Cross validation, bootstrap, bagging, random forests & boosting

Jay Urbain, PhD

Gareth James, Daniela Witten, Trevor Hastie , Robert Tibshirani. An Introduction to Statistical Learning: with Applications in R.

Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

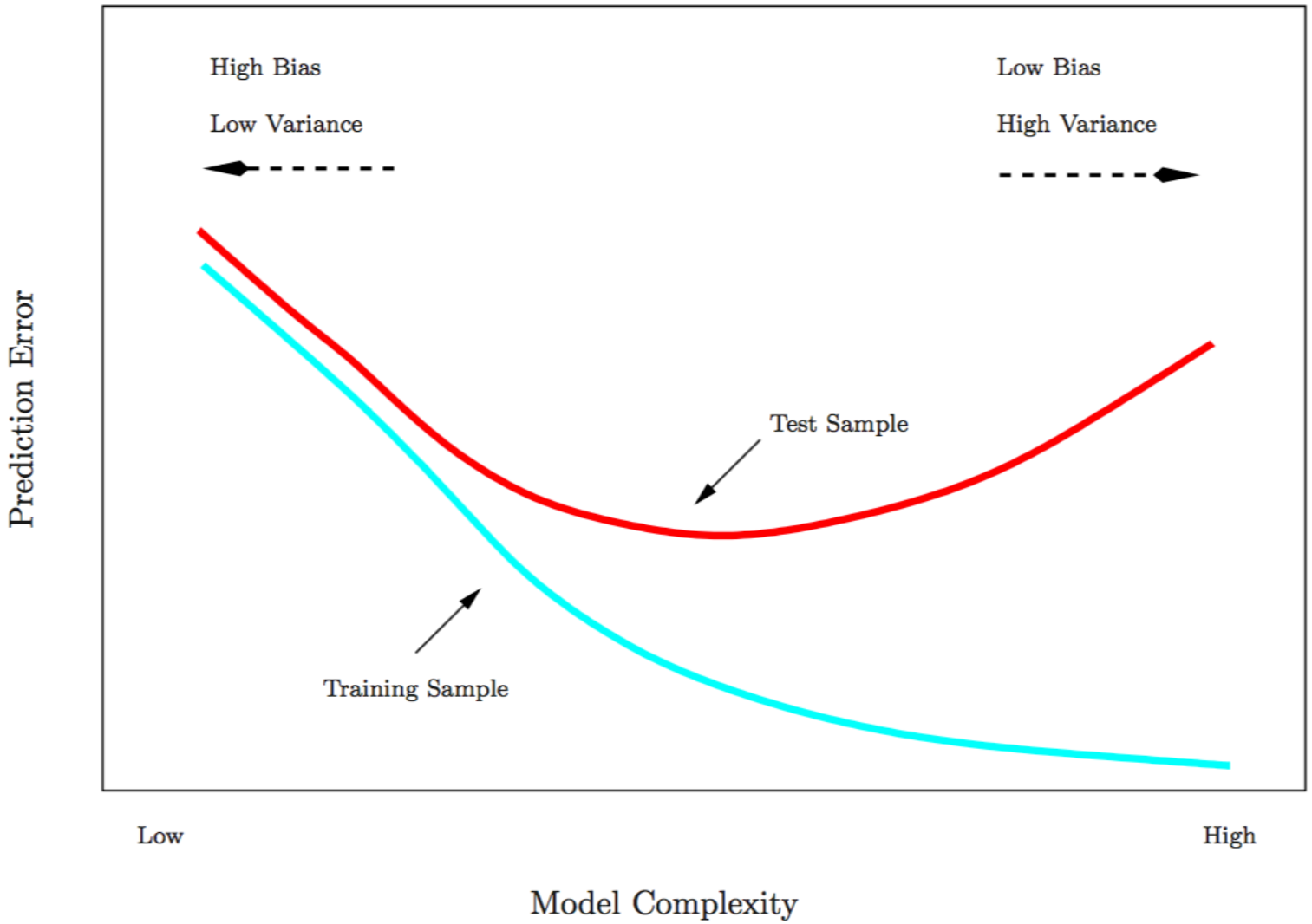
Cross-validation, the Bootstrap, Bagging, & Boosting

- These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.
- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates.

Training Error versus Test error

- Distinction between test error and the training error:
  - Test error is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
  - Training error can be easily calculated by applying the statistical learning method to the observations used in its training.
  - *But* the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.

