# ECON 950 — Winter 2018
# Prof. James MacKinnon

## 6. Bias, Variance, and Model Complexity

Suppose we are trying to predict $y$ using inputs $\boldsymbol{x}$ and some model estimated using a training set $\mathcal{T}$. We could be using any sort of procedure, such as $k$NN, lasso, elastic net, kernel regression, or random forests.

We need to use training data (or training and validation data) for two things:

1. Choose the best model (**model selection**).
2. Estimate the prediction error of applying the estimated model to new data (**model assessment**).

Ideally, we would have three separate sets of data: a training set, a validation set, and a test set.

The training set is used to fit the models, the validation set is used to estimate prediction error and decide which model to select, and the test set is used to assess the prediction error of the final model.

In practice, there are often not enough data to use a separate validation sample, so the same dataset is used for estimation and validation.

The test set may only become available after the fact, when we put the model to use making predictions.

That would actually be good. Using the test set more than once to choose among competing models will cause us ultimately to choose a model that fits it too well.

## 6.1. Loss Functions for Regression Models

In order to say anything about how well our procedure will perform in the wild, we need a measure of loss and a way to estimate it.

For regression models, natural measures of loss are **squared error loss**

$$L\big(y, \hat{f}(\boldsymbol{x})\big) = \big(y - \hat{f}(\boldsymbol{x})\big)^2 \tag{1}$$

and **absolute error loss**

$$L\big(y, \hat{f}(\boldsymbol{x})\big) = \big|y - \hat{f}(\boldsymbol{x})\big|. \tag{2}$$

These are actual losses for any $y$ and a particular training set.

We would like to know how well our procedure can be expected to perform conditional on the training set $\mathcal{T}$.

What we want is the **test error** or **prediction error** or **generalization error**,

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}\Big(L\big(y, \hat{f}(\boldsymbol{x})\big) \,|\, \mathcal{T}\Big). \tag{3}$$

In contrast, the **expected test error** or **expected prediction error** or **expected generalization error** is

$$\mathrm{Err} = \mathrm{E}\Big(L\big(y, \hat{f}(\boldsymbol{x})\big)\Big) = \mathrm{E}\big(\mathrm{Err}_{\mathcal{T}}\big), \tag{4}$$

and it is not conditional on $\mathcal{T}$.

In practice, we would like to know (3), because we have a particular training set that we have used to obtain $\hat{f}(\boldsymbol{x})$. The performance of our procedure may well depend on $\mathcal{T}$.

However, it is often infeasible to estimate (3) but feasible to estimate (4), so we use an estimate of $\mathrm{Err}_{\mathcal{T}}$ as our estimate of Err.

The most naive estimate of Err is the **training error**

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, \hat{f}(\boldsymbol{x}_i)\big). \tag{5}$$

But the training error can severely underestimate the test error, especially when the model is complex.

For a given level of complexity, the model always tends to fit the training data better than the test data. See ESL-fig7.01.pdf.

As the model becomes more complex, it tends to fit the training data better and better, even when complexity is penalized.

## 6.2. Loss Functions for Classification Models

For categorical data, we need something other than absolute or squared error loss.

If the response is $G$ and it can take on $K$ values, then an obvious loss function is the **deviance**, which is $-2$ times the contribution to the loglikelihood:

$$L\big(G, \hat{p}(\boldsymbol{x})\big) = -2\sum_{k=1}^{K} \mathbb{I}(G = k) \log \hat{p}_k(\boldsymbol{x}). \tag{6}$$

The test error and expected test error can be written as (3) or (4), with $L\big(G, \hat{p}(\boldsymbol{x})\big)$ replacing $L\big(y, \hat{f}(\boldsymbol{x})\big)$.

In this case, the training error is simply

$$\overline{\text{err}} = -\frac{2}{N}\sum_{i=1}^{N} \log \hat{p}_{g_i}(\boldsymbol{x}_i), \tag{7}$$

where $\hat{p}_{g_i}(\boldsymbol{x}_i)$ is the fitted probability of the observed outcome, say $g$, for the $i^{\text{th}}$ observation.

