Noname manuscript No.
(will be inserted by the editor)

Framework for Extreme Imbalance Classification

SWIM: Sampling WIth the Majority Class

**Colin Bellinger · Shiven Sharma ·
Nathalie Japkowicz · Osmar Zaïane**

Received: date / Accepted: date

Abstract The class imbalance problem is a pervasive issue in many real-world domains. Oversampling methods that inflate the rare class by generating synthetic data are amongst the most popular techniques for resolving class imbalance. However, they concentrate on the characteristics of the minority class and use them to guide the oversampling process. By completely overlooking the majority class, they lose a global view on the classification problem and, while alleviating the class imbalance, may negatively impact learnability by generating borderline or overlapping instances. This becomes even more critical when facing extreme class imbalance, where the minority class is strongly underrepresented and on its own does not contain enough information to conduct the oversampling process. We propose a framework for synthetic oversampling that, unlike existing resampling methods, is robust on cases of extreme imbalance. The key feature of the framework is that it uses

C. Bellinger
Data Science for Complex Systems
National Research Council of Canada
Ottawa, Canada
E-mail: colin.bellinger@nrc-cnrc.gc.ca

S. Sharma
Weather Telmatics Inc.
Ottawa, Canada
E-mail: shiven.cheema@gmail.com

N. Japakowicz
Department of Computer Science
American University
Washington D.C, USA
E-mail: japkowic@american.edu

O. Zaïane
Alberta Machine Intelligence Institute, Department of Computer Science
University of Alberta
Edmonton, Canada
E-mail: zaiane@ualberta.ca

the density of the well-sampled majority class to guide the generation process. We demonstrate implementations of the framework using the Mahalanobis distance and a radial basis function. We evaluate over 25 benchmark datasets, and show that the framework offers a distinct performance improvement over the existing state-of-the-art in oversampling techniques.

Keywords Machine learning · Imbalanced classification · Extreme imbalance · Synthetic oversampling · SMOTE

1 Introduction

In this paper, we address classification problems involving extreme class imbalance. We define extreme imbalances as having both an exceptional imbalanced ratio between the classes (over 1:1000), as well as a very low absolute number of minority class instances in the training set (often fewer than 20). These challenging properties appear in many important classification domains, such as gamma-ray spectral classification [17], fraud detection,[21], failure prediction [18], *etc.* In general, synthetic oversampling has shown to be very effective for managing class imbalance, and as a result, has received a large portion of the research focus in recent years [6, 7, 4]. These algorithms, however, fail on domains involving extreme imbalance. In spite of the importance of domains involving extreme imbalance, there remains a dearth of research into means of ameliorating performance on extremely imbalanced classification problems.

In our previous work, we postulated that in cases of extreme imbalance, there is insufficient information encoded in the minority class to employ standard techniques for synthetic oversampling. Under these circumstances, we demonstrated that superior performance improvements are achieved by employing a majority-focused strategy for generating synthetic minority training examples. We denote this majority focused strategy by the moniker SWIM: Sampling WIth the Majority [16]). In particular, we proposed an algorithm that generates synthetic minority training examples that are *a)* near to their minority seed, and *b)* have the same Mahalanobis distance to the mean of the majority class as their seed. This ensures that the synthetic instances do not spread into denser regions of the majority class where there is no statistical evidence that they should be.

The intuition behind SWIM is that *a*) the synthetic minority instances should be generated in regions of the data-space that have similar densities with respect to the majority class as the real minority instances, and *b*) that they should be generated in regions that neighbour the real minority instances. Specifically, instead of asking: *given the minority class data, where should the new minority class instances be generated*, we ask: *given the majority class data and the relative position of the minority class instances, where should new minority class instances be generated.* In this work, we generalize SWIM by offering a non-parametric alternative to the Mahalanobis distance, which uses radial basis functions. Moreover, we discuss how SWIM and SMOTE can be formed into a pipeline to combine their relative strengths.

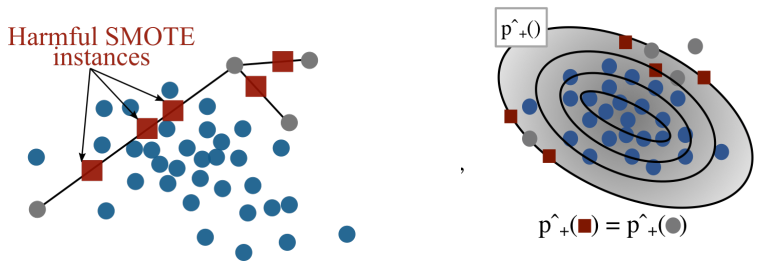


Fig. 1: Illustration of the SMOTE procedure generating synthetic minority instances (red squares) inside the majority class (blue circles) on the left, and the SWIM framework using the relative density of the minority class instances (grey circles) to guide the generation process and avoid erroneous regions of the majority class (right).

The SWIM framework is presented on the left in Figure 1 and contrasted with SMOTE on the right. SMOTE is the standard approach to synthetic oversampling. It generates new minority instances by interpolating points at random distances between nearest neighbours in the minority class [6]. In this sense, SMOTE is a minority-focused approach, which we contrast with our majority-focused approach. The minority-focused approach ignores the majority class, thereby rendering it susceptible to generating samples far from the local neighbourhood of the minority class seeds, and in high-density regions deep inside the majority class. This risk increases with the severity of the imbalance. Post-hoc cleaning by removing Tomek links and the Edit Nearest Neighbour Rule have been proposed to deal with this [19,22]. In practice, their effectiveness can be limited and hard to predict. This is exacerbated in cases of extreme imbalance.

Alternatively, the image on the right in Figure 1 demonstrates that the instances generated by SWIM (red squares) are in the same neighbourhood as their minority class seeds (grey circles), and that they are in regions of the data-space with the same density with respect to the majority class as their minority class seeds ($\hat{p}_+(GENERATED) = \hat{p}_+(SEED)$)[1]. The density contours are shown as black rings in the image.

We evaluate SWIM and five other state-of-the-art methods for resampling on 24 benchmark datasets of extreme imbalance. Our performance analysis shows that SWIM provides greater performance enhancements, in terms of the geometric mean (g-mean), than the existing state-of-the-art in oversampling techniques on extremely imbalanced domains.

2 Related Work

In this work, we focus on supervised binary classification problems over highly imbalanced domains. The process of binary classification utilizes a training set

[1] In practice, we soften the equality to be less than or equal to.

$X_{n\times m} \in \mathbb{R}$ and corresponding labels $Y_n \in \{0, 1\}$. The objective is to induce a function, $f(x_i) \rightarrow y_i$, that maps the training instances $x_i \in X$ to their corresponding class labels $y_i \in Y$. This problem is made more challenging in imbalanced domains where there are far fewer examples of the minority class X_{min}, $y = 1$ than of the majority class, X_{maj} $y = 0$. This has been shown to cause the induced classifier $f(\cdot)$ to become biased towards the larger class, thus leading to poor performance [11].

Two paradigms exist for dealing with imbalanced classification problems. When the minority class is rare or unavailable, one-class classification is applied. However, binary learning quickly becomes advantageous as the number of instances increases [5]. This has motivated research into extending the usefulness of binary classifiers to increasingly imbalanced domains. This is done via re-sampling the training data, cost-adjustment and/or algorithm modification [8,20]. Re-sampling is the most commonly applied of these techniques; it is often favoured because it has been shown to produce robust performance gains, and can be applied with any classifier. In this paper, we focus on re-sampling approaches.

The most basic re-sampling strategies are Random UnderSampling (RUS), and Random OverSampling (ROS). These balance class distributions in the training set by randomly discarding instances of the majority class, and/or by randomly replicating instances of the minority class. These strategies, however, suffer from the loss of information and the risk of overfitting, respectively.

To avoid these shortcomings, and to expand the regions of the data-space occupied by the minority training instances, the Synthetic Minority Oversampling TEchnique was proposed (SMOTE) [6]. It produces a balanced training set by interpolating synthetic instances between nearest neighbours in the set of minority class instances in the training set. This procedure relies entirely on the minority class training instances; the outcome is that the resulting synthetic data is situated within the convex-hull formed by the minority class. Because the majority class is disregarded, the convex-hull may overlap majority class. In such circumstances, applying SMOTE can actually degrade performance. This outcome becomes increasingly likely with greater class imbalance.

In order to address this weakness in SMOTE, more recent methods have incorporated the majority class into the re-sampling process, or used it to clean the re-sampled training data after SMOTE is applied. Cleaning techniques include the removal of Tomek links and the Edit Nearest Neighbour rule [19]. The re-sampling process has been altered to account for the class density around the minority class instances. This is the case with Adaptive Synthetic Oversampling (ADASYN), borderline SMOTE, and Majority Weighted Minority Oversampling Technique [9,10,3]; the only majority class information used is that which is present within the local neighbourhood of the generated sample. In these methods, the distribution of the minority class remains the key component of the generative process. Consequently, an insufficient number of minority samples will negatively impact the generative process.