Training- versus Test-Set Performance



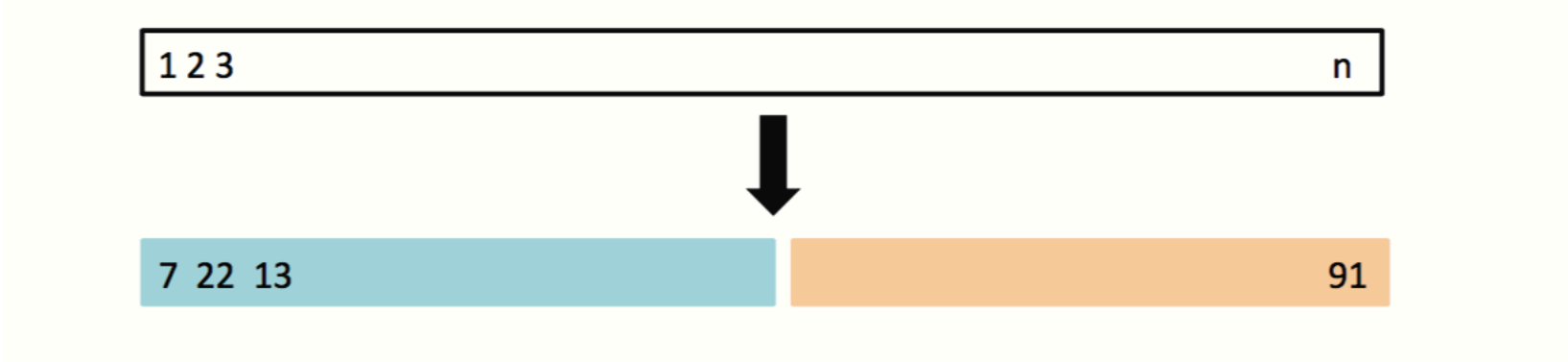
More on prediction-error estimates

- Best solution: a large designated test set. Often not available
- Some methods make a mathematical adjustment to the training error rate in order to estimate the test error rate.
  - These include the Cp statistic, AIC and BIC. They are discussed elsewhere in this course
- Instead, we consider a class of methods that **estimate the test error by holding out a subset of the training observations from the fitting process,** and then applying the statistical learning method to those held out observations

Validation-set approach

- Randomly divide the available set of samples into two parts: a **training set** and a **validation or hold-out set**.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error.
- This is typically assessed using MSE in the case of a quantitative response, and misclassification rate in the case of a qualitative (discrete) response.

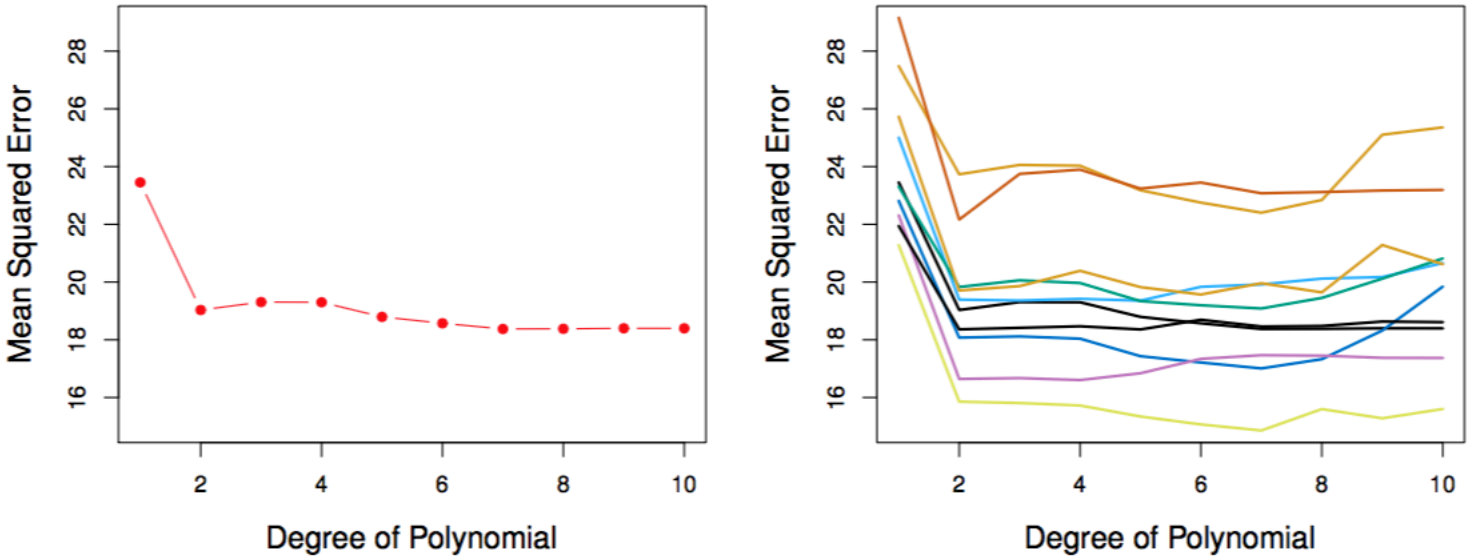
Validation proecess



A random splitting into two halves: left part is training set, right part is validation set

