction is $f^B(X)$

Key Parameters:

- Number of splits ***d***
- Number of trees ***B***
- Learning rate $\boldsymbol{\lambda}$

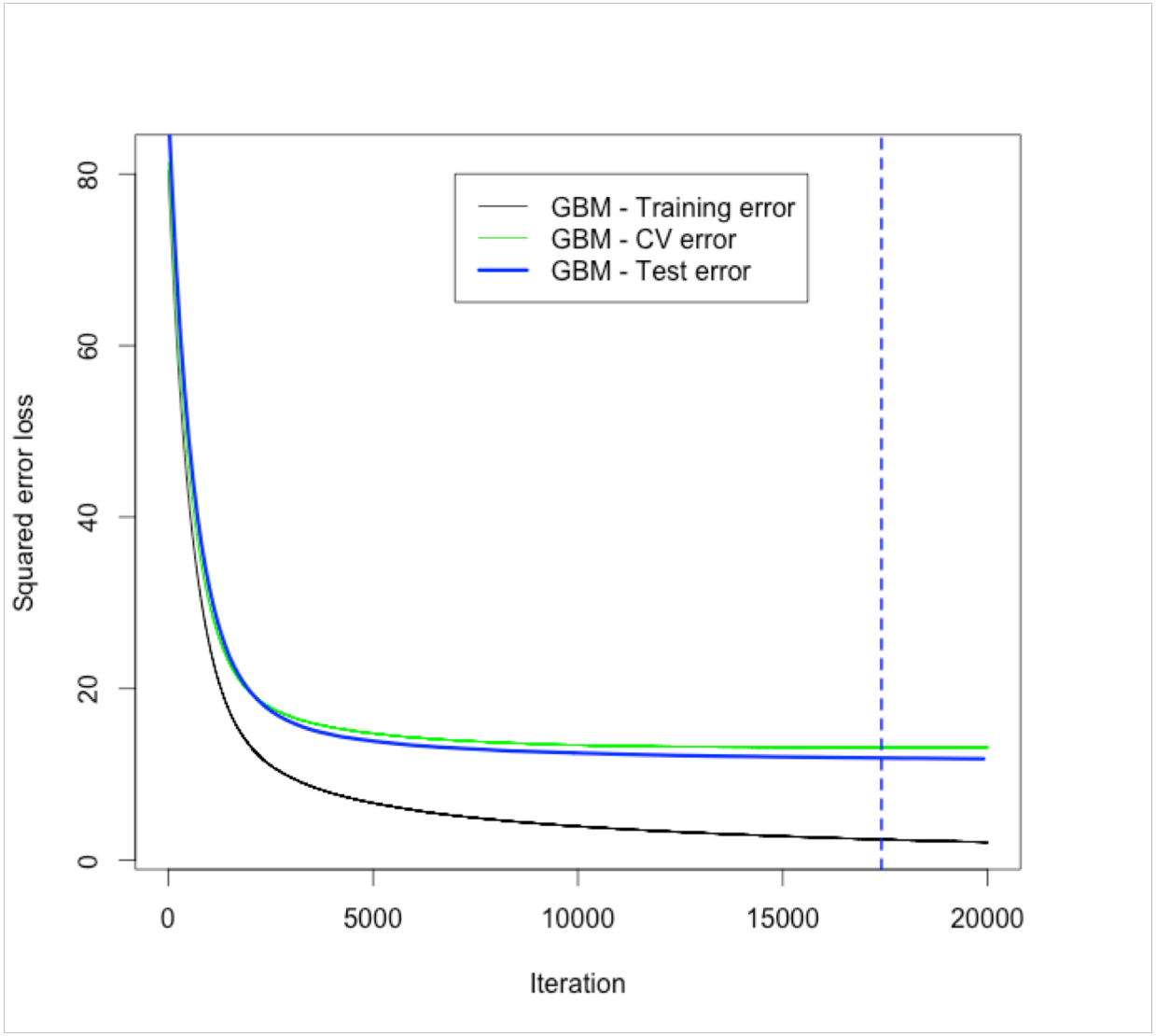
The Idea Behind

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns sequentially with *many* trees.

