Advances in Engineering Research, volume 127
3rd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2018)

Application of Machine Learning Method in Simulation Model Validation

Qi Lin* and Yong Chen
Space Engineering University, Beijing, China
*Corresponding author

***Abstract*—There exists distrust in simulation validation all the time. A quantitative approach is proposed to obtain measurable, comparable judgments of simulation correctness. The commonality between machine learning and simulation model validation is analyzed. We focus on the idea of applying cross validation in the area of simulation validation. Based on cross validation, a strategy is proposed to predict the fit of a simulation model to a validation set. Scaling factor is then introduced into the approach to improve its efficiency. The approach is applied in a simulation system to verify the usefulness of the approach proposed. The result shows it is convinient to get an effective estimate of correctness of simulation models with the method.**

Keywords—machine learning; simulation validation; error estimate; data composition

I. INTRODUCTION

Simulation is presented as a practical technique and is indispensable in several different tasks [1] because of the merits of simulation such as effectiveness and efficiency [2]. The use of high-speed, general-purpose digital computers to control and coordinate the elements of real-time physical systems poses several challenging problems in the area of system test and evaluation. First of all, its main blink side is the lack of correctness confirmation or credibility analysis of simulation itself [3]. It is necessary to design quantitative approaches which can be applied efficiently so as to obtain measurable, comparable judgments of simulation correctness. As there exists some similarities between simulation and machine learning, and machine learning has been deeply studied, it is sensible to explore the correspondence between the two areas and apply well-tested machine learning methods in simulation in an apropriate way.

II. MACHINE LEARNING AND SIMULATION

In order to propose an efficient method to validate the correctness of simulation models with the ideas we mentioned above, it is necessary to identify the common and different points between simulation and machine learning.

Machine learning is the procedure of fitting models or arguments of models with training sets, and then applying the models on validation sets to get the errors of the entire hypothesis. Based on the errors, the learning model with the least error will be picked up as the final model. For example, cross validation in machine learning takes full advantage of the cross or intersection of the two data sets. In this respect, it is a partition on the whole data set.

Simulation is another thing in which simulation model will be derived by physical attributes or indexes. What we should do is to confirm the correctness of the model based on simulation data and test data. There exists something in common that the data set is consistent with the model no matter which data set it is. This is the key point to combine the ideas of machine learning into the area of simulation.

III. A STRATEGY BASED ON CROSS-VALIDATION

A. Cross-validation

Cross-validation is a common used technique in the area of machine learning, which is a built–in automatic method of self testing a model for reliability to access how the results of a statistical analysis will generalize to an independent data set[4].

When evaluating different settings for estimators, there is a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can “leak” into the model and evaluation metrics no longer report on generalization performance. Conventional validation often partitions the data set into two sets for training and test at a fixed ratio, in which the error (e.g. Root Mean Square Error) on the training set is not a useful estimator of model performance and thus the error on the test data set does not properly represent the assessment of model performance because there are not enough data available or there is not a good distribution and spread of data to partition it. Meanwhile, by partitioning the available data into training set, validation set and test set, the number of samples which can be used for learning the model can be drastically reduced, and the results can depend on a particular random choice for the pair of (train, validation) sets. To solve this problem, Cross-Validation (CV) is introduced to combine or average measures of prediction error to correct for the optimistic nature of training error and derive a more accurate estimate of model prediction performance [5].

B. Leave-one-out cross-validation

In the basic Cross-Validation approach, called k-fold CV, the training set is split into k smaller sets. The following procedure is applied for each of the k “folds” [6]:

- A model is trained using k-1 of the folds as training data;
- The resulting model is validated on the remaining part of the data, which is used as a test set to compute a performance measure such as accuracy.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. K-fold CV methods are non-exhaustive cross validation methods which do not compute all ways of splitting the original sample, and are approximations of leave-p-out cross-validation. As an exhaustive CV, Leave-p-out cross-validation (LOOCV) involves using p observations as the validation set and the remaining observations as the training set. This is repeated on all ways to cut the original sample on a validation set of p observations and a training set. Leave-one-out cross-validation (LOOCV) is a particular case of leave-p-out cross-validation with p=1. LOOCV doesn't have the calculation problem of general LPOCV because $C_1^n = n$.

