Discovering Minority Sub-clusters and Local Difficulty Factors from Imbalanced Data

Mateusz Lango, Dariusz Brzezinski, Sebastian Firlik, and Jerzy Stefanowski

Institute of Computing Science, Poznan University of Technology,
ul. Piotrowo 2, 60–965 Poznan, Poland
{mlango,dbrzezinski,jstefanowski}@cs.put.poznan.pl

Abstract. Learning classifiers from imbalanced data is particularly challenging when class imbalance is accompanied by local data difficulty factors, such as outliers, rare cases, class overlapping, or minority class decomposition. Although these issues have been highlighted in previous research, there have been no proposals of algorithms that simultaneously detect all the aforementioned difficulties in a dataset. In this paper, we put forward two extensions to popular clustering algorithms, ImKmeans and ImScan, and one novel algorithm, ImGrid, that attempt to detect minority sub-clusters, outliers, rare cases, and class overlapping. Experiments with artificial datasets show that ImGrid, which uses a Bayesian test to join similar neighboring regions, is able to re-discover simulated clusters and types of minority examples on par with competing methods, while being the least sensitive to parameter tuning.

Keywords: class imbalance, minority class categorization, data difficulty factors, class overlapping, minority sub-clusters

1 Introduction

Improving classifiers learned from class-imbalanced data has been a topic of growing research in recent decades and several specialized algorithms have been introduced [2,6]. However, less effort has been put into studying the *characteristics* of imbalanced data, which make learning from imbalanced data so difficult.

It has been shown that neither the global imbalance ratio between the minority class and majority class nor the cardinality of the minority class are the main sources of difficulty. Other *data difficulty factors*, referring to internal characteristics of class distributions, are usually more influential. Several studies have demonstrated the high impact of the following factors: decomposition of the minority class into many sub-concepts [7,9], overlapping between the classes [5,16], and presence of many minority class examples inside the majority class region [13]. When these data difficulty factors occur *together* with class imbalance, they may seriously deteriorate the recognition of the minority class [12,13].

Identification of data difficulty factors may help in distinguishing different categories of imbalanced datasets (easier or more difficult to learn from). Consequently, specialized classifiers and preprocessing methods are more sensitive

to certain data categories [11,12]. Therefore, such an analysis of data characteristics and data difficulty factors may be important, on the one hand, to better understand the nature of class imbalanced data and, on the other, to aid the development of new classification methods.

Nevertheless, automatic discovery of the aforementioned factors in real world datasets is not an easy task and may not give unique results. Most known studies on difficulty factors have been carried out on synthetic data with ground truth knowledge. Discovering *sub-concepts* of the minority class is usually done with clustering algorithms such as k-means [9,14]. However, tuning parameters of clustering, the number of expected sub-concepts, dealing with complex, non-spherical shapes and outliers is problematic in real imbalanced datasets. Therefore, discovering minority sub-concepts still constitutes a research challenge.

Some other difficulty factors may be linked to *different types of examples* forming the minority class distribution with respect to their relative position. This view has led Napierala and Stefanowski to differentiate between safe and unsafe examples for recognizing minority instances [11]. The unsafe examples are further categorized into borderline, rare cases, and outliers. These authors have also introduced an approach to identify these types of examples by analyzing class label distributions in the neighborhood of minority examples [11,12]. The results of these works have been useful for constructing new preprocessing methods and specialized classifier ensembles for imbalanced data [19]. However, this approach is unable to detect sub-concepts inside the minority class.

Therefore, an open research question is: whether it is possible to construct a clustering approach that simultaneously discovers sub-concepts in complex imbalanced data and categorizes types of examples inside discovered clusters?

The main aim of this paper is to solve this research problem by introducing new specialized clustering algorithms. For this purpose, we put forward two extensions of popular clustering algorithms, ImKmeans and ImScan, as well as propose a novel approach, called ImGrid, dedicated to discovering minority sub-concepts and categorizing examples simultaneously. ImGrid uses spatial, density, and statistical characteristics of the attribute space, to detect and analyze minority class regions. The algorithms are experimentally evaluated on a comprehensive set of synthetic datasets with hidden sub-concept structures and various proportions of data difficulty factors.

The paper is organized as follows: related literature is discussed in Section 2; the proposed ImKmeans, ImScan, and ImGrid algorithms are described in Section 3; experimental results are discussed in Section 4; and finally conclusions and lines of future research are drawn in Section 5.

2 Related Work

2.1 Characteristics of Imbalanced Data

A dataset is considered to be imbalanced when the cardinality of the minority class is much smaller than the majority class (which is expressed by the global

imbalanced ratio between these two classes). In this paper we consider a standard formulation of the binary class or binarized multi-class imbalance problem [6].

It is worth noting that the global imbalance between classes may not pose difficulty for learning accurate classifiers by itself. Some, even highly, imbalanced data can be accurately learned by standard algorithms if the classes are well separated. When the rarity of the minority class is combined with other data difficulty factors concerning instance distributions in the attribute space [19], then it has a stronger negative impact on the recognition of the minority class.