- Each of these trees can be rather small (*weak learners*), with just a few terminal nodes.

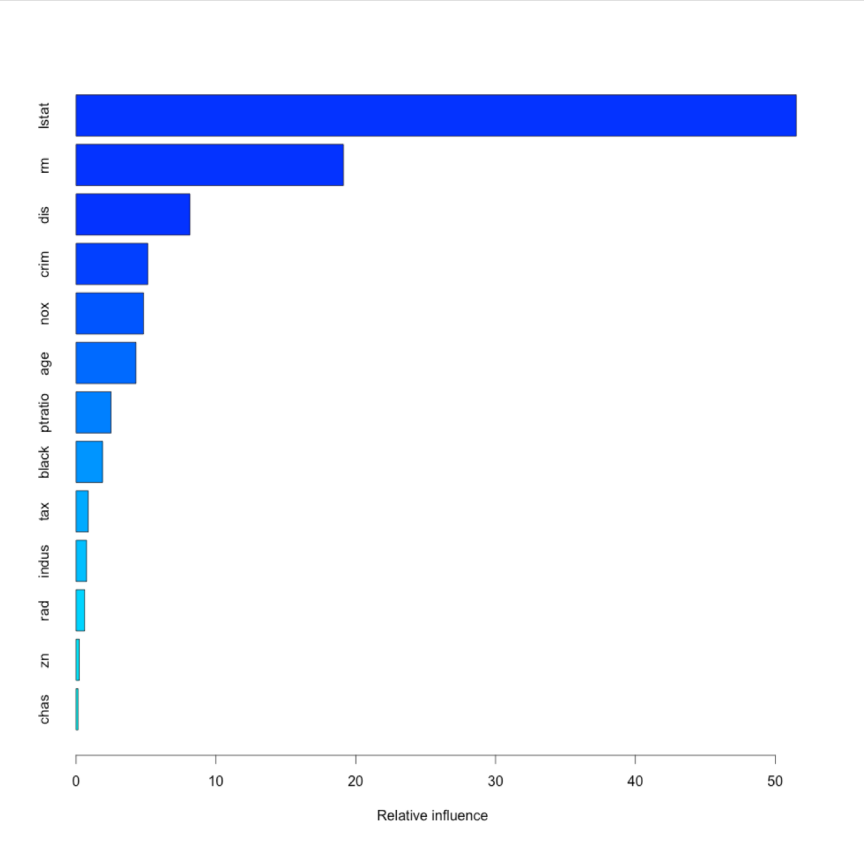
- Given the current model, we fit a small decision tree to the residuals from the model. We slowly improve f() in areas where it does not perform well. The *shrinkage* parameter slows the process down even further, allowing more and different shaped trees to attack the residuals.

