LECTURE 6: EFFECTIVE DEGREES OF FREEDOM

■ REFERENCES

Lecture notes, handouts + Shalizi Sec 1.5.3.2 and Sec 3.4.3.

■ OVERVIEW

The notion of *effective degrees of freedom* gives us a quantitative measure of the complexity of an estimator, and a means of comparing error curves for different methods. Degrees of freedom is also directly related to the difference between expected test and training errors, the so-called *optimism* of the estimator; hence, by estimating the degrees of freedom we can construct estimates of expected test error as faster alternatives to cross-validation.

■ QUESTIONS

1. How does one define the degrees of freedom of a regression model? What assumptions are we making here?
2. What are the degrees of freedom of a linear smoother? How is this useful in practice?
3. How do you estimate the degrees of freedom if they cannot be calculated analytically?
4. What is the difference between in-sample and out-of-sample prediction errors?

■ DEFINITIONS AND NOTATIONS

1. Degrees of freedom
2. Linear smoother
3. Resampling residuals

4. Optimism and Mallow's C_p statistic

5. Generalized cross-validation.

■ TOPICS

1. Degrees of freedom: Motivation, definition and examples

2. Estimating degrees of freedom via the bootstrap

3. Using degrees of freedom to estimate the prediction error

■ WHAT'S NEXT?

Lecture 7: More on Kernel Regression (Shalizi Chapter 4 and Sections 1.5 and 8.3)

Review: Kernel Smoothers, Splines and Tuning Parameters

Text reference: Shalizi Sec 1.5.3.2 and Sec 3.4.3

- A linear smoother has the form $\hat{r}(x) = \sum_i \ell_i(x) Y_i$.
  - In particular, $\hat{\vec{Y}} = L\,\vec{Y}$ at the observed X_i's.
- For a kernel smoother

$$\ell_i(x) = \frac{K((x - X_i)/h)}{\sum_{j=1}^n K((x - X_j)/h)}$$

  - The kernel smoother is a weighted average of the Y_i's in the neighborhood of x.
  - The key choice is the smoothing parameter h.
- A regression or smoothing spline is like linear regression with the X_{ij}'s replaced by the basis functions $B_j(X_i)$

$$\hat{r}(x) = \sum_j \hat{\beta}_j B_j(x).$$

  - The basis functions are obtained using B-splines determined by the observed X_i's.
  - For *regression splines*, the key choice is the placement and number of B-splines; $\hat{\beta}$ is obtained via least squares.
  - For *smoothing splines*, the key choice is the smoothing parameter λ, which penalizes the function for excess curvature; $\hat{\beta}$ is obtained via penalized least squares.

R Demo 6.1

(a) Simulate iid data according to the model $Y = X \cdot \sin(X) + \epsilon$, where $\epsilon \sim N(0, 1)$ and $X \sim U(0, 6\pi)$.

(b) Fit a *kernel regression* to the data. Use a Gaussian kernel and manually choose the kernel bandwidth; for example, try h=5, 1, and 0.1. What do you see?

(c) Choose six evenly spaced knots. Generate B-spline basis. Fit *regression splines* to the data. Comment on the choice of knots. How about using natural splines?

(d) Fit *smoothing splines* to the data. Choose $\lambda = 0.01, 2 \times 10^{-5}$, and 10^{-20}. What do you see?

To choose the tuning parameters in these estimators, we can use techniques such as K-fold cross-validation, leave-one-out cross-validation (LOOCV), Mallow's C_p-statistics, and generalized cross-validation (GCV). As we will see later, these approaches are closely related for linear smoothers.

