Test MSE vs training MSE

In Machine Learning, we want our models to generalize well to unseen data.

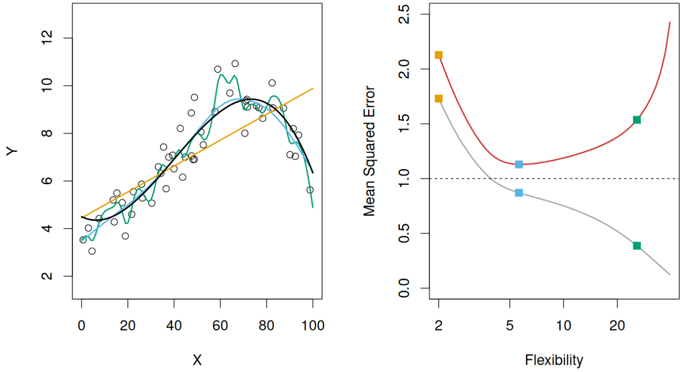
We want the test MSE to be low.

Unfortunately, test MSE is unknown to us: We use training MSE to fit our models.

Test MSE vs training MSE

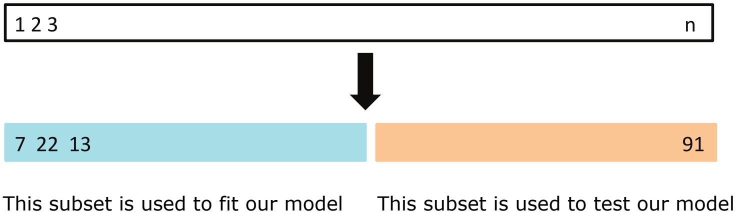
We can estimate test MSE by

- Making clever mathematical corrections to our training MSE.
- Leaving data out when we train our model and then use these unseen data to estimate the test MSE.



We seek to avoid *overfitting* to the training data. By using *validation sets*, we can avoid overfitting.

Validation set: A first attempt

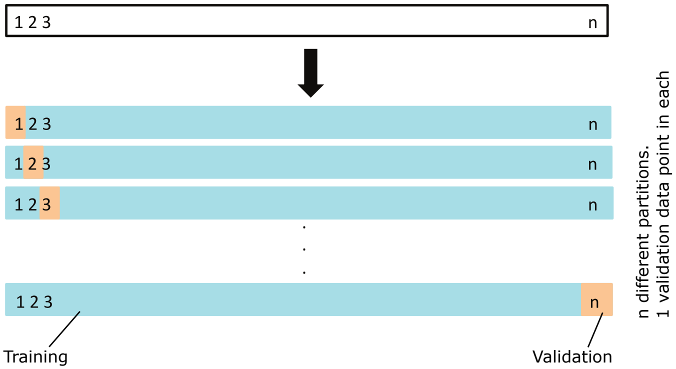

1 2 3
n
7 22 13
91
This subset is used to fit our model
This subset is used to test our model

Problems:

- Performance depends on what data points are in the two sets.
- Performance is underestimated because many data points are not used for the training.

One solution: Sample training and validation sets many times.

Leave-one-out crossvalidation (LOOCV)



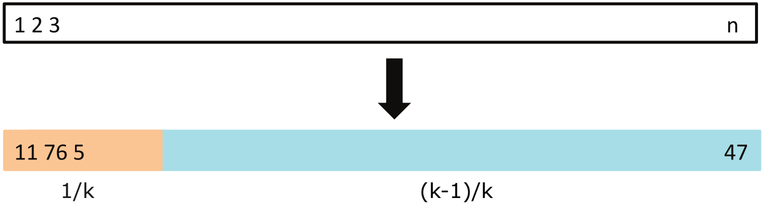
For each partition, i, the test MSE is $(y_i - \hat{y}_i)^2$.
The LOOCV estimate for the test MSE is the average of the test error estimates:

$$MSE = \frac{1}{n} \sum_{i=n}^{n} (y_i - \hat{y}_i)^2$$

Problem: Computationally expensive to fit n times.

k-fold crossvalidation

In k-fold crossvalidation, we leave out $1/k^{\text{th}}$ of the data at the time.

