Binary PSO with Mutation Operator for Feature Selection using Decision Tree applied to Spam Detection

Yudong ZHANG,1,2 Shuishua WANG,1,2 Preetha PHILLIPS,3 Genlin JI,1

1 School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210023, China
2 School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu 210046, China
3 School of Natural Sciences and Mathematics, Shepherd University, Shepherdstown, West Virginia, 25443, USA

Email: zhangyudong@njnu.edu.cn, wangshuishua@njnu.edu.cn, preetha.phillips@gmail.com, glj@njnu.edu.cn

Abstract: In this paper, we proposed a novel spam detection method that focused on reducing the false positive error of mislabeling nonspam as spam. First, we used the wrapper-based feature selection method to extract crucial features. Second, the decision tree was chosen as the classifier model with C4.5 as the training algorithm. Third, the cost matrix was introduced to give different weights to two error types, i.e., the false positive and the false negative errors. We define the weight parameter as \( \alpha \) to adjust the relative importance of the two error types. Fourth, K-fold cross validation was employed to reduce out-of-sample error. Finally, the binary PSO with mutation operator (MBPSO) was used as the subset search strategy. Our experimental dataset contains 6000 emails, which were collected during the year of 2012. We conducted a Kolmogorov-Smirnov hypothesis test on the capital-run-length related features and found that all the \( p \) values were less than 0.001. Afterwards, we found \( \alpha = 7 \) was the most appropriate in our model. Among seven meta-heuristic algorithms, we demonstrated the MBPSO is superior to GA, RSA, PSO, and BPSO in terms of classification performance. The sensitivity, specificity, and accuracy of the decision tree with feature selection by MBPSO were 91.02%, 97.51%, and 94.27%, respectively. We also compared the MBPSO with conventional feature selection methods such as SFS and SBS. The results showed that the MBPSO performs better than SFS and SBS. We also demonstrated that wrappers are more effective than filters with regard to classification performance indexes. It was clearly shown that the proposed method is effective, and it can reduce the false positive error without compromising the sensitivity and accuracy values.

Keyword: Spam Detection; Binary Particle Swarm Optimization; Mutation Operator; Feature Selection; Wrapper; Premature Convergence; Decision Tree; Cost Matrix

1 Introduction

It is a common occurrence for a user to receive hundreds of emails daily. Nearly 92% of these emails are spam [1]. They include advertisements for a variety of products and services, such as pharmaceuticals, electronics, software, jewelry, stocks, gambling, loans, pornography, phishing, and malware attempts [2]. The spam not only consumes the users’ time by forcing them to identify the undesired messages, but also wastes mailbox space and network bandwidth. Therefore, spam detection is becoming a bigger challenge to process for individuals and organizations.

Researchers have proposed different features of extraction technology. The TF-IDF (Term Frequency and Inverse Document Frequency) method extracts features by splitting each message into tokens based on spaces, tabs, and symbols [3]. A simpler model that can be used is by only considering individual keywords [4]. Other more complex models include tag-based features [5] and behavior-based features [6]. In this paper, we found that spam is likely to contain more captain letters, such as the following examples: “85% DISCOUNT ONLY for YOU” and “The LAST DAY to”. Therefore, it is natural to consider the statistical measures of captain letters.

Afterwards, classifications were done using machine learning approaches including artificial immune system [7], support vector machine (SVM) [8], artificial neural networks (ANN) [9], and case-based technique [10]. However, these methods neither achieve classification accuracy, nor give physical meanings. Naive Bayes classifier [11] assumes the features contribute independently, which is oversimplified for this study [12].

