Optimizing Training Data Selection for Decision Trees using Genetic Algorithms

Ahmad Alhindi[1†]
ahhindi@uqu.edu.sa
Computer Science Department, Umm Al-Qura University

Summary
Genetic algorithms (GAs) have recently been used as a search method for training set selection in supervised machine learning. The assumption is made that not all the data are equally useful in training supervised algorithms. In this paper, we empirically study the performance of classical GA for selecting a ‘good’ training set for decision tree classifiers. We also discuss different fitness functions and their influence on the results. A set of widely used classification datasets from Kaggle and UCI machine learning repository are used. Empirical results show that improved generalization can indeed be obtained using this approach.

Key words:
Genetic Algorithm (GA); Machine Learning (ML); Supervised Learning; Optimization; Training Set Selection; Data Mining; Pattern Recognition.

1. Introduction

Machine Learning (ML) and optimization are two different fields of fundamental computer science research. Optimization is aimed at modelling, designing and implementing solution techniques to find the optimal entity from a set of candidate alternatives for a particular computer search problem. Usually, an optimization process starts from a single or set of initial solutions and moves step by step towards the best solution directed by an objective function [1]. Classification of patterns and discovery of knowledge issues require the selection of a subset of features to represent the patterns to be classified. This is because the classifier’s performance and the classification cost are sensitive to the range of features used to construct the classifier [2]. Training data could contain noisy or incorrect information in many real-world applications, and the quality of the best classifiers could also deteriorate when dealing with these data [3]. Training Set Selection (TSS) [4] is an appropriate and solid approach to dealing with these issues. TSS is a pre-processing technique which selects only the relevant instances of the data set before performing training and classification tasks [5]. Genetic algorithms (GAs) offer an attractive approach. They are population-based search approach that can find optimal or near optimal solutions to such optimization problems quickly and effectively [2] [1]. GAs reflect a clever use of a random search used to solve problems of optimization. The GA’s basic techniques are designed to simulate the processes required for evolution in natural systems.

Supervised machine learning methods are given more attention and are widely used now days. These methods task is to discover relationship between the input attributes and the target attribute. The discovered relationship is referred to as model in machine learning community. Classification models, called classifiers, maps the input attribute to predefined target classes. There are many alternatives to represent classifiers. Decision tree (DT) [6] are probably the most used approaches for classification and widely used for and applied to numerous applications [7], [8], [9]. This study considers the DT classifier and empirically testing the impact of classical GA on selecting the training set for the DT with classification task. We show how the GA can be successfully applied for selecting the training data instances to improve the classification accuracy as well as reduce the size of the training set.

Dataset Splitting emerges as a necessity to eliminate bias to training data in ML algorithms. Hence, to get the best validation and learning outcome the data is divided into training data and testing data, see Figure 1. The decision tree is first trained against the training set, then asked to predict output from the testing set (or ’hold out’ set). In this paper, hold-out method is used to achieve this separation as we only have one model, the decision tree, to evaluate and no hyper-parameters to tune. A common split when using the hold-out method is using 20% of data for testing and the remaining 80% of the data for training. This work first builds and train the decision tree on the entire training set. Then, we experiment building and training the decision tree on a training selected set via the GA as instance selection of the training set. GA is known to have no assumptions on the data structure or the classifier family. Instead, it is a population-based method for instance selection that can be used to gain higher classification accuracy. This makes GAs good choice and suitable for training set selection.

Manuscript received April 5, 2020
Manuscript revised April 20, 2020

The paper is structured as follows. The next section provides related work. The training set selection is then presented and formulated in section three. Section four introduces the genetic algorithm and its components for the training set problem. Section five presents experiments and results. Finally, concluding remarks and future directions are presented in section six.

2. Related Work

Machine learning depends heavily on data. It is crucial to make algorithm training possible. In real-life applications, quality of data is extremely important. Thus, data preparation and preprocessing (e.g., remove duplicate and noise) is an integral step in machine learning projects. These applications within the projects do required many data, however, this does not mean to feed the system with every known data instance in related field. Machine learning models need to be fed (train) with carefully curated/selected data, hoping it can learn and extend or generalize to the future. Classification heavily relies on training the model with training data, which is a subset of the original dataset. In reality, original datasets have enormous number of elements and it is a difficult to train a model on such huge datasets. Since restrictions such as storage requirements and computational cost may impose on the application of machine learning algorithms, especially when dealing with large datasets [10].

