Indonesian Journal of Electrical Engineering and Computer Science
Vol. 29, No. 1, January 2023, pp. 496~508
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v29.i1.pp496-508

A new density core graph-cut class decomposition to improve neural network classification performance

Eakawat Tantamjarik, Thitipong Tanprasert
Department of Computer Science, Assumption University, Bangkok, Thailand

Article Info

Article history:

Received Jul 18, 2022
Revised Sep 18, 2022
Accepted Oct 3, 2022

Keywords:

Data preprocessing
Density core
Neural network
Shape representation
Supervised clustering

ABSTRACT

This research presents a new pre-processed class decomposition technique called density core graph-cut (DCGC). The method uses supervised clustering instead of a traditional unsupervised one to decompose the class. Supervised clustering requires additional label information to function and with that it gains a better understanding of the distribution. DCGC employs nearest neighbors to form a density core graph for each class. Then, the edges of each graph to be removed or cut is identified utilizing class information. Lastly, it yields final clusters by grouping all connected cores and assigning the remaining samples to a cluster where the nearest core belongs. Training neural network classifiers on complex label data will yield a better accuracy with the modified class representation. Intuitively, the decision boundaries separating classes based on the modified labels are less complex, and classifiers' chance to reach these hyperplanes is higher. The results present that training neural networks using label representations from DCGC significantly helps improve the classification accuracy of neural networks on syntactic datasets as high as 30%. For real-world problems, the experiment presents a mixed result in which some datasets moderately benefit from DCGC.

Corresponding Author:

Eakawat Tantamjarik
Department of Computer Science, Assumption University
Bangkok, Thailand
Email: p5929444@au.edu

1. INTRODUCTION

Classification is a problem in which the class knowledge is learned from the input data called training data, whose class is known. This learned information is then used to predict the new data with an unknown class. From this scenario, it can be seen that the classification performance depends on the class knowledge or label from the training data. One example of poor classification accuracy is insufficient representative information of the data which causes the classifier to learn incorrect classification. It can be solved using pre-process technique called data augmentation [1]. This process adds more samples to the training data by simply modifying the original sample in a way (rotation for images, synonym addition for text, or outpost vector for tabular data [2]) that the meaning of the sample does not change. Thus, it helps the classifier become more generalized and better perform with unforeseen data. Another problem that reduces classification performance is the complexity of class labels where data points belonging to different classes are aggregated in the same region. As a result, the classifiers can have difficulty modeling these classes correctly. In this case, a simple solution is to increase the complexity of the classifier. However, this comes with the cost of training time and may introduce overfitting [3] due to the classifier trying to fit the training data and losing the generalization. Another approach is to use fine-grain subclass labels, given that fine-grain subclass labels need to have meaningful hierarchical relationships with original coarse-grain class labels [4]. Otherwise, it will likely

degrade the classification performance [5]. Apart from subclass labels needing meaningful relationships, obtaining them is also a non-trivial task. One method to get subclass labels requires manual input from human experts which is time-consuming and expensive. A cheaper alternative is to generate subclass labels via class decomposition automatically.

Class decomposition or Subclassing is a preprocessing technique that can be used to help improve a classifier [6]. It is achieved by splitting the target class into multiple subclasses using some criteria then training with target subclasses instead of original classes. Using subclass instead of the original class has the benefit of making the classifier generate a more manageable decision boundary. Usually, correctly splitting class will yield simpler shaped data distributions [7]. A common method used to achieve class decomposition is through performing clustering. Clustering is an unsupervised learning technique that does not require the target class to operate. It starts by assuming that each data does not belong to any class, then completes when all of them are assigned to a class or when the criteria is reached. The idea of using unsupervised clustering to help improve supervised classification had been widespread in the earlier days of the area, such as local expert (or hierarchical classifier) [8], [9]. This mechanism uses a clustering technique to partition training data and train classifiers separately on each partition. Later, using subclasses for classification tasks was also discovered to improve performance. It was discovered much later because, from the initial understanding of the idea, it is counter-intuitive [10].

Increasing the number of classes had been thought to increase the complexity and result in more unsatisfactory performance, especially with the real-world data set in which the data distribution was not well understood and was hard to visualize. Although a higher number of classes indeed increases the difficulty, there are certain settings which increasing the number of classes yields better classification accuracy [11], [12]. One such setting is the use of class decomposition on data that has classes distributed in complex ways-by dividing the original class into subclasses in a meaningful way [10], [13], then using subclasses to train may result in improved accuracy than using the original class. The common clustering algorithms used in class decomposition are K-Means [13]-[17], EM clustering [6], [18], and hierarchical clustering [19], [20]. These clustering algorithms help create homogenous subclasses for the same label to some extent, but in each cluster, other classes' presence is completely ignored. This can cause the resulted clusters to be overlapped with other clusters as depicted in Figure 1. Although class decomposition has knowledge about label information beforehand, using the traditional unsupervised clustering technique was not optimal since that information was not being utilized. This additional information can be exploited to help guide the way classes are divided.

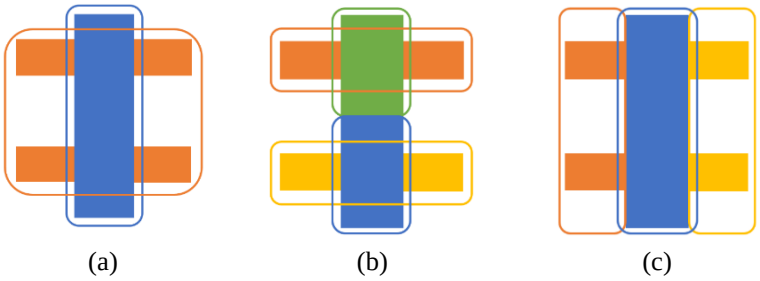

Figure 1. Overlap complication example for each class (rounded color border) of (a) 2-class dataset (orange and blue), (b) class decomposition by K-Means with k=2, and (c) ideal class decomposition

Recognizing the shortcoming of a traditional clustering method, [21] coined the term Supervised Clustering in their pioneer work that utilized the opposite class's information to help improve clustering performance. Unlike others that clustered each class separately, [21] proposed three modified supervised clustering based on K-medoid (SPAM), greedy approach (SRIDHCR), and genetic algorithm (SCEC) to group class data using a custom impurity fitness function. The function was created to encourage obtaining the cluster representative as being pure by having fewer samples from other classes within the neighbor and as few clusters as possible. Using the impurity idea, [22] proposed supervised growing neural gas (SGNG) to take class information into account and [23] updated SCEC to include additional objective fitness functions. To the best of our investigation, although the proposed impurity idea performed well on the supervised clustering task, it is not an ideal solution to class decomposition for training a classifier. For example, applying supervised clustering to the rectangle shape data will likely divide it into multiple square shape clusters since that usually yields a better impurity value. However, these additional separations complicate the classifier because it needs to learn more hyperplanes that divide those square clusters instead of only a circumference surrounding the rectangle. Thus, obtaining a better fitness function evaluation does not necessarily make it easier to train a

classifier. Supervised clustering also suffers from local optima when the data contains many isolated class distributions and the same class samples are scattered through those regions since any update in this scenario will rarely have an improvement.