In addition to the SMOTE-based methods that rely on the Euclidean distance to the k-nearest neighbours, Abdi *et. al* [1] proposed the use of the Mahalanobis distance (MD) for synthetic minority oversampling. The fundamental distinction with our method is that they do not utilize the majority class information. Rather, they generate synthetic samples using the MD calculated on the small, and potentially error prone, minority class training set; new samples are generated at the same MD as a reference minority point from the minority class mean. Therefore, this method is susceptible to failure due to the limitations of the dearth of minority class data in the training set, as the estimated mean and covariance matrix would be unrepresentative of the latent minority distribution. Radial basis functions have previously been demonstrated to work well for oversampling in case of moderate class imbalance [12]. This approach utilizes the difference between the density in the minority and majority classes to identify safe locations from which to generate samples. As with the previously discussed methods, this method limited due to its reliance on the minority class.

At their core, all current state-of-the-art oversampling methods still rely on the representativeness of the minority class instances to produce a beneficial synthetic set. Alternatively, our method does not make any assumptions regarding what the minority class represents, except where existing samples are positioned with respect to the majority class. The information for generating synthetic samples comes from the populous majority class, and thus, our method is effective for classification problems in which the minority class is rare, a situation that is both common and of great importance [13].

3 SWIM Framework

Instead of relying on the position and distance between minority class instances, the SWIM framework utilizes the density of each minority class instance with respect to the distribution of the majority class in order to determine where to generate synthetic instances. This is abstractly described in Algorithm 1. The key components of SWIM are the density estimation and the shift procedures. Any appropriate method for density estimation can be applied. In the subsequent sections we present the use of the Mahalanobis distance and the radial basis function for this purpose. The shift function takes a minority seed and serves to generate a synthetic instances that is close to the seed and has approximately the same density.

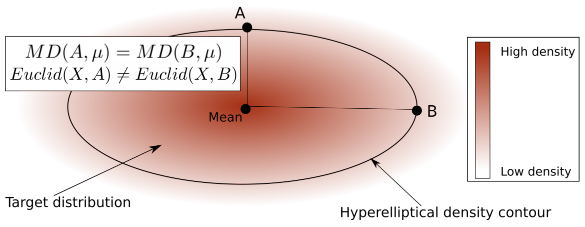


Fig. 2: Illustration of the Mahalanobis distance between two points **A** and **B** from the mean. Both points have the same Mahalanobis distance, but different Euclidean distances from the mean.

Algorithm 1 SWIM_Framework(A,B,n)

```
Input:
$A$, Majority class instances.
$B$, Minority class instances.
$n$, Number of minority samples to generate.
Output:
$B'$, $n$ synthetic minority samples.
Method:
 1: $\widehat{pdf_A} \leftarrow A$: Estimate a density function from $A$.
 2: $d \leftarrow \widehat{pdf_A}(B)$: Record the minority class densities.
 3: while more samples do
 4:     $b \leftarrow B_i$: select a random minority instance.
 5:     $d \leftarrow d_i$: get its density $w.r.t.\ A$.
 6:     $b' \leftarrow shift(b, d)$: shift $b$ to a neighbouring region with density $d$.
 7:     $B' \leftarrow [B', b']$: add $b'$ to $B'$
    return $B'$
```

3.1 Mahalanobis Oversampling

The Mahalanobis distance (MD) provides a fast and efficient means of implementing the SWIM framework. The MD calculates the distance between the mean of the majority class distribution, and a minority class seed point. The calculation accounts for the density along the path between the mean and seed points. Thus, two points have the same MD from the mean if they lie on the same hyperelliptical density contour. This is contrasted with Euclidean distance in Fig. 2. Given a minority seed, according to the SWIM framework, any point in the nearby regions of the data-space with the same MD can be sampled as a synthetic minority training instance.

The calculation of the MD involves knowing the mean μ and the covariance matrix Σ of the distribution. In practice, however, the parameters are estimated as $\overline{\mu}$ and $\overline{\Sigma}$ on a sample population. Larger, more representative sets, such as is typical in the majority class training data, produce better estimates of these parameters.

Line 1 of Algorithm 1 for SWIM$_{MD}$ involves applying some preprocessing steps to simplify the data generation procedure and the estimation of $\overline{\mu}$ and $\overline{\Sigma}$ from the majority class instances. The steps are as follows:

Step 1 **Centre the majority and minority classes**: Centring the data simplifies the calculation of the distances; this will be evident in the first step of the generation process. Let $\mu_{\mathbf{a}}$ be the feature mean vector of the majority class A. We centre the majority class to have $\mathbf{0}$ mean, and then centre the minority class with mean vector of the majority class:

$$\begin{aligned} A_c &= A - \mu_{\mathbf{a}} \\ B_c &= B - \mu_{\mathbf{a}} \end{aligned} \tag{1}$$

Step 2 **Whiten the minority class**: Let $\mathbf{\Sigma}$ denote the estimated covariance matrix of A_c, and $\mathbf{\Sigma}^{-1}$ denote its inverse. $\mathbf{\Sigma}^{-\frac{1}{2}}$ is the square root of $\mathbf{\Sigma}^{-1}$. We whiten the centred minority class as:

$$B_w = B_c \mathbf{\Sigma}^{-\frac{1}{2}} \tag{2}$$

The MD is equivalent to the Euclidean distance in the whitenend space of a distribution. Thus, by whitening, we simplify the calculations for generating synthetic data. This simplifies the shift procedure.

Step 3 **Find feature bounds**: These are used to bound the spread of the synthetic samples. For each feature f in B_w, we find its mean μ_f and standard deviation σ_f. We then calculate an upper and lower bound on its value, u_f and l_f, as follows:

$$\begin{aligned} u_f &= \mu_f + \alpha\sigma_f \\ l_f &= \mu_f - \alpha\sigma_f \end{aligned} \tag{3}$$

$\alpha \in \mathbb{R}$ controls the number of standard deviations we want the bounds to be. Therefore, larger α values cause a greater amount of spread along the corresponding density contour.

Lines 3 - 7 of Algorithm 1 are repeated until the user-specified number of synthetic samples are generated. In practice, we repeat this until the classes are balanced. Within the loop, we select a minority staple b_i at random and apply the shift procedure which returns a synthetic instances. The shift procedure for SWIM$_{MD}$ is carried out as follows:

Step 1 **Generate new samples**: For each feature f, we generate a random number between u_f and l_f. Thus, we obtain a sample point, b'_i in the whitened space, where each feature $b'_{i,f}$ is $l_f \leq b'_{i,f} \leq u_f$. We generate a sample that is at the same Euclidean distance from the mean of the majority class[2]. Since we centred the data, this implies that the new sample will have the same Euclidean norm as the minority seed b_i. Therefore, we transform b'_i as:

$$b_i^{norm} = s\frac{||b_i||_2}{||b'_i||_2} \tag{4}$$

[2] This takes advantage of the whitening done in the preprocessing, as instead of dealing with the MD, we can use the Euclidean distance.

Step 2 **Scale sample back to original space**: b_i^{norm} exists in the whitened space of the minority class, with the same Euclidean distance from the mean vector $\mathbf{0}$ as b_i in the whitened space. We now have to transform the point back into the original space. This is done as:

$$b_i'' = (\mathbf{\Sigma}^{-\frac{1}{2}})^{-1} b_i^{norm}, \tag{5}$$

where the synthetic sample b_i'' will be in the same density contour as its minority seed instances b_i.

As the method involves the computation of matrix inverses, if there are linearly dependent columns, the calculations will fail. To handle this case, we check the rank r of the majority class A. If $r < d$, where d is the dimensionality of A, then we calculate the QR-decomposition of A. The non-zero values of the resulting upper-triangular matrix correspond to the linearly independent columns of A. Using the steps outlined above, we can then oversample and classify the data in the sub-space defined by the features represented by these columns.

3.2 Radial Basis Oversampling

To implement a non-parametric form of SWIM, we propose the use of a radial basis function (RBF) with a Gaussian kernel to estimate the local density of the minority class samples with respect to the majority class. The Gaussian kernel takes a distance r and a smoothing parameter ϵ and is computed as:

$$\phi(r) = \exp -(\epsilon r)^2 \tag{6}$$

For ϵ values closer to zero, the result is a flatter, wider basis function, whereas as $\epsilon \to \infty$ all of the weight is placed at the sample points. The latter results in a distribution with density spikes at each majority class point. Typically, the optimal choice of ϵ is a value close to zero.

As a reasonable default, we set the shape parameter, ϵ to be $\epsilon = \alpha\sigma d$, where d and σ are the mean and standard deviation of the distance between points in A, and $\alpha = -0.5$. This keeps the parameter relatively small, and ensures it to be increasingly small for increasingly sparse data. This causes the function to be smoothing, leading to better generalization in practice.

