Hindawi
Security and Communication Networks
Volume 2017, Article ID 6790975, 11 pages
https://doi.org/10.1155/2017/6790975

Research Article

Improved Instance Selection Methods for Support Vector Machine Speed Optimization

Andronicus A. Akinyelu and Aderemi O. Adewumi

School of Mathematics, Statistics & Computer Science, University of KwaZulu-Natal, Private Bag X54001, Durban 4000, South Africa

Correspondence should be addressed to Aderemi O. Adewumi; laremtj@gmail.com

Received 1 July 2016; Accepted 22 August 2016; Published 9 January 2017

Academic Editor: Pascal Lorenz

Support vector machine (SVM) is one of the top picks in pattern recognition and classification related tasks. It has been used successfully to classify linearly separable and nonlinearly separable data with high accuracy. However, in terms of classification speed, SVMs are outperformed by many machine learning algorithms, especially, when massive datasets are involved. SVM classification speed scales linearly with number of support vectors, and support vectors increase with increase in dataset size. Hence, SVM classification speed can be enormously reduced if it is trained on a reduced dataset. Instance selection techniques are one of the most effective techniques suitable for minimizing SVM training time. In this study, two instance selection techniques suitable for identifying relevant training instances are proposed. The techniques are evaluated on a dataset containing 4000 emails and results obtained compared to other existing techniques. Result reveals excellent improvement in SVM classification speed.

1. Introduction

Support vector machine (SVM) has performed remarkably well in classification related and pattern recognition problems. Its high classification accuracy makes it one of the most preferred machine learning (ML) algorithms. However, SVM has high classification complexity which scales linearly with number of support vectors (SVs). SV increases with increase in dataset instances. That is, massive dataset produces many SVs and consequently increases SVM classification speed, hence making it unsuitable for real time systems. SVM training time is $O(n^2)$, where n is number of training instances [1, 2]. Instance selection techniques have been successfully used to improve SVM classification speed and training complexity. These techniques are used to minimize SVM training time by extracting relevant instances from training set. The extracted instances (also called SVs) are instances close to decision boundary. Eliminating instances that are non-SVs does not have negative impact on SVM training result [1]. This study presents two instance selection techniques and applies them to phishing email classification. A brief introduction to instance selection and phishing is presented next.

(1) Instance Selection. Numerous ML related problems require automatic classification of new instances. Prior to classification, a classifier is typically trained on a set of instances, called training set. Generally, training datasets contain redundant instances; hence, removal of these instances improves computational complexity of classifier. Instance selection techniques are designed to remove irrelevant instances from dataset. They aim at reducing training time of a classifier. Instance selection is particularly useful for instance-based classifiers, where classification of one instance involves the use of an entire training set [3]. Instance selection can start with either an empty set (incremental technique) or a training set (decremental technique) [3]. For incremental technique, instances are added into an empty set, and for decremental technique, instances are removed from training set [3]. Instance selection techniques can be classified into two: wrapper and filter [3]. Wrapper based instance selection depends on accuracy achieved by a classifier, and filter-based instance selection techniques do not depend on a classifier [3]. Filter-based instance selection techniques are typically faster than wrapper based techniques [3]. In this study, two filter-based instance selection techniques are developed and applied to email classification. Performances

of these techniques are evaluated and they yielded excellent results.

(2) Phishing. Phishing is an effort to acquire sensitive information from users by electronic means, generally for fraud. Typically, phishing is perpetrated by creating a replica website of a legitimate organization. Phishing attack is one of the major threats encountered by many online users in recent times. Since the advent of electronic commerce in 1994, phishing has advanced at a fast pace [4]. Undoubtedly, high patronage of online businesses is one of the primary causes of the raid increase in online fraud. In 2014, 1,198 companies lost 179 million US dollars to email scam [5]. Between October 2013 and August 2015, 7000 companies in USA lost about 750 million US dollars to phishing [5]. Moreover, between 2014 and 2016, total loss to email scam (by organizations) is estimated to be 2.3 billion US dollars [5]. The urgent need for a robust phishing detection system cannot be overemphasized. A secured phishing detection system should be capable of identifying and protecting users from known and novel phishing attacks [6]. Many solutions have been proposed in literature to handle phishing; however, ML-based techniques are one of the few techniques that yielded high classification accuracy, because of their ability to detect both existing and emerging fraudulent attacks. This paper proposes two improved SVM-based solutions for phishing email classification.

2. Related Works

Many instance selection techniques have been proposed in literature to reduce the computational complexity of SVM. Some proposed wrapper and filter-based instance selection techniques are presented in this section.

2.1. Wrapper Based Instance Selection Techniques. Wrapper techniques perform instance selection using a classification model [3]. During instance selection, datasets are divided into subsets, and each subset is used to train a model. Afterwards, each model is tested on a separate subset. Furthermore, weight for each subset is evaluated by calculating the number of correctly classified instances. Finally, subset with the best weight is selected and used to build the main model. Although wrapper based techniques typically select optimal subsets, the selection process is time consuming. Some proposed wrapper based instance selection techniques for SVM speed optimization are presented below.

