STAT 380: Polynomial, Splines and the corresponding Cross Validation

An Example using Linear Regression on BostonHousing.csv Data

Prediction Using a Statistical/Machine Learning Model

Let $\{y_i, \mathbf{X}_i\}_{i=1}^{n}$ be the observed data. And there is a statistical/machine learning model that provides a prediction for the responses y_i.

We denoted the predicted value for the responses to be $\hat{y}_i$

The Boston Housing Dataset

Boston Housing data set

Boston Housing data set

- The Boston Housing data contain information on census tracts in suburbs of Boston.
- Several measurements are included (e.g., crime rate, pupil-teacher ratio).
- 14 variables for each of the 506 houses.

Possible tasks:

- A supervised predictive task, where the outcome is the median value of a home.
- A supervised classification task, where the outcome is the binary variable *CAT.MEDV* that indicates whether the home value is above or below $\$30,000$.
- An unsupervised task, where the goal is to cluster houses.

Variables in the Boston housing data set

Variable	Name
Crime rate	CRIM
Percentage of residential land zoned for lots over 25,000 ft^2	ZN
Percentage of land occupied by nonretail business	INDUS
Does tract bound Charles River (= 1 if tract bounds river)	CHAS
Nitric oxide concentration (parts per 10 million)	NOX
Average number of rooms per dwelling	RM
Percentage of owner-occupied units built prior to 1940	AGE
Weighted distances to five Boston employment centers	DIS
Index of accessibility to radial highways	RAD
Full-value property tax rate per $10,000	TAX
Pupil-to-teacher ratio by town	PTRATIO
Percentage of lower status of the population	LSTAT
Median value of owner-occupied homes in $1000s	MEDV
Is median value of owner-occupied homes in tract above $30,000 (CAT.MEDV = 1) or not (CAT.MEDV = 0)	CAT.MEDV

Boston housing data set: Overview

The `head()`-command returns the first parts of a vector, matrix, table, data frame or function.

```
> head(Daten)
 CRIM ZN INDUS   RM  AGE   DIS RAD TAX LSTAT MEDV CMEDV
0.006 18  2.31 6.57 65.2 4.090   1 296  4.98 24.0     0
0.027  0  7.07 6.42 78.9 4.967   2 242  9.14 21.6     0
0.027  0  7.07 7.18 61.1 4.967   2 242  4.03 34.7     1
0.032  0  2.18 6.99 45.8 6.062   3 222  2.94 33.4     1
0.069  0  2.18 7.14 54.2 6.062   3 222  5.33 36.2     1
0.029  0  2.18 6.43 58.7 6.062   3 222  5.21 28.7     0
```

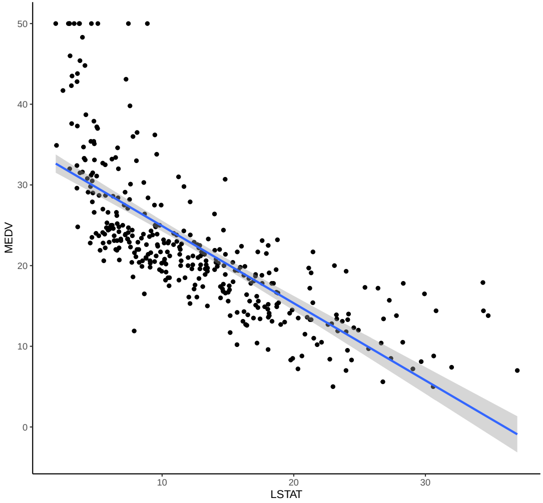
Boston housing data set: Overview

The `str()`-command displays the internal structure of an R object.

```
> str(Daten)
'data.frame': 506 obs. of  14 variables:
 $ CRIM      : num  0.00632 0.02731 0.02729 0.03237 ...
 $ ZN        : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ INDUS     : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 ...
 $ CHAS      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NOX       : num  0.538 0.469 0.469 0.458 0.458 ...
 $ RM        : num  6.58 6.42 7.18 7 7.15 ...
 $ AGE       : num  65.2 78.9 61.1 66.6 96.1 100 85.9 ...
 $ DIS       : num  4.09 4.97 4.97 6.06 6.06 ...
 $ RAD       : int  1 2 2 3 3 3 5 5 5 5 ...
 $ TAX       : int  296 242 242 311 311 311 ...
 $ PTRATIO   : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 ...
 $ LSTAT     : num  4.98 9.14 4.03 2.94 5.33 ...
 $ MEDV      : num  24 21.6 34.7 33.4 36.2 28.7  ...
 $ CAT..MEDV: int  0 0 1 1 1 0 0 0 0 0 ...
```

