CISC 271 Class 13

Cross-Validating Linear Regression

Texts: Hastie *et al.* [6] pp. 241–249

Main Concepts:

- *Cross-validation: assessing a linear regression*
- *Leave-one-out: simplest cross-validation*
- *K-fold cross-validation: common in practice*
- *Monte Carlo cross-validation: useful for large data sets*

Sample Problem, Machine Inference: How "good" is a linear regression for given data?

So far in this course, we have implicitly assumed that we should measure the quality of a linear regression by examining the overall error, such as the RMS value of the residual error. One conceptual difficulty in doing this is that we used the entire data to compute the regression coefficients and then used the entire data to estimate the residual error.

13.1 Training and Testing

In machine learning, there is usually a distinction between the process of recognizing a pattern and of determining how well the pattern applies to the data.

Training, for us, is the process of finding parameters or coefficients of a function. In the literature, this is most often accomplished by specifying an *objective*, which is a formula to be optimized. The process of fitting a curve, or approximating a function, is training a machine to recognize a specific pattern in the sparse data.

Testing, for us, is the process of evaluating how well the approximation performs. Testing is usually performed with a *score* that may be categorical, such as pass/fail, or may be continuous such as a goodness of fit. For linear regression we will test the goodness of fit.

An unsolved problem in artificial intelligence is how to divide data into a set for training and a set for testing. In this course, so far we have used the same data for training as for testing, with the understanding that there are serious concerns about doing so. Better methods would divide the data into a training set and a testing set (the Matlab default for neural networks is 85% training, 15% testing), or use a cross-validation method such as "leave-one-out".

13.2 Linear Regression and the Design Matrix X

In linear regression, we are provided with observations of the independent variables. So far in this course, we have written each observation as a data vector $\vec{a}_j$ and the dependent data are written as a vector $\vec{c}$. We are assuming that the data are linearly correlated and so we are trying to solve a problem in linear regression.

Recall, in Section 12.5 of these notes, that we observed that there were two choices for how to pose the problem of linearly related data: the "plain" data for linear regression with no intercept, and "augmented" data linear regression with an intercept. For the remainder of this course, we will assume that each observation has enough variables that we do not need to consider the intercept term explicitly. We will adopt some conventions from machine learning and we will write a problem in linear regression as

$$X\hat{w} \approx \vec{y} \tag{13.1}$$

To optimize Equation 13.1, we need to formulate a single error variable that can be minimized. The usual way to assess linear regression is to measure the error between the model and the data. The objective in an ordinary least squares (OLS) solution to linear regression is the squared error, which is the Euclidean norm of the error vector. Because the Euclidean norm of an error vector can be expected to increase with the number of entries in the error vector, it is common to correct for the number of entries and use the RMS error. For a proposed solution $\vec{w}$ to the problem $X\vec{w} \approx \vec{y}$, we can write the RMS error of Equation 12.19 as

$$\text{RMS}(\vec{w}; X, \vec{y}) \stackrel{\text{def}}{=} \sqrt{\frac{[X\vec{w} - \vec{y}]^T[X\vec{w} - \vec{y}]}{m}} \tag{13.2}$$

13.3 Measuring Error in Linear Regression

One difficulty with using the RMS error of a regression is that this is a measure of the *fit* of a model to data. it does not measure how good a linear model is at *predicting* new data.

To measure prediction – which we will call the *test* – we would need to acquire some number l of new data vectors $\vec{x}_{i=n+1\ldots n+l}$ and the corresponding values y_i. We would presumably use RMS error as an evaluation of the test. We could modify Equation 13.2 by using a design matrix derived from the new data in place of X, and a new data vector in place of $\vec{y}$; this would give us the RMS error of *testing* new data with the linear model described by the weight vector $\vec{w}$.

Rather than acquiring new data, a common method in machine learning is to divide the given data into two sets:

1. a *training set* of some size, and
2. a *test set* of the remaining data vectors

The proportions of the sizes of these data sets vary considerably, depending on the source of the data and the proposed application.