Observe that, if $\hat{p}_{g_i}(\boldsymbol{x}_i) = 1$, the loss is 0. As $\hat{p}_{g_i}(\boldsymbol{x}_i)$ becomes smaller, its logarithm becomes more negative, and the loss becomes larger.

Loglikelihoods can be used to measure loss in many other cases as well, such as Poisson regression (for count data) or hazard functions.

People sometimes use $\alpha$ to denote a **tuning parameter** and then denote the fitted value as $\hat{f}_\alpha(\boldsymbol{x})$ to reflect the fact that the predictions depend on $\alpha$.

If we do that, we can write

$$\overline{\text{err}}_\alpha = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, \hat{f}_\alpha(\boldsymbol{x}_i)\big). \tag{8}$$

## 6.3. The Bias-Variance Decomposition

In the regression case, where $y = f(\boldsymbol{x}) + \varepsilon$ and $\varepsilon \sim \text{IID}(0, \sigma_\varepsilon^2)$, the expected prediction error is

$$\begin{aligned}
\text{Err}(\boldsymbol{x}_0) &= \text{E}\Big(\big(y - \hat{f}(\boldsymbol{x})\big)^2 \,|\, \boldsymbol{x} = \boldsymbol{x}_0\Big) \\
&= \sigma_\varepsilon^2 + \big(\text{E}\hat{f}(\boldsymbol{x}_0) - f(\boldsymbol{x}_0)\big)^2 + \text{E}\big(\hat{f}(\boldsymbol{x}_0) - \text{E}\hat{f}(\boldsymbol{x}_0)\big)^2 \\
&= \sigma_\varepsilon^2 + \text{Bias}^2\big(\hat{f}(\boldsymbol{x}_0)\big) + \text{Var}\big(\hat{f}(\boldsymbol{x}_0)\big).
\end{aligned} \tag{9}$$

In the case of $k$NN estimation, the squared bias and the variance both take particularly simple forms. The squared bias is

$$\text{Bias}^2 = \left(f(\boldsymbol{x}_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(\boldsymbol{x}_{(\ell)})\right)^2, \tag{10}$$

where $\boldsymbol{x}_{(\ell)}$ is the training observation ranked $\ell^{\text{th}}$ from $\boldsymbol{x}_0$, and the variance is

$$\text{Var} = \frac{1}{k}\sigma_\varepsilon^2. \tag{11}$$

The bias arises because $f(\boldsymbol{x}_0) \neq f(\boldsymbol{x}_{(\ell)})$ for the neighbours that we average over. There would be no bias if $f(\cdot)$ were flat in a large enough neighbourhood around $\boldsymbol{x}_0$.

The variance is just the variance of an average of $k$ quantities with variance $\sigma_\varepsilon^2$.

For a linear regression model, we know that the forecast at $\boldsymbol{x}_0$ is

$$\hat{f}(\boldsymbol{x}_0) = \boldsymbol{x}_0\hat{\boldsymbol{\beta}} = \boldsymbol{x}_0(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}, \tag{12}$$

so that the variance of the forecast error is

$$\text{Var} = \sigma_\varepsilon^2\boldsymbol{x}_0(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{x}_0 = \sigma_\varepsilon^2\boldsymbol{x}_0(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{x}_0. \tag{13}$$

This can also be written as

$$\text{Var} = ||\boldsymbol{h}(\boldsymbol{x}_0)||^2\sigma_\varepsilon^2, \text{ where } \boldsymbol{h}(\boldsymbol{x}_0) \equiv \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{x}_0. \tag{14}$$

Since the average of $||\boldsymbol{h}(\boldsymbol{x}_0)||^2$ over all the sample values is just the trace of the matrix $\boldsymbol{P}_{\boldsymbol{X}}$, we find that

$$\frac{1}{N}\sum_{i=1}^{N}\mathrm{Err}(\boldsymbol{x}_i) = \sigma_\varepsilon^2 + \big(f(\boldsymbol{x}_i) - \mathrm{E}\hat{f}(\boldsymbol{x}_i)\big)^2 + \frac{p}{N}\sigma_\varepsilon^2. \tag{15}$$

