Feature selection for high dimensional imbalanced class data using harmony search

Moayedikia, Alireza; Ong, Kok-Leong; Boo, Yee Ling; Yeoh, William G. S.; Jensen, Richard

Published in:
Engineering Applications of Artificial Intelligence

DOI:
10.1016/j.engappai.2016.10.008

Publication date:
2017

Citation for published version (APA):
Moayedikia, A., Ong, K.-L., Boo, Y. L., Yeoh, W. G. S., & Jensen, R. (2017). Feature selection for high dimensional imbalanced class data using harmony search. *Engineering Applications of Artificial Intelligence*, *57*, 38-49. https://doi.org/10.1016/j.engappai.2016.10.008

Feature selection for high dimensional imbalanced class data using harmony search

Alireza Moayedikia[1a], Kok-Leong Ong[b], Yee Ling Boo[c], William GS Yeoh[a], Richard Jensen[d]

[a]*Department of Information Systems and Business Analytics, Deakin University, Victoria 3125, Australia*
[b]*SAS Analytics Innovation Lab, La Trobe University, Victoria 3086, Australia*
[c]*School of Business IT and Logistics, RMIT University, Victoria 3000, Australia*
[d]*Department of Computer Science, IMPACS, Aberystwyth University, Wales, UK*

Abstract

Misclassification costs of minority class data in real-world applications can be very high. This is a challenging problem especially when the data is also high in dimensionality because of the increase in overfitting and lower model interpretability. Feature selection is recently a popular way to address this problem by identifying features that best predict a minority class. This paper introduces a novel feature selection method call SYMON which uses symmetrical uncertainty and harmony search. Unlike existing methods, SYMON uses symmetrical uncertainty to weigh features with respect to their dependency to class labels. This helps to identify powerful features in retrieving the least frequent class labels. SYMON also uses harmony search to formulate the feature selection phase as an optimisation problem to select the best possible combination of features. The proposed algorithm is able to deal with situations where a set of features have the same weight, by incorporating two vector tuning operations embedded in the harmony search process. In this paper, SYMON is compared against various benchmark feature selection algorithms that were developed to address the same issue. Our empirical evaluation on different micro-array data sets using G-Mean and AUC measures confirm that SYMON is a comparable or a better solution to current benchmarks.

Keywords: Feature selection, harmony search, high-dimensionality, imbalanced class, symmetrical uncertainty

1. Introduction

The presence of imbalanced data is a problem for classification algorithms [1, 2]. An imbalanced data set is one where at least one class is under-represented compared to the others. Such data creates many challenges to the process of knowledge discovery and has many implications in real-world applications [3]. Addressing these issues brings about many good solutions, such as MIROS[2] that is used to detect the possibility of oil spilling [4], or to detect malicious activities of users in the context of network intrusion as seen in the AIDE [5] environment[3]. In this paper, we investigate the imbalanced class problem further by considering cases where the data set is also high in dimensionality (e.g. large-scale data sets [6]), thus making the problem more pronounced as the efficacy of learning algorithms is further reduced [7, 8, 9].

Different approaches have been proposed to address the imbalanced learning problem, including resampling [10, 11], one-class learning [10], cost-sensitive learning [12, 13] and feature selection [8, 9, 14]. In resampling, the two most common techniques used for the imbalanced data problem are (i) random oversampling and (ii) random undersampling [7, 9]. In the former,

[1]Corresponding author, E-mail: amoayedi@deakin.edu.au
[2]http://goo.gl/aR5elt
[3]http://aide.sourceforge.net/

random duplicates of instances from the minority class are added to the original data set, leading to longer classifier training time. With the latter, the instances from the majority classes are randomly discarded, thus the information loss [7] usually leads to sub-optimal learning outcomes.

One example of random oversampling is **S**ynthetic **M**inority **O**ver-sampling **TE**chnique (SMOTE) proposed by Chawla et al. [10, 11]. The algorithm generates artificial examples for the minority class by interpolating the current minority instances and has been shown [15] to improve classification performance over imbalanced data. Unfortunately, creating artificial examples may not always be possible, such as in critical applications like medical diagnosis tools that rely on real data for diagnosis [16]. In this case, artificial data might affect the accuracy of diagnosis adversely. Hence, solutions that do not attempt to alter the original data in the learning process remain desirable.

One-class learning is an exemplar that does not alter the original data during the learning process. It operates by classifying each instance based on a similarity threshold [10]. This approach minimises overfitting seen in other classifiers when one class is significantly over-represented than the others in the data. Consequently, one-class learners may lead to better predictive performance but that accuracy is dependent on the similarity threshold, which needs to be empirically tuned [10] to achieve the desired performance.

Another way to address the challenge of classification from imbalanced data is cost-sensitive learners. As the name suggests, these learners consider the cost of misclassification and therefore seek to minimise the likelihood of misclassifying a minority class through a cost matrix. They are also quite effective for large data sets as they concurrently minimise learning time [9] and the misclassification of minority classes.

Recently, researchers are gaining interest in using feature selection [17, 8, 18, 19] as a way to address the imbalanced class problem. Previous approaches (i.e. resampling, one-class learning and cost-sensitive learning) have focused on the samples of the training data. Feature selection on the other hand, takes a different view by shifting the focus to the features (i.e. dimensions) rather than the training examples. The key idea is to find a subset of features that optimise the contrast between classes in the data.

Within feature selection, there are three further sub-approaches: filter, wrapper and hybrid (also known as embedded). Generally, filter approaches will find a subset that is good but may not be optimal for a specific classifier [17]. Hence, wrapper [20, 8] and embedded approaches [9] were proposed to produce a more targeted feature subset. These approaches can be based on the ranking of features, where the criteria is often a loss function, e.g. the contribution of a feature to the classification rate [7, 9], or the discriminative power of features [8].

We argue that selecting features based on a loss function does not always yield the best learning outcome for the classifier. Rather, ranking features with respect to their dependency towards a class label and using that information to select the feature subset would give better performance, especially in predicting the minority class. As a result, the algorithm that we propose, called SYMON, is unique in the following ways:

- SYMON is a wrapper approach. Hence the chosen subset will be more relevant to the induced classifier [17].
- SYMON uses Symmetrical Uncertainty to rank features, giving insight to how relevant a feature is to a class label. This differs from other feature selection algorithms (addressing the imbalanced class problem) which select features based on a loss function. As seen later in the experimental results, this gives SYMON better overall performance.
- SYMON handles high dimensionality well. This is important especially when there is a large number of features and finding the best feature combination should be computationally efficient. SYMON uses Harmony Search to reduce the complexity of the search process [21], thus ensuring its relevance in practice.

- With high dimensionality, the likelihood of multiple features sharing the same rank is high. This presents a challenge as most feature selection algorithms lack a mechanism to pick the best subset from identically-ranked features. SYMON does not suffer from this issue as it incorporates vector tuning operations.

In the next section, we review recent feature selection algorithms that deal with learning from high dimensional imbalanced data. Once this context is established, we will introduce SYMON in Section 3 with discussion of the experimental results in Section 4. We then conclude with future work for SYMON in Section 5.

2. Related works

Given the specific focus of this paper, we shall only discuss the relevant literature that use feature selection in the context of tackling the imbalanced class problem in high dimensional settings.

One solution by Yin *et. al.* [8] is a variation of using Bayesian learning as a solution to the imbalanced class problem. The approach proposed by Yin *et. al.* works on the assumption that samples in the majority classes have a dominant influence on general feature selection techniques. The first step is thus to decompose classes with large examples into smaller pseudo-subclasses. Feature selection is then performed on the decomposed data, where the pseudo-subclasses balance the skew across classes, thus neutralising the influence the majority class examples have on feature selection algorithms. Their evaluation over synthetic data showed better feature selection performance once the imbalanced data has been decomposed.

Alibeigi *et. al.* [14] proposed a different approach with an algorithm called Density-based Feature Selection (DBFS). As the name suggests, features are ranked by their estimated probability density. This is done by exploring the contribution of each feature, taking into account the features' corresponding distributions over all classes and their correlations. This method has been evaluated in the context of high dimensional but low sample data sets, and is shown to be effective over well-known filter-based feature selection algorithms (e.g., Pearson Correlation Coefficient, Signal to Noise Correlation Coefficient, Chi-square and Information Gain).

Chen and Wasikowski [17] also studied the small sample imbalanced data problem. Their approach is encapsulated in an algorithm called FAST (Feature Assessment by Sliding Thresholds), which is based on the area under the receiver operating characteristic (ROC) curves generated from setting different decision boundaries for a single feature. The algorithm is inspired by the observation that most single feature classifiers set the decision boundary at the mid-point between the mean of the two classes. By moving this decision boundary, different numbers of true/false positives are obtained. In doing so, the algorithm will be able to measure which decision boundary will provide the best area under the ROC curve and then select the one that would yield the best predictive results. By computing this over all features, the algorithm will be able to select the best mix of features.