Polynomial Regression

Illustration: Motivation



Polynomial regression

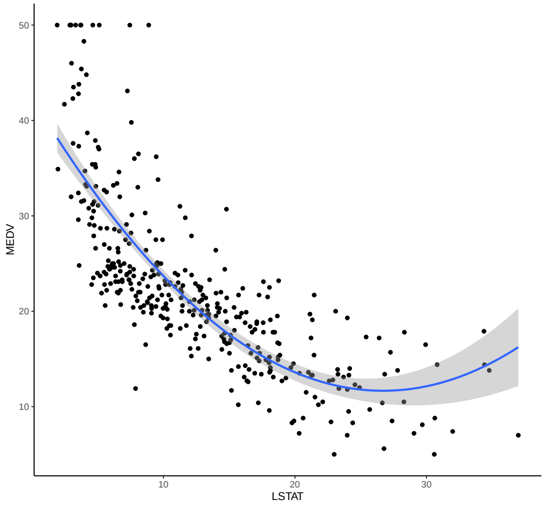
- It **extends** the linear model by adding extra predictors, obtained by raising each of the original predictors to a power.
- For example, a cubic regression uses three variables, X, X^2, and X^3, as predictors.
- This approach provides a simple way to provide a **nonlinear** fit to data.
- It is considered to be a special case of multiple linear regression.
- A polynomial regression may lead to increase in **complexity** as the number of covariates also increases.
- Polynomial models should be hierarchical, containing the terms X, X^2, and X^3, in a hierarchy.

Polynomial regression in R: Boston housing data

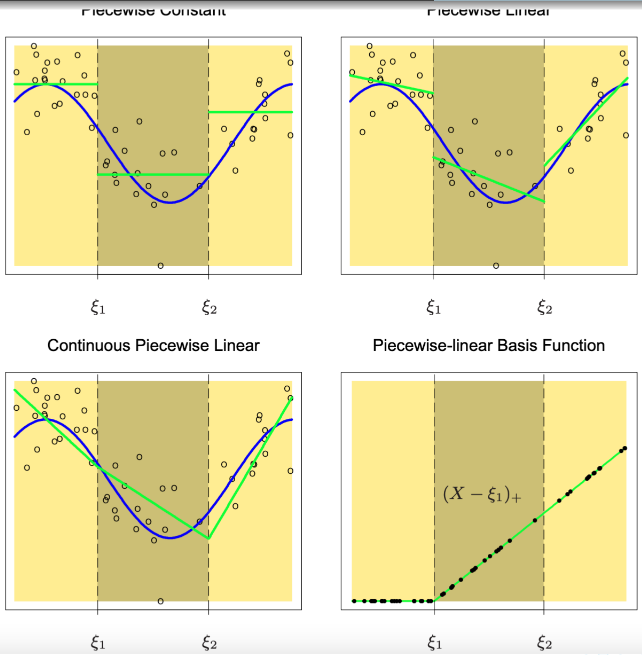
We can use the `poly()`-command for specifying a polynomial regression:

```r
# To fit a polynomial model
modelfinal <- lm(MEDV ~ poly(LSTAT, 2, raw = TRUE), data
    = train)
# Make predictions
predictions <- modelfinal %>% predict(test)
# Model performance
data.frame(RMSE = RMSE(predictions, test$MEDV),
R2 = R2(predictions, test$MEDV))
# Let's check the curve
ggplot(train, aes(LSTAT, MEDV) ) + geom_point() +
stat_smooth(method = lm, formula = y ~ poly(x, 2, raw =
    TRUE))
# Let's check the assumptions
residuals <- data.frame('Residuals' = modelfinal$
    residuals)
```