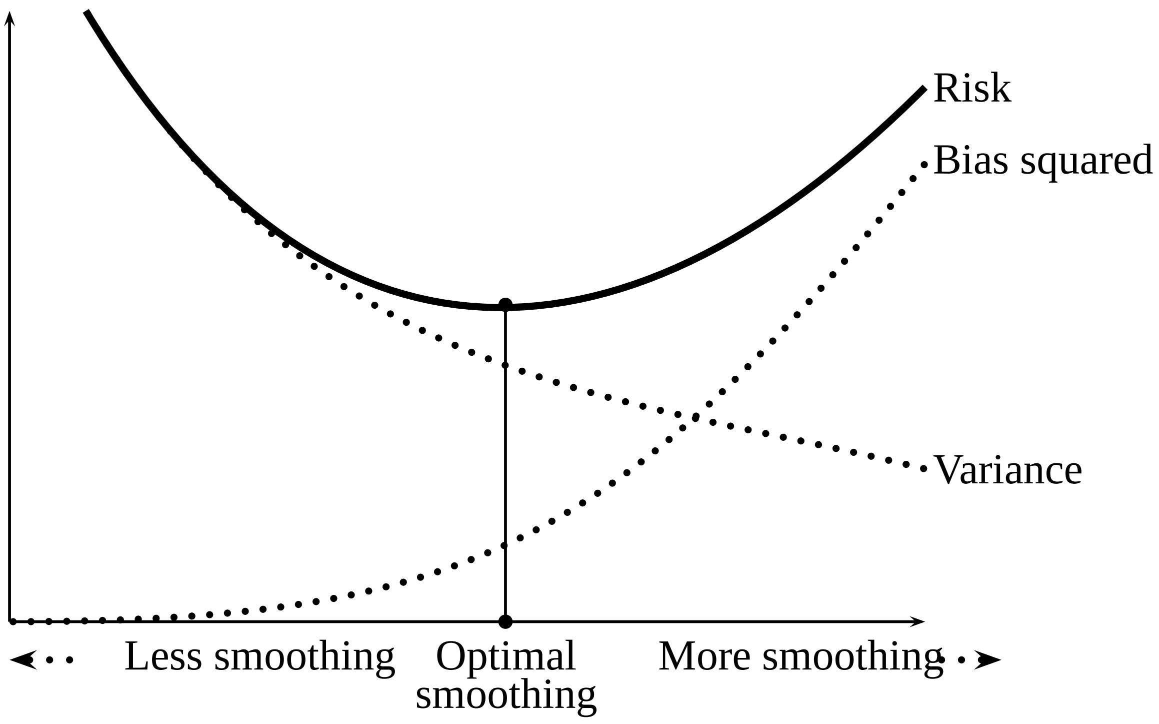

Figure 1: The familiar bias–variance tradeoff in curve estimation. But how do we choose the best amount of smoothing?

Effective Degrees of Freedom

Motivation

- So far we have seen several methods for estimating the underlying regression function $r(x) = \mathbb{E}[Y|X = x]$: linear regression, k-nearest-neighbors, kernel smoothing and splines. Often, in a predictive setting, we want to compare (estimates of) test error between such methods, in order to choose the best method.

- We have learned one method for model selection: cross-validation. But in a sense, comparing cross-validation curves between methods is not straightforward, because each curve is parametrized differently. For example: linear regression has no tuning parameters and we report just one error value; k-

nearest-neighbors give us an error curve with respect to the number of nearest neighbors k; kernel regression gives us an error curve over the bandwidth h; smoothing splines gives us an error curve over the smoothing parameter λ.

- So what does it actually mean to choose kernel regression with $h = 1.5$, over say, k-nearest-neighbors with $k = 10$ or smoothing splines with $\lambda = 0.417$? That is, does the $h = 1.5$ model from kernel regression correspond to a more of less complex estimate than the $k = 10$ model from k-nearest-neighbors, or the $\lambda = 0.417$ model from smoothing splines?

- The notion of *degrees of freedom* gives us a way of formally making this comparison. Roughly speaking, the degrees of freedom of a fitting procedure (like kernel regression with $h = 1.5$, or k-nearest-neighbors with $k = 10$) describes the *effective number of parameters* used by this procedure, and hence provides a quantitive measure of the *complexity* of an estimator.

- Keeping track of degrees of freedom therefore saves us from unsuspectingly comparing a procedure that uses say, 10 effective parameters to another one that uses 100.

Definition of Effective Degrees of Freedom

Even though the concept it represents is quite broad, degrees of freedom has a rigorous definition. Consider the following regression setting, where we condition on the values of the random predictors (that is, we treat the predictor measurements x_i, $i = 1, \dots, n$ as fixed):

Notes: We observe $(x_1, Y_1), \dots, (x_n, Y_n)$, with the x_is fixed instead of random.

The true relationship is that $Y_i = r(x_i) + \epsilon_i$, where ϵ_i is random, uncorrelated, and with common variance.

