Statistics 8330: Data Analysis III
Model Assessment and Resampling Methods

Suggested Reading: James, Witten, Hastie, and Tibshirani (JWHT), 2013, Chapter 5

Supplemental Reading: Hastie, Tibshirani, and Friedman (HTF), 2009, Chapter 7

Christopher K. Wikle

University of Missouri
Department of Statistics

Model Assessment

In statistical learning, the "*generalization*" performance of a learning method is related to how well it predicts independent test data.

As we mentioned in the first lecture, this is important because it provides an objective way to choose between models (both in terms of *model selection* and *model assessment*).

Recall that we have an output variable, Y, vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training set $\mathcal{T}$.

Generally, we consider errors between Y and $\hat{f}(X)$ through some **loss function**, $L(Y, \hat{f}(X))$. For example, we are most familiar with *squared-error loss*:

$$L(Y, \hat{f}(X)) = (Y - \hat{f}(X))^2,$$

but might consider other loss functions such as *absolute error loss*, $L(Y, \hat{f}(X)) = |Y - \hat{f}(X)|$ or *zero-one loss*, etc.

Types of Error

Recall that we mentioned the notion of *test error* in the first lecture. The test error, sometimes referred to as *generalization error*, is the *prediction error* over an independent test sample and is defined formally as:

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}_{X^o, Y^o}[L(Y^o, \hat{f}(X^o))|\mathcal{T}],$$

where it is assumed that X and Y are drawn randomly from their joint population distribution and it is assumed that the training set $\mathcal{T}$ is fixed. Thus, the test error refers to the error over an independent sample when $f(\cdot)$ is estimated with **this specific training set**.

We can also define the *expected prediction error* (or, *expected test error*) to be

$$\mathrm{Err} = \mathrm{E}_{\mathcal{T}}\mathrm{E}_{X^o, Y^o}[L(Y^o, \hat{f}(X^o))|\mathcal{T}] = \mathrm{E}_{\mathcal{T}}[\mathrm{Err}_{\mathcal{T}}],$$

so this **averages over possible training sets**.

Finally, the *training error* is the average loss over the **specific** training sample, given by: $\overline{\mathrm{err}} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(x_i))$.

Model Error (cont.)

Typically, interest is in the test error, $\text{Err}_{\mathcal{T}}$. However, this is exceedingly difficult to estimate well without a separate test data set. Why?

Within the context of a single data set, we typically can attempt to approximate such a validation step by adjusting the training error (as in AIC model selection) or by efficient re-use (cross-validation/bootstrap) of the training sample (see below). In most cases, we can get good estimates of Err (the expected prediction error) from these procedures, but not $\text{Err}_{\mathcal{T}}$.

Recall from the first lecture that there is a bias-variance tradeoff when we consider squared error loss. In fact, for some learning procedures (e.g., KNN, linear models) one can write out the forms of the bias and variance analytically (e.g., see HTF(2009), Ch. 7.3). This bias-variance tradeoff is related to the performance of the training error as a measure of test error.

Model Error/ Model Selection

We mentioned in the first lecture how using training error was typically *optimistic* in the sense that it is less than the true test error. One can actually quantify this *optimism* (e.g., HTF(2009), Ch. 7.4) and show that **the amount that $\overline{\text{err}}$ underestimates the true test error depends on how strongly y_i affects its own prediction.** That is, the larger $\text{cov}(\hat{y}_i, y_i)$, the more optimistic is the training error. As shown on the next slide, the greater the *effective number of parameters (degrees of freedom)* (more flexible model), the larger is this covariance.

One can show (e.g., HTF(2009), Ch. 7.5) that most of the well-known model selection criteria (e.g., AIC, BIC, C_p) try to estimate prediction error by estimating the optimism and adjusting the training error accordingly to try to obtain an unbiased estimate of prediction error.

Alternatively, cross-validation and bootstrap methods provide direct estimates of the expected prediction (test) error, Err.

Aside: Effective Number of Parameters (df)

Before we discuss resampling methods, consider the effective number of parameters (df) in linear models given in HTF(2009, Ch.7.6). Note, a model with more effective number of parameters is more flexible.