Example: automobile data

- Want to compare linear vs. higher-order polynomial terms in a linear regression
- Randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.



Left panel shows single split; right panel shows multiple splits

Drawbacks of validation set approach

- **Validation estimate of the test error can be highly variable**, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, **only a subset of the observations** — those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to overestimate (or underestimate) the test error for the model fit on the entire data set.

K-fold Cross-validation

- Widely used approach for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Randomly divide the data into K equal-sized parts. Leave out part k, fit the model to the other $K - 1$ parts (combined), and then obtain predictions for the left-out k^{th} part.
- This is done in turn for each part $k = 1,2,...K$, and then the results are combined.

K-fold Cross-validation in detail

Divide data into K roughly equal-sized parts ($K = 5$ here)

1	2	3	4	5
Validation	Train	Train	Train	Train

A useful special case

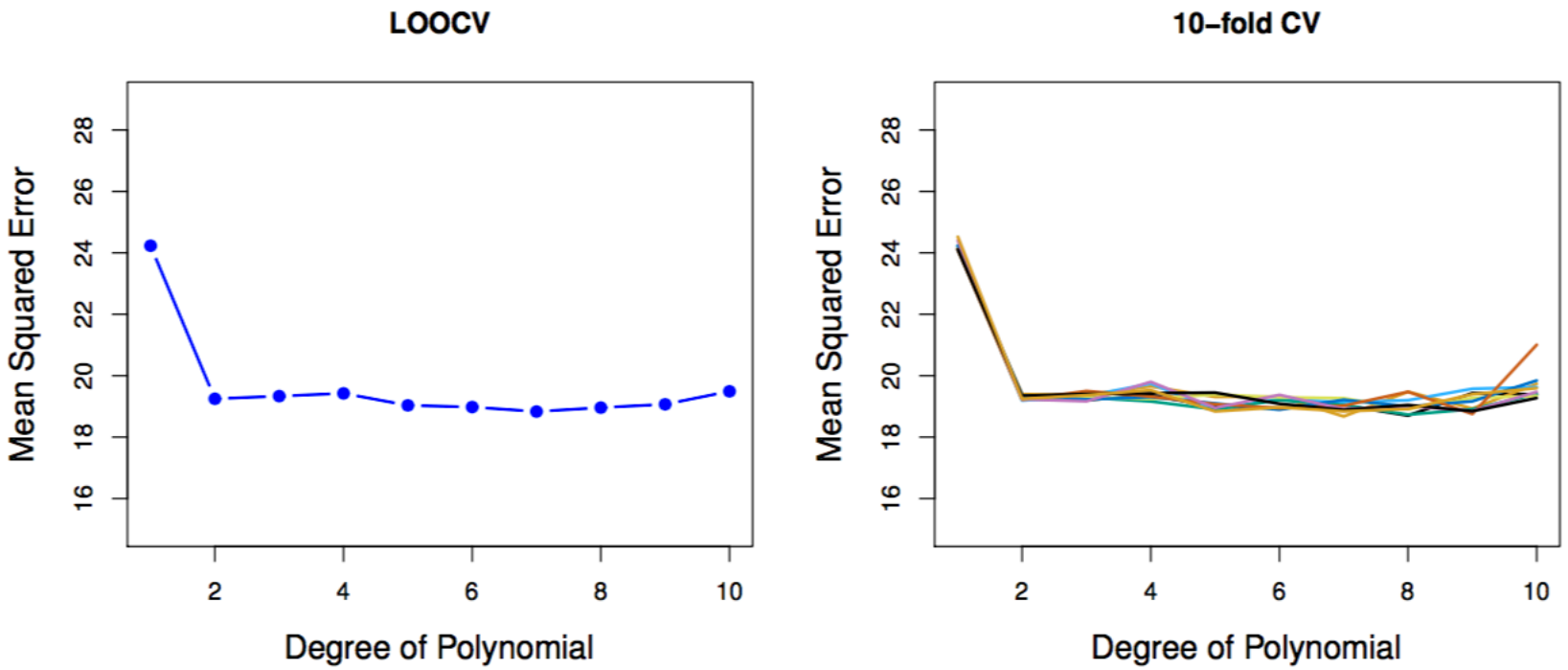
- **With least-squares linear or polynomial regression, a shortcut makes the cost of LOOCV the same as that of a single model fit.**
- The following formula holds:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