In this paper, we proposed a hybrid system that combined the feature selection method and decision tree. The key advantage of our method is that it can achieve high classification accuracy and also give physical meanings to the users. Another important advantage to the method is that it discriminates the two types of errors. One of these errors is predicting a spam as a normal message that causes the message to be automatically filtered on the server. However, this is not that problematic because users can just delete them manually. On the other hand, the error of predicting a normal message as a spam can be very harmful. In fact, these messages are automatically transferred to spam box, and the user is not informed about this transfer. Therefore, it is possible for a very important message to be mistakenly transferred to the spam box. These two different errors should be
taken into account differently.
We started the paper by describing the methodology in section 2. We first presented the complete original feature set, then introduced the wrapper-based feature selection, brought in the classifier model, and discussed the search strategy. Section 3 contained the experiments on 6000 emails collected during 2012. We demonstrated the effectiveness of capital-run-length related features, and compared the proposed method with other meta-heuristics algorithms and feature selection algorithms, and with other spam detection algorithms, respectively by using terms of classification performance and computation time. In addition, we showed how to choose the optimal weight parameter in our model, and demonstrated the superiority of wrappers over filters. Section 4 discussed the results and analyzed their underlying reasons. Final section 5 concluded the paper.

2 Methodology

2.1 Complete Feature Set

In this section, we discuss how to establish the original feature set from emails. We collected 6000 emails by using the same standard as the UCI machine learning repository did (http://archive.ics.uci.edu/ml/datasets/Spambase), except we changed “1999” to “2012”. This was because their dataset was obtained in 1999 and our dataset picked up emails received and sent in 2012. The features contain three different types (Table 1). The first is the frequency of 48 common words. The second is the frequency of 6 characters. The third type contains three capital-run-length related features. Following the procedure stated above, each email was transformed to a 48+6+3=57 dimensional feature vector.

Table 1 57 original features

<table>
<thead>
<tr>
<th>Types</th>
<th>Number of Features</th>
<th>Content</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency of Common Word</td>
<td>48</td>
<td>make, address, all, 3d, our, over, remove, internet, order, mail, receive, will, people, report, addresses, free, business, email, you, credit, your, font, 000, money, hp, hpl, george, 650, lab, labs, telnet, 857, data, 415, 85, technology, 2012, parts, pm, direct, cs, meeting, original, project, re, edu, table, conference</td>
<td>Continuous real [0, 1]</td>
</tr>
<tr>
<td>Character</td>
<td>6</td>
<td>[: ! $ #]</td>
<td>Continuous real [0, 1]</td>
</tr>
<tr>
<td>capital-run-length</td>
<td>3</td>
<td>Average, Longest, Total</td>
<td>Integer</td>
</tr>
</tbody>
</table>

The capital-run-length (last row in Table 1) was chosen as an important source of feature sets. As it is known that most spams use long unintermitted capital letters to attract the attention of users, as shown within the red borders in Figure 1. The detailed analysis was performed in this experiment.

ENDS MONDAY! FREE Shipping + FREE moisturizer mini!* (a)

POST SCRITUM: You have to keep everything secret as to enable the transfer to move very smoothly into the account you will prove to the bank. As you finished reading this letter call me immediately so that we discuss very well over this business. (b)

Figure 1 Two examples of spam with long unintermitted capital letters: (a) Example 1, here the spam contains “ENDS MONDAY” & “FREE”, (b) Example 2, here the spam contains “POST SCRITUM”.

2.2 The wrapper-based FS

Classification methods often begin with some type of dimension reduction, in which the data is approximated by points in a lower-dimensional space. There are two general approaches for performing dimensionality reduction: feature extraction (FE) and feature selection (FS). FE transforms the existing features into a lower dimension space, while FS selects a subset of the existing features without a transformation. The differences of these two approaches are shown in Figure 2.

Figure 2 Differences of (a) FS and (b) FE. Here \( N \) represents the number of original features, and \( M \)
represents the number of reduced features. $M < N$.

In this paper, FS was chosen over FE due to the fact that the users tend to use meaningful rules from the classifier. In addition, fewer features meant fewer parameters for the classifier, which would improve the generalization capabilities and reduce the complexity of the calculation.