Given the above limitation in both traditional unsupervised and supervised clustering methods, the main contribution of this work is to develop an algorithm to decompose data that has a complex class distribution into a simpler subclass distribution with the aim of obtaining a better accuracy when training the classifier using the generated subclasses compared to the original classes given the comparable classifier complexity. Furthermore, the proposed algorithm should eliminate the need to specify the hard-constraint number of cluster. Although some works address this issue [24], [25], it is not optimal for class decomposition because each class will have a different amount of ideal clusters and it is hard to determine the best combination of them. Lastly, the algorithm must balance split/ignore to avoid making clusters out of the already simple class distribution.

2. RESEARCH METHOD

2.1. Clustering step via density core graph-cut (DCGC) class decomposition

The proposed algorithm is based on an approach that uses a density of data samples in order to identify isolated regions. It is inspired by the concept of using density peaks [26] to represent dense regions and form a skeleton for each cluster. The algorithm uses core samples as a representative point to construct a graph of each cluster. Then, the data regions belonging to different representative points can be determined as either connected or disconnected. This connection concept is the key to overcoming both the limitation of traditional unsupervised clustering and supervised clustering. A simple counting is used to judge the connectedness. The connection of representative points implies that those points are in the same cluster, while the disconnection signifies the opposite. Finally, the algorithm outputs a new label for each sample and a dictionary mapping those labels back to the original class. The main function of DCGC consists of four steps; 2.1.1. identifies the representative samples, 2.1.2. creates density representative graph, 2.1.3. disconnects density representative graphs, and 2.1.4. cluster formation.

2.1.1. Identify representative sample

Let X be a set of data pairs denoted as $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where x_i is a sample that belongs to a class y_i, and n is the total number of samples. Each x_i is a vector in which each entry is an attribute that defines the characteristic of a sample x_i. This vector can be denoted as $(a_1, a_2, \dots, a_d)$, where a_j is an attribute value and d is the number of attributes of a sample. The space of all possible attribute vectors is called the input space X, and the space of all possible classes is called output space Y.

The first step of the algorithm starts by finding a k-nearest neighborhood of each sample x_i to calculate density [27], as this is much more robust technique than setting cutoff distance [28]. The first input parameter, which is denoted as κ and mathematically described in (1), is used to obtain the amount of density for each sample as expressed in (2).

$$\kappa = \{i \in N : 0 < i \leq n\} \tag{1}$$

where N is a set of natural numbers.

$$\rho(x_i) = \frac{\kappa}{\sum_{x_j \in kNNC(x_i)} dist(x_i, x_j)} \tag{2}$$

where $kNNC(x_i)$ is a set of k-nearest-neighbor samples that belong to same class of sample x_i and $dist(\cdot,\cdot)$ is a euclidean distance.

The reason behind adding a class constraint k-nearest-neighbor is to avoid having different class samples influencing the density calculation. In addition, it helps prevent the isolated samples that are close to other class samples from using those as their neighborhood and gaining higher density. In (2) also explains how the sample is distributed in their neighborhood by taking the distance from itself to all κ neighbors. The lower the average distance (higher ρ) to their neighbors indicates the tighter the neighborhood is, and the sample in question is located well in a dense region. These samples are considered prime candidates to become representative samples and identifying them is as simple as comparing the ρ value among their κ peers. The one which produces the highest ρ among their neighbors then becomes a representative sample as defined below.

$$RP_c = \{x_i \mid x_i \in X, \rho(x_i) > \rho(x_l), x_l \in kNNC(x_i), y_i = c\} \quad (3)$$

where RP_c is a set of all the representative samples belonging to class c. Figure 2 displays the original class dataset and Figure 3 illustrates the resulted representative samples (red X mark) which have the highest ρ among their neighbors (black lines) when κ = 6.

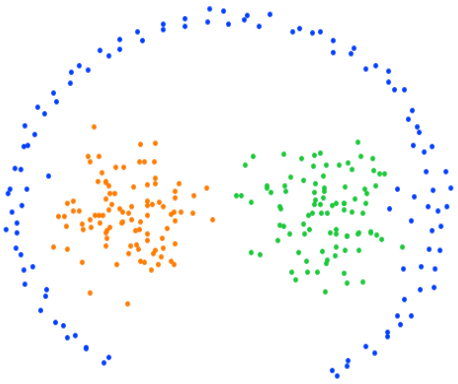

Figure 2. Original class dataset

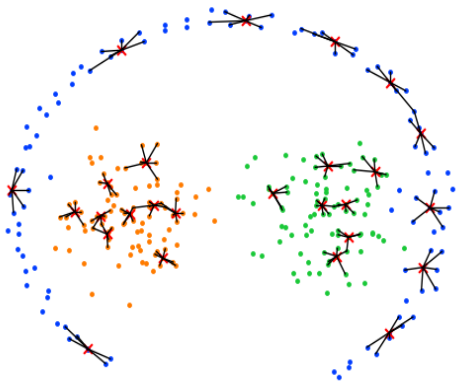

Figure 3. Representative points (red X mark) using κ = 6 and their class six-nearest neighbors

2.1.2. Create density representative graph

The second step of the algorithm is to form a skeleton graph for each class called representative graph via minimum spanning tree (MST). It is denoted as $G_{RP_c} = MST(V, E)$, where V is a set of representative samples from class c (RP_c), and E is a set of pairwise distances between all representative samples in RP_c. The resulted G_{RP_c} will have only $|V| - 1$ edges remain. The benefit of using MST is to guarantee that the average distance between representative points is one of the lowest. This property is the best representation of the skeleton aspect for a cluster [29] because it enforces that for the point on one end to reach the point on the other end needs to go through intermediate points along adjacent paths, which helps preserve the cluster's shape. Figure 4 illustrates the above scenario, for A to reach B, it needs to pass through all the representative samples from the blue cluster.

