Exploring Feature-Level Duplications on Imbalanced Data Using Stochastic Diffusion Search

Haya Abdullah Alhakbani* and Mohammad Majid al-Rifaie

Department of Computing, Goldsmiths, University of London
London, United Kingdom
{h.alhakbani,m.majid}@gold.ac.uk

Abstract. Swarm intelligence mimics the behaviours of social insects like bees, wasps and ants to offer powerful problem solving metaheuristic which lies in a network of interactions amongst the agents of a multi-agent system as well as with their environment. One of the computer algorithms inspired by swarm intelligence is the stochastic diffusion search (SDS). SDS uses some of the processes and techniques found in swarm to solve search and optimisation problems. In this paper, a hybrid approach is proposed to deal with real-world imbalanced data. The proposed model involves oversampling the minority class, undersampling the majority class as well as optimising the parameters of the classifier, Support Vector Machine (SVM). The proposed model uses Synthetic Minority Over-sampling Technique (SMOTE) to perform the oversampling and the agents of a swarm intelligence technique, SDS, to perform an 'informed' undersampling on the majority classes. The use of this swarm intelligence technique in conducting the undersampling tasks is investigated and its impact on improving the classification results is demonstrated. In addition to comparing the agents-led undersampling with random undersampling, the results are contrasted against other best known techniques on nine real-world datasets. Additionally, further experiments are designed to explore the behaviour of the SDS agents during the undersampling process.

Keywords: Swarm intelligence, Agents, Class imbalance, Stochastic Diffusion Search, SVM

1 Introduction

Class imbalance – a major problem in machine learning – occurs when the number of instances in the majority class is significantly higher than the number of instances in the minority class. Class imbalance is present in most real-world data in medicine, business and many other fields. For instance, in direct marketing, businesses are interested in identifying potential buyers which are only a

* Corresponding author

small proportion of the target audience, and in charities where potential givers are a very small segments of the population. The issue has been receiving some attention in the literature (e.g. [13, 19, 24, 26, 11]) mostly due to the fact that data mining models tend to be influenced with the skewed data distribution, therefore the minority class is usually misclassified leading to bad performance.

In this work, SDS is applied to the problem of class imbalance in machine learning. Primarily, SDS, in this context, is tasked with the undersampling of the majority class and SMOTE is tasked with the oversampling of the minority class. Moreover, to solve the problem at the algorithmic level SVM values; C and gamma are optimised using a grid search and 5 cross validations to train and test the classifier. key research questions raised in the paper are:

– What is the impact of the duplication at dataset's feature-level (the role of duplications on each individual feature) on the undersampling process?
– How could SDS provide a way to address the feature-level duplications?
– Whether the proposed model in the paper, which uses SDS for undersampling, provides any outperformance compared to random undersampling as well as other techniques?
– Is it possible to make a recommendation on when to use the method proposed in this paper? In other words, which types of datasets are more responsive to the proposed technique?

2 Related Work

In a number of real-world classification applications, like software prediction, oil spill detection from satellite images, detection of fraudulent online credit card, diagnosis of rare diseases, training data might be imbalance [12, 31, 25], where the number of data (instances) in some classes are extremely smaller than other classes. This is often due to the limitations of a data collection process; e.g. high cost or privacy problems. This can be seen in biomedical data, for example, that is derived from a rare disease and an abnormal condition, or some data that often obtained via expensive experiments.

The key issue regarding an imbalanced learning problem is its dataset restricts the performance of most standard learning algorithms, and such algorithms often assume balanced class distributions. Such problems lead to biased standard machine learning algorithms, which are biased towards the majority class, since these types of algorithms attempt to maximize overall accuracy. Conversely, in the event of complex imbalanced datasets, these algorithms will not reveal the distributive characteristics of the data, thus leading to unfavourable accuracies within the data classes. In the medical field, this issue is particularly crucial, since learning from these imbalanced data can help us discover useful knowledge to make important decisions while it can also be extremely costly to misclassify these data.

Several solutions have been suggested in the literature to address this issue, amongst which are data-level techniques, algorithmic-level techniques and a combination of both. At the data level, solution is achieved by applying sampling on the dataset until it is balanced. There are various existing ways of sampling which include oversampling the minority class, undersampling the majority class and a combination of both. However, this issue, faces the challenges of the loss of important information and over fitting to balance the data. Despite these challenges researches are still implementing machine learning pre-processing techniques to overcome this issue; one approach is to generate new synthetic instances from the minority class. The approach, Synthetic Minority Oversampling Technique (SMOTE), takes a subset of the minority class samples (two or more) and creates a synthetic example using k-Nearest Neighbour algorithm (k-NN) [11].