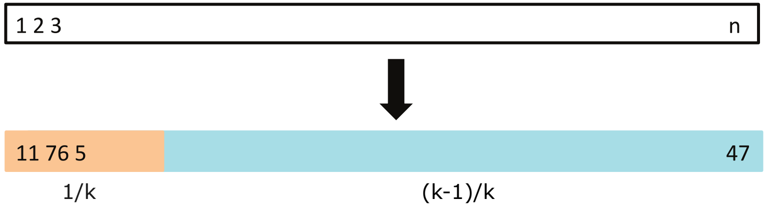


In i'th held-out fold, we compute MSE_i. At the end, average over the k obtained MSEs:

$$\frac{1}{k}\sum_{i=1}^{k}\text{MSE}_k$$

Make sure to randomly allocate observations to folds, for instance shuffle your data before creating the folds.

k-fold crossvalidation



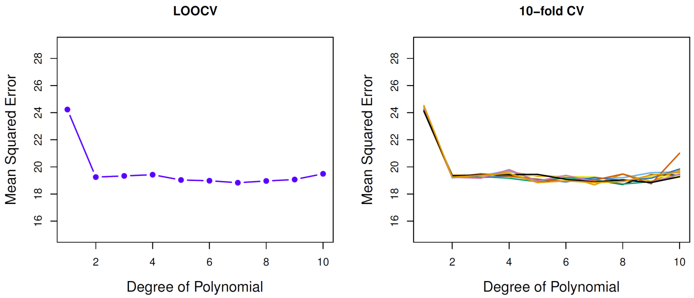
At the extremes we have

- N-fold cross-validation, i.e. leave-one-out!
- 2-fold cross-validation, where data is split in half.

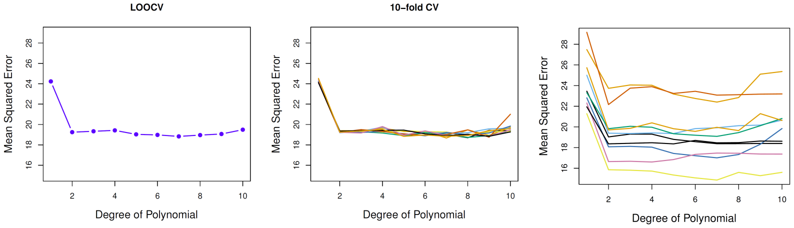
Need to fit the model as many times as there are folds.

For LOOCV there is just one way of allocating observations to folds.

For k-fold cross-validation there is randomness in how observations are allocated to $(k < n)$ folds, so the cross-validation MSE curve will be different if you do it again.



LOOCV, 10-fold, validation set



On the number of folds

Common choices are $k = 5$ or $k = 10$ folds.

It is hard to say something conclusive!

Fewer folds give fewer observations for training, which could give a more biased test error estimate: The trained method probably performs unnecessarily poorly, so could overestimate test error.

On the number of folds

What is the variance of the test error with k folds?

It is an average of k estimated errors. These errors are positively correlated, since trained on very similar data (Greater k, more similar).

Result: Variance is larger than if there were no correlations. See this by using mathematical identities to get..

$$\begin{aligned}\operatorname{Var}\left(\frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i\right) &= \frac{1}{k^2}\operatorname{Var}\left(\sum_{i=1}^{k}\mathrm{MSE}_i\right)\\ &= \frac{1}{k^2}\sum_{i=1}^{k}\left(\operatorname{Var}\mathrm{MSE}_i + 2\sum_{j=1}^{i}\operatorname{Cov}(\mathrm{MSE}_i,\mathrm{MSE}_j)\right)\end{aligned}$$

Regression models: Strategies with “many” features

Regression models: strategies with “many” features

With high number of features $p \approx n$ there is not enough information in the data to estimate the regression curve well.

With more parameters than features $p > n$ the MLE does not even exist.

What can we do?

Regression models: strategies with “many” features

Model selection

- Include only a subset of the features in the model.

Shrinkage (also referred to as Regularisation)

- Estimate parameters by minimising the log-likelihood *added a penalty term*.
- This corresponds to constraining the parameters.

Dimensionality reduction of the feature space

- We discuss PCA later in the course.

