MODEL SELECTION AND REGULARISATION

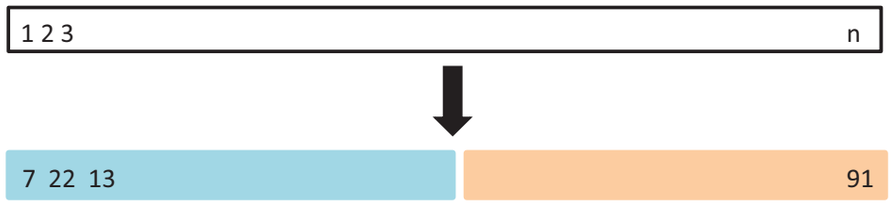
ESTIMATING THE ACCURACY OF THE MODEL

- We train the model with available data, and we will make predictions on unseen examples.
- Why do we need to estimate the accuracy of our model? (Instead of accuracy let's say figure of merit FoM, accuracy, log loss, correlation, AUC, etc)
  - To know what to expect (should we risk our money on our stock market predictions)
  - To **select the most accurate model!**
- **Model selection**
  - The type of the model (knn, linear, trees, svm, nn, etc)
  - Best combination of input variables or engineered features

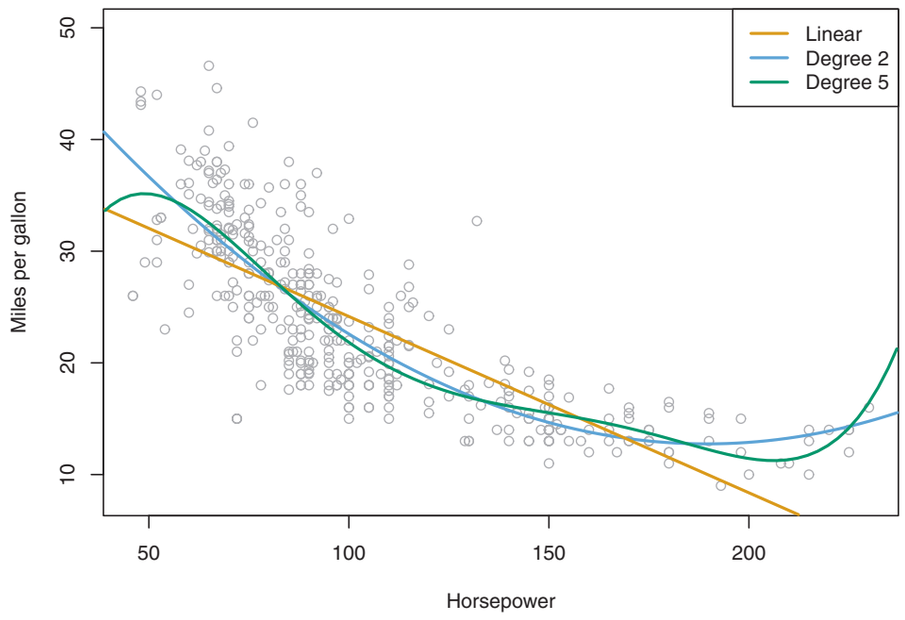
ESTIMATING THE ACCURACY OF THE MODEL

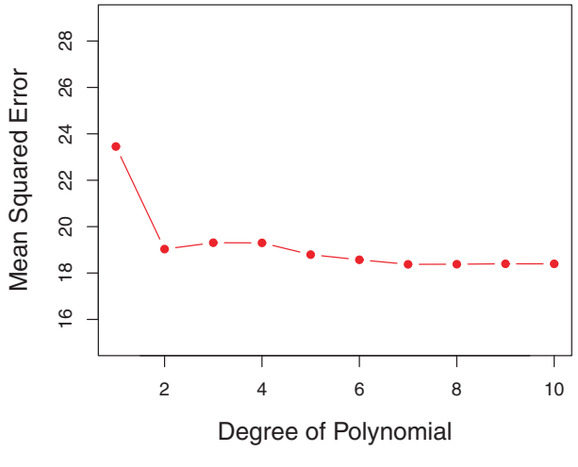
- Usually we can not estimate the FoM on the training data!
  - Each model is able to learn the noise to some varying extent. How much worse it will be on unseen data?
  - We cannot really tell whether one model is more accurate than the other just based on the training error.
  - Maybe it learned something general which will work on unseen data points, or maybe it simply learned to memorise the noise more.
- There are solutions for simple models (AIC, BIC), but not for the more complex ones
- We need a general framework for FoM estimation and model selection.