Maldonado *et. al.* [9] also considered the problem of high-dimensional and imbalanced data learning but in the context of binary classification. In this case, a family of algorithms inspired by the backward feature selection strategy in Support Vector Machines (SVM) was proposed. Different strategies of backward elimination of features were developed and used with SVM and SMOTE fitted with different loss functions: (a) standard 0-1; (b) balanced loss; and (c) predefined loss. The various algorithms were tested over six imbalanced microarray data with their algorithms showing better predictive performance over well-known feature selection algorithms (e.g. l_0, l_1 norm SVM, SVM Recursive Feature Elimination (SVM-RFE), Fisher + SVM, etc.) – while also using fewer features).

Not all feature selection algorithms for high-dimensional data work on the basis of a single ranking function or an inductive algorithm. Yang *et. al.* [20], for example, proposed to create multiple balanced data sets using random under/over-sampling from the original imbalanced

Table 1: SYMON parameters and notation definition.

Parameters	Definition	Initialization type
HMS	Harmony memory initialization	User defined
PAR_{min}	Minimum pitch adjustment rate	
PAR_{max}	Maximum pitch adjustment rate	
HMCR	Harmony memory consideration rate	
NI	Number of iterations	
r	Ripple factor	
d	Desired subset size	
t	The number of current iteration	Problem-specific
$F_s \in F$	Set of selected features	
$F_u \in F$	Set of unselected features	
F_w	Set of pair of weights and selected features	
F_s^w	Set of pair of weights and selected features	
F_u^w	Set of pair of weights and unselected features	
w	Set of weights if features	
C	Set of entire class labels	
$c \in C$	A given class label	
F	Set of entire features	
V_r	A random row selected from HM	
R_c	A randomly,generated number for comparisons with HMCR	
P_f	PAR value,generated using PAR() function for feature f	
NHV	Newly generated harmony vector	
$f_i \in F$	ith selected features	

data. Feature subsets are then evaluated over an ensemble of base classifiers that are trained over the balanced data sets. This combination of ensemble wrapper-based feature selection and multiple sampling in a unified framework is shown to perform better than other similar wrapper algorithms that only use a single inductive algorithm. This approach helps to eliminate any undesirable bias that a single inductive algorithm may have.

The prior studies showed the efficacy of their algorithms by comparing against the existing wrapper feature selectors, where they mainly consider support vector machines for classification. To test the efficacy, high-dimensional imbalanced microarray datasets have been used for experimentation with different assessment measures such as G-Mean (GM), Area Under Curve (AUC) and F-measure (F1), etc. However, the evaluation highlighted a few shortcomings. Those algorithms that are filter-based produces sub-optimal results; others manipulated the original data [20, 8], which is not desirable as discussed previously. For those feature selection algorithms that include/exclude features based on their weights, an important but unanswered question remains when dealing with high-dimensional imbalanced data: *which feature should be included in the final subset when there is more than one feature with the same weight?*.

We believe we have addressed these issues in our proposed solution SYMON, which we shall discuss and evaluate next against similar recent techniques [8, 9, 22] in Sections 3 and 4.

3. SYMON Algorithm

The proposed solution is made up of a number of components so for ease of exposition, we shall discuss the individual components separately before presenting the algorithm that binds these components together. To facilitate better understanding we have also summarised the SYMON notations and parameters in Table 1.

3.1. Harmony Search

Harmony search belongs to the family of meta-heuristic algorithms, and has seen many successful applications over the years [23, 24, 25, 26, 27, 28]. First proposed by Geem *et. al.* [29], it mimics the music improvisation process by modelling it as an optimisation algorithm. Readers interested in the details can refer to [29, 30, 31] but essentially there are five key steps in Harmony Search.

Initialisation The first step is to initialise the dynamic parameters of the algorithm, including harmony memory size (HMS), harmony memory consideration rate (HMCR), number of iterations (NI) and Pitch Adjustment Rate (PAR). Depending on the way Harmony Search is used, the parameter initialisation can be different: randomly, heuristically, or specified by the user.

Improvisation Next, a new harmony vector (NHV) is created. Along with its own characteristics, the NHV will also inherit some of the characteristics of the previously generated vectors.

Evaluation Once a new NHV is created, the goodness of the generated solution will be evaluated. Depending on the application that uses Harmony Search, different evaluation metrics will be used.

Replacement Once the newly created vector is evaluated it is compared to the existing vectors in the harmony memory (HM). This new vector replaces the worst vector in the harmony memory if it has a fitness that is better than the worst vector.

Stopping-criterion check Finally, at the conclusion of a single iteration in harmony search (HS), the algorithm checks the stopping criterion. The criterion is met when the number of iterations is reached or there is no further state of change (i.e. convergence) between the current iteration and the previous iterations.

It should be clarified that there are a number of feature selection algorithms based on Harmony Search [32, 33, 30, 31, 23, 21] but they are "general purpose" in the sense that they are not designed specifically for high-dimensional imbalanced data sets. SYMON is designed for this specific problem and is highlighted by its *feature weighting* component where it weights features according to their dependency against the set of class labels. This dependency is determined using Symmetrical Uncertainty.

3.2. Symmetrical Uncertainty

Symmetrical Uncertainty is shown to be effective in feature selection for large scale data sets [34, 35]. Based on entropy, it works by measuring the uncertainty of a random variable x to another variable y as given by Equations 1 and 2. In these equations, $P(x_i)$ is the prior probability for all values of x and $P(x_i|y_i)$ is the posterior probability of x given y.

$$H(x) = -\sum P(x_i)\log_2(P(x_i)) \tag{1}$$

$$H(x|y) = -\sum_i P(y_i)\sum_j P(x_i|y_i)\log_2(P(x_i|y_j)) \tag{2}$$

and from Equations 1 and 2, we obtain the information gain shown in Equation 3, $\mathcal{G}$ reflects the entropy loss of x once y is considered.

$$\mathcal{G}(x|y) = H(x) - H(x|y) \tag{3}$$

Algorithm 1: SYMON

```
Input  : F, Set of all features
         C, Set of all class labels
         NI, number of iterations
         HMS, harmony memory size
         HMCR, harmony memory consideration rate
         PAR_max, maximum pitch adjustment rate
         PAR_min, minimum pitch adjustment rate

Output: HM, optimised solution vectors in harmony memory

w = CalculateSU(F, C);
Initialise();
for t ← 1..NI do
    for f ∈ 𝓕 do
        R ← random number;
        if R_f > HMCR then
            Randomly select vector v_r from HM;
            NHV[f] ← v_r[f];
            R_p ← random number;
            P_f ← PAR(t) (Equation 6);
            if P_f < R_p then
                NHV[f] ← ¬NHV[f];
        else
            φ ← random number;
            if φ > 0.5 then
                NHV[f] ← 1;
            else
                NHV[f] ← 0;
    NHV = VectorTune(NHV, w, r, d);
    if f(NHV) > f(v) then
        HM = HM − {v} ∪ {NHV};
```

Since information gain $\mathcal{G}$ is a symmetrical measure for every pair of x, y, it is a natural candidate to determine the correlation between the pair. However, to compare each combination of x, y meaningfully, the values in Equation 3 have to be normalised (i.e., $\mathcal{S} \in [0, 1)$) as in Equation 4.

$$\mathcal{S}(x, y) = \frac{2\mathcal{G}(x|y)}{H(x) + H(y)} \tag{4}$$

where $\mathcal{S} = 1$ implies that x and y are fully correlated, while $\mathcal{S} = 0$ means that x and y are independent. In the context of SYMON, x is the current feature under consideration against y, which is in fact the target class label. For readability, we use f in place of x for features, and c in place of y for class labels from this point onwards. Thus, Equation 4 gives the symmetrical uncertainty of a feature f against a single target class label c. That means we can measure the weight of f over all class labels $\{c_1, \ldots\}$ using Equation 5, where the symmetrical uncertainty $\mathcal{M}(f_i, c)$ will indicate the correlation strength of f_i to c over all other class labels. If a feature f is strongly correlated with a class c_i, then the normalised weight of each feature calculated through $\mathcal{M}(f_i, c)$ will have the greatest value for all $\mathcal{S}(f_j|c)\, \forall j, j \neq i$.

$$\mathcal{M}(f_i, c) = \frac{\mathcal{S}(f_i|c)}{\sum_j \mathcal{S}(f_j|c)} \tag{5}$$

Algorithm 2: CalculateSU()

Input : F, Set of all features
C, Set of all class labels

Output: w, Set of feature weights

for $f \in |\mathcal{F}|$ **do**
  for $c \in |\mathcal{C}|$ **do**
    Measure $\mathcal{S}(f|c)$;
  Sum all the dependency values of f to class labels, $\sum_f \mathcal{S}(f|c)$;
  $w(f)$ = calculate the final weight of f using (5);

Algorithm 3: VectorTune()

