Programming-by-Example for Data Transformation to Improve Machine Learning Performance

Minori Narita
University of Tokyo
Tokyo, Japan
mnarita1207@gmail.com

Takeo Igarashi
University of Tokyo
Tokyo, Japan
takeo@acm.org

ABSTRACT

In this study, we propose a programming-by-example (PBE)-based data transformation method for feature engineering in machine learning. Data transformation by PBE is not new. However, we utilized the one proposed herein to improve the performance of machine learning in synthesizing a transformation rule from examples. Herein, the system first generates candidate rules, and then chooses the rule that achieves the highest performance in a target machine learning task. We tested this system with the Titanic dataset, and the result shows that the proposed method can avoid worst-case performance compared to the original PBE method.

CCS CONCEPTS

• **Human-centered computing → User interface programming**.

KEYWORDS

data transformation, programming-by-example, feature generation, program synthesis

ACM Reference Format:
Minori Narita and Takeo Igarashi. 2018. Programming-by-Example for Data Transformation to Improve Machine Learning Performance. In *IUI '19: ACM 24th International Conference on Intelligent User Interfaces, March 17–20, 2019, Los Angeles, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Data transformation in feature engineering is crucial for improving the performance of machine learning [2–4]. However, feature engineering, which mainly involves manual and tedious tasks with many trial and errors [4], is difficult and time consuming especially for non-programmers. Domain experts often know how data should look, but they do not have the skills to write a code that will achieve the desired result.

Programming-by-Example (PBE) can be useful in enabling non-programmers to process data with ease. For example, FLASHFILL [1] synthesizes string transformation program from input-output examples provided by users. Although, in general, these techniques are useful in data transformation; however, the performance of machine learning in the process have not been studied by the existing methods.

We propose a PBE-based data transformation method, which is specifically designed for feature engineering in machine learning. The main idea is to consider the prediction accuracy of machine learning in terms of synthesizing transformation rules. We evaluated the proposed system with the Titanic dataset obtained from Kaggle[1]. Our system successfully generated feature transformation rules that improve classification accuracy from only a limited number of input-output examples. Compared to the original PBE method, where one has to explicitly prioritize the rules , our method has an advantage of automatically selecting rules that avoid worst-case performances.

1.1 Method

We focused on transforming a single feature with regard to supervised learning during a classification task. Suppose we have a dataset comprising many data elements, which consist of input features and an output class. In the training phase, the system learns mapping from input features to an output class. In the classification phase, the system takes input features and returns a predicted output class. The system learns mapping using training data, and we evaluate its performance using test data.

Furthermore, we focused on transforming a feature, which is given as a text string. The user provides several examples of the desired transformation, where each example is a pair of an input string and an output string. Our algorithm, which internally constructs a string program using a domain specific language, is based on the one presented in FLASHFILL [1]. It first generates multiple candidate programs and selects the simplest one. We modified this ranking process to consider the classification accuracy, and we processed training and classification of a target task to evaluate the performance of each candidate program and selected the best among them.

2 EVALUATION

We tested the proposed synthesis algorithm with the Titanic dataset and applied it to a feature named "Cabin." Most values in this feature have an alphabet followed by a number (ex. "C231"). While some have more than one room number (ex. "A21 A22"), some data are found to be missing. To make this feature applicable, we first replaced missing data with some alternative values. Then, we categorized the data based on its alphabet, e.g., "C231" to "C." We also needed to handle outlier cases. Our approach allows the system to

[1] https://www.kaggle.com/francksylla/titanic-machine-learning-from-disaster

IUI '19, March 17–20, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

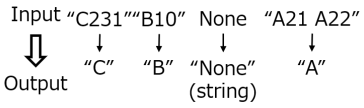

Figure 1: Input-output examples fed to the PBE systems.