To estimate a score for minority instance $b \in B$ with respect to the majority class A that is analogous to a density, we sum over the RBFs between b and each $a_i \in A$ instance in the majority class:

$$rbf\_score(b) = \sum_{i=1}^{|A|} \phi(||a_i - b||_2) \tag{7}$$

For sample generation, the rbf_score can be used to find regions of the data-space that neighbour the minority seed b and have similar estimated densities. In practice, we consider any region of the data-space with an equal

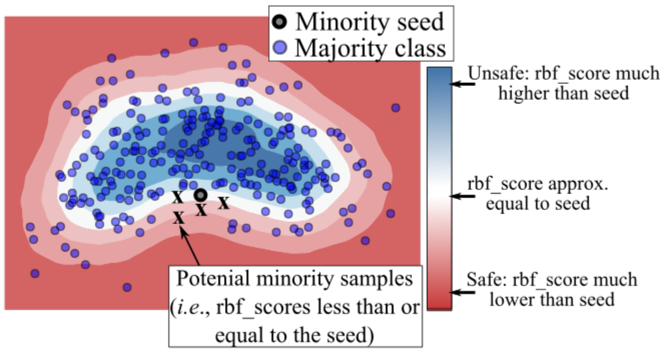


Fig. 3: Example of potential regions from which to draw synthetic minority samples

or lower rbf_score as being a candidate location for a synthetic minority instances.

The regions of the data-space that are shaded white in Figure 3 have the same density as the seed, and the areas shaded red have lower densities. The white and red regions form the set of potential points from which to generate synthetic minority samples. We constrain this by dictating that the samples be dawn from close to the seed.

The SWIM$_{RBF}$ implementation of Algorithm 1 does not require learning a model of the majority class instances A in line 1. However, we do apply some initial processing that is worth noting. In particular, we standardize the data to have mean zero and unit standard deviation, and calculate the mean and standard deviation of the distances between majority class instances. These are utilized in setting the ϵ parameter and the step size in the Gaussian jitter used in the shift function. At this point, a ball or KD Tree can also be construct to improve the efficiency of estimating the density.

The values of the vector d in line 2 of Algorithm 1 are calculated with the RBF approach by applying the rbf_score to each minority instance $b_i \in B$. Given a minority seed b_i, the shift procedure for the RBF implementation of SWIM is shown in Algorithm 2. The process involves sampling the data-space around the seed to find points with approximately the same density as the seed. In particular, for a given seed, b_i, a synthetic sample b'_i is produced by applying Gaussian jitter to the seed in line 2, and calculating the RBF density at the candidate sample point in line 3. This is repeated until a candidate sample with $b'_{score} \leq b_{score}$ is found. However, we limit the number of attempts to a fixed size. If after $maxAttempt = 5$ tries no point is sampled from in the neighbourhood of the seed with equal or lower density than the seed, then the seed itself is replicated. In practice, however, the max will not be reached if the step size is small and ϵ is sufficiently small.

The generation process is repeated by randomly selecting minority instances and generating a new synthetic instance until classes are balanced or a user-specified number of instances are produced.

Algorithm 2 shift$_{RBF}$(b, b_{score} A, σ)

Input:
A, Majority class instances.
σ, Standard deviation for Gaussian jitter.
b, Minority seed instance.
b_{score}, Estimated density of the seed, b, w.r.t A.
Output:
b', A synthetic minority sample.
Method:

```
1: repeat
2:     $b' \leftarrow b + \mathcal{N}(0, \sigma)$: Apply random jitter to $b$.
3:     $b'_{score} \leftarrow rbf\_score_A(b')$: Estimated density of $b'$ $w.r.t.$ $A$
4: until $b'_{score} \leq b_{score}$ & $attempts \leq maxAttempts$
5: if $attempts \leq maxAttempts$ then
6:     $b' \leftarrow b$
7: return $b'$
```

3.3 SWIM Demonstration

3.3.1 RBF Density Estimation

As previously discussed, the ϵ parameter of radial basis function controls the smoothness of the density distribution. As a result, ϵ is the parameter that impacts where SWIM$_{RBF}$ draws its set of penitential synthetic samples from. Although we find that setting the ϵ value based on the euclidean distance between majority class instances works well in practice, it is possible to optimize this using cross-validation.

We provide Figure 4 in order to assist readers in understanding the impact of setting the ϵ parameter. This figure shows the resulting density distributions estimated from the majority class samples (red squares) for ϵ values $\{1, 1.25, 1.5, 1.75, 2\}$. The effect is that the density distribution is smoother and less moon-shaped for $\epsilon = 1$, whereas, $\epsilon = 2$ has more variation in the density and a more exaggerated moon-shape.

By qualitatively examining these plots it appears that a good ϵ value is likely between 1.5 and 1.75[3]. The density distribution for this range of ϵ matches the moon-shape well, whilst the density drops off a moderate rate away from dense regions of the majority class. Alternatively, with the smaller ϵ values the shape is not well matched, and with a larger ϵ density scores quickly drops away from the majority class samples.

3.3.2 Resampling and Classification

Figure 6 presents a comparison of the decision surfaces produced by SVM classifiers on the moon-shaped imbalanced dataset without resampling (top left), with SMOTE (top right), with SWIM$_{MD}$ (bottom right) and with SWIM$_{RBF}$

[3] Using our default setting based on the mean distance between majority class samples, ϵ is set to 1.68.

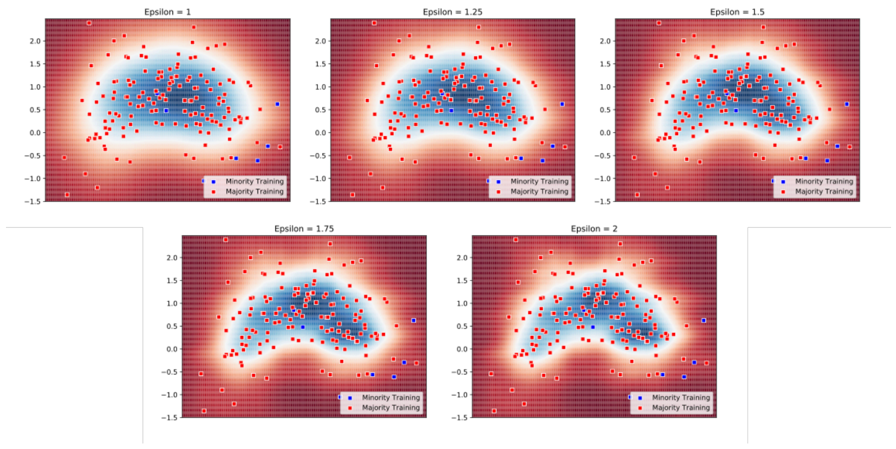


Fig. 4: This figure demonstrates how the increase in the epsilon parameter affects the smoothness of the density function on a toy dataset.

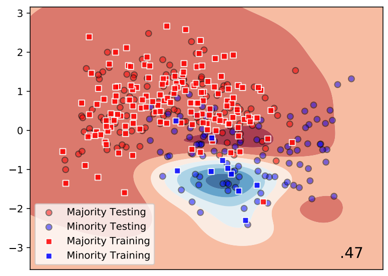
(bottom left). This figure depicts both the training, testing and synthetic instances. This is done in order to paint a clear picture of whether each resampling method generates synthetic instances in regions that are reflective of the underlying distribution. In addition, this enables the reader to see if the synthetic samples have a positive impact on the induced decision surface (*i.e.,* is the decision surface learned on the training data a good match to the test data?) The minority instances are blue and the majority instances are red in each plot. The training instances are plotted as squares and the testing instances are plotted as circles. The score in the bottom right corner of each figure shows the geometric mean on a test set.

This overlay provides evidence regarding the merit of each resampling method. The plots highlight how the different means of generating synthetic samples impact the learned decision surfaces. In particular, they demonstrate that SWIM produces a smoother synthetic minority distribution than SMOTE and that the distribution spreads in directions that have a more beneficial impact on the learned decision boundary.

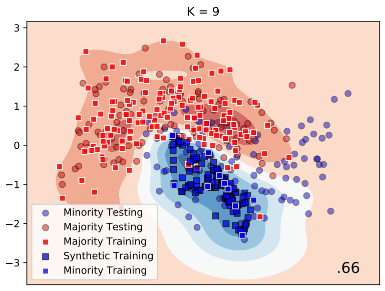
4 Experiments

4.1 Set-up

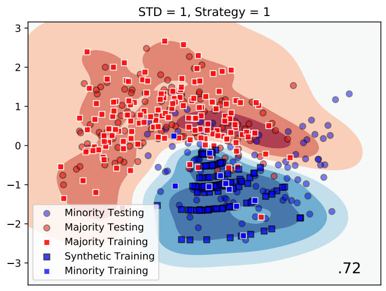
We compare the SWIM$_{MD}$ and SWIM$_{RBF}$ implementations of the SWIM framework to the state-of-the-art resampling methods for class imbalance. The latter includes SMOTE, SMOTE with the removal of Tomek links (SMOTE+TL), SMOTE with edit nearest neighbours (SMOTE+ENN), borderline SMOTE (Bord), adaptive synthetic sampling (ADASYN), and no resampling (None). For each of the SMOTE-based methods K=5 and K=7 were used.