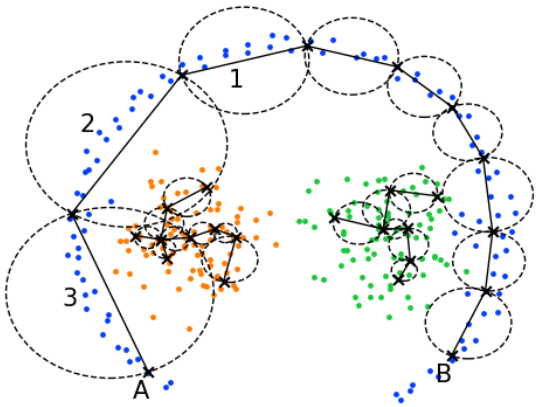

Figure 4. Representative points (x mark) using κ = 6 and enclosed region (dashed circle) along adjacent

Next, the algorithm uses the constructed G_{RP_c} to find the enclosing region between two representative points by first calculating the middle point between a pair of adjacent vertices v and w as (4).

$$M(v, w) = \frac{v+w}{2} \quad (4)$$

Then, the number of samples with the same class (NS_c) and different classes (NO_c) enclosed in a pair of adjacent points can be obtained using (5) and (6), respectively.

$$NS_c(v,w) = \left|\{x_i \mid x_i \in X, d\left(x_i, M(v,w)\right) < \frac{d(v,w)}{2}, y_i = c\}\right| \quad (5)$$

$$NO_c(v,w) = \left|\{x_i \mid x_i \in X, d\left(x_i, M(v,w)\right) < \frac{d(v,w)}{2}, y_i \neq c\}\right| \quad (6)$$

where y_i is a class of sample x_i.

Finally, the density information along the adjacent points is combined into a tuple and defined as (7).

$$D_c = (NS_c, NO_c) \quad (7)$$

Fundamentally, these equations are constructed to find samples enclosed in a hyper-sphere between adjacent points. This is inspired by the way overlapping is detected using enclosing circles with maxdist in [30], but modified to detect overlap along representative graph's edges. Figure 4 displays the hyper-sphere according to (5) and (6). For instance, D_c for the enclosed circle number "1" in Figure 4 will have the value (11, 0), circle "2" (18, 16), circle number "2" (19, 43).

2.1.3. Cut density representative graph

The density information D_c indicates whether the two adjacent representative points are in the same cluster. Consider the case where $NS_c > NO_c$, the region contains more samples with the same class between the two points, and this can be interpreted as the region probably is positioned at the cluster's heart. On the contrary, $NS_c < NO_c$, means that there are many other class samples between the two points, which can be thought of as an overlapping region. This interpretation allows the cut to be made to divide a representative graph into multiple representative subgraphs for those in adjacency with high NO_c since it is unlikely for the two points to belong to the same cluster. The cut-off ratio, denoted as γ and defined in (8), is a second parameter to DCGC algorithm. It controls how much tolerance the number of other class samples can be inside the enclosing region for the algorithm to consider removing the edge connecting two points as defined in (9). The γ should be less than 0.5 to not consider the opposite case.

$$\gamma = \{r \in R: 0 < r \leq 0.5\} \quad (8)$$

$$cut_c^i = \frac{NO_c^i}{NS_c^i + NO_c^i} > \gamma \quad (9)$$

where i indicates the i^{th} edge of G_{RP_c} and cut is a flag that indicates whether to remove this edge in class c. Figure 5 demonstrated the cut in action using γ = 0.2, Circle number "2" and "3" have been removed because the number of other class samples exceeds the ratio according to (9).

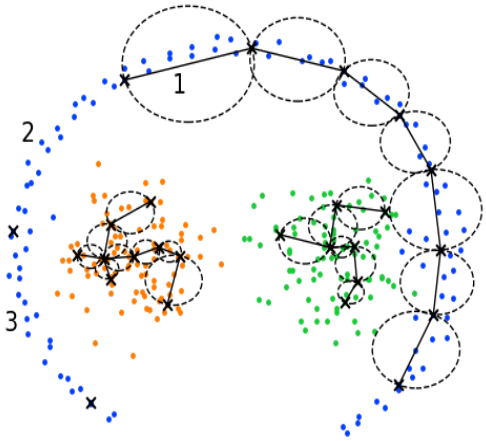

Figure 5. The density graph Figure 4 applying cut off with γ = 0.2

2.1.4. Cluster formation

In the last step, the representative graph cut result is then used to create clusters. Each subgraph, produced in the previous step in each class, is assigned a unique label. All representative samples in a subgraph are also assigned to the same label as the subgraph they belong to. Lastly, the remaining samples are assigned to the same label as their nearest representative samples from the same class, as shown in Figure 6. All the samples get their new labels, a dictionary mapping new subclasses to their original classes is created, and DCGC algorithm is completed.

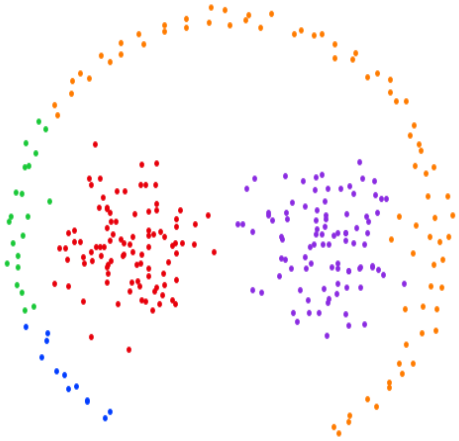

Figure 6. The result of DCGC algorithm with κ = 6 and γ = 0.2 in which the color of each sample represents a new subclass they belong to

2.2. Classification step

In this step, the classifier uses the result from DCGC to train and evaluate the performance. During the training phase, the new labels created by DCGC are used in place of the original labels from the dataset. The evaluation is done using the new labels to compute validation loss. Lastly, the accuracy is measured by comparing the original input class with an original label obtained by converting back new labels predicted from a classifier using the output dictionary from clustering step.

2.3. Complexity analysis

The computational complexity of DCGC is determined by four main parts: the search for density core samples, the construction of MST, the calculation for the cut, and the formation of clusters. In the first part, it can be further divided into two processes: nearest neighbors search $O(n^2)$ and locate density peak $O(\kappa n)$. n is number of samples and κ is input nearest neighbors. The nearest neighbor search can be further improved using k-d tree [31] and reduce to $O(nlogn)$. The second part requires $O(mlogm)$, where m is number of representative samples. Both the third and last part loop through all Representative Samples and count/assign samples, both computation complexities are $O(mn)$. Therefore, the overall computational complexity of DCGC can be expressed as $O(nlogn + \kappa n + mlogm + mn + mn)$. Usually, κ and m are significantly smaller than n, hence the complexity of DCGC is $O(nlogn)$.

3. RESULTS AND DISCUSSION

To assess the effectiveness of the DCGC class decomposition in improving neural network performance, we compared subclasses generated from DCGC with the original class and other class decomposition techniques: K-Mean [15] and supervised clustering using evolutionary computing (SCEC) [21]. DCGC is applied using five different neighborhoods (κ = 10, 30, 50, 100, 200) and two different cut-off ratios (γ = 0.2, 0.35). The number of clusters for each class (K_c) of K-mean is selected from the output of DCGC. For SCEC, we set number of generation to 200 and the remaining algorithm parameters are set according to [21] with the modification to ignore populations that do not contain samples from all original class. Four possible neural network architectures with 10, 50, 100, and 200 hidden neurons were employed during the classification phase. However, only the technique which creates the most clusters is set to those number of

neurons for a hidden layer. The hidden layer of neural networks for other techniques is adjusted to ensure the number of trainable parameters is identical. This additional step is required to guarantee that neural network complexity remains roughly equal between different class decomposition techniques. It will ensure that the improvement/loss in performance is from how the way subclass is distributed and not the more capable network.

