Precision Testing Methodologies in AI

Michael Yesudas
Watson Customer Engagement
IBM Corporation
Texas, United States of America
myesudas@us.ibm.com

Girish Menon S
Watson Customer Engagement
IBM United Kingdom Limited
London, United Kingdom
girishmenons@ibm.com

Nikita Mariam Thomas
Watson Customer Engagement
IBM India Private Limited
Bangalore, India
thomas13@in.ibm.com

Nithin Mathew
Watson Customer Engagement
IBM India Private Limited
Bangalore, India
nithin10@in.ibm.com

***Abstract*—Modelling and testing of Artificial Intelligence (AI) applications, unlike traditional software development, is probabilistic compared to the deterministic nature of software, thus making test oracles ineffective and AI testing harder. Unfamiliarity with the input data and lack of clarity with baseline accuracy often lead the black box tester to wrong assumptions about the model performance. This paper reviews the current methodologies used in AI testing and proposes a method to validate the adeptness of the machine learning model under test. The paper details how this validation method could be used to iteratively sample the input data and evaluate the convergence of the loss function and establish the model's performance. After sufficient tests, the test engineer can identify how well the algorithm models the existing dataset. Furthermore, the paper details the steps a test engineer can take, to establish the drawbacks of the model.**

Keywords*—*Precision AI testing, Model performance testing, AI Blackbox testing

I. Introduction

The primary difference between Artificial Intelligence (AI) systems and traditional software systems is the predictability of output. Validation of AI systems thus becomes hard where the test engineer cannot create a valid set of output to compare against, for a specific set of input values. The quality assurance of AI-driven systems thus poses a unique problem in software engineering where the behaviour of the learning system changes over time. Machine Learning (ML) systems fall into a category of 'non-testable programs', as described by Weyuker [1] in his paper as "Programs which were written in order to determine the answer in the first place. There would be no need to write such programs if the correct answer were known". Christian Murphy and Gail Kaiser [2] states that subtle errors in non-testable programs are hard to identify. They continue to note that while there are methods that apply machine learning techniques to software engineering and testing, very few methodologies apply software testing techniques in machine learning applications.

Orange [3] and WEKA [4] are examples of two of the earlier frameworks that were helping ML validation. The focus of these tools is on comparing the quality of the results from repositories of result data sets, rather than correctness or performance of the AI model. Thus, a model may predict very well, but it may still have defects. Validation of intrusion detection systems [5] [6], intrusion tolerant systems [7], and other security systems [8], use quantitative measurements like overhead, false alarm rates and ability to detect zero-day attacks. These methodologies although help in quantifying the results have not proven to help deliver precision testing to minimise defects. This paper discusses the newer validation methodologies like Metamorphic Testing [9], Fuzzing and Dual coding, but none of these methods provides a way to identify defects in the quality of input data or feature selection.

The test engineer's role in AI testing is more demanding, as the skills required are more than a typical black-box tester to validate an AI model. The tester should be skilled enough to identify and interpret the boundaries or tolerances of the target system's output. While testing the boundary of output values for accuracy, the tester should know the influence of the input values and the extent of control it has got on the result. Tester verifies an acceptance criterion (typically requiring the stakeholder to define tolerance) for an ML solution, where the training data play a crucial role. Thus, it is critical for the tester to have methodologies and tools that could help identify the correlation between the training data and predicted values. Such tools can be used to identify defects in the quality, distribution and suitability of the training data, that can validate the performance of a model with precision.

In this paper, we discuss the significance of precision testing in AI and the inherent gaps in model validation, when using existing methodologies. A mathematical representation to demonstrate the correlation of the output data to the training dataset is derived. From the distribution of the mean-squared error (*MSE*) values, the tester can identify defects in model training.

II. Background

One of the differentiators of Machine Learning and Deep Learning (DL) is the size and variety of real-time input data that is available for DL. Machine Learning systems that are typically used to derive insights from commercial software back-ends do not have this luxury.