Input : w, Set of feature weights
NHV, A feature vector
r, ripple factor
d, subset size

Output: Optimal feature subset vectors in harmony memory

$F_s^w \leftarrow \{f_i^w, f_j^w, \ldots\} \subset NHV$;
$F_u^w \leftarrow F^w - F_s^w$;

$F_s \leftarrow \{f_i, f_j, \ldots\} \subset NHV$;
$F_u \leftarrow F - F_s$;

if $|F_s| = d$ **then**
  ripple_add(r, d, F_s^w, F_u^w);
  ripple_remove(r, d, F_s^w, F_u^w);
if $|F_s| > d$ **then**
  while $|F_s| \neq d$ **do**
    ripple_remove(r, d, F_s^w, F_u^w);
if $|F_s| < d$ **then**
  while $|F_s| \neq d$ **do**
    ripple_add(r, d, F_s^w, F_u^w);
output: tuned $\mathcal{NHV}$;

3.3. Vector tuning operations

Vector tuning operations have been used in other works [36, 37] but they are based on a goodness criterion (e.g. classification accuracy). In contrast, the operations in SYMON will only pick features with the same weight and then include/exclude them in/from the final subset with respect to their impact on the performance metric. This is done with the use of Equation 5, which will give the best mix of a set of features to predict a given target class. Then as a refinement to fine tune the feature subset, the vector tuning operations adds or remove features with the same weight until only the most significant features are retained in the subset. To be able to do this, the vector tuning operations must be able *to identify the most and least significant features.*

Definition 1. *A feature is* ***most significant*** *if (i) its inclusion in the selected subset results in the best performance metric when compared to other selected features and (ii) its exclusion from the set significantly reduces the performance of the features when compared to the exclusion of other features from the same subset.*

Definition 2. *A feature is* ***least significant*** *when its exclusion from the final subset does not* *significantly reduce the performance of the selected set (in comparison to the exclusion of other features in the same set).*

The above definitions are incorporated into two operations: `Ripple_Add()` and `Ripple_Rem()`. These operations have two parameters: ripple factor (r) and desired subset size (d).

Ripple factor, (r) Determines the combination of features to consider and therefore has an influence on the results.

Desired subset size, (d) Controls the search space by limiting the features to be considered.

These parameters can be adjusted to seek the best feature subsets for predicting the minority class labels. All features in the data set (i.e. F) are divided into two subsets: $F_s = \{f_1, \ldots, f_n\}$ being the currently selected features and $F_u = F - F_s$ being the features currently not under consideration. `Ripple_Add(r)` will add r most significant features from F_u to F_s while removing $r-1$ least significant features from F_s in each iteration of SYMON. `Ripple_Rem(r)` will remove the r least significant features in F_s while adding $r-1$ most significant features from F_u to F_s. The process of adding/removing features starts from selected/unselected features with the same weight. If adding and removing features with the same weight does not satisfy the subset size criteria then the rest of the features will be considered in order to be included and/or excluded. Depending on the current size of the selected features $|F_s|$ and the required subset size d, there are three possible scenarios:

- The required size is equal to the number of selected features – F_s is changed by applying `Ripple_Rem(r)` and `Ripple_Add(r)`;
- The required subset size is more than the number of selected features – F_s is increased by applying `Ripple_Add(r)`; and lastly
- The required subset size is less than the number of selected features – F_s is decreased by applying `Ripple_Rem(r)`.

As with the above rules, either operation will only add one feature into the subset at any time regardless of the value of r. A large r however will have a greater impact as the mix of features face more adjustments. For example, `Ripple_Rem(3)` will remove the 3 least significant features from F_s and then add the 2 most significant features from F_u to F_s. `Ripple_Rem(2)` on the other hand will only remove the 2 least significant features from F_s and add a single significant feature from F_u back into F_s.

3.4. SYMON

In this section we discuss SYMON in Algorithm 1, with Algorithms 2 and 3 detailing the two components discussed earlier: Symmetrical Uncertainty and Vector tuning operations.

The first step of SYMON (Algorithm 1, line 1) is to rank all the features using Symmetrical Uncertainty (Algorithm 2) before HS is initialised. This initialisation is carried out using random binary values (line 2). Also in this stage, the fitness for each vector is calculated. The fitness assessment is introduced in Section 4. Then, the main steps of HS will be performed (lines 3 – 21). To produce a new harmony vector (NHV) during the improvisation steps, a random number R_c is generated (line 4).

If the random number R_c is higher than HMCR (line 6), then the current component will be selected with respect to the harmony memory, in the sense that a vector v_r will be selected randomly and the value of the corresponding component will be copied (lines 6 – 8). Otherwise, the component c is randomly assigned a binary 0 or 1 as shown (lines 14 – 18). The Pitch Adjustment Rate (PAR), as shown in Equation 6, only affects components that are filled with respect to HM (lines 10 – 12). Here, PAR_{min} and PAR_{max} are the lower and upper bounds of the PAR respectively, and t is the current iteration number.

$$\text{PAR}(t) = \frac{t}{\text{NI}} \times (\text{PAR}_{max} - \text{PAR}_{min}) \tag{6}$$

Once the NHV is created, it will be passed to the vector tuning operations (line 19) to measure the weight of features according to Equation 5 and evaluate the value $\mathcal{M}$ for each selected feature. Then the fitness (denoted $f()$ in Algorithm 1) of this NHV is evaluated (lines 20 – 21), where the vector will replace the worst vector of HM provided that the fitness of the newly generated harmony vector is better than worst fitness of HM. The fitness function can be classification accuracy, kappa statistics, G-Mean or any statistical measure such as Wilcoxon [8, 9, 23, 36, 38].

As discussed, Algorithm 2 measures the dependency between every feature and the class label. Hence, its output is the set of feature weights, w. This set of weights is used in Algorithm 3 to fine tune the set of features. The first step of Algorithm 3 is to create the initial feature subsets (lines 1 – 2). The selected features with the same weight are placed into F_s and the remaining unselected features with the same weight in to F_u. In lines 7 – 15, the algorithm alters the NHV by vector optimisation operations with respect to r and d. The two operators determine the feature(s) to remove or add by evaluating the feature combination in F_s and F_u at each iteration using the weights in w.

A mentioned previously, if adding and removing features with equal weights does not satisfy the required subset size (d) condition then the process continues to add and remove features with varying weights. The output of the vector tuning will be passed to the evaluation metric to assess its goodness. The related experiments are discussed in the next section.

4. Experimental results

The evaluation here will use a number of measures (classifier metrics, statistics and execution time) and different high-dimensional imbalanced datasets (microarray and imagery) for comparison against benchmark algorithms. For meaningful comparison, we draw upon the evaluation methods reported in similar works replicating the experiments using SVM as the underlying classifier and measuring the classifier performance using Area Under Curve (AUC), G-Mean (GM) and the Wilcoxon signed-rank sum. The results are promising and answer the following questions.

- *What are the best empirical settings to ensure SYMON's optimal performance?* SYMON's performance can be affected by the free parameters, so fine-tuning these parameters is crucial to ensuring the optimal performance of SYMON. Thus, we discuss how this near optimality can be achieved in Section 4.2.

- *How comparable is SYMON's performance to existing state-of-the-art feature selection algorithms designed for high dimensional imbalanced class problems?* This is clearly the key question that motivates the evaluation, hence we compared SYMON against similar works [22][8][9] as discussed earlier in Sections 4.3.1 and 4.3.2. The related comparisons are made with SVM-RFE [22], SVM-BFE [9] and Hellinger based feature selection algorithm [8].

- *How effective is the performance of SYMON in comparison to SMOTE as one of the well-established baseline algorithms?* This question investigates the performance of SYMON and compares against SMOTE [10, 11]. To answer this question in Section 4.3.1 we integrate filter-based ranking algorithms of ReliefF (RLF)[4] and Principal Component

[4]we executed RLF with $k = 10$ for kNN

Table 2: Data sets and their characteristics used in this paper for evaluation.

Datasets	Number of Samples	Number of features	Class (minority/majority)
Colon (COL)	62	2000	28/72
SRBCT	83	2308	13/87
Olivetti Faces (FACE)	400	4096	31/69
Central Nervous (CNS)	50	7129	25/75
DLBCL	59	7129	40/60
LEUKMIA (LEU)	38	7129	29/71
Cardio (CAR)	174	9182	15/85
Lung (LUG)	181	12534	17/83
Breast Cancer (BC)	78	24481	43/57

Analysis (PCA) with SMOTE. The variations are called SMOTE-RLF and SMOTE-PCA.

- *How robust is SYMON when presented with data sets possessing different levels of imbalance?* Flowing from the first key question, SYMON's performance should be stable across a variety of data sets. We evaluate SYMON with different levels of class imbalance for different data sets and then compare the results against other works. The results are reported in Section 4.3.4.