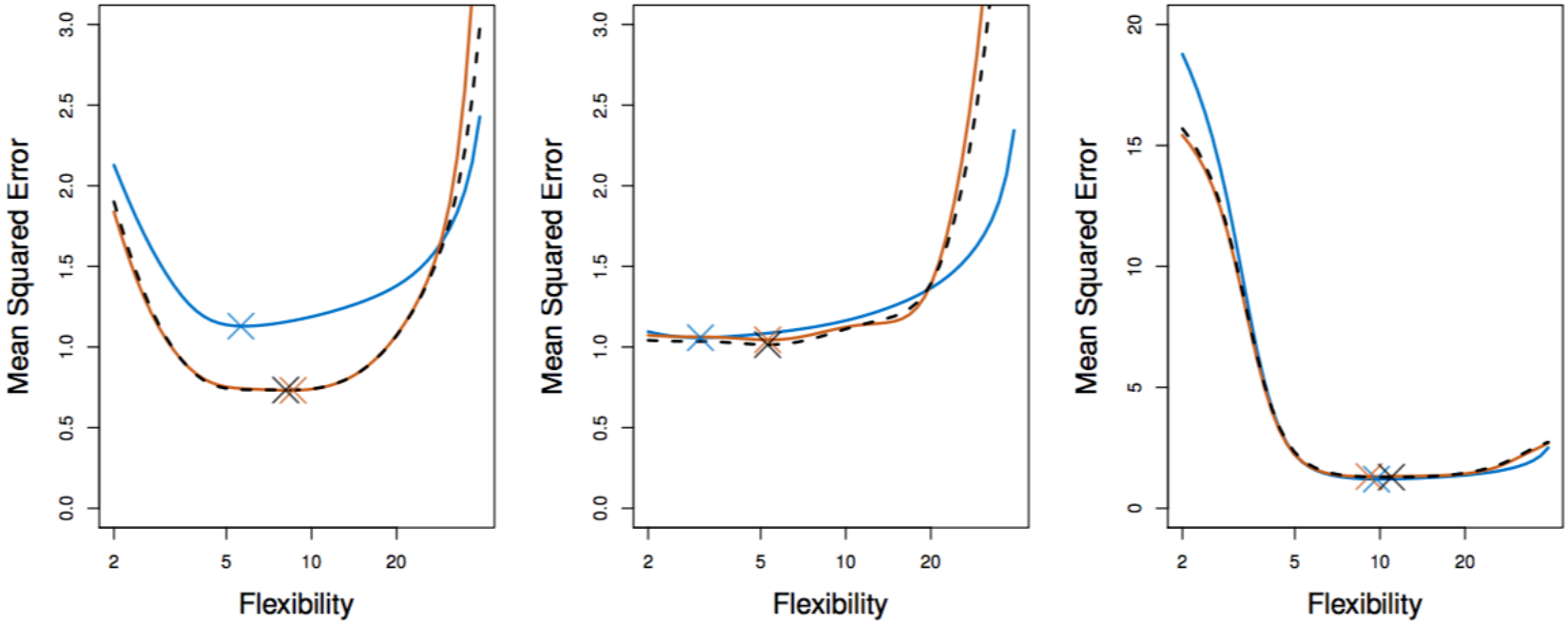
- $\hat{y}_i$ is the i^{th} fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat" matrix) Like the ordinary MSE, except the i^{th} residual is divided by $1 - h_i$.

- **LOOCV sometimes useful, but typically doesn't shake up the data enough.** The estimates from each fold are highly correlated and hence their average can have high variance.

Auto data revisited



True and estimated test MSE for the simulated data



Other issues with Cross-validation

- Since each training set is *(K − 1)/K* as big as the original training set, the estimates of prediction error will typically be biased upward.
- This bias is minimized when *K = n (LOOCV)*, but this estimate has high variance, as noted earlier.
- K = 5 or 10 provides a good compromise for this bias-variance tradeoff.