We get the fitted values, $\hat{Y}_i = \hat{r}(x_i)$.

We define the *effective degrees of freedom* of the estimator $\hat{r}$ by:

Notes:

$$\text{df}(\hat{r}) = \frac{1}{\sigma^2} \sum_{j=1}^{n} \text{Cov}(Y_j, \hat{Y}_j).$$

Recall that

$$\text{Cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right].$$

To reiterate: this covariance treats only Y_i, $i = 1, \dots n$ as random (and not x_i, $i = 1, \dots n$).

Intuition: The definition of degrees of freedom above looks at the amount of covariance between each point Y_j and its corresponding fitted values $\hat{Y}_j$. We sum these contributions over $j = 1, \dots n$, and divide the result by σ^2 (dividing by σ^2 gets rid of the dependence of the sum on the marginal error variance).

In some situations, it is helpful to write the definition of degrees of freedom *in matrix notation*:

Notes:

$$\mathrm{df}(\hat{r}) = \frac{1}{\sigma^2}\,\mathrm{trace}\left(\mathrm{Cov}(\vec{Y}, \hat{\vec{Y}})\right).$$

Recall that the covariance is an $n \times n$ matrix, and the trace sums up the entries on the diagonal, so this does the same thing as the previous definition.

Examples

To get a sense for degrees of freedom, it helps to work through several basic examples.

Example 1: Simple Average Estimator
Consider $\hat{\vec{Y}}_{\mathrm{ave}} = (\bar{Y}, \dots \bar{Y})^T$, where $\bar{Y} = \frac{1}{n}\sum_{i=1}^n Y_i$. Then, the effective number of parameters is given by

Notes: We have

$$\hat{r}(x_i) = \frac{1}{n}\sum_{j=1}^n Y_j,$$

so

$$\begin{aligned}\mathrm{df}(\hat{r}) &= \frac{1}{\sigma^2}\sum_{i=1}^n \mathrm{Cov}\left(\frac{1}{n}\sum_{j=1}^n Y_j, Y_i\right)\\ &= \frac{1}{\sigma^2}\sum_{i=1}^n \frac{1}{n}\mathrm{Var}(Y_i)\\ &= 1,\end{aligned}$$

since the $\mathrm{Cov}(Y_j, Y_i) = 0$ for $i \neq j$.

Think: does this result make sense to you?

Example 2: Identity Estimator

Consider $\hat{\vec{Y}}^{\text{id}} = (Y_1, \dots Y_n)^T$. Then, the effective number of parameters is given by

Notes:

$$\hat{r}(x_i) = Y_i,$$

so

$$\text{df}(\hat{r}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \text{Cov}(Y_i, Y_i) = n.$$

Think: does this result make sense to you?

Example 3: Linear Smoothers

Recall that a *linear smoother* has the form

$$\hat{Y}_i^{\text{linsm}} = \hat{r}^{\text{linsm}}(x_i) = \sum_{j=1}^{n} w(x_i, x_j) \cdot Y_j,$$

i.e., we can write

$$\hat{\vec{Y}}^{\text{linsm}} = L\vec{Y},$$

for the matrix $L \in \mathbb{R}^{n \times n}$ defined as $L_{ij} = w(x_i, x_j)$. Calculating degrees of freedom, in matrix form, we get

Notes: We have $\hat{r}^{\text{linsm}}$ as shown above. Plug it into the formula:

$$\begin{aligned}
\text{df}(\hat{r}) &= \frac{1}{\sigma^2}\sum_{i=1}^{n}\text{Cov}(\hat{Y}_i^{\text{linsm}}, Y_i) \\
&= \frac{1}{\sigma^2}\sum_{i=1}^{n}\text{Cov}\left(\sum_{j=1}^{n} w(x_i, x_j)Y_j, Y_i\right) \\
&= \frac{1}{\sigma^2}\sum_{i=1}^{n}\sigma^2 w(x_i, x_i) \\
&= \text{trace}(L).
\end{aligned}$$

This is a very useful formula! Recall that linear regression, k-nearest-neighbors, kernel regression and splines are all linear smoothers. We can calculate the degrees of freedom for all of these estimators by simply summing these weights. As concrete examples, we consider linear regression and k-nearest-neighbors regression.