Feature selection strategies

Strategies for (automatic) feature selection:

- Best subset – expensive, sometimes even impossible!
- Forward selection (start by including no/few variables)
- Backward selection (start by including all/many variables)
- Alternating between forwards and backward selection

Models can be compared, for instance, by AIC, BIC, or test error.

Two nested models can be compared by F-tests; rarely used in automated selection above, as we run into issues with multiple testing.

Use model inspection to see if you have a well-fitting model.

An Information Criterion (AIC)

$$\mathrm{AIC} = -2\log\hat{L} + 2p,$$

where p is the number of parameters under the model and $\hat{L}$ is the maximised likelihood function.

We seek the model with the smallest AIC (in a group of pre-specified models)

Small means better, but gives *no* indication of goodness of the fit.

A similar criterion is

$$\mathrm{BIC} = -2\log\hat{L} + p\log n.$$

Stepwise selection (here with AIC)

Backward stepwise selection

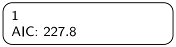
1. Decide on the largest model of interest
2. Iteratively remove the *least relevant* feature (largest drop in AIC)
3. Stop if removing a feature does not improve the model.

Forward stepwise selection

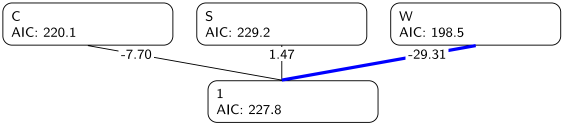
1. Decide on the smallest model of interest – typically including no features and just the intercept.
2. Iteratively add the *most relevant* feature (largest drop in AIC)
3. Stop if adding a feature does not improve the model.

Note: Slightly different procedures than in Chapter 6 of ISLwR.

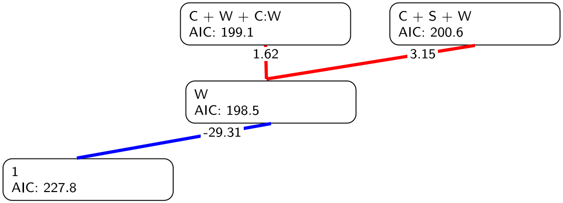
Forward selection by AIC



Forward selection by AIC



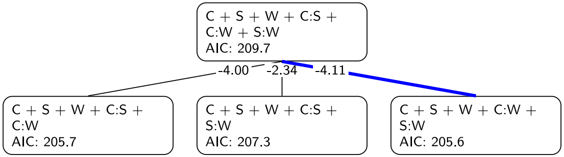
Forward selection by AIC



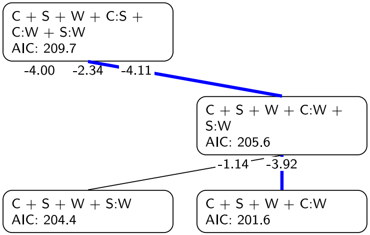
Backward selection by AIC

C + S + W + C:S +
C:W + S:W
AIC: 209.7

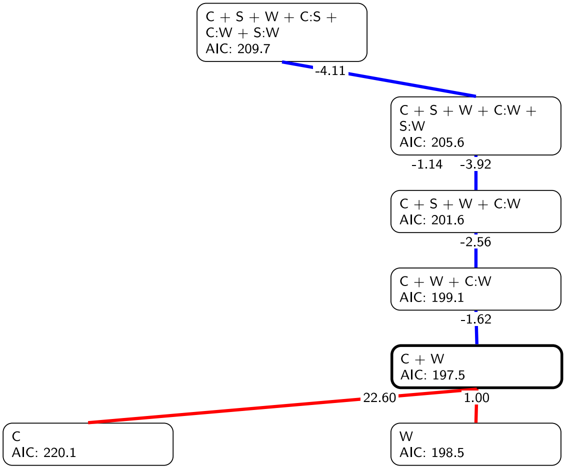
Backward selection by AIC



Backward selection by AIC



Backward selection by AIC



Thoughts on automatic model selection

From an *interpretational* perspective you may like to

- keep certain variables in the model, even if they are not significant.
  Example: keep variables of interest to the research question.

- remove variables, even if they are significant.
  Example: complex interactions that complicate interpretations.