Let $\mathbf{y} = (y_1, \ldots, y_n)'$ (and, similarly define $\hat{\mathbf{y}}$). Then, as we saw in DA I, a linear fitting method can always be written:

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y},$$

where $\mathbf{S}$ is an $n \times n$ matrix depending on the input vectors x_i but not on y_i. E.g., in linear regression, $\mathbf{S} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. We will see that other types of flexible methods (e.g., basis function expansions) can also be written this way.

The nice result is that the *effective number of parameters (df)* is given by:

$$\text{df}(\hat{\mathbf{y}}) = \text{df}(\mathbf{S}) = \text{trace}(\mathbf{S}),$$

which is just the sum of the diagonal elements of $\mathbf{S}$.

Furthermore, if $\hat{\mathbf{y}}$ arises from an additive-error model, $Y = f(X) + \epsilon$, with $\text{var}(\epsilon) = \sigma_\epsilon^2$, then one can show

$$\text{df}(\hat{\mathbf{y}}) = \frac{\sum_{i=1}^n \text{cov}(\hat{y}_i, y_i)}{\sigma_\epsilon^2}.$$

Cross-Validation

In principle, if you had a fairly large data set, you could randomly divide the observations into a training set to fit the model and a *validation* (or holdout, or test) set to evaluate the test error rate relative to your loss function of choice (e.g., MSE). This is easy to do but it has two serious drawbacks:

1. it can be highly variable
2. only a subset of the observations are used to fit the model (most statistical models perform worse when trained on fewer observations); thus, the validation set error rate may overestimate the test error rate

However, an intuitive and simple modification to this strategy can make it a very effective estimate of the average test error rate.

K-Fold Cross-Validation

In **K-fold cross-validation** we randomly split the available data into K roughly equal-size parts (or, “folds”). Each fold is held out and the model is trained on the remaining $K-1$ folds and then the loss function of choice is evaluated on the fold that was held out.

Consider $k = 1, \ldots, K$ folds and let $\hat{f}^{-k}(x)$ denote the fitted function computed with the kth fold removed. One could then compute the loss based on the samples in the kth fold. For example, if we were interested in squared-error loss, we would compute MSE_k for the observations in the kth fold, $k = 1, \ldots, K$. The K-fold cross-validation estimate is then:

$$CV_{(K)} = \frac{1}{K} \sum_{k=1}^{K} MSE_k.$$

It has been shown empirically, that good choices for K are 5 or 10.

Leave-One-Out CV (LOOCV)

A special case of K-fold CV occurs when $K = n$. This is called *leave-one-out cross-validation* (LOOCV). In this case, only a single observation is used for validation and the remaining are used to make up the training set. This is repeated for all n observations. In this case,

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i,$$

where $MSE_i = (y_i - \hat{f}^{-i}(x_i))^2$ (note, any loss function could be used here).

LOOCV has low bias as an estimate of the expected test error rate, but (not surprisingly) it can have high variance. This is why the choice of $K = 5$ or $K = 10$ provides a better compromise between bias and variance. Note, the individual MSE's in LOOCV are quite highly correlated since they are all fit with the same observations except one.

It is also the case that LOOCV can be expensive to implement in general, since it requires one to fit the model n times; but not always.

Generalized Cross-Validation

Assume we have a linear predictor of the form $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$. Recall, the wonderful simplifying relationship for LOOCV under squared error loss that we used in the PRESS statistic in Data I:

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2,$$

where S_{ii} is the ith diagonal element of $\mathbf{S}$ (e.g., the ith element of the "hat matrix" diagonal in linear regression). We only have to fit the model once!!

Generalized cross-validation (GCV) provides a convenient approximation in this setting when the trace of $\mathbf{S}$ can be computed more easily than the individual elements S_{ii}:

$$GCV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(\mathbf{S})/n}\right]^2$$

CV and Classification

Cross-validation works exactly the same way for classification problems as for "regression" problems, with the difference coming in the type of loss function that is used.

One of the most common is 0-1 loss. Recall from Lecture 1 that we can predict the category based on the highest estimated conditional probability given the inputs, say $\hat{G}(X)$. Then, 0-1 loss is defined as the error rate:

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X)).$$

We can also base loss-functions in terms of deviances that factor in the estimated conditional probability directly (e.g., HTF(2009), Ch. 7.2). If $p_k(X) = \Pr(G = k|X)$ and $\hat{G}(X)$ is the k that gives the maximum $\hat{p}_k(X)$,

$$L(G, \hat{p}(X)) = -2 \sum_{k=1}^{K} I(G = k) \log \hat{p}_k(X) = -2 \log \hat{p}_G(X),$$

which is just -2 times the log-likelihood (i.e., the *deviance*).