We conduct experiments on eleven datasets: six syntactic datasets and five real-world datasets. Each dataset is split into training and testing sets with a ratio of 0.8 and 0.2, respectively. Both the class decomposition step and training neural networks are operated on only the training set. The testing set is used during the final evaluation for classification accuracy. Training neural networks are run five times with 100 epochs to obtain an average classification accuracy on the testing set.

3.1. Experiment 1-syntactic dataset

The first experiment used 2-d six syntactic datasets with various distribution shapes to assess DCGC. Figure 7 visualizes all the datasets and Table 1 displays the number of subclasses obtained from class decomposition for each techniques. From Table 1, it can be inferred that DCGC is very robust with various values of κ and γ on clean datasets (D11-D13). The amount of subclasses generated is somewhat the same from multiple configurations. This is expected since the nature of DCGC will merge clusters between representative points when there are a small amount of other class samples. Clean datasets have no such noise to prevent the merger, so whether DCGC starts with κ=10 or κ=200, both merge most representative samples. In contrast, κ is more affected by noise in the dataset when the value is relatively low. The result of D14-D16 when κ=10 creates more than a hundred subclasses which are incorrect from a human perspective. This is because small κ produces a higher number of representatives and causes the enclosing regions between them to be smaller. Some enclosing regions are small enough to make it disconnected no matter what value γ is specified, resulting in more subclasses. For example, considering an enclosing region between two representative samples, which contains two samples from another class and one from the same class, unless γ=0.5, other cases will consider separating these representative samples.

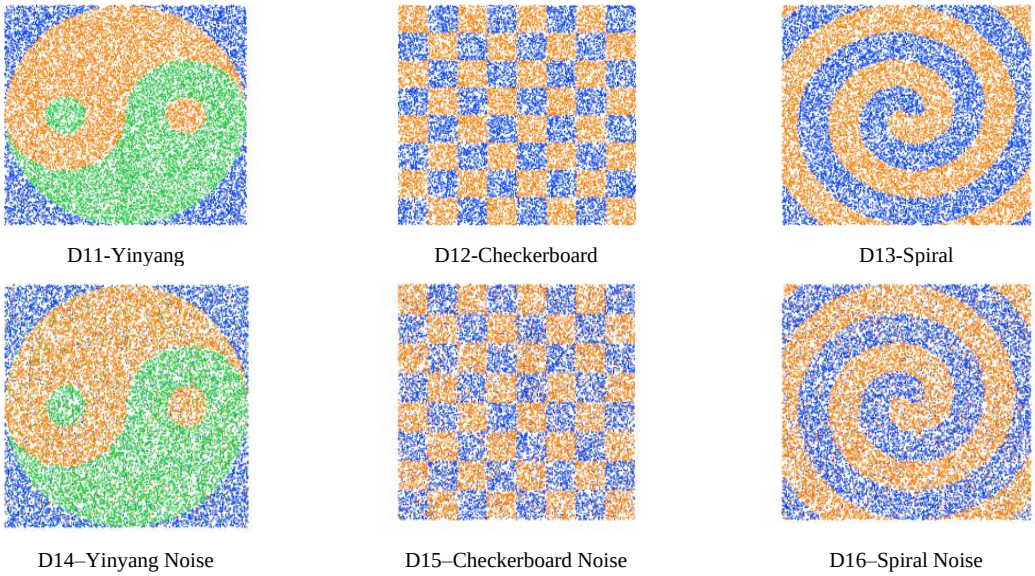

Figure 7. Experiment syntactic datasets

Table 1. Syntactic - Number of subclasses created with different algorithm

			DCGC and K-Means									
	Original	SCEC	10		30		50		100		200	
			0.2	0.35	0.2	0.35	0.2	0.35	0.2	0.35	0.2	0.35
D11	3	27	28	15	13	9	9	9	9	8	8	8
D12	2	37	80	58	66	62	64	63	64	64	64	64
D13	2	47	17	12	12	9	15	8	12	9	20	12
D14	3	25	219	108	37	26	20	16	14	10	11	8
D15	2	26	207	104	86	64	74	64	64	64	64	64
D16	2	51	171	55	31	10	15	8	17	8	25	18

Table 2 illustrates the best classification results using both original and subclass from different class decompositions. In Table 2 Params represent the number of trainable parameters and in case of DCGC it presents the value of κ and γ that is used in order to obtain the accuracy (Acc). We select the DCGC parameters that produce the best accuracy on each neuron network configuration and compare them with original class and other class decomposition techniques that have about the same trainable parameters.

It can be seen that using DCGC as class decomposition improves classification accuracy on all datasets compared to original class and outperform both K-Means and SCEC. The accuracy gained from applying class decomposition is as high as 30% compared with using original class on D12 and D15 dataset, where the class distributions are scattered in multiple regions. This type of dataset causes neural networks to struggle to separate those class. Increasing complexity (more neurons) does not help much since the accuracy starts to drop for the original class with the highest complexity neural network K-Means as class decomposition seems to perform well on this particular type of dataset given that the optimal K can be obtained. In this case, it performs comparatively with DCGC in D12 and is more robust to noise in D15. On the other hand, the improvement on D11 and D14 dataset is not significant since the class distributions are already well separated. Applying class decomposition in this case seems to reduce accuracy on both SCEC and K-Means while DCGC improves by less than 1% which can result from training fluctuation. However, when the dataset contains noise as in D14, using subclasses from DCGC appears more stable than the original class and improves accuracy by around 2% on low complexity neural network. For structure non-linear shape datasets D13 and D16, increasing complexity of neural network helps find those structures as accuracy does improve greatly for original class. Nevertheless, class decomposition also significantly improves this type of dataset accuracy, especially on small neural networks. All three methods remarkably boost classification accuracies and DCGC is the best of all.

