Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 354 (2020) 45–60

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Weighted Complete Graphs for Condensing Data

A. Guzmán-Ponce [1] J. Raymundo Marcial-Romero [2]
R.M. Valdovinos-Rosas [3]

Universidad Autonóma del Estado de México, Facultad de Ingeniería
Toluca, Estado de México, México

J.S. Sánchez-Garreta[4]

Institute of New Imaging Technologies, Department of Computer Languages and Systems
Universitat Jaume I, Castelló de la Plana, Spain

Abstract

In many real-world problems (such as industrial applications, chemistry models, social network analysis, among others), their solution can be obtained by transforming the problem in terms of vertices and edges, that is to say, using graph theory. Data Science applications are characterized by processing large volumes of data, in some cases, the data size can be higher than the resources for their processing, situation that makes prohibitive to use the traditional methods. In this way, to develop solutions based on graphs for condensing data can be a good strategy for handling big datasets. In this paper we include two methods for condensing data based on graphs, the two proposals consider a weighted complete graph by acquiring an induced subgraph or a minimum spanning tree from the whole datasets. We conducted some experiments in order to validate our proposals, using 24 benchmark real-datasets for training the 1NN, C4.5, and SVM classifiers. The results prove that our methods condensed the datasets without reducing the performance of the classifier, in terms of geometric means and the Wilcoxon's test.

Keywords: Weighted Graph, Induced Subgraph, Minimum Spanning Tree, Condensed data, Data Science.

1 Introduction

Nowadays a promised area for extract knowledge from data is called Data Science. Data Science is a multi-disciplinary approach for finding, extracting, and discovering patterns in data through several methods from data mining, deep learning,

[1] Email: mailto:angelicagp1416@hotmail.com
[2] Corresponding author Email: mailto:jrmarcialr@uaemex.mx
[3] Email: mailto:rvaldovinosr@uaemex.mx
[4] Email: mailto:sanchez@uji.es

https://doi.org/10.1016/j.entcs.2020.10.005

forecasting, machine learning, optimization, predictive analytics, and statistics, between others [3]. The main problem in Data Science is the data volume, in some cases, it can be so great that is prohibitive to apply traditional statistics methods and even some machine learning or data mining algorithms. For this reason, it is necessary to have strategies for obtaining a representative subset of the data with which the classification or prediction algorithms do not affect their performance.

Graph theory is becoming a popular technique in other areas of science for giving solution in real-world problems. For example, in chemistry a graph can represent the topology of a molecule, in physics, a graph can be used for describing the grade of thermodynamic stability, on electrical engineering the graph theory can be applied over configuration of antennas and their frequencies, in urban planning can be used for programing bus paths or the traffic lights, and many others optimization problems can be solved by graph theory because it is a way of knowledge abstraction, thus allowing get reliable solutions [13,16,8,6].

In Data science are several approaches based on graphs, where the main idea is based on extracting knowledge from graph topologies. Newman et al. [11] proposed an approach for clustering communities through iterative remove edges from a graph, the algorithm gives more weight to the borderline edges between communities than the edges inside it. Their results showed that it is possible to extract community structures from networks artificially generated using information from known communities.

Zhang and Hancock [18] proposed a graph-based method to feature selection. They used weight graphs, where a vertex represents a feature and their pairwise relationship is an edge, the weight of an edge was given by the degree of relevance between two features. Their proposal uses the multidimensional interaction information criterion for feature selection, this criterion detects the relationships between feature combinations with greater relevance. Therefore, the proposal gets the most information that a class has.

On the other hand, Maillo et al. [10] proposed a fuzzy kNN approach based on Hybrid Spill-Tree, where through a tree structure the nearest-neighbors are approximated. This method has two types of binary trees: a Metric-Tree and a Spill-Tree. The Metric-tree organizes the instances of a dataset in a spatial hierarchical manner, where its root vertex represents all instances, and each vertex represents an instance. An internal vertex is partitioned into two subsets, who are disjoint sets, i.e they do not have repeated instances. While, the Spill-Tree is similar to a Metric-Tree, with the difference that Spill-Tree enables repeated instances.

In this paper, we introduce two methods based on graph theory, that using a weighted complete graph for condensing data. The main contributions can be summarized as follows:

- We propose the use of graph theory to obtain an induced subgraph, which allows getting the borderline of the classes, and build a Minimum Spanning Tree (MST) which include the core of the data.
- Depending on the balance between the classes included in the data, the number

of representative instances that will be considered by the induced subgraph and the MSP is computed.

- The experimentation shows that when the condensed datasets obtained with our methods, the classifier do not diminish their performance, in fact, it is increased.

The remainder of the paper is structured as follows. In Section 2 we review the main definitions used in the paper. Details about the two proposals are detailed in Section 3. While in Section 4 we summarized the time complexity required by the methods proposed. The experimental setup is reported in Section 5. After that, in Section 6 the experimental results are explained. Finally, in Section 7 the main concluding remarks and some open lines are exposed.

2 Preliminaries

Let $G = (V, E)$ be an undirected simple graph (i.e loop-less and without parallel edges) consisting of a nonempty vertex set V and a set of edges E, where each edge is a non-order pair of vertices, denoted as $\{u, v\}$. Two vertices v and v are named *adjacent* if there is an edge $\{u, v\} \in E$, joining them. A *complete graph* is a simple graph in which any two vertices are adjacent.

The *neighbourhood* of a vertex v in a graph $G = (V, E)$ is $N(v) = \{\forall u \in V \mid \{v, u\} \in E\}$, i.e $N(v)$ is the set of all vertices adjacent to v without itself and its *closed neighbourhood* when $N(v) \cup v$, which is denoted as $N[v]$. Note that v is not in $N(v)$, but it is in $N[v]$.