C. LOOCV Error Estimation

Given n models $H_1, H_2, ...H_n$ and n corresponding algorithms $A_1, A_2, ...A_n$, the goal is to pick a H_m with small $E_{out}(g_m)$ instead of just $E_{in}(g_m)$. E_{out} means out-of-sample error which shows the extent to which the average prediction over all data sets, including test sets and even unknown future data. In this respect, E_{out} is an index to measure the generalizability of models. In contrast, in-sample error E_{in} is sensitive to the observed data or validation sets and cannot represent real spread of the models. The selection is made by test E_{test}, which is evaluated on a fresh test set (D_{test}). This can be described as:

$$m^* = \arg\min_{1\le m\le n}(E_m = E_{test}(A_m(D))) \tag{1}$$

According to finite-bin Hoeffding, generalization guarantee can be obtained:

$$E_{out}(g_{m^*}) \le E_{test}(g_{m^*}) + O(\sqrt{\frac{\log M}{N_{test}}}) \tag{2}$$

However, D_{test} is difficult to obtain in reality, so validation set D_{val} which is on-hand simulation of test set is introduced to connect E_{val} (evaluated on D_{val}) with E_{out} provided that D_{val} is clean. In order to make sure D_{val} is clean, only D_{train} can be fed into A_m for model selection. If D_{val} is kept as size of *K*, then the size of D_{train} is *N-K*.

So (1) can be modified as follows:

$$m^* = \arg\min_{1\le m\le n}(E_m = E_{val}(A_m(D_{train}))) \tag{3}$$

And (2) is changed to:

$$E_{out}(g_{\bar{m}}) \le E_{val}(g_{\bar{m}}) + O(\sqrt{\frac{\log M}{K}}) \tag{4}$$

$g_{\bar{m}}$ ($1 \le m \le n$) is obtained by applying $H_{\bar{m}}$ on D_{train}, and is used to apply on D_{val} to get error estimation from which the best $g_{\bar{m}}$ with the least error can be picked up as g_{m^*}:

$$E_{out}(g_{m^*}) \le E_{out}(g_{\bar{m}^*}) \tag{5}$$

According to (4), we can get:

$$E_{out}(g_{m^*}) \le E_{out}(g_{\bar{m}^*}) \le E_{val}(g_{\bar{m}^*}) + O(\sqrt{\frac{\log M}{K}}) \tag{6}$$

Take $K = 1$ as the extreme case, the LOOCV estimate is calculated as follows:

$$E_{LOOCV}(H, A) = \frac{1}{N}\sum_{n=1}^{N} e_n = \frac{1}{N}\sum_{n=1}^{N} err(g_{\bar{n}}(x_n), y_n) \tag{7}$$

So expected $E_{LOOCV}(H, A)$ (depicted as $\underset{D}{\mathcal{E}} E_{LOOCV}(H, A)$) can be transformed as follows:

$$\begin{aligned}
\underset{D}{\mathcal{E}} E_{LOOCV}(H, A) &= \underset{D}{\mathcal{E}} \frac{1}{N}\sum_{n=1}^{N} e_n = \frac{1}{N}\sum_{n=1}^{N} \underset{D}{\mathcal{E}} e_n \\
&= \frac{1}{N}\sum_{n=1}^{N} \underset{D}{\mathcal{E}} \underset{D}{\mathcal{E}} err(g_{\bar{n}}(x_n), y_n) \\
&= \frac{1}{N}\sum_{n=1}^{N} \underset{D}{\mathcal{E}} E_{out}(g_{\bar{n}}) \\
&= \frac{1}{N}\sum_{n=1}^{N} \overline{E_{out}}(N-1) \\
&= \overline{E_{out}}(N-1)
\end{aligned}$$

As such,

$$\underset{D}{\mathcal{E}} E_{LOOCV}(H, A) = \overline{E_{out}}(N-1) \tag{8}$$

It means expected $E_{LOOCV}(H, A)$ is almost an unbiased estimate of $E_{out}(g)$.

D. Why LOOCV

When an explicit validation set is not available, LOOCV will be a good way to predict the fit of a model to a hypothetical validation set, which gives us a hint to apply the idea in the area of simulation.

From the above deduction, it can be concluded that $E_{LOOCV}(H, A)$ has close relationship with the goal we mentioned before, which is the lowest $E_{out}(g)$. Although there exists some

criticisms on LOOCV so that someone prefers iterated k-fold CV to LOOCV if models are not stable, this is not the case of simulation in which models are stable enough to omit the random uncertainty problem which LOOCV is subject to.

In addition, simulation involves time-series and other data where the object order matters, so LOOCV-based method is perfect and of great help to estimate how accurately a simulation model will perform in practice.

IV. IMPROVED WITH ADABOOST

By means of LOOCV, data set can be made full use of without any waste. However, this approach can be computationally expensive. In order to solve the problem, scaling factor is introduced to be used in combination to the LOOCV-based method.

The idea of scaling factor comes from adaptive boosting (AdaBoost), which can be used in conjunction with many other types of learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent weak algorithms are tweaked in favor of those instances misclassified by previous classifiers [7]. It is often referred to as the best out-of-the-box classifier. In short, scaling factor is used to perform 'optimal' re-weighting, which can be defined as follows:

$$> t = \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \tag{9}$$

where ε_t is weighted error rate of g_t.