García et al. [7] introduced an evolutionary algorithm (EA) based technique for imbalanced classification in exemplar-based learning. In the study, authors calculated the distance of each piece of data to different exemplars and used EA to select the best exemplar. The selected exemplar was then used for training. In another work, Cano et al. [8] performed a study on performance of EA-based instance selection techniques. In the study, authors focused on four EA models and compared their performance to non-EA algorithms. Result obtained from study revealed that EA performed better compared to non-EA algorithms. Li et al. [9] proposed a SVM-based instance selection technique. In the study, authors combined SVM and a KNN-based instance selection technique (called DROP2 [10]). SVM was used to select SVs, and DROP2 was used to further reduce the selected SVs. The resultant dataset was then used to train SVM. Garain [11] proposed an instance selection technique based on Artificial Immune System (AIS). In the study, authors used the idea of AIS to select the fittest set of instances from a dataset. Zhang and Sun [12] proposed a tabu search based technique for instance selection. In the study, different subsets were selected and tabu search was applied to each subset. Each subset was evaluated, and subset that produces the best classification accuracy was selected.

2.2. Filter-Based Instance Selection Techniques. Filter-based instance selection technique performs instance selection using a choice function [3]. Instance selection is performed based on scores assigned to instances. Unlike wrapper-based techniques, instance subsets produced by filter-based techniques are usually not tailored to a certain type of classification model; they are more general. Some filter-based techniques are discussed next.

Riquelme et al. [13] proposed an instance selection technique for selecting boundary instances. Authors designed a selection rule that discards weak instances far from a boundary. Weakness of an instance is determined by weakness of all attributes which describes the instance. That is, weakness(I) = n, where n is the number of features describing instance I. Lyhyaoui et al. [14] proposed a clustering-based instance selection technique for obtaining boundary instances in multiclass datasets. In the study, authors obtained boundary instances by selecting cluster centers close to opposite classes. In another work, De Almeida et al. [15] proposed a clustering-based technique using K-means algorithm. The technique was designed with an assumption that training vectors close to a separating margin are prospective SVs, and training vectors far from margin are likely non-SVs. In the study, authors divided the training dataset into different clusters. Afterwards, training vectors in clusters containing only one class were discarded (only their cluster centers were considered) and training vectors in clusters containing more than one class were selected for training. Selection was based on the assumption that clusters with multiple classes possibly contain SVs, because they are near a separating margin. In another study, Chen et al. [16] proposed a clustering-based instance selection technique. In the study, authors used clustering algorithm to obtain cluster centers of instances in a positive class. Afterwards, the cluster centers were used as reference points to select boundary instances. The algorithm was designed on the assumption that negative instance close to cluster centers of positive class and positive instance far from cluster centers of positive class are close to the boundary. In other words, positive instances close to a boundary contribute less to the decision surface and negative instances close to a boundary contribute more to the decision surface.

Panda et al. [1] proposed an instance selection technique capable of selecting data instances close to a decision boundary. The selected boundary instances are believed to be SVs. The technique consists of two stages. The first stage is

responsible for identifying a set of nearest neighbors for all instances in a dataset, and the second stage is responsible for selecting instances close to a boundary. Authors developed a scoring function that assigns high scores to instances closer to a decision boundary. In another study, García et al. [17] introduced an instance selection algorithm based on memetic algorithm. Memetic algorithm combines EA and local search. In the study, authors designed the local search to select relevant instances and also improve classification accuracy.

In this study, for comparison purpose, two of the reviewed instance selection techniques were implemented and applied to phishing emails. The two techniques (Chen et al. [16] and Panda et al. [1]) and their results are presented next.

As aforementioned, Panda et al. [1] designed a scoring function for selecting instances close to a decision boundary. The scoring function is given in

$$C\left(X_i, X_j\right) = \exp \frac{\left(-\left\|X_i - X_j\right\|_2^2\right) - \tau_j}{\gamma}. \quad (1)$$

$C(X_i, X_j)$ denotes the score accorded to X_i by X_j. τ_j is the squared distance from X_j to the closest instance of the opposite class on its neighborhood list. γ is the mean of $-\|X_i - X_j\|_2^2 - \tau_j$. During implementation in this study, squared Euclidean distance was used for distance computation. Pseudocode for the scoring function is shown in Algorithm 1, and classification step is given as follows [1]:

(i) Identify all nearest neighbors (NN) for each instance in the dataset.

(ii) Compute exponential decay score for each instance and its NN belonging to opposite class.

(iii) Determine the score for each instance.

(iv) Based on the scores, select boundary instances.

Result for the KNN-based technique is shown in Table 1. The table shows the result for varying number of K and varying number of subsets (i.e., boundary instances). The result reveals an improvement in SVM classification speed. Also, clustering-based technique proposed by Chen et al. [16] was implemented. Algorithm for the technique is shown in Algorithm 2 and classification steps are shown as follows [16]:

(i) Select instances from dataset, D, for positive class, PC.

(ii) Select instances for negative class, NC, where NC = D − PC.

(iii) Apply clustering to positive class to obtain cluster centers (or means).

(iv) Select boundary instances using obtained cluster centers. To achieve this, do the following.

