Cross-validation and prediction error

Patrick Breheny

February 25, 2025

Introduction

- Today we will discuss the selection of λ as well as the related but somewhat different task of estimating (and obtaining confidence intervals for) the prediction error of a model
- For the lasso, both of these involve tend to revolve around cross-validation, although we will discuss a few different approaches

Degrees of freedom

- In our discussion of ridge regression, we used information criteria to select λ
- All of the criteria we discussed required an estimate of the degrees of freedom of the model
- For linear fitting methods, we saw that $\mathrm{df} = \mathrm{tr}(\mathbf{S})$
- The lasso, however, is not a linear fitting method; there is no exact, closed form solution to $\mathrm{Cov}(\mathbf{y}, \hat{\mathbf{y}})$

Degrees of freedom for the lasso

- A natural proposal would be to use $\mathrm{df}(\lambda) = \|\widehat{\boldsymbol{\beta}}(\lambda)\|_0$, the number of nonzero coefficients
- From one perspective, this might seem to underestimate the true degrees of freedom, as the variables were not prespecified
- For example, in our forward selection example from our first class, we selected 5 features but the true df was ≈ 19
- On the other hand, shrinkage reduces the degrees of freedom in an estimator, as we have seen in ridge regression; from this perspective, $\|\widehat{\boldsymbol{\beta}}(\lambda)\|_0$ might seem to overestimate the true degrees of freedom

Degrees of freedom for the lasso (cont'd)

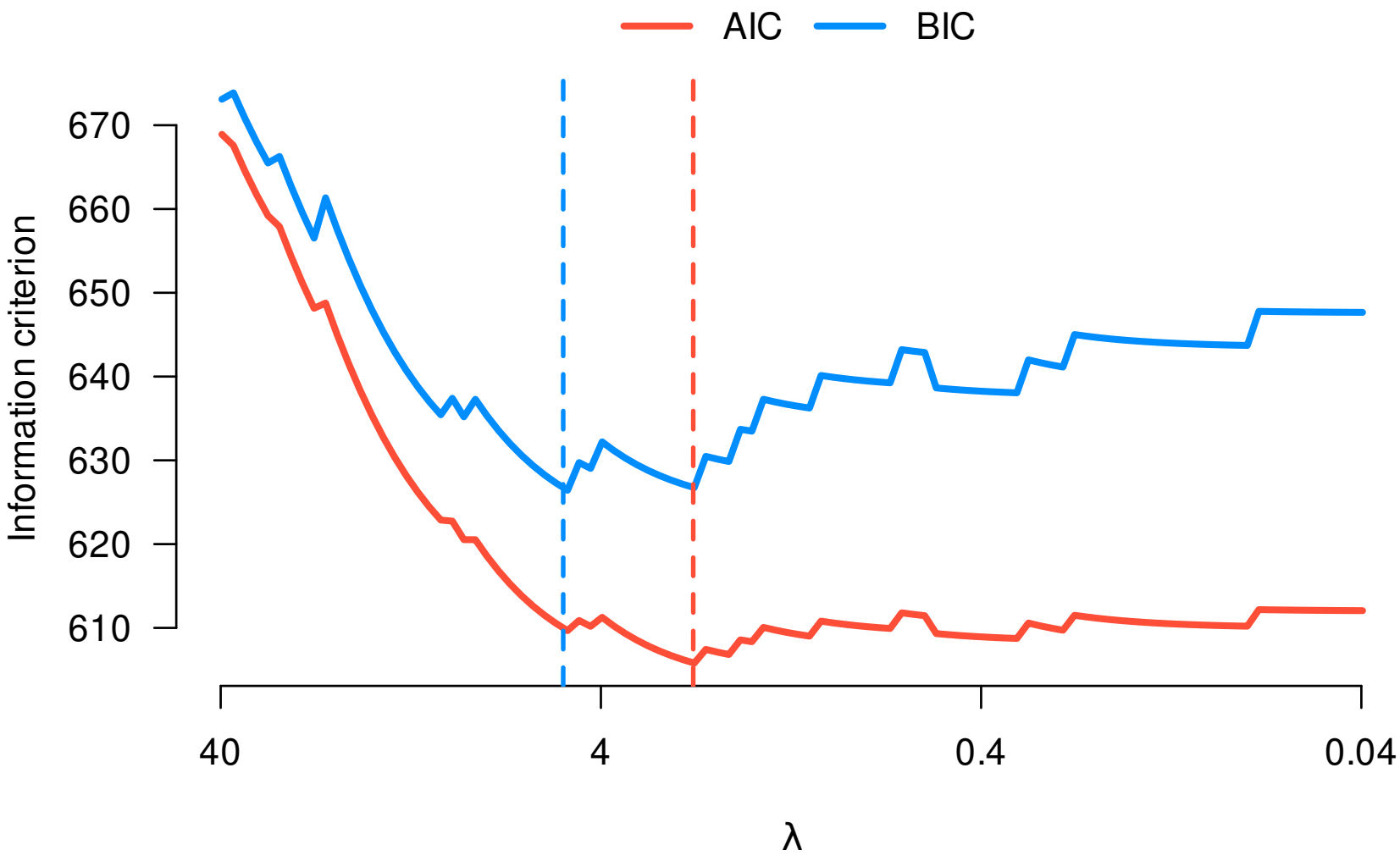
- Surprisingly, it turns out that these two factors exactly cancel and $\mathrm{df}(\lambda) = \|\widehat{\boldsymbol{\beta}}(\lambda)\|_0$ can be shown to be an unbiased estimate of the lasso degrees of freedom
- Given this estimate, we can then use information criteria such as BIC for the purposes of selecting λ