![Outline of (a) FS, and its two variants: (b) Filters & (c) Wrappers](image)

There are essentially two requirements on the FS as shown in Figure 3(a), one requirement is an objective function and the other requirement is a search strategy. The objective functions are divided in two groups, i.e., filters and wrappers, shown in Figure 3(b-c), respectively. Their corresponding methods are called filter-based FS and wrapper-based FS. The filters evaluate feature subsets by their information content, typically the distance, correlation, statistical dependence or information-theoretic measures [13]. Alternatively, the wrappers set the objective function as a pattern classifier, which evaluates feature subsets by their predictive measure [14]. Their advantages and disadvantages are listed and shown in Table 2.

<table>
<thead>
<tr>
<th>Objective Function</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filters</td>
<td>Fast execution &amp; Generality</td>
<td>Tendency to select large subsets</td>
</tr>
<tr>
<td>Wrappers</td>
<td>Accuracy</td>
<td>Slow execution &amp; Lack of generality</td>
</tr>
</tbody>
</table>

In this study, the wrappers were chosen as the objective function because of its high classification accuracy. However, there are some disadvantages of using wrappers. The first disadvantage (slow execution) can be solved by an introduced global optimization method. The second disadvantage (lack of generality) can be solved by K-fold cross validation. From Figure 3(c), we saw that the wrapper method contain two essential elements, the classifier and search strategy. They will be discussed in sections 2.3 and 2.4, respectively.

### 2.3 Decision Tree

There are many classifiers such as statistical classifier, neural network classifier [15], syntactic classifier etc. In this paper, we chose a tree-based classifier because of its simple properties, the explicit meaning, and easily transformation to “if-then” rules [16, 17]. The definition of a decision tree is a decision support tool that uses a dendritic graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility [18]. Decision trees are commonly used in operations research, specifically in decision analysis in order to help identify a strategy most likely to reach a specific goal. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

#### 2.3.1 Cost Matrix

Mislabeling spam has different side effects. A spam marked as non-spam will leave a burden on the users who need to read through and delete it. However, a non-spam that is marked as spam, is usually followed by the sequence of automatic deletion causing the user to lose important email or with automatic transfer to the spam box the user will most likely ignore it. Therefore, we need to use the confusion matrix and cost matrix to describe different types of errors and to measure how serious they are, respectively.

In the confusion matrix (Table 3), each column represents the instances in a predicted class and each row represents the instances in an actual class. A specific table layout allows visualization of the performance of a classifier. In this paper, the positive class represented the spam and the negative class represented the nonspam.
Here, TP represented true positives, FP represented false positives, FN represented false negatives, and TN represented true negatives.

<table>
<thead>
<tr>
<th>Table 3 Confusion matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Spam prediction</td>
</tr>
<tr>
<td>(Actual Spam that were correctly classified as spams)</td>
</tr>
<tr>
<td>Nonspam prediction</td>
</tr>
<tr>
<td>(Spam that were incorrectly classified as nonspams)</td>
</tr>
</tbody>
</table>

The two errors rates are FN and FP. In this paper, FN describes the number of spam that was incorrectly classified as nonspam, and FP describes the number of nonspams that were incorrectly classified as spam. The cost matrix was useful in adjusting the values of the two types of errors. Generally, using cost matrix will lead to the number of total errors that increases while the cost of the errors decreases. The cost matrix in this study was defined as

\[
\text{CostMatrix} = \begin{bmatrix}
0 & \alpha \\
1 & 1 + \alpha \\
\frac{1}{1+\alpha} & 0
\end{bmatrix}
\]

(1)

Here \(\alpha\) was the weight parameter measuring the degree of seriousness of FP compared to FN. The cost of TP and TN are zero since they represent the correct classification. The cost of FP is set as \(\alpha\) times as the cost of FN. Usually \(\alpha > 1\) because we wanted to avoid to labeling nonspam as spam. The total cost was set as

\[
C = \text{sum} \left[ \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \times \begin{bmatrix}
0 & \alpha \\
1 & 1 + \alpha \\
\frac{1}{1+\alpha} & 0
\end{bmatrix} \right]
\]

\[
\frac{FN + \alpha \times FP}{1 + \alpha}
\]

(2)

Here “\(\times\)” denotes element-wise multiplication.