(v) For each cluster center,

(a) compute distance between cluster center and selected positive instances;

(b) sort distance and remove positive instances that are close to the boundary;

(c) compute distance between cluster centers and negative class;

(d) add negative instances that are close to the boundary.

(vi) End For

(vii) Use selected positive instance for training.

As shown in Table 2, the algorithm improved SVM classification speed without degrading classification accuracy.

Notation
n = # of instances in dataset
k = # of nearest neighbors
S_{x_1} = Normalized score of instance x_1
$\#_{x_1}$ = # of contributors to the score of instance x_1
$k\text{NN}_{x_1}(x_1)$ = kth nearest neighbors of instance x_1
$D^j_{x_1}$ = squared distance between x_1 and $k\text{NN}_j(x_1)$
procedure Determine_Scores
Input: n, k, X, y
Output: S
/* **Determine exponential decay parameter** γ */
$\gamma = 0$
counter = 0
for $i = 1$ to n
  nearest-opposite-neighbor-found = false
  for $j = 1$ to k
    if $y_i \neq y_{k\text{NN}_j}(x_i)$
      if (!nearest-opposite-neighbor-found)
        nearest-opposite-neighbor-found = true
        $\tau_i = D^j_{x_1}$
      $\gamma = \gamma + D^j_{x_i} - \tau_i$
      counter = counter + 1
$\gamma = \frac{\gamma}{\text{counter}}$
/* **Determine the score of instances** */
for $i = 1$ to n
  nearest-opposite-neighbor-found = false
  for $j = 1$ to k
    if $y_i \neq y_{k\text{NN}_j}(x_i)$
      if (!nearest-opposite-neighbor-found)
        nearest-opposite-neighbor-found = true
        $\tau_i = D^j_{x_i}$
      $S_{k\text{NN}_j}(x_i) += \exp\left(-\frac{D^j_{x_i} - \tau_i}{\gamma}\right)$
      $\#_{k\text{NN}_j}(x_i)$++
for $i = 1$ to n
  $S_{x_i} = \frac{S_{x_i}}{\#_{x_i}}$
return S

Algorithm 1: Pseudocode for scoring function [1].

3. Proposed Instance Selection Techniques

This section presents two instance selection techniques proposed in this study. The first technique is based on firefly algorithm, and the second technique is based on

Table 1: Result for technique proposed by Panda et al. [1].

Number of instances in subset	K	Average prediction accuracy	FP (%)	FN (%)	Time (s)
50	50	98.95	1.20	0.00	30.81
100	50	99.90	0.11	0.00	40.14
150	50	98.70	1.49	0.00	29.59
200	50	99.08	1.06	0.00	33.31
250	50	99.63	0.43	0.00	39.26
300	50	99.78	0.26	0.00	44.16
350	50	99.78	0.26	0.00	54.76
400	50	99.78	0.26	0.00	57.14

Key: FP: false positive and FN: false negative.

Input: training set T with L classes and N instances, the ratio of selected instances r.
Output: S_c, the set of selected instances with class **c** ($1 \le c \le L$) is treated as positive and the rest classes as negative
Procedure:
(1) **for** each class c, $1 \le c \le L$
(2) $S_c \leftarrow \{x \mid \mathbf{x}$ is from class $c\}$;
(3) Perform **k**-means clustering on class c and get cluster centers $M_1, M_2, \ldots, M_k$
(4) **for** each center M_i
(5) compute $d(M_i, x)$ between M_i and each instance x;
(6) **if** $N_c \ge r \cdot (N/3) + k$
(7) get $\overline{n}_c$ instances in S_c closest to M_i and delete them from S_c, where $\overline{n}_c = (N_c - r \cdot (N/3))/k$
(8) **end if**
(9) **for** each class $l \ne c$, $1 \le l \le L$
(10) Search n_i instances of class l with least distance metrics and select them into S_c, where $n_i = (r \cdot N - n_c)/(k \cdot (L-1))$ and $n_c = \{r \cdot N/3,\ N_c \ge r \cdot N/3 + k;\ N_c,$ otherwise$\}$
(11) **end for**
(12) **end for**
(13) **endfor**

Algorithm 2: Multiclass instance selection.

edge detection in image processing. Both techniques were evaluated on a dataset consisting of 3500 ham emails and 500 phishing emails. The ham emails were obtained from SpamAssassin [18] and the phishing emails were obtained from https://monkey.org/ [19]. The datasets contain higher proportion of ham emails, because, in real world, mail users receive more legitimate emails than phishing emails. All the emails were well labelled and evenly distributed into 10 folders. Afterwards, 10-fold cross validation was performed. A brief introduction to firefly algorithm (FFA) and edge detection is discussed in Sections 3.1 and 3.2, respectively.

3.1. Dataset Processing and Feature Extraction. Prior to classification, 16 features were first extracted from emails in the dataset. Features extracted are similar to the features used in one of our previous studies [20]. Furthermore, the extracted features were normalized, and IG for all the features was calculated. Afterwards, best nine features were selected and converted to the input format required by libSVM [21], the SVM library used in this study. During classification, Gaussian transformation is used to scale down the feature vectors, to ensure that each vector has a mean of zero and a unit variance. Firefly parameters used in this study are similar to the parameters suggested by Yang [22]. Also, parameter selection technique used in this study is similar to the technique recommended by Hsu et al. [23]. More details are provided in Tables 10 and 11.