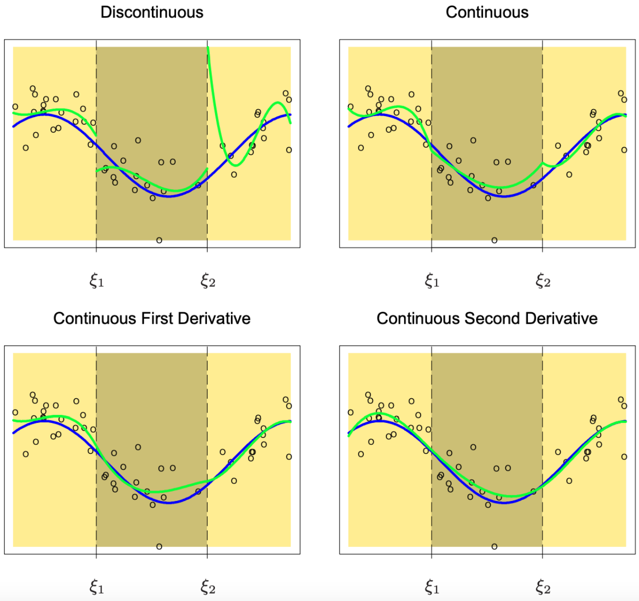
Polynomial regression: Boston housing data



Regression Splines


Piecewise Constant
Piecewise Linear
ξ_1
ξ_2
ξ_1
ξ_2
Continuous Piecewise Linear
Piecewise-linear Basis Function
$(X - \xi_1)_+$
ξ_1
ξ_2
ξ_1
ξ_2

Piecewise Cubic Polynomials



Regression splines

- Regression splines is a flexible class of basis functions that extends upon the polynomial regression.
- Instead of fitting a high-degree polynomial over the entire range of X we can make a **zoom** into the data and fit such polynomial there.
- Then, we move to a next small region and fit again such a polynomial.
- As a result, we obtain a connection of these little polynomials so that we end up with a **continuous smooth curve** through the points.
- Smoothing splines is a kind of **piecewise continuous function** composed by those polynomials to model the entire data set.

Regression splines: knots

- It is a way of smoothly interpolating between fixed points, called **knots**.
- The knots can be found throughout cross-validation.
- And after defining the points, the polynomial regression is computed between those knots.
- Regression splines often give superior results to polynomial regression. This is because unlike polynomials, which must use a high degree to produce flexible fits, splines introduce flexibility by increasing the number of knots but keeping the degree fixed.

Smooting Splines

Smoothing splines are obtained by minimizing the follopwing:

- Let $\{y_i, \mathbf{X}_i\}_{i=1}^n$ be the observed data. And there is a statistical/machine learning model that provides a prediction for the responses y_i.

We try to fit a general function equation

$$y_i = f(x_i) + \epsilon_i$$

We minimize

$$RSS(f, \lambda) = \sum_{i=1}^n \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt$$

Smoothing splines: ideas behind

- In statistics and image processing, to **smooth** a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena. (Wikipedia)
- Smoothing splines are function estimates obtained from a set of noisy observations in order to balance a measure of goodness of fit with a derivative based measure of the **smoothness**. (Wikipedia)

Smoothing splines: definition

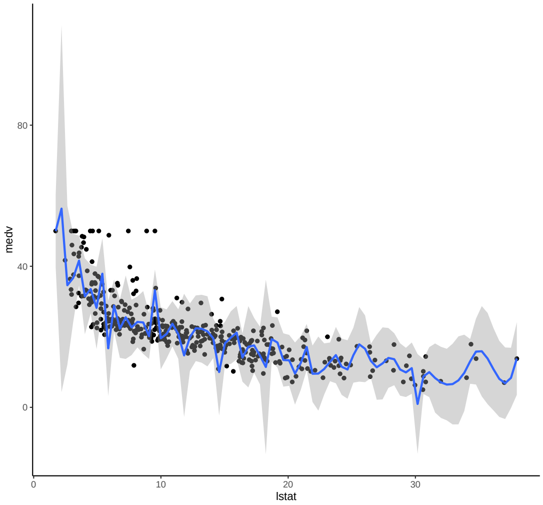
- Smoothing splines are a popular approach for **non-parametric** regression problems.
- The knots and smoothness are controlled via the **tuning parameter** α. This controls the roughness of the smoothing spline, and hence the effective degrees of freedom.
- The knot selection problem completely by using a **maximal set of knots**. The complexity of the fit is controlled by "**regularization**".
- It penalises for the **roughness of the fitting**.
- **Overfitting** disadvantage.