Although many of the considered data factors are also known to affect learning in balanced domains, when they occur *together* with class imbalance the deterioration of classification performance is amplified and affects mostly (or sometimes only) the minority class. In this study, we focus on the following data difficulty factors: decomposing the minority class into sub-concepts, overlapping between classes, presence of outliers, and rare instances.

The influence of *class decomposition* into smaller sub-parts has been noticed by Japkowicz et al. [9]. Their experimental studies have demonstrated that the degradation of classification performance has resulted from the fragmentation of the minority class, rather than from changing the global imbalance ratio. Such sub-clusters of minority examples, surrounded by majority examples correspond to, so called, *small disjuncts*, which are harder to learn and cause more classification errors than larger sub-concepts [16]. Other experiments [13] showed that classification performance drops when decision boundaries around sub-clusters are non-linear and overlap with majority class examples. Finally, a visual analysis of projections of popular imbalanced UCI data [12] confirmed that the minority class often does not form a compact homogeneous distribution, but is scattered into many smaller sub-clusters surrounded by majority examples.

High *overlapping* between regions of different classes in the attribute space has already been recognized as particularly influential for standard, balanced, classification problems. However, its impact is even stronger when recognizing minority examples, see e.g. experimental studies [5]. The authors of these studies have also shown that the local imbalance ratio inside the overlapping region is more influential than the global ratio.

Other researchers characterize difficulty factors by considering mutual positions of a minority example with respect to other examples. One of the first studies in this direction [10] distinguished between *safe* and *unsafe* examples. More precisely, examples located in homogenous sub-regions populated by examples from the same class were called safe, whereas all other examples were denoted as unsafe. Napierala and Stefanowski proposed to further categorize unsafe examples into *borderline*, *rare cases*, and *outliers* [11]. Borderline examples can either be located in the overlapping between classes or positioned very close to complex non-linear decision boundaries. The two other types of examples occur deeper inside the safe region of the majority class. Outliers are isolated minority class singletons, whereas rare examples correspond to very small groups (pairs or triples) of examples. Comprehensive experiments [11,12] have shown that most benchmark imbalanced datasets contain mainly unsafe minority ex-

amples and the categorization of unsafe data correlates with the performance of classifiers.

2.2 Identification of sub-concepts and types of minority examples

Nearly all approaches to identify sub-concepts apply clustering algorithms that are run on examples of a single class, without analyzing their relation to remaining classes. Japkowicz et al. [7,14] proposed a k-means based oversampling method, where random oversampling is applied to majority and minority class clusters until the global class distribution becomes balanced. Other researchers discover within-class sub-concepts while constructing a classifier, by exploiting classifier predictions to tune the number of clusters k [18]. Nevertheless, the use of clustering algorithms for real-world datasets is still a non-trivial task. In case of k-means, the main difficulty is to tune the number of clusters k. It is also not obvious which optimization criteria should be considered as most clustering evaluation metrics were proposed for purely unsupervised frameworks. Moreover, existing works focus on the minority class without taking into account its local relationship with majority examples and challenges, such as class overlapping, rare cases, and outliers.

In an attempt to address these issues, as one of the contributions of this paper, we verify the utility of density-based clustering algorithms for the task of detecting sub-concepts. One of the analyzed algorithms is DBSCAN [4], which is capable of finding clusters of any shape and does not require the specification of the number of clusters. Nevertheless, DBSCAN requires specification of the following parameters: the minimal number of data points *min_points* and the maximal distance among those points ϵ in order to begin the formation of a new cluster. Just as finding a suitable k in k-means, the tuning of *min_points* and *epsilon* is not trivial.

Additionally, the proposed ImGrid algorithm is inspired by grid-based clusterers [3]. Algorithms from this group divide the attribute space into a set of cells, which are later joined in order to form clusters. To the best of our knowledge, grid-based clusterers have not been applied to imbalanced data analysis.

Concerning the identification of types of minority class examples, Napierala and Stefanowski [11] proposed to identify four types of examples (safe, borderline, rare, outlier) by analyzing class label distributions inside the neighborhood of each minority class example. The authors considered two ways of modeling the neighborhood, either with k-nearest neighbors or kernels. Depending on the number of examples from the majority class inside the neighborhood, it is estimated how safe or unsafe a minority example is. If all, or nearly all, its neighbors belong to the minority class, this example is treated as a safe one, otherwise it is categorized as one of three unsafe types: borderline, rare, outlier. The decision which of the four types should be assigned to a given example can either depend on the parameter k or thresholds on the within-region class probabilities.[1]

[1] Details on tuning the size of the neighborhood and a comparison between the k-NN and kernel-based approach can be found in [12]

3 Clustering and Categorizing Minority Examples

Following the critical discussion on limitations of existing approaches in the previous section, the main goal of our work is to create a clustering algorithm that is capable of not only discovering minority class sub-concepts, but also revealing their underlying example types. For this purpose, we put forward a novel learning approach, called *ImGrid*, and devise modifications of k-means and DBSCAN, called *ImKmeans* and *ImScan*.