Classification Example; JWHT(2013, Ch. 5) *CKW*

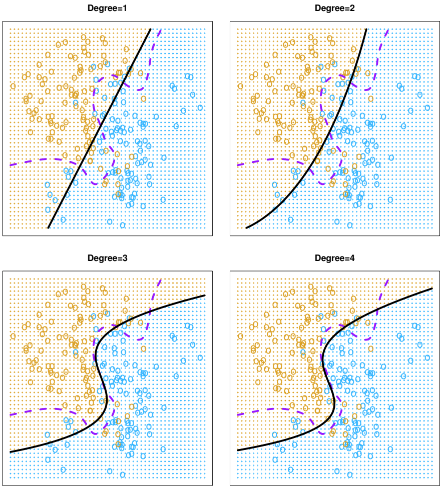

FIGURE 5.7. *Logistic regression fits on the two-dimensional classification data displayed in Figure 2.13. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from linear, quadratic, cubic and quartic (degrees 1–4) logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively* 0.201, 0.197, 0.160, *and* 0.162, *while the Bayes error rate is* 0.133.

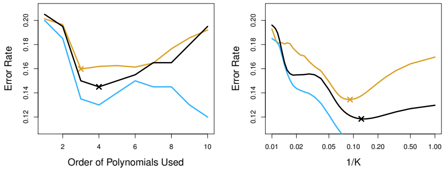

FIGURE 5.8. *Test error (brown), training error (blue), and 10-fold CV error (black) on the two-dimensional classification data displayed in Figure 5.7. Left: Logistic regression using polynomial functions of the predictors. The order of the polynomials used is displayed on the x-axis.* Right: *The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.*

Being Careful with CV Procedures

CKW

HTF(2009, Ch. 7.10.2) make a strong point that one has to be careful when using cross-validation.

In particular, with complex multistep modeling procedures, the **cross-validation must be applied to the entire sequence of modeling steps.** That is, samples must be removed before any selection or filtering steps are applied so long as these steps involve the outputs. An exception would be if there was an unsupervised step in the process associated with the inputs. Since this would not be related to the outputs, it is not necessary to do CV on this component.

HTF give the following example of the “wrong way” and “right way” to do a particular cross-validation problem.

Being Careful with CV Procedures (cont.)

HTF(2009, Ch. 7.10.2) Proteomics example: large number of predictors.

Wrong way!

1. Screen the predictors: find a subset of "good" predictors that show fairly strong (univariate) correlation with the response
2. Using just this subset of predictors, build a multivariate classifier
3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model

Correct way!

1. Divide the samples into K cross-validation folds at random
2. For each fold, k=1,2,...,K
    1. Find a subset of "good" predictors that show fairly strong (univariate) correlation with the outputs, using all of the samples except those in fold k
    2. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in the fold k
    3. Use the classifier to predict the output for the samples in fold k

The Bootstrap

We saw in DA I that an effective way to quantify the uncertainty associated with a given estimator is the *bootstrap*.

The procedure is simple in that we use a computer to emulate the procedure for obtaining new sample data sets by sampling from the observed data set many times with replacement.

Sample B different sets and calculate the quantity of interest about which you want to do inference, say $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$. Then, you can look at this distribution of samples and apply Monte Carlo procedures to do inference. For example, if you were interested in the standard error, you would calculate

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

Bootstrap (cont.)

Although one can use the bootstrap to evaluate model error, it typically over fits because the training and test samples contain some of the same observations (typically). This can be improved by "leave-one-out" bootstrap procedures but there are still typically large training set bias problems with the bootstrap. HTF (2009, Ch. 7.11) give an estimator that helps reduce this bias.

In general, I believe that cross-validation is more effective for evaluating model performance and the bootstrap is more effective when evaluating the properties of estimators so that they can be used for inference.

Final Comments

In some "big data" prediction problems, one often combines cross-validation and hold-out validation. Specifically,

- Use K-fold cross-validation on the training data to choose the parameters that control flexibility of the model
- Use the entire training data to fit the model selected by cross-validation
- Use a large hold-out sample to evaluate the test error (I like to compare this to the cross-validation error)