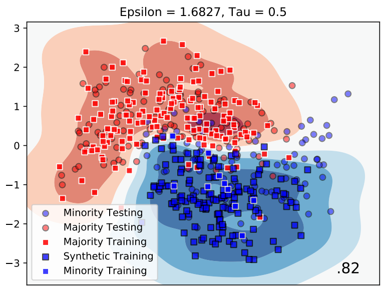
SVM decision surface without resampling. This produced a test geometric mean of 0.47.



SVM decision surface with SMOTE resampling. This produced a test geometric mean of 0.66.



SVM decision surface with SWIM$_{MD}$ resampling. This produced a test geometric mean of 0.72.



SVM decision surface with SWIM$_{RBF}$ resampling. This produced a test geometric mean of 0.82.

Fig. 6: This figure demonstrates how the decision surface (shown as the red-blue gradient) of an SVM classifier changes as a result of resampling with SMOTE, SWIM$_{MD}$ and SWIM$_{RBF}$ on a toy dataset.

We set $\alpha = 0.5$ and test $\epsilon = \{0.5, 1, 3, 5\}$ for SWIM$_{RBF}$. SWIM$_{MD}$ has a single parameter $\alpha \in \mathbb{R}$, which controls the potential step size from the seed. We tested $\alpha = \{0.25, 0.5, 1\}$.

In each experiment, we test the following base classifiers: naïve Bayes, k-nearest neighbours, decision trees, support vector machines, and multi-layer perceptron classifiers. For each classifier, the default settings from the Python Scikit-learn [15] are used. From the list base classifiers, we select the best coupling in classifier with each of resampling method for each domain and perform our analysis is performed using these sets ups.

The evaluation is performed on 24 benchmark datasets (see Table 1) from the KEEL repository [2]. From each KEEL dataset, D, we created 6 sub-problems in which the minority class size is artificially down-sampled to 3, 5, 10, 15, 20, 30. Therefore, we ran experiments on $24 \times 6 = 156$ imbalanced classification problems. This down-sampling strategy enables us to study the

Table 1: This table lists the dimensionality of each dataset used in the evaluation, along with the number average number of majority class training instances, and the imbalance ratio for training sets with 5,15 and 30 minority class instances.

Dataset	Dimensions	$\lvert Maj\rvert$	$5/\lvert Maj\rvert$	$15/\lvert Maj\rvert$	$30/\lvert Maj\rvert$
abalone9-18	8	342	0.015	0.043	0.058
bands	19	114	0.044	0.133	0.263
coil2000	85	4621	0.001	0.003	0.006
ecoli1	7	131	0.038	0.114	0.236
ecoli3	7	142	0.035	0.098	0.113
glass1	9	68	0.074	0.234	0.405
heart	13	82	0.061	0.190	0.380
ionosphere	33	116	0.043	0.134	0.283
mammographic	5	207	0.024	0.068	0.141
new-thyroid1	5	90	0.056	0.165	0.163
poker 8,9 vs 6	10	733	0.007	0.019	0.021
ring	20	1860	0.003	0.008	0.016
segment0	19	987	0.005	0.015	0.031
sonar	60	55	0.091	0.288	0.517
spambase	57	1383	0.004	0.011	0.021
spectfheart	44	102	0.049	0.138	0.267
vehicle0	18	322	0.016	0.047	0.091
vehicle1	18	316	0.016	0.047	0.092
vehicle3	18	315	0.016	0.047	0.094
vowel0	13	446	0.011	0.033	0.067
wdbc	30	180	0.028	0.088	0.164
wisconsin	9	214	0.023	0.070	0.134
yeast3	8	660	0.008	0.023	0.044
yeast4	8	660	0.008	0.023	0.044

behaviour of each resampling algorithm on increasingly extreme levels of absolute and relative imbalance. We randomly repeat each sub-problem 30 times and record the average performance in terms of the g-mean.

The g-mean provides a combined assessment of accuracy on the target and the outlier class in a single value [14]. Given the accuracy on the target class a^+ and the accuracy on the outlier class a^-, the g-mean for a classification model f on test set X is calculated as: $g - mean_{f(X)} = \sqrt{a^+ \times a^-}$.

4.2 Results

4.3 Rank Analysis

Table 2 shows the number of times each resampling method led to the best performing classifier for experiments with increasingly extreme levels of imbalance. The resampling methods are listed in the first column. Each of the remaining columns correspond to the sub-problems of training on 3, 5, 10, 15, 20, and 30 minority instances for each of the 24 datasets. The table shows that the SWIM framework (particularly SWIM$_{RBF}$) dominates on the more extreme levels of imbalance. For minority training sizes 20 and 30, the SWIM framework remains competitive, but the results are more mixed.

Table 2: Number of times each resampling method was best across all datasets on each minority training set size.

	Minority Training Size					
Method	3	5	10	15	20	30
SWIM$_{RBF}$	12	13	12	12	6	7
SWIM$_{MD}$	7	3	1	1	5	5
ADASYN	3	2	3	3	4	2
Bord	0	0	1	1	2	1
SMOTE	1	2	2	2	4	1
SMOTE+TL	1	3	2	2	3	6
SMOTE+ENN	1	1	2	2	0	2
None	0	1	2	2	1	1

Figure 7 presents box plots of the ranks of each resampling method on the six imbalanced sub-problems. These plots provide additional evidence of the superiority of the SWIM framework on highly imbalanced problems. In particular, they show that SWIM$_{RBF}$ and SWIM$_{MD}$ have the best average ranks, and that low variances in their ranks relative to the other methods. This is most notably the cases for up to minority training sizes of 15.

We employed the Friedman test to asses the statistical significance of the rankings for each sub-problem and found that the null hypothesis can be reject at an α value of 0.05 for minority training sizes of 3,5,10,15. The Nemenyi post-hoc test finds a statistical significance between SWIM$_{RBF}$ and all methods except SWIM$_{MD}$ for minority training sizes 3,5,10.

4.4 Relative Performance Analysis

The bar plot in Figure 8 shows the relative difference in g-mean produced by the best SWIM system versus the best alternative for each data set with minority training sizes 3,5,10 and 20. We have left out sizes 15 and 30 to ensure the presentation remains clear. Nonetheless, the trend is consistent across the omitted sizes. The dataset names are stated on the y-axis and the difference g-mean$(SWIM)$ − g-mean(Alt) are plotted against the x-axis. Bars to the right show better performance for SWIM, and those to the left indicate better performance for the alternative resampling method. In order to clarify the performance trends, the datasets are sorted according to the relative differences over minority training sets of size 3.

These results once again demonstrate that SWIM performs well on all training sizes, but has a stronger advantage over more extreme imbalance. In addition to the number of wins, it also wins by a larger margin on these datasets. The tables showing the g-mean scores for each method are included in the appendix.

In the comparison between the use of MD versus RBF implementations of SWIM, the trend is that RBF has an advantage over MD on the more extreme cases of imbalance. This is demonstrated in Figure 8. It shows the performance of the best classification systems involving SWIM$_{RBF}$, SWIM$_{MD}$

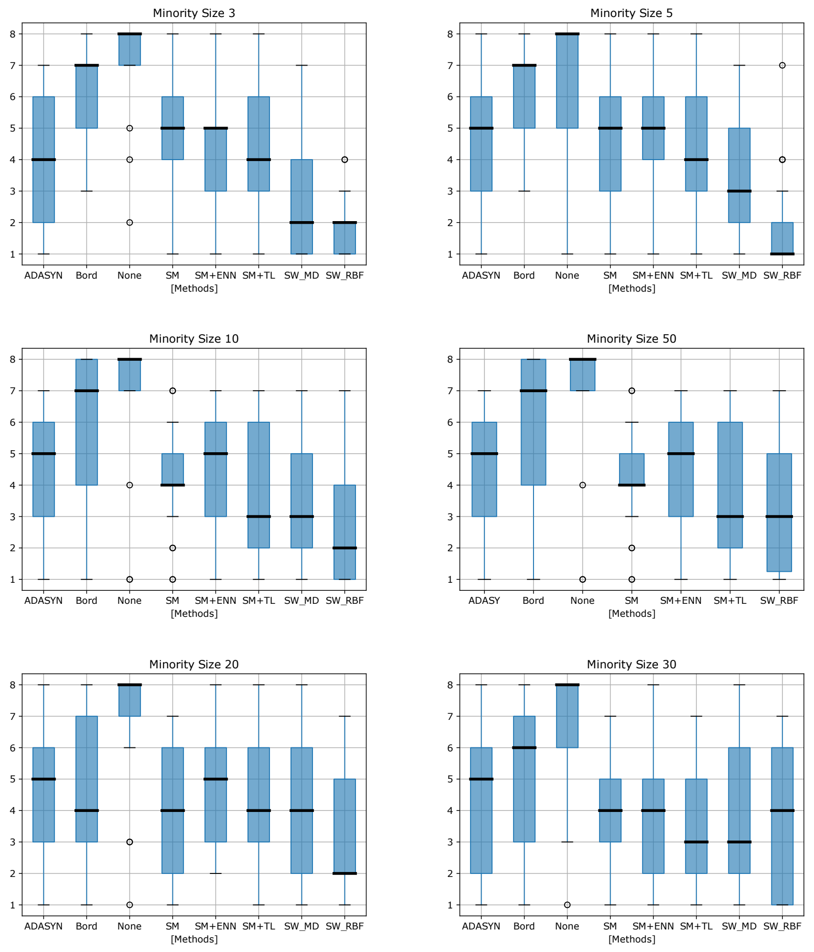


Fig. 7: Boxplots showing rank of each resampling method over all of the datasets in each sub-problem of minority training sizes 3,5,10,15,20 and 30. In these figures SW indicates SWIM and SM indicates SMOTE. The y-axis shows the ranks over which each method is distributed and the x-axis specifies the resampling method.

and None on the sub-problems with minority training sizes 20 and 3. In general, preprocessing with either SWIM beats the baseline on all cases for size 3 and all but one cases for size 20. With respect to the SWIM implementations, SWIM$_{RBF}$ is best on 17 out of 25 datasets for minority training size 3, and 13 out of 25 times for minority training size 5.