For instance, Jansowski and Grochowski pointed out in their survey [11] that large datasets may contain noise (incorrect data) and several algorithms may be fragile to the noise. Chin et al. in [12] highlighted large datasets require increased memory usage and have computational complexity. These problems give rise to a need of searching for methods suitable to reduce the size of the dataset without a loss of quality of the provided information [10].

Training set selection (TSS), also called instance selection (IS), is carried out by selecting a subset of samples from the original training set [5]. The right training set selection is designed to choose representative sample by reducing or removing noisy and/or redundant ones. Clearly, TSS technique based on the goals to reduce the training time and dataset size without losing classification accuracy. TSS has been applied to different areas of machine learning including classification, regression, and time series prediction [13]. Following, we briefly review some works.

Cano et al. [14] discussed a combination of stratification and evolutionary technique-based algorithm for training set selection in large size datasets. They used stratification to reduce domain size then the evolutionary algorithm selects the most representative instances from the stratified selection.

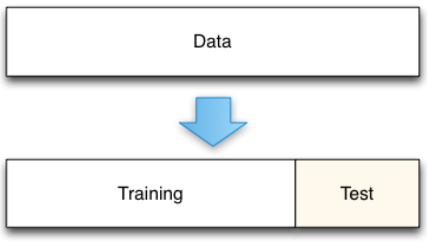


Fig. 1 Illustration of data splitting process for training and testing a classifier. The whole data instances split into two parts: training and testing.

Garcia and Herrera [15] consider imbalanced datasets in their research on evolutionary training set selection. Imbalanced sets refer to the situation when data contains many outcomes belonging to one class and few ones from the other class. This work applies evolutionary TSS by under-sampling that is, removing outputs mainly from majority class and replicating or generating new minority output variables. The work uses C4.5 decision tree classifier. Results were compared with other under-sampling and over-sampling techniques on the basis of classification accuracy.

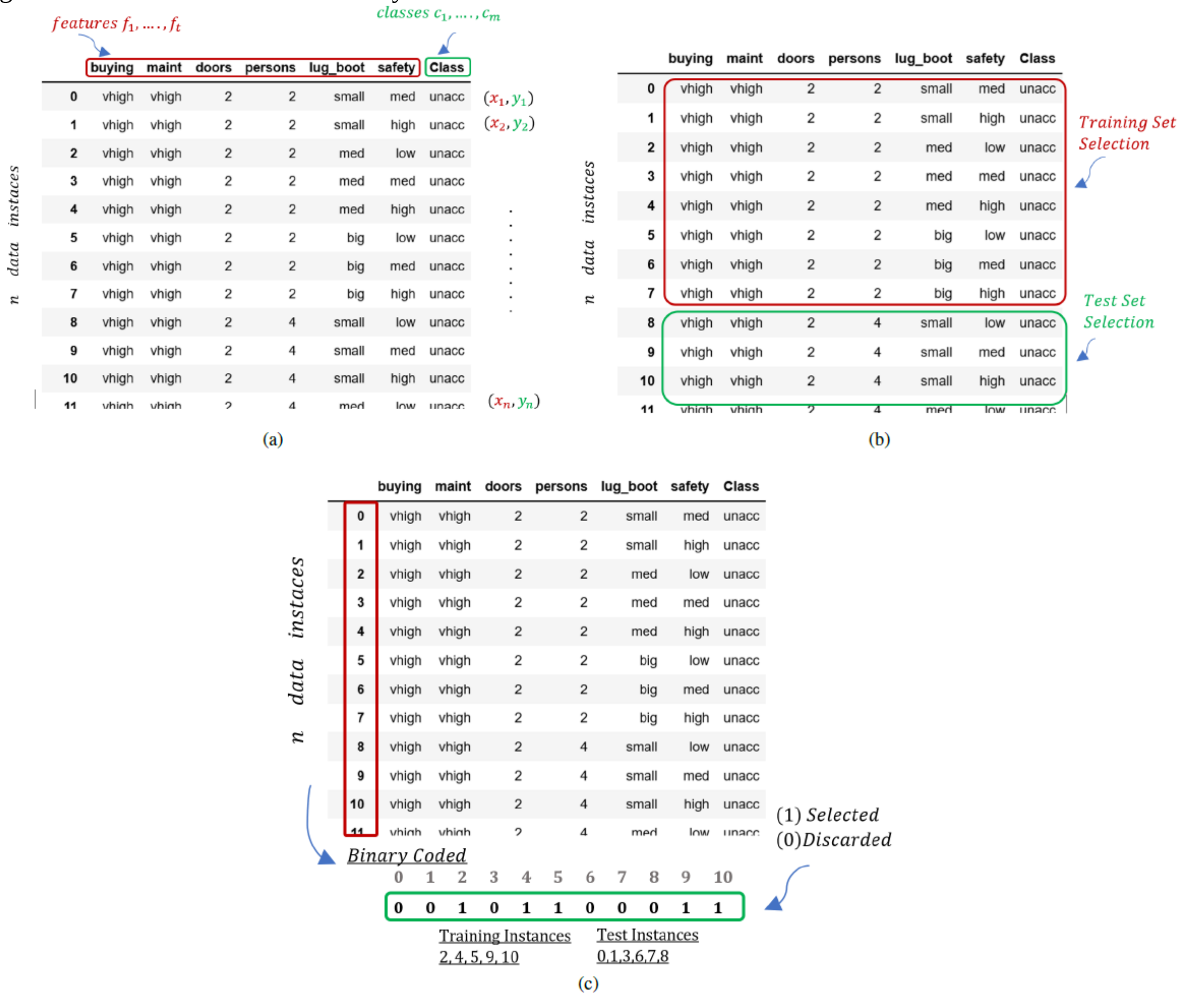
To improve the accuracy of SVM, Verbiest et al. [16] studied evolutionary approaches for TSS. Three evolutionary wrapper techniques were proposed for TSS, namely, Generational Genetic Algorithm, CHC evolutionary algorithm, and the Steady State Genetic Algorithm. Training sets generated from these three evolutionary techniques were compared with five wrapper approaches for classification accuracy of the SVM.

