Metamorphic Exploration of an Unsupervised Clustering Program

Sen Yang, Dave Towey*
School of Computer Science,
University of Nottingham Ningbo China,
Ningbo, Zhejiang 315100,
People's Republic of China
Email: {zy18733, dave.towey}@nottingham.edu.cn

Zhi Quan Zhou
Institute of Cybersecurity and Cryptology,
School of Computing and Information Technology,
University of Wollongong,
Wollongong, NSW 2522, Australia
Email: zhiquan@uow.edu.au

***Abstract*—Machine learning has been becoming increasingly popular and widely-used in various industry domains. The presence of the oracle problem, however, makes it difficult to ensure the quality of this kind of software. Furthermore, the popularity of machine learning and its application has attracted many users who are not experts in this field. In this paper, we report on using a recently introduced method called *metamorphic exploration* where we proposed a set of hypothesized metamorphic relations for an unsupervised clustering program, Weka, to enhance understanding of the system and its better use.**

***Index Terms*—metamorphic testing, metamorphic exploration, machine learning, clustering, K-means, unsupervised machine learning.**

I. Introduction

With the growing recognition of the application of machine learning (ML) in natural language processing, image recognition, computer vision and many other domains, ML has been increasingly looked to for potential solutions to scientific and engineering problems [1]. Mitchell formally defined ML as: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E" [2, p. 2]. ML enables computers to make decisions and act based on its learned experiences, and can gradually improve through successive encounters with new data. ML algorithms are often categorized as supervised or unsupervised. Supervised ML algorithms aim to construct a mathematical model enabling prediction of expected outputs based on the class labels and features of training data [3]. Unsupervised ML algorithms, in contrast, usually take data without labels or classifications, and attempt to find structure in the data [4].

Clustering is a general unsupervised ML task, and is also a common technique in statistical data analysis in many fields. An operational definition of clustering, according to Jain, is: "Given a representation of n objects, find K groups based on a measure of similarity such that the similarities between objects in the same group are high while the similarities between objects in different groups are low" [5, p. 652]. Clustering algorithms are often categorized as partitional or hierarchical, with the K-means algorithm being one of the most popular and simplest partitional algorithms. Since its first appearance [6] in 1967, many improved implementations and optimized versions based on the standard K-means algorithm have appeared [7]–[9]. However, in contrast to supervised ML algorithms, evaluating the quality of clustering algorithms can be especially challenging: There is no label for validation and users' subjective expectations can strongly impact on the evaluation of clustering performance.

In traditional software testing, verification techniques usually involve setting up inputs and examining the corresponding outputs through execution of the software under test (SUT). If an output is correct, then the test passes. The mechanism against which testers can decide the correctness of test case executions is known as a test oracle [10]. When the test oracle is unavailable, or is available but cannot be practically used, the tester is said to face the test oracle problem (or just oracle problem) [11]. Because of the difficulty finding and using a test oracle for clustering programs (for example to evaluate the appropriateness of the groupings of data points given by the SUT), clustering programs may also face the oracle problem. Although the oracle problem has rendered many traditional testing approaches ineffective, one approach that has been identified that very effectively alleviates this problem is metamorphic testing (MT) [12], [13]. MT alleviates the oracle problem by examining relations among multiple executions of the SUT. These relations are called metamorphic relations (MRs) [14]. MT was originally developed as a verification technique, with MRs being necessary properties of the SUT. MT has been used by both professional software developers and end-user programmers [15], and has been developed into a framework covering verification, validation, and other types of software quality assessment [16], with MRs not only defined by developers, but also by the users. More recently, MRs have been applied to enhance system understanding and use [17], where MRs need not be necessary properties for software correctness, but can instead be properties hypothesized by the users, who can use such MRs to explore the software system, enhancing their understanding of the system, and hence using it in a better way. This approach is called *metamorphic exploration*. In this work, we conduct a small-scale case study

*Dave Towey is the corresponding author.

978-1-7281-2235-9/19/$31.00 ©2019 IEEE
DOI 10.1109/MET.2019.00015

of metamorphic exploration using a clustering program, a type of unsupervised machine learning program.

The rest of this paper is laid out as follows: Section II introduces some of the background to this paper, including explaining the basic K-means clustering algorithm, and Weka, the SUT. Our hypothesized metamorphic relations are explained in Section III. Section IV presents our study, including the experimental setup and results. Section V analyzes and discusses the results of the study. In Section VI, we discuss some potential future work. Finally, Section VII concludes the paper.