The two main phases in the development of a machine learning model are training and testing. Initially, the data is pre-processed to perform feature engineering. Feature engineering is a process of choosing the right features for the model. During feature engineering, many data sets could be ignored based on false correlations. Based on the type of problem (classification, regression or clustering) the developer chooses the right algorithm for training.

输入数据不熟悉与基线准确度的不明确导致测试人员对模型性能的错误假设。
收敛
貌似只能作用在数值型的AI系统中
精细的
已有的工作
零时差攻击
虚警概率
提高了方法和工具可后面并没有提工具得事情
第一段介绍了AI测试中存在三个问题：1，测试语气难以获得；2，精细的故障难以排查；3，很多传统的方法不用直接运用到AI测试中。
估计值与真值差的平方的均值

Generally, a small percentage of the input data set is kept aside for testing the model. Training the model involves learning the patterns of the input features and their relationship to the target variable.

Once the model has learned all the patterns, it is tested on unseen data, and the model eventually attains a baseline accuracy. The developer determines the initial accuracy of the model by comparing the predicted results against the actual results. At this point, the validation phase starts. Testing using a small portion of data is not the same as testing the model for any arbitrary input for which the actual values are unknown. The tools used to test deterministic systems do not apply to machine learning models as the inputs are arbitrary and there are no test oracles. The outcome of machine learning models is highly volatile where the same function behave differently with different sets of data. The quality of an ML model is dependent on the quality, correlation and distribution of the input data.

Typical defects that a tester identifies with an ML model can be attributed various root causes. In their paper, Dwarakanath et al. [10] describe the reasons as - a) deficient (e.g. biased) training data; b) poor ML architecture used; c) the ML algorithm learnt a wrong function; or d) there is an implementation bug. They state that the current practice is to attribute an incorrect output to the deficiency of the training data and the testers collect more diverse training data to continue testing. The trend seen in the industry is that the tester acts as the test oracle and comes up with a large number of real-time inputs to validate the model. Such test methodologies would be time consuming and expensive. It also depends on the domain knowledge and skills of the tester that determine the quality of the input data used in such tests. Such tests cannot be categorised under precision testing as it is dependent on the skills of the tester. The paper continues to explain the use of metamorphic testing in identifying implementation defects, that are not data specific.

III. Existing AI Testing Methodologies

A survey of the trends and methodologies in the validation of non-testable programs is described by Barr et al. in their Journal [11]. The survey points to the fact that none of these methods is found to be completely solving all aspects of non-testable programs. ML solutions being in the same category requires a hybrid approach to assure test coverage. The initial model performance testing in AI is limited to testing only a small fraction of the data that is obtained from the actual dataset which is used for modelling. The performance (baseline accuracy) of the model on the test data can be deceptive. The arbitrary inputs to the model can influence the model performance. It is possible that the model is an overfit or underfit. Here are some methodologies in practice.

A. *Metamorphic testing*

Metamorphic testing [9] identifies the metamorphic relationship between the features and the target variable. Metamorphic testing is a process of generating consecutive test cases for testing machine learning models. The consecutive test cases are derived based on causation. A test case failure means that the model might not be the preferred one, or there is a bug.

B. *DLFuzz*

DLFuzz [12] is defined as fuzzing testing for deep learning models. It is commonly used for neural networks. It is a process of combining fuzzing testing and deep learning testing to improve the precision of the model steadily. DLFuzz has higher coverage of neurons, and it follows a specific strategy to activate all the neurons in a deep learning model.

C. *Dual coding*

Dual coding [13] [14] is a testing methodology in which the data is trained on multiple algorithms like Random forest, logistic regression or support vector machine. Based on the performance of each algorithm, the best model is chosen. The precision of the best model can be tested by comparing the predictions against the other models. If there is variation in predictions among the best model and the alternate model, a tester can report a defect in the model.

From the existing literature on machine learning models, the test methodologies stated above does not recognise the importance of correlating the input and output data for validating a model with precision. While researches on the topic of machine learning repeatedly state that the output behaviour of an ML model is linked to the input, we have not come across any methodologies that could be used to validate the model performance.