For ridge regression, the forecast variance has the same form as (14), except that

$$\boldsymbol{h}(\boldsymbol{x}_0) = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \alpha\mathbf{I})^{-1}\boldsymbol{x}_0. \tag{16}$$

For all linear regression estimators, such as ridge, the average squared bias is

$$\begin{aligned}&\mathrm{E}_{\boldsymbol{x}_0}\big(f(\boldsymbol{x}_0) - \mathrm{E}\hat{f}_\alpha(\boldsymbol{x}_0)\big)^2 \\ &\quad= \mathrm{E}_{\boldsymbol{x}_0}\big(f(\boldsymbol{x}_0) - \boldsymbol{x}_0^\top\boldsymbol{\beta}_*\big)^2 + \mathrm{E}_{\boldsymbol{x}_0}\big(\boldsymbol{x}_0^\top\boldsymbol{\beta}_* - \mathrm{E}(\boldsymbol{x}_0^\top\hat{\boldsymbol{\beta}}_\alpha)\big)^2,\end{aligned} \tag{17}$$

where $\boldsymbol{\beta}_*$ denotes the value of $\boldsymbol{\beta}$ that minimizes

$$\mathrm{E}\big(f(\boldsymbol{x}) - \boldsymbol{x}^\top\boldsymbol{\beta}\big)^2, \tag{18}$$

where the expectation is taken over the distribution of $\boldsymbol{x}$.

From (17), we see that the average squared bias is the sum of two things.

The first thing is the average **squared model bias**, which arises because $f(\boldsymbol{x}_0) \neq \boldsymbol{x}_0^\top \boldsymbol{\beta}$ for any $\boldsymbol{\beta}$.

The second thing is the average **squared estimation bias**, which arises because $\mathrm{E}(\boldsymbol{x}_0^\top \hat{\boldsymbol{\beta}}_\alpha) \neq \boldsymbol{x}_0^\top \boldsymbol{\beta}_*$.

For linear models fit by OLS, the estimation bias is zero. All bias is model bias, which reflects misspecification.

But for regularized linear models, such as ridge regression and lasso, it is not zero. More aggressive regularization increases estimation bias, but the consequent reduced variance may make it worthwhile.

ESL-fig7.03.pdf illustrates the bias-variance tradeoff in four cases.

The top two panels are for regression with squared error loss, for both $k$NN and best subset regression.

The bottom two panels are for classification with 0-1 loss.

Variance is blue, squared bias is green, and expected prediction error is orange.

In the top panels, which deal with regression, prediction error is the sum of squared bias and variance.

In the bottom panels, which deal with classification, prediction error is no longer the sum of squared bias and variance.

With 0-1 loss in a classification problem, mistakes that do not affect how we classify an observation do not affect loss.

Even if the true probability is 0.98, and the model estimate is 0.51, the model makes the right choice.

This is one reason why I prefer a loss function based on deviance, that is, the loglikelihood function. See (6).

## 6.4. The Training Error Rate

The test (or prediction, or generalization) error of a model $\hat{f}$ is

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}_{y_0, \boldsymbol{x}_0}\Big(L\big(y_0, \hat{f}(\boldsymbol{x}_0)\big) \,|\, \mathcal{T}\Big). \tag{19}$$

The point $(y_0, \boldsymbol{x}_0)$ is a new test data point drawn from the joint distribution of the data. We take expectations conditional on the training set $\mathcal{T}$.

Because (19) is conditional on $\mathcal{T}$, if we happened to obtain a different training set, $\hat{f}$ would be different, and so would $\mathrm{Err}_{\mathcal{T}}$.

Using the training error (5) to estimate (19) may not work well, because the same data are used to fit the model and assess its performance.

There are fundamentally two problems.