Estimating test error directly

With validation set approach or the cross-validation set approach, we estimate test error directly and can use it for selecting between models.

The estimated MSE can be used also to select between different types of machine learning models, where AIC/BIC often cannot.

The *one-standard-error rule* chooses the simplest model among those within a standard error of the best model found by cross-validation.

Shrinkage methods

Shrinkage methods help us estimate coefficients β in a constrained way.

Shrinkage helps us keep the β small.

It turns out that this reduces coefficient variance and can improve the fit.

2 key shrinkage methods:

- Ridge regression,
- Lasso.

Shrinkage methods: Ridge regression

Coefficients β are estimated by minimising

$$\mathrm{RSS} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\lambda\|\beta\|_2^2}$$

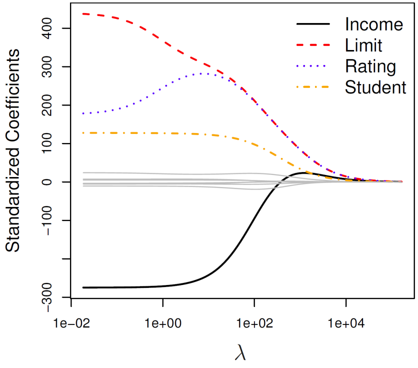
Here, $\lambda \geq 0$ is a tuning parameter.

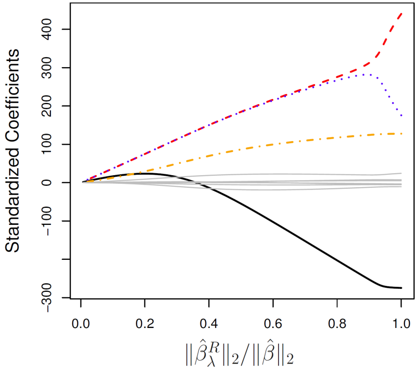
Final term is really a penalty. λ controls penalty size.

Note that the intercept β_0 is not regularized!

- $\lambda = 0$ no penalty, so β_λ^R are just the MLE estimates $\hat{\beta}$.
- $\lambda \to \infty$ would shrink all coefficients to 0.

Ridge regression: Effect on the coefficients





Shrinkage methods: Lasso

Coefficients β are estimated by minimising

$$\mathrm{RSS} + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{\lambda\|\beta\|_1}$$

Again, the final term is really a penalty. λ controls penalty size.

As the penalty increases, often more and more coefficients are set to zero, wich corresponds to eliminating some features from the model.

Lasso vs. Ridge

One can show that Ridge Regression coefficient estimates solve

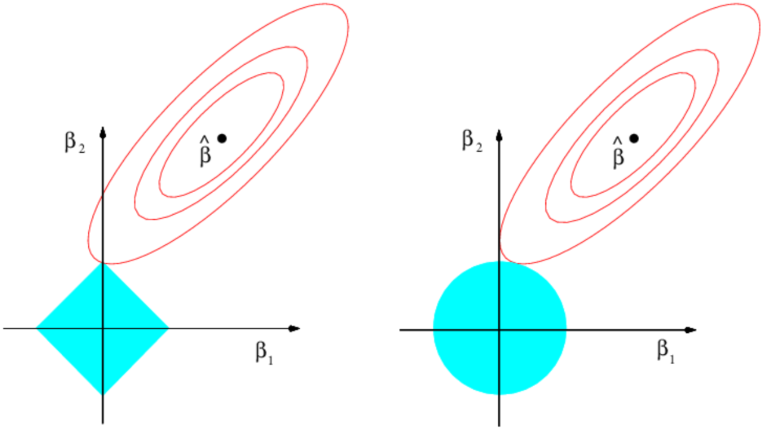
$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s, \tag{6.9}$$

And Lasso coefficient estimates solve

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s \tag{6.8}$$

s can be thought of as a “budget” for $\sum_{i=1}^{p} \beta_i$.