II. Background

A. K-means Clustering Algorithm

The K-means algorithm [5] attempts to (iteratively) partition a dataset into K distinct non-overlapping subgroups (clusters) where each data point belongs to one and only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different as possible, which means maximizing the distance between clusters. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at a minimum. The less variation within a cluster, the more homogeneous are the data points within that cluster.

The basic steps of the standard K-means algorithm are as follows:

1) Choose the number of clusters (parameter K).
2) Initialize centroids by first shuffling the dataset and then randomly selecting K distinct data points as the centroids.
3) If a terminating criterion is met (e.g. there has been no change in the assignment of the data points to clusters for two consecutive epochs[1]), then stop, otherwise keep iterating. The iterated steps are:
    a) Compute the sum of the squared distance between all points and centroids.
    b) Assign each data point to the corresponding closest cluster (centroid).
    c) Compute the centroids for the clusters by taking the average of the all data points belonging to each cluster.

Fig. 1 [18] illustrates how the K-means algorithm basically works in 2D. As the figure shows, the best clustering centroids are found by iteratively assigning data points to the current cluster centroids and finding new centroids based on the data points assigned in the previous step.

[1]An epoch is one complete presentation of the dataset to be learned to a learning machine.

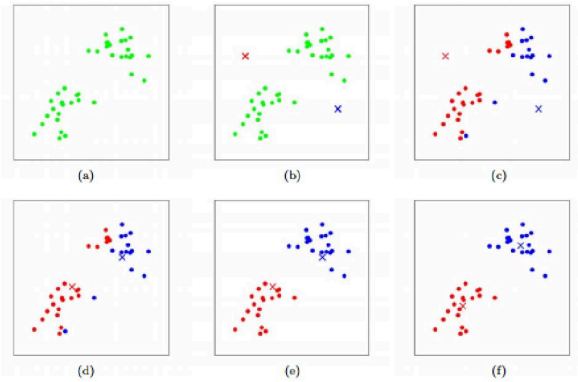

Fig. 1. Basic steps of K-means algorithm ($k = 2$) from [18]. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations.

B. Weka

In our study, we selected Weka [19] as our SUT. Weka is one of the most popular data science platforms for ML and data mining, with an easily-operated graphical user interface, and built-in implementations of ML algorithms for classification, regression, clustering, and association rules mining.

III. Metamorphic Relations

Based on the six fundamental metamorphic relations proposed by Murphy et al. [20] (additive, multiplicative, permutative, invertive, inclusive, and exclusive), Xie et al. [21] proposed a set of metamorphic relations for Weka 3.5.7 [19]. Although their work was for classification in supervised ML, we were inspired to apply their idea and approach to unsupervised ML. In keeping with recent work related to metamorphic exploration [17], because we do not assume that the MRs identified in this process have been rigorously evaluated and confirmed to strictly be MRs [14], in this instance we refer to them as *hypothesized* metamorphic relations (HMRs). By making reference to Xie et al. [21] and Jarman et al. [22], we list the following HMRs for the K-means clustering algorithm:

- HMR1: Translation of 2D points along a line parallel to the x- or y-axis should not have an impact on the clustering results.
- HMR2: Adding a duplicate point should not have an impact on the clustering results.
- HMR3: Moving an existing point towards the cluster center should not have an impact on the clustering results.
- HMR4: Adding a dimension in which all points have an equal value should not have an impact on the clustering results.
- HMR5: Swapping the x- and y-coordinates of each and every point (which is essentially conducting a geometric transformation on the existing points) should not have an impact on the clustering results.
- HMR6: Using the same set of points while changing their input order to the SUT should not have an impact on the clustering results.

- HMR7: In 2D, flipping the data points along one (x- or y-) axis should not have an impact on the clustering results.

It should be noted that although the above hypothesized MRs are explained in the context of a 2D space, the ideas can be readily extended into higher dimensional spaces.

IV. Experimental Study

A. Experimental Setup

Considering the convenience of operation and obtaining of source test cases for the experiment, we decided to use Weka 3.8.3 [19] (latest stable version) to conduct metamorphic testing/exploration. We chose iris.2D.arff as the source test data sample, which is from the data provided by Weka, and is a popular dataset for data mining [23]. The iris.2D.arff file has 150 instances, each with two numerical attributes, the petal length and petal width. It contains three plant classes (Iris setosa, Iris versicolor and Iris virginica). Our decision to use 2D data was related to the ease of finding the cluster center and manipulating the data. For every input source test case, a corresponding follow-up test case was generated using the hypothesized MRs (Section III). Both test cases were executed, and the resulting outputs examined to confirm whether or not the hypothesized MRs were violated.