A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Let $X \in V(G)$ a set of vertices deleted, the induced subgraph is denoted by $G - X$; if $Y = V \setminus X$ represents the set of vertices that remain undeleted, the induced subgraph is denotes by $G[Y]$ and referred to as subgraph of G induced by Y, where Y is the set of vertices of G and whose set of edges consist of all edges of G that have both ends in Y.

A path from a vertex v to a vertex u in a graph is a sequence of edges: $v_0v_1, v_1v_2, \cdots, v_{n-1}v_n$, such that $v = v_0, v_n = u, v_k$ is adjacent to v_{k+1} and the length of the path is n. A simple path is a path such that $v_0, v_1, \cdots, v_{n-1}$ are all distinct. A cycle is just a nonempty path such that the first and last vertices are identical, and a simple cycle is a cycle in which no vertex is repeated, except the first and last vertices.

A *connected graph* is a graph $G = (V, E)$ if every pair of vertex in G has a path between them. If the graph is not connected, each maximal connected piece is named a *component*.

A weighted graph $G_w = (V, E)$, is a graph where each edge $e \in E$ let there be associated a real number $w(e)$, called its weight. The adjacency matrix of a weighted graph G_w is a $V \times V$ matrix $M_G = (w_v u)$ where each element (v_i, v_j) contains the weight $w(e)$ assigned to the edge $e = v_i, v_j$ or 0 according to wheter the vertices v_i and v_j are adjacent or not in the graph.

If H is a subgraph such that $H \subset G_w$ the weight $w(H)$ of the H is the sum

of the weights $\sum w(e)$ on its edges. A subgraph of a certain type with minimum (or maximum) weight, is a graph which path of minimum (or maximum) weights connecting two specified vertices u_0 and v_0. A *Minimum Spanning Tree* (MST) is an induced subgraph, where the set of edges connects all vertices, without any cycles and the sum of its edge weights are the minimal.

3 Proposed methods for condensing datasets

In this section, we present two methods based on graphs for condensing data. The proposals start by dividing a two-class dataset with n instances into two subsets denoted as C^- and C^+, instances from negative class and positive class respectively. The C^+ commonly is the most important class to identify and it is less represented respect with another class or classes. Our proposals work only over the negative class, in order to diminish the cardinality of it until it can be similar to C^+. For that, from the data included in C^- we building of a weighted complete graph before generating an induced subgraph or a Minimum Spanning Tree, which aims purpose is obtaining a borderline or a core of negative class respectively.

The Algorithm 1 describes the general procedure. The proposals build a weighted complete graph (GraphProcedure in the Algorithm 1) to generate an induced subgraph (Section 3.1) or a Minimum Spanning Tree (Section 3.2). Note that, IRm is a desirable imbalance ratio, that is, the desirable ratio of the positive class size to the negative class size.

Algorithm 1 Condensed data

Require: $DS = \{p_1, p_2, \ldots, p_n\}$, IRm
Ensure: DS' condensed data set
1: Split DS into two subsets C^- and C^+.
2: $C'^- \leftarrow$ GraphProcedure(C^-, IRm, C^+)
3: $DS' = C^+ \cup C'^-$

Given a dataset DS formed by n-instances with m-features, each instance p_n is a tuple $(f_{n,1}, f_{n,2}, \ldots, f_{n,m}, \omega)$, where, f_m is the value of the $m-th$ feature of a instance p_n. This instance belongs to a class ω. Graphs are used to model practical problems and get optimal solutions, thus our proposals use a graph-based method to obtain a subset C'^-. We consider the set C^- as a weighted complete graph denoted as G_w, and it is built as follows:

- $V(G) = \{\forall p_i \in C^- \mid i \in V(G)\}$ the set of vertex.
- $E(G) = \{\{v, u\} \mid v, u \in V(G)\}$ the set of edges.
- $\forall e = \{v, u\} \in E(G)$, $w(e) = d(p_v, p_u)$ where $d(p_v, p_u)$ is the euclidean distance between v and u.

3.1 Induced Graph Under-Sampling (IG-US)

IG-US is a proposed method to get an induced subgraph, which aims is to keep borderline instances, i.e instances that are further away from each other. The IG-US proposal described in the Algorithm 2 condense the negative class through

generating an induced subgraph of G_w, which set of vertices will have size according to the equation 1. The size will be updated and stored in the variable *Sample* until the algorithm obtains an imbalance ratio (IR [5]) equal to $maxIR$ (lines 2-7 in Algorithm 2).

$$f(x) = \begin{cases} \frac{|C_1^-| \quad \sigma^2 Z^2}{e^2(|C_1^-|-1)+\sigma^2 Z^2} & \text{if } x = 1 \\ \\ \frac{f(x-1) \quad \sigma^2 Z^2}{e^2(|C_1^-|-1)+\sigma^2 Z^2} & \text{otherwise} \end{cases} \tag{1}$$

where $Z = 1.96$, $\sigma = 0.5$ and $e = 0.05$ for a confidence level of 95%.

Once the size of the negative class subset is obtained, the Algorithm 2 computes the adjacency matrix of G_w (line 9) by the distance between each pair of instances (vertices). The function *GetMaximum* ensures a set of edges (pairs of instances), which have a high distance according to the adjacency matrix (line 11). For each edge in this set, the vertices that have not been visited yet are added into C_2^- to be marked as visited (lines 12-20). In the end, the set C_2^- is the condensed dataset from the negative class.