4.1. Data sets

We selected eight large-scale DNA microarray data sets and one imagery dataset called Olivetti Faces. The various data characteristics are shown in Table 2. These data sets are either multi-class or binary class. In the multi-class data sets, our set up involves selecting the least frequent class label as the minority and the rest as the majority. The ratio of this minority to majority class labels is shown in Table 2 as Class (minority/majority).

To meaningfully test for true performance, we want to use samples that were not used in model creation, so instead of a leave-one out cross-validation procedure we will use the hold-out strategy instead. In this case, we divide each data set into three disjoint sets: training, testing and validation in proportion of 50%, 30% and 20%, respectively. This ratio is the hold-out condition and in the case of multi-class data sets this means that when selecting a minority class we also have to ensure that the hold-out condition is met.

4.2. Parameter tuning

SYMON belongs to the family of meta-heuristic algorithms because of its underlying use of harmony search, thus the calibration of its parameters is important in order to achieve optimal results. To determine the optimal values for the free parameters, we conducted the following empirical studies to acquire the settings for PAR_{min}, PAR_{max}, HMS and HMCR. We have done this for three scenarios as shown in Table 3 using the LEU dataset. This dataset (LEU) showed more sensitivity in comparison to other datasets, i.e. significant changes in performance are seen for slight changes in the parameter settings. The reported results are averaged over 10 iterations.

Experiment I was carried out to determine the optimal values for PAR_{min} and PAR_{max}. The interval $[0; 1]$ is divided into two halves with identical lengths in which PAR_{min} values can select values from the lower half (i.e., $[0.05; 0.45]$), while PAR_{max} values are selected from the upper half of the interval (i.e., $[0.55; 0.95]$). As the results in Figure 1 show, it is better to set PAR values to higher boundaries of their intervals to achieve better results. Accordingly, in our experiments we set PAR_{min} and PAR_{max} values to 0.45 and 0.9, respectively.

Table 3: Scenarios for various parameter settings.

Experiments	Test variable	Fixed variable
Experiment I	PARs: (PARmin, PARmax)	HMS = 5, HMCR = 0.5
Experiment II	HMCR	HMS = 5, PARs values from Experiment I
Experiment III	HMS	PARs and HMCR values from Experiments I and II

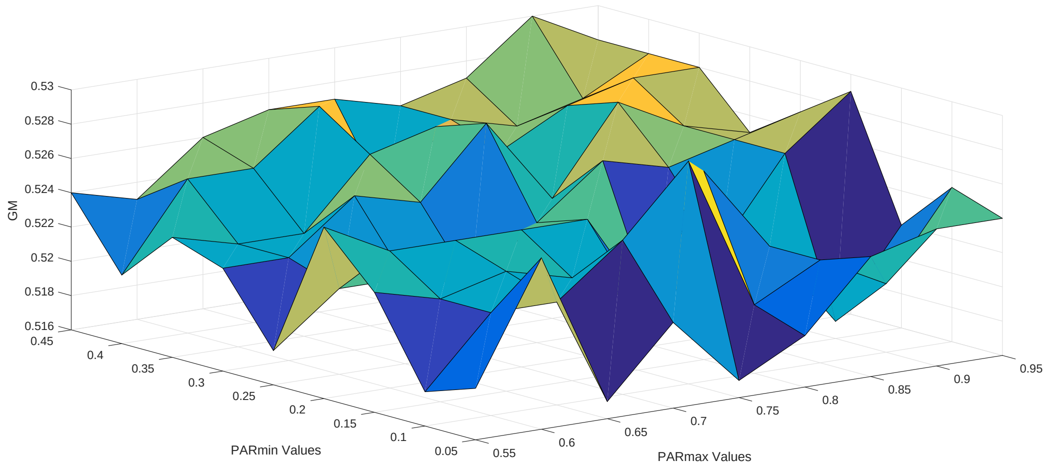

Figure 1: Finding proper values for PAR_{min} and PAR_{max}

In Experiment II, we wanted to find the most suitable values of HMCR $\in \{0.05, 0.1, \ldots, 0.95\}$. HMCR determines the consideration rate of the algorithm on the harmony memory in future improvisation steps. As shown in Figure 4.2, setting HMCR to 0.75, ensures the algorithm performs at its best.

Experiment III empirically determines the optimal harmony memory size. Since the performance of random harmony search is directly correlated with the harmony memory size, it is important to get this setting right. A harmony memory size that is too small prevents the algorithm from reaching optimal parts of the solution space. On the other hand, a large harmony memory size will lead to an unnecessarily long execution time. As shown in Figure 3, setting the harmony memory size (HMS) to 35 yields the best performance.

Finally, the parameter settings of SYMON is as follows: HMS = 35, HMCR = 0.75, PAR_{min} = 0.45, PAR_{max} = 0.9, d $\in \{F/5, 2F/5, 3F/5, 4F/5\}$, r $\in \{1, 2, 3, 4\}$ and NI = 200. Also parameter setting of competitor algorithms are as follows: in the Hellinger based feature selection algorithm (in this paper called, D-HELL) [8] $k = 5$, NI = 200, in SVM-RFE [22], SVM-BFE [9] the only required parameter for setting is the number of top features to select, which is equal to value of d in SYMON.

As was shown in the latter paragraphs of Section 2, the recent state-of-the-art algorithms consider SVM as their underlying classifier. An investigation was undertaken to determine whether this is indeed a good choice, but the results are omitted here for brevity. We compared SVM with different classifiers: rule induction, k-nearest-neighbor (kNN), Bayesian, decision trees and artificial neural networks (ANN). It was discovered that ANN and SVM have the best results, probably due to their better handling of noisy data. However, SVM exhibits better runtime performance (i.e. it is faster). Therefore it was determined that SVM is the most suitable.

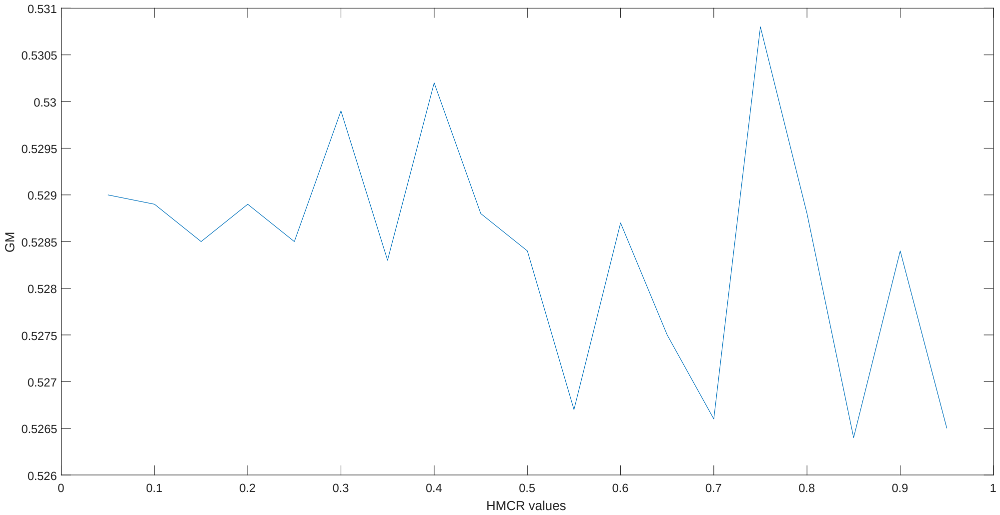

Figure 2: HMCR value fine tuning with respect to Experiment II introduced in Table 3.

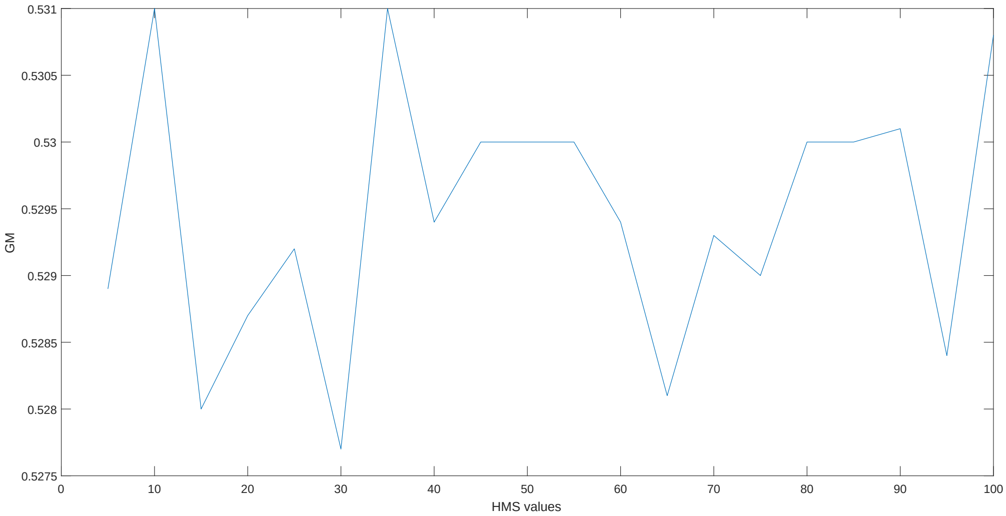

Figure 3: HMS value fine tuning with respect to Experiment III introduced in Table 3.