At the algorithmic level, valuable progress has been made in finding better ways to overcome the challenges of sampling by investigating possible algorithmic solutions. When a machine learning algorithm is trained on imbalanced datasets, it assumes that the misclassification costs are the same, which is not true in all cases; a stark example would be in medical diagnosis and tumour detection where the cost of misclassifying cancerous cells could potentially lead to risking the patients' heath. Solutions can be applied by adjusting the cost parameter to improve the model's performance on imbalanced datasets. This approach is shown to perform well when dealing with imbalanced dataset [29].

Numerous machine learning algorithms have performed well on imbalanced datasets. These include adjusted decision tree [4], Support Vector Machines (SVMs) and genetic programming [15]. SVMs are forms of binary classification algorithms and are supervised learning methods which analyse data and discover patterns; they can be used for classification and regression in data mining. They are widely used cost sensitive machine learning algorithms as they have some distinctive advantages such as performance speed [35, 28]. Various researchers have combined data level solutions with a cost sensitive SVM. One way is to adjust the classification cost of negative and positive instances and combine this with a data level balancing techniques to improve the overall performance [5]. In another method applied by Akbani, Kwek and Japkowicz [1], SMOTE and Different Error Cost (DEC) are used in SVM to overcome the class imbalance problem.

In summary, there is no clear answer as to what is the best method to use when dealing with the challenging imbalanced datasets. A study by Weiss, McCarthy and Zabar [32] compares cost-sensitive learning and data level solutions in handling the issue of class imbalance; the authors highlight that the choice of method is dependant on the data in use; for example, when working on a datasets with $10,000$ instances, cost-sensitive learning performed well when compared with sampling. Another study by Tang and Chawla [28] to compare four variations of SVMs, shows that the application of granular SVMs-reptitive undersampling algorithm (GSVM-RU) outperformed the other variations of SVMs in aspects like effectiveness and efficiency.

2.1 Swarm intelligence in data mining

Finding efficient solutions to search and optimisation problems has inspired many researchers to utilise nature informed algorithms, where the interactions of simple agents could lead to promising solutions to challenging problems [7]. Swarm intelligence meta-heuristic models have been gaining increasing level of interest for solving common optimisation problems [27]. These technique often mimic the behaviours of social insects like bees, wasps and ants to offer powerful tools to solve problems. The emergence of collective intelligence, which is the core of such techniques, lies in a network of interactions among social insects and their environment; this process is described as a self-organised and decentralised system of collective behaviour [6].

When applied to data mining, swarm intelligence techniques are applicable to two broad areas: the first consists of a search technique in which individuals members of of the swarm move through the solution space in search of solution to a data mining task; the second area is a data organising approach which occurs by swarms moving data away from low dimensional feature space so they can achieve a clustering solution of the data. These two areas are, to some extent, explored through a few swarm intelligence algorithms, including ant colony optimisation algorithms which was utilised to evaluate solutions to single and multi-objective data for fitness functions [22].

One of the established swarm intelligence techniques which has been applied to various areas of optimisation, medical diagnosis, data clustering, and many more, is Stochastic Diffusion Search (SDS) [8]. Unlike many other nature inspired algorithms, such as ant colony optimisation that modifies physical properties of a simulated environment, SDS, which has a strong mathematical framework, uses the direct communication between agents inspired by the tandem calling mechanism found on one species of ants called *Leptothorax Acervorum.*

2.2 Stochastic diffusion search

Stochastic diffusion search (SDS) is a population based algorithm that implements direct communication patterns to evaluate the search and optimisation hypothesis. SDS is originally attributed to Bishop in 1989 [9], as a population based matching algorithm that uses direct communication patterns such as co-operative transport found among social insects to perform evaluation of search space.

In SDS, the agents population have 'hypotheses' about the possible solutions; these hypothesis are partially evaluated in order to provide feedback that ensure the agents convergence on promising solutions. Using SDS, agents communication and the 'partial' evaluation of hypotheses play the critical role in the performance of the agents and the emergence of "intelligence" [9]. SDS as shown in Algorithm 1 comprises of three key phases which are described below:

Initialisation phase

The initialisation phase is where each agent randomly selects a hypothesis (i.e. an element's index or in case of a dataset, the instance number) from the search space. These 'pointers' are later used to lead the search process of the SDS population.

Test phase

After the initialisation phase, where each agent is assigned to a random hypothesis in the search space, in the test phase, each agent's hypothesis is evaluated individually based on the objective function; if the hypothesis evaluation is successful, the agent is set to active, otherwise inactive. Therefore, at the end of the test phase, each agent adopts either of the two possible boolean outcomes.

Diffusion phase

During the diffusion phase, information about the hypotheses are exchanged amongst agents. In this work, passive recruitment strategy is used where each inactive agent selects another agent randomly; if the randomly selected agent is active, the hypothesis of the active agent is *diffused* to the inactive agent; otherwise, the inactive agent randomly selects a hypothesis from the search space.