3. The Training Set Selection Problem

This section introduces and formulate the training set problem. It also states the optimization objectives.

3.1. Problem Formulation

This paper considers the training set selection (TSS). Let us assume that there is a training set comprised of instances. Consider every instance as a pair (with where is defined as an input vector with representing its corresponding class label. Be every input vector contains input information representing its corresponding instance. We can define the goal of a TSS method is to be able to produce a set of instances such that. Theoretically, any classifier can be trained using so that it can classify new instances with same or better accuracy when compared to training the classifier on. If the cardinality of set should be



	buying	maint	doors	persons	lug_boot	safety	Class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc
5	vhigh	vhigh	2	2	big	low	unacc
6	vhigh	vhigh	2	2	big	med	unacc
7	vhigh	vhigh	2	2	big	high	unacc
8	vhigh	vhigh	2	4	small	low	unacc
9	vhigh	vhigh	2	4	small	med	unacc
10	vhigh	vhigh	2	4	small	high	unacc
11	vhigh	vhigh	2	4	med	low	unacc

Fig. 2. An illustration of dataset and process of splitting and encoding the data instances. (a) shows data instances and its components where each of which consists of pair (x_i, y_i). Here, x_i consists of t features and y_i is the corresponding class labels. (b) an illustration to demonstrate the splitting of the data sets into two parts: training and testing. (c) Another illustration to demonstrate the GA's chromosome binary coded and how the selected instances form a training set and the discarded instances form a test set.

lesser than that of the original set, classifier training time can be considerably reduced.

Definition 1. (TSS) *Given an instance data set DS with n instances. Each data instance consists of pairs $(x_i, y_i), i = 1, \dots, n$, where x_i denotes the i-th instance that consist of t features $f_1, \dots, f_t$ and $y_i \in \{c_1, \dots, c_m\}$ is the corresponding class label. Here, m is the number of classes. The problem of selecting training instances is to find a subset $TR \subseteq DS$ with size n' $(n' < n)$, which yield to the same or higher predictive capabilities as the original set.*

One can see that selecting of training set is a search optimization problem with defined objectives, hence, a genetic algorithm can be applied.