4.3. Discussion of results

We report three evaluations to determine if SYMON's performance is comparable to existing approaches. As with the related works, we have done so using classifier metrics (Section 4.3.1). We then compare how robust SYMON and its counterparts are in the presence of unseen data using the Wilcoxon signed-rank sum (Section 4.3.2). One of the characteristics of a feature selection algorithm that distinguishes it from other similar works is its power in dealing with different levels of imbalancement; this is investigated in Section 4.3.4. In these evaluations, SYMON proved to perform on-par or better than current solutions. Given these promising results, in Section 4.3.3 we further confirm that SYMON is feasible in practice by evaluating its runtime performance.

4.3.1. Classifier metrics

We start by considering how much improvement in classifier performance SYMON delivers. Our evaluation uses the same measures: AUC and G-Mean, as reported in [22, 9, 8]. We also use the same experimental setup, selecting SVM as the classifier, specifically SVM-RFE [22], SVM-BFE [9], D-HELL [8], and SMOTE-integrated algorithms of SMOTE-RLF and SMOTE-PCA. We implemented these algorithms with SYMON using Matlab and tested them using similar reported experimental conditions.

The results are summarised in Table 4. While classification accuracy has been a popular and traditional way to measure the performance of feature selection algorithms [1], He and Garcia [12] noted that it is not a suitable reflection of classification performance for imbalanced data sets. For example, if a given data set includes 5% of the target class instances and 95% of majority examples, a naive approach of classifying every instance to the majority class would provide an accuracy of 95%; a strong performance by the traditional classification accuracy measure. If the misclassification cost [9, 12] for the 5% target class is significant (and usually this is the case with imbalanced data sets), then such a performance (despite a 95% classification accuracy) is not acceptable. In light of this, the G-Mean is used instead to better reflect the misclassification costs (as seen in [8, 9]) and is given by Equation 7:

$$\text{G-Mean} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{7}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives. The higher value of GM indicates better performance.

We see SYMON performing very well across the four data sets: COL, FACE, DLBCL and LEU in the sense that its G-Mean values are the best in all settings compared to the rest. In the other data sets: CAR, LUG and BC, SYMON is either on-par or better across the various test settings. In the case of BC ($d = F/5$), while the G-Mean score is the same, SYMON uses smaller feature subsets than D-HELL to achieve this same score. This lower number of features has practical implications in terms of model interpretation. Finally, in the SRBCT data set, the performance of SYMON is similar to the other three under evaluation.

The other measure used is the Area Under Curve (AUC), which considers the area under the Receiver Operating Characteristic (ROC) curve [7, 20, 17, 8]. The ROC illustrates the trade-off between positive detection rates and false alarm rates. Consequently, the AUC is a measure of a classifier's discriminative strength between these two rates without considering misclassification costs or class prior probabilities. Using the AUC we can compare how much improvement SYMON and its counterpart deliver to the classifiers across different subset sizes.

As noted in Table 4, all algorithms were fine-tuned to the different values of desired subset size (d). Consequently, the same was done for SYMON and across its four ripple factor values (r). These results are shown in Table 5 in the form of $x(y)$, where x and y are the AUC value and the proportion of features that the AUC was gained, respectively. SYMON displays similar performance across three data sets: LEU, SRBCT and LUG, with the values of 0.875, 1 and

Table 4: Comparisons of Filter-based ranking, state-of-the-arts and SYMON using **test** data in terms of GM. Higher GM means better performance.

Datasets	d	SYMON				Filter based ranking		State-of-the-arts		
		r = 1	r = 2	r = 3	r = 4	SMOTE-RLF	SMOTE-PCA	SVM-RFE	SVM-BFE	D-HELL
COL (F = 2000)	F/5	0.674	0.674	0.674	0.674	0.603	0.585	0.6	0.56	0.67
	2F/5	0.715	0.715	0.715	0.715	0.603	0.63	0.6	0.6	0.6
	3F/5	0.674	0.674	0.715	0.715	0.603	0.63	0.56	0.56	0.64
	4F/5	0.715	0.715	0.715	0.715	0.674	0.684	0.56	0.6	0.6
SRBCT (F = 2308)	F/5	1	1	1	1	1	1	1	1	1
	2F/5	1	1	1	1	1	1	1	1	1
	3F/5	1	1	1	1	1	1	1	1	1
	4F/5	1	1	1	1	1	1	1	1	1
FACE (F = 4096)	F/5	0.982	0.982	0.982	0.982	0.962	0.91	0.982	0.887	0.975
	2F/5	0.991	0.991	0.991	0.991	0.988	0.963	0.982	0.935	0.988
	3F/5	0.982	0.982	0.982	0.982	0.982	0.988	0.988	0.963	0.982
	4F/5	0.982	0.982	0.982	0.982	0.982	0.982	0.988	0.982	0.982
CNS (F = 7129)	F/5	0.79	0.79	0.79	0.79	0.577	0.624	0.745	0.745	0.707
	2F/5	0.79	0.79	0.79	0.79	0.667	0.745	0.745	0.697	0.745
	3F/5	0.79	0.79	0.79	0.79	0.745	0.745	0.745	0.745	0.745
	4F/5	0.79	0.79	0.79	0.79	0.745	0.745	0.745	0.745	0.745
DLBCL (F = 7129)	F/5	0.296	0.296	0.296	0.316	0.274	0.387	0.25	0.25	0.223
	2F/5	0.296	0.296	0.296	0.316	0.274	0.547	0.273	0.25	0.25
	3F/5	0.316	0.316	0.418	0.418	0.274	0.296	0.273	0.25	0.25
	4F/5	0.296	0.296	0.296	0.316	0.274	0.274	0.273	0.273	0.274
LEU (F = 7129)	F/5	1	1	1	1	0.7	0.7	0.316	0.316	0.5
	2F/5	1	1	1	1	0.7	0.7	0.316	0.316	0.836
	3F/5	1	1	1	1	0.7	0.7	0.316	0.316	0.447
	4F/5	1	1	1	1	0.86	0.86	0.316	0.316	0.316
CAR (F = 9182)	F/5	0.935	0.935	1	1	0.935	0.935	0.935	1	1
	2F/5	0.935	0.935	1	1	0.935	0.935	0.935	1	0.935
	3F/5	0.935	0.935	0.935	0.935	0.935	0.935	0.935	1	0.935
	4F/5	0.935	0.935	0.935	0.935	0.935	0.935	0.935	1	0.935
LUG (F = 12533)	F/5	1	1	1	1	0.968	0.968	0.973	1	1
	2F/5	1	1	1	1	0.968	0.968	0.973	1	1
	3F/5	1	1	1	1	0.968	0.968	0.973	1	1
	4F/5	1	1	1	1	0.968	0.968	0.973	1	1
BC (F = 24481)	F/5	0.626	0.626	0.664	0.664	0.524	0.524	0.56	0.664	0.626
	2F/5	0.626	0.626	0.664	0.664	0.585	0.524	0.626	0.58	0.626
	3F/5	0.626	0.626	0.664	0.664	0.524	0.524	0.626	0.52	0.664
	4F/5	0.626	0.626	0.664	0.664	0.626	0.524	0.626	0.52	0.626

Table 5: AUC evaluation using various data sets and different feature subset sizes for the four different algorithms including SYMON. $F/5$ means 20% of the total features in F and $2F/5$ means 40% of the totals features in F, and so on.

Datasets	SYMON	Filter based ranking		State-of-the-arts		
		SMOTE-RLF	SMOTE-PCA	SVM-RFE	SVM-BFE	D-HELL
COL (F = 2000)	0.72 (0.8)	0.616 (0.2)	0.676 (0.8)	0.613 (0.2)	0.67 (0.4)	0.738 (0.4)
SRBCT (F = 2308)	1 (0.2)	1(0.2)	1(0.2)	1(0.2)	1(0.2)	1(0.2)
FACE (F = 4096)	0.988 (0.2)	0.988 (0.4)	0.988 (0.6)	0.988 (0.6)	0.988 (0.8)	0.988 (0.4)
CNS (F = 7129)	0.652 (0.2)	0.75 (0.6)	0.75 (0.4)	0.46 (0.6)	0.652 (0.4)	0.652 (0.6)
DLBCL (F = 7129)	0.787(0.2)	0.637 (0.2)	0.637 (0.8)	0.687 (0.2)	0.687 (0.2)	0.687 (0.6)
LEU (F = 7129)	0.875(0.2)	0.875(0.2)	0.875(0.2)	0.875(0.2)	0.875(0.2)	0.875(0.2)
CAR (F = 9182)	1(0.2)	0.937 (0.2)	0.937 (0.2)	0.937(0.2)	1(0.2)	1(0.2)
LUG (F = 12533)	0.968 (0.2)	0.968 (0.2)	0.968 (0.2)	0.968 (0.2)	0.968 (0.2)	0.968 (0.2)
BC (F = 24481)	0.792(0.2)	0.652 (0.8)	0.593 (0.2)	0.75(0.2)	0.291(0.4)	0.75(0.4)

0.968, respectively. With the other data sets, SYMON is superior compared to the related works. This superiority is seen in both the subset size and AUC in the DLBCL and BC data sets, and in the subset size in the FACE, CNS and CAR data sets. The reasons for SYMON's performance can be attributed to the way it selects and weights features.