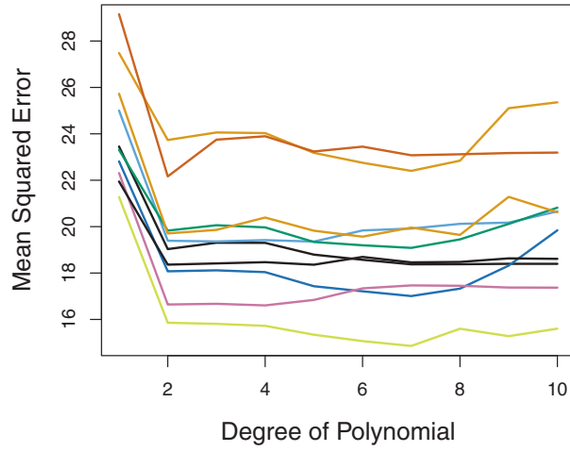
TRAIN – VALIDATION SPLIT



- Cut the training data into 2 parts, training and validation
- Not ideal, we use **only a subset for validation**: our estimate of the FoM will not the most precise.
- Balancing between training and validation dataset size, 10, 20, 30% usually.
- Typically used when datasets are huge, and training is very expensive: this is the standard in image recognition.
- Note, that in small sets the accuracy may be widely varying among different splits.
- Note: we split the training data! Competitions use a held-out test data to ensure fair evaluation but that is a different setup.

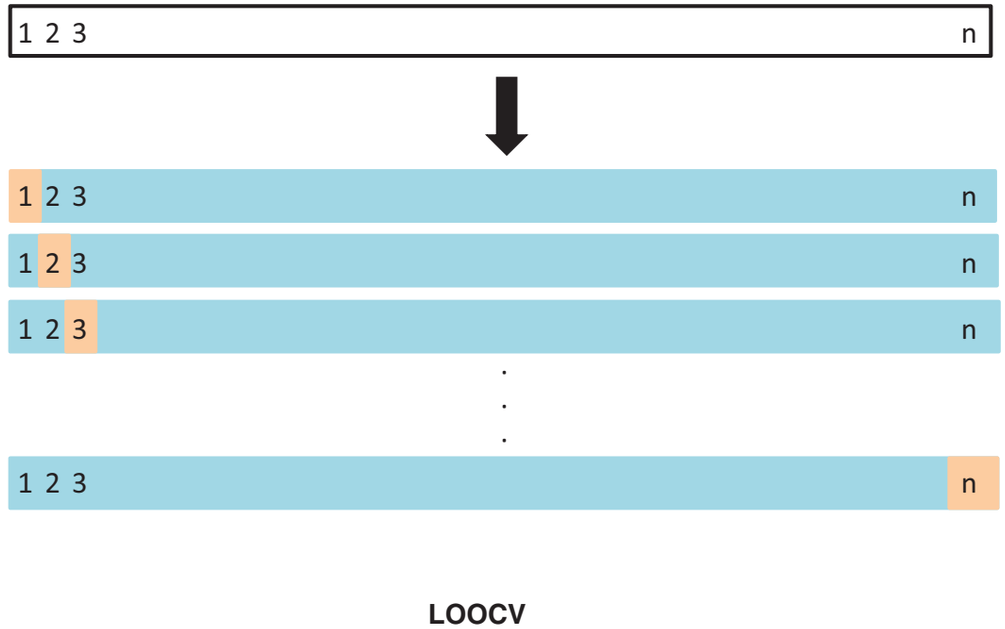




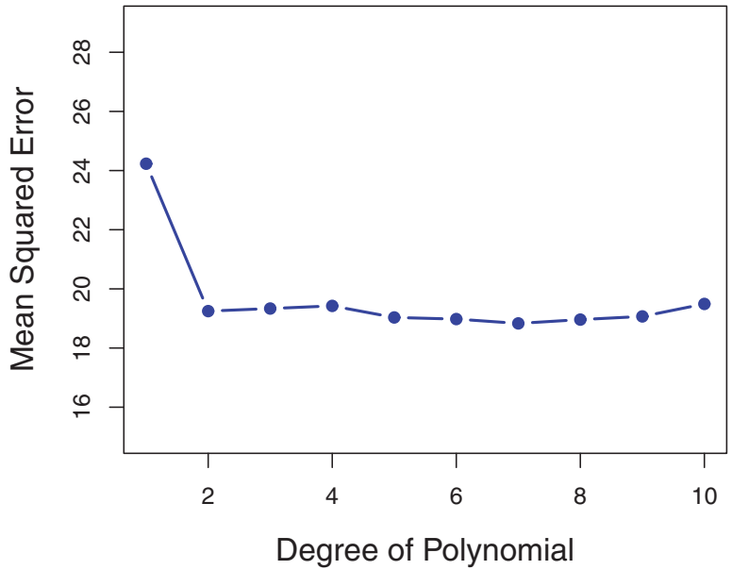


LEAVE ONE OUT CROSS VALIDATION

- With N data points, cut the training data into 2 parts, training and validation, N times. Each point is used for validation once.
- The FoM is estimated as the mean accuracy on each example. It is trivial for independent metrics (MSE, accuracy).
- Not so trivial for ranking or correlation metrics! One can use the LOOC predictions on the full dataset, but then predictions from different models are mixed.
- Each data point is used for estimation, estimation of accuracy will be as accurate as possible.
- It can take forever, it is practically impossible to use with most datasets.
- Note: there is no random process!
- Magic formula (only) for linear regression!