ncvreg

- To illustrate, we will use the `ncvreg` package to fit the lasso path
- The primary purpose of `ncvreg` is to provide penalties other than the lasso, which we will discuss in our next topic
- However, it provides a `logLik` method, unlike `glmnet`, so it can be used with R's `AIC` and `BIC` functions:

```
fit <- ncvreg(X, y, penalty="lasso")
AIC(fit)
BIC(fit)
```

AIC, BIC for pollution data



Remarks

- As we would expect, BIC applies a stronger penalty for overfitting and chooses a smaller, more parsimonious model than does AIC
- The main advantage of AIC and BIC is that they are computationally convenient: they can be calculated using the fit of the lasso model at very little computational cost
- The primary disadvantage is that both AIC and BIC rely on a number of asymptotic approximations that can be quite inaccurate for high-dimensional data

Cross-validation: Introduction

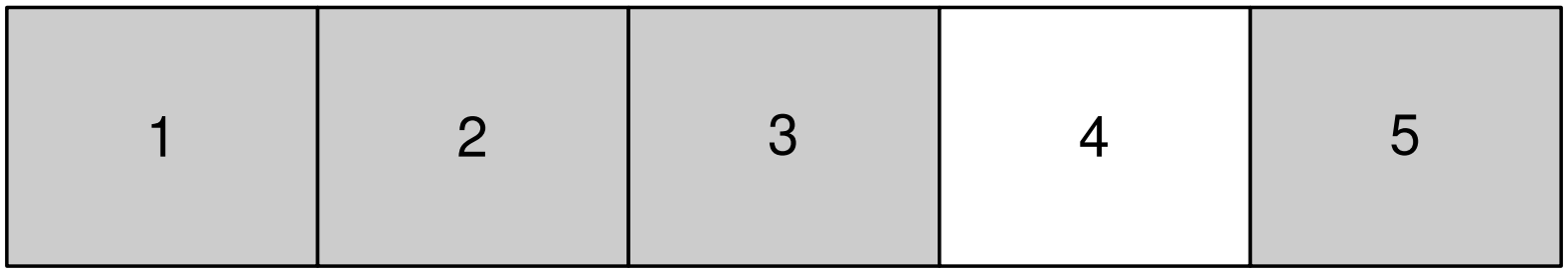
- As we have discussed, a reasonable approach to selecting λ in an objective manner is to choose the value of λ that yields the greatest predictive power
- An alternative to the approximations of AIC and BIC is to assess predictive power more directly and empirically through a technique called cross-validation
- Cross-validation is much more reliable, although it comes at an added computational cost

Sample splitting

- Using the observed agreement between fitted values and the data is too optimistic; we require independent data to test predictive accuracy
- One solution, known as *sample splitting*, is to split the data set into two fractions, a training set and test set, using one portion to estimate $\widehat{\boldsymbol{\beta}}$ (i.e., "train" the model) and the other to evaluate how well $\mathbf{X}\widehat{\boldsymbol{\beta}}$ predicts the observations in the second portion (i.e., "test" the model)
- The problem with this solution is that we rarely have so much data that we can freely part with half of it solely for the purpose of choosing λ