3.2. Optimization Objectives

The TSS in this paper is to maximize the accuracy of the classification task via carefully selecting a representative subset of training instances. The goal is to build a classifier trained on this small subset, while maintaining the same (or even higher) accuracy compared to when training on all the available training set. Thus, the task of the GA is to optimize the following two objectives:

- maximize classification accuracy
- minimize the number of training instances

Our aim by using the evolutionary search through the GA, is to increase classifier's performance and at the same time decrease the number of selected training data instances.

4. Genetic Algorithm for Training Set Selection

This section describes the design of a classical GA for training set selection.
Genetic algorithm maintains a population of solutions that allow the convergence to the optimal area in the search space quickly and effectively. In this paper evolutionary search, i.e., genetic algorithm, is considered to address the problem of selecting training data instances. The GA tries to optimize the aforementioned two objectives of data reduction rate and accuracy of decision tree and support vector machine classifiers.
Typically, population of individuals is initialization randomly. Each chromosome represents a training instance contained in a dataset. In what follows, we define the necessary components to apply GA to the TSS problem.

4.1 Chromosome Representation

The TSS problem can be regarded as grouping data instances into two classes, those being selected (training instances) and those being discarded (test instances). See Figure 2. Obviously, it is a grouping problem similar to the well-known Knapsack problem [17], [18]. Therefore, the binary code for the problem is very straightforward; just use a locus to represent the data instance and an allele to represent whether the data instance is selected (1) or not (0). Hence, this paper represents a solution z as a sequence of the digits, 0’s and 1’s (i.e., $z = (z_1, ..., z_n) \in \{0,1\}^n$). Also, selected data instances form a training set, while discarded ones form a testing set of instances. An example chromosome of the 10-instances problem is illustrated by Figure. 2(c).

4.2 Fitness Function

In this paper, we consider two maximization fitness functions for the task of instance selection. The first, is the classification accuracy alone, which defined as:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \qquad (1)$$

Another form of the above equation can be given in terms of positive and negatives in case of two class classification as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP and TN are the true positives and negatives, FP and FN are false positives and negatives.
We also define another maximization fitness function that combines the classification accuracy (Accuracy) and the percentage of reduction of selected instances (Reduction):

$$Fittness = (\alpha \times \text{Accuracy}) + ((1 - \alpha) \times \text{Reduction}) \qquad (2)$$

with

$$\text{Reduction} = 100 \times (\frac{|DS| - |TR|}{|DS|})$$

using this fitness function allow the simultaneous optimization multiple objectives that is accuracy as well as size of training set. Note that, this is not a multi-objective approach in that both objectives are dealt independent of the other.

4.3 Genetic Operators

Since the TSS problem is the genetic operators used are one-point crossover operator and single bit-flip mutation, which are standard operators.
More specifically, two individuals (parents) are selected to cross over, we assign the probability of crossover p_c of typical value from 0.6 to 0.9 [19] in order to control the possibility of performing it. Then, a random point is selected and the genes elements from each parent are combined, producing a new solution with features of both. For mutation we mutate the generated new solution, then undergo mutation operator, where a random position is selected and flipped (i.e., 1 to 0 or 0 to 1) with the probability p_m of typical values from 0.001 to 0.1 [20, 21].

5. Experimental Design and Result Analysis

In this section, we empirically conduct experiments using the GA on real-life Kaggle and UCI datasets to select training instances for the decision tree classifier. The programming language is Python and the scikit-learn machine learning library [22], which exposes a wide variety of supervised and unsupervised machine learning algorithms including decision trees is used. All the statistics are based on 30 independent runs.
The experiments were conducted on Windows 10 operating system with Intel Core i7 CPU and 8 GB of

memory. The source code of all the experiments is available upon request.

5.1 Description of the Datasets

In our experiments, selection training data approaches are tested on 10 public data sets. These are classic data sets for classification that are accessible from the UCI machine leering repository and Kaggle open dataset. Table I shows more details on the datasets. In the table, the datasets are sorted according to their sizes, i.e., to the number of data instances.
In the Table I, "Instances", "Attributes" and "Classes" refer respectively to the number of data instances, features and the classes in the datasets. For each dataset, we report the average values of 30 runs as the results.