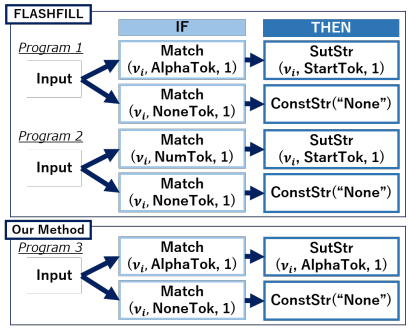

Figure 2: Transformation rules obtained using the original PBE and our method. "Match" is a predicate. "SubStr" and "ConstStr" are atomic expressions described in [1].

synthesize such transformations only from examples in a way to improve prediction accuracy.

Out of the nine original features in the Titanic dataset[2], we used five features in addition to "Cabin" to predict a passenger's survival ("Survived"). We removed the other 3 features (Fare, Pclass, and SibSp) because they are highly correlated with "Cabin" and thus obscure the effect of its transformation. We randomly split the entire data (894) into training data and test data by 9:1. The training data is used for training as well as selecting a transformation rule. We split the training data into five, and we used 4 among them for training and 1 for testing, to get predicted accuracy for each possible transformation. Then we average five possible combinations and chose the rule that provides the best prediction accuracy. Hence, we used only the training data in the whole synthesis process. Also, we used the random forest for the classification algorithm.

We compared the result obtained by applying the proposed PBE method to the "Cabin" feature and that obtained by employing the original PBE method [1]. We also consider the baseline (without "Cabin" feature). We configured the PBE systems so that when the synthesized program fails, they return the most frequently appearing class in the successful transformation results. As for the original PBE, we tested two variations of the method presented in [1], prioritizing the rules differently. Original PBE1 prefers numbers to alphabets, while Original PBE2 prefers the opposite.

3 RESULTS

Figure 1 shows the examples provided to the PBE system, while Figure 2 shows examples of programs synthesized by the original PBE method and the proposed method. Figure 3 shows examples of the transformation results. Program 2 outputs "None" (String) for input "T" because the program cannot handle a single alphabet,

[2]Age,Cabin,Embarked,Fare,Parch,Pclass,Sex,SibSp,Title.

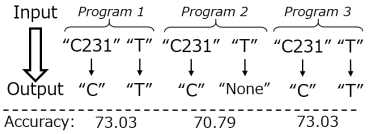

Figure 3: The outputs generated by programs shown in Figure. 2.

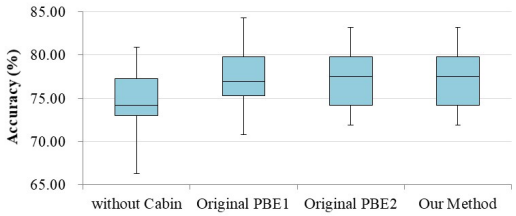

Figure 4: Distribution of prediction accuracy with random selection of training (test) data.

and thus, replaced with the most frequent output in the successful transformations ("None").

Figure 4 shows the distribution of the prediction accuracy obtained by processing training and classification 30 times by randomly changing the selection of training and test data. This shows that the performance of the three PBE methods is similar. Original PBE1 is less stable (larger variance) than Original PBE2. Selection of PBE1 or PBE2 is rather arbitrary (one has to decide which rule to give priority), but our method automatically chooses more stable rules, thereby avoiding worst-cases.

4 LIMITATION AND FUTURE WORK

The result obtained is promising but is arguably just the starting point, indicating that further research is necessary. First, we shall test the proposed method with datasets other than Titanic, especially, with a much larger dataset. Second, we shall improve our method to support numerical features. Third, we require to support transformations combining multiple features. We also plan to have the synthesized program in a way that users can modify it. This is especially important because users cannot fully trust black box behavior and prefer transparency. Finally, we need to run user study to ascertain the usability and effectiveness of the proposed method.

5 ACKNOWLEDGMENTS

This work was supported by JST CREST JPMJCR17A1.

REFERENCES

[1] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (2012).
[2] Zhongjun Jin, Michael R. Anderson, Michael Cafarella, and H. V. Jagadish. 2017. Foofah: A programming-by-example system for synthesizing data transformation programs. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1607–1610.
[3] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *DSAA*.
[4] David Kofoed Wind. 2014. *Concepts in predictive machine learning*. Master's thesis. Technical Univ. of Denmark, Denmark.