(3a) Linear Regression

Consider the fitted values from linear regression of Y on X,

$$\hat{\vec{Y}}^{\text{linreg}} = \mathbb{X}\hat{\beta} = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\vec{Y} = H\vec{Y}.$$

Here $\mathbb{X}$ is the $n \times q$ predictor or "design" matrix (with observation x_i along its ith row). Then, relying on the above result, we get

Notes: We call H the "hat" matrix, but it is just an example of L with (i, j) entry being the j coordinate of $x_i^\top(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}$. So the effective degrees of freedom calculation is:

$$\begin{aligned}\mathrm{df}(\hat{r}) &= \mathrm{trace}(H)\\ &= \mathrm{trace}(\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T)\\ &= \mathrm{trace}(\mathbb{X}^T\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1})\\ &= \mathrm{trace}(I_q)\\ &= q,\end{aligned}$$

where q is the number of covariates plus the intercept, i.e. the number of parameters $\beta_0, \beta_1, \ldots, \beta_p$.

Think: does this result make sense to you?

(3b) k-Nearest-Neighbors Regression

Consider k-nearest-neighbors regression with some fixed value of $k \geq 1$. Recall that here

$$w(x, x_j) = \begin{cases} \frac{1}{k} & \text{if } x_j \text{ is one of the } k \text{ closest points to } x \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the effective number of parameters is given by

Notes:

$$\begin{aligned}
\mathrm{df}(\hat{r}) &= \frac{1}{\sigma^2}\sum_{j=1}^{n}\mathrm{Cov}(Y_j, \hat{Y}_j) \\
&= \frac{1}{\sigma^2}\sum_{j=1}^{n}\mathrm{Cov}\left(Y_j, \sum_{i=1}^{n} w(x_i, x_j) Y_i\right) \\
&= \frac{1}{\sigma^2}\sum_{j=1}^{n}\frac{\sigma^2}{k} \\
&= \frac{n}{k}.
\end{aligned}$$

Think: what happens for small k? Large k? Does this match your intuition for the complexity of the k-nearest-neighbors fit?

Estimating the Effective Degrees of Freedom

How do we estimate the degrees of freedom in more complex settings?

Degrees of freedom cannot always be calculated analytically, as we did above. In general, it is rather uncommon for this to be the case. As an extreme example, if the fitting procedure $\hat{r}$ is just a black box (e.g., just an R function whose mechanism is unknown), then we would have no way of analytically counting its degrees of freedom. However, the expression of degrees of freedom is still well-defined for any fitting procedure $\hat{r}$; to get an estimate of its degrees of freedom, we can try to estimate the covariance terms $\mathrm{Cov}(\hat{Y}_i, Y_i)$ via the bootstrap method.

Question: Which bootstrap sampling scheme would be appropriate for our application and why? (Think about what is random and what is fixed in the definition of the effective degrees of freedom)

Notes: We are fixing the x_i and assuming that the model is close to correct. Then, the residuals $\hat{\epsilon}_i = Y_i - \hat{Y}_i$ (sometimes called e_i) stand in for the irreducible errors ε_i. So, we resample residuals.

That is, after fitting $\hat{Y}_i = \hat{r}(x_i)$, $i = 1, \dots n$ using the original sample (X_i, Y_i), $i = 1, \dots n$, we record the (empirical) residuals

$$\hat{\epsilon}_i = Y_i - \hat{Y}_i, \quad i = 1, \dots n.$$

Then, form a bootstrap sample of size n by resampling the residuals with replacement and recompute the regression fit on the bootstrap sample:

Notes: Instead of resampling pairs with replacement, we *resample residuals with replacement*. That is, for $b = 1, \dots, B$, we repeat:

1. draw bootstrap samples $(x_i, Y^*_{i,b})$, $i = 1, \dots n$ according to

$$Y^*_{i,b} = \hat{r}(x_i) + \hat{\varepsilon}^*_{i,b}, \text{ where } \hat{\varepsilon}^*_{i,b} \overset{\text{i.i.d.}}{\sim} \text{Unif}\{\hat{\varepsilon}_1, \dots, \hat{\varepsilon}_n\}, \quad i = 1, \dots, n,$$

   i.e., instead of resampling pairs with replacement, we're resampling residuals with replacement;