The parameter configuration used to set up the experiments was: "weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 5 -A "weka.core. EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10" [2]. This configuration means implementation of the simple K-means algorithm, using a random initialization method, Euclidean distance [24] for distance measurements and a maximum of 500 iterations. Most of the parameters were set as the default. The most important parameters in the configuration were: "-N" (the parameter K, as the initial number of clusters set); and "-S" (the seed used to initialise the psuedo-random choice of initial K cluster centriods — arbitrarily chosen to be 10). Although the change of seed number will influence the clustering results, the reason why we chose 10 is only to let the program give the same results for each run. We used the Euclidean distance [24] to calculate the distances between points.

The K-means algorithm aims to find centre points that optimize the following cost function, which minimizes the sum of the squared error over all K clusters, (Equation 1) [5]. Let X = $\{x_i\}$, $i = 1, ..., n$ be the set of n d-dimensional points to be clustered into a set of K clusters, C = $\{c_k\}$, $k = 1, ..., K$. Let μ_k be the mean of cluster c_k.

$$E = \sum_{k=1}^{K} \sum_{x(i) \in C(k)} \|x_i - \mu_k\|^2 \quad (1)$$

The value for K used in our study was 5, which was an arbitrary selection for this metamorphic exploration.

B. Experiment Results

The clustering results from running the source test case are shown in Fig. 2. The rest of this section reports on the satisfaction or violation of our seven hypothesized MRs (Section III), based on the examination of the output from the relevant follow-up test cases. Table I summarises the details of clusters, sum of squared error, and violation or satisfaction of the hypothesised MRs.

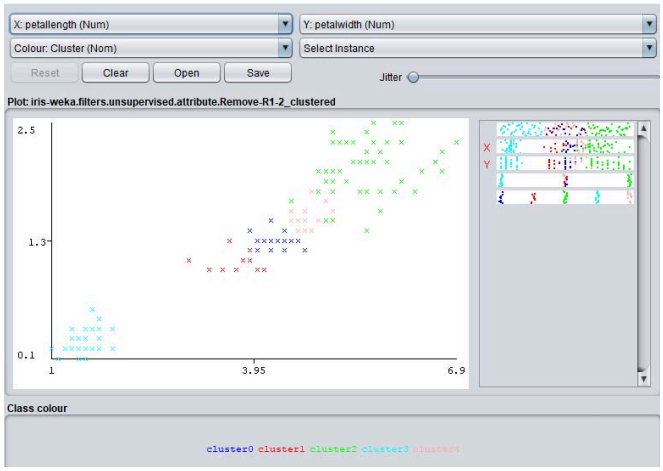

Fig. 2. Clustering results of source test data

1) HMR1: Satisfied: We used the transformation function $f(x) = 2x + 1$ to create follow-up test data. As Fig. 3 and Table I show, the geometric transformation did not impact on the clustering model, with the calculated sum of squared errors also being the same.

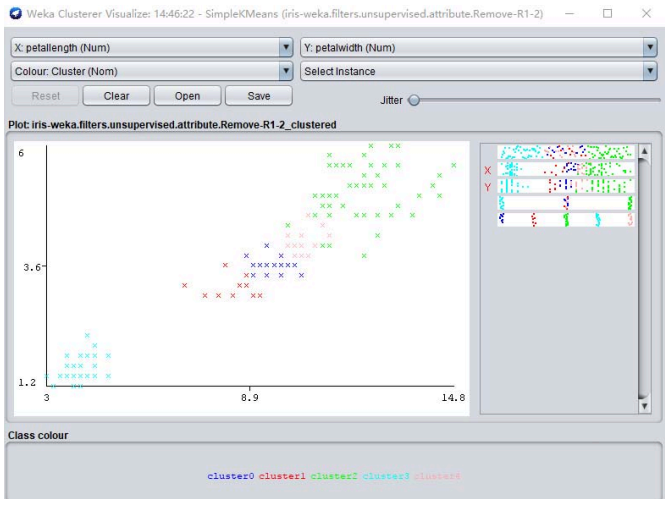

Fig. 3. Test results for HMR1

2) HMR2: Violated: The hypothesized MR2 says that addition of a duplicated point should not impact on the clustering

[2]Extract from Weka command line.

TABLE I
SUMMARY OF CLUSTERING, SUM OF SQUARED ERROR, AND VIOLATION STATUS FOR HMR1–7.