Existing studies have tested the efficacy of different SDS algorithms and their fitness to solve search problems. For instance, the performance of different types of SDS algorithms and RANSAC were compared for a hyperplane estimation task, where SDS is shown to have similar performance as RANSAC but with better potential for turning into particular search problems for outcomes [33]. SDS has also been placed in context with other biological and traditional search algorithms in numerous studies. In most search cases, SDS have been found to outperform other algorithms especially for partial search functions.

In principle SDS is applicable to changing objective functions which makes it suitable for dynamic problems as well as the static ones. Given this feature, SDS has been applied to applications like eye tracking in facial images by using a combination of a Stochastic Search network and an n-tuple network [16], site decision for transmission equipment for wireless networks [21] and mouth locating in human faces images [17].

Algorithm 1: SDS Algorithm

```
Initialising agents
While (stopping condition is not met)
        Test hypotheses
        Diffusion hypotheses
End
```

In this paper more work has been carried out to make SDS more applicable to solve the problem of class imbalance in machine learning in a reliable yet computationally effective way.

3 Experiment

Some previous work (e.g. [10]) have shown that a combination of data level and algorithmic level solutions can improve the model performance on imbalanced datasets . In the experiments conducted for this work, data level processes are used and SDS is set to undersample the majority classes. Additionally, SMOTE algorithm is used to oversample the minority classes. At the algorithmic level, the model uses SVM algorithm, where parameters like C (misclassification cost) and gamma for the radial kernel are optimised using a grid search, which is a simple search through a range of parameters. The range for C has been defined as $[2^{-5}, 2^{15}]$ and the range of gamma as $[2^{-15}, 2^{3}]$ [20]. The search is performed using 5 cross validation and multi-threading to run multiple process at a time. The hybrid approach is then contrasted against random undersampling (along with SVM optimisation).

To evaluate the proposed model, nine imbalanced datasets are used in this work. The datasets are available from the University of California, Irvine (UCI) Machine Learning Repository, plus the Oil Spills dataset [23]. The datasets, all collected from real-world cases, vary greatly in their class distributions, sizes and features characteristics (continuous and discrete features). The full list of dataset used are shown in Table 1. In the experiments reported here, for the Abalone dataset, the authors applied the algorithm on the classes '9' versus '18'; for the Yeast dataset, the model is applied on the classes 'CYT' versus 'POX'; and for the Vehicle dataset, the class 'Van' vs the others is used, in order to have a highly imbalanced dataset. Additionally, all datasets values are normalised, and missing values are removed.

Balancing the datasets

The first task is the application of SDS to undersample the majority class where the aim is to reduce the size of majority class (SDS's search space). The proposed model uses SDS to undersample the majority class to around fifty percent; in cases where the minority class instances need to be oversampled (to reach a comparable size with undersampled majority class) SMOTE is applied, with the following configurations: class is set to zero to detect the minority class automatically; nearest neighbours is set to 5 as this will create synthetic instances from the five nearest neighbours; the percentage of instances to create depends on the majority class size; and the number of seeds used for the sampling is set to 0.

Applying SDS to undersample the majority class instances

In this experiment, SDS is implemented to perform the undersampling for the majority class. The number of SDS agents are empirical set to be half of the search spaces (half the number of instances); quarter of the SDS population size is set for the number of iterations to undersample the majority class. Initially a model (an instance from the majority class) is randomly selected from the

Table 1: Summary of datasets used in this experiment

Dataset	**No. of instances**	**Missing Values**	**Minority Class**	**Majority Class**	**Distribution**	**Continuous Features**	**Discrete Features**	**SMOTE**
Oil Spills	937	No	41	896	0.04:0.96	49	0	Yes
Yeast	483	No	20	463	0.04:0.96	8	0	Yes
Abalone	731	No	42	689	0.06:0.94	7	1	Yes
Vehicle	846	No	199	647	0.23:0.77	18	0	Yes
Breast Cancer	699	Yes	241	458	0.34:0.66	9	0	No
Bank Marketing	4119	No	451	3668	0.11:0.89	10	10	Yes
Thoracic Surgery	470	No	70	400	0.15:0.85	3	13	Yes
Ionosphere	351	No	126	225	0.35:0.65	34	0	No
Hepatitis	155	Yes	32	132	0.21:0.79	6	13	Yes

search space (the entire majority class instances) and the agents are set to find the closest match (an instance) from the remaining items of the search space. Once a match or the most similar item is found (further details as to how this is carried out are provided later), it is removed from the majority class with the aim of removing redundant data. Given this process aims at undersampling the majority class without removing useful information, removing the closest item (instance) to a randomly selected model prevents the deletion of useful information from the search space.