A preliminary AdaBoost algorithm can be described as follows:

$$\text{Given } u^{(1)} = [\frac{1}{N}, \frac{1}{N}, ... \frac{1}{N}] \tag{10}$$

1) Obtain g_t by $A(D, u^{(t)})$, where A tries to minimize $u^{(t)}$-weighted error ;

2) Update $u^{(t)}$ to $u^{(t+1)}$ by $> t$.

3) Compute $\alpha_t = \ln(> t)$

Return $G(x) = sign(\sum_{t=1}^{T} \alpha_t g_t(x))$

If the algorithm is applied on incorrect data, the corresponding weight will be multiplied by scaling factor $> t$, or else, with correct data, the corresponding weight will be divided by $> t$.

In general, error rate is no more than 50% which can be depicted as $\varepsilon_t \le \frac{1}{2}$. Thus the physical meaning of scaling factor is to scale up the weight of incorrect samples and scale down the weight of correct samples. Good g_t means smaller error ε_t, accordingly, the scaling factor $> t$ will be larger so that we can get larger α_t which will influence the re-weighting process so as to improve the efficiency of learning.

V. PROCEDURE AND STEPS

With the above analysis in mind, it is necessary to make sure how to use simulation data in a right way so as to combine the LOOCV method into simulation validation. The validation approach can be represented as Figure I.

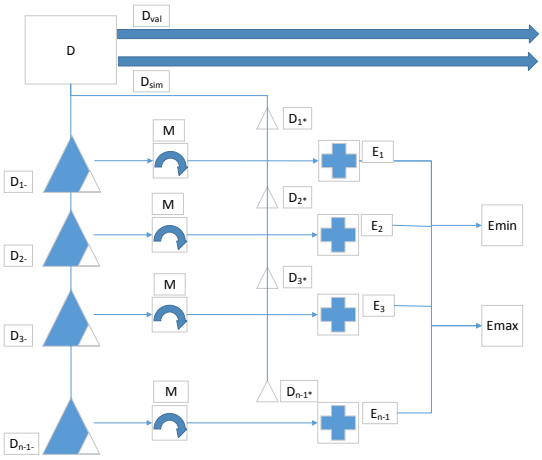

FIGURE I. PROPOSED VALIDATION APPROACH

First of all, it is necessary to build whole data set by combining D_{sim} and D_{val}. Two problems should be solved in this phase: data source and data size.

It is obvious to use data generated by simulation applications as simulation data set D_{sim}. Meanwhile, the ideal validation set D_{val} can be provided by experimental environment. However, if it is too hard or infeasible to provide such a physical system, there is always a corresponding prototype system which applies the same simulation models to get emulation results. It is appropriate to use the emulation results as a source of validation set D_{val}.

Compared to validation set D_{val}, the D_{sim} can be obtained much easier because simulation can run as required. So if the size of D_{val} is described as *K*, the size of D_{sim} can be a linear function of *K*, which is defined as:

$$f(K) = aK + b$$

Where *a* and *b* are constants chosen according to practical requirements. A rule of thumb is to select *a* from the a range [0,1] $(0 \le a \le 1)$ and to select *b* from a range [1,5] $(1 \le b \le 5)$.

Apply the basic LOOCV approach to the data set just built, with *N = K+ f(K)*:

1. let *i* =1

2. Let (x_i, y_i) be the ith record (*Di**)

3. Temporarily remove (x_i, y_i) from the *D* (*Di-*)

4. Compute error with least-squares linear or polynomial regression. A scaling factor can be introduced in this step to perform optimal re-weighting so as to improve the efficiency.

As D_{val} is taken from real experiments or from prototype system, it can be considered as correct data that we can adjust the weight to be multiplied by a scaling factor $>t$ when we calculate error in this case. On the other side, it is not for sure about the correctness of D_{sim}, so the corresponding weight should be divided by the scaling factor $>t$.

$$Z_i = \frac{\sum_{k \in Di-} u^{(i)} (y_i - \hat{y}_k)^2}{N-1} \tag{11}$$

$$E_i = \left| \frac{Z_i - \hat{Z}_i}{\hat{Z}_i} \right| \tag{12}$$

Where, $\hat{y}_k$ is obtained from Di-, initial $u^{(i)}$ is calculated with the algorithm introduced in section VI.

$$u^{(i)} = \begin{cases} u^{(i)} * > t & (\hat{y}_k \in D_{val}) \\ u^{(i)} / > t & (\hat{y}_k \in D_{sim}) \end{cases} \tag{13}$$

The main point is to adjust the factors to put different annotations on the data in sample space and re-weight the calculation components, which would be a boost for validation process.