An immediate question we might have is: does the computed performance of linear regression depend on exactly which data we use for training? To address this question, we might perform the division more than one time and assess the repeated performance.

There are two general methods used to perform the division into multiple paired sets training data and testing data:

1. in *cross-validation*, the data are partitioned
2. in a *Monte Carlo* method, the data are randomly assigned

We will examine these general methods in turn.

13.4 Cross-Validation of Linear Regression: Leaving Data Out

There are three common ways of performing cross-validation of linear regression. These are *leave-one-out*, *leave-many-out*, and k*-fold validation*.

Leave-One-Out Validation

The simplest method of performing cross-validation of linear regression is to repeatedly hold out one data pair $(\vec{x}_j, y_j)$. These would be written as the "new" data pair $(\vec{u}_j, c_j)$.

The regression would be trained, or computed, to the $m - 1$ other data and the fit would be assessed using the RMS error of Equation 13.2. The hold-out data pair $(\vec{u}_j, c_j)$ would be assessed using Equation 13.2 for $l = 1$.

We could examine the training-subset errors for the m hold-outs and, separately, the testing-subset errors for the same m hold-outs.

The main advantages of leave-one-out validation are simplicity, speed, and the ability to detect a single statistical outlier in the given data $(\vec{x}_i, y_i)$. The main disadvantage is that real data seldom seem to have a single statistical outlier.

Leave-Many-Out Validation

We could modify the leave-one-out algorithm by increasing the number l of hold-out data pairs. This increases the robustness of the validation but at an exponentially increasing cost: to leave out l data pairs from m given data, we would need to perform

$$\begin{pmatrix} m \\ l \end{pmatrix}$$

computations. This is seldom feasible for large data sets.

13.5 Cross-Validation of Linear Regression: K-Fold Analysis

For medium-sized sets of data, a currently common method is *k-fold cross-validation*. We can explore this method by imagining a simple scenario.

Suppose that we are given all of the data at once; that is, the data do not come sequentially over time. We could divide the data into two sets, $\mathbb{S}_1$ and $\mathbb{S}_2$, of equal size. We could then:

1. train the regression on set $\mathbb{S}_1$ and test the regression on set $\mathbb{S}_2$, and then
2. train the regression on set $\mathbb{S}_2$ and test the regression on set $\mathbb{S}_1$

This two-fold validation would give us some idea of how well a linear model can be fit to the given data. A shortcoming is that only 1/2 of the data are used to train the regression; in practice, this is not sufficient to capture the richness of real data.

We could extend this by blending the concept of leave-one-out with the concept of two-fold validation. Suppose that we partition the data into k data sets: $\mathbb{S}_1$, $\mathbb{S}_2$, ..., $\mathbb{S}_k$. We could then, for each data set $\mathbb{S}_j$,

1. train the regression on the combined data in all of the sets $i \neq j$, and then
2. test the regression on the data in set $\mathbb{S}_j$

This process is commonly performed using $k = 5$ for small-size data sets and $k = 10$ for medium-size data sets.

The advantages of k-fold cross-validation are that it requires k passes through the m data pairs, which is linear in the size of the data set, and that it is reasonably robust in the presence of multiple statistical outliers in the given data. If $k = m$, then this reduces to leave-one-out so it is not clear that "bigger is better" for k-fold cross-validation.

A main disadvantage of k-fold cross-validation is that it generates only k samples of how the linear regression performs on the data set. This sparse sampling can produce a high variance in the

tested performance of the regression. If the data set is large, or if differences between the training subset and the testing subset are major concerns, a Monte Carlo method may be a better choice for evaluating the regression.

13.6 Cross-Validation of Linear Regression: Monte Carlo Methods

For very large data sets, it may be computationally ineffective to perform even $k = 5$ or $k = 10$ passes on the data. This can be managed by randomly sampling the data into a training set $\mathbb{S}_1$ and a testing $\mathbb{S}_2$, then proceeding as in k-fold cross-validation.