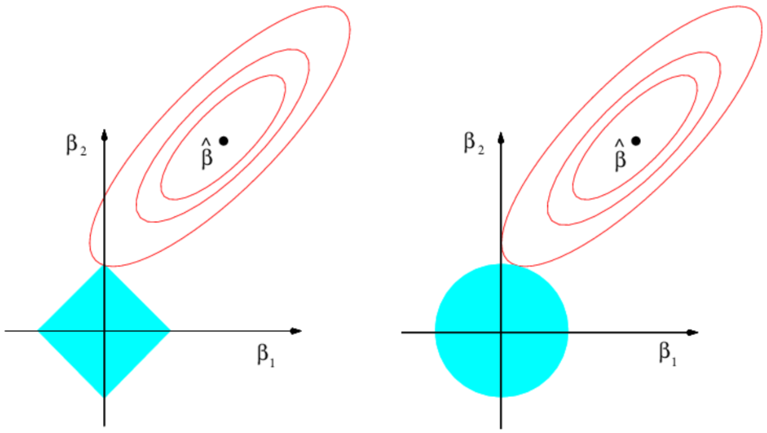
Lasso vs. Ridge regression



Budget for Lasso

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s \qquad (6.8)$$

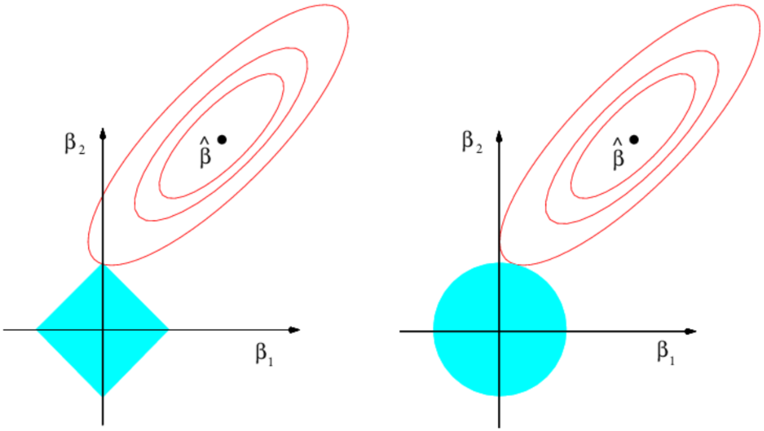
Lasso vs. Ridge regression



Budget for Ridge

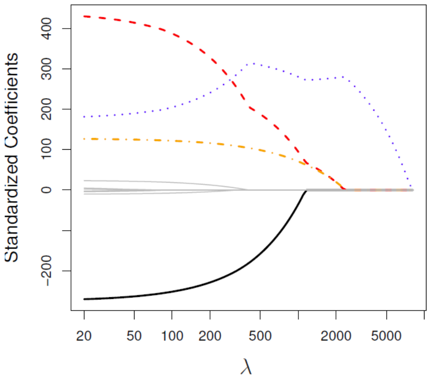
$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \le s, \tag{6.9}$$

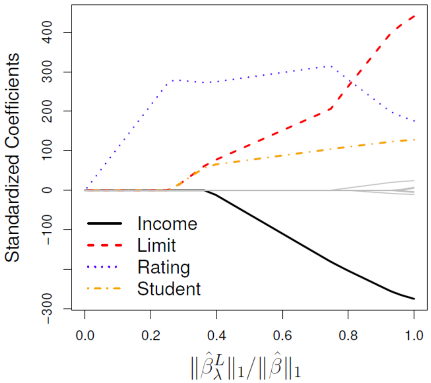
Lasso vs. Ridge regression



Q: The contour is more likely to hit the "budget corner" for Lasso. What happens when this happens?

Lasso: Effect on the coefficients





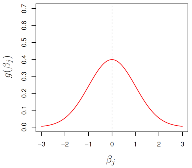
Final comments on shrinkage

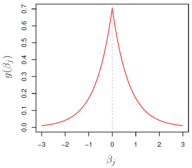
Choose the regularization parameter λ by crossvalidation.

Common to standardize all features before regularizing, since the “importance” of a feature is sensitive to the scale of features. (For least squares, scaling features just scales the coefficients.)

What are the estimates compared to OLS? Ridge essentially “scales” everything down, whereas Lasso “shifts” towards zero and truncates small enough coefficients.

Posterior mode interpretation for ridge regression and lasso.


$g(\beta_j)$
β_j


$g(\beta_j)$
β_j