### 2.3.2 Training Algorithm

Traditional methods of training a decision tree dates back to ID3 [19]. Subsequently, Quinlan proposed the famous C4.5 algorithms that built decision trees from a set of training data in the same way as ID3 using the concept of information entropy. At each node of the tree, C4.5 chose one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion was the normalized information gain [20] that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain was chosen to make the decision. The C4.5 algorithm then recurs on the smaller sub-lists.

This algorithm has three base cases. First, all the samples in the list belong to the same class. When this happens, it simply created a leaf node for the decision tree saying to choose that class. Second, none of the features provided any information gain. In this case, C4.5 created a decision node higher up the tree using the expected value of the class. Third, the instance of previously unseen class encountered. Again, C4.5 created a decision node higher up the tree using the expected value. Upon above three base cases, the pseudo-codes of the C4.5 were written as follows.

**Step 1** Check for base cases;
**Step 2** For each attribute \(a_i\), find the normalized information gain was found from splitting on \(a_i\);
**Step 3** Let \(a_{\text{best}}\) be the attribute with the highest normalized information gain;
**Step 4** Create a decision node that splits on \(a_{\text{best}}\);
**Step 5** Recur to Step 1 on the sub-l lists obtained by splitting on \(a_{\text{best}}\), and add those nodes as children of current node.

### 2.3.3 K-fold Cross validation

Obviously if there were no conflicting cases, the decision tree would correctly classify all training cases. This “overfitting” is generally thought to lead to a loss of predictive accuracy in most applications [21]. Traditional generalization methods focus on the pruning technique. Those pruning algorithms based on posterior calculations such as BIC (MDL) and MEP are faster [22], but they sometimes produce too big or small trees that yield poor generalization errors [23]. Therefore, in this study cross-validation strategy was employed other than the pruning technique. The error rate on the training set without validation technique is called “in-sample” error
estimate, whereas, the cross-validation estimate is an “out-of-sample” error estimate.

Cross validation methods consist of three types: random subsampling, K-fold cross validation, and leave-one-out validation. The K-fold cross validation is applied due to its properties being simple, easy, and using all data for training and validation. The mechanism was first to create a K-fold partition of the whole dataset, repeat K times to use K-1 folds for training and a left fold for validation, and finally average the error rates of K experiments.

A challenge was to determine the number of folds. If K was set too large, the bias of the true error rate estimator would be small; however, the variance of the estimator would be large and the computation would be time-consuming. Alternatively, if K was too small, the computation time would decrease, causing the variance of the estimator to be small, but the bias of the estimator would be large. The advantages and disadvantages of setting K large or small are listed in Table 4. In this study, K is set as 15 to compromise the corresponding bias and variance of the estimator [24]. The corresponding schematic diagram is shown in Figure 4.

### Table 4 Large K versus small K

<table>
<thead>
<tr>
<th>K value</th>
<th>Estimator Bias</th>
<th>Estimator Variance</th>
<th>Computation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>↓</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>small</td>
<td>↑</td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>

### 2.4 Search Strategy

Suppose we are selecting the best subset out of N features, the feasible combinations would be $2^N$. For example, an exhaustive algorithm of the original 57 email features involved $3.6 \times 10^{16}$ subsets (see Table 1). Therefore, a powerful search strategy was needed to direct the FS process because it explores the space of all possible combinations. There were increasingly more search strategies of which can be grouped in three categories [25]. Table 5 gives the state-of-the-art methods of FS. The first category is the exponential algorithm. These algorithms evaluated a number of subsets that grew exponentially with the dimensionality of the search space such as the exhaustive search. The second category is the sequential algorithms, which added or removed features sequentially, but had a tendency to become trapped in local minima. Typical algorithms included sequential forward selection (SFS) and sequential backward selection (SBS). The third category is randomized algorithms, which incorporated randomness into their search procedure to escape local minima. The randomized algorithms perform better than the other two types, and are currently the research hot spot, so we focused on the third category in this paper.