Algorithm 2 IG-US algorithm

Require: C_1^-, R_{max}, C^+
Ensure: C_2^- a condensed dataset
1: Build $G_w = (V, E)$ a weighted graph with $V = C_1^-$
2: $Sample \leftarrow |C_1^-|$
3: $IR \leftarrow \frac{Sample}{|C^+|}$
4: **while** $IR > maxIR$ **do**
5: $\quad Sample \leftarrow \frac{Sample \quad \sigma^2 Z^2}{e^2(|C_1^-|-1)+\sigma^2 Z^2}$
6: $\quad IR \leftarrow \frac{Sample}{|C^+|}$
7: **end while**
8: $C_2^- \leftarrow []$
9: $M_G \leftarrow$ adjacencyMatrix(G_w)
10: **while** $|C_2^-| < Sample$ **do**
11: $\quad Maximum \leftarrow$ GetMaximum(M_G)
12: $\quad$ **for all** $\{v, u\}$ in $Maximum$ **do**
13: $\quad\quad$ **if** v has not been visited **then**
14: $\quad\quad\quad$ Mark v as visit
15: $\quad\quad\quad C_2^- \leftarrow C_2^- \cup \{v\}$
16: $\quad\quad$ **end if**
17: $\quad\quad$ **if** u has not been visited **then**
18: $\quad\quad\quad$ Mark u as visited
19: $\quad\quad\quad C_2^- \leftarrow C_2^- \cup \{u\}$
20: $\quad\quad$ **end if**
21: $\quad\quad$ Remove $\{v, u\}$ from M_G
22: $\quad$ **end for**
23: **end while**
24: **return** C_2^-

3.2 Minimum Spaning Tree Under-Sampling (MIST-US)

We will denote the second proposal as MIST-US, this approach builds a minimum spanning tree from a weighted complete graph. In general, the Algorithm 3 finds a

[5] represent the difference between the cardinality of C^- and C^+

representative core of the negative class, trough a MST. The goal of using MST is to build a subgraph whose sum of edge weights is as small as possible, therefore the dispersion of the condensed negative class is less.

Algorithm 3 MIST-US algorithm

Require: C^-, IRm, C^+
Ensure: $C'^- \subset C^-$ a condensed dataset
1: Build $G_w = (V, E)$ a complete graph from C^-.
2: $S \leftarrow |C^-|$
3: $IR \leftarrow \frac{S}{|C^+|}$
4: $C'^- \leftarrow []$
5: $M_G \leftarrow$incidenceMatrix(G_w)
6: $MST \leftarrow$GetMST(G_w, M_G)
7: **while** $IR > IRm$ **do**
8: $\quad S \leftarrow \frac{S \quad \sigma^2 Z^2}{e^2(|C^-|-1)+\sigma^2 Z^2}$
9: $\quad IR \leftarrow \frac{S}{|C^+|}$
10: **end while**
11: **for all** $\{v, u\}$ in $E(MST)$ **do**
12: $\quad C'^- \cup u$
13: $\quad C'^- \cup v$
14: $\quad$ **if** $|C'^-| > S$ **then**
15: $\quad\quad$ **return** C'^-
16: $\quad$ **end if**
17: **end for**

In our proposal, firstly an incidence matrix M_G is built before process the*GetMST* method which ensures a MST. *GetMST* requires the weight graph G_w and its incidence matrix to process the Prim's [12] algorithm, described as follows:

(i) Choose a vertex v, this is the start vertex.

(ii) Select the edge $e = \{v, u\}$ with the lower weight in M_G, mark v as visited. Now, the next vertex to be analyzed is u.

(iii) Repeat step 2 whenever the chosen edge links a vertex that has been visited and others that have not been visited, provided any cycles.

(iv) The MST will be building until all vertices have been visited.

Although by definition a MST contains all the vertices of a graph, in our proposal, the condensed negative class is built by only the first S-vertices from the edge set of the MST. MIST-US algorithm finishes when the size of C'^- is greater than S. The S value is the representative amount of instances from negative class, and it is computed by equation 1 (lines 7-10).

4 Time complexity

Time complexity is estimated counting the number of operations performed by an algorithm [2], thus the worst-case is used to measure the time complexity. In our proposals, IG-US and MIST-US time complexity is described as follows:

IG-US algorithm has a time complexity of $\mathcal{O}(n^2)$. In this case, the complexity is mostly governed by the following set of instruction:

- Updating *Sample* and IR (lines 4-7) has a complexity of n.
- Computing the adjacency matrix (line 9) has a complexity of n^2.

- Generating a majority class subset of size *Sample* (lines 10-23) has a complexity of n^2, which comes from n iterations to compute the set $Maximum$ with a complexity of n.

Thus, the time complexity of IG-US algorithm becomes $\mathcal{O}(n^2)$ in the worst-case. In similar way, MIST-US is computed on $\mathcal{O}(n^2)$ too, because of:

- The incidence matrix from G_w in the worst case over n^2 is computed.
- Get a Minimum Spanning Tree, based on Prim's algorithm, in the worst case takes n^2 because it is necessary to visit all edges with an endpoint that has not been visited in the incidence matrix M_G with the lower weight.

5 Experimental set-up

In order to validate the methods proposed, we conducted an experimental study designed over 24 real-problem datasets with several imbalanced ratio (IR) i.e. $IR > 9$. The next sections describe the datasets used, the metrics employed for measured the methods performance and the classifiers used to validate our proposals.