Table 1: The detailed characteristics of the selected 10 dataset

Dataset	**Instances**	**Attributes**	**Classes**
Zoo	101	18	7
Iris	150	4	3
Wine	178	13	3
Breast Cancer	569	30	2
Pima India Diabetes	214	11	7
Tic Tac Toe	958	10	2
Banknote Authentication	1371	4	2
Car Evaluation	1728	6	4
Optical Recognition of Handwritten Digits	5620	64	10

5.2 Default Parameter

Settings of parameters has impact on algorithms. For GA, this may influence the performance, i.e., how efficiently it finds an optimal or near optimum solution.
The default parameters of GA used in the experiments were as follow: number of generations - 50, population size - 20, crossover probability pc - 0.7, mutation probability pm - 0.1, selection method - tournament selection, stop criterion - 50 generations. As for the decision tree classifier, it get trained with default parameters values set by the scikit-learn library on all the experiments.

Table 2: The average classification accuracy values obtained by Hold-out and GA among 30 runs. The number in parentheses represent the standard deviation.

Datasets	Decision Tree Accuracy (%)		
	Hold-out	GA-f_1	GA-f_2
Zoo	0.912 (0.039)	0.972 (0.008)	0.983 (0.009)
Iris	0.949 (0.034)	0.989 (0.006)	0.993 (0.007)
Wine	0.941 (0.052)	0.989 (0.008)	0.998 (0.006)
Breast Cancer	0.921 (0.020)	0.959 (0.006)	0.965 (0.006)
Pima India Diabetes	0.682 (0.029)	0.766 (0.008)	0.767 (0.009)
Tic Tac Toe	0.877 (0.029)	0.932 (0.009)	0.939 (0.011)
Banknote Authentication	0.682 (0.021)	0.961 (0.007)	0.951 (0.006)
Car Evaluation	0.972 (0.008)	0.987 (0.002)	0.985 (0.002)
Optical Recognition of Handwritten Digits	0.846 (0.014)	0.876 (0.007)	0.869 (0.007)

Table 3: Comparison between hold-out and GA on the size of training Data selected for decision tree.

Datasets	Training Set Size (%)		
	Hold-out	GA-f_1	GA-f_2
Zoo	0.75	0.406	0.307
Iris	0.75	0.506	0.353
Wine	0.75	0.443	0.371
Breast Cancer	0.75	0.525	0.525
Pima India Diabetes	0.75	0.509	0.444
Tic Tac Toe	0.75	0.537	0.447
Banknote Authentication	0.75	0.515	0.457
Car Evaluation	0.75	0.480	0.460
Optical Recognition of Handwritten Digits	0.75	0.158	0.148

5.3 Results

GA is compared with hold-out method for selecting training set for the decision tree classifier. A set of ten datasets with different sizes are used in our experimental studies.
Table II shows the mean of the percent classification accuracy values that obtained by the decision tree on training sets generated by the hold-out and the GA methods when using two different objective fitness functions among 30 runs. Table III gives the size of the generated training sets via hold-out and GA with f1 and f2 fitness functions. Figure 3 plots the evolution and the convergence of the GA for both population fitness and global-best fitness of each dataset.
Table II shows that the classification accuracy of the decision tree trained on GA-based selection of training

data is better than hold-out based selection of training data.