	CI 0	CI 1	CI 2	CI 3	CI 4	Sum of squared error	Violation?
Source case	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35904121104071	
HMR1	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35904121104071	Satisfied
HMR2	14 (9%)	37 (25%)	27 (18%)	50 (33%)	23 (15%)	1.40253305286367	Violated
HMR3	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35831783141108	Satisfied
HMR4	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35904121104071	Satisfied
HMR5	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35904121104071	Satisfied
HMR6	16 (11%)	11 (7%)	50 (33%)	50 (33%)	23 (15%)	1.17581667609006	Violated
HMR7	20 (13%)	12 (8%)	50 (33%)	50 (33%)	18 (12%)	1.35904121104071	Satisfied

[a] CI refers to Clustering Instance.

results. However, after we added a duplication of an original point, the clustering result changed as shown in Fig. 4. This change was also apparent in the details of the clusters and sums of errors (Table I).

Fig. 4. Test results for HMR2

3) HMR3: Satisfied: The hypothesized MR3 states that moving an existing point closer to its cluster center should not change the clustering results. To test this, we examined the five cluster centers and shifted one data point of each cluster slightly closer to its cluster center. As shown in Fig. 5 and Table I, the clustering results did not change, and there is only a small change in the sum of squared errors.

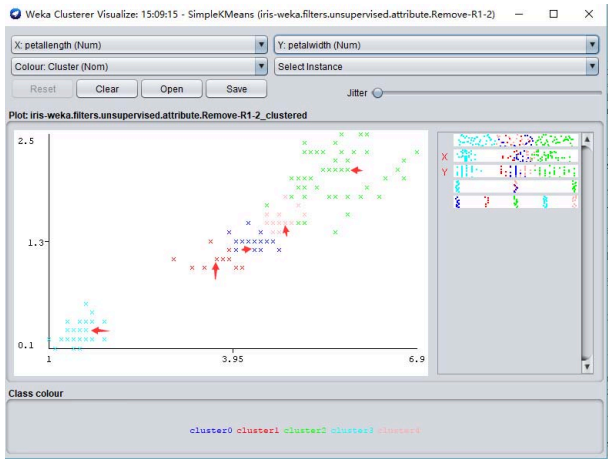

Fig. 5. Test results for HMR3 (the red arrow shows the shifted data points)

4) HMR4: Satisfied: We manually added one new attribute (dimension) which is irrelevant to the existing elements and set the value for all points for this dimension to 1.0. As shown in Fig. 6 and Table I, the clustering results remained the same.

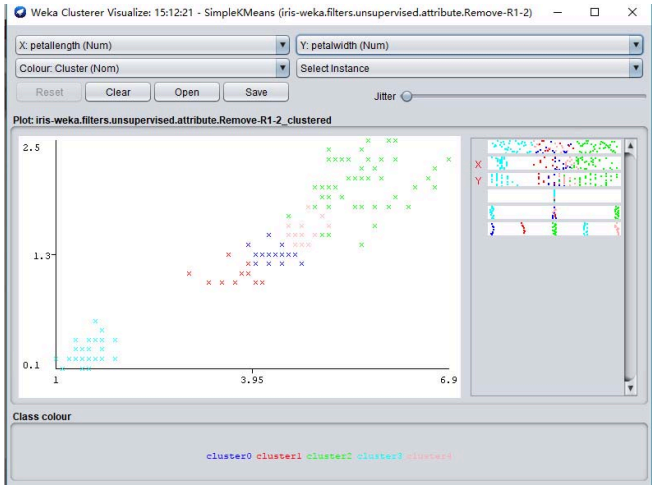

Fig. 6. Test results for HMR4

5) HMR5: Satisfied: The hypothesized MR5 involved a geometric transformation of features — simply switching the position of the two attributes without changing the sequence

of instances ((x, y) became (y, x) for all points). Although the distribution of data points in Fig. 7 appears different to that in Fig. 2, as shown in Table I, the change in the order of attributes did not change the clusters.