5.1 Datasets

The experiments were carried out on 24 two-class data sets taken from the KEEL Data Set Repository (https://sci2s.ugr.es/keel/imbalanced.php#subA), each dataset has different distribution between the classes i.e IR. Table 1 summarizes the main characteristics of the datasets used in the experiments, the data sets are sorted on an increase in IR value. The IR is obtained by dividing the number of patterns included in C^+ between the number of patterns of C^-. As we can see in Table 1, the range of IR is around more than 9 to 82, which indicates a high imbalance ratio.

All datasets have been partitioned using 10-fold cross-validation in order to avoid biased results [5]. Each original data set was randomly divided into 10 stratified parts, for each fold, 9 blocks have been used as a training set, and the remaining portion was used as a test set. We evaluated our methods, comparing their benefit in contrast with the most popular methods for condensing data included in the literature (see Table 2).

5.2 Evaluation metrics

The performance of the condensing methods was tested with a Decision Tree classifier (C4.5), the 1-Nearest Neighbor classifier (1-NN) and a Support Vector Machine (SVM) classifier with all default values of the WEKA software. To evaluate the behavior of learning classifier, a confusion matrix like that in Table 3 (for a two-class problem) is usually employed [4]. From this, four simple measures can be directly obtained: TP and TN denote the number of positive and negative cases correctly classified, while FP and FN refer to the number of misclassified positive and negative examples, respectively.

Table 1
A brief summary of the basic characteristics of the database.

	Data Set	Class distribution	#Instances	IR
1	yeast-0-5-6-7-9_vs_4	51 - 477	528	9.35
2	vowel0	90 - 898	988	9.98
3	glass-0-1-6_vs_2	17 - 175	192	10.29
4	glass2	17 - 197	214	11.59
5	shuttle-c0-vs-c4	123 - 1706	1829	13.87
6	yeast-1_vs_7	30 - 429	459	14.30
7	glass4	13 - 201	214	15.47
8	ecoli4	20 - 316	336	15.80
9	page-blocks-1-3_vs_4	28 - 444	472	15.86
10	glass-0-1-6_vs_5	9 - 175	184	19.44
11	shuttle-c2-vs-c4	6 - 123	129	20.50
12	yeast-1-4-5-8_vs_7	30 - 663	693	22.10
13	glass5	9 - 205	214	22.78
14	yeast-2_vs_8	20 - 462	482	23.10
15	flare-F	43 - 1023	1066	23.79
16	yeast4	51 - 1433	1484	28.10
17	yeast-1-2-8-9_vs_7	30 - 917	947	30.57
18	yeast5	44 - 1440	1484	32.73
19	ecoli-0-1-3-7_vs_2-6	7 - 274	281	39.14
20	abalone-17_vs_7-8-9-10	58 - 2280	2338	39.31
21	yeast6	35 - 1449	1484	41.40
22	shuttle-2_vs_5	49 - 3267	3316	66.67
23	kdd-buffer_oflow_vs_back	30 - 2203	2233	73.43
24	poker-8-9_vs_5	25 - 2050	2075	82.00

Table 2
Condensing methods

Method	Description
RUS	Random elimination of instances from C^-.
CNN	Hart's condensed [7], removes instances from C^- that are far from the decision borderline
TL	Tomek's links [15] remove instances from C^- considering that if two instances form a link, either of them is eliminated.
EUS	*Evolutionary Under-Sampling* [9], remove instances from C^- using a genetic algorithm.
RUSBOOST	This method combine RUS with the AdaBoost algorithm [14].
SBC	Undersampling Based on Clustering [17], divide the dataset into k clusters and randomly selects a number of instances from C^- in each cluster according their IR.

Table 3
Confusion matrix

	Predictive Positive	Predictive Negative
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

When the distribution between classes is not equal in the dataset, the most recommended measure is the geometric mean, because it tries to maximize the accuracy on each of the two classes while keeping these accuracies balanced [1]:

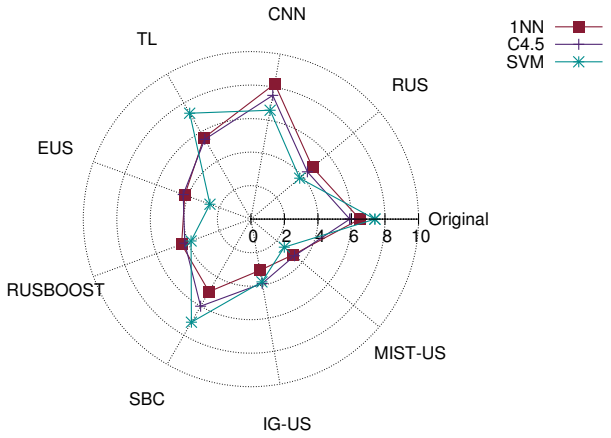

Fig. 1. Average Rank

$g = \sqrt{a^+ \cdot a^-}$, where a^+ indicate the accuracy for the minority class ($\frac{TP}{TP+FN}$) and a^- is the majority class accuracy ($\frac{TN}{TN+FP}$).

6 Analysis of results

In this section, we analyze the results according two ways:

(i) A comparison between the proposed method and the techniques described in the Table 2, using the classifiers 1NN, C4.5 and SVM .

(ii) Influence of the percentage reduction on the behavior of the classifiers 1NN, C4.5 and SVM when the condense datasets are used.

6.1 Comparison on Performance

The experimental results given in Figure 1 correspond to the average Fisher's rank where the best condensing method is the one with the lowest value. These results are obtained by the overall geometric mean, which results are reported in the Appendix A. Results with the original dataset i.e without reduction, were included for comparison purposes.

From the results reported in this section, some preliminary conclusions can be drawn. First, from results in Figure 1, independently the classifier used IG-US and MIST-US get the best performance according to the Fisher's rank. When compare the behavior between the condensed methods, we can see that our methods outperform strategies that are based on neighborhood, such as CNN and TL. Nevertheless, some random methods can obtain competitive results, just as, RUS and RUSBOOTS, whose Fisher's ranks in classifiers as C4.5 and SVM get similar results that ours.