2. Use the bootstrap sample to fit the model and obtain $\hat{Y}^*_{1,b}, \dots, \hat{Y}^*_{n,b}$. Store these.

In the end, we approximate the covariance of $\hat{Y}_i$ and Y_i by the empirical covariance between $\hat{Y}^*_{i,b}$ and $Y^*_{i,b}$ averaged over $b = 1, \dots B$, i.e.,

$$\mathrm{Cov}(\hat{Y}^*_i, Y_i) \approx \frac{1}{B}\sum_{b=1}^{B}\left(\hat{Y}^*_{i,b} - \frac{1}{B}\sum_{r=1}^{B}\hat{Y}^*_{i,r}\right)\cdot\left(Y^*_{i,b} - \frac{1}{B}\sum_{r=1}^{B}Y^*_{i,r}\right),$$

and sum this over $i = 1, \dots, n$ to yield our bootstrap estimate for degrees of freedom:

$$\mathrm{df}(\hat{r}) \approx \frac{1}{\sigma^2}\sum_{i=1}^{n}\left(\frac{1}{B}\sum_{b=1}^{B}\left(\hat{Y}^*_{i,b} - \frac{1}{B}\sum_{r=1}^{B}\hat{Y}^*_{i,r}\right)\cdot\left(Y^*_{i,b} - \frac{1}{B}\sum_{r=1}^{B}Y^*_{i,r}\right)\right).$$

(For simplicity, you can assume that σ^2 is known; otherwise, we would have to estimate it too.)

R Demo 6.2: Estimating Degrees of Freedom via the Residual Bootstrap

a. Create a data example for linear regression. Set $\sigma^2 = 1$. Use simulation to estimate the effective degrees of freedom.

b. Estimate the degrees of freedom using resampling of residuals. Compare with the result in part a and discuss.

c. *(optional)* Estimate the degrees of freedom using resampling of cases. Compare with the results in parts a and b and discuss.

Degrees of Freedom and the Optimism of the In-Sample Error

Big picture: Recall from Lecture 1 (e.g. Demo 1) and Lecture 7 that the training error underestimates the risk or prediction error. It turns out that the effective degrees of freedom are directly related to the so-called *optimism* of the in-sample error. Once we see the precise relationship, it will be clear how we can use this knowledge to construct computationally efficient alternatives to cross-validation procedures for estimating risk. We will also show how one can use effective degrees of freedom to estimate the (extra-sample) prediction error for linear smoothers.

Set-up: Consider a regression model where the predictors x_i, $i = 1, \ldots, n$ are fixed, and the residual errors ε_i are uncorrelated with each other and have common mean 0 and common variance $\sigma^2 > 0$. That is, assume:

Notes: $Y_i = r(x_i) + \varepsilon_i$, where $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0$ for $i \neq j$, $\mathbb{E}(\varepsilon_i \mid X_i) = 0$, and $\mathrm{Var}(\varepsilon_i) = \sigma^2$, for all i.

Let $\hat{r}(x)$ be any estimator, derived from observations (x_i, Y_i), $i = 1, \ldots, n$, and let $\hat{Y}_i = \hat{r}(x_i)$ be the fitted value of the regression at x_i. Then, the *expected training error* of $\hat{r}$ is given by

Notes: $\mathbb{E}[\hat{R}_{\text{train}}] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[(Y_i - \hat{r}(x_i))^2]$

Now suppose we draw a *new observation of Y_i at the observed covariate value x_i*. Denote this new response value by Y_i' (important: the covariate values x_i are here the same; that is, we are treating these as fixed). Then, the *in-sample prediction risk* or *expected (in-sample) test error*, R_{in}, is given by

Notes: $R_{\text{in}} = \frac{1}{n}\mathbb{E}[\{Y_i' - \hat{r}(x_i)\}^2]$, where Y_i' is independent of $Y_1, \dots, Y_n$

Now remember that these two quantities—training and test errors—behave differently, and it is usually the *expected test error* that we seek for the purposes of model validation or model selection. Interestingly, it turns out that (in this simple setup, where x_i, $i = 1, \dots n$ are fixed) we have the relationship

Notes:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\{Y_i' - \hat{r}(x_i)\}^2] &= \frac{1}{n}\sum_{i=1}^{n}\left[\mathbb{E}[\{Y_i - \hat{r}(x_i)\}^2] + 2\text{Cov}(\hat{Y}_i, Y_i)\right] \\ &= \mathbb{E}[R_{\text{training}}] + 2\sigma^2\text{edf}/n,\end{aligned}$$

where edf is the effective degrees-of-freedom for $\hat{r}$.