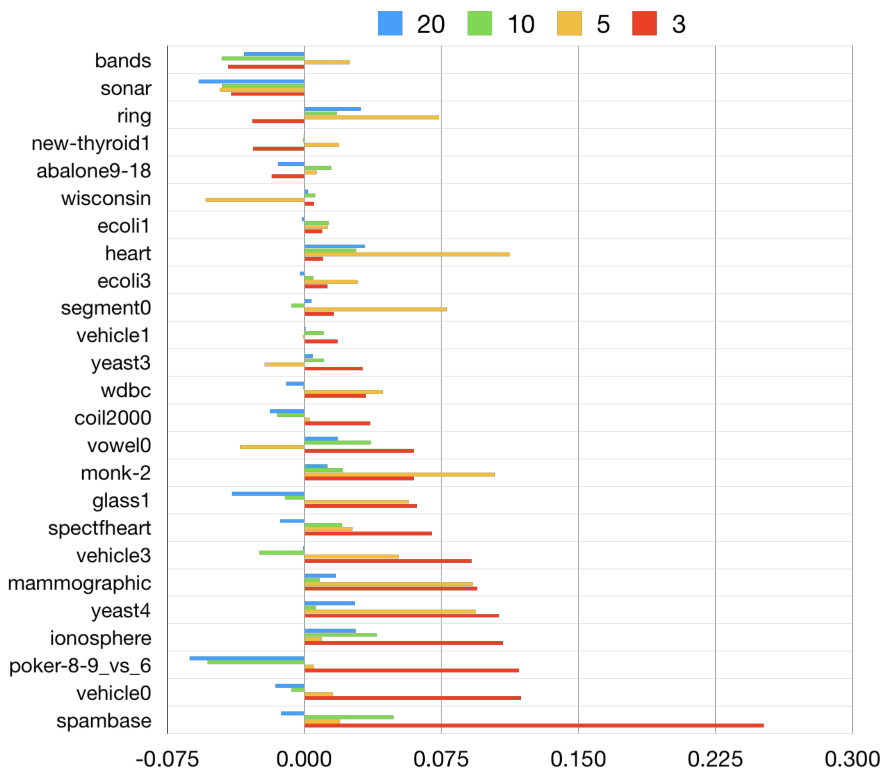


Fig. 8: Bar plots highlighting the difference in g-mean for the best SWIM system versus the best alternative resampling system (Alt best) on each dataset with minority training sizes 3, 5, 10 and 20. The differences are calculated as g-mean($SWIM$) − g-mean(Alt) and displayed on the x-axis. Each dataset is specified on the y-axis, and the colours of the bars indicate the minority training size. Positive values indicate better performance by SWIM, where as negative values indicate better performance by the alternative method.

4.5 Performance Curves

Figure 9 displays the performance curves for each resampling method on the monk-2 and vowel0 datasets. The plots show the minority training size on the x-axis and the average g-mean on the y-axis. These represent typical examples of how the relative performances of the resampling method evolve with the changing level of imbalance. In particular, we see that SWIM generally has a strong advantage on the extreme levels of imbalance, and the performances of the resampling methods begin to converge around 20 or 30 minority samples.

5 Discussion

Using PCA analysis, we are able to identify categories of datasets where SWIM works best, and others where the SMOTE-based alternatives are strong. Fig-

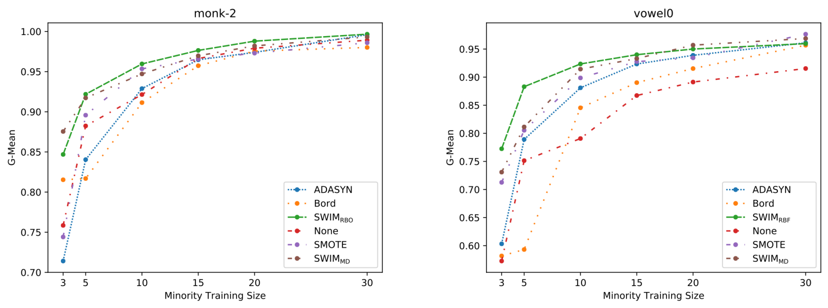


Fig. 9: Performance curves showing the relationship between g-mean and the number of minority training instances on the monk-2 (left) and vowel0 (right) datasets for each resampling method.

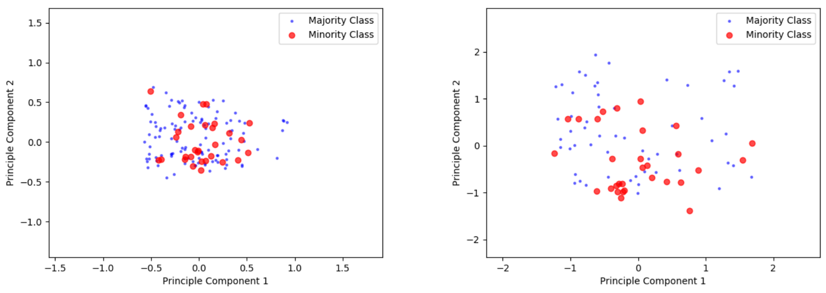


Fig. 10: PCA plots for the Band and Sonar datasets.

ure 8 reveals that the SMOTE-based alternative is better on the Bands and Sonar datasets at all levels of imbalance. The PCA plots for these datasets are presented in Figure 10.

Through our empirical analysis, we have found that The SMOTE-based strategy (particularly with some cleaning) have an advantage over datasets in which the minority class has a single cluster of points that are close together. In these cases, linear interpolation is an effective sampling bias. Heavy overlap of the single minority cluster with the majority class is also a property that appears in the datasets where SMOTE is generally strong. In these cases, generating in dense regions of the majority space that neighbour the minority seeds is essential to improving the classification performance. Alternatively, the SWIM methods avoid higher density areas of the majority class space. Therefore,on domains with these properties, classifiers induced on data from SWIM undergeneralize. It is possible to relax this property of the SWIM algorithm, but we expect that this would often be harmful.

SWIM produces good results on most datasets that do not have the aforementioned properties. These represent a large majority of the datasets in our study, and in real-world applications. During our analysis, we have identified,

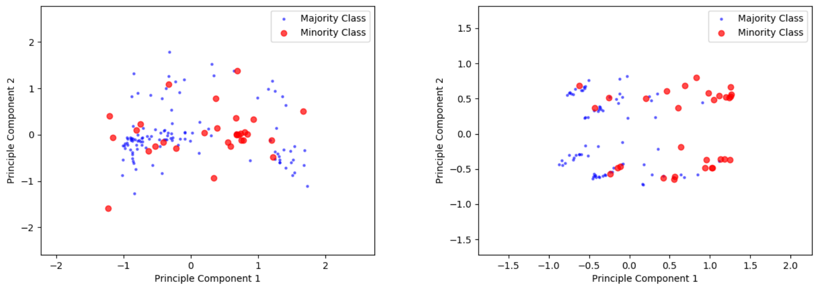


Fig. 11: PCA plots for the Ionosphere and heart datasets.

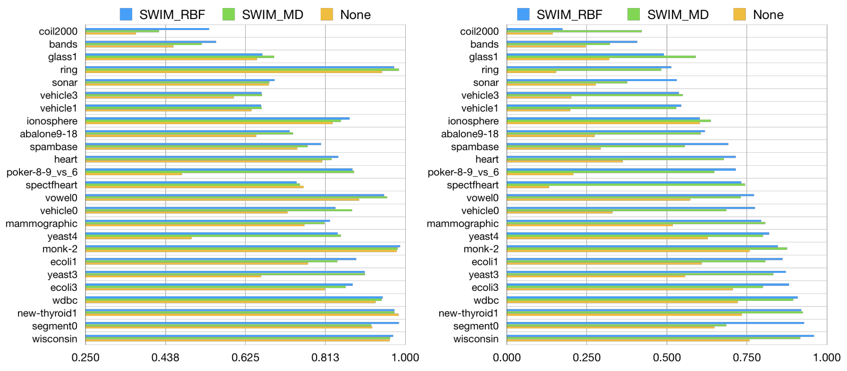


Fig. 12: Comparison of using RBF versus Mahalanobis implementations of SWIM. The plot shows the g-mean for datasets with minority training sizes 20 and the right plot is for minority training sizes of 3.

however, that SWIM is particularly strong on datasets in which the minority class data is spread out, and potentially has many clusters. We also note that SWIM performs well when the majority class density is non-uniform between the minority class instances. This is the case in the Ionosphere and heart datasets. The PCA plots of these are presented in Figure 11.