3.2. Firefly Algorithm. FFA is a nature inspired (NI) algorithm, developed by Yang [24]. It is based on the flashing behavior of fireflies [25]. Most firefly species produce short flashlight at regular intervals to attract mating partners and prey and to send warning signals to predators [24]. Firefly light intensity is inversely proportional to the square of the distance between fireflies. Additionally, as distance increases, light is absorbed in the atmosphere, and light intensity decreases [24]. Flashlight can be formulated, such that it will be associated with the value of objective function. FFA has many variants; however, this study focuses on the original algorithm, formulated using three idealized rules as follows [24]:

(1) Fireflies are unisex; hence they can be attracted to each other irrespective of their sex.
(2) Firefly attractiveness and brightness are proportional, and they decrease with respect to distance. Therefore, brighter firefly attracts less bright fireflies. Also, fireflies move randomly if they are of equal light intensity.
(3) Firefly brightness is determined by the objective function landscape.

Table 2: Result for technique proposed by Chen et al. [16].

Total email	APA (%)	GB	FP (%)	FN (%)	R (%)	Pr (%)	FM (%)	T (s)
4000	99.98	100	0.03	0.00	100	99.80	99.80	1438.39

Key: GB: global best, FP: false positive, FN: false negative, APA: average prediction accuracy, R: recall, Pr: precision, FM: F-measure, and T: time.

Table 3: Result for FFA_Clus.

Total email	NFF	NI	APA (%)	GB	FP (%)	FN (%)	R (%)	Pr (%)	FM (%)	T (s)
4000	5	100	99.98	100	0.03	0.00	100	99.80	99.90	29.66
4000	5	150	99.95	100	0.06	0.00	100	99.61	99.61	45.69
4000	5	200	99.83	100	0.20	0.00	100	98.67	99.32	61.92
4000	5	250	99.78	100	0.26	0.00	100	98.29	99.12	83.92
4000	5	300	99.93	100	0.09	0.00	100	99.41	99.70	121.38
4000	10	100	99.90	100	0.11	0.00	100	99.22	99.60	29.87
4000	10	150	99.93	100	0.09	0.00	100	99.42	99.70	52.65

Key: NFF: number of fireflies, NI: number of instances, GB: global best, APA: average prediction accuracy, FP: false positive, FN: false negative, R: recall, Pr: precision, FM: F-measure, and T: time.

3.2.1. FFA-Based Instance Selection Technique. This study introduces an instance selection technique (called FFA_IS) based on FFA. FFA_IS is designed with an objective of minimizing the number of instances used for training. Given a set of training instances, fireflies are evaluated (using the objective function defined in (2)), and the best firefly is selected and used to train SVM. Each firefly consists of a binary array of N instances (called instance mask), where 1 indicates that an instance is selected, and 0 indicates otherwise. During experiment, instance mask for each firefly is randomly initialized to 0 and 1. Afterwards, objective function for each firefly is evaluated and the global best is saved. Furthermore, fireflies are moved from one position to another, their attractiveness is calculated, and their fitness value is updated. The process is repeated until a predefined number of generations are reached. Finally, the best firefly is selected, its instance mask is processed, and instances with the value of 1 are selected and used to train SVM. A constrain is added to ensure that at least N number of instances are selected for training, where N is user defined. Hence, if the total number of selected instances (T) is less than N, P additional instances are randomly selected, where $P = N - T$. This constraint is added to eliminate the possibility of having zero selected instances.

3.2.2. Objective Function for FFA_IS. Objective function for FFA_IS is given in (2). As aforementioned, the ultimate goal is to minimize number of selected instances; hence percentage reduction is the criteria used in designing the objective function. Objective function assigns more weight to fireflies with low percentage reduction. Firefly with the highest weight is selected and used for training. As aforementioned, FFA_IS is designed to ensure that at least N instances are selected for training.

$$\text{fitness}_i = 100 * \frac{(\text{TNI} * \text{TS})}{\text{TNI}}, \tag{2}$$

where TNI is size of instance mask and TS is total number of selected instances.

3.2.3. Result and Discussion. FFA_IS algorithm (shown in Algorithm 3) was evaluated on a dataset consisting of 4000 emails, and it yielded promising results. During evaluation, different experiments were performed. For each experiment, different sizes of instance masks and different number of fireflies were used. As shown in Table 4, classification accuracy obtained ranges between 99.25% and 99.68%. Moreover, speed obtained ranges between 23.54 seconds and 213.17 seconds. Although the proposed technique is not designed to select boundary instances, as reflected in the result, it can be used to select relevant instances for SVM training and consequently improve SVM classification speed. During the experiments performed on the clustering-based technique (proposed by Chen et al. [16]), it was observed that over 80% of training dataset was selected for training. Hence, in this study, FFA_IS was used to further reduce the number of training instances selected by the clustering-based technique. Two sets of experiments were performed to test the performance of the hybridized technique (called FFA_Clus). In the first set, 100% of the selected instances were used to train SVM, and in the second set, instances selected by FFA_Clus were used to train SVM. Table 3 shows the results of the experiments. Result reveals that FFA_Clus improved the classification speed of the clustering-based technique by 98%, without degrading the classification accuracy. This implies that robust instance selection techniques can be developed using FFA_IS in combination with clustering-based techniques.