### Table 5 The start-of-the-art methods of FS

<table>
<thead>
<tr>
<th>Exponential Algorithms</th>
<th>Sequential Algorithms</th>
<th>Randomized Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exhaustive Research</td>
<td>Sequential Forward Selection</td>
<td>Random Generation</td>
</tr>
<tr>
<td>Branch and Bound</td>
<td>Sequential Backward Selection</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>Beam Search</td>
<td>Plus-l Minus-r Selection</td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td></td>
<td>Bidirectional Search</td>
<td>Ant Colony Optimization</td>
</tr>
<tr>
<td></td>
<td>Sequential Floating Selection</td>
<td>Particle Swarm Optimization</td>
</tr>
</tbody>
</table>

However, when using genetic algorithm (GA) or simulated annealing (SA) algorithm for FS, the results are sensitive to initial values and the computation times are too long [26, 27]. Particle Swarm Optimization (PSO) was a simple metaheuristic with biological inspiration in swarming behavior of birds, fishes, bees, etc. Contrarily to GA, it does not include any genetic operators, which made it simpler than GA and also reduced the number of parameters to adjust. Initially PSO is developed for continuous problems. Later the binary PSO (BPSO) was proposed to solve discrete problems. In this study, the BPSO was chosen since selecting or not selecting was a binary problem.
2.4.1 Encoding

For the FS problem, binary encoding was commonly used. Each potential solution was represented as a particle, with which two properties (position $x$ and velocity $v$) were associated. Suppose $x$ and $v$ of the $i$th particle were given as

\[
x_i = (x_{i1}, x_{i2}, \cdots, x_{iN})
\]

\[
v_i = (v_{i1}, v_{i2}, \cdots, v_{iN})
\]

Here $N$ stands for the dimensions of the problem, and $i$ the particle index. Besides, $x_{ij}$ are binary variables, and they describe the corresponding $j$th feature is to be selected or not. Figure 5 illustrates a 6-dimensional problem. Here the $i$th particle $x_i = [1, 0, 0, 1, 1, 0]$ that shows the corresponding $1^{st}$, $4^{th}$, and $5^{th}$ features are selected.

\[\text{Figure 5 A 6-dimensional problem. Here } x_i = [1, 0, 0, 1, 1, 0] \text{ indicate that the } 1^{st}, 4^{th}, \text{ and } 5^{th} \text{ features are selected.}\]

2.4.2 PSO Algorithm

In each iteration, a fitness function was evaluated for all the particles in the swarm. The velocity of each particle was updated by keeping track of the two best positions. One was the best position a particle had traversed so far and called “\(pBest\)”. The other was the best position that any neighbor of a particle had traversed so far. It is a neighborhood best called “\(nBest\)”. When a particle took the whole population as its neighborhood, the neighborhood at best became the global best and was accordingly called “\(gBest\)”. Hence, a particle’s velocity and position were updated as follows

\[
v = \omega \times v + c_1 \times \text{rand()} \times (pBest - x) + c_2 \times \text{rand()} \times (nBest - x)
\]

(5)

\[x = x + v
\]

(6)

Here \(\text{rand}()\) is a function that returned a random number within \([0, 1]\). These random numbers were updated every time when they occurred. \(\omega\) is called the “inertia weight” which controlled the impact of the previous velocity of the particle on its current one. If \(\omega > 1\) the particle favored exploration over exploitation, else if \(\omega < 1\) the particle gave more importance to the current best positions. The parameters \(c_1\) and \(c_2\) are positive constants, called “acceleration coefficients”. From psychological view, \(c_1\) was the cognitive component that measured the degree of self-confidence of a particle and measured the degree at which it trusts its performance. \(c_2\) is the social component that reflected the capability of the swarm to find better candidate solutions [28].