Cross-Validation for Classification Problems

- We divide the data into K roughly equal-sized parts $C_1, C_2, \ldots C_K$. C_k denotes the indices of the observations in part *k*. There are n_k observations in part k: if n is a multiple of *K*, then $n_k = n/K$.
- Compute:

$$\mathrm{CV}_K = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{Err}_k$$

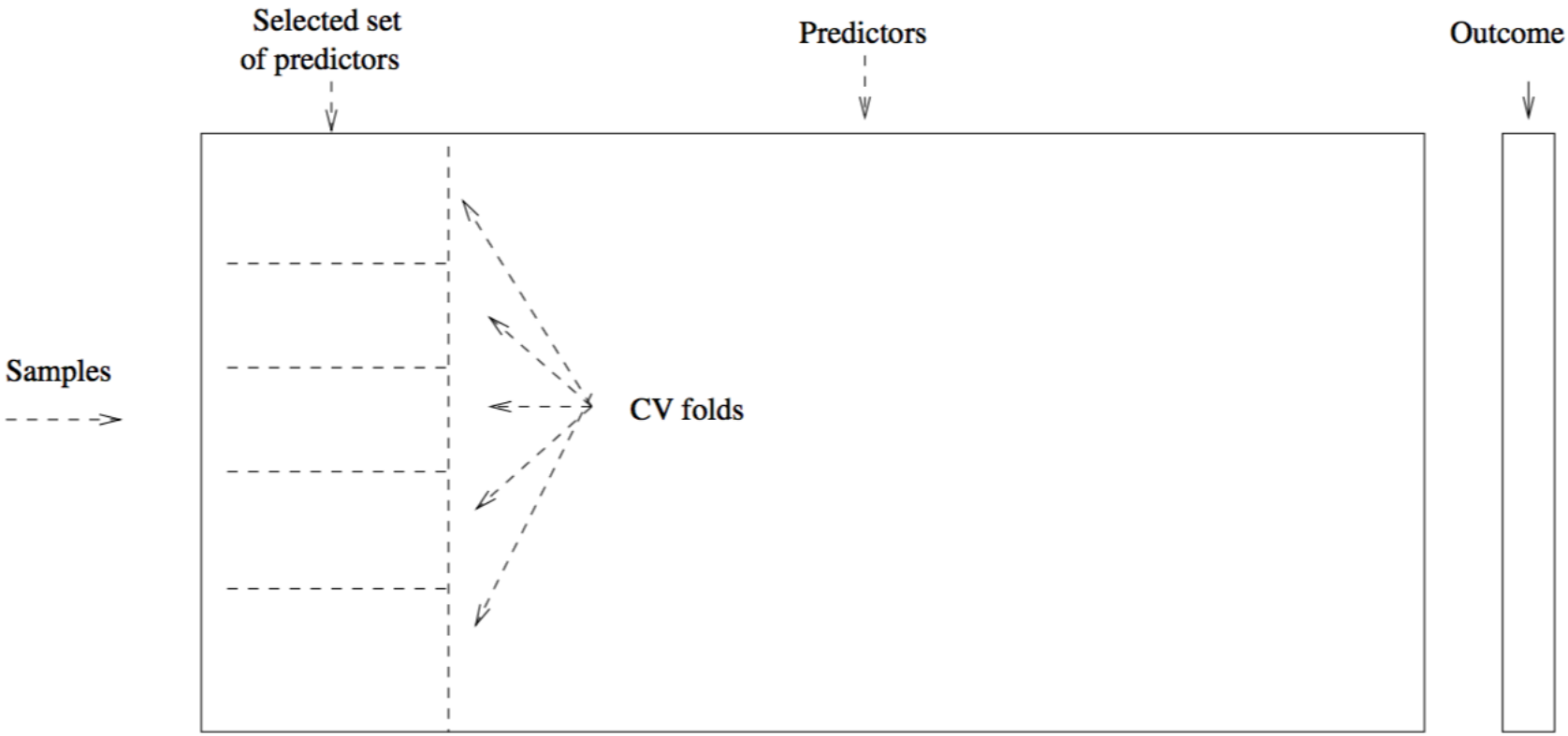
where $\mathrm{Err}_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i)/n_k$.