- The point $\boldsymbol{x}_0$ typically does not coincide with any of the $\boldsymbol{x}_i$. If it did coincide, then $k$NN with $k = 1$ would be unbiased.
- When $\boldsymbol{x}_0$ is a long way from the nearest $\boldsymbol{x}_i$, perhaps outside the convex hull of the sample points, all methods have to extrapolate.

- Flexible methods, such as higher-order polynomials, often extrapolate very badly. That is why natural cubic splines are constrained to be linear at the boundary.
- Even if $\boldsymbol{x}_0$ does coincide with one of the $\boldsymbol{x}_i$, $y_0$ is random, and so $y_0 \neq y_i$.

To separate these two issues, we can define the **in-sample error** as

$$\mathrm{Err}_{\mathrm{in}} = \frac{1}{N}\mathrm{E}_{y_0}\Big(L\big(y_{i0}, \hat{f}(\boldsymbol{x}_i)\big) \,|\, \mathcal{T}\Big); \tag{20}$$

compare (19). Here we (conceptually) observe $N$ new values $y_{i0}$ at the same $N$ training points.

The **optimism** is

$$\mathrm{op} \equiv \mathrm{Err}_{\mathrm{in}} - \overline{\mathrm{err}}, \tag{21}$$

and the **average optimism** is

$$\omega \equiv \mathrm{E}_{\boldsymbol{y}}(\mathrm{op}). \tag{22}$$

The expectation in (22) is taken over the values of $y_0$ holding the $\boldsymbol{x}_i$ constant.

In (21) and (22), we are holding the values of $\boldsymbol{x}_i$ in the training set fixed. Of course, we have to do that for $\overline{\text{err}}$, because it is just something we compute; see (5) and (7).

For many loss functions, it can be shown that

$$\omega = \frac{2}{N}\sum_{i=1}^{N}\text{Cov}(\hat{y}_i, y_i). \tag{23}$$

Thus the average optimism will be high when $y_i$ greatly affects its own prediction and low when $y_i$ has little effect on its own prediction.

It is interesting to see why (23) is true for linear regression.

The fitted values are $\hat{\boldsymbol{y}} = \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y}$. Thus the sample covariance of $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ for a correctly specified model is

$$\boldsymbol{u}^\top\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y} = \boldsymbol{u}^\top\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u} + \boldsymbol{u}^\top\boldsymbol{X}\boldsymbol{\beta}. \tag{24}$$

The second term here has expectation zero, and the expectation of the first term is $\sigma^2\text{Tr}(\boldsymbol{P}_{\boldsymbol{X}}) = p\sigma^2$. Thus

$$\frac{1}{N}\sum_{i=1}^{N}\text{Cov}(\hat{y}_i, y_i) = \frac{p}{N}\sigma^2. \tag{25}$$

At the same time, $\overline{\text{err}}$ is just $1/N$ times the SSR. Of course, the SSR is equal to $\boldsymbol{u}^\top \boldsymbol{M}_{\boldsymbol{X}} \boldsymbol{u}$ and has expectation $(N-p)\sigma^2$. It is not quite as easy to see that

$$\begin{aligned}
\text{Err}_{\text{in}} &= \frac{1}{N}\text{E}_{\boldsymbol{y}}\big((\boldsymbol{y}_0 - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y})^\top(\boldsymbol{y}_0 - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y})\big) \\
&= \frac{1}{N}\text{E}_{\boldsymbol{y}}\big((\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{u}_0 - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u})^\top(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{u}_0 - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u})\big) \\
&= \frac{1}{N}\text{E}_{\boldsymbol{y}}\big((\boldsymbol{u}_0 - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u})^\top(\boldsymbol{u}_0 - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u})\big) \qquad (26) \\
&= \frac{1}{N}\text{E}_{\boldsymbol{y}}(\boldsymbol{u}_0^\top\boldsymbol{u}_0 + \boldsymbol{u}^\top\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u} - 2\boldsymbol{u}_0^\top\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u}) \\
&= \sigma^2 + \frac{p}{N}\sigma^2.
\end{aligned}$$

Thus $\text{Err}_{\text{in}} - \text{E}_{\boldsymbol{y}}\overline{\text{err}}$ is

$$\sigma^2 + \frac{p}{N}\sigma^2 - \frac{N-p}{N}\sigma^2 = \frac{2p}{N}\sigma^2. \qquad (27)$$

This is indeed twice the covariance of $\hat{y}_i$ and $y_i$; see (25).