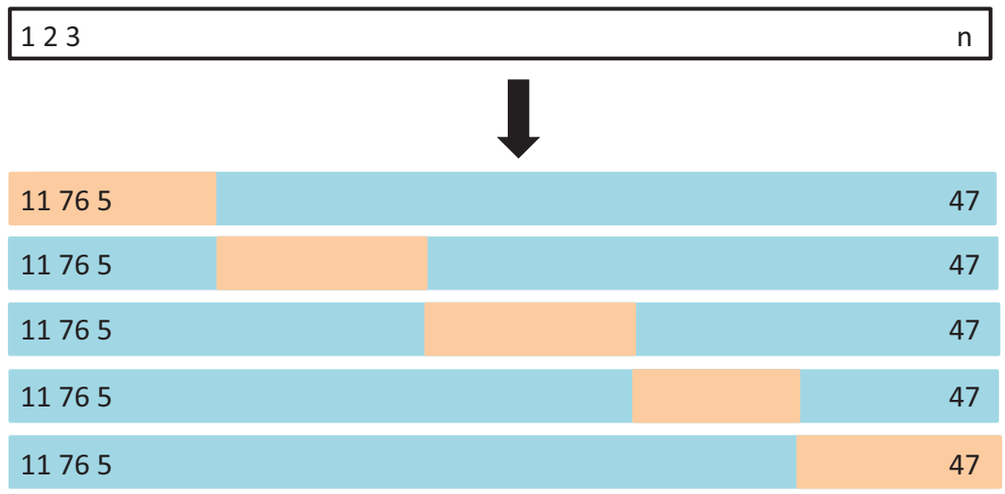
LOOCV

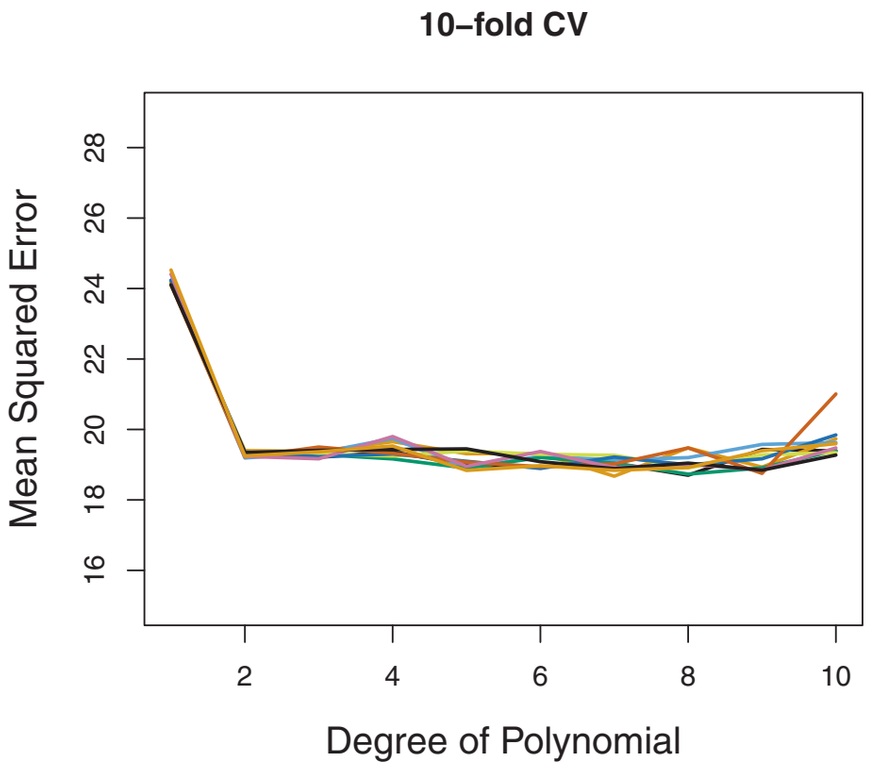


$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

K-FOLD CROSS VALIDATION

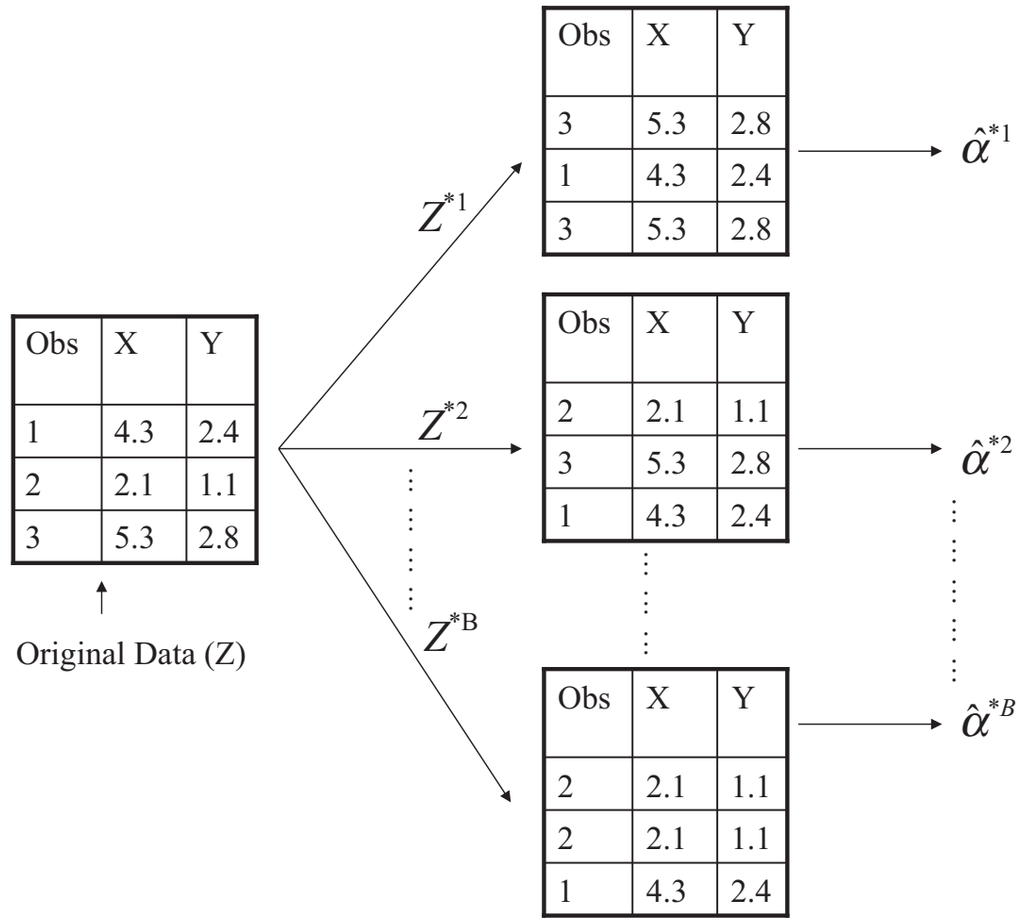
- With N data points, cut the training data into K parts, use K-1 for training and 1 for testing, K times. K is usually **3, 5,10.**
- The FoM is estimated as the mean accuracy on each set. Works for raking and correlation metrics too, because each validation set has a large number of points.
- 80% of the data points are used in training: the model will be close to be as good as possible
- Each data point is used for validation, estimation of the FoM will be good
- Usually can be done for large sets, the model needs to be trained only 5-10 times. One exception is image recognition, because then it is too expensive to even train 5 times (a week vs a month)
- Note: there is a random process of splitting, but variation is not as wild as in a train-validation split because in the end the same data points are used
- THE STANDARD





BOOTSTRAP

- We estimate parameters from data points, samples
- These samples are random examples from a true population. Parameters inferred from these data points will be somewhat different from the true relationship
- We need to estimate the uncertainties of the estimates
- E.g.: Linear regression coefficients. But generally we do not have formulas like in the case of linear regression (AUC), we need a more general approach.
- We can not generate new data points, but we can **resample**!
  - Select N data points from N data points with replacement.
  - Estimate the parameter on each example.
  - Calculate the standard error (or quantiles) of those estimates.
- Note **replacement**, and keep in mind.