- The estimated standard deviation of CV_k is:

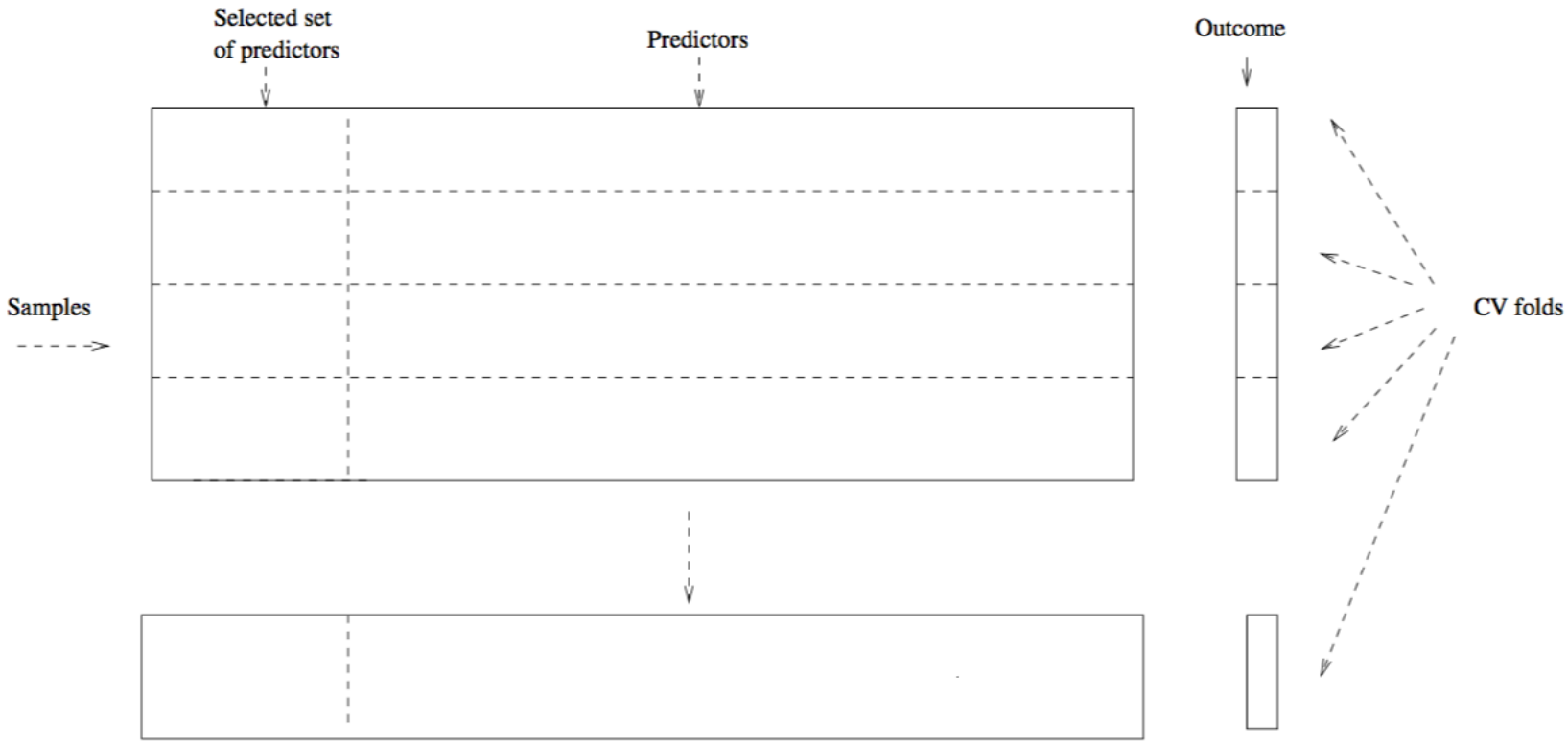
$$\widehat{\mathrm{SE}}(\mathrm{CV}_K) = \sqrt{\sum_{k=1}^{K} (\mathrm{Err}_k - \overline{\mathrm{Err}_k})^2/(K-1)}$$

Have to be careful about tuning your mode and overfit.

Applying Cross Validation – wrong way



Applying Cross Validation – rightway



The Bootstrap

- The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

Bagging

- Bootstrap aggregation, or bagging, is a general-purpose procedure for **reducing the variance** of a statistical learning method.
- Particularly useful and frequently used with decision trees.

- Given a set of *n* independent observations:
  - $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n.
  - In other words, **averaging a set of observations reduces variance.**
  - Of course, this is not practical because we generally do not have access to multiple training sets.