Table 2. Summary of classification result

		Number of neurons in hidden layer for base case							
		10		50		100		200	
		Params	Acc	Params	Acc	Params	Acc	Params	Acc
D11	Original	327	97.22	1527	97.92	3027	98.15	6027	98.21
	SCEC	327	94.25	1527	96.95	3027	97.41	6027	97.56
	K-Means	327	95.62	1521	97.42	3021	97.82	6021	98.09
	DCGC	327 (100-0.35)	**97.63**	1521 (50-0.35)	**98.48**	3021 (50-0.2)	**98.52**	6021 (50-0.2)	**98.67**
D12	Original	732	65.62	3412	72.67	6762	76.72	13462	71.55
	SCEC	717	73.63	3397	80.24	6764	81.95	13437	83.09
	K-Means	734	92.85	3414	**95.47**	6764	94.76	13464	**95.94**
	DCGC	734 (200-0.35)	**94.05**	3414 (200-0.2)	95.39	6764 (200-0.35)	**95.42**	13464 (200-0.2)	95.22
D13	Original	547	78.47	2547	87.43	5047	89.25	10047	90.10
	SCEC	547	89.44	2547	93.55	5047	94.46	10047	95.40
	K-Means	537	84.15	2547	91.91	5037	92.38	10047	93.16
	DCGC	537 (50-0.2)	**94.56**	2547 (30.02)	**96.58**	5037 (50-0.2)	**97.03**	10047 (30-0.2)	**96.90**
D14	Original	303	92.24	1473	94.36	2925	95.78	5823	95.96
	SCEC	305	90.98	1453	94.31	2909	94.57	5821	94.71
	K-Means	296	89.96	1476	93.90	2926	94.37	5826	94.49
	DCGC	296 (100-0.35)	**94.55**	1476 (30-0.35)	**96.11**	2926 (30-0.35)	**96.48**	5826 (30-0.35)	**96.38**
D15	Original	732	61.48	3412	68.10	6762	66.50	13462	58.71
	SCEC	722	70.31	3390	78.32	6754	78.84	13453	79.22
	K-Means	734	**93.37**	3414	**93.84**	6764	**93.38**	13464	**93.50**
	DCGC	734 (200-0.35)	93.08	3414 (200-0.35)	93.16	6764 (200-0.35)	92.81	13464 (200-0.35)	92.31
D16	Original	587	79.07	2747	86.26	5447	86.53	10847	84.10
	SCEC	591	88.11	2751	92.23	5451	92.55	10851	92.86
	K-Means	577	86.18	2737	90.36	5437	91.12	10843	91.59
	DCGC	577 (100-0.2)	**91.80**	2737 (100-0.2)	**94.03**	5437 (30-0.2)	**94.17**	10843 (30-0.2)	**94.04**

To understand where such an enhancement comes from, Figure 8 illustrates the decision boundary during five epochs Figures 8(a)-(e) and Figure 9 shows loss Figure 9(a) and accuracy Figure 9(b) at various epochs during the neural network training on D12 dataset. For this dataset, the network learned much slower on original label and mostly struggled to obtain improvement, which can be observed in Figure 9(b). The accuracy of the original label has a larger fluctuation at around 5% between epochs, which is considerably higher than using a subclass. Figure 9(a) also displays that through all epochs, it decreases only a tiny amount of loss. Although at the start, the loss for DCGC is higher than Original, it is because DCGC begins with 64 classes and only 1/64 chance to predict correct classes as compared to 1/2 from original. Splitting to more classes in this way also makes the network easier to track the progress, that it continues to shift the decision boundary in the right direction since each hyperplane can focus on correcting the direction to one region of the class label. On the contrary, the network has to obtain all the complex correct adjustments at once when the

same class samples are scattered in multiple clusters on a different part of the data space in which it fails, as confirmed by the tiny improvement of loss and the fluctuation of accuracy in Figure 9. For instance, in Figure 8(b) from original, the decision boundary A and B are responsible for samples in areas C and D, respectively. If areas C and D are the same class, then training a neural network with samples in C will shift A and B slightly toward the top-left corner. At the other end, feeding samples in D will adjust A and B toward the bottom-right. This causes both A and B to be back at the start and make no progress. In contrast, if C and D are different classes, samples in C will only affect A and samples in D will only affect B. These are evidence that the original label likely causes the network to be struck at a local optimum and with DCGC, the neural network gains a massive increase in performance.

Another improvement comes from the better label representation in which DCGC outperforms other subclassing algorithms. Figure 10 visualizes the clustering subclass and prediction boundary from neural networks. Due to the limited space, only D13 is selected since it is the most clear. Figure 10(a) displays the original class data, and Figures 10(b)-(d) show subclass results from SCEC, K-Means and DCGC, respectively. In Figure 10(e), the hyperplane from a neural network train with original class does not has any hint of forming a spiral shape. In contrast, the ones train with subclasses Figures 10(f)-(h) all have a resemblance with moderate faulty prediction in A2, B2, and D3. These occur because SCEC has wrong representative points and K-Means has centroid on other class regions. It creates a situation where many clusters of the same class are separated by clusters of other classes A1, B1, and C1, making it harder for the neural network to place decision boundaries. Although DCGC does not completely eliminate all cases, it has low enough of those type of clusters that make neural networks almost achieve perfect boundary, as shown in Figure 10(h).

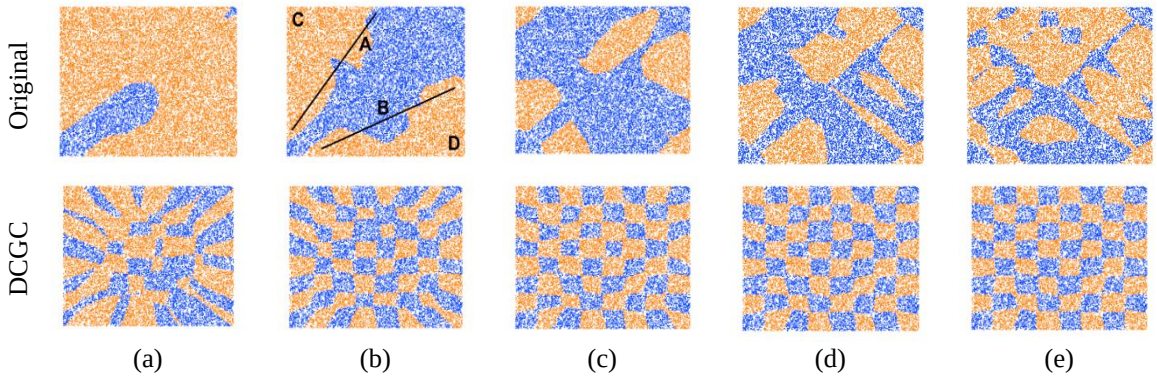

Figure 8. Compare decision boundary for D12 dataset with 10-neural network at (a) 4, (b) 10, (c) 30, (d) 50, and (e) 100 epochs between original and DCGC with $\kappa = 50$, $\gamma = 0.2$