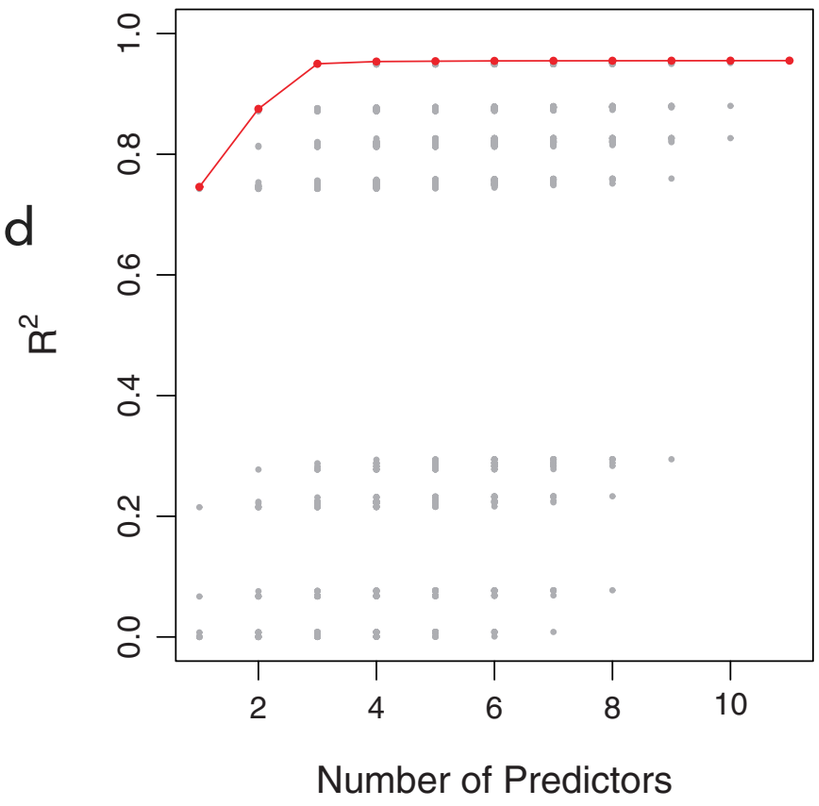
$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}$$

MODIFYING THE SOLUTION OF LINEAR REGRESSION

- Before: Least squares fit, why change it?
  - LS is the most accurate on the training data, but thats not our goal, our goal is accuracy of unseen new data
  - LS uses all predictors -> with many predictors it is hard to interpret the model. Can we select a good model with only a few predictors?
- Pursuing test accuracy:
  - LS will work well when the number of data points (N) is multiple orders of magnitude larger than the number of predictors (p)
  - When N is not much larger than p (e.g.:100 vs 10), then we might fit the noise to some extent, or in other words the coefficients will have large variance
  - When N<p, there is no unique LS fit

SUBSET SELECTION

- Fewer variables:
  - When the number of data points is low, a simpler model will not fit the noise that much, and it will be more accurate on unseen data.
  - It is also easier to interpret the model
- Brute force: best subset selection
  - 2^p choice. How to identify the best? Start from 0 variables and move to 1,2,...
  - Select the best model for each p. Comparing models with the same number of variables is OK.
  - Compare the best models with different p
    - This is tricky: the training error will always be smaller when using more variables. Cross validation error, AIC, BIC or adjusted R value
- Computational problem: d=10: 1000 runs, d=20: 1 million runs...



SUBSET SELECTION

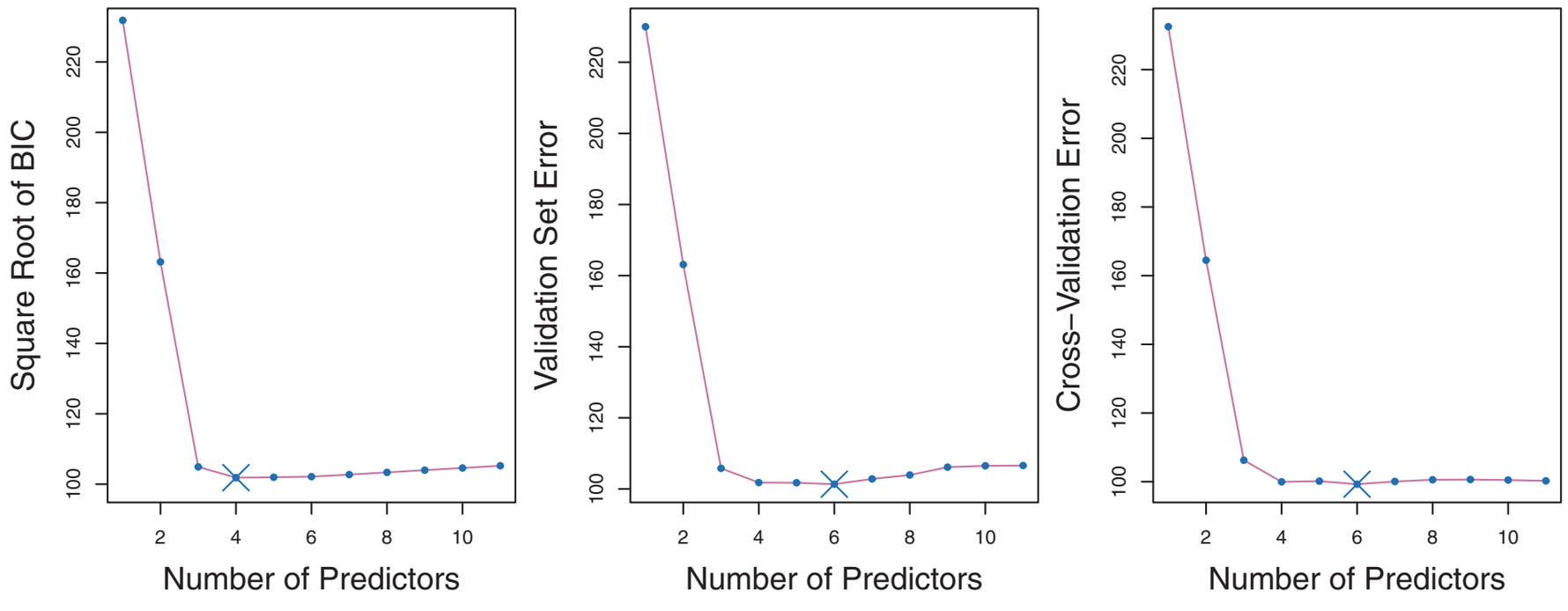
- Fast approach: stepwise greedy selection
  - Forward: Start with 0 variables, iteratively add the one which improves the training error the most (stop after a while)
  - Backward: Start with all variables, iteratively remove the one which degrades the training error the least
  - Compare the best models with different p. Cross validation error, AIC, BIC or adjusted R value
- Note, use different metric for selecting next variable and the best p to avoid overfitting!