Bagging continued

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate *B* different bootstrapped training data sets.
- Then train our learning algorithm on the b^{th} bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x.
- We then average all the predictions to obtain

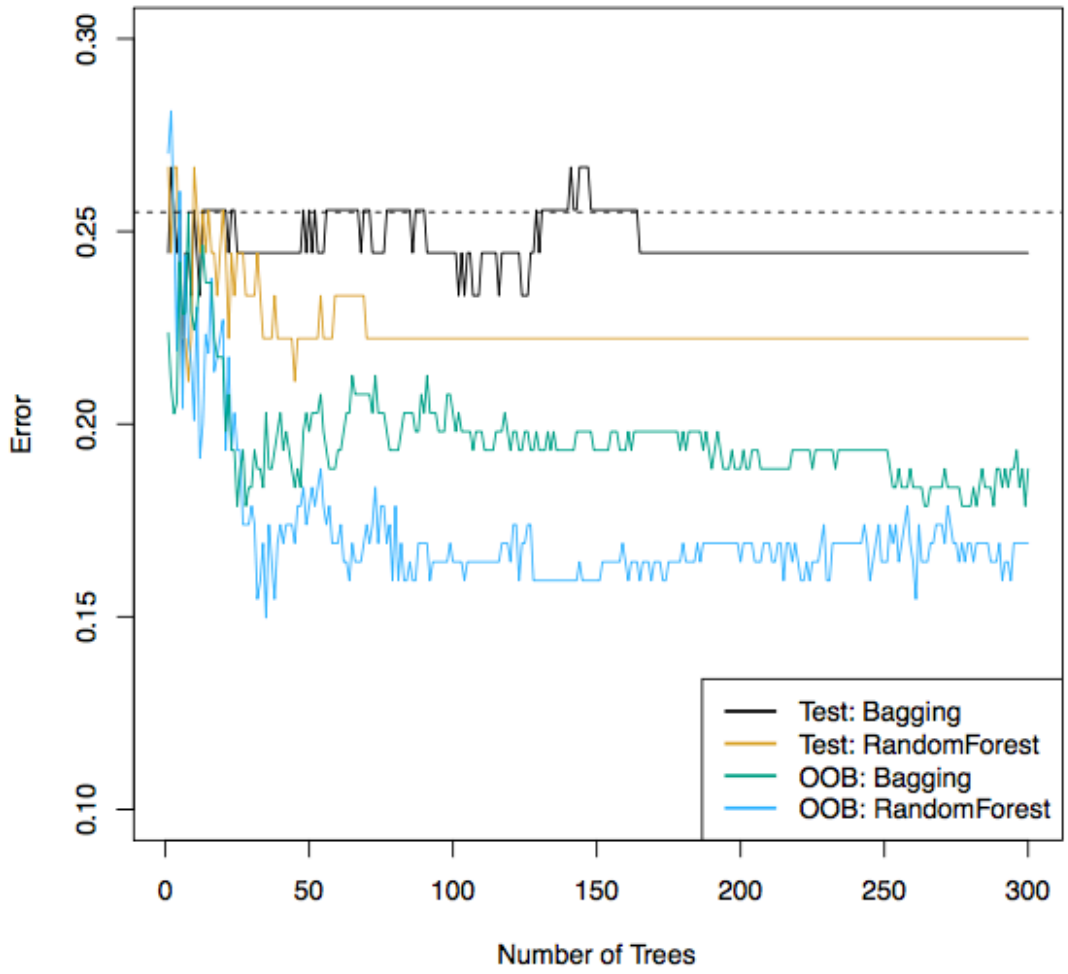
$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

- This is called bagging. Works for regression trees.

Bagging classification trees

- For each test observation, record the class predicted by each of the *B* trees.
- Take a majority vote: the overall prediction is the most commonly occurring class among the *B* predictions.