Comparing both proposals (Table A.3), we can observe that IG-US is still superior to MIST-US over 1NN and C4.5 classifiers on 14 and 13 datasets per each classifier, while MIST-US has 6 and 8 datasets, respectively. Only in SVM classifier, MIST-US get better performances with 11 datasets compared to IG-US in 10

datasets.

Finally, we can conclude that get a core of the negative class as a condensing method has better performance (MIST-US) than keeps a borderline from the negative class (IG-US), because MIST-US keeps a representative subset of negative class which is far away from the borderline between classes. According to the results shown in this section, we consider necessary to make a statistical analysis, which is included in the next section.

6.2 Statistic significant analysis

In order to complete the analysis we apply the Wilcoxon's test with a level of confidence of 0.9 and 0.95, in order to identify the statistical significance between the results. Tables 4-6 summarize the Wilcoxon's test, where the upper diagonal is part of the level significance $\alpha = 0.9$, and the lower diagonal to $\alpha = 0.95$. The symbol "•" denote that the method in the row improves the method of the column, otherwise, we used the symbol "○". The last two rows indicate the number of methods where the column method gets statistical significance compared with the rest of the methods according to the significance level.

Table 4
Summary of the Wilcoxon's test using the 1NN classifier.

	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
Original	-	○	•	○	○	○	○	○	○
RUS	•	-	•	•				○	
CNN	○	○	-	○	○	○	○	○	○
TL	•	○	•	-	○	○	○	○	○
EUS	•		•	•	-		•	○	
RUSBOOST	•		•	•		-		○	
SBC	•		•	•	○		-	○	○
IG-US	•	•	•	•	•	•	•	-	•
MIST-US	•		•	•			•		-
$\alpha = 0.9$	1	3	0	2	4	3	3	7	4
$\alpha = 0.95$	1	3	0	2	4	3	3	7	4

In order to conduct the analysis, we take as an example Table 5, focusing on the proposal IG-US, when we compare by pairs, with a significance level of 0.95 (row in bold), there are four methods in which IG-US improves to the other methods (Original, CNN, TL, and SBC). However, with a significance level of 0.90 (column in bold), there are six methods in which IG-US provides statistical significance over the methods: Original, RUS, CNN, TL, EUS, and SBC.

From results in Tables 4-6 it is possible to identify that methods which remove instances considered as redundant, for example, CNN, the performance is lower than the rest of the methods because in any classifier there are elements that improve the results obtained by CNN compared to other methods. Nonetheless, methods such as RUS, TL, RUSBOOST, and SBC show a behavior statistically significantly below our proposals over the 1NN classifier, but with SVM classifier the EUS method has a better statistical significant than IG-US, but MIST-US is significantly superior

Table 5
Summary of the Wilcoxon's test using the C4.5 classifier.

	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	**IG-US**	MIST-US
Original	-	○	●		○	○		○	○
RUS	●	-	●	●			●	○	○
CNN	○	○	-	○	○	○	○	○	○
TL		○	●	-	○	○		○	○
EUS	●		●	●	-		●	○	
RUSBOOST	●		●	●		-	●		
SBC		○	●		○	○	-	○	○
IG-US	●		●	●			●	-	
MIST-US	●		●	●			●		-
$\alpha = 0.9$	1	4	0	1	4	4	1	6	5
$\alpha = 0.95$	1	4	0	1	4	4	1	4	4

Table 6
Summary of the Wilcoxon's test using the SVM classifier.

	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
Original	-	○			○	○		○	○
RUS	●	-	●	●			●		○
CNN		○	-		○	○		○	○
TL		○		-	○	○		○	○
EUS	●		●	●	-	●	●		
RUSBOOST	●		●	●	○	-	●		○
SBC		○			○	○	-	○	○
IG-US	●		●	●			●	-	
MIST-US	●	●	●	●		●	●		-
$\alpha = 0.9$	0	4	0	0	5	4	0	4	6
$\alpha = 0.95$	0	4	0	0	5	4	0	4	6

overall methods. From these results, we can conclude that weighted complete graphs for condensing data are better than another method used in the experiments.

6.3 Influence of percentage reduction

An important aspect that determines if a condensed method is viable or not, is the percentage of IR obtained. Figure 2 plots the average percentage reduction per method, details of the reduction percentages per each method and dataset are in the Appendix B.

From Figure 2, we can remark that:

- IB-US and MIST-US are the methods with the highest IR reduction, it is around 1.0.
- The methods that not have a good percentage of reduction are TL and SBC. The first one removes instances according to achieve the form a link, so in most datasets, it is possible that in the negative class there are not instances that form a Tomek link. While the second method, cluster construction is probably the reason, due to the low density of the negative class.

- Finally, IB-US and MIST-US have the best behavior in terms of geometric mean with datasets condensed with too low IR.

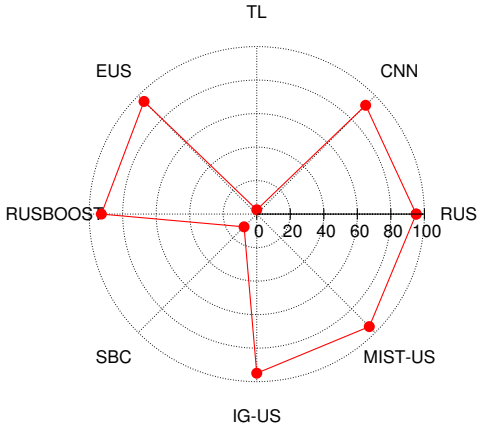

Fig. 2. Average percentage reduction

7 Concluding Remarks

In this work, we propose two new condensing methods called IG-US and MIST-US. The new methods consider a weighted complete graphs for obtaining two structures: the first one is the generation of an induced subgraph, which keeps the borderline instances from negative class, and the second one obtains a Minimum Spanning Tree constructed with the patterns included in the core of the negative class.

