Outlier Detection in Neuro-Fuzzy Modeling

Jyh-Shing Roger Jang
CS Department, Tsing Hua University, Hsinchu, Taiwan
Email: jang@cs.nthu.edu.tw

Keywords: Fuzzy modeling, neuro-fuzzy modeling, outlier detection, linear/nonlinear regression, leave-one-out error, least-squares estimator.

Abstract

This paper describes an efficient method to compute the leave-one-out (LOU) error of a model that is linear in the parameters. The LOU error can be used for detecting outliers in a given data set. An example of detecting outliers in polynomial fitting is used to show the method's concept. The proposed method is used in neuro-fuzzy modeling for a public-domain data set and it successfully identifies an outlier that could be introduced during manuscript editing when the data set was published.

1. Introduction

One of the most accurate methods to estimate the performance of a pattern classifier is the **leave-one-out technique**, also called the **jackknife procedure** [1]. In the procedure, one sample from a data set containing m samples is saved for testing and a classifier is designed using the remaining $m-1$ samples. The sample that was withheld is then tested. The procedure is repeated for each sample; the average error rate of the design method is k/m, where k is the number of errors. The estimate of error rate is *unbiased* because no samples were used for both training and testing a given classifier. Moreover, the estimate of error rate will be as accurate as possible since all m samples were used for testing. In fact, the technique nearly doubles the effective size of the data set when compared with the common practice of dividing the data set into disjoint training and test sets.

For regression problems, a **leave-one-out (LOU) error** can be defined similarly. An efficient matrix manipulation technique for computing the LOU error of a linear model is proposed in our previous work [4], and the LOU error is used as an unbiased performance index for input selection. In this paper, we shall revisit the computation of the LOU error and propose a way of using the error for outlier detection for models that are linear in the parameters. Such models include simple linear regression models and complex neuro-fuzzy models. We shall demonstrate the concept using an example of outlier detection in polynomial fitting. An advanced example of outlier detection in neuro-fuzzy modeling will also be described, where the proposed method is embedded in a neuro-fuzzy modeling approach (ANFIS [3, 6]) for a public-domain data set. The method can successfully identified an outlier which could be introduced during the manuscript editing when the data set was published [7].

2. An Incremental Method for Computing Leave-one-out Errors

This section briefly reviews the efficient matrix manipulation technique [4] for computing the leave-one-out (LOU) least-squares estimator of a linear model. Based on the technique, we then derive a method to derive the LOU error directly, without knowing the LOU least-squares estimator in advance. The proposed method is applied to outlier detection in polynomial fitting and neural-fuzzy modeling, as described in the next section.

The least-squares problem can be formulated as the solution to a set of linear equations:

$$\boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{y},$$

where $\boldsymbol{A}$ is a $m \times n$ matrix; $\boldsymbol{y}$ is a $m \times 1$ vector; and $\boldsymbol{\theta}$ is an $n \times 1$ vector of unknown linear parameters. In general, the number of training data m is greater than the number of parameters n, so there is no exact solution $\boldsymbol{\theta}$ to the above equation. Instead, we seek to find the **least-squares estimator** (LSE) $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ that can minimize $E(\boldsymbol{\theta})$:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\theta}).$$

*Presented in International Joint Conference on Information Science, Research Triangle, 1998.

†Research supported by National Science Council NSC88-2213-E-007-007 (Error Estimate and Structure Determination in Soft Computing), Taiwan

The solution to the above minimization problem can be solved by a number of methods [6], including direct differentiation, completing the square, and intuitive geometric interpretation. The resulted formula is

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{y}. \tag{1}$$

For on-line computation, the above formula can be transformed into an equivalent iterative form, called the **recursive least-squares estimator** [2]:

$$\begin{cases} _{k+1} &= _k - \frac{_k \, _{k+1} \, ^T_{k+1} \, _k}{1 + ^T_{k+1} \, _k \, _{k+1}}, \\ \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + _{k+1} \, _{k+1} (y_{k+1} - ^T_{k+1} \, \boldsymbol{\theta}_k), \end{cases} \tag{2}$$

where k ranges from 0 to $m-1$ and the overall least-squares estimator $\hat{\boldsymbol{\theta}}$ is equal to $\boldsymbol{\theta}_m$, the estimator using all m data pairs.