Cross-validation

To finesse this problem, *cross-validation* splits the data into K folds, fits the data on $K-1$ of the folds, and evaluates prediction error on the fold that was left out



Common choices for K are 5, 10, or n (also known as leave-one-out cross-validation)

Cross-validation: Details

(1) Specify a grid of regularization parameter values $\Lambda = \{\lambda_1, \ldots, \lambda_K\}$

(2) Divide the data into V roughly equal parts $D_1, \ldots, D_V$

(3) For each $v = 1, \ldots, V$, compute the lasso solution path using the observations in $\{D_u, u \neq v\}$

(4) For each $\lambda \in \Lambda$, compute the fold-specific cross-validation error:

$$\mathrm{CV}_v(\lambda) = \frac{1}{n_v} \sum_{i \in D_v} \{y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_{-v}(\lambda)\}^2,$$

where n_v is the number of observations in D_v, then average these values:

$$\mathrm{CV}(\lambda) = \frac{1}{V} \sum_{v=1}^{V} \mathrm{CV}_v(\lambda)$$

Remarks

- Letting λ_{CV} denote the value that minimizes $\text{CV}(\lambda)$, we would then use $\widehat{\boldsymbol{\beta}} \equiv \widehat{\boldsymbol{\beta}}(\lambda_{\text{CV}})$ as the estimator of the regression coefficients
- An equivalent formulation of the cross-validation error is to first obtain the cross-validated predictions

$$\hat{\mu}_i(\lambda) = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_{-v(i)}(\lambda) \quad \text{for } i = 1, 2, \ldots n$$

where $v(i)$ denotes the fold containing observation i, and the cross-validation error is

$$\text{CV}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \{y_i - \hat{\mu}_i(\lambda)\}^2,$$

Variability of CV estimates

- With this second formulation, we can see more clearly that cross-validation error is a mean, and its standard error is

$$\mathrm{SE}_{\mathrm{CV}}(\lambda) = \frac{\mathrm{SD}_{\mathrm{CV}}(\lambda)}{\sqrt{n}},$$

  where $\mathrm{SD}_{\mathrm{CV}}(\lambda)$ denotes the sample standard deviation of

$$\{y_i - \hat{\mu}_i(\lambda)\}^2$$

- Note that:
  - Regardless of the number of cross-validation folds, each observation in the data appears exactly once in a test set
  - The estimand here is PE, not PE_X