$$C_p = \frac{1}{n}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right),$$

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right)$$

$$\mathrm{BIC} = \frac{1}{n\hat{\sigma}^2}\left(\mathrm{RSS} + \log(n)d\hat{\sigma}^2\right).$$

$$\text{Adjusted } R^2 = 1 - \frac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}.$$

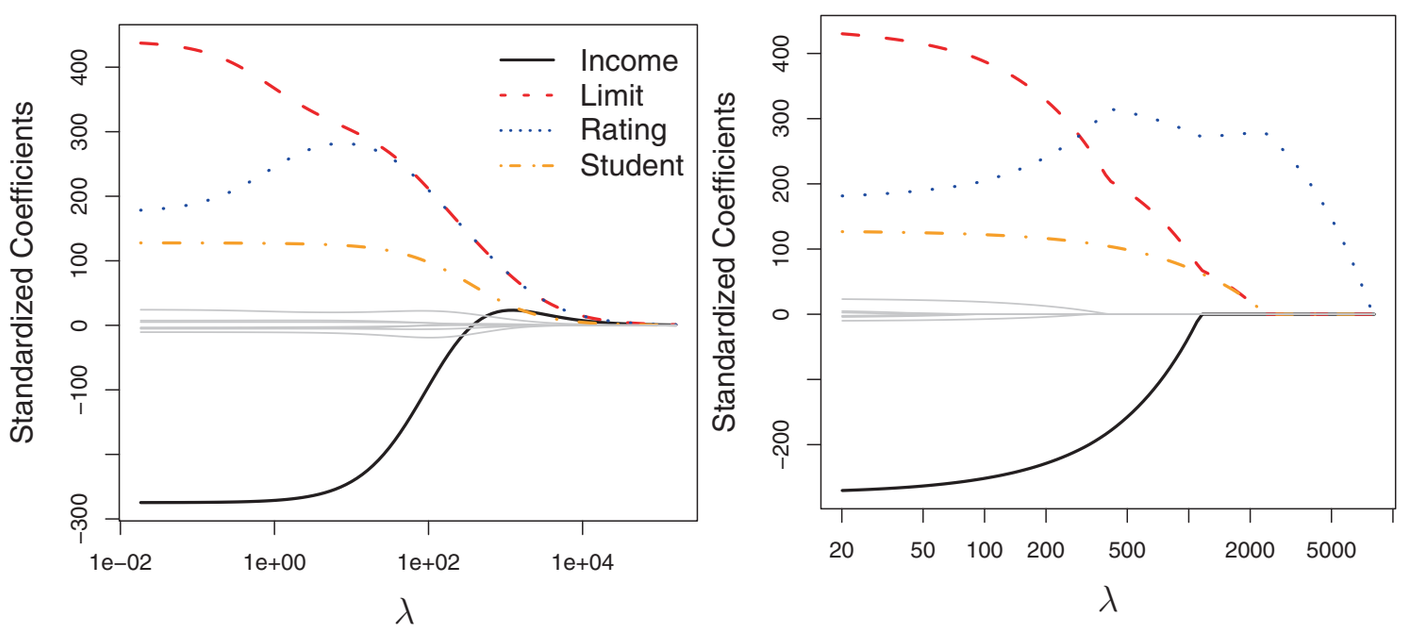


SHRINKAGE (RIDGE, LASSO)

- Shrinkage: a simpler model has **smaller** coefficients, instead of cancelling large coefficients (zero coeff is subset selection):
- Ridge-regression, L2 penalty (**Weight-decay**)
- Lasso-regression, L1 penalty. **Sparse regression**. Often results in 0 coefficients
- Elastic-net: both
- They both increase the training error! (but hope to improve generalisation by creating a simpler model)
- Essential for underdetermined problems! (SVD would not fail but not ideal)
- Note: The scale of predictive variables did not matter before! Now it does!