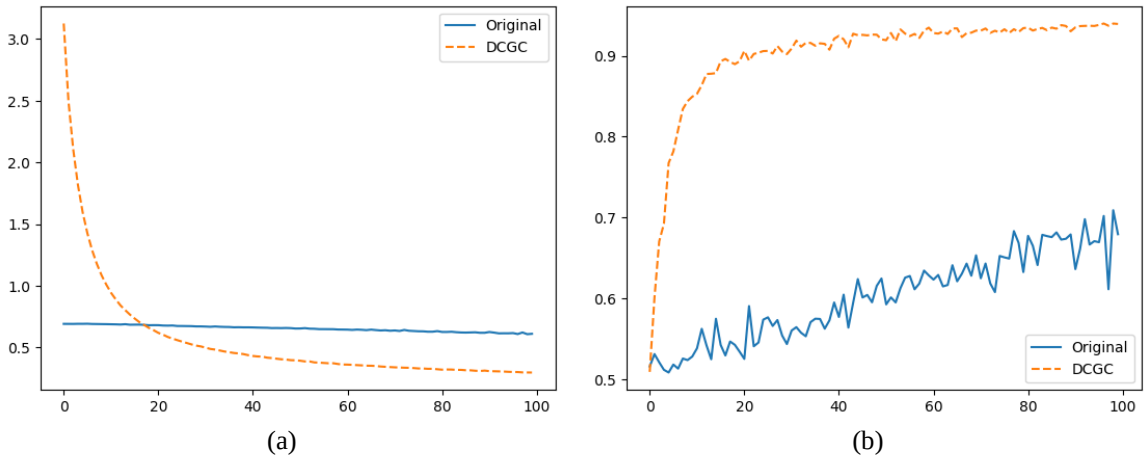

Figure 9. Comparing classification performance at each epoch (x-axis) for D12 dataset with 10-neural network trained using original label and label from DCGC with $\kappa = 50$, $\gamma = 0.2$ for (a) loss and (b) accuracy

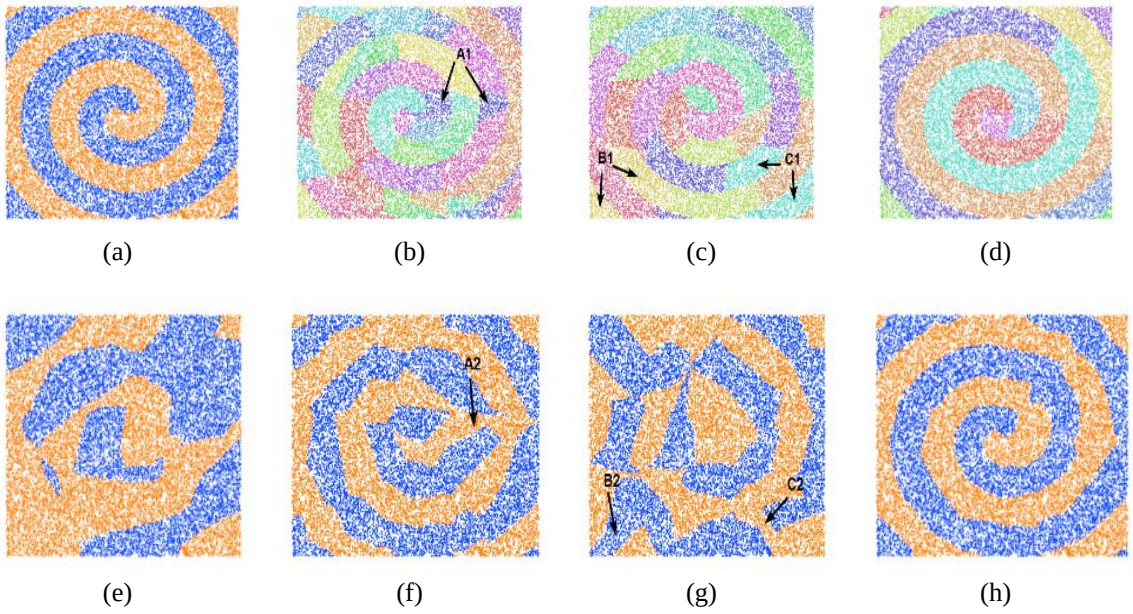

Figure 10. Subclass results where DCGC uses κ = 50, γ = 0.2 and prediction of 10-neural network for D13 dataset, (a) original, (b) SCEC, (c) K-Means, (d) DCGC, (e) original – predict, (f) SCEC – predict, (g) K-Means – predict, and (h) DCGC – predict

3.2. Experiment 2 - UCI dataset

To explore the practicality of DCGC algorithm, we employed five real-world public benchmark datasets from UCI machine learning repository. They were Avila (D21), Avila Reduced (D22), Letter (D23), Pendigits (D24), and Sensorless Drive (D25). The Avila Reduced is the same as Avila dataset, except only Feature 1, 4, 5 are selected. Furthermore, only the integer/float features from the five datasets were used. All the category features were ignored since they were irrelevant to Euclidean distance and required a particular distance function to handle them. Table 3 summarizes all the characteristics of the five datasets that were experimented with in this section.

Table 4 displays subclass results, it gives one insight indicating that D24 and D25 datasets have apparent clusters because the number of subclasses generated in each combination of κ and γ are pretty the same. While the number of clusters for D25 is a potentially suitable subclass, D24 indicates a well-distributed original class because the number of clusters from DCGC is close to the number of original classes. This represents that the subclass obtained from D24 will have the identical classification performance to the original class another finding is the considerable number of clusters on D21 and D22 when on small κ, showing that both datasets may have significant noise samples.

Table 3. The characteristic of real-world datasets

	Name	Train	Test	Features	Classes	Source
D21	Avila	16,693	4,174	10	12	[32]
D22	Avila Reduced	16,693	4,174	3	12	[32]
D23	Letter	16,000	4,000	16	26	[33]
D24	Pendigits	7,494	3,498	16	10	[33]
D25	Sensorless Drive	46,807	11,702	18	11	[33]

Table 4. Real - Number of subclasses created with different algorithm

			DCGC and K-Means									
	Original	SCEC	10		30		50		100		200	
			0.2	0.35	0.2	0.35	0.2	0.35	0.2	0.35	0.2	0.35
D21	12	39	300	244	91	81	51	48	24	23	18	18
D22	12	32	303	288	156	136	100	87	61	56	32	30
D23	26	168	91	70	79	69	54	47	43	38	29	27
D24	10	52	13	13	14	14	14	14	12	12	11	11
D25	11	115	66	62	71	67	73	67	69	62	64	61

The best classification result for real-world datasets is depicted in Table 5. The table shows that performing class decomposition helps boost accuracy in only two datasets, D22 and D25. For D23 and D24, The classification accuracies of K-Means and DCGC are about the same as the original class. Furthermore, the amount of subclasses used to obtain those accuracies is almost identical to the number of original classes. These indicate that class decomposition does not help on these datasets. Moreover, SCEC performs even worst than K-Means and DCGC as it significantly degrades classification performance on most datasets (D21-D24). It demonstrates that class decomposition is no panacea and that flawed label representation can lead to a substantial performance loss. On the positive side, class decomposition slightly improves accuracy on D25, although only on a small neural network. The gain is likely because D25 dataset only has a moderately complex class distribution and increasing neural network complexity helps guide those hyperplanes. As evidenced from Table 5, the boost subclasses gain over original classes is reduced when the number of neurons increases. A more notable improvement is when applying class decomposition to D22. It results in a more than 10% boost in accuracy for the 200-neural network compared to the original class. The improvement here is also a surprising discovery because the accuracy from DCGC and K-Means of D22 is even higher than the best accuracy of D21 even though it is the same dataset and D22 uses less features. These results support a well-known principle that using more features does not necessarily lead to improvement in classification accuracy and why sometimes feature selection is needed. In some circumstances, an improved label representation can be very effective in helping improve classification accuracy. As for D21 and D22 cases, since the original source [32] used different feature subsets for each class, it is logical that using all features makes it more complicated.