Example: Boston Housing Data



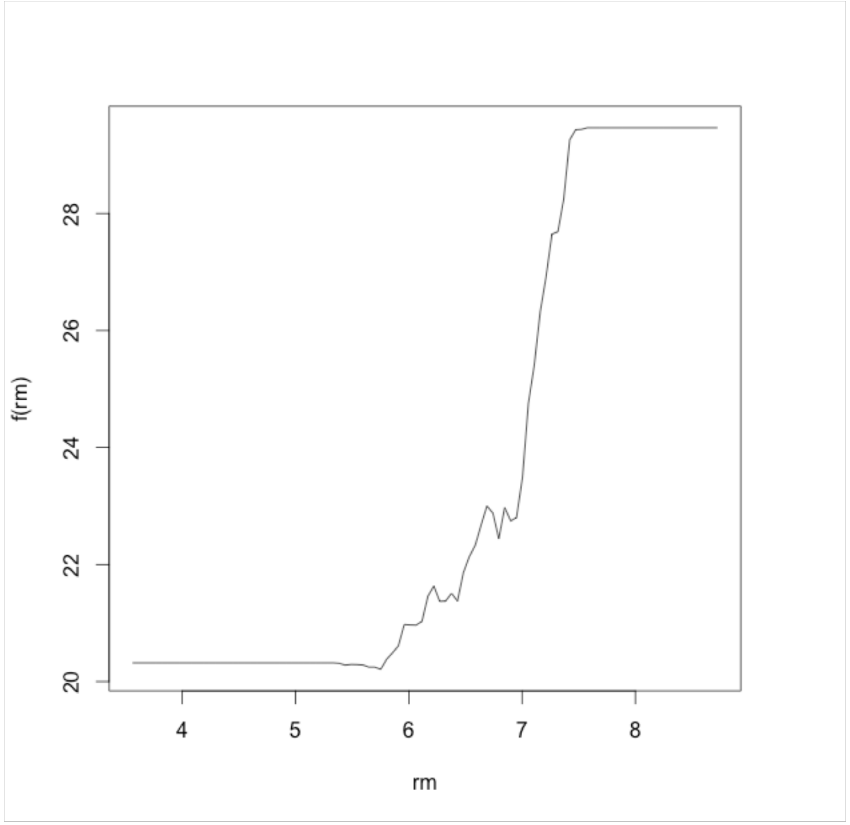
Example: Boston Housing Data

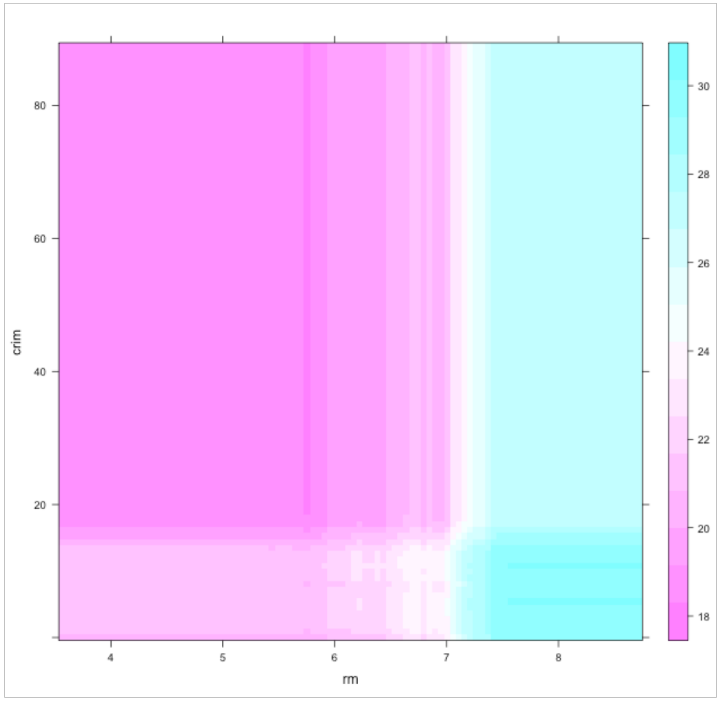
- (relative) variable importance plots



Example: Boston Housing Data

- Partial plots (1-D or 2-D)





Tuning parameters for GBM

- The number of trees B (*nrounds*)
  - Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all.

- The shrinkage parameter λ (*eta*)
  - A small positive number controls the rate at which boosting learns.
  - Typical values are 0.01 or 0.001, and the right choice can depend on the problem.
  - Very small λ can require using a very large value of B in order to achieve good performance.

- The number of splits d in each tree (*max.depth*)
  - Parameter d controls the complexity of the trees (number of nodes).
  - It controls the interaction order, since d splits can involve at most d variables.
  - When $d = 1$, each tree is a "stump" consisting of a single split and resulting in an additive model.