ImGrid (Imbalanced Grid) is inspired by grid clustering algorithms [3]. The main steps of the algorithm involve: 1) dividing the attribute space into grid cells, 2) joining similar adjacent cells taking into account their minority class distributions, 3) labeling examples according to difficulty factors, 4) forming minority sub-clusters.

Since cells are joined based on example distributions, each cell should contain enough examples to make the estimation of the example density feasible. Hence, the presented algorithm divides each dimension of the attribute space into $\lceil \sqrt[m]{|D|/10} \rceil$ equally wide intervals, where $|D|$ is the size of the dataset and m is the number of dimensions of the attribute space. This formula, inspired by histogram bin count heuristics, ensures that, on average, we have 10 data points in each cell. The value 10 was chosen to make the cell as small as possible, while retaining a reasonable amount of data for statistical comparisons of cell distributions.

The second step of ImGrid requires a method for joining adjacent cells. The joining mechanism takes into account the distribution of minority and majority class examples in grid cells and combines them only if the distribution of the classes is similar. In particular, the algorithm aims at connecting cells that contain examples of similar difficulty, and one way of achieving this goal is to use the statistical hypothesis testing framework. The most popular tests for the comparison of discrete distributions are Pearson's chi-squared test and its exact alternatives, such as Fisher's exact test or Barnard's test [1]. However, since those tests cannot directly state that the distributions are identical (they can only fail to reject the null hypothesis), we decided to use a Bayesian test [8], which allows to calculate the hypotheses' probability. Using this test, ImGrid joins adjacent cells when the data statistically shows that the distributions are similar. The level of required confidence in order to merge cells is a parameter of the algorithm α; note that in this case α should be always greater than 0.5 (probability that the distributions are similar should be higher than they are not). Since in binary classification the comparison of the class distribution reduces to the analysis of two proportions, we have chosen a test constructed on the Bayes factor for the beta-binomial model. As prior distribution of the classes we use a non-informative Jeffreys prior [8].

In the third step of ImGrid, one of four difficulty labels (safe, borderline, rare, or outlier) is assigned to each cluster based on the ratio of minority and majority class examples. We refer to previous studies on modeling data difficulties [12] and use the following thresholds to assign the labels: if the proportion of minority examples p is greater than 0.7, the safe label is assigned; if $0.7 \geq p > 0.3$ then

borderline label is attached; the rare or outlier label is assigned if $0.3 \geq p > 0.1$ or $0.1 \geq p > 0$, respectively [12].

Finally, having a clustering that divides the data into regions of different difficulties, adjacent cells containing minority examples are joined. By joining minority examples regardless of their difficulty labels, the algorithm forms minority sub-clusters. The pseudocode of ImGrid is presented in Algorithm 1.

Algorithm 1 ImGrid

Input: D: m-dimensional dataset, α: threshold for statistical test
Output: *types_grid*: grid with detected types, *clustering_grid*: grid with clustering

grid ← split dimensions into $\left\lceil \sqrt[m]{|D|/10} \right\rceil$ equi-width intervals ▷ 1) Create grid
to each *cell* ∈ *grid* assign corresponding data points
while $[cell_1, cell_2]$ ← FIND_CELLS_TO_JOIN(grid) **do** ▷ 2) Join similar cells
    grid.join($cell_1, cell_2$)
for *cell* ∈ *grid* **do** ▷ 3) Assign type to examples in cells
    p ← *cell.minority_num*/*cell.example_num*
    cell.assign_label(p)
types_grid ← *grid.copy*()
for *cell* ∈ *grid* **do** ▷ 4) Form minority clusters
    for *neighbor* ∈ *cell.get_neighbors*() **do**
        if *cell.has_minority*() **and** *neighbor.has_minority*() **then**
            grid.join(*cell*, *neighbor*)
clustering_grid ← *grid*
return [*types_grid*, *clustering_grid*]

function FIND_CELLS_TO_JOIN(grid)
    sort cells in *grid* by the prevalence of minority class in descending order
    for *cell* ∈ *grid* **do**
        for *neighbor* ∈ *cell.get_neighbors*() **do**
            neighbor.p ← probability that the distribution of examples in *neighbor* and *cell* is the same
        $[p, best\_neighbor]$ ← *neighbor* with the highest *neighbor.p*
        if $p > \alpha$ **then return** [*cell*, *best_neighbor*]
return false

We also put forward extensions of the k-means and DBSCAN clustering algorithms. The proposed approaches consist of: 1) dividing the whole dataset into two datasets with examples of one class only, 2) performing standard clustering on the dataset with minority examples, 3) incorporating majority examples into the clustering result, and finally, 4) assigning difficulty labels to each cluster.

In the third step of the extensions we attempted to imitate the philosophy of the original clustering algorithms. ImKmeans assigns each majority example to the minority cluster with the nearest centroid. This naive approach relies solely on the global clustering information available in k-means, thus, local difficulty

type categorization produced ImKmeans may be very imprecise. Conversely, ImScan attaches majority examples to the cluster of its nearest minority example, but only if the distance to the nearest minority example does not exceed ϵ. Both ImScan and ImKmeans assign difficulty labels using the same rules as ImGrid. Algorithms 2 and 3 present the pseudocodes of the proposed extensions.