By applying the same concept of the recursive least-squares estimator, Jang [4] has proposed an efficient way of computing the leave-one-out (LOU) least-squares estimator. The least-squares estimator without using the pth data pair is computed as

$$\hat{\boldsymbol{\theta}}_{\bar{p}} = (\boldsymbol{A}^T_{\bar{p}} \boldsymbol{A}_{\bar{p}})^{-1} \boldsymbol{A}^T_{\bar{p}} \boldsymbol{y}_{\bar{p}}, \tag{3}$$

where $\boldsymbol{A}_{\bar{p}}$ is obtained from $\boldsymbol{A}$ after deleting its pth row ^T_p, and $\boldsymbol{y}_{\bar{p}}$ is obtained from $\boldsymbol{y}$ after deleting its pth element y_p. The efficient formula proposed by Jang [4] can be expressed as

$$\hat{\boldsymbol{\theta}}_{\bar{p}} = \hat{\boldsymbol{\theta}} - _{\bar{p}} \, (y - ^T \, \hat{\boldsymbol{\theta}}), \tag{4}$$

$$_{\bar{p}} = + \frac{^T}{1 - ^T}, \tag{5}$$

where is the inverse of $\boldsymbol{A}^T \boldsymbol{A}$, and $[^T; \ y]$ is the deleted pth input-output data pair. (The subscript p for $[^T; \ y]$ is left out for simplicity.) The computational complexity of the above formula is analyzed in detail in ref. [4]. The obtained LOU LSE $\hat{\boldsymbol{\theta}}_{\bar{p}}$ can be used to check the effects of deleting pth sample data pair.

For easy reference, we shall define two type of RMSE (root mean square error) associated with the pth sample data. The LOU test RMSE for pth sample data is

$$\text{LOU-RMSE}_{\text{test}} = |y_p - ^T_p \, \hat{\boldsymbol{\theta}}_{\bar{p}}|. \tag{6}$$

The LOU training RMSE for pth sample data is

$$\text{LOU-RMSE}_{\text{training}} = \sqrt{\frac{1}{m-1} \sum_{i=1, i \neq p}^{m} (y_i - ^T_i \, \hat{\boldsymbol{\theta}}_{\bar{p}})^2}. \tag{7}$$

In practice, we do not need to know $\hat{\boldsymbol{\theta}}_{\bar{p}}$ in order to compute the LOU test RMSE defined in Equation (6). This is derived as follows.

First, let us define two types of errors associated with the pth sample data pair ^{T} and y:

$$e_p = y - ^T \, \hat{\boldsymbol{\theta}}, \tag{8}$$

$$e_{\bar{p}} = y - ^T \, \hat{\boldsymbol{\theta}}_{\bar{p}}, \tag{9}$$

where e_p is the resubstitution error using the overall LSE $\hat{\boldsymbol{\theta}}$, and $e_{\bar{p}}$ is the test error using the LOU LSE $\hat{\boldsymbol{\theta}}_{\bar{p}}$. After plugging Equation (4) into Equation (9), we have

$$\begin{aligned} e_{\bar{p}} &= y - ^T \, \hat{\boldsymbol{\theta}} \\ &= y - ^T \, [\hat{\boldsymbol{\theta}} - ^T_{\bar{p}} \, (y - ^T \, \hat{\boldsymbol{\theta}})] \\ &= y - ^T \, \hat{\boldsymbol{\theta}} + ^T_{\bar{p}} \, ^T (y - ^T \, \hat{\boldsymbol{\theta}}) \\ &= (1 + ^T_{\bar{p}} \, ^T)(y - ^T \, \hat{\boldsymbol{\theta}}) \\ &= (1 + ^T_{\bar{p}} \, ^T) e_p \end{aligned} \tag{10}$$

We pre- and post-multiply Equation (5) by $^T_{\bar{p}}$ and $_{\bar{p}}$, yielding a scalar equation:

$$\begin{aligned} ^T_{\bar{p}} \, _{\bar{p}} &= ^T + \frac{^T \, ^T}{1 - ^T} \\ &= \frac{^T}{1 - ^T}, \end{aligned}$$

or equivalently,

$$1 + ^T_{\bar{p}} \, _{\bar{p}} = \frac{1}{1 - ^T}. \tag{11}$$

By plugging the above equation into Equation (10), we have the following relationship between e_p and $e_{\bar{p}}$:

$$\begin{aligned} e_{\bar{p}} &= (1 + ^T_{\bar{p}} \, ^T) e_p \\ &= \frac{e_p}{1 - ^T \, ^T}. \end{aligned} \tag{12}$$

In other words, the steps to find $e_{\bar{p}}$ for p = 1 to n can be summarized as follows:

1. Find the matrix as the inverse of $\boldsymbol{A}^T \boldsymbol{A}$.
2. Find the overall least-squares estimator $\hat{\boldsymbol{\theta}} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{y} = \boldsymbol{A}^T \boldsymbol{y}$.
3. Compute the resubstitution error e_p associated with the pth data pair, for p = 1 to m, according to Equation (8).
4. Compute the LOU error $e_{\bar{p}}$ associated with the pth data pair, for p = 1 to m, according to Equation (12).

