Cross-Validation and Its Applications

September 1, 2020

Resampling methods are an indispensable tool in modern statistics. They involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model. For example, in order to estimate the variability of a linear regression fit, we can repeatedly draw different samples from the training data, fit a linear regression to each new sample, and then examine the extent to which the resulting fits differ. Such an approach may allow us to obtain information that would not be available from fitting the model only once using the original training sample.

Resampling approaches can be computationally expensive, because they involve fitting the same statistical method multiple times using different subsets of the training data. However, due to recent advances in computing power, the computational requirements of resampling methods generally are not prohibitive.

Here, we discuss the *cross-validation*, one of the most commonly used resampling methods. For example, cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance, or to select the appropriate level of flexibility. The process of evaluating a model's performance is known as *model assessment*, whereas model the process of selecting the proper level of flexibility for a model is known as *model selection.* In this document, we will show two examples to assess the model by selecting proper K for KNN and leading term for linear regression through cross-validation.

1 Cross-Validation

When we deal with the statistical learning through a learning model such as KNN or linear regression, we would love to know how well the model will have for the new observation, which does not include in the training dataset. Furthermore, we would love to test many new observations and obtain the average error rate among those observations. We call this rate the *test error rate.* Given a dataset, the use of a particular statistical learning method is warranted if it results in a low test error. The test error can be easily calculated if a designated test set is available. Unfortunately, this is usually not the case. In contrast, the training error rate can be easily calculated by applying the statistical learning method to the observations used in its training. However, the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter. For example, when we consider KNN model with $K = 1$, the training error rate is 0 since the closest neighbor to each training data point is itself. However, this model will have high test error rate when dealing with the new data point which is different from any training data point.

In the absence of a very large designated test set that can be used to directly estimate the test error rate, a number of techniques can be used to estimate this quantity using the available training data. Here, we consider a class of methods that estimate the test error rate by *holding out* a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

1.1 The Validation Set

Suppose that we would like to estimate the test error associated with fitting a particular statistical learning method on a set if observations. The *validation set* approach is a very simple strategy for this task. It involves randomly dividing the available set of observations into two parts, a *training set* and a *validation set* or *hold-out set*. The model is trained through the training set, and the trained model is used to predict the responses for the data in the validation set. The resulting validation set error rate – typically assessed using the *mean squared error* (MSE) in the case of a quantitative response or using the average classification error in the case of a classification response – provides an estimate of the test error rate.

Although the validation set approach is conceptually simple and is easy to implement, it has two obvious drawbacks:

1) The testing error quite depends on how we separate the dataset into the training and test sets. If those two sets are quite different to each other, the testing result may cause great error. However, if those two sets have almost same distribution, the testing result may have small error.

2) Since we only include partial data for training, the model is not well trained. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate tend to overestimate the test error rate for the model fit on the entire dataset.

Therefore, we need to perform this validation set approach multiple times through different random partition so that all data have the chance to fit into the model. We call this approach, a refinement of the validation set approach, cross-validation and will be discussed in the following two subsections.

1.2 *Leave-One-Out* Cross-Validation (LOOCV)

Leave-one-out cross-validation (LOOCV) involves splitting the set of observations into two parts. However, instead of creating two subsets of comparable size, only single observation is used for the validation set, and all the remaining $N-1$ observations make up the training set.

The general procedure of LOOCV can de described as follows:

Step 1. LOOCV puts the first data point into the testing set. The statistical learning method is fit on the $N-1$ training data, and a prediction $\widehat{y_1}$ is made for the excluded observation through the attribute of the excluded data point. Since the excluded first data point was not used in the training process, $MSE_1 = (y_1 - \widehat{y_1})^2$ provides an approximately unbiased estimate for the test error.

Step 2. LOOCV repeats Step 1 to obtain MSE_2 by setting aside the second data point and training the model with the remaining $N-1$ data points. LOOCV repeats Step 1 to obtain $MSE_3, MSE_4, \ldots, MSE_N$.

Step 3. LOOCV estimates the testing error by taking the average of these N MSEs as $CV = \frac{1}{N}\sum_{i=1}^{N} MSE_i$.

If the statistical learning methods are used for the classification instead of the regression, we only have to change from MSE_i to δ_i, where $\delta_i = 1$ if $y_i \neq \widehat{y_i}$ and $\delta_i = 0$ if $y_i = \widehat{y_i}$.