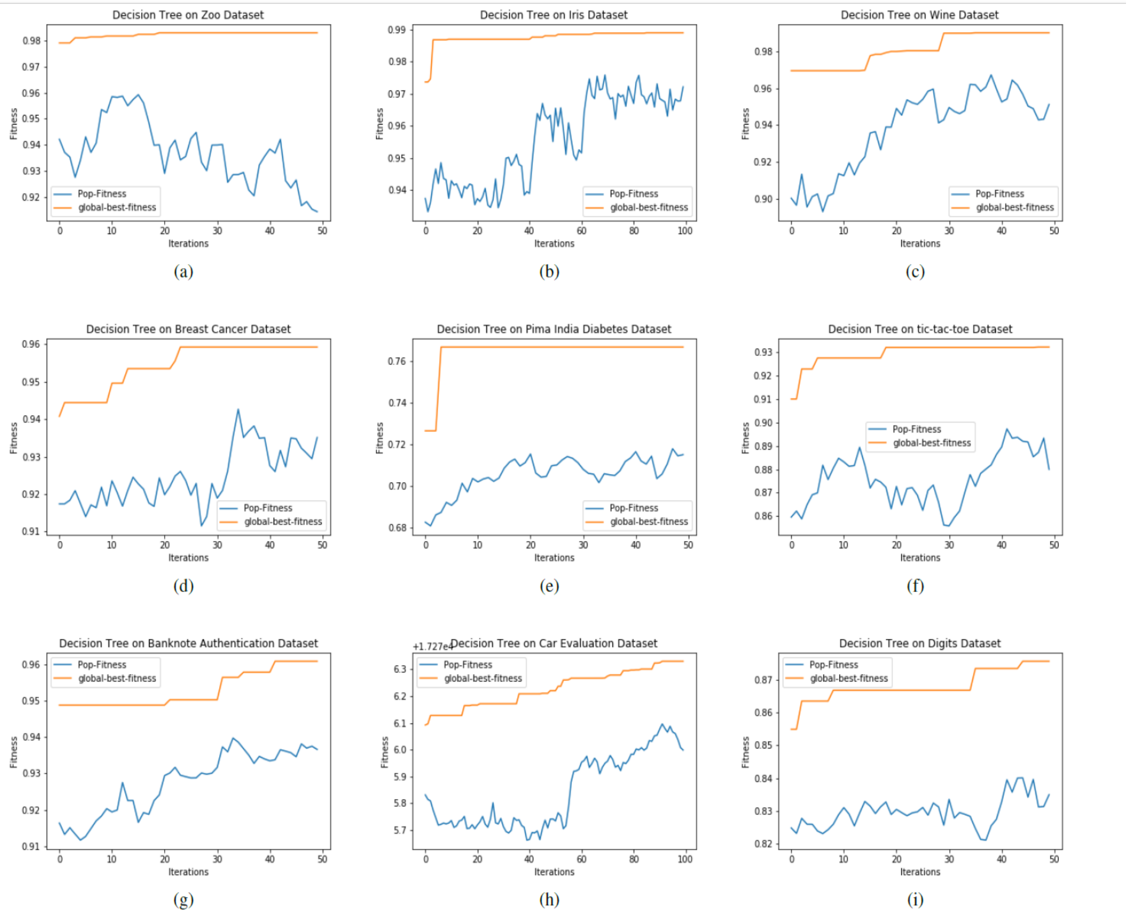


Fig. 3. Genetic algorithm evolution and convergence of global-best fitness and the the population fitness after running the algorithm for selecting of training Data for Decision Trees classification over all datasets.

The results reflect the extend of enhancement GA made to the training process and the ability to optimize the training data.

Taking the tic-tac-toe dataset as example, decision tree has accuracy value of 87.7% when the hold-out was used and 93.9% when the GA was used.

In term of the size of training set, we can see from Table III that a clear reduction on the training set after the application of GA optimization on all dataset.

It is evident from Tables II and III that the training data selection with simultaneous optimization of accuracy and reduction of selected data instances is better than optimizing accuracy alone.

These results ensure the importance choice of a fitness function for the optimization process. In order to get more insight on the behavior of GA, Figure 3 visualizes the evolution and convergence behavior for decision tree on all the ten datasets.

We can see how optimizing training set selection-based accuracy helps to achieve higher rate of convergence to good solutions. The results shown in this figure are consistent with the observation on the obtained accuracy values of Table II.

Moreover, Figure 3 reveals the different search behavior of GA in small and large search spaces. GA converges toward promising regions of the search space efficiently, effectively, and intelligently.

6. Conclusion

Searching and selecting of relevant data instances is essential for enhancing the performance of classification algorithms. In this paper, a genetic algorithm search method has been used for optimizing training data selection for decision tree. Our experimental results on various dataset with different characteristics indicates optimizing training data selection for decision tree is a very effective technique. Our method showed signification reduction rates compared to classical methods among all the considered datasets. We have also shown that using different fitness function for selecting training data has its impact on the result. As future work, more datasets and genetic operators will be investigated.

Acknowledgments

The author would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: 19-COM-1-01-0022.

References

[1] H. Song, I. Triguero, and E. O¨ zcan, "A review on the self and dual interactions between machine learning and optimisation." Progress in Artificial Intelligence (2192-6352), vol. 8, no. 2, p. 143, 2019.