Table 5. Real - summary of classification result

		Number of neurons in hidden layer for base case							
		10		50		100		200	
		Params	Acc	Params	Acc	Params	Acc	Params	Acc
D21	Original	518	**62.31**	2519	**66.14**	5026	67.75	51233	**70.92**
	SCEC	539	47.24	2539	60.60	5039	63.97	51239	68.43
	K-Means	533	58.62	2512	66.07	5035	67.79	51244	67.66
	DCGC	533 (100-0.35)	60.36	2512 (200-0.2)	65.85	5035 (200-0.2)	**67.85**	51244 (10-0.35)	70.00
D22	Original	3372	60.24	15644	61.52	30988	63.29	61692	63.71
	SCEC	3344	49.31	15620	50.01	30992	50.27	61700	50.15
	K-Means	3373	**60.74**	15653	67.09	31003	69.13	61703	72.64
	DCGC	3373 (10-0.2)	57.69	15653 (10-0.2)	**67.17**	31003 (10-0.2)	**70.10**	61703 (10-0.2)	**74.43**
D23	Original	2004	**89.01**	9400	**94.89**	18645	**95.59**	37135	95.82
	SCEC	2018	79.24	9418	88.23	18668	90.60	37168	92.03
	K-Means	2007	88.76	9399	94.66	18639	95.53	37163	**95.84**
	DCGC	2007 (200-0.35)	88.83	9399 (200-0.35)	94.87	18639 (200-0.35)	95.53	37163 (200-0.35)	95.76
D24	Original	739	96.63	3493	97.34	6949	97.32	13834	97.38
	SCEC	742	95.15	3502	96.24	6952	96.30	13852	96.47
	K-Means	739	96.63	3486	97.43	6927	97.38	13840	97.44
	DCGC	739 (200-0.2)	**97.07**	3486 (30-0.35)	**97.44**	6927 (30-0.35)	**97.55**	13840 (30-0.2)	**97.47**
D25	Original	1751	95.81	8291	98.17	16511	98.54	32891	98.93
	SCEC	1755	97.02	8315	98.18	16515	98.65	32915	98.61
	K-Means	1751	98.46	8247	99.36	16490	99.52	32895	99.42
	DCGC	1751 (30-0.2)	**99.16**	8247 (50-0.2)	**99.50**	16490 (100-0.35)	**99.58**	32895 (50-0.35)	**99.50**

4. CONCLUSION

In this paper, a method to improve classification accuracy via class decomposition is proposed. The DCGC algorithm creates representative core samples and forms skeleton graphs using density, which shows that it is more effective than the current supervised clustering in producing a less complex subclass. The cut graph part is also confirmed to be a better separation criterion since it ensures that the regions between representative samples contain only samples belonging to the same class, unlike performing class decomposition using traditional clustering algorithms. The technique has outperformed both traditional clustering and supervised clustering algorithms in improving classification accuracy on six syntactic datasets. Especially when the dataset has many clusters from the same class distributed throughout data space, the improvement can be as significant as 30%. However, when datasets have a simple shape, DCGC may not lead to an improved classification accuracy. For the real-world datasets, utilizing DCGC was found to boost classification accuracy for some datasets. However, many real-world datasets may not benefit from DCGC because their features are already exhaustive, or the inherent simple correlations among features may not form scattered clusters. An exhaustive set of features usually help simplify the shape of the data, making the classifier face an easier training task. Although that is not always the case, as apparent in the reduced features on one of

the real-world datasets, when applied DCGC, it is able to obtains a massive 10% increase in accuracy. Aside from the characteristics of the datasets, the required κ and γ parameters for DCGC algorithm are also not easy to select as initially thought they would be. Although they are much more robust than selecting the number of clusters, finding such optimum parameter values depend upon the dataset and need to be determined on a case by case basis As an introductory version, the current DCGC algorithm assumes that κ and γ are identical for every original class which in practice is not always the case.

A possible direction for future work is to look into creating a method to determine κ and γ. This particular task will not only solve the problem of having the same κ and γ for all classes, but it will also eliminate the need to specify parameters. Another interesting investigation is to develop a procedure to determine the dataset's characteristics for class decomposition. With such a procedure, the dataset can quickly be determined whether to proceed to class decomposition instead of blind guessing.

REFERENCES

[1] A. Hernández-García and P. König, "Data augmentation instead of explicit regularization," *arXiv preprint arXiv*, 2018, [Online]. Available: http://arxiv.org/abs/1806.03852.

[2] P. Fuangkhon and T. Tanprasert, "Reduced multi-class contour preserving classification," *Neural Process. Lett.*, vol. 43, no. 3, pp. 759–804, 2016, doi: 10.1007/s11063-015-9446-1.

[3] X. Ying, "An overview of overfitting and its solutions," *J. Phys. Conf. Ser.*, vol. 1168, no. 2, 2019, doi: 10.1088/1742-6596/1168/2/022022.

[4] Y. Mo, S. D. Scott, and D. Downey, "Learning hierarchically decomposable concepts with active over-labeling," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 340–349, 2017, doi: 10.1109/ICDM.2016.165.

[5] Z. Chen, R. Ding, T. W. Chin, and D. Marculescu, "Understanding the impact of label granularity on CNN-based image classification," *IEEE Int. Conf. Data Min. Work. ICDMW*, 2019, pp. 895–904, doi: 10.1109/ICDMW.2018.00131.

[6] R. Vilalta, M. K. Achari, and C. F. Eick, "Class decomposition via clustering: A new framework for low-variance classifiers," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, 2003, pp. 673–676, doi: 10.1109/icdm.2003.1251005.

[7] F. Ocegueda-Hernandez and R. Vilalta, "An empirical study of the suitability of class decomposition for linear models: When does it work well?," *Proc. 2013 SIAM Int. Conf. Data Mining, SDM 2013*, vol. 2, 2013, pp. 432–440, doi: 10.1137/1.9781611972832.48.