Bagging the heart data



Previous figure

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

Random Forests

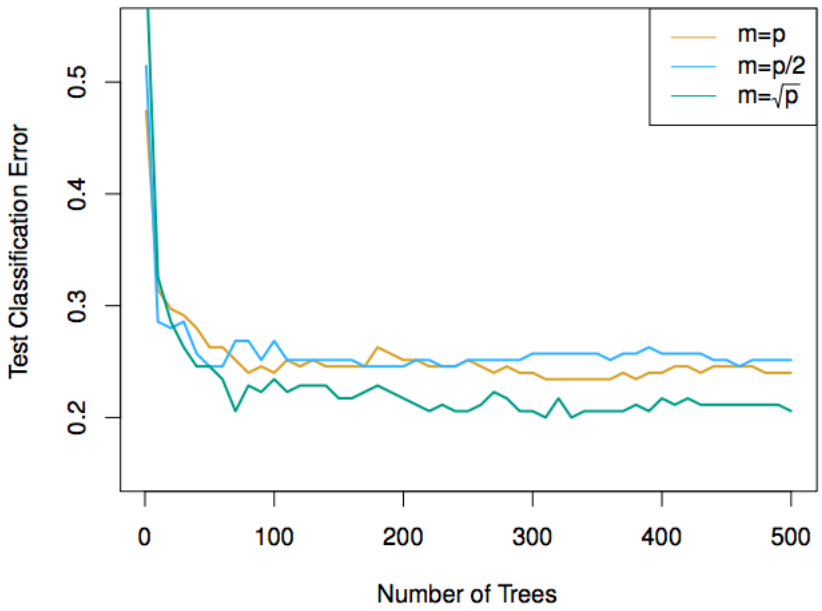
- Random forests provide an improvement over bagged trees by way of a small tweak that **decorrelates** the trees.
- This reduces the variance when the trees are averaged.
- As in bagging, build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, a **random selection of *m* predictors** is chosen as split candidates from the full set of ***p* predictors**.
- The split is allowed to use only one of those ***m* predictors**.
- A fresh selection of ***m predictors*** is taken at each split, and typically you choose ***m ≈ sqrt(p)*** — i.e, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Example: gene expression data

- Applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: **either normal or one of 14 different types of cancer.**
- We can use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- Data randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

Results: gene expression data

- Results from random forests for the fifteen-class gene expression data set with p = 500 predictors.
- The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node.
- Random forests (m < p) lead to a slight improvement over bagging (m = p). A single classification tree has an error rate of 45.7%.



Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- Bagging involves creating multiple copies of the original training data set using the **bootstrap**, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except **that the trees are grown sequentially: each tree is grown using information from previously grown trees.**

Boosting algorithm 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:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

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

   - 2.3 Update the residuals,

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

3. Output the boosted model,

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

Idea behind boosting procedure

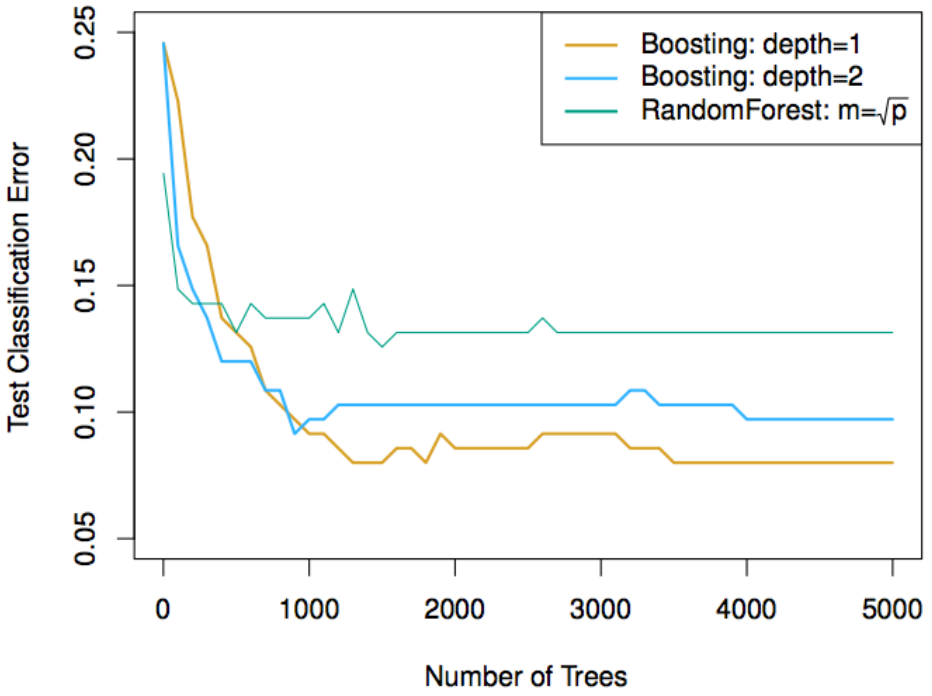
- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially over fitting, the boosting approach instead **learns slowly**.
- Given the current model, we fit a decision tree to the residuals from the model.
- We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well.
- The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex.
- The R package gbm (gradient boosted models) handles a variety of regression and classification problems.

Previous figure

- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal.
- The test error is displayed as a function of the number of trees. For the two boosted models, λ = 0.01.
- Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tre is 24%.



Tuning parameters for boosting

- The number of trees B.
- Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This 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, which controls the complexity of the boosted ensemble. Often d = 1 works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model.
- More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy
- 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.