Splines in R: Boston housing data

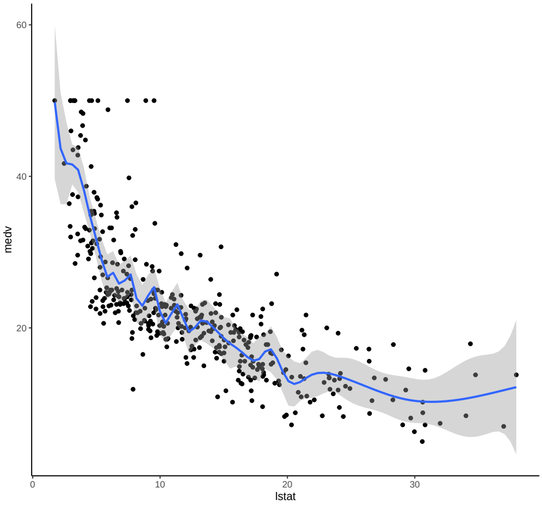
From the library splines we can fit a smoothing splines using the `bs()`-command:

```r
# To fit smoothing splines
library(splines)
# Build the model
knots <- quantile(train$LSTAT, p = c(0.25, 0.5, 0.75))
modelss <- lm (MEDV ~ bs(LSTAT, knots = knots), data =
    train)
# Make predictions
predictions <- modelss %>% predict(test)
# Model performance
data.frame(
RMSE = RMSE(predictions, test$MEDV),
R2 = R2(predictions, test$MEDV))
# Let's check the curve
ggplot(train, aes(LSTAT, MEDV) ) + geom_point() +
stat_smooth(method = lm, formula = y ~ splines::bs(x, df
    = 205))
```

Smoothing splines (larger df): Boston housing data



Smoothing splines (smaller df): Boston housing data



Measuring Predictive Performance

Evaluating predictive performance

Some first comments:

- Predictive accuracy is not the same as goodness-of-fit (R^2).
- Classical statistical measures of performance are aimed at finding a model that fits well to the data on which the model was trained.
- We are interested in models that have high predictive accuracy when applied to new records.
- For assessing prediction performance, we use **several measures**.
- The measures are based on the validation or test set, which serves as a more objective ground than the training set to assess predictive accuracy.

Naive benchmark: The average

- The **benchmark criterion** in prediction is using the average outcome value (thereby ignoring all predictor information).
- The prediction for a new record is simply the average across the outcome values of the records in training set.

$$\hat{y}_j = \frac{1}{n_t}\sum_{i=1}^{n_t} y_i = \overline{y}_t \quad \text{for all } \; j \in Vali$$

- A good predictive model should outperform the benchmark criterion in terms of predictive accuracy.

A more complex approach: Linear regression

- For making predictions the **multiple linear regression** model is used.
- Two popular but different objectives behind fitting a regression model are:
  - Explaining or **quantifying** the average effect of inputs on an outcome.
  - **Predicting** the outcome value for new records, given their input values.
- In the course, the focus is on predicting new individual records. Here we are not interested in the coefficients themselves, nor in the *average unit*, but rather in the **predictions** that this model can generate for new records.

A more complex approach: Linear regression

Remember that to predict the values of the outcome variable for a record with predictor values **X**, we use the equation:

$$\hat{y} = X\hat{\beta}.$$

Predictions based on this equation are the best predictions possible in the sense that they will be unbiased and will have the smallest MSE compared to any unbiased estimates under particular assumptions.

More prediction accuracy measures: R output

Remember, the **prediction error** for unit i is defined as: $e_i = y_i - \hat{y}_i$.
A few popular numerical measures of predictive accuracy are:

- Mean absolute error $MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i|$.
- Mean error $ME = \frac{1}{n}\sum_{i=1}^{n} e_i$, gives an indication of whether the predictions are on average over- or underpredicting the outcome.
- Mean percentage error $MPE = \frac{100}{n}\sum_{i=1}^{n} \frac{e_i}{y_i}$, gives the percentage score of how predictions deviate from the actual values, taking into account the direction.
- Mean absolute percentage error $MAPE = \frac{100}{n}\sum_{i=1}^{n} |\frac{e_i}{y_i}|$.
- Root mean squared error $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$.