In words: *The expected test error is exactly the expected training error plus a constant factor ($2\sigma^2/n$) times the effective degrees of freedom.*

Can you prove this result? (Hint: see Lecture 3.)

From this decomposition, we can also immediately see that *the larger the degrees of freedom, i.e., the more complex the fitting method, the larger the gap between the expected testing and training errors* (see Figure 2).

Estimating the (In-Sample) Prediction Error

The relationship discussed in the last section leads to a very natural estimate for the expected test error. Consider the training error of $\hat{r}$ plus $2\sigma^2/n$ times its degrees of freedom,

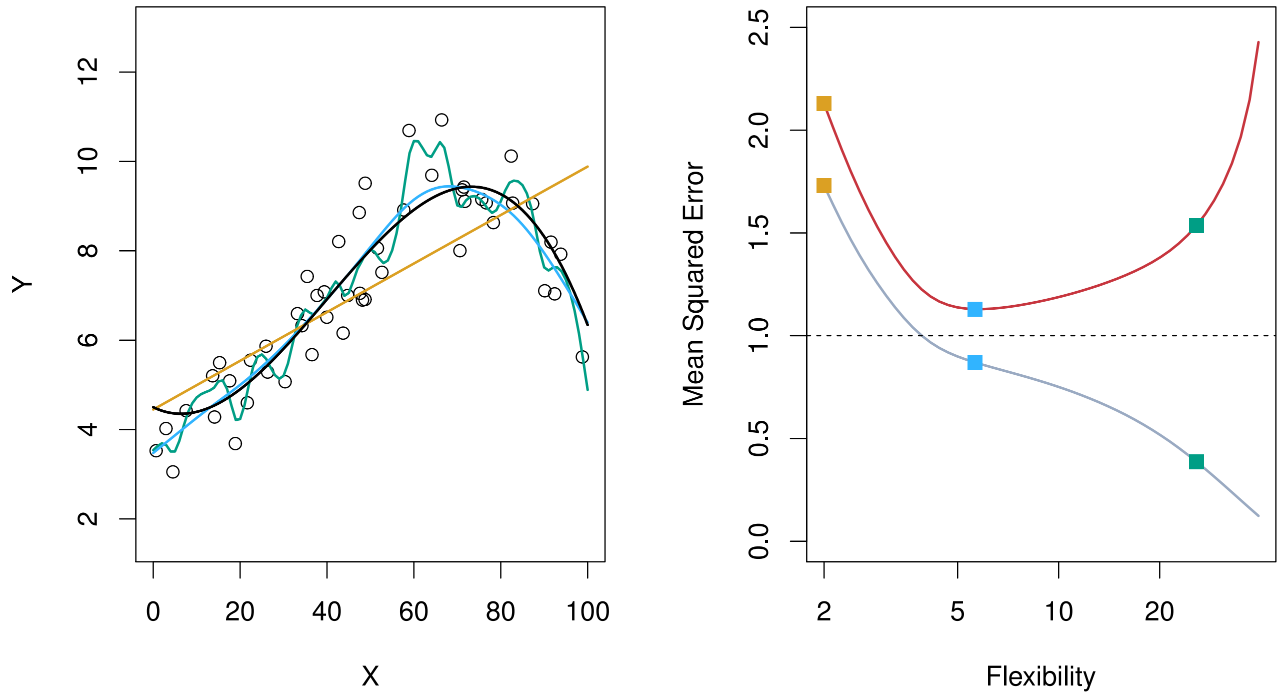

Figure 2: *Left:* Data simulated from a regression model f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves). *Right:* Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel. [Credit to James et al, "An Introduction to Statistical Learning"]

Notes:
$$\hat{R}_{\text{in}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \frac{2\sigma^2}{n}\text{df}(\hat{r}),$$

From the previous section, we know that $\hat{R}_{\text{in}}$ is an *unbiased estimate of the (in-sample) prediction risk*, that is

Notes: $\mathbb{E}[\hat{R}_{\text{in}}] = R_{\text{in}}$.