IV. Proposal

We derive a mathematical representation of the mean-squared error (*MSE*), that can indicate how the input data model the algorithm. This help AI testers validate a machine learning model. Our findings on ML models and identifying the correlation between input and output data to predict issues with the model performance are the basis of this proposal.

Let us consider the developer had taken 80% of the data for training and 20% of the data for testing to obtain a baseline accuracy. Testing a small portion of data does not give the actual picture regarding the model. The value of the loss function minimised on the 20% of the data that was taken for testing the model might be very different from the value of the loss function obtained on transforming an arbitrarily chosen fraction of the data. The difference might be because the algorithm has not modelled the dataset effectively. As described before, this can be due to a deficiency in the training data (e.g., a correlation between features and target variable, distribution of data and bias in the data) causing overfit or underfit of the model, infeasible architecture or an implementation bug [10].

The AI tester can iteratively sample the input dataset and calculate the value of the loss function for each of these samples to identify if the value of the loss function converges to the actual value of the loss function obtained while training the model. Loss function defines an objective against which the performance of the model can be

这两句话之间的逻辑是什么？还有第一句话那样说有啥用？
不稳定性
首选的
不同的数据集获得的损失函数的值不同。
采样的数据集的个数？
时间开销？当程序的规模较大的时候，需要的时间回很多的

对数损失

measured. The setting of weight parameters learned by the model is determined by minimising a chosen loss function.

均方差

Appropriate loss function should be chosen as per the model, i.e., mean-squared error if it is a regression problem, and log-loss if it is a classification problem. For this paper, we consider a regression model and hence mean-squared error (*MSE*) as our loss function which is expressed on a given sample of size "m" as follows,

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(Y_i - Y_i^P)^2$$

一些公式看起来有点奇怪，感觉是图片

Here 'Y_i' is the actual or observed value of the predicted variable, 'Y_i^P' is the predicted value of that variable, and 'm' is the number of records in the sample.

方差

Variance is the squared deviation of a random variable from its mean which is the measure of the deviation of a set of numbers or variables from their average value. Variance (V) of a variable in a given dataset of size 'n' can be expressed as,

$$V(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

Here '$\bar{X}$' is the mean value of X, 'X_i' is the 'i'th value of X and 'n' is the number of records in the dataset. According to the Bias-Variance decomposition [15], we know that,

$$MSE = V(Y) + Noise + Bias^2$$

这个写出来有啥用？

偏差

Here $V(Y)$ is the variance in Y and noise also known as irreducible error represents the observed noise in the dataset, i.e., the randomness in the dataset. An irreducible error is an error that cannot be reduced by improving the model as it depends only on the distribution of data points in the models. Bias is the difference between the average prediction of the model and the correct value which it is trying to predict, which indicates how well the algorithm models the data.

怎么获得最优的平衡

The right balance must be identified between bias and variance to build a good model such that it minimises the total error. An optimal balance of bias and variance will ensure that the model does not underfit or overfit the data.

For a well-built model, the sum of the noise to the squared value of bias is almost a constant for a given dataset. The value of bias is almost *0* for an unbiased model and noise depends only on the distribution of the data. Therefore, we can postulate that if we take random samples from the dataset and plot the values of *MSE* for each of these samples, it will cluster around the value of the variance of the target variable for a well-built model.

The tester can use the distribution of *MSE* as an approach to identify or hypothesise how precisely the algorithm models the dataset. This entire process can be automated, to iteratively sample the dataset and calculate the *MSE* for each of these samples and see if the value is clustered in the close neighbourhood of the variance of the target variable. If the points are clustered far from the variance point, then it indicates that the model overfits or underfits and does not perform well on data other than the training data. This automated tool can be used to identify faults in the model.