Note that all these measures are influenced by **outliers**. To check outlier influence, we can compute median-based measures or simply plot a histogram or boxplot of the errors.

Prediction errors: Boston housing data

We investigate prediction error metrics from a model for the median value of a home: Training and validation.

```
# Run linear regression model: Median value ~ Crime rate
   + River 1/0 + Number of rooms + Teacher/Pupil ratio
> fit<-lm(MEDV~CRIM+CHAS+RM+PTRATIO,data=train)
> summary(fit)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.28257    4.32789  -0.527  0.59820
CRIM        -0.20104    0.03201  -6.280 8.84e-10 ***
CHAS         2.97189    1.14598   2.593  0.00985 **
RM           7.24178    0.42543  17.022  < 2e-16 ***
PTRATIO     -1.09026    0.14484  -7.527 3.45e-13 ***

Residual standard error: 5.691 on 401 degrees of freedom
Multiple R-squared:  0.6265,  Adjusted R-squared:  0.6228
F-statistic: 168.2 on 4 and 401 DF,  p-value: < 2.2e-16
```

Prediction errors: Boston housing data

The `accuracy()`-command returns the prediction error metrics.

```
# Loading libraries and the data
> library(forecast)

# Create predictions
pred_t_reg<-predict(fit,newdata = train)
pred_v_reg<-predict(fit,newdata = vali)

# Evaluate performance
  # Training - linear regression
>   accuracy(pred_t_reg,train$MEDV)
         ME   RMSE    MAE     MPE    MAPE
Test set  0 5.6556 3.8768 -6.5779 21.0999
  # Validation - linear regression
>   accuracy(pred_v_reg,vali$MEDV)
              ME  RMSE    MAE     MPE    MAPE
Test set -0.2754 6.367 4.0833 -8.3027 21.1321
```

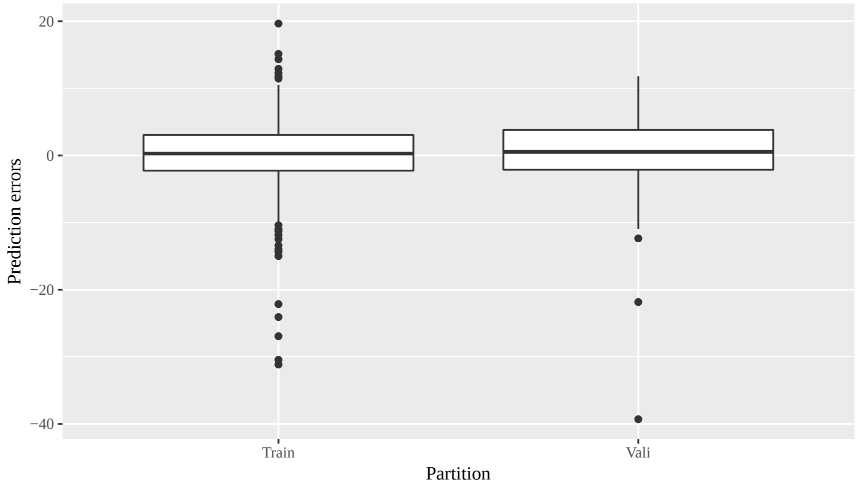
Prediction errors: Boston housing data

Comparing **training and validation performance**:

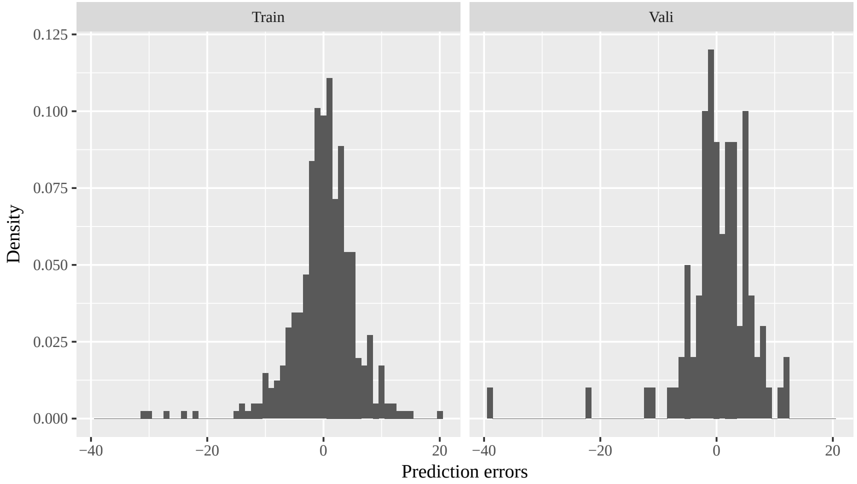
- Errors that are based on the training set tell us about model fit, whereas those that are based on the validation set measure the model's ability to predict new data.
- We expect training errors to be smaller than the validation errors, and the more complex the model, the greater the likelihood that it will overfit the training data.
- In an extreme case of overfitting, the training errors would be zero (perfect fit of the model to the training data), and the validation errors would be non-zero and non-negligible.

For this reason, it is important to compare the error plots and metrics of the training and validation sets.

Prediction errors: Boston housing data



Prediction errors: Boston housing data



“Cross Validation” to measure the predictive accuracy of a supoervised learning method

Cross-validation

When the **number of units is small**, data partitioning might not be advisable as each partition will contain too few records for model building and performance evaluation.

One alternative to data partitioning is cross-validation:

- **Cross-validation** (CV) is a procedure that starts with partitioning the data into *folds*, or non-overlapping subsamples.
- Each time, one of the **folds** is used as the validation set and the remaining $k - 1$ folds serve as the training set.
- Combine the model's predictions on each of the k validation sets in order to evaluate the overall performance of the model.

Cross-validation is also used for choosing/ tuning the parameters in algorithms.

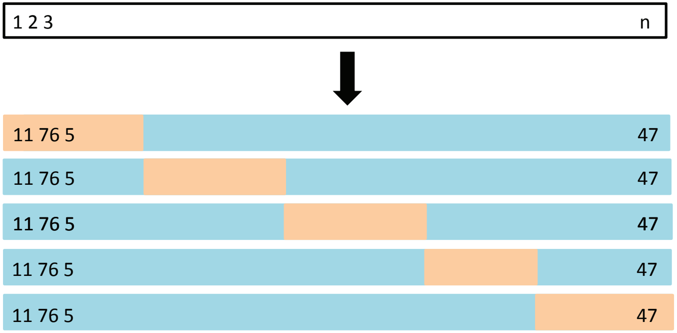
k-fold cross-validation

- Approach involves randomly k-**fold CV** dividing the set of observations into k groups, or folds, of approximately equal size.
- The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds.
- The mean squared error, MSE_1, is then computed on the observations in the held-out fold.
- This procedure is repeated k times; each time, a different group of observations is treated as a validation set.
- The k-fold CV estimate is computed by averaging these values

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i.$$

- In practice, one typically performs k-fold CV using $k = 5$ or $k = 10$.

5-fold cross-validation: A schematic display



A set of n observations is randomly split into 5 non-overlapping groups. Each of these fifths acts as a validation set, and the remainder as a training set. The test error is estimated by averaging the 5 resulting MSE estimates.

k-fold cross-validation

What is the advantage of using $k = 5$ or $k = 10$ rather than $k = n$?

- **Leave-one-out cross-validation (LOOCV)** with $k = n$ requires fitting the statistical learning method n times - computational expensive.
- k-fold CV gives more accurate estimates of the test error rate than does LOOCV.

Bias-variance trade-off for k-fold cross-validation:

- LOOCV gives approximately unbiased estimates of the test error, since each training set contains $n - 1$ observations, which is almost as many as the number of observations in the full data set.
- LOOCV has much higher variance than does k-fold CV with $k < n$.
- LOOCV averages outputs of n fitted models on an almost identical data - outputs are highly (positively) correlated.
- Since the mean of highly correlated quantities has higher variance than does the mean of quantities that are not as highly correlated, the test error estimate (LOOCV) has higher variance than the one (k-fold CV).