If we count multiplication/division or addition/subtraction as a single FLOP (floating point operation), then steps 1 and 2 requires $m(6n^2 + 5n - 1) \approx 6mn^2$ FLOPs [4]. Similarly, step 3 requires $2mn$ FLOPs and step 4 requires $m(4n^2 - n + 1) \approx 4mn^2$. Therefore the total number of FLOPs required to compute the LOU errors $e_{\bar{p}}$, for p = 1 to m, is equal to $10mn^2 + 4mn = m(10n^2 + 4n) \approx 10mn^2$. This computational complexity compares favorably with

$6m^2n^2$ [4], which is the complexity of the naive approach that computes $\hat{\boldsymbol{\theta}}_{\bar{p}}$ from scratch every time the pth data is removed.

Since $e_{\bar{p}}$ represents the fitting error of a model designed via all sample data except the pth one, the magnitude of $e_{\bar{p}}$ indicates the difference between the pth data and all the other sample data in terms of the identified model. In other words, the bigger the magnitude of $e_{\bar{p}}$ is, the more likely the pth sample data is an outlier.

Note that one may choose to use the magnitude of e_p as a criterion for detecting outliers. However, according to Equation (12), the ranking of e_p does not necessarily have the same order as that of $e_{\bar{p}}$. Therefore detecting outliers according to $e_{\bar{p}}$ is a more justified approach.

3. Computational Complexity

The computation complexity of Equation (2) for a given k is listed in the following two tables. (For simplicity, the subscript indices of k is deleted.)

Steps	Operations	Counts for mul. or div.	Counts for add. or sub.
1		n^2	$n(n-1)$
2	$= \quad [][]^T$	n^2	0
3	$[]$	n	$n-1$
4	$1+[]$	0	n^2
5	$\frac{[]}{[1+]}$	n^2	0
6	$-\left[\frac{}{1+}\right]$	0	n^2
1-6	FLOPs from 1 to 6	$3n^2+n$	$3n^2-1$
1-6	Total FLOPs	$6n^2+n-1$	

Steps	Operations	Counts for mul. or div.	Counts for add. or sub.
1	$\boldsymbol{\theta}$	n	$n-1$
2	$y-[\boldsymbol{\theta}]$	0	1
3	$[][y-\boldsymbol{\theta}]$	0	n
4	$\boldsymbol{\theta}+[(y-\boldsymbol{\theta})]$	0	n
1-4	FLOPs from 1 to 4	n	$3n$
1-4	Total FLOPs	$4n$	

In the above two tables, each square-bracketed quantity (in the "Operations" column) is readily available from the previous steps. If we count both multiplication/division and addition/subtraction as a single FLOP (floating point operation), then the computation of Equation (2) requires $(6n^2+n-1)+(4n)=6n^2+5n-1$ FLOPs and it takes $m(6n^2+5n-1)\approx 6mn^2$ FLOPs to find the least-squares estimator $\boldsymbol{\theta}$. Note that the formulas to compute $\boldsymbol{\theta}_{\bar{p}}$ from $\boldsymbol{\theta}$ (see Equation (4) and (5)) are almost the same as the formulas in Equation (2), hence the complexity for computing$\boldsymbol{\theta}_{\bar{p}}$ can be derived accordingly as explained in the following.

Now we have two methods to compute the leave-one-out RMSE. For a simple-minded approach, we need to compute the least-squares estimator $\boldsymbol{\theta}_{\bar{p}}$ for $p=1$ to m. If it is done via the recursive formulas in Equation (2), the whole process requires about $(m-1)[6(m-1)n^2]\approx 6m^2n^2$ FLOPs. In other words, it takes m times as long to compute all $\boldsymbol{\theta}_{\bar{p}}$, $p=1$ to m. This could be inhibitively long since m is the number of training data pairs and its value, though depends on applications, is usually more than 100.

To reduce the computation load, we can employ the incremental method proposed earlier. First we need to find the overall and $\boldsymbol{\theta}$ when all data set is used. This requires about $6mn^2$ FLOPs if Equation (2) is used. Then we need to remove the effect of each data pair one at a time; this requires another $6mn^2$ FLOPs. So the total number of FLOPs is about $12mn^2$, which is much smaller than that ($6m^2n^2$) of the simple-minded approach. Therefore via the proposed approach, it only takes twice as long to compute all $\boldsymbol{\theta}_{\bar{p}}$, $p=1$ to m, as to compute $\boldsymbol{\theta}$.