Algorithm 2 ImKmeans

Input: D: dataset, k: number of clusters
Output: *clustering*: a set of clusters with detected types

```
[D+, D−] ← split D based on class labels                    ▷ 1) Divide dataset
clustering ← K-MEANS(D+, k)                                 ▷ 2) Cluster minority examples
for maj_example ∈ D− do                                     ▷ 3) Add majority examples to clusters
    c ← cluster with the nearest centroid to maj_example
    c.add_example(maj_example)
for c ∈ clustering do                                       ▷ 4) Assign type to examples in clusters
    p ← c.minority_num/c.example_num
    c.assign_label(p)
return clustering
```

Algorithm 3 ImScan

Input: D: dataset, ϵ: radius of considered neighborhood, *min_points*: minimum number of points required to form a dense region
Output: *clustering*: a set of cluster with detected types

```
[D+, D−] ← split D based on class labels                    ▷ 1) Divide dataset
clustering ← DBSCAN(D+, ε, min_points)                      ▷ 2) Cluster minority examples
for maj_example ∈ D− do                                     ▷ 3) Add majority examples to clusters
    nearest ← minority example closest to maj_example
    if distance(nearest, maj_example) < ε then
        c ← cluster of nearest
        c.add_example(maj_example)
for c ∈ clustering do                                       ▷ 4) Assign type to examples in clusters
    p ← c.minority_num/c.example_num
    c.assign_label(p)
return clustering
```

To the best of our knowledge, there have been no previous proposals of algorithms that simultaneously detect minority sub-concepts and identify local data difficulty factors. In the following section, we examine the utility of the proposed algorithms in terms sub-concept discovery and minority example categorization.

4 Experimental study

In this section, we experimentally evaluate ImGrid, ImScan, and ImKmeans, on 78 synthetic datasets with controlled proportions and placement of data difficulty

factors. The proposed algorithms are analyzed in terms of their ability to discover class sub-concepts and detect different difficulty labels. Hence, the clustering performance and the accuracy of difficulty type categorization is measured. Finally, we asses how well the algorithms balance clustering and categorization tasks, and compare their processing time.

4.1 Experiment setup

In our experiments, we compare the proposed three algorithms (ImGrid, ImScan, ImKmeans) and the algorithm for the identification of example difficulty type by Napierala and Stefanowski [11] (Napierala) with the following parameters:

- ImGrid: $\alpha \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$;
- ImScan: $\epsilon \in \{10, 30, 50, 70, 90\}$, $min\_points \in \{2\}$;
- ImKmeans: $k \in [1, 9]$;
- Napierala: number of neighbors $k \in \{5, 7, 9, 11\}$.

The algorithm of Napierala and Stefanowski was chosen as a baseline for categorizing minority class examples, however, this algorithm is not applicable to clustering tasks and will not be compared with the remaining approaches in terms of detecting sub-concepts. On the other hand, when analyzing clustering performance, ImKmeans and Imscan default to standard definitions of k-means and DBSCAN, and, therefore, can be considered as baseline approaches in terms of detecting minority sub-concepts. We note that the *min_points* parameter in ImScan was set to 2 to ensure that rare cases can be identified as separate clusters and distinguished from outliers.

Clustering was evaluated using Adjusted Mutual Information [17] (AMI) which takes into account not only the total number of clusters but also the correctness of example assignation to sub-concepts. Categorization was treated as a classification task and assessed using G-mean [10] over four difficulty types (safe, borderline, rare, outlier). These measures were selected, as they are deemed suitable for imbalanced data [6,17]. We note that traditionally if at least one class is unrecognized by the classifier, G-mean resolves to zero. To alleviate this property, we changed the recall of unrecognized classes from zero to 0.001. To differentiate from traditional G-mean, we denote the used measure as G-mean$^{0.001}$.

All the algorithms and evaluation methods were implemented in Python using the *scikit-learn* library [15].[2] Experiments were conducted on a machine equipped with an Intel i7-5500U 2.4Ghz processor and 8 GB of RAM.

4.2 Datasets

In our experiments, we used 78 synthetic binary classification datasets with six basic shapes of minority sub-clusters and varying proportions of data difficulty

[2] Source code, datasets, and reproducible test scripts available at: https://github.com/langus0/imgrid

factors. The datasets were created with the imbalanced dataset generator of Wilk et al. [20], which provides the ground truth for sub-concept and difficulty type detection. As this study focuses on local data difficulty factors, all the datasets have a constant 1:5 global class imbalance ratio. Table 1 presents the main characteristics of each dataset.