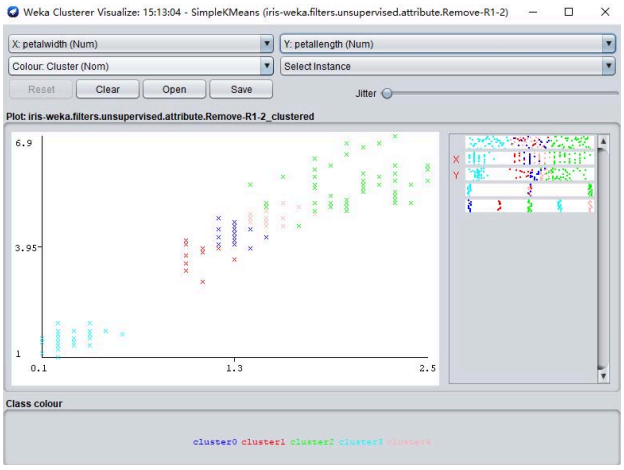

Fig. 7. Test results for HMR5

6) HMR6: Violated: The hypothesized MR6 stated that changing the input order of the data points should not alter the identified clusters. To test this, we simply moved the first 50 data points to the end of the input dataset. As shown in Fig. 8 and Table I, this change in order did impact on the clustering results.

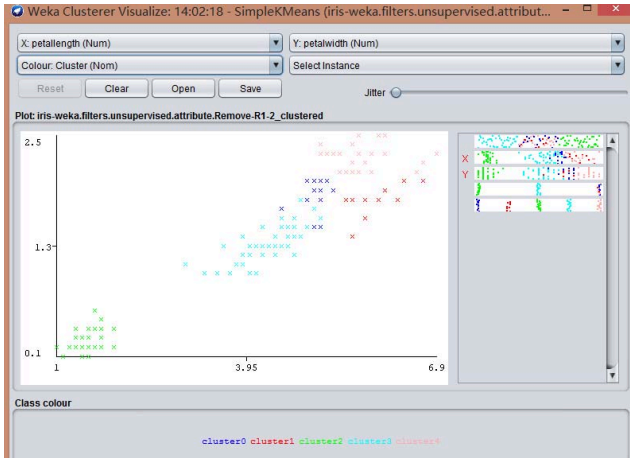

Fig. 8. Test results for HMR6

7) HMR7: Satisfied: The hypothesized MR7 stated that, in a 2D space, flipping the points through one axis should not have an impact on the clustering results. For this, we chose to move through the y-axes: (x,y) became (-x,y). Although the distribution of data points in Fig. 9 appears different to that in Fig. 2, as shown in Table I, the change in the order of attributes did not change the clusters.

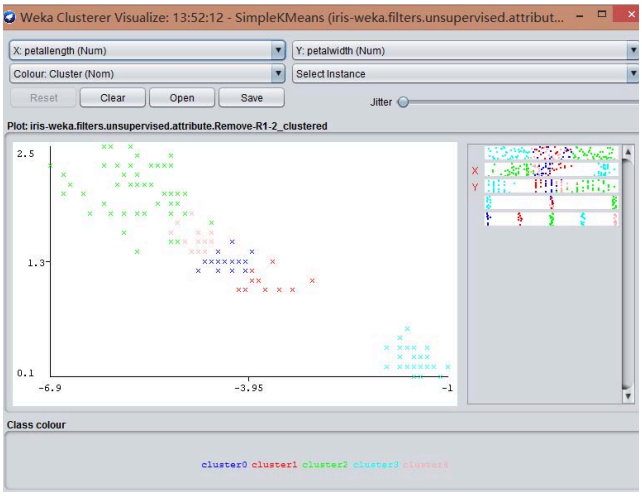

Fig. 9. Test results for HMR7

V. Analysis and Discussion

In summary, in this case of metamorphic exploration, we found that five of our hypothesized MRs were satisfied (HMR1, HMR3, HMR4, HMR5, and HMR7), but two did appear to have been violated (HMR2 and HMR6).

Theoretically, adding a duplication of a data point should not affect the clustering results, which should just simply coincide in the perspective of space. After researching the algorithm more deeply, the cause of this problem was identified as being related to the algorithm's characteristics: Adding a new data point leads to a new round of calculation of the Euclidean distance when re-locating the new cluster centroids, and it will also affect the weight of that point, which may cause a change in the probability of which cluster that point belongs to. This can lead to the shift in the clustering centroids. In summary, this will influence the clustering results. In addition, it will also influence the selection of initial cluster center points because of the change of the data sample order. This helps us to understand that the entry position of data matters, because the randomness of the initial cluster centroid selection will have impact on the clustering results.

The violation of HMR6 appears to have a similar cause: Changing of the data entry order. In consequence, this will effect the clustering results.

The analysis reflects a characteristic of the K-means clustering algorithm itself, and is not a defect in the implementation. In addition, we also tested this part using hierarchical clustering [25], which does not rely on the selection of parameter K. It indicates that adding new duplicate data will not influence the clustering results. This also gives us an idea about how to choose the clustering algorithm when dealing with data containing duplicates: If the user needs to add new data which contains some duplicates to the original dataset, then it is better to choose a hierarchical clustering algorithm, to keep a stable clustering performance.