We compare the performance of SWIM$_MD$ to SWIM$_RBF$ in Figure 12. These show that on certain cases, SWIM$_MD$ has a strong advantage of SWIM$_RBF$. In conjunction with these results, we preformed the same empirical analysis using PCA plots to examine the properties of the datasets where SWIM$_MD$ works better that SWIM$_RBF$. This confirmed our hypothesis that SWIM$_RBF$ would have a strategic advantage on datasets with more complex, non-Gaussian majority classes, such as Segment0 and spambase. Alternatively, SWIM$_MD$ performs very will on datasets, such as Coil2000 and Monk-2 and Glass, where the distribution is closer to a Gaussian. We present an example of PCA plot for each of these in Figure 13.

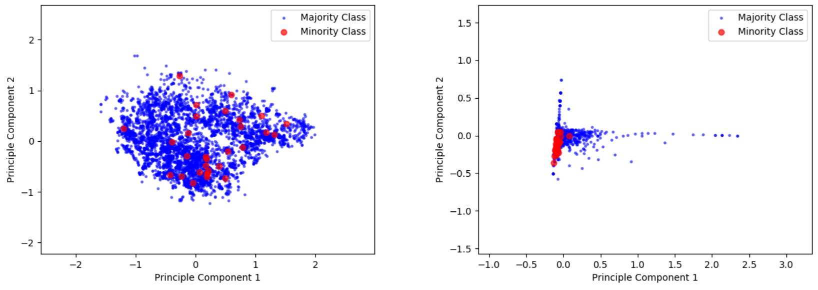


Fig. 13: PCA plots for the coil2000 and spambase datasets.

5.1 Resampling Parameter

The parameters in many resampling methods can have a significant impact on the performance of the induced classifiers. SMOTE, SWIM$_{MD}$ and SWIM$_{RBF}$ each have parameters (K, α, σ) that impact the spread of the synthetic instance through the data-space. With respect to SMOTE, K is an integer in the range of 1 and one minus the size of the minority class ($K \in \{1, |B| - 1\}$). Increasing value of K allows synthetic samples to be generated between minority class seed and more of its distant minority class neighbours. Larger K-values cause the induced classifier to generalize a larger area of the data-space for the minority class. Setting K too large, however, can lead to synthetic instances being generated deep inside the majority class space (as we previously demonstrated in Figure 1.)

Similarly to the K value in SMOTE, increasing α parameters in SWIM$_{MD}$ and the σ value in SWIM$_{RBF}$ increases the spread of the synthetic samples, and therefore the generalization of the induced classifier. The key difference is that the SWIM algorithm will only generate synthetic instances in regions of the data-space with similar densities as the minority seeds. Therefore, unlike SMOTE, there is no risk of spreading synthetic samples deep inside the majority class. Because the SWIM algorithm will not generate samples in harmful regions, using a naive setting of it is less likely to have a negative impact on classifier performance than the K parameter in SMOTE.

Another advantage of the real-valued spread parameter in SWIM in comparison to SMOTE is that small adjustments in the parameters of SWIM cause smooth, small changes in the distribution of the synethetic samples. Alternatively, increasing or decreasing the K vale of SMOTE by one can have a drastic impact on the distribution of the synthetic instances. This is particularly the case in domains with extreme imbalance. As a result, the performance of classifiers induced on data including synthetic instances from SWIM is consistent across small changes in the spread parameters, whereas classifier performance may change drastically with the value of K.

5.2 Run-time Complexity

An advantage of the SMOTE algorithm is that it has a relatively low run-time complexity. For the basic algorithm, it requires finding the k-nearest minority class neighbours of each minority class instance. For a single minority class instance, its k-nearest neighbours can be found in $\mathcal{O}(kn)$, where n is the number of sample, and k is the number of neighbours. This is repeated n times (once for each minority class seed), and then synthetic samples can be generated a between the seed and a random neighbour in constant time.

In comparison, the SWIM algorithms are composed of two algorithms; the modelling step (Algorithm 1) and the shift step (Algorithm 2), which generates a sample from a minority seed.

To efficiently implement SWIM$_{RBF}$, the first step involves constructing a ball tree from the majority class instances. This can be done in $\mathcal{O}(n \log n)$. Conversely to SMOTE, the n in this case represents the number of instances in the majority class, which is much larger. Once the tree is constructed, each rbf_{score} of each candidate synthetic sample is calculated in $\mathcal{O}(n \log n)$ time. In the the worst case, this is repeat for $m \times maxAttempts$ times, where m is the number of synthetic samples needed, and $maxAttempts$ is the maximum number of times we can attempt to satisfy the condition $b'_{score} \leq b_{score}$ in line 4 of Algorithm 2.

The run-time complexity of SWIM$_{MD}$ is dictated by the matrix inversion and multiplication operations. Let a be the number of instances in the majority class, and b be the number of instances in the minority class. Let d be the dimensionality of the data, and $n = a + b$ be the total number of instances available for training. The complexities of the computation of the mean vector, covariance matrix and the inverse of the covariance matrix are $\mathcal{O}(ad)$, $\mathcal{O}(ad^2)$ and $\mathcal{O}(d^{2.37})$ respectively. The centring step has a complexity of $\mathcal{O}(n)$, whereas the computation of the square root and the whitening operation have $\mathcal{O}(d^3)$ and $\mathcal{O}(bd^2)$ time complexity. Finding the feature bounds has $\mathcal{O}(d)$ complexity, and sample generation has $\mathcal{O}(btd)$ complexity, where t is the number of samples to generate for each minority class sample. Finally, scaling back the generated samples to the original space involves matrix multiplication with the square root of the inverse, and thus the operation has a complexity of $\mathcal{O}(btd^2)$.

Although the run-time complexities of the SWIM algorithms are slightly greater than SMOTE, they come with the benefit of better results on case of extreme imbalance. We argue that it is a necessary trade-off to achieve good result on highly imbalanced problems. Moreover, it is only performed once during the preprocessing stage, and thus, has a relatively minor impact on the overall learning cost.

5.3 SWIM-SMOTE Cascade Ensemble

Given that SWIM has an advantage on the more extreme cases of imbalance, and SMOTE is strong on the less extreme imbalance, a natural question is

Table 3: Number of times each resampling method was best across all datasets on each minority training set size.

	Minority Training Size					
Method	3	5	10	15	20	30
SWIM	13	10	10	10	8	8
SMOTE	3	4	5	5	6	6
SWIM+SMOTE	9	11	10	10	11	11

whether they can be applied as a cascade ensemble to produce better overall results. We are specifically interested in if we can achieve performance gains by applying SWIM to address the extreme imbalance, and then applying SMOTE to the combined set of real and synthetic minority training examples. The bottom row of Table 3 shows the number of times the the ensemble cascade of SWIM$_{MD}$+SMOTE was superior to just applying SWIM$_{MD}$ or SMOTE.

The combination leads to improved performance on approximately 10 of the 25 datasets. Interestingly, these results appears to be independent of the minority class training size. This is counter to our hypothesis that the combination would be most helpful on cases of more extreme imbalance. However, more research is required on exactly how to optimize ensembles of resampling systems. We set this out as a future direction for research in class imbalance.

6 Conclusion

Extreme class imbalance occurs in a wide variety of important domains. Research related to it, however, has largely failed to design synthetic oversampling methods that are effective on such domains. To address this, we introduce a framework for synthetic oversampling, SWIM (Sampling WIth the Majority). The key advantage of SWIM versus existing methods on extreme imbalance is that SWIM utilizes the information offered by the majority class data to generate synthetic minority class instances. This enables SWIM to generate synthetic data in a manner that leads to a more general decision boundary without encroaching too deeply into the majority class. Alternatively, traditional methods, such as SMOTE, apply a minority-focused resampling strategy. This causes them to be heavily impacted by extreme imbalance, and often leads to harmful encroachments into the majority class space.

We evaluate our proposed parametric and non-parametric implementations of SWIM (SWIM$_{MD}$ and SWIM$_{RBF}$) against the state-of-the-art resampling methods on 24 benchmark datasets of extreme imbalance. Our results, based on the g-mean evaluation metric, show that classifiers trained on datasets preprocessed with SWIM generally rank better than those trained with any other method in cases of extreme imbalance, *i.e.* when datasets have fewer than 20 minority samples. In comparison between SWIM$_{MD}$ and SWIM$_{RBF}$, our results suggest that SWIM$_{RBF}$ robust on each of the evaluated minority class sizes (3 - 50). However, SWIM$_{MD}$ is comparable or better on the larger minority class sizes (30-50).