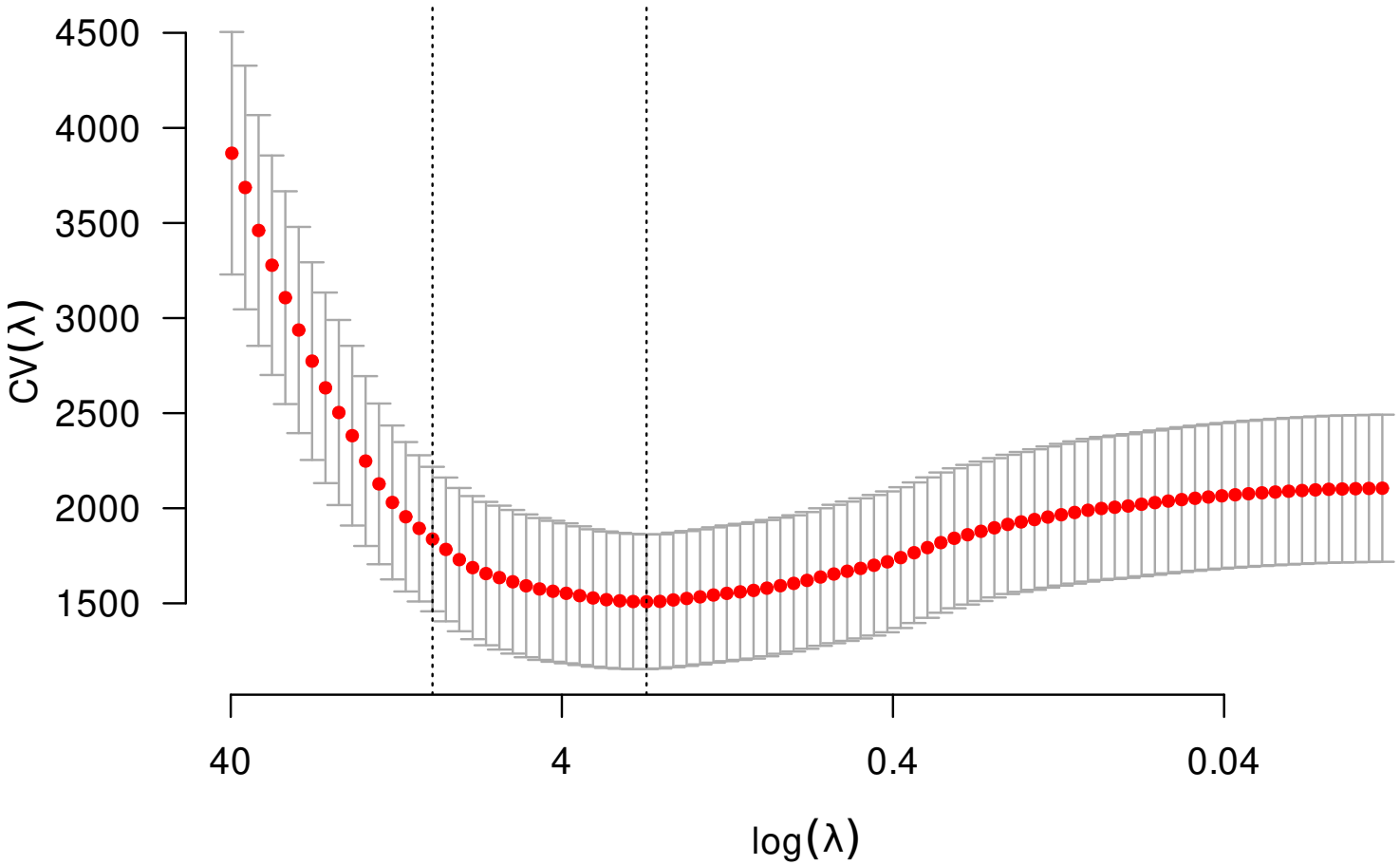
glmnet

- The cross-validation procedure described in this section, along with the estimates of $\mathrm{CV}(\lambda)$ and its standard error, are implemented in `glmnet` and can be carried out using

```
cvfit <- cv.glmnet(X, y)
plot(cvfit)
```

By default, `cv.glmnet` uses $V = 10$ folds, but this can be changed through the `nfolds` option.

CV plot for lasso: Pollution data



Intervals are $\pm 1\text{SE}$

Remarks

- The value $\lambda = 1.84$ minimizes the cross-validation error, at which point 9 variables are selected
- However, as the confidence intervals show, there is substantial uncertainty about this minimum value
- A fairly wide range of λ values ($\lambda \in [0.12, 9.83]$) yield $\mathrm{CV}(\lambda)$ estimates falling within $\pm 1\mathrm{SE}_{\mathrm{CV}}$ of the minimum
- This is almost always the case in model selection: a large number of models could reasonably be considered the "best" model, subject to random variability

Randomness and cross-validation

- Note that $\mathrm{CV}(\lambda)$, and hence $\widehat{\boldsymbol{\beta}}$, will change somewhat depending on the random folds
- To avoid this, some people carry out *repeated cross-validation*, and select λ according to the average CV error
- Another option is to carry out n-fold cross-validation, in which there is only one way to select the fold assignments
- It is important to realize, however, that neither of these approaches does anything to eliminate actual uncertainty with respect to the selection of λ (neither does setting the seed!)

Nested cross-validation and conformal prediction

- Some recent work has shown that these simple SE calculations have a tendency to be too small, and the confidence intervals for the true prediction error have lower than advertised coverage
- Various solutions to this problem have been proposed, including a nested cross-validation scheme as well as a very different approach altogether called conformal prediction, although both these methods are much more computationally intensive than ordinary CV
- Furthermore, regardless of what exactly CV is estimating, or how accurately it estimates it, picking the λ value that minimizes CV is usually reasonable (which is usually the primary concern in high-dimensional regression)

Coefficient of determination

- A related goal is estimating the proportion of variance in the outcome that can be explained by the model
- This quantity, familiar from classical regression, is known as the *coefficient of determination* and denoted R^2
- The coefficient of determination is given by

$$R^2 = 1 - \frac{\mathrm{Var}(Y|\mathbf{X})}{\mathrm{Var}(Y)}$$

  - Estimation of $\mathrm{Var}(Y)$ is straightforward
  - Estimation of $\mathrm{Var}(Y|\mathbf{X})$ is (more or less) what CV estimates