If we are more interested in obtaining the LOU test error, we can simply apply Equation (12) to save more computation. The computational complexity of Equation (12) is given next.

Steps	Operations	Counts for mul. or div.	Counts for add. or sub.
1		n^2	$n(n-1)$
2	$[]$	n^2	$n(n-1)$
3	$1-[]$	0	n
4	$\frac{e_p}{[1-]}$	1	0
1-4	FLOPs from 1 to 4	$2n^2+1$	$2n^2-n$
1-4	Total FLOPs	$4n^2-n+1$	

4. Experimental Results

4.1. Example 1: Outlier Detection in Polynomial Fitting

In this example, we use the proposed method to efficiently compute LOU test RMSE for polynomial fitting, and demonstrate that the obtained $e_{\bar{p}}$ can be used as an indicator for finding possible outliers. The sample data is obtained

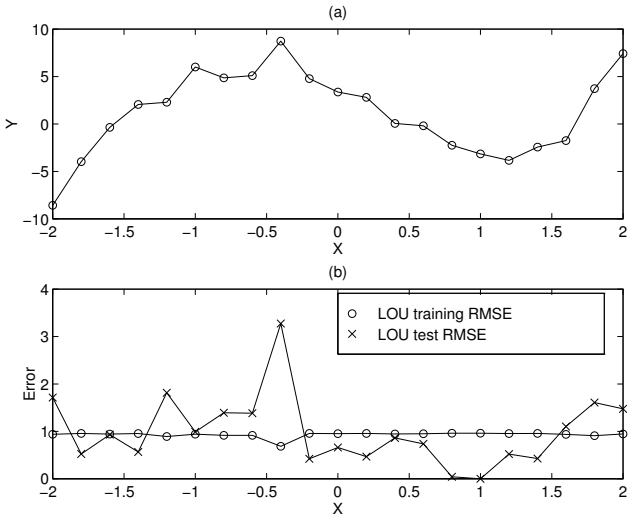


Figure 1. *Outlier detection in polynomial fitting: (a) Noisy sample data (circles). (b) LOU training RMSE (circles) and LOU test RMSE (crosses) with respect to x.*

from a fourth order polynomial:

$$y = \underbrace{0.3256x^4 + 3.0088x^3 - 2.5463x^2 - 7.4908x + 3.7636}_{signal} + \underbrace{n}_{noise}, \quad (13)$$

where n is a Gaussian distributed random variable with zero mean and unity variance. We picked 21 x values (that are evenly distributed from -2 to 2) as the input part of the sample data; the corresponding output part was obtained by evaluating Equation (13) at the 21 x values. These 21 sample data are all we have, and our mission is to identify the optimal order for the fitting polynomial, and then the likelihood of each data point being an outlier.

The optimal order of the fitting polynomial occurs when the LOU test RMSE is minimal, as described in ref [4]. Thus the optimal order is found to be 4. Then we can examine each $e_{\bar{p}}$, for p = 1 to 21, to find the data point with maximal $e_{\bar{p}}$. The result is shown in Figure 1 (b), where the maximum occurs when $p = 9$ or $x = -0.4$. We conclude that the data point at $x = -0.4$ is more likely to be an outlier than all the other points. Visual inspection on Figure 1 (a) also conforms to our conclusion. However, visual inspection becomes harder when we are dealing with high-dimensional data. Hence we have to rely on the proposed computational routine to identify outliers for high-dimensional data.

4.2. Example 2: Outlier Detection in Neuro-Fuzzy Modeling

In this example, we apply the proposed method to compute the LOU errors of ANFIS [3, 6] for a data set from a nonlinear function [7]. The data set is generated from a nonlinear function:

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2, \ 1 \le x_1, x_2 \le 5. \quad (14)$$

A surface plot of the above function is shown in Figure 2 (a). 50 data points were collected according to the above equation, as printed in Table II of ref. [7]. The scatter plot of these 50 data points, together with their indices, are shown in Figure 2 (b). The data set is also available from the "working group on data modeling benchmarks", at

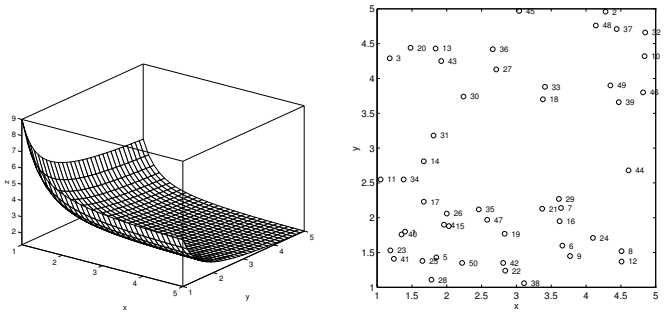
`http://neural.cs.nthu.edu.tw/jang/benchmark/`

(Note that the original data set contains 4 inputs, two of which are dummy inputs that can be easily removed by using input selection techniques in refs. [7] or [4]. The following discussion assumes that the correct inputs have been selected already.)