So the claim that the average optimism is twice the average covariance between $\hat{y}_i$ and $y_i$ is true for OLS regression.

Similar results are true asymptotically for other regression models and for models estimated by maximum likelihood.

Equivalently, we can write

$$\mathrm{E}_{\boldsymbol{y}}\mathrm{Err}_{\mathrm{in}} = \mathrm{E}_{\boldsymbol{y}}\overline{\mathrm{err}} + \frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i) \tag{28}$$

So the expected in-sample error always exceeds the expectation of the training error. What is responsible for the difference is the optimism, which arises because $y_i$ affects its own fitted value.

We have already seen that, for a linear regression model, the second term in (28) is $2(d/N)\sigma^2$; see (25). I used $d$ instead of $p$ here, because $d$ could be a number of basis functions rather than a number of inputs.

Thus (28) becomes

$$\mathrm{E}_{\boldsymbol{y}}\mathrm{Err}_{\mathrm{in}} = \mathrm{E}_{\boldsymbol{y}}\overline{\mathrm{err}} + \frac{2d}{N}\sigma^2. \tag{29}$$

This tells us that the optimism is proportional to $d/N$.

Equation (29) implies that we can also write

$$\frac{2}{N}\sum_{i=1}^{N}\text{Cov}(\hat{y}_i, y_i) = 2\frac{d}{N}\sigma_\varepsilon^2. \tag{30}$$

Of course, these results would not hold if, for every $d$, we searched over a set of models and chose the best one.

## 6.5. Estimating In-Sample Prediction Error

One way to estimate $\text{Err}_{\text{in}}$ is to add an estimate of $\omega$ to $\overline{\text{err}}$. Based on (29), we should add $2(d/N)\hat{\sigma}_\varepsilon^2$ to $\overline{\text{err}}$ in the case of a regression-like model.

Here $\hat{\sigma}_\varepsilon^2$ should be based on a low-bias model, so that it provides a good estimate of $\sigma_\varepsilon^2$. This yields a variant of the $C_p$ statistic:

$$C_p = \overline{\text{err}} + 2\frac{d}{N}\hat{\sigma}_\varepsilon^2. \tag{31}$$

As we have seen, the AIC is similar to $C_p$. It is based on the fact that

$$-2\mathrm{E}\big(\log(\hat{\Pr}(y))\big) = -\frac{2}{N}\mathrm{E}\big(\mathrm{logl}(\hat{\boldsymbol{\theta}})\big) + 2\frac{d}{N}, \tag{32}$$

where $\mathrm{logl}(\hat{\boldsymbol{\theta}})$ denotes the maximized loglikelihood.

Thus we get

$$\mathrm{AIC} = -\frac{2}{N}\mathrm{logl}(\hat{\boldsymbol{\theta}}) + 2\frac{d}{N}. \tag{33}$$

For regression models with normal errors, AIC and $C_p$ are equivalent.

When there is a tuning parameter $\alpha$ and we can write the effective degrees of freedom as $d(\alpha)$, we can plot

$$\mathrm{AIC}(\alpha) = \overline{\mathrm{err}}(\alpha) + 2\frac{d(\alpha)}{N}\hat{\sigma}_\varepsilon^2 \tag{34}$$

as an estimate of the **test error curve**. We find $\hat{\alpha}$ that minimizes (34).

This procedure is legitimate in some cases, but not in others. In particular, if the basis functions are chosen adaptively, the optimism will exceed $2(d(\alpha)/N)\sigma_\varepsilon^2$.

The problem is that, by choosing the best set of basis functions for any $\alpha$, we effectively use up more than $d$ degrees of freedom.