[8] A. Rida, A. Labbi, and C. Pellegrini, "Local experts combination through density decomposition," *Proc. Seventh Int. Work. Artif. Intell. Stat.*, pp. 130–136, 1999, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.8371.

[9] S. Godbole, "Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers," *Prog. Report, IIT Bombay*, p. 17, 2002, [Online]. Available: http://www.it.iitb.ac.in/~shantanu/work/report.pdf.

[10] A. Hoffmann, R. Kwok, and P. Compton, "Using subclasses to improve classification learning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2167, pp. 203–213, 2001, doi: 10.1007/3-540-44795-4_18.

[11] Y. Luo, "Can subclasses help a multiclass learning problem?," *IEEE Intell. Veh. Symp. Proc.*, pp. 214–219, 2008, doi: 10.1109/IVS.2008.4621136.

[12] F. Abramovich and M. Pensky, "Classification with many classes: Challenges and pluses," *J. Multivar. Anal.*, vol. 174, p. 104536, 2019, doi: 10.1016/j.jmva.2019.104536.

[13] N. Japkowicz, "Supervised learning with unsupervised output separation," *Int. Conf. Artif. Intell. Soft Comput.*, 2002, pp. 1–5, [Online]. Available: http://www.sige.uottawa.ca/~nat/Papers/new-iasted-final.pdf.

[14] J. Wu, H. Xiong, and J. Chen, "COG: Local decomposition for rare class analysis," *Data Min. Knowl. Discov.*, vol. 20, no. 2, pp. 191–220, 2010, doi: 10.1007/s10618-009-0146-1.

[15] D. Fradkin, "Clustering inside classes improves performance of linear classifiers," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, vol. 2, 2008, pp. 439–442, doi: 10.1109/ICTAI.2008.29.

[16] R. Janning, C. Schatten, and L. Schmidt-Thieme, "Automatic subclasses estimation for a better classification with HNNP," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8502 LNAI, pp. 93–102, 2014, doi: 10.1007/978-3-319-08326-1_10.

[17] E. Elyan and M. M. Gaber, "A fine-grained random forests using class decomposition: an application to medical diagnosis," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2279–2288, 2016, doi: 10.1007/s00521-015-2064-z.

[18] Z. S. Abdallah and M. M. Gaber, "KB-CB-N classification: Towards unsupervised approach for supervised learning," *IEEE SSCI 2011 Symp. Ser. Comput. Intell. - CIDM 2011 2011 IEEE Symp. Comput. Intell. Data Min.*, pp. 283–290, 2011, doi: 10.1109/CIDM.2011.5949435.

[19] I. Polaka and A. Borisov, "Clustering-based decision tree classifier construction," *Technol. Econ. Dev. Econ.*, vol. 16, no. 4, pp. 765–781, 2010, doi: 10.3846/tede.2010.47.

[20] S. Banitaan, A. B. Nassif, and M. Azzeh, "Class decomposition using K-means and hierarchical clustering," *Proc. - 2015 IEEE 14th Int. Conf. Mach. Learn. Appl. ICMLA 2015*, 2016, pp. 1263–1267, doi: 10.1109/ICMLA.2015.169.

[21] C. F. Eick, N. Zeidat, and Z. Zhao, "Supervised clustering - Algorithms and benefits," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, 2004, pp. 774–776, doi: 10.1109/ICTAI.2004.111.

[22] A. Jirayusakul and S. Auwatanamongkol, "A supervised growing neural gas algorithm for cluster analysis," *Int. J. Hybrid Intell. Syst.*, vol. 4, no. 2, pp. 129–141, 2016, doi: 10.3233/his-2007-4205.

[23] V. Thananant and S. Auwatanamongkol, "Supervised clustering based on a multi-objective genetic algorithm," *Pertanika J. Sci. Technol.*, vol. 27, no. 1, pp. 81–122, 2019.

[24] M. Z. Hossain, M. N. Akhtar, R. B. Ahmad, and M. Rahman, "A dynamic K-means clustering for data mining," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 13, no. 2, pp. 521–526, 2019, doi: 10.11591/ijeecs.v13.i2.pp521-526.

[25] A. F. N. Alrammahi and K. B. S. Aljanabi, "A new approach for improving clustering algorithms performance," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 20, no. 3, pp. 1569–1575, 2020, doi: 10.11591/ijeecs.v20.i3.pp1569-1575.

[26] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science (80-.).*, vol. 344, no. 6191, pp. 1492–1496, 2014, doi: 10.1126/science.1242072.

[27] G. Wang and Q. Song, "Automatic clustering via outward statistical testing on density metrics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4347, no. c, pp. 1–14, 2016, doi: 10.1109/TKDE.2016.2535209.

[28] Z. H. I. Liu, C. Wu, Q. Peng, and J. I. A. Lee, "Local peaks-based clustering algorithm in symmetric neighborhood graph," *IEEE Access*, vol. 8, pp. 1600–1612, 2020, doi: 10.1109/ACCESS.2019.2962394.
[29] S. Zhou, Z. Xu, and F. Liu, "Method for determining the optimal number of clusters based on agglomerative hierarchical clustering," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 12, pp. 3007–3017, 2017, doi: 10.1109/TNNLS.2016.2608001.
[30] A. E. Danganan, A. M. Sison, and R. P. Medina, "OCA: Overlapping clustering application unsupervised approach for data analysis," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 14, no. 3, pp. 1471–1478, 2019, doi: 10.11591/ijeecs.v14.i3.pp1471-1478.
[31] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975, doi: 10.1145/361002.361007.
[32] C. D. Stefano, M. Maniaci, F. Fontanella, and A. S. di Freca, "Reliable writer identification in medieval manuscripts through page layout features: The 'Avila' Bible case," *Eng. Appl. Artif. Intell.*, vol. 72, no. October 2016, pp. 99–110, 2018, doi: 10.1016/j.engappai.2018.03.023.
[33] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases," *Univ. Calif.*, 1998.

BIOGRAPHIES OF AUTHORS

Eakawat Tantamjarik is a doctoral candidate in Computer Science at Vincent Mary School of Science and Technology, Assumption University of Thailand. He received his bachelor degree in Computer Science from King Mongkut's Institute of Technology Ladkrabang in 2012, the master degrees in Computer science from Assumption University of Thailand in 2016. His research interest is data science and neural network computation. He can be contacted at email: p5929444@au.edu.

Thitipong Tanprasert is an Assistant Professor in Computer Science at Vincent Mary School of Science and Technology, Assumption University of Thailand, where he is also directing the Intelligent Data Analytics Research Laboratory. He received his bachelor degree in Electrical Engineering from Chulalongkorn University in 1987, the master and doctor of philosophy in Computer Engineering degrees from University of Louisiana at Lafayette in 1989 and 1993, respectively. His research interests are in neural network computation, data science, and machine learning applications. He can be contacted at email: thitipong@scitech.au.edu.