Future work will explore other methods and aim to derive insights into the relationships between their efficacy for generating samples, and the properties of the dataset. Furthermore, the integration of both SWIM and SMOTE is an exciting avenue for future work, as it harnesses the powers of both methods, with SWIM generating enough data to rectify the extreme imbalance, and then SMOTE generating more instances over a less extremely imbalanced dataset.

References

1. Abdi, L., Hashemi, S.: To combat multi-class imbalanced problems by means of oversampling techniques. IEEE transactions on Knowledge and Data Engineering **28**(1), 238–251 (2016)
2. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic & Soft Computing **17** (2011)
3. Barua, S., Islam, M.M., Yao, X., Murase, K.: Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Transactions on Knowledge and Data Engineering **26**(2), 405–425 (2014)
4. Bellinger, C., Drummond, C., Japkowicz, N.: Manifold-based synthetic oversampling with manifold conformance estimation. Machine Learning **107**(3), 605–637 (2018)
5. Bellinger, C., Sharma, S., Japkowicz, N.: One-class versus binary classification: Which and when? In: Machine Learning and Applications, 11th International Conference on, vol. 2, pp. 102–106. IEEE (2012)
6. Chawla, N., Bowyer, K., Hall, L., W.P., K.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002)
7. 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)
8. Elkan, C.: The foundations of cost-sensitive learning. In: International joint conference on artificial intelligence, vol. 17, pp. 973–978. Lawrence Erlbaum Associates Ltd (2001)
9. Han, H., Wang, W.y., Mao, B.h.: Borderline-SMOTE : A New Over-Sampling Method in Imbalanced Data Sets Learning. Advances in intelligent computing pp. 878–887 (2005)
10. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (3), 1322–1328 (2008)
11. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering **21**(9), 1263–1284 (2009)
12. Koziarski, M., Krawczyk, B., Woźniak, M.: Radial-based approach to imbalanced data oversampling. In: International Conference on Hybrid Artificial Intelligence Systems, pp. 318–327. Springer (2017)
13. Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5**(4), 221–232 (2016)
14. 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)
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
16. Sharma, S., Bellinger, C., Krawczyk, B., Japkowicz, N., Zaïane, O.: Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In: Proceedings of In IEEE International Conference on Data Mining (2018)
17. Sharma, S., Somayaji, A., Japkowicz, N.: Learning over subconcepts: Strategies for 1-class classification. Computational Intelligence **34**(2), 440–467 (2018)
18. Siers, M.J., Islam, M.Z.: Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Information Systems **51**, 62–71 (2015)

19. Tomek, I.: Modifications of CNN. IEEE Trans. System, Man, Cybernetics **6**(11), 769–772 (1976)
20. Wang, H., Gao, Y., Shi, Y., Wang, H.: A fast distributed classification algorithm for large-scale imbalanced data. In: IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain, pp. 1251–1256 (2016)
21. Wei, W., Li, J., Cao, L., Ou, Y., Chen, J.: Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web **16**(4), 449–475 (2013)
22. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics (3), 408–421 (1972)

A Tabulated Results

Table 4: G-Means for minority training size 3.

Dataset	None	SW_{RBF}	SW_{MD}	ADSN	Bord	SM	SM_{ENN}	SM_{TL}
abalone9-18	0.274	0.619	0.606	0.515	0.288	0.637	0.497	0.495
bands	0.247	0.407	0.322	0.449	0.166	0.275	0.409	0.351
coil2000	0.143	0.174	0.422	0.385	0.155	0.135	0.110	0.095
ecoli1	0.609	0.862	0.808	0.852	0.764	0.815	0.879	0.805
ecoli3	0.706	0.881	0.800	0.868	0.811	0.861	0.877	0.872
glass1	0.320	0.491	0.590	0.528	0.403	0.458	0.424	0.526
heart	0.362	0.715	0.678	0.560	0.610	0.705	0.693	0.579
ionosphere	0.603	0.602	0.636	0.528	0.447	0.449	0.513	0.436
mammographic	0.519	0.794	0.807	0.712	0.667	0.690	0.745	0.668
monk-2	0.758	0.847	0.875	0.714	0.815	0.711	0.744	0.725
new-thyroid1	0.734	0.919	0.925	0.953	0.817	0.875	0.880	0.820
poker-8-9_vs_6	0.208	0.715	0.648	0.497	0.246	0.598	0.602	0.690
ring	0.154	0.513	0.482	0.541	0.228	0.170	0.219	0.131
segment0	0.648	0.929	0.686	0.913	0.721	0.909	0.878	0.889
sonar	0.278	0.530	0.376	0.427	0.458	0.570	0.543	0.593
spambase	0.293	0.692	0.556	0.441	0.266	0.437	0.503	0.482
spectfheart	0.132	0.732	0.744	0.643	0.506	0.674	0.668	0.678
vehicle0	0.330	0.776	0.686	0.644	0.382	0.657	0.765	0.666
vehicle1	0.198	0.545	0.530	0.440	0.188	0.527	0.483	0.490
vehicle3	0.201	0.538	0.550	0.459	0.099	0.441	0.396	0.530
vowel0	0.573	0.772	0.731	0.603	0.582	0.713	0.515	0.643
wdbc	0.721	0.909	0.895	0.875	0.835	0.821	0.844	0.832
wisconsin	0.758	0.960	0.917	0.955	0.896	0.883	0.837	0.923
yeast3	0.557	0.871	0.832	0.840	0.568	0.779	0.825	0.860
yeast4	0.628	0.819	0.801	0.692	0.466	0.713	0.714	0.645

Table 5: G-Means for minority training size 5.

Dataset	None	SW$_{RBF}$	SW$_{MD}$	ADSN	Bord	SM	SM$_{ENN}$	SM$_{TL}$
abalone9-18	0.459	0.690	0.664	0.621	0.543	0.665	0.613	0.713
bands	0.458	0.485	0.447	0.422	0.376	0.531	0.447	0.466
coil2000	0.508	0.476	0.503	0.389	0.430	0.412	0.418	0.394
ecoli1	0.784	0.893	0.806	0.856	0.841	0.874	0.875	0.863
ecoli3	0.757	0.881	0.827	0.875	0.867	0.875	0.872	0.860
glass1	0.523	0.568	0.578	0.632	0.612	0.631	0.584	0.592
heart	0.655	0.769	0.763	0.744	0.749	0.756	0.724	0.741
ionosphere	0.695	0.808	0.757	0.695	0.626	0.681	0.654	0.622
mammographic	0.696	0.828	0.833	0.804	0.749	0.776	0.820	0.809
monk-2	0.882	0.922	0.917	0.840	0.817	0.844	0.874	0.896
new-thyroid1	0.942	0.976	0.948	0.977	0.906	0.879	0.917	0.949
poker-8-9_vs_6	0.254	0.822	0.768	0.683	0.760	0.844	0.854	0.847
ring	0.612	0.896	0.887	0.853	0.655	0.410	0.483	0.474
segment0	0.797	0.947	0.833	0.917	0.908	0.944	0.918	0.950
sonar	0.384	0.648	0.417	0.499	0.552	0.683	0.676	0.656
spambase	0.522	0.724	0.630	0.620	0.431	0.576	0.670	0.693
spectfheart	0.527	0.710	0.786	0.729	0.712	0.684	0.681	0.704
vehicle0	0.543	0.815	0.812	0.788	0.639	0.785	0.747	0.827
vehicle1	0.424	0.651	0.653	0.602	0.400	0.570	0.579	0.612
vehicle3	0.352	0.656	0.631	0.563	0.416	0.547	0.557	0.544
vowel0	0.751	0.883	0.811	0.789	0.593	0.775	0.742	0.805
wdbc	0.924	0.938	0.933	0.928	0.922	0.912	0.885	0.904
wisconsin	0.910	0.967	0.955	0.961	0.923	0.950	0.942	0.951
yeast3	0.630	0.901	0.891	0.815	0.793	0.885	0.859	0.837
yeast4	0.545	0.828	0.805	0.721	0.602	0.808	0.813	0.787

Table 6: G-Means for minority training size 10.