ESL-fig7.04.pdf shows an example of using AIC for classification (for the case of phoneme recognition).

The tuning parameter is $M$, the number of basis functions for a natural cubic spline.

The knots are chosen uniformly over the range of frequencies.

The figure shows the training error, $\overline{\text{err}}$, the AIC, and an independent measure of Err based on a test sample.

For loglikelihood loss, AIC and Err track each other closely until $M = 256$, which is very large.

For 0-1 loss, where (30) is not true, AIC and Err differ quite a bit but yield similar conclusions and do not diverge sharply at $M = 256$.

## 6.6. The Effective Number of Parameters

Any linear fitting method can be written as

$$\hat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{y}. \tag{35}$$

Such methods include OLS, where $\boldsymbol{S} = \boldsymbol{P}_{\boldsymbol{X}}$, linear regression on basis functions, where $\boldsymbol{S} = \boldsymbol{P}_{\boldsymbol{B}}$ and $\boldsymbol{B}$ is a matrix of basis functions, and methods that use quadratic smoothing such as ridge regression and smoothing splines.

In all such cases, the **effective number of parameters**, or **effective degrees of freedom**, is simply the trace of $\boldsymbol{S}$:

$$\mathrm{df}(\boldsymbol{S}) = \mathrm{Tr}(\boldsymbol{S}). \tag{36}$$

We proved in (25) that, for linear regression models,

$$\mathrm{Tr}(\boldsymbol{S})\sigma_\varepsilon^2 = \sum_{i=1}^{N} \mathrm{Cov}(\hat{y}_i, y_i). \tag{37}$$

This suggests that, whether or not a fitting method is linear, it is reasonable to define df as

$$\mathrm{df}(\hat{\boldsymbol{y}}) = \frac{1}{\sigma_\varepsilon^2} \sum_{i=1}^{N} \mathrm{Cov}(\hat{y}_i, y_i). \tag{38}$$

So if there were no covariance between $\hat{y}_i$ and $y_i$, the fitting method would not use any degrees of freedom.

In Section 7.6 there is also a result for df for neural networks, but since we have not encountered them yet, it seems premature to write it down.

I will skip BIC (discussed briefly in slides 1), minimum description length, and the Vapnik-Chervonenkis dimension.

The VC dimension is a way of measuring the complexity of a class of functions. It leads to methods called **structural risk minimization**.

## 6.7. Cross-Validation

This method directly estimates the expected prediction error

$$\text{Err} = \text{E}\Big(L\big(y, \hat{f}(\boldsymbol{x})\big)\Big) \tag{39}$$

using the training data. It often estimates Err quite well.

However, despite using the training data, it generally does not estimate $\text{Err}_{\mathcal{T}}$ well. This may be surprising.

The idea of cross validation is not to use all the training data at once when estimating the fit of the model. This greatly reduces $\text{Cov}(\hat{y}_i, y_i)$.

## 6.8. $K$-fold Cross-Validation

We split the data into $K$ (roughly) equal-sized parts, or **folds**.

If the data are ordered at random, it does not really matter how we divide it into folds.

However, if the data are not ordered randomly, we may have to be careful. It will often be good to put observations 1, $K+1$, $2K+1$ and so on into fold 1, observations 2, $K+2$, $2K+2$ and so on into fold 2, and similarly for the other folds.

For each fold indexed by $k = 1, \ldots, K$, we fit the model to the other $K-1$ folds and calculate the prediction error for fold $k$.

Let $\kappa(i)$ be an indexing function which indicates which fold of the data observation $i$ belongs to.

For example, if $\kappa(1323) = 4$, observation 1323 belongs to fold 4.

Let $\hat{f}^{-k}(x)$ denote the fit at the point $x$ using data from $K-1$ folds, with fold $k$ omitted.

Then the cross-validation estimate of prediction error is

$$\mathrm{CV}(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L\big(y_i, \hat{f}^{-\kappa(i)}(x_i)\big), \tag{40}$$

where $\hat{f}^{-\kappa(i)}(x_i)$ is the fitted value for observation $i$ using all of the data except the fold to which observation $i$ belongs.