3.3. Edge Detection. Edge detection in image processing is a technique used to identify object boundaries in images [26]. Object boundaries are points in images with sharp change in image brightness [26]. Generally, images contain some quantity of redundant data that requires pruning, for effective classification. Hence, to reduce computational complexity, edge detection is a highly essential preprocessing step [27]. Edge detection is applied to images with the aim of identifying important features, removing less relevant information, and consequently reducing the image size. Generally, edge detection is used for segmentation of images,

Notation
N = number of fireflies
NS = number of selected instances
M = Max Generation
Min = Minimum number of selected instances
β_0 = Initial Attractiveness Value
α = alpha
γ = Gamma
$G(x)$ = Objective function, where $x = (X_1, \ldots, X_n)$ and n = number of fireflies
IM = Instance Mask or Subset of instances.
S = Size of Instance Mask
D = Dataset
GB = Global Best
TS = Training subset
L = light intensity
Input: N, D, M, β_0, α, γ, S
Output: TS
(1) Define $G(x)$
(2) Initialize IM of each firefly
(3) Evalute $G(x)$ to determine L for each firefly
(4) Select firefly with the highest L, and save in GB
(5) **while** ($i < M$)
  (5.1) **for** $j = 1$ to N
    (5.1.1) **for** $k = 1$ to N
      (5.1.1.1) **If** ($L_j < L_k$)
        (5.1.1.1.1) Move firefly j towards firefly k
      (5.1.1.2) **end if**
      (5.1.1.3) Calculate attrativeness variance with distance r using $\exp(-\gamma r)$
      (5.1.1.4) Evaluate $G(x)$ to determine new fitness value for firefly
      (5.1.1.5) Update light intensity of firefly
    (5.1.2) **end** k
  (5.2) **end** j
  (5.3) Update GB
(6) **end while**
(7) Calculate NS of GB
(8) **if** NS in GB < Min
  (8.1) update GB by assigning 1 to (Min – NS) instances that was not selected
(9) **End if**
(10) **For** $i = 1$ to S
  (10.1) **If** IM_i in GB is equal to 1
    (10.1.1) TS = IM_i
  (10.2) **End if**
(11) **Output** TS

Algorithm 3: FFA_instance_selection.

Table 4: Result for FFA_IS.

Total email	NFF	NI	APA (%)	GB	FP (%)	FN (%)	R (%)	Pr (%)	FM (%)	T (s)
4000	5	100	99.25	100	0.29	4.00	96.00	97.98	96.83	23.54
4000	5	150	99.55	100	0.20	2.20	97.80	98.67	98.11	39.07
4000	5	200	99.55	100	0.11	2.80	97.20	99.20	98.08	62.24
4000	5	250	99.60	100	0.17	2.00	98.00	98.83	98.32	79.12
4000	5	300	99.65	100	0.11	2.00	98.00	99.22	98.49	109.70
4000	10	100	99.38	100	0.17	3.80	96.20	98.87	97.28	28.57
4000	10	150	99.55	100	0.26	1.80	98.20	98.29	98.13	45.94
4000	10	200	99.63	100	0.23	1.40	98.60	98.46	98.46	84.30
4000	10	250	99.68	100	0.09	2.00	98.00	99.42	98.59	117.98
4000	10	300	99.60	100	0.17	2.00	98.00	98.83	98.30	213.17

Key: NFF: number of fireflies, NI: number of instances, GB: global best, APA: average prediction accuracy, FP: false positive, FN: false negative, R: recall, Pr: precision, FM: F-measure, and T: time.

Notation
N = number of dataset instances
X = Dataset
K = number of nearest neighbors
E = Edge
EI = Edge Instances
V = Vote for each instance. Vote is an array of size N.
Input: N, X, K
Output: EI
Initialize V
(1) **For** i = 1 to N
  (1.1) **For** j = 1 to N
    (1.1.1) Compute distance between X_i and X_j, where $i \neq j$
  (1.2) **End** j
  (1.3) Select Instance X_j with largest distance
  (1.4) Increment V for X_j
(2) **End** i
(3) Select instance X_i with highest vote
(4) $E = X_i$
(5) **For** M = 1 to K
  (5.1) Select K-nearest neighbors for EI and save in EI
(6) **End** M
(7) **Return** EI

Algorithm 4: Edge instance selection algorithm.

Figure 1: Example of edge detection [26].

feature extraction, and feature detection in image processing, computer vision, and machine vision [26–28]. Edge defection conserves essential structural properties of images and computer space [27]. Edge detection algorithms include Canny algorithm, Sobel algorithm, and Roberts algorithm. Figure 1 shows an example of an image and its detected edges.

Concept of edge detection in image processing is similar to concept of boundary detection in SVM classification. Edge detection aims to select objects located at boundary positions, and boundary detection algorithms aim to select instances (also called SVs) close to a decision boundary. In this study, an instance selection technique based on edge detection is proposed.