Dataset	None	SW$_{RBF}$	SW$_{MD}$	ADSN	Bord	SM	SM$_{ENN}$	SM$_{TL}$
abalone9-18	0.611	0.718	0.709	0.698	0.635	0.703	0.705	0.686
bands	0.412	0.541	0.507	0.438	0.549	0.587	0.538	0.563
coil2000	0.413	0.452	0.468	0.433	0.483	0.440	0.426	0.419
ecoli1	0.726	0.892	0.843	0.864	0.879	0.877	0.877	0.884
ecoli3	0.802	0.887	0.857	0.863	0.873	0.882	0.875	0.876
glass1	0.551	0.605	0.686	0.662	0.682	0.696	0.689	0.681
heart	0.684	0.814	0.787	0.783	0.778	0.785	0.775	0.793
ionosphere	0.782	0.837	0.827	0.792	0.777	0.797	0.808	0.822
mammographic	0.645	0.831	0.816	0.822	0.784	0.779	0.773	0.801
monk-2	0.921	0.960	0.947	0.929	0.911	0.939	0.953	0.928
new-thyroid1	0.988	0.974	0.975	0.976	0.964	0.973	0.970	0.981
poker-8-9_vs_6	0.339	0.867	0.865	0.844	0.918	0.920	0.950	0.930
ring	0.886	0.968	0.960	0.950	0.755	0.865	0.883	0.877
segment0	0.880	0.962	0.885	0.969	0.958	0.957	0.953	0.958
sonar	0.626	0.677	0.660	0.641	0.686	0.722	0.744	0.705
spambase	0.659	0.776	0.726	0.692	0.643	0.727	0.748	0.767
spectfheart	0.766	0.763	0.763	0.742	0.685	0.708	0.718	0.704
vehicle0	0.709	0.837	0.863	0.870	0.823	0.827	0.840	0.840
vehicle1	0.546	0.645	0.676	0.650	0.574	0.665	0.630	0.676
vehicle3	0.555	0.641	0.641	0.665	0.572	0.631	0.609	0.630
vowel0	0.791	0.924	0.914	0.881	0.845	0.887	0.899	0.877
wdbc	0.922	0.945	0.935	0.946	0.941	0.922	0.929	0.951
wisconsin	0.958	0.972	0.961	0.966	0.951	0.961	0.958	0.969
yeast3	0.529	0.904	0.895	0.892	0.855	0.893	0.891	0.908
yeast4	0.440	0.831	0.820	0.825	0.806	0.819	0.817	0.816

Table 7: G-Means for minority training size 15.

Dataset	None	SW_{RBF}	SW_{MD}	ADSN	Bord	SM	SM_{ENN}	SM_{TL}
abalone9-18	0.611	0.718	0.709	0.698	0.635	0.703	0.705	0.686
bands	0.412	0.541	0.507	0.438	0.549	0.587	0.538	0.563
coil2000	0.413	0.452	0.468	0.433	0.483	0.440	0.426	0.419
ecoli1	0.726	0.892	0.843	0.864	0.879	0.877	0.877	0.884
ecoli3	0.802	0.887	0.857	0.863	0.873	0.882	0.875	0.876
glass1	0.551	0.605	0.686	0.662	0.682	0.696	0.689	0.681
heart	0.684	0.814	0.787	0.783	0.778	0.785	0.775	0.793
ionosphere	0.782	0.837	0.827	0.792	0.777	0.797	0.808	0.822
mammographic	0.645	0.831	0.816	0.822	0.784	0.779	0.773	0.801
monk-2	0.921	0.960	0.947	0.929	0.911	0.939	0.953	0.928
new-thyroid1	0.988	0.974	0.975	0.976	0.964	0.973	0.970	0.981
poker-8-9_vs_6	0.339	0.867	0.865	0.844	0.918	0.920	0.950	0.930
ring	0.886	0.968	0.960	0.950	0.755	0.865	0.883	0.877
segment0	0.880	0.962	0.885	0.969	0.958	0.957	0.953	0.958
sonar	0.626	0.677	0.660	0.641	0.686	0.722	0.744	0.705
spambase	0.659	0.776	0.726	0.692	0.643	0.727	0.748	0.767
spectfheart	0.766	0.763	0.763	0.742	0.685	0.708	0.718	0.704
vehicle0	0.709	0.837	0.863	0.870	0.823	0.827	0.840	0.840
vehicle1	0.546	0.645	0.676	0.650	0.574	0.665	0.630	0.676
vehicle3	0.555	0.641	0.641	0.665	0.572	0.631	0.609	0.630
vowel0	0.791	0.924	0.914	0.881	0.845	0.887	0.899	0.877
wdbc	0.922	0.945	0.935	0.946	0.941	0.922	0.929	0.951
wisconsin	0.958	0.972	0.961	0.966	0.951	0.961	0.958	0.969
yeast3	0.529	0.904	0.895	0.892	0.855	0.893	0.891	0.908
yeast4	0.440	0.831	0.820	0.825	0.806	0.819	0.817	0.816

Table 8: G-Means for minority training size 20.

Dataset	None	SW_{RBF}	SW_{MD}	ADSN	Bord	SM	SM_{ENN}	SM_{TL}
abalone9-18	0.728	0.737	0.650	0.738	0.688	0.751	0.743	0.728
bands	0.557	0.522	0.456	0.539	0.580	0.589	0.584	0.589
coil2000	0.539	0.422	0.368	0.531	0.429	0.558	0.522	0.536
ecoli1	0.885	0.841	0.772	0.887	0.877	0.878	0.874	0.876
ecoli3	0.876	0.860	0.811	0.861	0.869	0.878	0.871	0.875
glass1	0.664	0.692	0.651	0.706	0.732	0.729	0.688	0.703
heart	0.842	0.827	0.805	0.809	0.805	0.806	0.805	0.797
ionosphere	0.869	0.849	0.829	0.821	0.841	0.840	0.824	0.854
mammographic	0.823	0.810	0.763	0.804	0.806	0.798	0.817	0.803
monk-2	0.988	0.982	0.979	0.974	0.976	0.972	0.957	0.973
new-thyroid1	0.974	0.975	0.984	0.973	0.968	0.975	0.982	0.980
poker-8-9_vs_6	0.876	0.880	0.476	0.856	0.933	0.943	0.940	0.960
ring	0.974	0.985	0.945	0.953	0.634	0.954	0.958	0.957
segment0	0.985	0.920	0.922	0.972	0.981	0.967	0.983	0.967
sonar	0.693	0.681	0.680	0.709	0.751	0.740	0.729	0.760
spambase	0.802	0.770	0.746	0.774	0.762	0.814	0.815	0.837
spectfheart	0.745	0.753	0.762	0.715	0.766	0.750	0.764	0.752
vehicle0	0.836	0.875	0.724	0.890	0.860	0.857	0.852	0.867
vehicle1	0.662	0.663	0.639	0.658	0.662	0.663	0.654	0.652
vehicle3	0.663	0.664	0.597	0.665	0.647	0.663	0.655	0.663
vowel0	0.950	0.957	0.891	0.939	0.915	0.934	0.932	0.924
wdbc	0.946	0.944	0.930	0.956	0.948	0.939	0.943	0.950
wisconsin	0.971	0.965	0.963	0.969	0.969	0.968	0.964	0.963
yeast3	0.905	0.905	0.661	0.901	0.891	0.881	0.904	0.901
yeast4	0.841	0.849	0.498	0.821	0.821	0.820	0.818	0.820

Table 9: G-Means for minority training size 30.

Dataset	None	SW_{RBF}	SW_{MD}	ADSN	Bord	SM	SM_{ENN}	SM_{TL}
abalone9-18	0.731	0.727	0.643	0.737	0.694	0.736	0.750	0.727
bands	0.573	0.545	0.526	0.558	0.599	0.590	0.606	0.608
coil2000	0.545	0.504	0.318	0.542	0.426	0.610	0.620	0.582
ecoli1	0.890	0.838	0.817	0.886	0.877	0.873	0.873	0.876
ecoli3	0.883	0.858	0.797	0.866	0.868	0.866	0.869	0.876
glass1	0.704	0.723	0.670	0.734	0.725	0.726	0.700	0.716
heart	0.838	0.822	0.821	0.819	0.800	0.818	0.809	0.815
ionosphere	0.881	0.884	0.859	0.805	0.855	0.864	0.851	0.875
mammographic	0.817	0.809	0.791	0.808	0.787	0.795	0.804	0.811
monk-2	0.997	0.993	0.989	0.996	0.980	0.986	0.968	0.981
new-thyroid1	0.977	0.979	0.986	0.976	0.968	0.978	0.977	0.973
poker-8-9_vs_6	0.882	0.890	0.490	0.846	0.935	0.955	0.934	0.965
ring	0.967	0.980	0.955	0.952	0.610	0.969	0.970	0.967
segment0	0.987	0.945	0.950	0.976	0.984	0.982	0.982	0.980
sonar	0.715	0.695	0.693	0.710	0.763	0.767	0.722	0.752
spambase	0.813	0.801	0.774	0.787	0.794	0.814	0.817	0.825
spectfheart	0.736	0.774	0.752	0.723	0.763	0.764	0.766	0.764
vehicle0	0.862	0.891	0.785	0.877	0.896	0.875	0.879	0.874
vehicle1	0.657	0.676	0.642	0.662	0.659	0.665	0.664	0.683
vehicle3	0.671	0.671	0.660	0.663	0.657	0.658	0.663	0.669
vowel0	0.960	0.969	0.915	0.961	0.957	0.965	0.969	0.976
wdbc	0.943	0.947	0.916	0.962	0.953	0.951	0.948	0.950
wisconsin	0.970	0.969	0.964	0.970	0.967	0.969	0.964	0.964
yeast3	0.906	0.909	0.715	0.894	0.899	0.895	0.904	0.909
yeast4	0.834	0.842	0.493	0.838	0.820	0.832	0.833	0.810