Table 1: Dataset characteristics; superscripts denote versions of datasets with different proportions of minority example types: ^{u}-unsafe, ^{b}-borderline, ^{r}-rare; subscript $_{s}$ denotes "sparse" versions of datasets, with much less examples

Dataset	Inst.	Attr.	Clust.	Safe	Border	Rare	Outlier
`clover`	1500	2/3	1	100 %	0 %	0 %	0 %
`dis`	1500	2/3/5	3	100 %	0 %	0 %	0 %
`hyp`	1500	2/3/5	1	100 %	0 %	0 %	0 %
`joined`	1500	2	4	100 %	0 %	0 %	0 %
`normal`	1500	2/3/5	1	100 %	0 %	0 %	0 %
`rothyp`	1500	2	1	100 %	0 %	0 %	0 %
`<dataset>`^{u}	...	...	+13	80 %	12 %	6 %	2 %
`<dataset>`^{b}	...	...	...	40 %	60 %	0 %	0 %
`<dataset>`^{r}	...	...	+50	30 %	40 %	20 %	10 %
`<dataset>`$_{s}$	250	...	...	100 %	0 %	0 %	0 %
`<dataset>`$^{u}_{s}$	250	...	+2	80 %	12 %	6 %	2 %

The `clover` dataset resembles a clover (Fig. 2a) with five prolonged leaves in 2d- (`clover`) and 3d-attribute space (`clover`$_3$). `dis` constitutes an example of spherical minority class sub-clusters in 2, 3, or 5 dimensions. Datasets `hyp` and `rothyp` exemplify a simple and rotated hyperplane decision boundary between two classes. The `joined` dataset allows to test the algorithms on overlapping sub-clusters, whereas `normal` is a uniformly distributed sphere in 2, 3 or 5 dimensions. Each of the six basic ("safe") datasets (`clover`, `dis`, `hyp`, `joined`, `normal`, `rothyp`) has five additional versions:

- ^{u}**-unsafe**: where the minority class also contains borderline instances, rare cases, and outliers;
- ^{b}**-borderline**: where the minority class is surrounded by a thick border of examples overlapping with the majority class;
- ^{r}**-rare**: where the number of safe examples is smaller than the number of unsafe examples;
- $_{s}$**-sparse**: where the dataset has much less examples, which introduces sparsity to the attribute space;
- $^{u}_{s}$**-sparse**: where the dataset is sparse and contains unsafe examples.

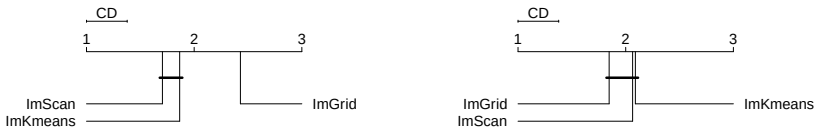
It is important to note that rare cases and outliers introduce additional minority sub-clusters to the dataset.

4.3 Minority class sub-clusters

Due to the large number of datasets and space limitations, detailed tabular results for each algorithm can be found in online supplementary materials.[3] In this and the following section, we summarize the results by means of selected plots, tabular summaries, and statistical hypothesis tests.

To compare the minority clustering performance of the proposed algorithms, we calculated Adjusted Mutual Information (AMI) for each clustering. However, since clustering is an unsupervised learning task and results can strongly depend on algorithm parameters (e.g. k in k-means), we compare the algorithms on two levels. The first level involves comparing *best models*, i.e., we choose the best parametrization of a given algorithm for each dataset separately, and report this "best" value for each dataset. This level corresponds to assessing algorithms, as if we explicitly knew how to tune them (which is usually not true). The second level involves comparing *mean models*, i.e., reporting the algorithms performance averaged over all parameterizations. This scenario corresponds to comparing algorithms as if they were parametrized by chance.

Table 2 shows detailed results for best models in terms of AMI on three selected datasets: $\texttt{clover}^u$, $\texttt{dis}^b$, and $\texttt{rothyp}^r$. Additionally, upper panels of Figure 2 show sub-concept clusterings for the selected datasets. The results confirm commonly known, complementary, characteristics of k-means and density-based clustering algorithms. On the $\texttt{dis}^b$ dataset, ImKmeans is capable of finding the perfect clustering, as this dataset has only homogenous sub-concepts. On $\texttt{clover}^u$ and $\texttt{rothyp}^r$, however, ImKmeans has trouble with noisy examples. Conversely, ImScan and ImGrid perform quite well on noisy datasets, but fail to detect the right number of clusters, when the sub-concepts overlap.



(a) Best model (b) Average performance

Fig. 1: Performance ranking using AMI. Algorithms that are not significantly different according to the Nemenyi test (at $\alpha = 0.05$) are connected.

Figure 1 graphically presents the results of the Friedman test with Nemenyi post-hoc analysis for both levels of comparison. The null-hypothesis that best models for each algorithm perform similarly was rejected with $p < 0.001$. Ideally tuned versions of ImKmeans and ImScan are significantly better in terms of AMI than ImGrid. However, the null-hypothesis of the Friedman test for comparing mean models cannot be rejected ($p = 0.245$). Moreover, ImGrid obtains the highest mean rank in this comparison. This shows how crucial parameter tuning is to the performance of k-means and DBSCAN.