Tuning parameters for GBM

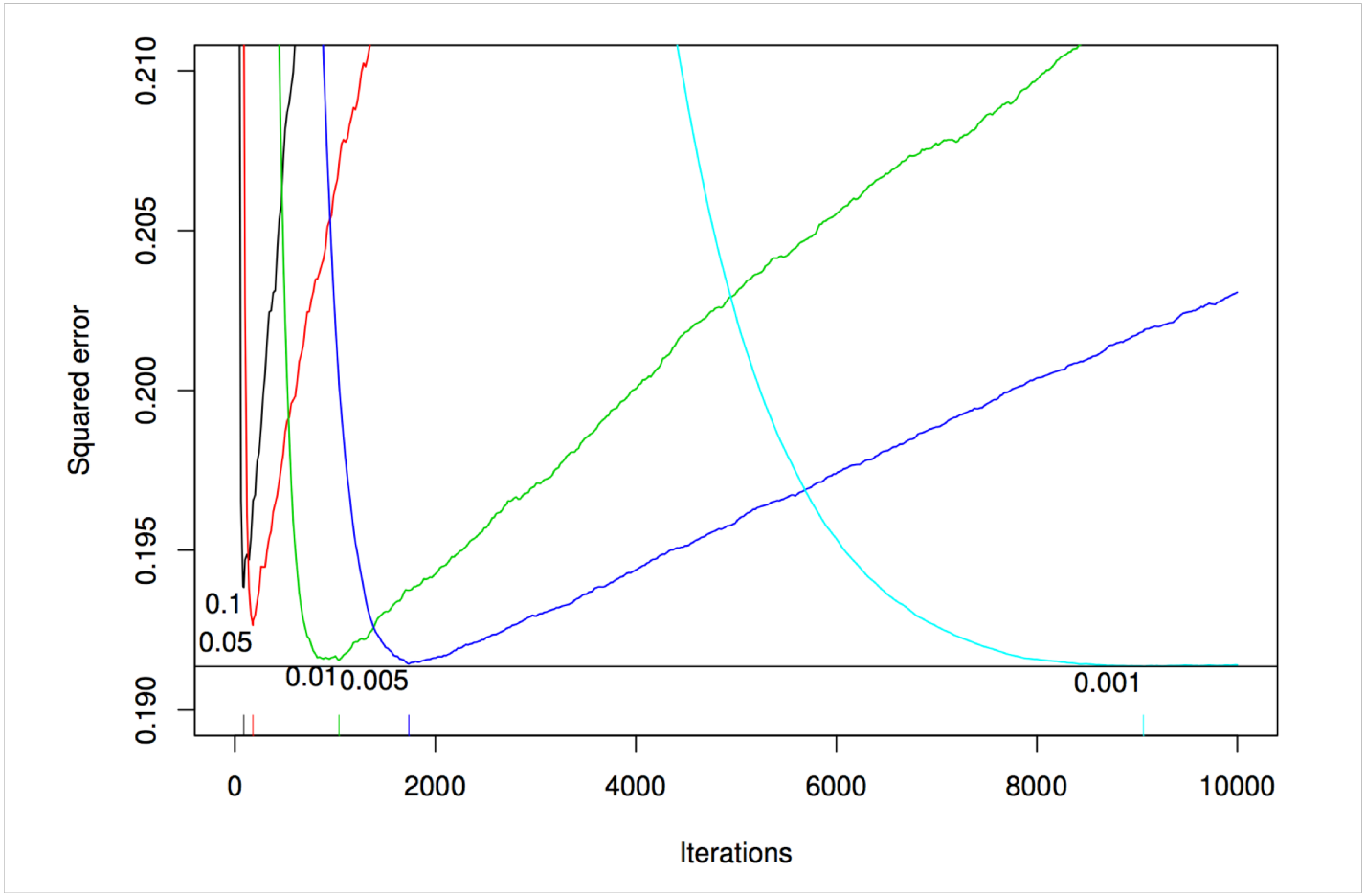
- Subsampling ratio (*subsample*)
  - The proportion of data points used in constructing the next tree
  - "row" sampling (similar to the bootstrap sampling process)

- Predictor sampling ratio (*colsample_bytree*)
  - The proportion of predictors used in constructing the next tree
  - "column" sampling (similar to the random forest process)

- …

Shrinkage vs. Iterations

- https://github.com/harrysouthworth/gbm/raw/master/inst/doc/gbm.pdf



Summary

- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.

- The latter two methods, random forests and boosting, are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

- Random Forest: less tuning, no cross-validation, parallelizable, good accuracy out of the box
- GBM: serial algorithm, heavy tuning (but parallelizable), better prediction accuracy after tuning