LOOCV has a couple of major advantages over the validation set approach. First, it has far less bias. In LOOCV, we repeatedly fit the statistical learning method using training sets that contain $N-1$ observations, almost as many as are in the entire data set. This is in contrast to the validation set approach, in which the training set is typically around half the size of the original data set. Consequently, the LOOCV approach tends not to overestimate the test error rate as much as the validation set approach does. Second, in contrast to the validation approach which will yield different results when applied repeatedly due to randomness in the training/validation set splits, performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits.

LOOCV is a very general method and can be used with any kind of statistical learning method. However, LOOCV has the potential to be expensive to implement since the model has to be fit N times. This can be very time consuming if N is large, and if each individual model is slow to fit. Therefore, we need time-efficient approach for cross-validation.

1.3 *k-Fold* Cross-Validation (k-Fold CV)

An alternative to LOOCV is *k-fold CV*. The general procedure is shown as follows.

Step 0. Randomly separate the dataset into k groups, or *folds*, of approximately equal size. We mark k groups as $G_1, G_2, ...G_k$.

Step 1. *k-fold CV* puts the first fold, G_1, as the testing set. The statistical learning method is fit on the remaining data as the training data, and a prediction $\widehat{y_i}$ is made for the i^{th} estimated observation through the attribute of the i^{th} data point in the testing set. $MSE_1 = \frac{1}{|G_1|}\sum_{i \in G_1}(y_i - \widehat{y_i})^2$ provides an estimate for the test error.

Step 2. *k-fold CV* repeats Step 1 to obtain MSE_2 by setting aside G_2, and training the model with the remaining data points. *k-fold CV* repeats Step 1 to obtain $MSE_3, MSE_4, \ldots, MSE_N$.

Step 3. *k-fold CV* estimates the testing error by taking the average of these k *MSE*s as $CV = \frac{1}{k}\sum_{j=1}^{k} MSE_j$.

If the statistical learning methods are used for the classification instead of the regression, we only have to change from MSE_j to δ_j, where δ_j is the number of times of misclassification.

It is not hard to see that LOOCV is a special case of *k-fold CV* in which $k = N$. In practice, one typically performs *k-fold CV* using $k = 5$ or $k = 10$. The most obvious advantage is computational. LOOCV requires fitting the statistical learning method N times. This has the potential to be computationally expensive. However, cross-validation is a very general approach that can be applied to almost any statistical learning method. Some statistical learning methods have computationally intensive training procedures, and so performing LOOCV may pose computational problems, especially if N is extremely large. In contrast, even performing 10-fold CV only requires performing training procedure only 10 times, which may be much more feasible.

2 K Selection for KNN through Cross-Validation

We use cross-validation to select proper K for the KNN classification, as a function of the value of K (which in this context indicates the number of neighbors used in the KNN classifier, rather than the number of CV folds used). The training error rate declines as the method becomes more flexible, and so we see that the training error rate cannot be used to select the optimal value for K. Though the cross-validation error curve slightly underestimates the test error rate, it takes on a minimum very close to the best value for K.

Therefore, we use cross-validation to find the best parameter value of K when the estimated testing error is minimized. The procedure for K selection can be described as follows.

Step 1. Set $K = 1$ for KNN and use *k-fold CV* to estimate the testing error.

Step 2. Set $K = K + 2$ for KNN and use *k-fold CV* to estimate the testing error.

Step 3. If the testing error decreases, goto Step 2; otherwise, we choose current K for KNN.

3 Leading Term Decision for Linear Regression through Cross-Validation

We illustrate the k-fold CV on linear regression. At the linear regression, we say that we can apply linear regression for the nonlinear function. However, how do we determine the leading term for a non-linear relationship. For example, there appears to be a non-linear relationship between *mileage per gallon* (*mpg*) and *horse power* (*hp*), and that a model that predicts *mpg* using *hp* and hp^2 gives better results than a model that uses only a linear term. It is natural to wonder whether a cubic or higher-order fit might provide even better results. We can use the k-fold CV to determine the leading term. The procedure for the leading term decision can be described as follows.

Step 1. Set leading term $n = 1$ (linear relationship) for linear regression and use *k-fold CV* to estimate the testing error.

Step 2. Set $n = n + 1$ for linear regression and use *k-fold CV* to estimate the testing error.

Step 3. If the testing error does not largely improved, stop the process and use $n = n - 1$ as the leading term; otherwise, goto Step 2.

We still determine the non-linear relationship between *mpg* and *hp*. The cubic term in the regression does not lead to better prediction than simply using a quadratic term and therefore, we determine quadratic term as the leading term.