If we knew an estimator's effective degrees of of freedom, then we could use $\hat{R}_{\text{in}}$ to approximate its prediction error.

Q: What happens when we don't know its degrees of freedom?

Notes: If we have a good estimate $\widehat{\mathrm{df}}(\hat{r})$ for the degrees of freedom of $\hat{r}$, then we can simply form the error estimate

$$T = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \frac{2\sigma^2}{n}\widehat{\mathrm{df}}(\hat{r}),$$

and by the same logic, $\mathbb{E}(T)$ will be close to the expected test error as long as $\mathbb{E}[\widehat{\mathrm{df}}(\hat{r})]$ is close to $\mathrm{df}(\hat{r})$. Such an estimate $\widehat{\mathrm{df}}(\hat{r})$ could have come from, say, the bootstrap, as discussed earlier (but in this case, you should be aware of the fact that these bootstrap calculations themselves may be roughly as expensive as cross-validation). But the estimate $\widehat{\mathrm{df}}(\hat{r})$ can also come from other means, e.g., from an analytic form. We'll see an example of this later in the course (when we discuss ℓ_1 regularization, i.e., the lasso)

Q: What about σ^2?

Notes: In general, if we don't know its true value (which is going to pretty much going to always be the case in practice), then we will have to estimate it too. Notice however that if we used the bootstrap to form the estimate $\widehat{\mathrm{df}}(\hat{r})$, then this already provides us with an estimate of $2\sigma^2\mathrm{df}(\hat{r})$, so we don't have to estimate σ^2 on its own.

Q: How is the definition of expected test error R_{in} different from the prediction risk $R = \mathbb{E}[(Y - \hat{r}(X)]^2$ (see Lecture 3), which we also estimated in Lecture 1, R Demo 1, and in HW 1?

Notes: In those cases, we were replacing all pairs (X_i, Y_i) by (X_i', Y_i') when computing the out-of-sample risk. We were estimating

$$R = \mathbb{E}[(Y - \hat{r}(X))^2],$$

where $\hat{r}(X)$ was fit to a training sample, and (X, Y) is a new *pair* drawn from the population distribution.

Put another way, we were not conditioning on the x_i all being known in advance as we are doing with in-sample test risk.

Leave-One-Out Cross-Validation and Generalized Cross-Validation

In Lecture 3, we described how to estimate the prediction risk $R = \mathbb{E}(Y - \widehat{r}(X))^2$ of a model with a method called K-fold cross-validation. Often people take $K = 5$ or $K = 10$ (a larger value for K leads to a less biased estimate of the risk but at the expense of a higher variance).

The choice $K = n$ corresponds to leave-one-out cross-validation (LOOCV), but for LOOCV there's actually no need to actually drop each observation and re-fit the model n times, as we will show.

Recall that the leave-one-out cross-validation score is defined by

$$\hat{R}_{\text{LOOCV}} = \frac{1}{n} \sum_{i=1}^{n} [Y_i - \hat{r}_{(-i)}(X_i)]^2 \tag{1}$$

where $\hat{r}_{(-i)}$ is the estimator obtained by omitting the i^{th} pair (X_i, Y_i). Recall that the leave-one-out cross-validation score is a nearly unbiased estimate of the risk; that is, $\mathbb{E}(\hat{R}_{\text{LOOCV}}) \approx$ predictive risk. What makes this score especially useful is that there is a shortcut formula for computing $\hat{R}_{\text{LOOCV}}$. For most *linear smoothers* $\hat{r}$, the leave-one-out cross-validation score can be written as

$$\hat{R}_{\text{LOOCV}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{r}(X_i)}{1 - L_{ii}}\right)^2 \tag{2}$$

where L_{ii} is the i^{th} diagonal element of the smoothing matrix L. (See Shalizi section 3.4.3 for derivation.)

Rather than minimizing the cross-validation score, an alternative is to use *generalized cross-validation* in which each L_{ii} in equation (2) is replaced with its average $n^{-1}\sum_{i=1}^{n} L_{ii} = \nu(\lambda)/n$ where $\nu(\lambda) = \text{trace}(L)$ is the effective degrees of freedom and λ is the smoothing or tuning parameter used in our linear smoother (such as bandwidth or the λ in smoothing splines). This saves some calculation.