In (40), we sum the loss over all observations. The predictions are based on $K$ different models, which differ across observations.

Common choices of $K$ are 5, 10, and $N$. The last of these is used mainly in special cases, such as linear models, where it is cheap to compute. In many other cases, it is very expensive to compute.

If $N$ happens to be divisible by a number like 6, 7, 8, or 9, it is natural to set $K$ equal to that number.

If there is a tuning parameter $\alpha$, we can think of CV as being a function of it. Then we can graph $\text{CV}(\hat{f}, \alpha)$ against $\alpha$ to estimate the test error curve.

As with (34), we then find $\alpha$ to minimize $\text{CV}(\hat{f}, \alpha)$.

When $K = N$, we have leave-one-out cross-validation. In this case, there is very little bias, because $N - 1 \approx N$, but variance tends to be large, because the samples we use are almost the same.

With, say, $K = 5$, variance is lower, but bias can be higher if bias depends strongly on the sample size.

If the prediction error based on samples of size $4N/5$ or $9N/10$ differs substantially from those based on samples of size $N$, bias will be a problem.

ESL defines the **learning curve** as a function that relates Err (or 1−Err in their example) to the sample size $N$. When it is flat near the actual $N$, then bias should be low. When it is steep, bias may be high.

Since Err diminishes with $N$, the $K$-fold CV estimate of Err with small $K$ will be biased upwards.

## 6.9. Generalized Cross-Validation

**Generalized cross-validation** is a convenient alternative to leave-one-out cross-validation for linear models with squared-error loss.

For many linear models,

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{f}^{-i}(\boldsymbol{x}_i)\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - S_{ii}}\right)^2, \tag{41}$$

where $S_{ii}$ is the $i^{\text{th}}$ diagonal element of $\boldsymbol{S}$.

The GCV approximation to (41) is

$$\text{GCV}(\hat{\boldsymbol{f}}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}(\boldsymbol{x}_i)}{1 - \text{Tr}(\boldsymbol{S})/N}\right)^2. \tag{42}$$

Recall that the effective number of parameters is $\text{df} = \text{Tr}(\boldsymbol{S})$. In the case of a linear regression, this is just $p$.

So we shrink every training-set residual by the same factor instead of by a factor that is larger for high-leverage observations.

ESL claims that GCV can alleviate the tendency of CV to undersmooth.

However, shrinking every training-set residual by the same factor seems dubious.

It is interesting to look at these results from the perspective of HCCMEs. For a linear regression model, $S_{ii} = h_i$, the $i^{\text{th}}$ element of the diagonal of the $\boldsymbol{P_X}$ matrix. The $\text{HC}_2$ estimated residual is

$$\ddot{u}_i = \frac{\hat{u}_i}{(1 - h_i)^{1/2}}, \tag{43}$$

and we know that its square has expectation $\sigma^2$ when the errors are homoskedastic.

Similarly, the $\mathrm{HC}_3$ estimated residual is

$$\ddot{u}_i = \frac{\hat{u}_i}{1 - h_i}, \tag{44}$$

and we know that its square has expectation greater than $\sigma^2$ when the errors are homoskedastic. Since (44) is a special case of (41), it seems highly likely that the latter will also overestimate $\sigma^2$ for observation $i$.

Of course, we want it to overestimate $\sigma^2$, because the test error is the sum of $\sigma^2$ and the error that arises from estimation error for $\boldsymbol{\beta}$.

From this perspective, GCV is related to the $\mathrm{HC}_1$ estimated residual, which is

$$\ddot{u}_i = \left(\frac{N}{N - p}\right)^{1/2} \hat{u}_i. \tag{45}$$

The square of this is the expectation of $u_i^2$, on average, which is not what we want.

Instead, the GCV formula (42) effectively uses $N/(N-p)$, since

$$\left(\frac{1}{1-p/N}\right)^2 = \frac{N^2}{(N-p)^2}.$$

Thus, like leave-one-out cross-validation, GCM blows up the error in an effort to take account of estimation error.