3.3.1. Edge Instance Selection Algorithm. This study proposes an instance selection technique called Edge Instance Selection Algorithm (EISA). EISA borrows its idea from edge detection in image processing. Given a set of training instances, EISA identifies an edge instance and selects N instances close to it. EISA consists of two main stages: distance computation stage and edge instance selection stage. In the first stage, EISA computes squared Euclidian distance between each instance and all other instances in the dataset. Furthermore, in this stage, for each instance_i in the dataset, based on the proximity of other instances to instance_i, EISA votes a corresponding instance_k (i.e., edge instance), where instance_k is the farthest distance from instance_i. In the second phase, firstly, edge instance with the highest vote is selected. Afterwards, K-nearest neighbors of the voted edge instance are computed and used to train SVM model. Algorithm 4 shows the full algorithm of EISA. Some experiments were performed to test the efficiency of EISA, and result reveals that EISA significantly improved SVM classification speed.

Table 5: Result for EISA.

Total email	K	APA (%)	GB	FP (%)	FN (%)	R (%)	Pr (%)	FM (%)	T (sec)
4000	100	99.50	100	0.03	3.80	96.20	99.80	97.89	130.69
4000	150	99.90	100	0.11	0.00	100	99.24	99.61	199.61
4000	200	99.90	100	0.11	0.00	100	99.24	99.61	231.91
4000	250	99.98	100	0.03	0.00	100	99.80	99.90	184.35
4000	300	100	100	0.00	0.00	100	100	100	191.62
4000	350	100	100	0.00	0.00	100	100	100	228.55
4000	400	100	100	0.00	0.00	100	100	100	269.39

Key: NFF: number of fireflies, NI: number of instances, GB: global best, APA: average prediction accuracy, FP: false positive, FN: false negative, R: recall, Pr: precision, FM: F-measure, and T: time.

Table 6: Comparison between techniques.

Total email	Technique	APA (%)	GB	FP (%)	FN (%)	R (%)	Pr (%)	FM (%)	T (s)
4000	EISA	100	100	0.00	0.00	100	100	100	191.62
4000	Clustering	99.98	100	0.03	0.00	100	99.80	99.80	1438.39
4000	KNN	99.78	100	0.26	0.00	100	98.27	99.12	47.73
4000	FFA	99.55	100	0.26	1.80	98.20	98.29	98.13	45.94
4000	FFA_Clus	99.98	100	0.03	0.00	100	99.80	99.90	29.66
4000	SVM	99.98	100	0.02	0.00	100	99.86	99.93	2544.72

Key: GB: global best, APA: average prediction accuracy, FP: false positive, FN: false negative, R: recall, Pr: precision, FM: F-measure, and T: time.

3.3.2. Results and Discussion. EISA was evaluated on a dataset consisting of 4000 emails. During evaluation, different values of K were used, and as shown in Table 5, EISA produced a classification accuracy of 100% and FP and FN rate of 0.00%, when $K \geq 300$. This implies that 300 edge instances are sufficient to build an excellent SVM classifier. Furthermore, results obtained from EISA, FFA_IS, FFA_Clus, KNN-based technique, and clustering-based technique were compared to each other. Also, to evaluate the impact of instance selection on SVM, an additional experiment was performed. In the experiment, all training instances were used to train SVM; no instance selection technique was applied prior to training. Result obtained from the experiment was also compared to the techniques implemented in this study. As shown in Table 6, EISA produced the best classification accuracy, FP rate, and FN rate, followed by clustering-based technique and FFA_Clus. Furthermore, in terms of classification speed, all the results obtained show enormous improvement in SVM classification speed. EISA improved SVM speed by 92.5%, and FFA_Clus improved SVM speed by 98.8%. Furthermore, clustering-based technique, KNN-based technique, and FFA improved SVM classification speed by 43%, 98.1%, and 98.2%, respectively. Overall, EISA and FFA_Clus produced the best speed-accuracy trade-off compared to the other techniques.

Another set of experiments was performed on Spambase dataset, consisting of 4600 emails and 57 features. Spambase was obtained from UCI ML repository [29]. The experiment was performed with the aim of comparing the performance of the proposed techniques to other instance selection techniques in literature. In the experiment, EISA, FFA_IS, and FFA_Clus were compared to five other instance selection techniques, namely, PSC [30], DROP 3 [10], DROP 5 [10], GCNN [31], and POCNN [32]. Results for the experiment are shown in Table 7 and Figures 2 and 3. As shown,

Table 7: Comparison between techniques, Spambase.

Technique	APA (%)	T (s)
EISA	78.83	496.15
FFA	83.59	73.58
FFA_Clus	92.35	82.28
PSC	71.95	189.57
DROP 3	78.44	3782.57
DROP 5	78.72	2226.42
GCNN	73.54	348.56
POCNN	75.37	735.08

Key: APA: average classification accuracy, T: time.

FFA_Clus and FFA yielded the best performance, in terms of classification accuracy and classification speed. Moreover, in terms of classification accuracy, EISA performed better than PSC, GCNN, DROP 3, and POCNN. Although, in terms of classification accuracy, DROP 5 performed better than EISA, EISA has better speed-accuracy trade-off. EISA is also faster than POCNN.