Underpinning SYMON's feature selection strategy is a meta-heuristic optimisation algorithm capable of finding the near-optimal part of the solution space. We opted for near-optimality as it is often hard and at times impossible to locate the optimal solution [38, 39]. Nevertheless, this is sufficient to deliver SYMON a subset of features that is better than the compared works. In their case, features are weighed and ranked with the top-k features picked as the final subset. This one-off "weigh and rank" strategy fails to consider the relationship between features that only SMYON's meta-heuristic optimisation algorithm will uncover.

Furthermore, the ranking of features often does not discriminate between two or more features having the same weight. Yet this is important as features with the same weight can have different levels of dependency to a target class label. In the case of the minority class, having the most discriminative feature subset will deliver the strongest predictive outcome. While SYMON's meta-heuristic search inherently takes the feature-class correlation into account, comparable algorithms fall short in the follow ways:

- SVM-RFE [22] uses a recursive feature elimination technique, where the SVM classifier performance is used to determine the weight of a feature. Once all features are evaluated, the top-k features are selected as the subset, i.e. ignoring the case when multiple features in the top-k set have the same weight.

- SVM-BFE [9] was introduced by Maldonado *et.al.* [9] with two different loss functions: (i) 0-1 loss and (ii) balanced-loss. The latter is preferred as the former assumes equal cost in the error between the binary classes. Like SVM-RFE, SVM-BFE weights each feature based on the contribution towards improving the SVM classifier performance. Similarly it ignores the case when multiple features in the top-k set have the same weight.

Table 6: Wilcoxon comparisons using **test** data.

	2nd Algorithm					
		SVM-RFE	SVM-BFE	D-HELL	SMOTE-RLF	SMOTE-PCA
1st Algorithm	SYMON(r=1)	4.8828e-04 (1)	4.8828e-04 (1)	9.7656e-04 (1)	7.56E-06 (1)	0.00030732 (1)
	SYMON(r=2)	4.8828e-04 (1)	4.8828e-04 (1)	9.7656e-04 (1)	7.56E-06 (1)	0.00030732 (1)
	SYMON(r=3)	4.8828e-04 (1)	4.8828e-04 (1)	9.7656e-04 (1)	2.31612E-06 (1)	1.41532E-04 (1)
	SYMON(r=4)	4.8828e-04 (1)	4.8828e-04 (1)	9.7656e-04 (1)	2.31612E-06 (1)	1.02778E-04 (1)

Table 7: Wilcoxon comparisons using **train** data.

	2nd Algorithm					
		SVM-RFE	SVM-BFE	D-HELL	SMOTE-RLF	SMOTE-PCA
1st Algorithm	SYMON(r=1)	2.5893e-05 (1)	0.0011 (1)	0.0013 (1)	0.1222 (0)	0.3032 (0)
	SYMON(r=2)	2.5893e-05 (1)	0.0011 (1)	0.0013 (1)	0.1222 (0)	0.3032 (0)
	SYMON(r=3)	3.4283e-06 (1)	1.7946e-04 (1)	5.8884e-05 (1)	0.1222 (0)	0.3032 (0)
	SYMON(r=4)	1(0)	0.9722 (0)	0.0012 (1)	0.1222 (0)	0.3032 (0)

- D-HELL [8] weights features with respect to class labels in the same spirit as SYMON. However, D-HELL changes the underlying data characteristics while SYMON avoids changing the underlying structure. With D-HELL, the frequent class labels are further decomposed into sub-class labels resulting in a new data set. This decomposition is based on a clustering algorithm. Therefore, the accuracy of the decomposed class labels and the new dataset depends on how accurately the clustering algorithm can group similar data together.

- SMOTE [10, 11] generates artificial samples for minority class labels to (roughly) balance all the class labels. Then it is integrated with RLF and PCA to weight the features and select the top-k features. This algorithm first assigns weights to features without considering their correlation to class label(s). Also, equally-weighted features is a problem for this algorithm.

4.3.2. Statistical analysis

Clearly, the practical utility of SYMON lies in its performance for unseen data. To evaluate this, we conducted the Wilcoxon signed-rank test which, according to [40], is a more sensible measure than a t-test as it assumes commensurability of differences, but only qualitatively. In other words, it is desirable to note the differences but the absolute magnitude quantifying these differences is not considered. The test is also considered to be 'safer' as it does not assume normal distributions and outliers have less impact on the final result.

The purpose of the Wilcoxon signed-rank test is to show if the results from the two algorithms are independent (i.e. rejecting the null hypothesis). The test results are shown in Tables 6 and 7 in the form of $p(h)$, where p refers to the test value and h indicates if the null hypothesis should be rejected (i.e., $h = 1$). As seen in Table 6 (which is the evaluation against unseen data), the results produced by SYMON versus the other comparable algorithms have been confirmed to be significant.

4.3.3. Execution time

To evaluate SYMON's execution time compared to recent works, we measure the runtime of the two key stages of each algorithm: feature weighting and feature selection. The total execution time, feature weighing time and feature selection time for SYMON, D-HELL, SVM-RFE, and SVM-BFE are shown in Table 8. We can see that the feature weighting performance of SYMON is extremely competitive but where it fails is in the feature selection stage. The extended runtime comes as no surprise because SYMON's feature selection step uses Harmony Search, which treats feature selection as an optimisation problem. Additionally, SYMON considers all available features thus, the execution time grows exponentially as the number of features grow.

Although SYMON produces a better subset of features for predicting minority classes, the high runtime would impede its uptake. This is because SYMON considers the different combination of features available (especially among features with similar weights) while other algorithms simply pick the top-k features as the final solution. So on the one hand, we have SYMON that searches for the best combination of features to give the best contrast to a minority class; and on the other, we have a straightforward ordering of all features by their weight and then, picking the top-k features. A balance between the two is desirable.

An observation about the weights is that our Symmetrical Uncertainty measure ($\mathcal{M}$) already encodes the correlation strength of the feature to a class label. Therefore, we do not have to consider all features available. At the same time, we know that the top-k features will not yield the best results as confirmed in our experiments. Therefore, the required and ideal number of features to be considered is somewhere between $k+1$ and $|F|$. This led us to investigate a range of subset sizes that SYMON should consider at the selection stage.

In other words, given d the required subset size, we constrain SYMON to operate on $d + d_z$ features (instead of exploring all $|F|$ features), where $d_z << |F| - d$ is an addition number of attributes to consider at the feature selection stage. As $d + d_z << |F|$ the search space is substantially smaller. Since we are using the top-$(d + d_z)$ features, we are working with features that have a high correlation level to the minority class label. Taking this approach, we re-conducted our experiments with different numbers of features (Figures 4 and 5). Taking a subset size of $d = 0.2F$, we evaluated this variation of SYMON by considering d_z values of 1%, 5% and 10% of F. The runtime is significantly reduced as shown in Table 8.

While the runtime remains higher than the benchmark D-HELL, we believe it is now within a range (minutes rather than many hours) that is acceptable. This is especially the case if the most accurate predictive outcome on the minority class is sought, e.g., in critical applications like medical diagnosis or fraud detection. We refer to this variation as SYMON$_k$, $k = d + d_z$. With this reduced execution time, SYMON$_k$'s AUC and GM performance remain comparable to the original SYMON and more importantly, it is also comparable or outperforms the benchmark D-HELL algorithm. The results on the AUC and GM measures are shown in Figures 4 and 5 for SYMON$_k$ over a range of values for d_z and compared to SYMON and D-HELL.

4.3.4. Robustness to class imbalance

Lastly, we are interested in evaluating the imbalance rate, r_{imb} in SYMON and the other algorithms. The imbalance rate, given by Equation 8, is a ratio of the number of samples in the majority class to the number of samples in the minority class. It gives an indication of how robust an algorithm is to the imbalancement of a data set.

SMOTE - regardless of how the dataset is imbalanced - generates some artificial samples to make the dataset (roughly) balanced. The other compared algorithms, and also SYMON, do not make changes to the original dataset. Hence, the imbalance rate of the experimental dataset has an effect on the performance of the algorithms. Therefore we exclude SMOTE variations from imbalance experiments.