[3] http://www.cs.put.poznan.pl/dbrzezinski/software/MinorityAnalysis.html

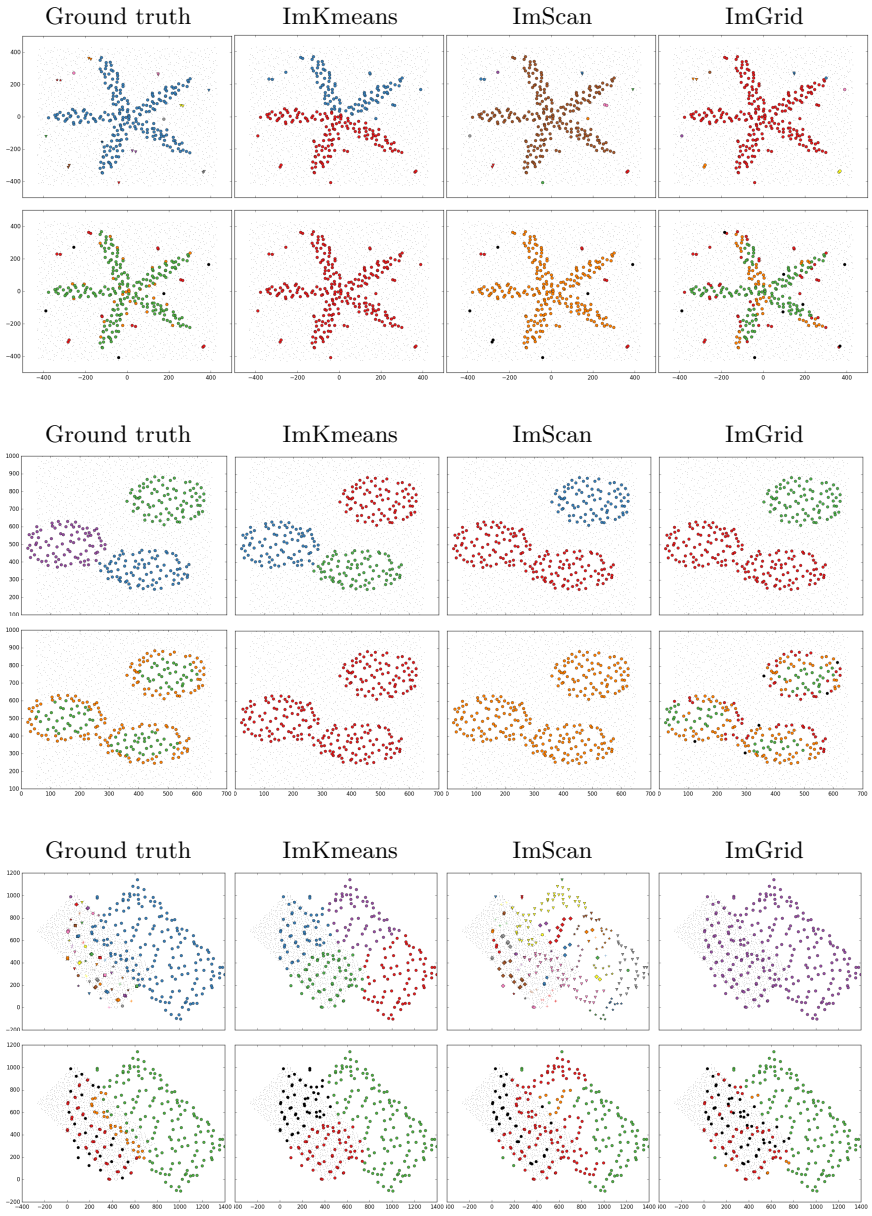


Fig. 2: Comparison of clustering results (upper panel of each pair) and minority type identification (lower panels) on the $\texttt{clover}^u$, $\texttt{dis}^b$, and $\texttt{rothyp}^r$ datasets. Clusters in upper panels are differentiated using shapes and colors. Types of minority class examples in lower panels are color-coded as follows: safe - green, borderline - orange, rare - red, outlier - black. Figure should be read in color.

Table 2: Clustering and categorization results for three selected datasets: $\texttt{clover}^u$, $\texttt{dis}^b$, and $\texttt{rothyp}^r$. Napierala added for comparison on categorization.

	G-mean$^{0.001}$	AMI	Time	Clusters	Safe	Border	Rare	Outlier
				$\texttt{clover}^u$				
ImKmeans	0.006	0.038	0.640	2	0	0	250	0
ImScan	0.150	0.613	0.483	11	0	235	8	7
ImGrid	0.530	0.486	0.142	9	140	58	44	8
Napierala	0.758	-	0.032	-	105	117	23	5
				$\texttt{dis}^b$				
ImKmeans	0.002	0.980	0.699	4	0	0	500	0
ImScan	0.032	0.577	0.438	2	0	250	0	0
ImGrid	0.526	0.577	0.156	2	66	121	57	6
Napierala	0.526	-	0.031	-	60	123	66	1
				$\texttt{rothyp}^r$				
ImKmeans	0.117	0.167	2.183	9	111	0	82	57
ImScan	0.106	0.183	0.428	43	82	10	123	35
ImGrid	0.134	0.000	0.165	1	134	8	59	49
Napierala	0.171	-	0.024	-	127	6	61	56

4.4 Minority example categorization

We also compared the ability of the algorithms to detect difficulty types of minority examples. Lower panels of Figure 2 show example categorization corresponding to clusterings from the upper panels.