需不需要重新训练模型！！如果需要的话，时间开销考不考虑？

有没有一个评判标准？

模型已经用已有的数据集中的一部分训练过了，另外一部分用来测试，那么如果选取多个测试集的话，肯定会选到训练模型的数据啊

A. *Mathematical Description*

Let 'S' be the dataset with 'l' data points and 'M' be the model trained on the dataset. We can calculate the variance of the target variable in the dataset as follows,

$$V(y) = \frac{1}{l}\sum_{i=1}^{l}(y_i - \bar{y})^2$$

Where y_i is the 'i'th value of the target variable y, 'l' is the number of records in the dataset, i.e. the number of elements in the target variable set and $\bar{y}$ is the mean value of the target variable. The dataset 'S' is split iteratively into 'n' parts such that

数学表达不行啊

$$S = \{X_1{:}Y_1, X_2{:}Y_2, X_3{:}Y_3, X_4{:}Y_4, X_5{:}Y_{5,\ldots}\ X_n{:}Y_n\},$$

where $X_i = \{X_{i1}, X_{i2}, X_{i3}, X_{i4,\ldots}\ X_{im} : m$ is the number of feature vectors in each set$\}$ is the set of feature vectors and $Y_i = \{Y_{i1}, Y_{i2}, Y_{i3}, Y_{i4,\ldots}\ Y_{im} : m$ is the number of target variables in each set$\}$ is the set of labels respective to each feature vector in the set.

$$M(X_i) = Y_i^P$$

公式没有标号

i.e., $Y_i^P = \{Y_{i1}{}^P, Y_{i2}{}^P, Y_{i3}{}^P, \ldots\ Y_{im}{}^P : m$ is the number of target variables in each set$\}$ is the value obtained by transforming X_i on the model 'M'.

The mean-squared error (*MSE*) (the loss function), for each sample of the dataset, is expressed as follows,

$$MSE_i = \frac{1}{m}\sum_{j=1}^{m}(Y_{ij} - Y_{ij}^P)^2$$

Where 'MSE_i' is the value of the Loss function when X_i is transformed on the model and 'm' is the number of records in set X_i.

For each feature vector set in the set $S = \{X_1{:}Y_1, X_2{:}Y_2, X_3{:}Y_3, X_4{:}Y_4, X_5{:}Y_{5,\ldots}\ X_n{:}Y_n\}$, calculate loss function on each element in the set to obtain the set of loss function as follows,

$$MSE = \{MSE_1, MSE_2, MSE_{3,\ldots}\ MSE_n\}$$

Suppose we have chosen 35 random samples and plot the values of the *MSE* in a graph where X-axis denotes the iteration and Y-axis denotes the value of the loss function (*MSE*) as shown in Fig. 1.

空的太多了把

也没有开发工具

都没有做实验。。。假设的值分析个啥？

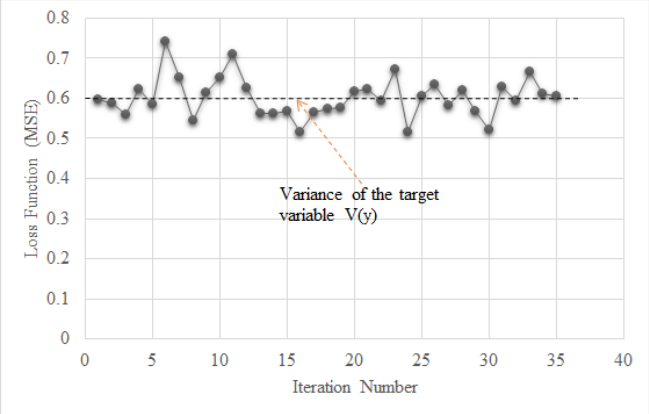

Fig. 1. The plot of loss function for each iteration

Here, the mean of the values of the loss function converges close to the value of the variance. Indicating that these points are clustered around the value of the variance of the target variable. From the graph (*Figure. 1)* it can be observed that the value of the loss function is almost uniform and converges to the variance of the target variable. Hence, we know that the model has been trained well.