Statistical analysis of the results was performed using one-sample t-test. The goal of the statistical analysis is to know whether it can be concluded, with 95% confidence level, that the proposed technique performs better (in terms of classification speed and accuracy) than PSC, DROP 3, DROP 5, GCNN, and POCNN. As aforementioned, 10-fold cross validation was performed, hence the reason for using t-test. Also, since the number of samples is 10, from t-test table, critical value is 2.2622. Result of the analysis is reported in Tables 8 and 9. As shown in Table 8, in terms of classification accuracy, there is a statistically significant difference between EISA and PSC. There is also a statically significant

Table 8: Statistical analysis, classification accuracy.

Technique	Number of samples	Standard variation	t-test values ($\alpha = 0.05$)
EISA versus PSC	10	7.51404	−2.8953
EISA versus DROP 3	10	7.51404	−0.1641
EISA versus DROP 5	10	7.51404	−0.0462
EISA versus GCNN	10	7.51404	−2.0158
EISA versus POCNN	10	7.51404	−1.2457
FFA versus PSC	10	3.5072	−10.4952
FFA versus DROP 3	10	3.5072	−4.6438
FFA versus DROP 5	10	3.5072	−4.3911
FFA versus GCNN	10	3.5072	−9.0616
FFA versus POCNN	10	3.5072	−7.4116
FFA_Clus versus PSC	10	0.7447	−86.626
FFA_Clus versus DROP 3	10	0.7447	−59.0670
FFA_Clus versus DROP 5	10	0.7447	−57.8781
FFA_Clus versus GCNN	10	0.7447	−79.8520
FFA_Clus versus POCNN	10	0.7447	−72.1035

Table 9: Statistical analysis, classification speed.

Technique	Number of samples	Standard variation	t-test values ($\alpha = 0.05$)
EISA versus PSC	10	1.8080	−537.22
EISA versus DROP 3	10	1.8080	5748.10
EISA versus DROP 5	10	1.8080	3026.32
EISA versus GCNN	10	1.8080	−258.14
EISA versus POCNN	10	1.8080	0.2591
FFA versus PSC	10	0.5671	3.130
FFA versus DROP 3	10	0.5671	20682.17
FFA versus DROP 5	10	0.5671	12004.72
FFA versus GCNN	10	0.5671	1533.35
FFA versus POCNN	10	0.5671	3688.67
FFA_Clus versus PSC	10	0.5691	593.00
FFA_Clus versus DROP 3	10	0.5691	20561.13
FFA_Clus versus DROP 5	10	0.5691	11914.19
FFA_Clus versus GCNN	10	0.5691	1479.62
FFA_Clus versus POCNN	10	0.5691	3627.37

Table 10: SVM parameters used for evaluations.

SVM parameters [23]							
	$C=$	2^{-11}	2^{-9}	$\cdots$	2^{1}	2^{3}	2^{5}
	$\gamma=$	2^{-5}	2^{-3}	$\cdots$	2^{7}	2^{9}	2^{11}

C = regularization constant and γ = gamma.

Table 11: FFA parameters used for evaluations.

Firefly parameters [22]	α	γ	β_0	N_g
	0.2	1	1	5

Key: α = alpha, γ = gamma, β_0 = beta, and N_g = number of generations.

difference between FFA and PSC, DROP 3, DROP 5, GCNN, and POCNN. Moreover, there is a statistically significant difference between FFA_Clus and PSC, DROP 3, DROP 5, GCNN, and POCNN. Furthermore, as shown in Table 9, in terms of classification speed, there is a statistically significant difference between EISA and DROP 3 and DROP 5. There is also a statistically significant difference between FFA and PSC, DROP 3, DROP 5, GCNN, and POCNN. Moreover, there is a statistically significant difference between FFA_Clus and PSC, DROP 3, DROP 5, GCNN, and POCNN.

4. Conclusion and Future work

Instance selection techniques have been successfully used to reduce SVM speed complexity. Two types of instance selection techniques include filter and wrapper. Filter-based instance selection techniques are generally faster than wrapper based techniques. In this study, two filter-based instance selection techniques are introduced. Performance of the two techniques was evaluated, and result was compared to other existing techniques. Results reveal excellent improvement in SVM classification speed without significant reduction in classification accuracy. Moreover, the two techniques

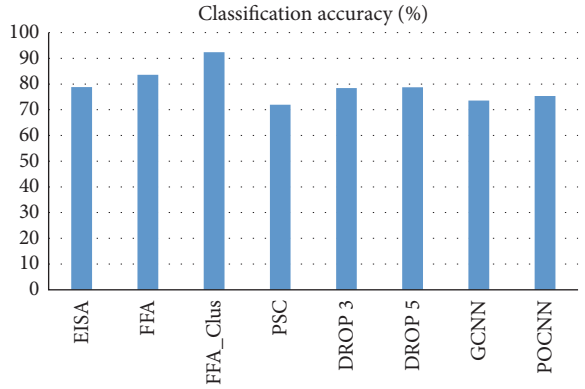

Figure 2: Comparison between classification accuracies of techniques.