One can notice that ImScan and ImGrid obtain quite accurate difficulty type predictions. It is worth noting, however, that on the presented plots ImGrid produces more accurate predictions. This is due to the fact that the plots present best models in terms of the clustering performance (AMI). Contrary to ImGrid, ImScan's results are not robust to the change of parameters, hence its best parametrization for clustering did not usually correspond with the best model for categorization. It is also worth noting that ImGrid achieves G-mean$^{0.001}$ values fairly close to those obtained by Napierala, which is an algorithm designed strictly for detecting example difficulty types and has no clustering capabilities.

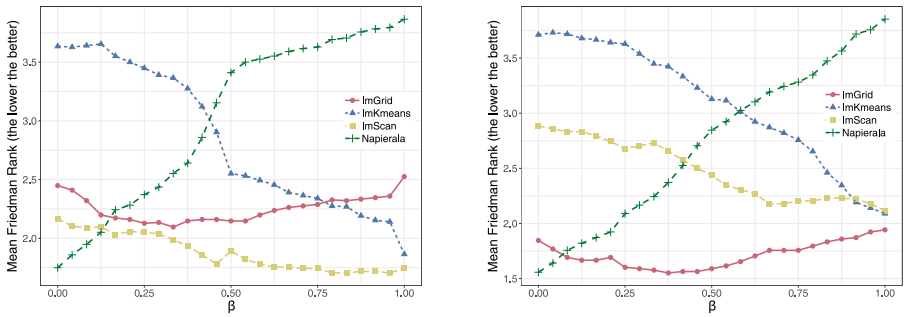
As it was done for clustering performance, we also performed the Friedman test to assess the significance of differences in performance for best and mean models. However, in this comparison we additionally analyze the performance of the algorithm of Napierala and Stefanowski. As Figure 3 shows, in terms of G-mean$^{0.001}$, Napierala is the best categorization algorithm, but it is not significantly better than ImGrid when looking at mean models. Moreover, once again it can be noticed that ImScan is highly sensitive to parameter tuning.

(a) Best model (b) Average performance

Fig. 3: Performance ranking using G-mean$^{0.001}$. Algorithms that are not significantly different according to the Nemenyi test (at $\alpha = 0.05$) are connected.

4.5 Balancing clustering and categorization

Our final view on the performance of the algorithms involved assessing the trade-off between clustering and categorization performance. For this purpose, we decided to evaluate the algorithm using a linear combination of AMI and G-mean$^{0.001}$, as follows: $\beta AMI + (1-\beta) Gmean^{0.001}$. By varying the parameter β, one can control which aspect of the task, clustering or categorization, is more important. Figure 4 shows mean ranks of the Friedman test with varying β, for both the best and mean models.



(a) Best model (b) Average performance

Fig. 4: Mean Friedman test ranks when evaluating the algorithms according to: $\beta AMI + (1-\beta) Gmean^{0.001}$

For $\beta = 0$ and $\beta = 1$, the mean ranks in Figure 4 correspond to results presented for categorization and clustering, respectively. Nonetheless, in the range $\beta \in (0, 1)$ one can analyze the trade-off offered by each algorithm. Looking at best models, it can be seen that ImScan can be successfully tuned to any value of β, with ImGrid being usually second. However, when comparing mean models, it can be noticed that on average ImGrid produces better results, suggesting that is much less prone to parameter tuning. This is an important finding, as the goal of this study was to simultaneously detect minority sub-clusters and data difficulty factors, without prior knowledge about how to cluster examples.

Furthermore, we measured the running time of each algorithm and performed a Friedman test (Fig. 5). One can notice that Napierala is the fastest approach,

however, it is an algorithm for the recognition of difficulty types only. Among algorithms which provide information about both minority sub-concepts and examples difficulty, ImGrid is significantly the fastest method. In terms of concrete values, ImGrid is on average almost two times faster than ImScan, its best competitor.

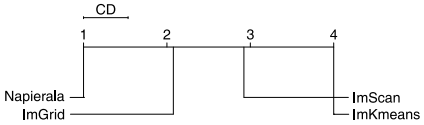


Fig. 5: Running time ranking. Algorithms that are not significantly different according to the Nemenyi test (at $\alpha = 0.05$) are connected.

5 Conclusions and Future Research

The main aim of this study was to find novel ways of discovering local data difficulty factors from imbalanced data. Up till now, efforts in this field have concentrated, *separately*, on detecting sub-concepts of the minority class and detecting local relationships between minority and majority examples. We argue that existing approaches to identifying minority sub-concepts are impractical as they heavily rely on non-trivial parameter tuning and are sensitive to outliers and other difficulty factors.