The initialisation, test and diffusion phases of SDS are expanded with more details to shed more light as to how SDS is adapted for the purpose of undersampling. In the initialisation phase each agent is assigned to a hypothesis from the search space (i.e. a random instance number from the majority class). Subsequently, in the *test phase*, a randomly selected micro-feature (*one of the attributes of the instance*) is compared against the corresponding micro-feature of the model (i.e. the corresponding attribute of the model); if the randomly selected micro-feature of the hypothesis is within the threshold of the model's micro-feature, the agent is set to active, otherwise inactive (threshold vector calculation is described in Section 3.1). This process is repeated for all the agents, after which all agents are either active or inactive. Once the status of all the agents are determined in the test phase, the next phase starts. In the *diffusion phase*, each inactive agent randomly picks another agent; if the randomly selected agent is active, its hypothesis (i.e. instance number) is shared with the inactive agent, otherwise the selecting agent picks a random hypothesis (a random instance number) from the search space.

The cycle of test-diffusion phases are repeated equal to the number of iterations allowed. Then the instance which has attracted the largest number of agents is labelled as the '*closest match*' and thus removed from the search space. The model is then transferred to another list 'models list'. In the next step, another model is randomly chosen from the remaining instances, its closest match is found and removed from the search space and the new model is then added to the 'models list'. Once the sum of the size of the models list and the remaining search space reaches the number of interest (i.e. when the majority class is downsized), the undersampling process is terminated.

3.1 Feature Dependent Threshold Vector

There are two types of features or attributes in the datasets (i.e. continuous and discrete). Depending on their types, feature's threshold is calculated accordingly and separately (for each feature). For continuous features, the thresholds are found by calculating the median values (excluding the zeros) of the difference between the values of the features.

Following the same analogy, for the discrete features, the threshold is calculated using the following formula:

$$\tau_i = \frac{1}{n-1} \tag{1}$$

where τ_i is the threshold of feature i, and n is the number of discrete values. Therefore, τ returns the value of the 'gap' between each neighbouring discrete value.

Using the method described for calculating the threshold vector, $\boldsymbol{\tau}$, the algorithm can perform an evaluation as to whether any two selected values from the same feature can be considered 'adjacent' values.

Therefore, using $\boldsymbol{\tau}$ during the test phase for each agent, the proximity of the instances can be partially evaluated (though each individual feature comparison). SDS has shown in many other applications, that after several iterations, it is capable of finding the closest match, which can then be removed as part of the undersampling process. This process guarantees that while the most similar item is removed from the search space, the model, which represents the deleted item is kept and used later during the classification process. This process is repeated until the dataset is completely undersampled to the desired size.

4 Results

This section summarises the result of applying SDS (i.e. SDS-SVM) to undersample the majority classes of all the datasets used. In order to fairly evaluate the performance of the proposed model, various performance measurements have been used (i.e. G-mean, F-measure, AUC, accuracy, sensitivity and specificity).

The summary of the results are shown in Table 2 where the results of SVM classifying after random undersampling (RND-SVM) and SDS undersampling (SDS-SVM) are reported and contrasted against other methods. Table 1 shows in which of the datasets SMOTE is used to oversample the minority class as with RND-SVM and SDS-SVM.

The results in Table 2 shows that in most cases SDS-SVM is outperforming RND-SVM. In order to investigate the reason behind this different of performance, the redundancy at instance and feature levels will be discussed in the next section.

Considering the existing literature, there have been other relevant experiments on the same datasets (see Table 2); for example, Zhang and Li [34] implemented a positive biased nearest neighbour algorithm (PNN) on real-world datasets including the Oil Spills and the Vehicle datasets. The proposed model gave the best results when compared against K-nearest neighbour algorithm, other sampling method including SMOTE as well as the general method for making a classifier cost sensitive known as MetaCost. The model has been evaluated using AUC and PNN gave the best results with AUC equal to 0.847 for the Oil Spills dataset and 0.983 for the Vehicle dataset. In both instances SDS-SVM outperforms PNN.

Guo and Viktor [18] proposed a new model that combines boosting, data generation and an ensemble based learning algorithm (DataBoost-IM). The model was evaluated using F-measure, G-mean and Accuracy using seventeen imbalanced datasets including: Ionosphere, Hepatitis, Abalone, Yeast, Oil spills and Breast Cancer datasets. For each datasets the model was compared with C4.5, AdaBoostM1, DataBoost, CSB2, AdaCost [30] and SMOTEBoost [14]. The proposed model scored high on highly imbalanced datasets in terms of the F-measure and is comparable (in some instances higher) with other models when it comes to G-mean and Accuracy. The authors have also found that the DataBoost-IM is not biased toward one class while giving a high overall accuracy for both classes. This algorithm has been outperformed by the model proposed in this paper.

In another research, Chawla and Tang [28] evaluated four different variations of SVM on seven datasets including; Oil spills, Abalone and Yeast. The authors found that GSVM-RU is the best and it outperforms other models in term of both effectiveness and efficiency.

The next section aims at exploring the behaviour of the algorithm proposed for undersampling in this paper. The aim is to provide some insight as to where it would be recommended to use SDS-SVM.