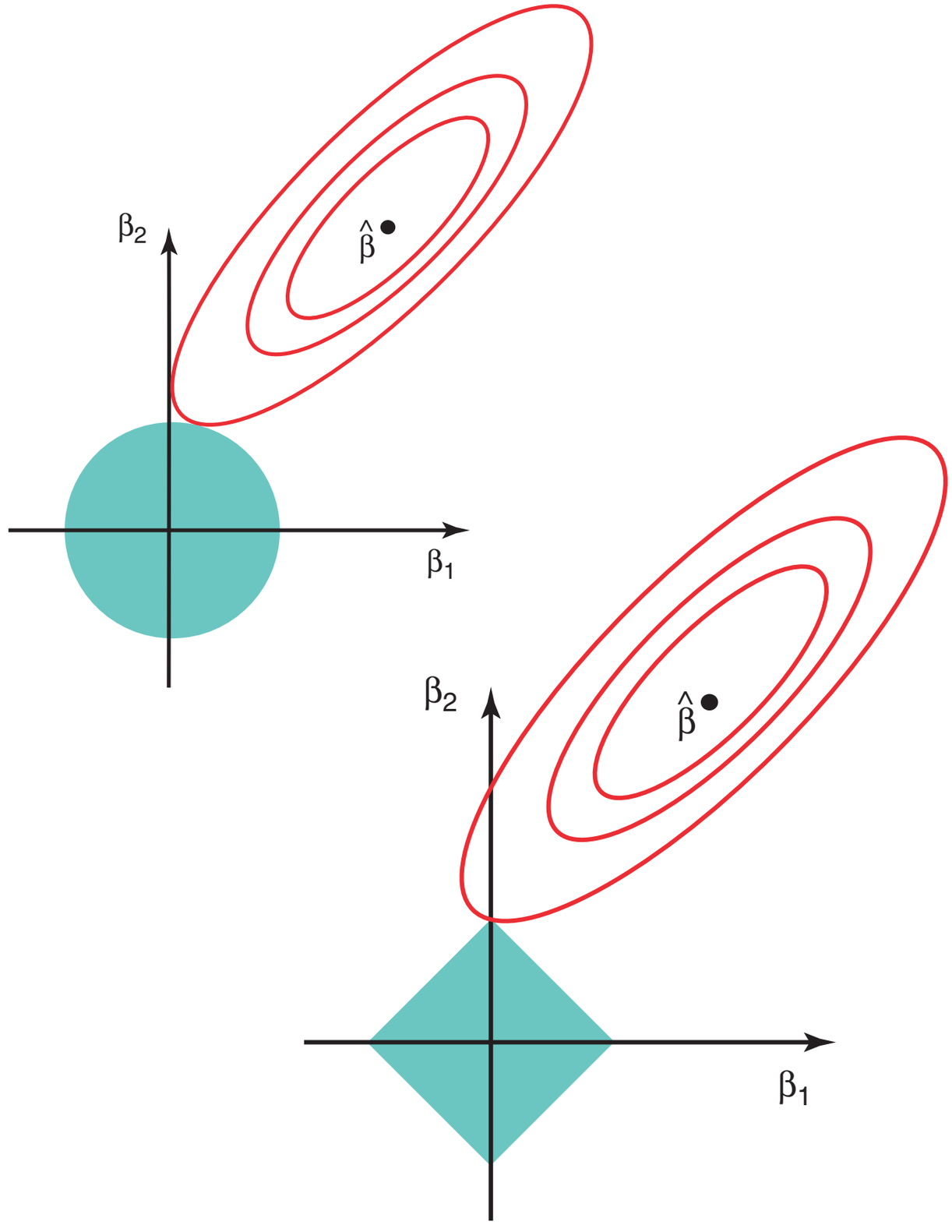
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2,$$