Over a range of imbalance rates, we compare the GM and AUC scores of SYMON and the other algorithms. The results are given in Figures 6 and 7. As SYMON relies on meta-heuristics to find the most optimal solution possible, we see that the results are strongly in favour of SYMON especially when its fine-tuning operations subsequently consider combinations of highly correlated features to the class labels to ensure that the best results are achieved in each case.

Table 8: Execution time of SYMON and similar algorithms across data sets with different numbers of dimensions: 2K means 2,000 features and all algorithms are to identify a 20% feature subset that gives the best classifier performance on the minority classes. Total execution time is in seconds.

Execution time	Algorithms	Datasets					
		2K(COL)	4K(FACE)	7K(DLBCL)	9K(CAR)	12K(LUG)	24K(BC)
Feature weighting execution time	SVM-RFE	2	6	7	10	16	28
	SVM-BFE	80	1112	358	1668	25066	48569
	D-HELL	2	4	7	10	12	21
	SYMON	2	4	7	10	14	22
	SMOTE-RLF	3.95	134.9	11.613	103.02	193.85	89.860
	SMOTE-PCA	1.512	81.89	11.033	25.3	56.736	1099.4
Feature selection execution time	SVM-RFE	0.6	0.61	0.58	0.6	0.67	0.68
	SVM-BFE	0.64	0.64	0.52	0.6	0.7	0.73
	D-HELL	0.51	0.6	0.57	0.55	0.57	0.58
	SYMON	287	1008	2615	4127	2615	31783
	SYMONd$_{d+1\%}$	77	271	264	342	683	1536
	SYMONd$_{d+5\%}$	80	276	285	415	707	2096
	SYMON$_{d+10\%}$	81	440	347	420	732	3635
	SMOTE-RLF	1.095	2.037	0.165	2.28	2.21	2.46
	SMOTE-PCA	1.243	2.376	0.272	2.435	2.73	34.75
Total execution time	SVM-RFE	2.6	6.61	7.58	10.6	16.67	28.68
	SVM-BFE	80.64	1112.64	358.52	1668.6	25066.7	48569.73
	D-HELL	2.51	4.6	7.57	10.55	12.57	21.58
	SYMON	289	1012	2622	4134	17023	31805
	SYMONd$_{d+1\%}$	79	275	271	352	697	1558
	SYMON$_{d+5\%}$	82	280	292	425	721	2118
	SYMON$_{d+10\%}$	83	444	354	430	746	3657
	SMOTE-RLF	5.045	136.937	11.778	105.3	196.05	92.32
	SMOTE-PCA	2.755	84.268	11.305	27.735	59.466	1134.15

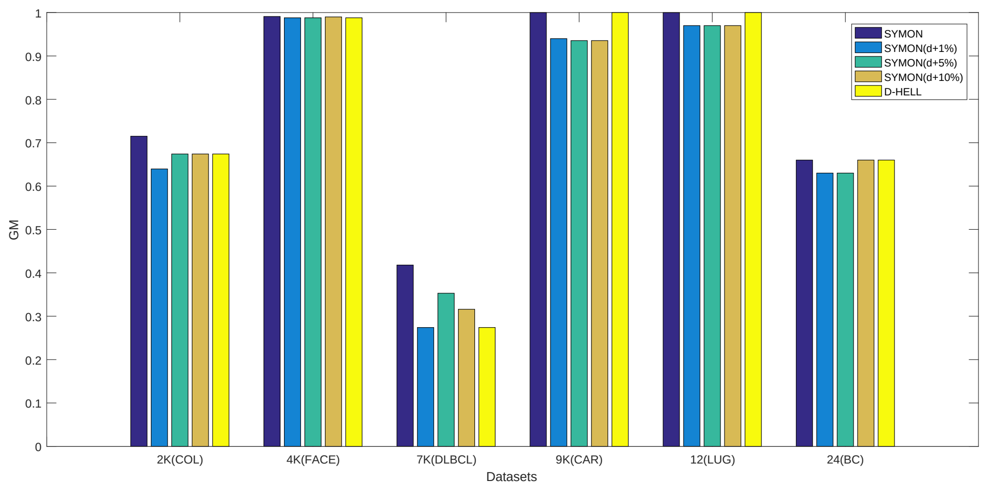

Figure 4: Evaluation of SYMON$_{d+d_z}$ on two GM classifier metric across various data sets. Note that SYMON and SYMON$_k$ performances shown here are indicative as a different NI (iterations) and HMS (Harmony memory size) setting will produce a different feature subset, thus affecting GM scores but a higher NI and HMS for the same d_z setting will always give a higher GM score.

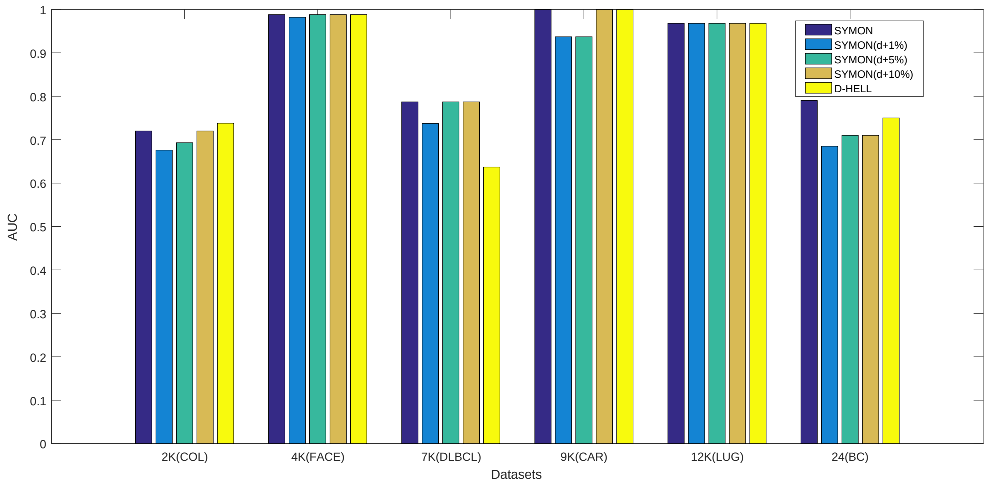

Figure 5: Evaluation of SYMON_{d+d_z} on the AUC classifier metric across various data sets. Note that SYMON and SYMON_k performances shown here are indicative as a different NI (iterations) and HMS (Harmony memory size) setting will produce a different feature subset.

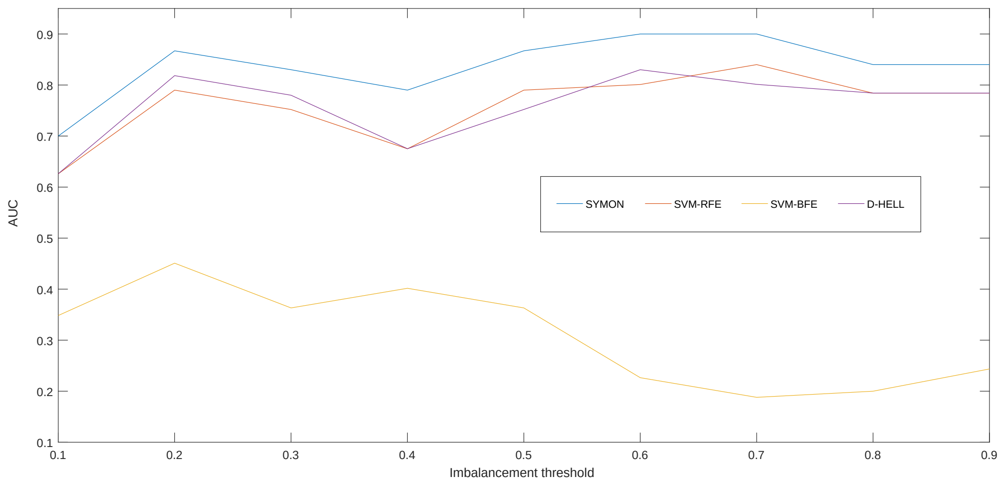

Figure 6: Robustness of various algorithms to different rates of imbalancement based on AUC.

$$(r_{imb}) = \frac{\#of Samples_{minority}}{\#of Samples_{majority}} \tag{8}$$

5. Conclusion and future works

In this paper we introduced SYMON as a new feature selection algorithm for high dimensional imbalanced class datasets. Similar to other related works, SYMON is a two stage algorithm, The first stage, feature weighting, measures the features' weights (or importance). In the second stage, known as feature selection, the top-k features are selected based on their weights. What distinguishes SYMON from similar works are (i) its capability in measuring the feature weight with respect to the dependency to class label(s) and (ii) dealing with the situation where different features have the same weight (or importance). SYMON was empirically compared against the state-of-the-art and baseline algorithms and the results showed comparable or better performance (in terms of GM and AUC) over different high dimensional datasets.