The experimental study was carried out using three supervised classifiers: 1NN, C4.5 and SVM, which allowed validate the effectiveness of the methods proposed.

We compare the performance of both proposals with some well-know condensed methods, finding that both methods obtain datasets with an IR very low with high quality, where the classifiers outperform their behavior respect than the obtained with the when other condensed methods are used. In addition, the computational cost of both proposals, in the worst-case, is computed on $\mathcal{O}(n^2)$.

Finally, using the Wilcoxon test was possible to emphasize that the methods proposed obtain high condensed datasets without lost useful information, with difference statistically significant and better behavior in terms of geometric mean, in comparison with othrers condensed methods widely used in the state-of-art.

The open lines pointing out to study the multi-class problems, to apply another classifiers, as well as to use datasets with greater size.

Acknowledgment

This work was partially supported by the Universitat Jaume I under grant [UJI-B2018-49], the Mexican CONACYT under scholarship [702275] an by the 5046/2020CIC UAEM proyect.

References

[1] Cleofas-Sánchez, L., J. Sánchez, V. García and R. Valdovinos, *Associative learning on imbalanced environments: An empirical study*, Expert Systems with Applications **54** (2016), pp. 387–397.

[2] Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms," MIT press, 2009.

[3] Dhar, V., *Data Science and Prediction*, Commun. ACM **56** (2013), pp. 64–73.

[4] Galar, M., A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2012), pp. 463–484.

[5] García, V., A. I. Marqués and J. S. Sánchez, *Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction*, Information Fusion **47** (2019), pp. 88–101.

[6] González, A., E. Barra, A. Beghelli and A. Leiva, *A sub-graph mapping-based algorithm for virtual network allocation over flexible grid networks*, in: *2015 17th International Conference on Transparent Optical Networks (ICTON)*, 2015, pp. 1–4.

[7] Hart, P., *The Condensed Nearest Neighbor rule*, IEEE Transactions on Information Theory **14** (1968), pp. 515–516.

[8] Hassani, L., M. R. Moosavi and P. Setoodeh, *A graph based approach to analyse metabolic networks for strain engineering*, in: *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, 2019, pp. 1839–1843.

[9] Liu, X., J. Wu and Z. Zhou, *Exploratory Undersampling for Class-Imbalance Learning*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39** (2009), pp. 539–550.

[10] Maillo, J., J. Luengo, S. García, F. Herrera and I. Triguero, *A preliminary study on hybrid spill-tree fuzzy k-nearest neighbors for big data classification*, in: *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018, pp. 1–8.

[11] Newman, M. E. J. and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E **69** (2004), p. 026113.
URL https://link.aps.org/doi/10.1103/PhysRevE.69.026113

[12] Prim, R. C., *Shortest connection networks and some generalizations*, The Bell System Technical Journal **36** (1957), pp. 1389–1401.

[13] Samaddar, A., T. Goswami, S. Ghosh and S. Pal, *An algorithm to input and store wider classes of chemical reactions for mining chemical graphs*, in: *2015 IEEE International Advance Computing Conference (IACC)*, 2015, pp. 1082–1086.

[14] Seiffert, C., T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano, *RUSBoost: A Hybrid Approach to Alleviating Class Imbalance*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans **40** (2010), pp. 185–197.

[15] Tomek, I., *wo Modifications of CNN*, IEEE Transactions on Systems, Man, and Cybernetics **SMC-6** (1976), pp. 769–772.

[16] Turvill, D., L. Barnby and A. Anjum, *A conceptual framework for the use of graph representation within high energy physics analysis*, in: *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2018, pp. 384–385.

[17] Yen, S.-J. and Y.-S. Lee, *Cluster-based under-sampling approaches for imbalanced data distributions*, Expert Systems with Applications **36** (2009), pp. 5718–5727.

[18] Zhang, Z. and E. R. Hancock, *A graph-based approach to feature selection*, in: X. Jiang, M. Ferrer and A. Torsello, editors, *Graph-Based Representations in Pattern Recognition* (2011), pp. 205–214.

A Classification results

Tables included in this appendix summarizes the performance classification with the 1-NN, C4.5 and SVM classifier over condensed datasets.

Table A.1
Geometric mean results obtained by 1-NN.