In this paper, we put forward ImGrid, an algorithm that *simultaneously* detects minority sub-clusters, outliers, rare cases, and class overlapping in imbalanced data. Additionally, we proposed two extensions to popular clustering algorithms, ImKmeans and ImScan, that incorporate knowledge about relationships between minority and majority examples. A comprehensive series of experiments characterized the strengths and weaknesses of each algorithm, showing that, depending on parameter tuning, each of the proposed algorithms is capable of successfully detecting sub-concept or characterizing difficulty types. However, the results highlighted ImGrid as a fast and easily parametrized trade-off between minority class clustering and example categorization.

Due to its small dependency on parameter tuning, ImGrid could be used to analyze real world datasets. Nevertheless, as future work the topic of defining its grid space for real data may be revisited, as more flexible approaches to dividing the attribute space can still be proposed. Moreover, the combination of clustering and example categorization gives the user two layers of information about an imbalanced dataset. These layers could be combined in a data difficulty metric, which would inform the user about the main difficulties in the dataset and suggest possible actions. Finally, it would be very interesting to use the gathered information to improve specialized algorithms for imbalanced data.

Acknowledgments. The authors' research was partly funded by the Polish National Science Center under Grant No. DEC-2013/11/B/ST6/00963. D. Brzezinski acknowledges the support Institute of Computing Science Statutory Funds.

References

1. Barnard, G.: A new test for 2× 2 tables. Nature 156, 177 (1945)
2. Branco, P., Torgo, L., Ribeiro, R.P.: A survey of predictive modeling on imbalanced domains. ACM Comput. Surv. 49(2), 31:1–31:50 (2016)
3. Cheng, W., Wang, W., Batista, S.: Grid-based clustering. In: Data Clustering: Algorithms and Applications, pp. 127–148. CRC Press (2013)
4. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) Proc. 2nd Int. Conf. Knowl. Disc. Data Mining. pp. 226–231 (1996)
5. García, V., Sánchez, J.S., Mollineda, R.A.: An empirical study of the behavior of classifiers on imbalanced and overlapped data sets. In: Proc. 12th Iberoamericann Congress on Pattern Recognition. LNCS, vol. 4756, pp. 397–406. Springer (2007)
6. He, H., Ma, Y. (eds.): Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press (2013)
7. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intell. Data Anal. 6(5), 429–449 (2002)
8. Jeffreys, H.: Some tests of significance, treated by the theory of probability. In: Proceedings of the Cambridge Philosophical Society. vol. 31, pp. 203–222 (1935)
9. Jo, T., Japkowicz, N.: Class imbalances versus small disjuncts. SIGKDD Explorations 6(1), 40–49 (2004)
10. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: One-sided selection. In: Proc. Int. Conf. Mach. Learn. pp. 179–186 (1997)
11. Napierala, K., Stefanowski, J.: Identification of different types of minority class examples in imbalanced data. In: Proc. 7th Int. Conf. Hybrid Artificial Intelligent Systems, Part II. LNCS, vol. 7209, pp. 139–150. Springer (2012)
12. Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. J. Intell. Inf. Syst. 46(3), 563–597 (2016)
13. Napierala, K., Stefanowski, J., Wilk, S.: Learning from imbalanced data in presence of noisy and borderline examples. In: Proc. 7th Int. Conf. Rough Sets Current Trends Comp. LNCS, vol. 6086, pp. 158–167. Springer (2010)
14. Nickerson, A., Japkowicz, N., Milios, E.E.: Using unsupervised learning to guide resampling in imbalanced data sets. In: Proc. 8th Int. Workshop Artif. Intell. Stat. pp. 261–265. Society for Artificial Intelligence and Statistics (2001)
15. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
16. Prati, R.C., Batista, G.E., Monard, M.C.: Class imbalances versus class overlapping: An analysis of a learning system behavior. In: Proc. 3rd Mexican Int. Conf. Artif. Intell. LNCS, vol. 2972, pp. 312–321. Springer (2004)
17. Romano, S., Vinh, N.X., Bailey, J., Verspoor, K.: Adjusting for chance clustering comparison measures. Journal of Machine Learning Research 17(134), 1–32 (2016)
18. Sobhani, P., Viktor, H., Matwin, S.: Learning from imbalanced data using ensemble methods and cluster-based undersampling. In: Proc. 3rd Int. Workshop New Frontiers Mining Complex Patterns. LNCS, vol. 8983, pp. 69–83. Springer (2014)
19. Stefanowski, J.: Dealing with data difficulty factors while learning from imbalanced data. In: Challenges in Computational Statistics and Data Mining, Studies in Computational Intelligence, vol. 605, pp. 333–363. Springer (2016)
20. Wojciechowski, S., Wilk, S.: Difficulty factors and preprocessing in imbalanced data sets: An experimental study on artificial data. Foundations of Computing and Decision Sciences 42(2), 149–176 (2017)