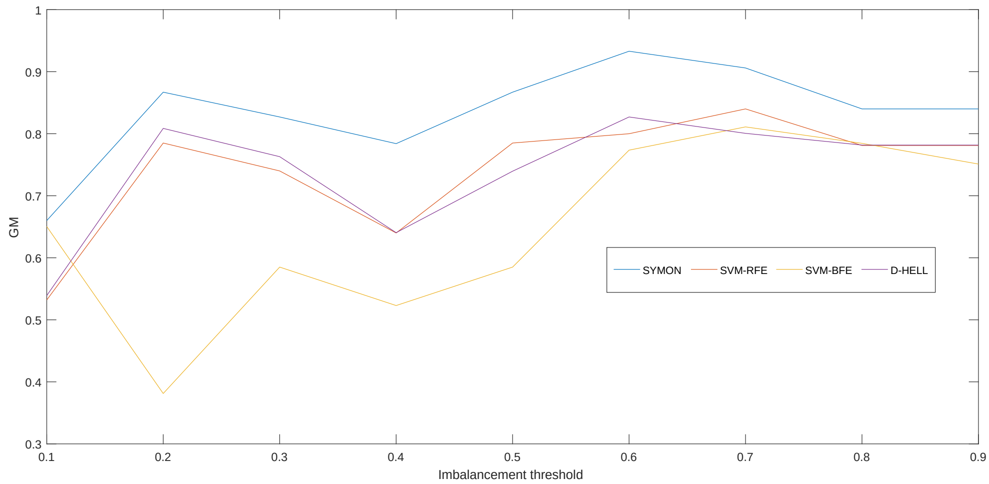

Figure 7: Robustness of various algorithms to different rates of imbalancement based on G-Mean (GM).

This performance can be attributed to its use of symmetrical uncertainty to weight features and the vector tuning operations embedded in the feature selection stage.

On the limitations, SYMON has two that we will address for the future work. The first limitation is its high computational time. Even though we experimentally showed that SYMON can be improved in terms of execution time, by focusing on a proportion of the most significant features, a better solution is to explore a faster harmony search core to improve its runtime. The other limitation is to confine feature selection to a desired subset size (d). At the moment, the vector tuning operations are highly dependent on (d) and the ripple factor (r). Instead, a more flexible d will allow more optimal parts of the solution space to be discovered. This could be another avenue to improve SYMON's runtime performance.

References

[1] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[2] Loïc Cerf, Dominique Gay, Nazha Selmaoui-Folcher, Bruno Crémilleux, and Jean-François Boulicaut. Parameter-free classification in multi-class imbalanced data sets. *Data & Knowledge Engineering*, 87:109–129, 2013.

[3] Jason Van Hulse and Taghi Khoshgoftaar. Knowledge discovery from imbalanced and noisy data. *Data & Knowledge Engineering*, 68(12):1513–1542, 2009.

[4] K Topouzelis, V Karathanassi, P Pavlakis, and D Rokos. Detection and discrimination between oil spills and look-alike phenomena through neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(4):264–270, 2007.

[5] Basant Subba, Santosh Biswas, and Sushanta Karmakar. Intrusion detection in mobile ad-hoc networks: Bayesian game formulation. *Engineering Science and Technology, an International Journal*, 2015.

[6] Duílio ANS Silva, Leandro C Souza, and Gustavo HMB Motta. An instance selection method for large datasets based on markov geometric diffusion. *Data & Knowledge Engineering*, 101:24–41, 2016.

[7] Jason Van Hulse, Taghi M Khoshgoftaar, Amri Napolitano, and Randall Wald. Feature selection with high-dimensional imbalanced data. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pages 507–514. IEEE, 2009.

[8] Liuzhi Yin, Yong Ge, Keli Xiao, Xuehua Wang, and Xiaojun Quan. Feature selection for high-dimensional imbalanced data. *Neurocomputing*, 105:3–11, 2013.

[9] Sebastián Maldonado, Richard Weber, and Fazel Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information Sciences*, 286:228–246, 2014.

[10] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[11] T Deepa and M Punithavalli. An e-smote technique for feature selection in high-dimensional imbalanced dataset. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 2, pages 322–324. IEEE, 2011.

[12] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[13] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[14] Mina Alibeigi, Sattar Hashemi, and Ali Hamzeh. Dbfs: An effective density based feature selection scheme for small sample size and high dimensional imbalanced data sets. *Data & Knowledge Engineering*, 81:67–103, 2012.

[15] Dudyala Anil Kumar and V Ravi. Predicting credit card customer churn in banks using data mining. *International Journal of Data Analysis Techniques and Strategies*, 1(1):4–28, 2008.

[16] Nguyen Tho Thong et al. Intuitionistic fuzzy recommender systems: An effective tool for medical diagnosis. *Knowledge-Based Systems*, 74:133–150, 2015.

[17] Xue-wen Chen and Michael Wasikowski. Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 124–132. ACM, 2008.

[18] Youwei Wang, Yuanning Liu, Lizhou Feng, and Xiaodong Zhu. Novel feature selection method based on harmony search for email classification. *Knowledge-Based Systems*, 73:311–323, 2015.

[19] Irena Koprinska, Mashud Rana, and Vassilios G Agelidis. Correlation and instance based feature selection for electricity load forecasting. *Knowledge-Based Systems*, 82:29–40, 2015.

[20] Pengyi Yang, Wei Liu, Bing B Zhou, Sanjay Chawla, and Albert Y Zomaya. Ensemble-based wrapper methods for feature selection and class imbalance learning. In *Advances in Knowledge Discovery and Data Mining*, pages 544–555. Springer, 2013.

[21] Ren Diao and Qiang Shen. Feature selection with harmony search. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(6):1509–1523, 2012.

[22] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[23] Rana Forsati, Alireza Moayedikia, and Bahareh Safarkhani. Heuristic approach to solve feature selection problem. In *Digital Information and Communication Technology and Its Applications*, pages 707–717. Springer, 2011.

[24] Rana Forsati, MohammadReza Meybodi, Mehrdad Mahdavi, and Azadeh Ghari Neiat. Hybridization of k-means and harmony search methods for web page clustering. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 329–335. IEEE Computer Society, 2008.

[25] Rana Forsati, Mehrdad Mahdavi, Mehrnoush Shamsfard, and Mohammad Reza Meybodi. Efficient stochastic algorithms for document clustering. *Information Sciences*, 220:269–291, 2013.

[26] Mohsen Mirkhani, Rana Forsati, Alireza Mohammad Shahri, and Alireza Moayedikia. A novel efficient algorithm for mobile robot localization. *Robotics and Autonomous Systems*, 61(9):920–931, 2013.

[27] Rana Forsati, Alireza Moayedikia, and Mehrnoush Shamsfard. An effective web page recommender using binary data clustering. *Information Retrieval Journal*, 18:167–214, 2015.

[28] Yin-Fu Huang, Sheng-Min Lin, Huan-Yu Wu, and Yu-Siou Li. Music genre classification based on local feature selection using a self-adaptive harmony search algorithm. *Data & Knowledge Engineering*, 92:60–76, 2014.

[29] Zong Woo Geem. Novel derivative of harmony search algorithm for discrete design variables. *Applied mathematics and computation*, 199(1):223–230, 2008.

[30] M Mahdavi, Mohammad Fesanghary, and E Damangir. An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, 188(2):1567–1579, 2007.

[31] Mahamed GH Omran and Mehrdad Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[32] Messaouda Nekkaa and Dalila Boughaci. Hybrid harmony search combined with stochastic local search for feature selection. *Neural Processing Letters*, pages 1–22, 2015.

[33] Ling Zheng, Ren Diao, and Qiang Shen. Self-adjusting harmony search-based feature selection. *Soft Computing*, 19(6):1567–1579, 2014.

[34] S Senthamarai Kannan and N Ramaraj. A novel hybrid feature selection via symmetrical uncertainty ranking based local memetic search algorithm. *Knowledge-Based Systems*, 23(6):580–585, 2010.

[35] Roberto Ruiz, José C Riquelme, Jesús S Aguilar-Ruiz, and Miguel García-Torres. Fast feature selection aimed at high-dimensional data via hybrid-sequential-ranked searches. *Expert Systems with Applications*, 39(12):11094–11102, 2012.

[36] Rana Forsati, Alireza Moayedikia, Richard Jensen, Mehrnoush Shamsfard, and Mohammad Reza Meybodi. Enriched ant colony optimization and its application in feature selection. *Neurocomputing*, 142:354–371, 2014.

[37] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1424–1437, 2004.

[38] Alireza Moayedikia, Richard Jensen, Uffe Kock Wiil, and Rana Forsati. Weighted bee colony algorithm for discrete optimization problems with application to feature selection. *Engineering Applications of Artificial Intelligence*, 44:153–167, 2015.

[39] Mina Husseinzadeh Kashan, Nasim Nahavandi, and Ali Husseinzadeh Kashan. Disabc: A new artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, 12(1):342–352, 2012.

[40] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.