5. If $i < N$, goto 2, else increase i by 1.

6. Sort all E_i.

Based on the sorted error sequence, select the maximum E_i as $E_{\max}$, and the minimum E_i as $E_{\min}$, which can define the error scope of the simulation model. In turn, the error scope generated from the process is an effective estimate of $E_{out}(g)$ which can represent the correctness of simulation models appropriately.

VI. EXAMPLES AND RESULTS

A. Simulation Scenario

We set up a scenario to study the dynamics and simulation of tethered satellite systems, which consists of a sequence of orbit transporting and involves several simulation models. Take s a first order J2 perturbation propagator model as the object to be validated, the model's behavior and data sets obtained from flying-around phases are focused. Based on the simulation data and real orbit data, the validation approach is applied to analyze the correctness of the simulation model.

B. Results and Analysis

The result is a set of tuples consisting of velocity and position of the satellite in 3 directions of x, y z, (Vx, Vy, Vz, x, y, z). The observed data set (D_{val}) is chosen from T_0: 796328.422, which consists of 850 records.

Thus, K= 800, According to general experience and the ranges of coefficients a and b, we select 0.8 and 4 as a and b respectively. The size of simulation data set D_{sim} which would be used to calculate the error range is determined as:

$$f(K) = aK + b = 0.8 * 850 + 4 = 684$$

$$N = K + f(K) = 1534$$

The simulation data set is chosen as specified. Following the steps in Section V, the error sequences are calculated. Taking one sample point every 100 data points, the tendencies of the sequences are depicted in Figure II. After the sequences are sorted, we can get the minimum and maximum values of each sequence, which is also the error scope of the simulation model we wanted. The error scopes are then used as an effective estimate of the correctness of simulation models.

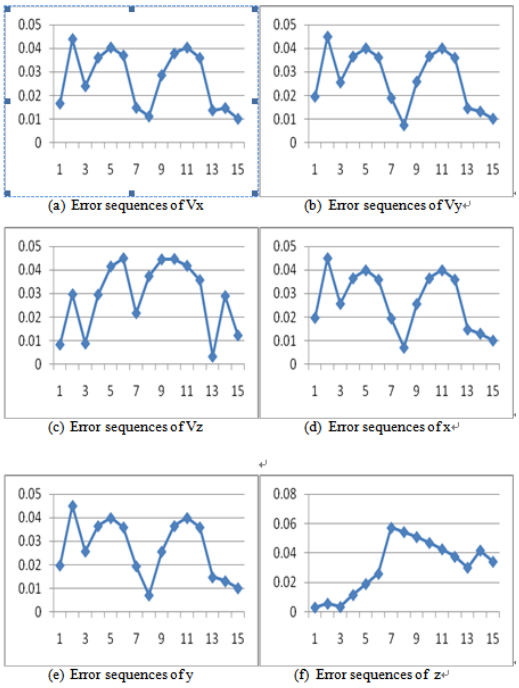

FIGURE II. ERROR SEQUENCES

TABLE I. ERROR SCOPES

	E-Vx	E-Vy	E-Vz	E-x	E-y	E-z
Min	0.010147 79	0.007506	0.003185	0.007136	0.007151	0.003 149
Max	0.085694 16	0.084143	0.086316	0.08404	0.0841	0.057 287

Based on the result, it shows that the error scopes can be described as [0.01,0.085], [0.007,0.084], [0.003,0.086], [0.007,0.084], [0.007,0.084], and [0.003,0,057]. The highest error is 0.086, as such; the correctness of the model is 91.4%, which absolutely meet the common requirement of correctness of simulation models of 85%.

REFERENCES

[1] R E Nance, R G Sargent, Perspectives on the evolution of simulation. Operations Research, 2002, 50(1):161-172.

[2] Li Hui, Gu Xuemai, Design of Multilayered Satellite Communication Networks and Geometrical Analysis of ISLs. Journal of Astronautics, 2005.26(B10):59-64,69.

[3] Lin Qi, Xiong Zhang, Li Zhi, Semantic modeling in satellite network simulation. 2008 Asia Simulation Conference - 7th International

Conference on System Simulation and Scientific Computing, ICSC, 2008:877-881.

[4] Kohavi, Ron, A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (San Mateo, CA: Morgan Kaufmann), 2010, 2 (12): 1137–1143.

[5] Grossman Robert, Seni Giovanni, Elder John, Agarwal Nitin, Liu Huan, Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. Morgan & Claypool. 2010

[6] Sabrina Lyngbye Wendt, Ajenthen Ranjan, Jan Kloppenborg Møller, et al., Cross-Validation of a Glucose-Insulin-Glucagon Pharmacodynamics Model for Simulation Using Data From Patients With Type 1 Diabetes. Journal of Diabetes Science and Technology, 2017, 2(11): 1101-1111.

[7] T. Zhang, Statistical behavior and consistency of classification methods based on convex risk minimization. Annals of Statistics, 2004,32 (1): 56-85.