5 Discussion

5.1 Analysing instance and feature levels redundancy

It is intuitive that having a dataset with a high level of duplication would mean that picking a randomly selected instance and removing it as part of the undersampling process is less likely to cause the removal of important information. This hypothesis is clearly demonstrated with two of the datasets used (i.e. Yeast, and Breast Cancer) where there are duplications at instance level making all the features of some samples identical with some others. Table 3 (No. of Duplicates) shows the duplications (percentage of duplicate instances) for both of these datasets.

While this justifies the outperformance of RND-SVM over SDS-SVM, this neither justifies the outperformance of RND-SVM in Bank Marketing nor offers a strong reason for the outperformance of SDS-SVM in all the remaining datasets. For this purpose the redundancy at the feature level is explored as shown in Table 3 where the number of repetition in each feature is calculated and then the median, mean and standard deviation of all the feature repetitions are taken into account. Considering these figures, a link can be established between a high

Table 2: Results for the datasets

		G-mean	**AUC**	**F-measure**	**Accuracy**	**Sensitivity**	**Specificity**
Oil Spills	RND-SVM	35.27%	0.648	69.61%	56.27%	**100.00%**	12.44%
	SDS-SVM	**98.74%**	**0.999**	**98.74%**	**98.74%**	99.58%	97.92%
	DataBoost-IM [18]	67.70%	NA	55.0%	96.60%	46.30%	**98.90%**
	PNN [34]	NA	0.847	NA	NA	NA	NA
Yeast	RND-SVM	**91.43%**	**0.969**	**90.86%**	91.26%	**94.00%**	88.94%
	SDS-SVM	90.33%	0.965	89.74%	90.11%	94.00%	86.81%
	DataBoost-IM [18]	66.9%	NA	58.0%	**97.3%**	45.00%	**99.90%**
	GSVM-RU [28]	NA	0.845	68.8%	NA	NA	NA
Abalone	RND-SVM	88.69%	0.951	89.29%	88.62%	**91.43%**	88.00%
	SDS-SVM	**89.83%**	**0.957**	**89.39%**	**89.77%**	91.11%	88.57%
	GSVM-RU [28]	86.5%	NA	60.4%	NA	NA	NA
	DataBoost-IM[18]	61.1%	NA	45.0%	94.6%	38.0%	**98.1%**
Vehicle	RND-SVM	**98.45%**	0.995	**98.46%**	**98.45%**	99.06%	97.85%
	SDS-SVM	**98.45%**	**0.999**	**98.46%**	**98.45%**	**99.37%**	97.54%
	DataBoost-IM[18]	95.7%	NA	93.7%	97.0%	93.4%	**98.1%**
	PNN [34]	NA	0.983	NA	NA	NA	NA
Breast Cancer	RND-SVM	**97.70%**	**0.996**	**97.71%**	**97.70%**	**98.33%**	97.08%
	SDS-SVM	95.81%	0.972	95.77%	95.83%	97.07%	94.58%
	DataBoost-IM [18]	96.40%	NA	95.2%	96.70%	95.40%	**97.3%**
Bank Marketing	RND-SVM	**92.91%**	0.972	**93.43%**	93.06%	97.05%	88.95%
	SDS-SVM	90.96%	0.966	91.46%	91.07%	94.04%	88.00%
	HybridDA [3]	NA	**0.98**	NA	**96.73%**	**97.93%**	**94.82%**
Thoracic Surgery	RND-SVM	71.69%	0.755	70.51%	71.82%	68.88%	74.63%
	SDS-SVM	**73.51%**	**0.767**	**72.78%**	**73.59%**	**71.94%**	**75.12%**
	Boosted SVM [36]	65.7%	NA	NA	NA	60.00%	72.00%
Ionosphere	RND-SVM	94.01%	0.979	93.77%	94.15%	**97.69%**	90.48%
	SDS-SVM	**95.32%**	**0.986**	**95.27%**	**95.31%**	96.03	94.62%
	CSB2 [30]	93.00%	NA	89.7%	82.90%	96.5%	89.7%
	DataBoost-IM [18]	92.3%	NA	91.2%	94.0%	87.3%	**97.7%**
Hepatitis	RND-SVM	91.02%	0.960	90.62%	91.21%	**93.55%**	88.57%
	SDS-SVM	**91.98%**	**0.963**	**91.47%**	**92.42%**	87.10%	**97.14**%
	CSB2 [30]	80.9%	NA	63.4%	80.6%	81.3%	80.5%
	DataBoost-IM [18]	76.2%	NA	62.6%	83.8%	65.6%	88.6%

level of similarity (duplications) between the features (e.g combination of median (or average) and standard deviation) and the performance of SDS-SVM. For instance in the case of the Oil Spills dataset, the median repetition of 81.47% and the standard deviation of 32.61% indicate a varying level of duplications across various features, which leads to the the outperformance of SDS-SVM which partially evaluates the instances. In terms of the Bank Marketing where there are no duplications of instances, there is a high level of duplication at feature level with median of 99.72% and standard deviation of only 4.14% which justifies the good performance of RND-SVM. In all other cases (Oil Spills, Abalone, Vehicle, Thoracic Surgery, Ionosphere and Hepatitis), where feature-level duplications are not high, and there are no large standard deviations (causing a larger level of oscillations), SDS-SVM is a recommended method to use.