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$



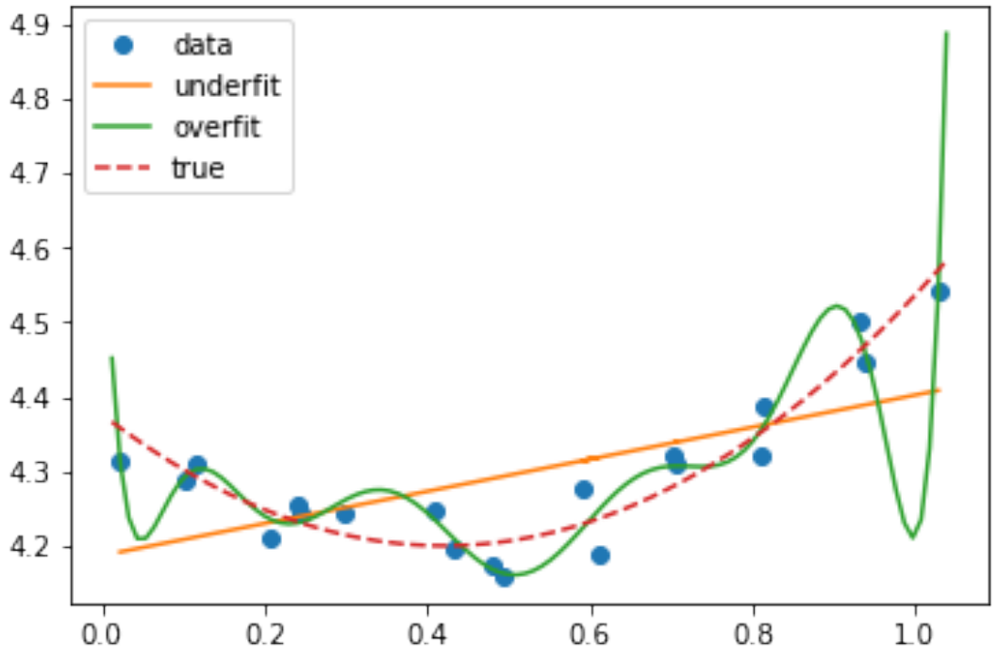
SHRINKAGE (RIDGE, LASSO)

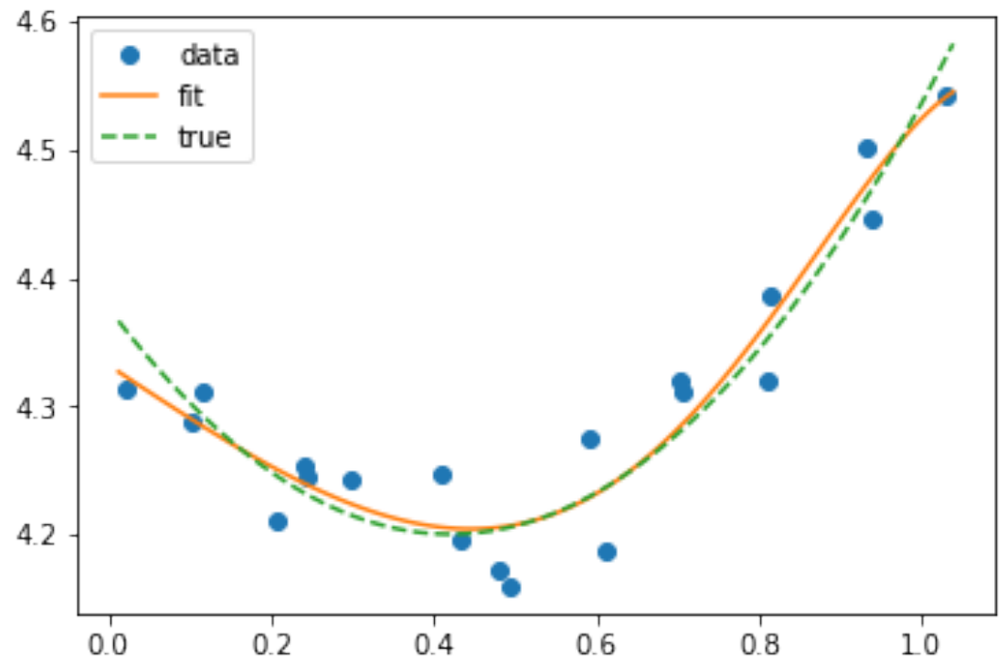
- Shrinkage: a simpler model has **smaller** coefficients, instead of cancelling large coefficients (zero coeff is subset selection):
- Ridge-regression, L2 penalty (**Weight-decay**)
- Lasso-regression, L1 penalty. **Sparse regression**. Often results in 0 coefficients
- Elastic-net: both
- They both increase the training error! (but hope to improve generalisation by creating a simpler model)
- Essential for underdetermined problems! (SVD would not fail but not ideal)
- Note: The scale of predictive variables did not matter before! Now it does!



REGULARISATION

- Subset selection (training error is larger!)
- Shrinkage (training error is larger!)
- Optimisation: reduce error/loss of training data
- Regularisation: reduce error/loss of unseen data, while increasing training error
  - Improving generalisation of a model: simpler, smoother models
  - Most often meant inside a function family, (but the structure of function can be thought of as a regularisation)
- Controlling the capacity of functions to match the problem: large capacity model + regularisation





REFERENCES

- ISLR chapter 6.