By repeatedly performing the random sampling, a linear regression can be tested for even a very large data set. Because there is no requirement that the given data are partitioned, it is possible that there is some value i such that

$$\mathbb{S}1 \cup \mathbb{S}_2 \neq \{(\vec{x}_i, y_i)\}$$

The main advantage and disadvantage of Monte Carlo cross-validation of linear regression is that not all of the given data are used. For small data sets, this might mean that crucial information is not tested. For large data sets, this might mean that the method is computationally effective.

13.7 Example: 13 Data With 2 Outliers

We can use 5-fold cross-validation to study the effect of linear regression on a simple data set. Suppose that we use the sample data that is shown in Table 13.1.

When we standardize the data in Table 13.1 as a design matrix X and a dependent vector $\vec{y}$, and we reconstruct the regression and compare it to the data vector $\vec{c}$, we compute the values

$$\begin{array}{rclrcl} \bar{c} & = & 19.2974 & w & = & 0.9936 \\ \sigma_{\vec{c}} & = & 11.6885 & \text{RMS} & = & 0.1088 \end{array} \tag{13.3}$$

The data in Table 13.1, and the linear regression that has the values in Equation 13.3, are illustrated in Figure 13.1.

Let us assess the solution by performing 10 passes of 5-fold cross-validation. We will combine RMS errors by taking the RMS of the RMS errors, which gives us the data in Table 13.2.

Table 13.1: Sample data that are derived from a linear function with two outliers. Data are shown using four digits of numerical precision.

$\vec{a}$	$\vec{c}$
0.0000	-1.8584
1.0000	5.8599
2.0000	8.5782
3.0000	11.2964
4.0000	14.0147
5.0000	16.7330
6.0000	19.4513
7.0000	22.1696
8.0000	24.8878
9.0000	27.6061
10.0000	30.3244
11.0000	33.0427
12.0000	38.7610

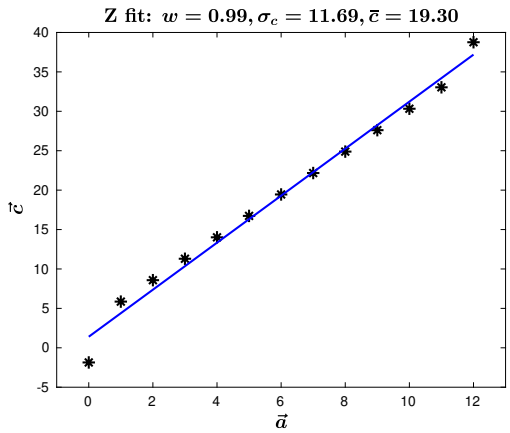

Figure 13.1: Sample data that are derived from a linear function with two outliers, shown as black asterisks, and a linear regression, shown as a blue line.

Table 13.2: RMS of fits for training and testing subsets in a 5-fold cross-validation of OLS.

	Train	**Test**
	0.7372	1.1496
	1.1504	1.8840
	1.2280	1.7711
	1.3208	1.8735
	1.2622	2.2049
	1.2340	1.8694
	1.2712	1.7658
	0.0000	0.0000
	0.7353	1.1707
	1.3415	1.6849
Mean:	1.0280	1.5374
Std:	0.4246	0.6294

Observations on Table 13.2:

- For OLS, the test errors appear to be substantially greater that the training errors
- The mean of OLS training fits is low and the variance is high, suggesting a *poor model* of the data
- The mean of OLS tests is higher than the mean of training, and the variance is also higher, suggesting a *poor model* and *high variance* in tests

Statistical explanations of an estimator's bias and variance have a long history; consequently, many thorough analyses are available. The interested student is encouraged to explore the relationship between bias, variance, and intrinsic error for a model that includes a random variate with a Gaussian distribution.