As can be seen, feature-level duplication analysis also cater for the instance-level duplication analysis, thus providing a better insight on which of the two algorithms to use.

Table 3: Instance and Feature duplication rates

	Instance Level Figures	**Features Level Figures**			**Best Model**
Datasets	**No. of Duplicates**	**Median**	**Average**	**Standard Deviation**	
Oil Spills	0	81.47%	68.65%	32.61	**SDS-SVM**
Yeast	25 (5.39%)	91.25%	92.54%	4.61	RND-SVM
Abalone	0	61.62%	58.01%	33.57	**SDS-SVM**
Vehicle	0	94.12%	88.46%	13.43	**SDS-SVM**
Breast Cancer	231 (50.43%)	95.19%	95.26%	0.2	RND-SVM
Bank Marketing	0	99.72%	98.48%	4.18	RND-SVM
Thoracic Surgery	0	99.50%	94.90%	10.63	**SDS-SVM**
Ionosphere	0	4.88%	7.67%	6.3	**SDS-SVM**
Hepatitis	0	97.01%	81.30%	26.15	**SDS-SVM**

5.2 Investigating agents behaviour

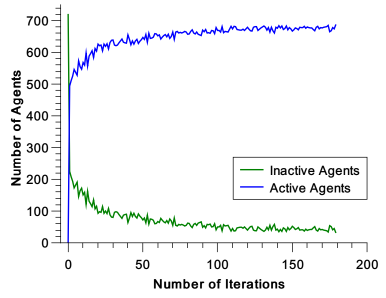
Convergence of agents is defined as the number of iterations needed for the agents to form a stable population of active agents [2]. SDS algorithm adopted for the purpose of undersampling is responsive towards feature-level duplications and when there are many duplications at feature level, the number of active agents is higher; this is illustrated in the graphs of Fig. 1, where the Bank dataset with the high feature-level duplications is shown (on the left) as opposed to the Ionosphere dataset (on the right) where the feature-level duplication is much lower (with the median of 4.88% and the standard deviation of 6.3). The oscillating behaviour of the population's activity is attributed to the micro-feature evaluation of each of the agents. In other words, if an agent picks a certain micro-feature and becomes active, it is likely that other agents are attracted towards the hypothesis of that agent, thus adopting the same hypothesis (but a randomly selected and likely different micro-feature); if the newly selected micro-feature is not within the threshold, this would lead to the agent inactivity for the next cycle. This mechanism assists the agents to only maintain their activities when a hypothesis is (in most of its micro-feature selections) within the calculated threshold.

One interesting feature of the algorithm is that high activity level of the population does not always correspond to convergence to a single instance. While this in itself is a useful feature for identifying more (than one) similar instances, this characteristics is yet to be experimented in the future work. Also it would be worthwhile to explore whether each trial (i.e. removal of one similar instance from the search space) could be terminated depending on the activity level of the populations.

5.3 Search Space Coverage

As stated before, one of the important aspect of SDS algorithm is partial function evaluation which manifests itself in the micro-feature evaluation of the search space. It is known that swarm intelligence algorithms are mostly used when dealing with large search spaces where neither pre-existing knowledge about the problem space exists nor it is possible to visit all the elements of the search space. In case of the SDS algorithm and its partial function evaluation, while SDS might visits each instance 'briefly' (i.e. checking one of few features), it

Fig. 1: Convergence of agents over the iterations allowed for the Bank datasets (left) and the Ionosphere datasets (Right)



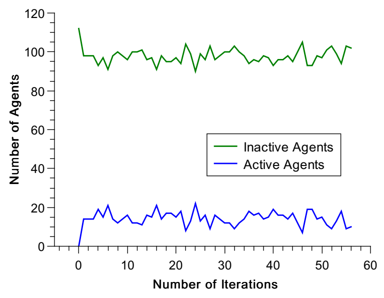
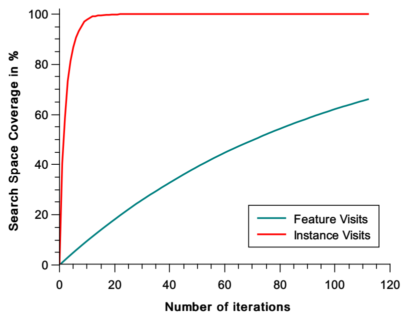
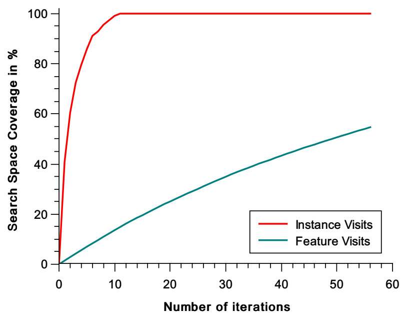