[2] J. Yang and V. Honavar, Feature Subset Selection Using a Genetic Algorithm. Boston, MA: Springer US, 1998, pp. 117–136.

[3] N. Verbiest, J. Derrac, C. Cornelis, S. Garc´ıa, and F. Herrera, "Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis." Applied Soft Computing, vol. 38, pp. 10 – 22, 2016.

[4] S. Garc´ıa, A. Ferna´ndez, and F. Herrera, "Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems," Applied Soft Computing, vol. 9, no. 4, pp. 1304 – 1314, 2009.

[5] G. Acampora, F. Herrera, G. Tortora, and A. Vitiello, "A multi-objective evolutionary approach to training set selection for support vector machine," Knowledge-Based Systems, vol. 147, pp. 94–108, 2018.

[6] L. Rokach and O. Z. Maimon, Data mining with decision trees: theory and applications. World scientific, 2008, vol. 69.

[7] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE transactions on systems, man, and cybernetics, vol. 21, no. 3, pp. 660–674, 1991

[8] S. B. Kotsiantis, "Decision trees: a recent overview," Artificial Intelligence Review, vol. 39, no. 4, pp. 261–283, 2013

[9] V. Stankovski and J. Trnkoczy, "Application of decision trees to smart homes," in Designing smart homes. Springer, 2006, pp. 132–145

[10] M. Grochowski, "Simple incremental instance selection wrapper for classification," in International Conference on Artificial Intelligence and Soft Computing. Springer, 2012, pp. 64–72

[11] N. Jankowski and M. Grochowski, "Comparison of instances seletion algorithms i. algorithms survey," in International conference on artificial intelligence and soft computing. Springer, 2004, pp. 598–603

[12] Y.-c. I. Chang, Y.-J. Lee, H.-K. Pao, M.-H. Lee, and S.-Y. Huang, "Data visualization via kernel machines," in Handbook of Data Visualization. Springer, 2008, pp. 539–559

[13] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance election algorithm to reconstruct training set for support vector machine," Knowledge-Based Systems, vol. 116, pp. 58–73, 2017

[14] J. R. Cano, F. Herrera, and M. Lozano, "On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining," Applied Soft Computing, vol. 6, no. 3, pp. 323–332, 2006

[15] S. Garc´ıa and F. Herrera, "Evolutionary training set selection to optimize c4. 5 in imbalanced problems," in 2008 Eighth International Conference on Hybrid Intelligent Systems. IEEE, 2008, pp. 567–572

[16] N. Verbiest, J. Derrac, C. Cornelis, S. Garc´ıa, and F. Herrera, "Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis," Applied Soft Computing, vol. 38, pp. 10–22, 2016

[17] A. Alhindi, Q. Zhang, and E. Tsang, "Hybridisation of decomposition and grasp for combinatorial multiobjective optimisation," in 2014 14th UK Workshop on Computational Intelligence (UKCI). IEEE, 2014, pp. 1–7

[18] D. Pisinger and P. Toth, "Knapsack problems," in Handbook of combinatorial optimization. Springer, 1998, pp. 299–428

[19] D. Whitley, "A genetic algorithm tutorial," Statistics and computing, vol. 4, no. 2, pp. 65–85, 1994

[20] L. Davis, "Handbook of genetic algorithms," 1991

[21] R. Wazirali, W. Alasmary, M. M. Mahmoud, and A. Alhindi, "An optimized steganography hiding capacity and imperceptibly using genetic algorithms," IEEE Access, vol. 7, pp. 133 496–133 508, 2019

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in python," Journal of machine learning research, vol. 12, no. Oct, pp. 2825–2830, 2011

Ahmad Alhindi received the B.Sc. degree in computer science from Umm Al-Qura University (UQU), Makkah, Saudi Arabia in 2006, the M.Sc. degree in computer science and the Ph.D degree in computing and electronic systems from University of Essex, Colchester, UK , in 2010 and 2015, respectively. He is currently an assistant professor in Artificial Intelligence (AI) with computer science department at UQU. His research focuses on Evolutionary Multi-objective Optimization and Machine Learning techniques. Currently, he is working on AI algorithms, focusing particularly on machine learning and optimization with a willingness to implement them in a context of decision making and solving combinatorial problems in real world projects.