We shall use the ANFIS approach to model the data set. In general, ANFIS is a nonlinear model that consists of both premise and consequent parameters. However, if the premise parameters are fixed, then ANFIS becomes a linear-in-the-parameters model and the proposed method to find $e_{\bar{p}}$ can be applied directly.

For simplicity, we adopt the grid partitioning [5] of the input space, and each input is assigned three generalized bell membership functions. The computed $e_{\bar{p}}$ for p = 1 to 50 is shown in Figure 3. Apparently the most significant peak corresponds to the 42nd data point. When further examined this data point, we found that it is indeed an outlier since the input-output pair does not satisfy Equation (14) at all. If we assume the input vector is correct, then the output should be 3.11 instead of 1.97, as listed in Table II of ref. [7]. Therefore we conclude that the proposed method successfully found an outlier, which is likely to be introduced during manuscript editing of ref. [7].

The other two significant peaks corresponds to the 11th and 38th data point. These two data points are not outliners since they all satisfy Equation (14). Their unusual large LOU test errors are due to the fact that they are located at the boundary of the whole data set. This can be seen from Figure 2 (b), where the 11th data point is located left-most while the 38th data point is located lowest in vertical direction. These boundary points are likely to induce high LOU errors and thus should be taken into consideration when trying to detect outliers.



(a) (b)

Figure 2. *Data set from a nonlinear function: (a) surface plot of the nonlinear function; (b) scatter plot of the data set.*

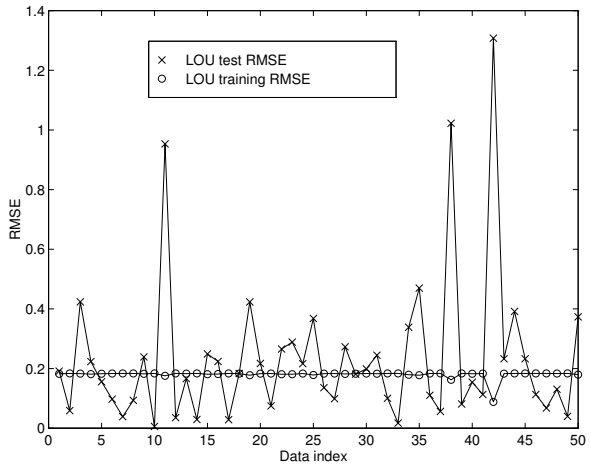


Figure 3. *LOU training and test RMSEs* **.**

5. Concluding Remarks

We have proposed an incremental method to efficiently compute the LOU (leave-one-out) error of a linear model. We have analyzed the computational complexity of the proposed method, and applied it to identify outliers in polynomial fitting and neuro-fuzzy modeling. In particular, the method did found an outlier in a public-domain data set of a nonlinear function.

Sample data in high-dimensional space tend to be sparse, which inevitably aggravates the boundary effect that assigns large LOU test errors to boundary data points, as demonstrated in the example of neuro-fuzzy modeling. How to take boundary effects into consideration when detecting outliers using LOU error is currently under study.

References

[1] E. Gose, R. Johnsonbaugh, and S. Jost. *Pattern Recognition and Image Analysis*. Prentice Hall, Upper Saddle River, NJ, 1996.

[2] T. C. Hsia. *System identification: least-squares methods*. D. C. Heath and Company, 1977.

[3] J.-S. R. Jang. ANFIS: Adaptive-Network-based Fuzzy Inference Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(03):665–685, May 1993.

[4] J.-S. R. Jang. Leave-one-out regularity criterion for input selection in fuzzy modeling. In *Proceedings of IEEE International Conference on Fuzzy Systems*, Alaska, May 1998.

[5] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *The Proceedings of the IEEE*, 83(3):378–406, Mar. 1995.

[6] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. MATLAB Curriculum Series. Prentice Hall, Upper Saddle River, NJ, 1997.

[7] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, Feb. 1993.