Fig. 2: Search space coverage for the Oil Spills dataset (left) and the Ionosphere dataset (right)





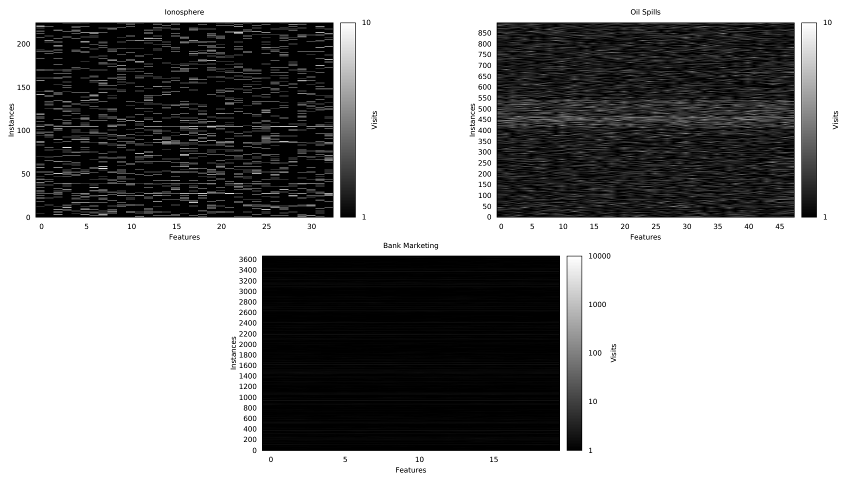
does not run a greedy comparison on all of the instance's features. Therefore, the agent aims to 'form an idea' before spending further computational time (by itself or by attracting other agents).

This behaviour of the agents can be summarised in two sets of experiments: the first would be to explore the percentage of the instances visited by the SDS agents, and the second is to calculate the percentage of all features visited from the whole of the dataset (i.e. all the features of all the instances). The results of these two experiments are shown in the graphs of Fig. 2. It is shown that while the empirically chosen values for the number of agents and the iterations suffice in visiting the instances at least once, not all the features are (or need to be) visited.

In another experiment, the frequency of agents visiting each feature is explored, investigating the distribution of agents' exploration capability in the search space with the ultimate goal of finding the closest match. For this purpose, three datasets with varying degrees of feature-level duplications are chosen and the results are illustrated in Fig. 3.

For instance, in the case of the Bank Marketing dataset where the duplication is very high the median of 99.72% and standard deviation of 4.14%, it is shown

Fig. 3: Frequency of visiting individual features in three datasets.



the agents are converged to the closest matches (showing themselves as stripe of while lines) while in the case of the Oil Spills and Ionosphere, the agents presence is distributed in the search space. Also in terms of the Oil Spills dataset, instances in position 450 to 550 are attracting more agent visits which is attributable their similarity to the model which is located at position 471.

6 Conclusion and Future Work

This paper proposes a model which uses a swarm intelligence algorithm (Stochastic Diffusion Search or SDS) which is assigned to perform the undersampling of the majority classes in imbalanced datasets. Acknowledging the role of removing exact duplicates (redundancies) of instances from the majority on the outcome of the classifiers, the proposed SDS algorithm conducts its process at the feature-level (attribute-level) of the datasets. This algorithm demonstrates to be effective in identifying feature-level duplications which are shown to be also playing an important role in the performance of the classifier.

This work presents an analysis of both instance and feature levels redundancies and establishes a link between the feature-level duplications and the role of feature-level undersampling mechanism. This analysis is accompanied by investigating the behaviour of the agents through their activity level during the undersampling process. It is shown that the agents activity is directly proportional to the level of redundancy in the datasets (not only at the instance level, but more importantly, the feature level).

Another investigation carried out in this work is the ability of the algorithm to comprehensively explore the search space without having to greedily investigate all the features of all the instances in the dataset.

As part of the future research, various coverage percentages could be explored and thus associating the coverage percentage with the termination criteria. This might shed light on the 'bare essential' coverage needed before removing an instance. Another ongoing study is being conducted on the link between the agents activity level and the termination criteria as well as the possibility of removing more than one instance from the dataset where the agents share a 'similar interest' in multiple instances. At last but not least, during the undersampling process, there is a gradual decrease in the size of the search space. Therefore, by establishing a more dynamic population size and iteration numbers, the coverage of the search space could be made homogeneous throughout the undersampling process.