R^2: Calculation in R

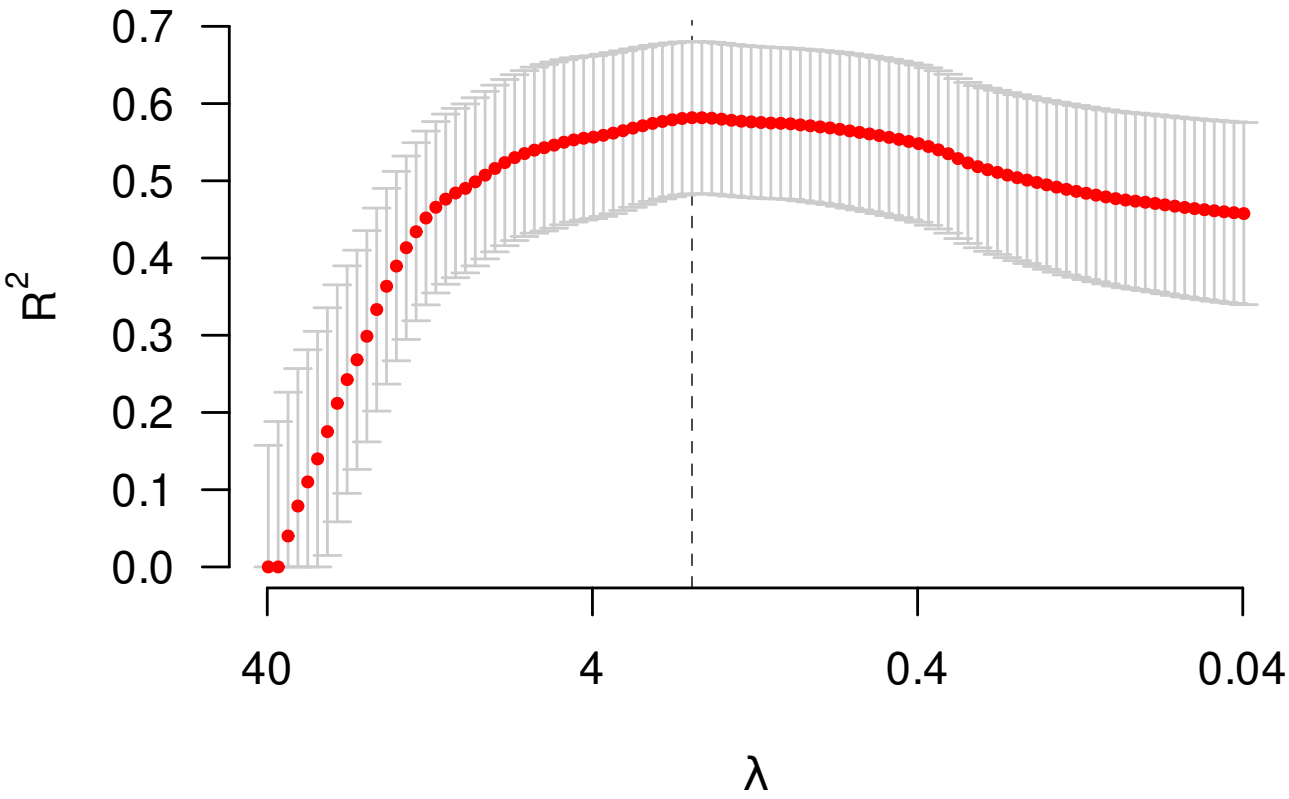
- Once cross-validation is done, calculating R^2 is straightforward
- With `glmnet`:

```
cvfit <- cv.glmnet(X, y)
rsq <- 1-cvfit$cvm/var(y)
```

- Also, the coefficient of determination is available as a plot type in `ncvreg`:

```
cvfit <- cv.ncvreg(X, y, penalty="lasso")
plot(cvfit, type="rsq")
```

R^2 plot: Pollution data



Only a small amount of explained variability comes from pollution: $\max R^2 = 0.58$ with the pollution variables; $\max R^2 = 0.56$ without them

summary.cv.ncvreg

ncvreg also provides a summary() method for its cross-validation object that reports all of this information:

```
summary(cvfit)
# lasso-penalized linear regression with n=60, p=15
# At minimum cross-validation error (lambda=1.4949):
# -------------------------------------------------
#   Nonzero coefficients: 11
#   Cross-validation error (deviance): 1628.06
#   R-squared: 0.57
#   Signal-to-noise ratio: 1.34
#   Scale estimate (sigma): 40.349
```