Identifying optimal model based on Cross Validation

True and estimated test MSE: Two goals

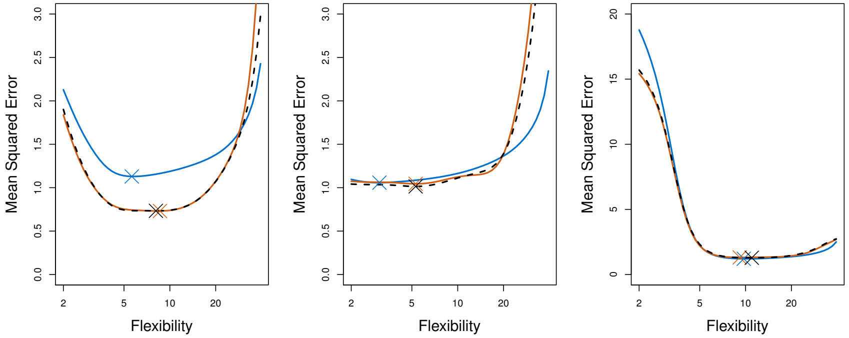
First goal:

- ... might be to determine how well a given statistical learning procedure can be expected to perform on independent data - the actual estimate of the test MSE is of interest.

Second goal:

- ... might be to determine the location of the minimum point in the estimated test MSE curve.
- We perform CV on a number of statistical learning methods, or on a single method using different levels of flexibility to identify the method that results in the lowest test error.
- Location of the minimum point in the estimated test MSE curve is important, but the actual value of the estimated test MSE is not.

True and estimated test MSE: Two goals



True and estimated test MSE for the simulated data sets. The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.

Cross-validation: Boston housing data

Cross-validation can be automatically computed for any generalized linear model using the `cv.glm()`-command.

```
> # Run linear regression model: Median value ~ Crime
   rate + Number of rooms + Teacher/Pupil ratio
> fit<-glm(MEDV~CRIM+RM+PTRATIO,data=Daten)

> # Leave-one-out cross-validation
> cv_one_err<-cv.glm(Daten,fit)
> cv_one_err$delta
[1] 35.00064 34.99989

> # 5-fold cross-validation
> cv_5_err<-cv.glm(Daten,fit,K=5)
> cv_5_err$delta
[1] 34.98018 34.89865
```

Identifying “best” polynomial model based on Cross Validation:

An Example using

BostonHousing.csv data

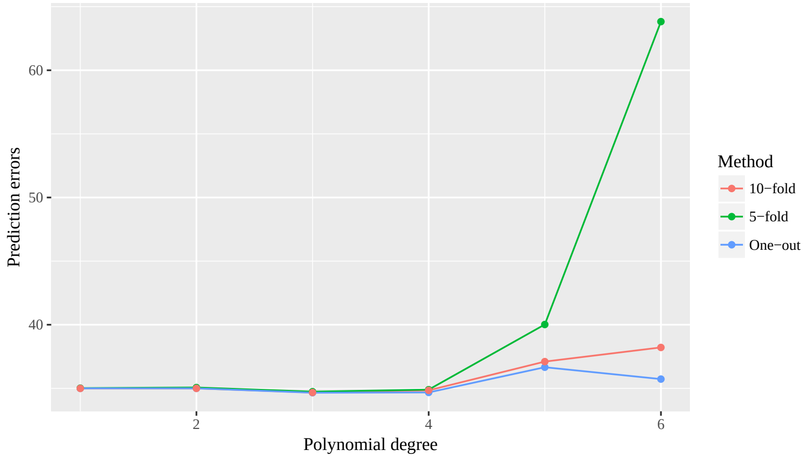
Cross-validation: Boston housing data

We investigate prediction error metrics from a model for the median value of a home: Linear vs. polynomial model for variable *Crime rate*.

```
> cv_error<-NULL
>
> for(i in 1:6){
+   fit_poly<-glm(MEDV~poly(CRIM,degree=i)+RM+PTRATIO,
   data=Daten)
+   cv_error[i]<-cv.glm(Daten,fit_poly,K=10)$delta[1]
+ }
> cv_error
[1] 34.708 34.813 35.076 34.423 41.874 51.530
```

The results don't show a clear improvement from using higher-order (> 4) polynomials. However, results may depend on the random split.

Cross-validation: Boston housing data



Each CV approach was run 100 separate times, each with a different random split of the data into K parts. The figure shows the median prediction error over replications.

Thank You