In summary, it appears that the apparent violations were related to the changed order and relative entry position of

the data. This has a strong connection with the selection of initial cluster centroids, which will directly influence the clustering process and results. In the *.arff* file, each instance occupies one line, and the data is stored in order — from top to bottom. HMR6 reordered the data instances, and HMR2 inserted a new instance into the existing dataset, which not only changed the number of instances, but also changed the relative entry positions. An illustration of this is as follows: For the source input, the input data sequence in the iris.arff file was: instance 1, instance 2, instance 3, ..., instance 150. HMR6 generated a follow-up input which is a permutation of the original input order such as: instance 101, instance 12, instance 43, ... HMR2 inserted a duplicate into the original data file, such as: instance 1, instance 2, instance 2, instance 3, ..., instance 150. Therefore, the follow-up inputs of both HMR6 and HMR2 have changed the relative entry positions of the data, and the SUT appears to be sensitive to the orders and therefore produced different outputs.

Compared with our understanding of the SUT before the experiment, we have gained new knowledge and understanding of the system. It should be noted that this information was not documented in the software's user manual but rather was revealed through metamorphic exploration.

With this new knowledge, we can better use the software. For example, if we want to re-generate an earlier test result, we should keep the original order of the data points when they are entered into the system.

VI. Future Work

This case study used Weka 3.8.3. There are still two main issues that need to be addressed. The source test cases used in this study came directly from Weka's attached dataset, but in real life, the source data are usually random and unpredictable. Use of such a dataset has the potential defect of insufficient coverage and sensitivity to prediction. Consequently, it is essential to include some randomly generated source test inputs. In the future, we plan to also conduct metamorphic exploration on other popular machine learning platforms, including scikit-learn [26]. This should provide further experience of the general applicability of metamorphic relations to help users use the SUT in a better way.

VII. Conclusion

In this paper we have provided a small case study that shows the usefulness of metamorphic exploration for a clustering program that is based on unsupervised machine learning. This approach can not only enhance users' understanding of the target software, but can also lead the the user to better use the algorithm, including choosing suitable data entry formats and solutions.

Acknowledgments

This research was supported in part by a linkage grant of the Australian Research Council (Project ID: LP160101691). It was also partially supported by the UNNC Visiting Scholar Scheme (VSS01), and a UNNC Faculty of Science and Engineering conference grant.

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[3] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[4] E. Alpaydin, *Introduction to Machine Learning*, 3rd ed. The MIT Press, 2014.

[5] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[7] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert systems with applications*, vol. 40, no. 1, pp. 200–210, 2013.

[8] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-means clustering," *Pattern recognition letters*, vol. 25, no. 11, pp. 1293–1302, 2004.

[9] Y.-M. Cheung, "k*-means: A new generalized k-means clustering algorithm," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2883–2893, 2003.

[10] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[11] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?" *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 4–22, 2014.

[12] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.

[13] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing without the need of oracles," *Information and Software Technology*, vol. 45, no. 1, pp. 1–9, 2003.

[14] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[15] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "An effective testing method for end-user programmers," in *ACM SIGSOFT Software Engineering Notes 30 (4), Proceedings of the 1st Workshop on End-User Software Engineering (WEUSE I)*. ACM Press, 2005, pp. 1–5.

[16] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2016.

[17] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, DOI: 10.1109/TSE.2018.2876433.

[18] C. Piech, "K means," http://stanford.edu/ cpiech/cs221/handouts/kmeans.html, accessed January 1, 2019.

[19] WEKA, "Weka 3: Data mining software in java," https://www.cs.waikato.ac.nz/ml/weka/, accessed November 1, 2018.

[20] C. Murphy, G. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing," in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE08)*, 2008, pp. 867–872.

[21] X. Xie, J. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Application of metamorphic testing to supervised classifiers," in *Proceedings of the 9th International Conference on Quality Software (QSIC '09)*. IEEE, 2009, pp. 135–144.

[22] D. C. Jarman, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing for Adobe data analytics software," in *Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET '17), in conjunction with the 39th International Conference on Software Engineering (ICSE '17)*, 2017, pp. 21–27.

[23] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[24] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[25] O. A. Abbas, "Comparisons between data clustering algorithms." *International Arab Journal of Information Technology (IAJIT)*, vol. 5, no. 3, 2008.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.