Thus, we would minimize

Notes:

$$\begin{aligned}
\hat{R}_{\text{GCV}}(\lambda) &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{r}(x_i)}{1 - \nu(\lambda)/n}\right)^2 \\
&= \frac{1}{(1-\nu(\lambda)/n)^2}\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}(x_i))^2 \\
&= \frac{\hat{R}_{\text{train}}}{(1-\nu(\lambda)/n)^2}.
\end{aligned}$$

It looks like we are just minimizing a constant times the training error. But both ν and the training error change as the model (or tuning parameter) change. Typically, the training error goes up (down) when ν goes up (down). So both numerator and denominator of the formula for GCV typically move in the same direction. This means that the GCV criterion can go either way as ν varies, even though the training error is monotone in ν.

Usually, the smoothing parameter that minimizes the generalized cross-validation score is close to the smoothing parameter that minimizes the cross-validation score. (The R function `smooth.spline` has both generalized and leave-one-out CV as options for choosing tuning parameters, and uses GCV by default.)

Model Scoring and Model Selection

The above ideas naturally apply to the model selection problem. Suppose our predictor $\hat{r} = \hat{r}_\lambda$ depends on a tuning parameter λ; we also write $\widehat{\vec{Y}}_\lambda$ for the vector of fitted values at λ. Then over a grid of values, say $\lambda \in \{\lambda_1, \dots \lambda_m\}$, we can compute the score from k-fold CV, LOOCV or GCV, and choose the λ that minimizes the cross-validation score (similar to Lecture 3).

Alternatively, we can as shown before compute the *in-sample prediction error* (= training error + complexity penalty/bias correction)

$$\hat{R}_{\text{in}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - (\widehat{\vec{Y}}_\lambda)_i\right)^2 + \frac{2\sigma^2}{n}\text{df}(\widehat{\vec{Y}}_\lambda)$$

(replacing the degrees of freedom and σ^2 above with estimates, if needed), and choose λ to minimize $\hat{R}_{\text{in}}(\lambda)$. That is, we select

$$\hat{\lambda} = \underset{\lambda \in \{\lambda_1, \dots \lambda_m\}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - (\widehat{\vec{Y}}_\lambda)_i\right)^2 + \frac{2\sigma^2}{n}\text{df}(\widehat{\vec{Y}}_\lambda) \right\}$$

This expression may look familiar to you if we consider the case of linear regression on any number of predictor variables between 1 and p. In the linear regression setting, λ indexes the number of predictors used in the regression; to make things look more familiar, we will rewrite this parameter as k. Hence $k \in \{1, \dots p\}$, and the above model selection criterion becomes

$$\hat{k} = \underset{k \in \{1, \dots p\}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - (\widehat{\vec{Y}}_k)_i\right)^2 + \frac{2\sigma^2}{n}k \right\}.$$

You may recall this as **Mallow's C_p criterion** for model selection in linear regression (related to AIC, and then there is BIC, and other model scoring criteria...).

Remark: In-sample prediction error (risk) is not usually of direct interest since future values of the features are not likely to coincide with their training set values. But for comparison between models, in-sample prediction error is convenient and often leads to effective model selection. The reason is that the relative (rather than absolute) size of the error is what matters.

Question: Can you see that the GCV is closely connected to Mallow's C_p statistic? Hint: Use the approximation $(1-x)^{-2} \approx 1 + 2x$.

Notes: Replace $(1 - \nu/n)^{-2}$ in the formula for $\hat{R}_{\text{GCV}}$ by the approximation $1 + 2\nu/n$. This means

$$\hat{R}_{\text{GCV}} \approx \frac{1}{n} \sum_{i=1}^{n} [Y_i - \hat{r}(X_i)]^2 \left[1 + 2\frac{\nu}{n}\right].$$

Now $\sum_{i=1}^{n} [Y_i - \hat{r}(X_i)]^2 / n$ is an estimator of σ^2, so the above is approximately

$$\frac{1}{n} \sum_{i=1}^{n} [Y_i - \hat{r}(X_i)]^2 + \frac{2\nu\sigma^2}{n},$$

which looks a lot like C_p if $k = \nu$.