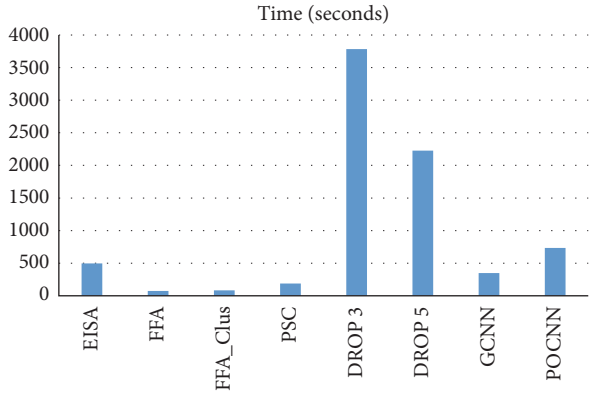

Figure 3: Comparison between classification speeds of techniques.

produced balanced speed-accuracy trade-offs. In the future, the two proposed techniques will be tested on other ML algorithms, and more NI-based instance selection techniques will be developed and tested.

Competing Interests

The authors declare that they have no competing interests.

References

[1] N. Panda, E. Y. Chang, and G. Wu, "Concept boundary detection for speeding up SVMs," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 681–688, Pittsburgh, Pa, USA, June 2006.

[2] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *The Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2001.

[3] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[4] K. Lamond, "Credit Card Transactions Real World and Online," 1996, http://www.virtualschool.edu/mon/ElectronicProperty/klamond/credit_card.htm.

[5] KrebsOnSecurity, "FBI: $1.2B Lost to Business Email Scams," 2015, http://krebsonsecurity.com/2015/08/fbi-1-2b-lost-to-business-email-scams/.

[6] RSASecurity, "2013 A Year in Review," 2014, http://www.emc.com/collateral/fraud-report/rsa-online-fraud-report-012014.pdf.

[7] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 3–12, 2012.

[8] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2003.

[9] Y. Li, Z. Hu, Y. Cai, and W. Zhang, "Support vector based prototype selection method for nearest neighbor rules," in *Proceedings of the International Conference on Natural Computation*, pp. 528–535, Berlin, Germany, 2005.

[10] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[11] U. Garain, "Prototype reduction using an artificial immune model," *Pattern Analysis and Applications*, vol. 11, no. 3-4, pp. 353–363, 2008.

[12] H. Zhang and G. Sun, "Optimal reference subset selection for nearest neighbor classification by tabu search," *Pattern Recognition*, vol. 35, no. 7, pp. 1481–1490, 2002.

[13] J. C. Riquelme, J. S. Aguilar-Ruiz, and M. Toro, "Finding representative patterns with ordered projections," *Pattern Recognition*, vol. 36, no. 4, pp. 1009–1018, 2003.

[14] A. Lyhyaoui, M. Martínez, I. Mora, M. Vázquez, J.-L. Sancho, and A. R. Figueiras-Vidal, "Sample selection via clustering to construct support vector-like classifiers," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1474–1481, 1999.

[15] M. B. De Almeida, A. De Pádua Braga, and J. P. Braga, "SVM-KM: speeding SVMs learning with a priori cluster selection and k-means," in *Proceedings of the 6th Brazilian Symposium on Neural Networks (SBRN '00)*, pp. 162–167, IEEE, Rio de Janeiro, Brazil, November 2000.

[16] J. Chen, C. Zhang, X. Xue, and C.-L. Liu, "Fast instance selection for speeding up support vector machines," *Knowledge-Based Systems*, vol. 45, pp. 1–7, 2013.

[17] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: a scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693–2709, 2008.

[18] C. Group, "SpamAssassin Data," 2006, http://www.csmining.org/index.php/spam-assassin-datasets.html.

[19] J. Nazario, "Phishing Corpus," 2009, http://www.monkey.org/jose/wiki/doku.php?id=phishingcorpus.

[20] A. A. Akinyelu and A. O. Adewumi, "Classification of phishing email using random forest machine learning technique," *Journal of Applied Mathematics*, vol. 2014, Article ID 425731, 6 pages, 2014.

[21] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, pp. 1–27, 2011.

[22] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2010.

[23] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, *A Practical Guide to Support Vector Classification*, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2003.

[24] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Beckington, UK, 2008.

[25] X. S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013.

[26] MathWorks, "Edge Detection," 2016, http://www.mathworks.com/discovery/edge-detection.html.

[27] D. Ray, "Edge Detection in Digital Image Processing," 2013, https://www.math.washington.edu/~morrow/336_13/papers/debosmit.pdf.

[28] S. E. Umbaugh, *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools*, CRC Press, 2010.

[29] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2007, http://archive.ics.uci.edu/ml/datasets.html.

[30] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis and Applications*, vol. 13, no. 2, pp. 131–141, 2010.

[31] C.-H. Chou, B.-H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as a data reduction method," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, pp. 556–559, IEEE, Hong Kong, August 2006.

[32] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm," *Pattern Recognition Letters*, vol. 26, no. 10, pp. 1554–1567, 2005.