## 6.10. The Wrong Way to Perform Cross-Validation

It is extremely important that the way we obtain the CV fits is the same (except for the samples used) as the way we performed the original fit.

For example, suppose we have a classification problem with many predictors. We might follow this approach:

1. Find the correlations between every predictor and $y$, and retain the $d$ predictors with the highest correlations.

2. Build some kind of classifier using these $d$ predictors.

3. Use cross-validation to estimate the tuning parameter(s) and hence the parameters of the final model.

4. Base our estimate of Err on $K$-fold cross-validation using the chosen tuning parameter(s).

This is madness!

We used the entire training set to choose the best $d$ predictors. The fact that we leave out folds of the sample at later stages does not solve the problem.

I have seen people do similar things with kernel regression. They use CV to estimate Err, but they hold the bandwidth fixed at the value chosen using the entire sample.

This sort of thing can save a lot of computer time, but it leads to invalid results.

ESL do an experiment in which $N = 50$, $p = 5000$, there are two equal-sized classes, and all of the predictors actually contain no information.

At the first stage, they choose the 100 predictors most highly correlated with the class labels. They then use 1NN prediction.

The true error rate should be 50%, since the predictors contain no information. But the average CV error rate was just 3%!

The correct way to perform cross-validation in this sort of case is:

1. Divide the sample into $K$ folds at random. Then iterate over all the folds for $k = 1, \ldots, K$.
2. Using data from all folds except fold $k$, find a subset of $d$ "good" predictors.
3. Again using data for all folds except $k$, build a multivariate classifier based on the $d$ chosen predictors.
4. Use this classifier to predict the class labels for the observations in fold $k$

Cross-validation must be applied to the entire sequence of modeling steps!

It is often tempting to reduce computational cost by performing some modeling steps on the entire sample before beginning the cross-validation. But, in most cases, it is wrong!

There is one exception. We can perform *unsupervised* screening steps (i.e., ones that use information about the $\boldsymbol{x}_i$ but do not use any information about the $y_i$).

For example, we could compute principal components just once. For large samples the PCs computed on the whole sample should coincide with the ones computed using the folds, and they will not related to $\boldsymbol{y}$.

ESL states that papers in top journals do this wrong all the time, especially in cases with many predictors, such as genomic studies.

## 6.11. Conditional or Expected Test Error

As noted previously, what we really care about is $\text{Err}_{\mathcal{T}}$, but we may actually be estimating Err, the expected test error.

ESL-fig7.14-15.pdf sheds some light on this. These figures deal with best-subset regression, where we are using cross-validation to determine the subset size, $p$.

The upper left panel of Figure 7.14 shows Err as a thick red curve, along with 100 $\text{Err}_{\mathcal{T}}$ curves for different training samples.

The upper right and lower left panels show CV error curves for 10-fold and $N$-fold (leave-one-out) cross-validation, respectively.

The thick black curves are "expected" cross-validation curves. I think these are just averages over the 100 training samples.

10-fold seems less variable than $N$-fold, especially for small $p$.

Lower-right panel shows three types of absolute approximation error.

The CV curves work best for Err, and 10-fold works better for $\text{Err}_{\mathcal{T}}$ than $N$-fold.

Figure 7.15 shows plots of the true conditional error and CV error for various subset sizes (1, 5, 10) in the first three panels.

The lower right panel shows correlations for both 10-fold and $N$-fold as a function of subset size.

Oddly, the correlations are quite negative for small $p$, somewhat negative for large $p$, and positive for intermediate $p$. Why? Is this plot correct?

## 6.12. Possible future work:

How reliable are various cross-validation methods in the context of more interesting problems than $k$NN and best subset regression?

When does it matter that we treat each observations separately rather than averaging over them as in GCV?

How can we choose a model that predicts well when there is heteroskedasticity of unknown form? When there is clustering of unknown form?

Of course, we should try to answer these questions in the context of models and datasets of interest to economists.