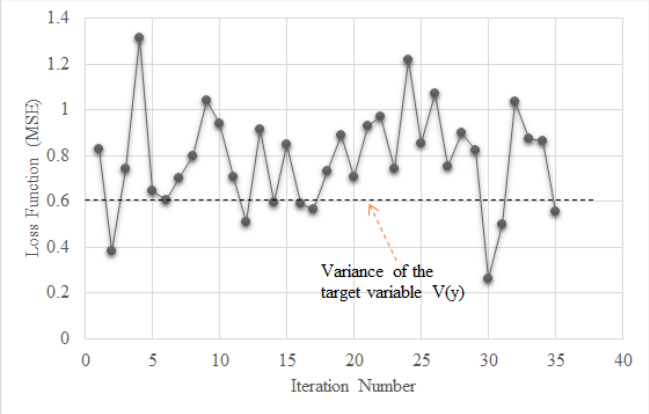

Fig. 2. Loss function does not converge

In the above graph (*Figure 2*), unlike in the earlier graph, we see that the value of the loss function varies significantly for each iteration and does not converge to the value of the variance of the target variable of the model. These points are clustered away from the variance and hence, indicates that the model might not have been trained well on the given dataset.

V. Conclusion

Because of the complex nature, it takes a hybrid approach with many methodologies to verify an AI solution. Engineers who have the appropriate domain skills to ingest vast amounts of information and analyse it for insights are in demand and significantly increased over the years. Although the area of AI testing is in the boom, the unavailability of precision testing methods and tools is a real problem. Our method is intended for AI test engineers to identify and isolate issues with insufficient or insignificant input data, thus providing the data scientist to improve the model performance. The method proposed by this paper can be used by the tester to hypothesise how well the algorithm models the dataset using the distribution of the value of loss function observed for various samples of the dataset. This entire process can be automated, and a tool can be developed for an AI tester to precision test the model using this approach.

VI. References

[1] E.J. Weyuker, “On Testing Non-Testable Programs”, Computer Journal vol.25 no.4, November 1982, pp.465-470.

[2] C. Murphy and G. Kaiser, “Metamorphic runtime checking of nontestable programs”, Technical Report cucs-012-09, Dept. of Computer Science, Columbia University, 2009.

[3] Demsar, J., Zupan, B., and Leban, G. Orange: From experimental machine learning to interactive data mining. [www.ailab.si/orange], Faculty of Computer and Information Science, University of Ljubljana.

[4] Witten, I. H. and Frank, E. 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition. Morgan Kaufmann.

[5] Mell, P., Hu, V., Lippmann, R., Haines, J., and Zissman, M. 2003. An overview of issues in testing intrusion detection systems. Tech. Rep. Tech. Report NIST IR 7007, National Institute of Standard and Technology

[6] Nicholas, J. P., Zhang, K., Chung, M., Mukherjee, B., and Olsson, R. A. 1996. A methodology for testing intrusion detection systems. IEEE Transactions on Software Engineering 22, 10, 719–729.

[7] Madan, B., Go˘seva-Popstojanova, K., Vaidyanathan, K., and Trivedi, K. S. 2004. A method for modeling and quantifying the security attributes of intrusion tolerant systems. Performance Evaluation Journal 56, 1-4, 167–186.

[8] Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., and Vigna, G. 2008. Saner: Composing static and dynamic analysis to validate sanitization in web applications. In Proc. of the 2008 IEEE Symposium on Security and Privacy. 387–401.

[9] T. Y. Chen, S. C. Cheung, and S. Yiu. Metamorphic testing: a new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, 1998.

[10] Dwarakanath, Anurag, Manish Ahuja, Samarth Sikand, Raghotham M. Rao, R. P. Bose, Neville Dubash, and Sanjay Podder. "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing." In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 118-128. ACM, 2018.

[11] Barr, Earl T., Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "The oracle problem in software testing: A survey." IEEE transactions on software engineering 41, no. 5 (2015): 507-525.

[12] Guo, Jianmin, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. "DLFuzz: differential fuzzing testing of deep learning systems." In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 739-743. ACM, 2018.

[13] Clark, James M., and Allan Paivio. "Dual coding theory and education." Educational psychology review 3, no. 3 (1991): 149-210.

[14] Murphy, Christian, and Gail E. Kaiser. "Improving the dependability of machine learning applications." (2008).

[15] Maksym Zavershynskyi, “MSE and Bias-Variance decomposition”. [Online]. Available: https://towardsdatascience.com/mse-and-bias-variance-decomposition-77449dd2ff55