DataSet	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
yeast-0-5-6-7-9_vs_4	61.2	73.5	50.7	69.0	78.4	76.7	77.1	**93.7**	75.8
vowel0	100	97.2	70.7	**100**	98.9	98.7	99.8	**100**	98.8
glass-0-1-6_vs_2	47.0	74.4	51.0	52.6	68.3	64.1	53.1	**91.1**	63.4
glass2	40.7	76.2	51.6	41.0	64.7	68.7	53.1	**93.9**	68.6
shuttle-c0-vs-c4 7	99.6	99.6	**100**	99.6	99.6	99.6	99.6	99.6	99.6
yeast-1_vs_7	62.3	74.8	46.2	67.6	66.6	65.9	67.2	**94.9**	70.7
glass4	86.6	79.9	65.3	86.6	80.7	87.4	86.6	**88.4**	96.1
ecoli4	86.3	95.0	81.4	86.3	92.5	92.2	91.9	94.9	**100**
page-blocks-1-3_vs_4	98.2	98.2	94.5	98.2	94.5	96.6	**99.9**	98.1	98.2
glass-0-1-6_vs_5	80.9	77.0	65.1	80.9	94.3	91.0	94.0	88.9	**94.9**
shuttle-c2-vs-c4	91.3	91.3	81.6	91.3	**100**	98.2	91.3	91.3	92.6
yeast-1-4-5-8_vs_7	44.1	64.5	39.7	47.9	69.7	65.1	57.0	**90.6**	70.5
glass5	81.1	94.3	75.2	81.2	88.2	91.2	93.3	94.3	**94.9**
yeast-2_vs_8	77.0	59.8	58.3	77.3	54.8	75.1	80.2	**92.2**	65.5
flare-F	30.3	81.4	30.6	33.7	78.9	77.3	26.1	76.6	**82.4**
yeast4	58.8	83.3	43.7	62.2	84.3	77.1	78.6	**88.4**	72.3
yeast-1-2-8-9_vs_7	47.6	56.6	42.5	54.2	64.5	65.0	47.6	**94.9**	73.0
yeast5	82.1	**100**	65.5	87.6	93.2	94.6	82.1	86.3	97.5
ecoli-0-1-3-7_vs_2-6	83.7	78.2	48.8	84.0	**92.6**	81.9	84.2	65.5	87.3
abalone-17_vs_7-8-9-10	50.6	77.6	44.8	45.3	73.9	78.6	58.3	**97.2**	89.9
yeast6	71.1	69.9	54.8	79.0	81.0	83.7	71.2	**93.5**	79.2
shuttle-2_vs_5	**100**	99.0	96.1	**100**	**100**	99.9	**100**	**100**	**100**
kdd-buffer_oflow_vs_back	**100**	**100**	**100**	**100**	**100**	**100**	**100**	98.4	**100**
poker-8-9_vs_5	19.9	53.1	14.8	20.0	61.2	65.1	34.5	**87.7**	49.6
Avg. Rank	6.5	4.8	8.2	5.6	4.2	4.4	5.0	**3.1**	3.3

Table A.2
Geometric mean results obtained by c4.5.

DataSet	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
yeast-0-5-6-7-9_vs_4	65.1	70.6	60.2	65.8	73.2	77.0	74.3	**87.9**	73.0
vowel0	**96.9**	92.8	80.3	**96.9**	95.0	95.9	94.5	96.0	96.5
glass-0-1-6_vs_2	52.3	60.0	43.8	53.1	46.7	63.5	52.0	**94.1**	33.4
glass2	53.1	90.7	22.4	33.2	55.2	65.2	58.2	**97.0**	82.4
shuttle-c0-vs-c4 7	**100**	**100**	99.6	**100**	**100**	99.9	99.9	**100**	**100**
yeast-1_vs_7	54.3	64.5	57.2	65.2	71.6	67.4	62.8	**84.9**	70.9
glass4	82.2	73.0	73.0	82.2	88.4	89.1	72.4	80.7	**96.1**
ecoli4	82.9	87.5	**92.3**	82.9	76.5	86.3	77.1	81.4	82.8
page-blocks-1-3_vs_4	96.1	**100**	94.5	99.8	98.2	97.5	97.1	94.3	98.2
glass-0-1-6_vs_5	99.4	88.2	95.3	93.7	**100**	92.5	99.7	83.1	**100**
shuttle-c2-vs-c4	90.9	**100**	81.6	90.9	91.3	99.4	90.9	**100**	**100**
yeast-1-4-5-8_vs_7	0.0	59.6	0.0	0.0	66.6	53.1	25.8	**91.3**	62.5
glass5	98.8	**100**	85.3	99.5	94.3	93.9	**100**	83.1	**100**
yeast-2_vs_8	22.4	61.2	69.2	0.0	74.8	72.1	31.6	**79.1**	64.7
flare-F	15.2	85.9	32.7	0.0	84.8	84.2	0.0	78.9	**89.4**
yeast4	53.9	78.3	0.0	57.4	**83.3**	78.2	68.4	82.3	77.3
yeast-1-2-8-9_vs_7	48.2	53.7	47.9	44.6	59.9	67.3	40.8	**96.5**	66.6
yeast5	86.3	**96.6**	77.8	89.0	87.4	94.2	88.9	87.7	95.2
ecoli-0-1-3-7_vs_2-6	84.4	71.4	70.4	84.4	70.0	74.5	84.4	57.1	**94.3**
abalone-17_vs_7-8-9-10	34.7	75.0	0.0	34.6	78.8	75.5	57.0	**92.7**	88.7
yeast6	73.3	72.8	70.2	73.3	82.6	81.9	71.4	**90.8**	85.3
shuttle-2_vs_5	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
kdd-buffer_oflow_vs_back	**98.3**	95.0	0.0	**98.3**	**98.3**	97.7	**98.3**	95.0	96.6
poker-8-9_vs_5	0.0	47.8	0.0	0.0	44.0	44.6	0.0	**56.6**	44.4
Avg. Rank	5.9	4.4	7.5	5.5	4.3	4.2	6.0	3.9	**3.4**

Table A.3
Geometric mean results obtained by SVM.