References

1. Akbani, R., Kwek, S., Japkowicz, N.: Applying support vector machines to imbalanced datasets. In: Machine learning: ECML 2004, pp. 39–50. Springer (2004)
2. al-Rifaie, M.M., Bishop, J.M.: Stochastic diffusion search review. Journal of Behavioral Robotics 3, 155–173 (2013)
3. Alhakbani, H. A., a.R.M.M.: Handling class imbalance in direct marketing. dataset using a hybrid data and algorithmic level. solutions. IEEE (2016)
4. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM Sigkdd Explorations Newsletter 6(1), 20–29 (2004)
5. Batuwita, R., Palade, V.: Class imbalance learning methods for support vector machines. Imbalanced learning: Foundations, algorithms, and applications 83 (2013)
6. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. In: Robots and Biological Systems: Towards a New Bionics?, pp. 703–712. Springer (1993)
7. Bhattacharyya, S.: Handbook of Research on Swarm Intelligence in Engineering. IGI Global (2015)
8. Bishop, J.: Stochastic searching networks. In: Proc. 1st IEE Conf. on Artifical neural networks. pp. 329–331 (1989)
9. Bishop, J.: Stochastic searching networks. In: Proc. 1st IEE Conf. on Artificial Neural Networks, London, UK. pp. 329–331 (1989)
10. Burez, J., Van den Poel, D.: Handling class imbalance in customer churn prediction. Expert Systems with Applications 36(3), 4626–4636 (2009)
11. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 853–867. Springer (2005)
12. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 875–886. Springer (2009)
13. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research pp. 321–357 (2002)
14. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: Smoteboost: Improving prediction of the minority class in boosting. In: European Conference on Principles of Data Mining and Knowledge Discovery. pp. 107–119. Springer (2003)
15. Ganganwar, V.: An overview of classification algorithms for imbalanced datasets. vol. 2, pp. 42–47. Citeseer (2012)
16. Grech-Cini, H.: Locating Facial Features. Ph.D. thesis (1995)

17. Grech-Cini, H., McKee, G.T.: Locating the mouth region in images of human faces. In: Optical Tools for Manufacturing and Advanced Automation. pp. 458–465. International Society for Optics and Photonics (1993)
18. Guo, H., Viktor, H.L.: Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. ACM SIGKDD Explorations Newsletter 6(1), 30–39 (2004)
19. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: Advances in intelligent computing, pp. 878–887. Springer (2005)
20. Hsu, C.W., Chang, C.C., Lin, C.J., et al.: A practical guide to support vector classification (2003)
21. Hurley, S., Whitaker, R.M.: An agent based approach to site selection for wireless networks. In: Proceedings of the 2002 ACM symposium on Applied computing. pp. 574–577. ACM (2002)
22. Kennedy, R.L.: Solving data mining problems through pattern recognition (1997)
23. Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Machine learning 30(2-3), 195–215 (1998)
24. Kubat, M., Matwin, S., et al.: Addressing the curse of imbalanced training sets: one-sided selection. In: ICML. vol. 97, pp. 179–186. Nashville, USA (1997)
25. Lesperance, Y., Wagnerg, G., Birmingham, W., r Bollacke, K., Nareyek, A., Walser, J.P., Aha, D., Finin, T., Grosof, B., Japkowicz, N., et al.: Aaai 2000 workshop reports. AI Magazine 22(1), 127 (2001)
26. Ling, C.X., Li, C.: Data mining for direct marketing: Problems and solutions. In: KDD. vol. 98, pp. 73–79 (1998)
27. Nasuto, S., Bishop, M.: Convergence analysis of stochastic diffusion search. Parallel algorithms and application 14(2), 89–107 (1999)
28. Tang, Y., Zhang, Y.Q., Chawla, N.V., Krasser, S.: Svms modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39(1), 281–288 (2009)
29. Thai-Nghe, N., Gantner, Z., Schmidt-Thieme, L.: Cost-sensitive learning methods for imbalanced data. In: Neural Networks (IJCNN), The 2010 International Joint Conference on. pp. 1–8. IEEE (2010)
30. Ting, K.M.: A comparative study of cost-sensitive boosting algorithms. In: In Proceedings of the 17th International Conference on Machine Learning. Citeseer (2000)
31. Weiss, G.M.: Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter 6(1), 7–19 (2004)
32. Weiss, G.M., McCarthy, K., Zabar, B.: Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? DMIN 7, 35–41 (2007)
33. Williams, H., Bishop, M.: Stochastic diffusion search: A comparison of swarm intelligence parameter estimation algorithms with ransac. Algorithms 7(2), 206–228 (2014)
34. Zhang, X., Li, Y.: A positive-biased nearest neighbour algorithm for imbalanced classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 293–304. Springer (2013)
35. Zhu, G.q., Liu, S.r., Yu, J.s.: Support vector machine and its applications to function approximation. Journal of east china university of science and technology 28(5), 555–559 (2002)
36. Zieba, M., Tomczak, J.M., Lubicz, M., Swiatek, J.: Boosted svm for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. Applied soft computing 14, 99–108 (2014)