DataSet	Original	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
yeast-0-5-6-7-9_vs_4	0.0	78.3	55.0	0.0	78.4	77.7	44.2	**93.0**	74.8
vowel0	78.0	90.0	45.6	78.0	93.9	89.7	91.3	**93.4**	92.7
glass-0-1-6_vs_2	0.0	55.2	0.0	0.0	45.6	32.1	0.0	**68.3**	43.1
glass2	0.0	68.6	0.0	0.0	47.1	21.8	0.0	**84.0**	67.6
shuttle-c0-vs-c4 7	99.6	**100**	99.6	99.6	**100**	**100**	99.6	99.6	**100**
yeast-1_vs_7	0.0	81.2	0.0	0.0	83.3	64.5	0.0	**87.9**	76.9
glass4	39.2	68.8	40.7	39.2	84.3	87.6	39.2	53.8	**96.1**
ecoli4	63.2	92.5	92.3	63.2	97.5	94.9	74.2	92.2	**100**
page-blocks-1-3_vs_4	65.4	71.9	74.5	65.4	79.4	73.1	65.4	**90.3**	84.5
glass-0-1-6_vs_5	0.0	81.6	73.9	0.0	94.3	86.4	0.0	66.7	**94.9**
shuttle-c2-vs-c4	90.9	91.3	**100**	90.9	**100**	**100**	90.9	91.3	**100**
yeast-1-4-5-8_vs_7	0.0	60.6	0.0	0.0	58.9	47.3	0.0	**84.5**	72.7
glass5	0.0	88.2	75.2	0.0	88.2	81.8	0.0	74.5	**100**
yeast-2_vs_8	74.1	72.3	73.4	**74.2**	**74.2**	73.3	**74.2**	**74.2**	63.6
flare-F	0.0	78.2	36.2	15.2	76.8	81.7	0.0	73.2	**84.4**
yeast4	0.0	82.4	0.0	0.0	**85.2**	80.9	0.0	70.8	78.0
yeast-1-2-8-9_vs_7	0.0	52.4	0.0	0.0	69.7	56.4	0.0	**98.3**	71.6
yeast5	0.0	98.9	70.9	0.0	92.9	95.8	0.0	89.4	**95.0**
ecoli-0-1-3-7_vs_2-6	84.1	78.2	78.7	84.0	92.6	84.1	84.0	60.6	**92.6**
abalone-17_vs_7-8-9-10	0.0	74.1	0.0	0.0	69.8	69.6	0.0	**98.1**	89.9
yeast6	0.0	78.5	0.0	0.0	**88.4**	87.7	0.0	75.0	83.7
shuttle-2_vs_5	86.9	91.7	67.9	86.9	**97.9**	94.2	86.9	85.9	94.8
kdd-buffer_oflow_vs_back	98.3	100	98.3	98.3	**100**	**100**	98.3	98.4	**100**
poker-8-9_vs_5	0.0	51.8	0.0	0.0	44.9	30.1	0.0	53.6	**58.3**
Avg. Rank	7.4	3.8	6.6	7.3	**2.6**	3.8	7.1	3.8	**2.6**

B Percentage of Reduction

Table B.1 summarizes the final percentage reduction after applying the condensing methods.

Table B.1
% Reduction.

DataSet	RUS	CNN	TL	EUS	RUSBOOST	SBC	IG-US	MIST-US
yeast-0-5-6-7-9_vs_4	89.30	81.55	2.28	89.30	83.06	20.10	88.64	88.64
vowel0	89.98	97.39	0.02	89.98	84.97	10.49	89.51	89.51
glass-0-1-6_vs_2	94.82	88.41	47.85	94.82	92.38	53.95	94.82	94.82
glass2	91.37	82.74	1.54	91.37	87.31	14.73	91.37	91.37
shuttle-c0-vs-c4 7	92.79	99.64	0.00	92.79	89.21	0.35	92.55	92.55
yeast-1_vs_7	93.01	80.42	1.86	93.01	89.51	20.28	93.01	93.01
glass4	93.54	93.54	0.05	93.54	90.55	2.04	93.54	93.54
ecoli4	93.67	93.99	0.00	93.67	90.51	5.70	93.34	93.34
page-blocks-1-3_vs_4	93.69	93.95	0.24	93.69	90.54	7.22	93.46	93.46
glass-0-1-6_vs_5	94.86	93.71	0.55	94.86	92.57	3.98	94.86	94.86
shuttle-c2-vs-c4	95.12	97.56	0.00	95.12	92.68	0.00	95.12	95.12
yeast-1-4-5-8_vs_7	95.48	81.60	1.36	95.48	93.21	23.68	95.15	95.15
glass5	95.61	94.63	0.50	95.61	93.66	3.91	95.61	95.61
yeast-2_vs_8	95.67	89.83	1.08	95.67	93.51	9.09	95.44	95.44
flare-F	95.80	86.80	0.39	95.80	93.74	2.34	95.80	95.80
yeast4	96.44	91.00	0.91	96.44	94.70	21.43	95.97	95.97
yeast-1-2-8-9_vs_7	96.73	86.04	1.10	96.73	95.09	3.28	96.62	96.62
yeast5	96.94	96.53	0.42	96.94	95.42	2.23	96.64	96.64
ecoli-0-1-3-7_vs_2-6	97.45	94.53	0.72	97.45	96.35	4.01	97.45	97.45
abalone-17_vs_7-8-9-10	97.46	93.68	0.57	97.22	96.18	21.14	97.22	97.22
yeast6	97.58	93.03	0.69	97.51	96.41	2.83	97.36	97.36
shuttle-2_vs_5	98.50	99.60	0.00	98.50	97.77	5.90	98.40	98.40
kdd-buffer_oflow_vs_back	98.64	99.90	0.00	98.64	97.96	3.26	98.59	98.59
poker-8-9_vs_5	98.78	93.80	0.34	98.78	98.20	15.66	98.73	98.73
Avg. reduction	95.13	91.83	2.60	95.12	92.73	10.73	94.97	94.97