The population of particles was moved according to (5) and (6), and tended to cluster together from different directions. However, a maximum velocity \(v_{\text{max}}\) was not be exceeded by any particle to keep the search within a meaningful solution space [29]. The PSO algorithm ran through these processes iteratively until the termination criterion was satisfied [30].

2.4.3 BPSO Algorithm

The BPSO initialized the positions and velocities of the particle swarm randomly by

\[
x_{ij} = \begin{cases} 
1, & \text{if } \text{rand()} > 0.5 \\
0, & \text{otherwise}
\end{cases}
\]

(7)

\[
v_{ij} = -v_{\text{max}} + 2 \times \text{rand()} \times v_{\text{max}}
\]

(8)

The positions $x_{ij}$ for each variable was calculated by

\[
x_{ij} = \begin{cases} 
1, & \text{if } S(v_{ij}) > \text{rand()} \\
0, & \text{otherwise}
\end{cases}
\]

(9)

Here $S(.)$ represented the logistic function, and it served as the probability distribution for the position $x_{ij}$.

\[
S(v_{ij}) = \frac{1}{1 + \exp(-v_{ij})}
\]

(10)
The velocities $v_{ij}$ are iteratively updated by the aforementioned formula (5).

Unlike canonical PSO, the particle positions of BPSO were restricted within the Hamming space [31], which was the set of all $2^N$ binary strings of length $N$. Thus, there was no risk of divergence. However, the problem of premature convergence arose. Considering the formula (10), the $S(v_{ij})$ would have approximated 1 if the velocity reached a high value (e.g., $v_{ij} = 10$), resulting in $x_{ij}$ are always 1 so that new solutions were never tried. Hence, the velocity threshold $v_{max}$ was crucial to avoid premature convergence, namely, the value $S(v_{max})$ was smaller than one, which would increase the possibility to generate new solutions. Compared to GA, the velocity clamping was similar to the mutation in GA, with $v_{max}$ playing the role of the mutation rate [32].

2.4.4 Mutation Operator

In order to improve the diversity of BPSO without compromising with the solution quality, we introduced the mutation operator, which could further explore untried areas of the search space by

$$x_i^{ij} \begin{cases} x_i \quad \text{if } \text{rand()} \leq r_{mut} \\ x_i^{ij} \quad \text{otherwise} \end{cases}$$

where $r_{mut}$ stands for the probability of random mutation. After updating the positions properties in (9), each bits of the solution candidates were mutated with a probability $r_{mut}$. It was common to set $r_{mut} = 1/N$, which indicated that it was expected one bits in each solution candidate would be flipped. This BPSO with mutation was abbreviated as MBPSO.

2.5 The whole process

The complete flowchart of this proposed algorithm was detailed in Figure 6 with the corresponding pseudocodes listed as follows:

Step 1  Initialization. Read the dataset and exert normalization. The dataset was stratified divided into 15 folds.
Step 2  The wrapper-based FS. Let $i = 0$.
  a)  Take $i$th fold as the validation set, and the remained folds as the training set. (see Figure 4).
  b)  Use decision tree as the classifier model with the cost as the objective function.
  c)  Train the decision tree by C4.5 algorithm.
  d)  Run MBPSO as the subset search strategy until the in-sample cost on the training set reaches global minimum.
  e)  Submit the validation set to the trained decision tree, and record the out-of-sample cost $C_i$ on the validation set (See formula (2)).
  f)  Let $i = i+1$. If $i < 15$, return to a); otherwise jump to Step 3.
Step 3  Output the average Cost by $C = (C_1 + C_2 + \cdots + C_{15})/15$ as the final 15-fold cross validation cost.

3 Experiments and Results

The computer programs were in-house developed and they ran on HP laptop with Intel core i3 3.2GHz processor and 2G RAM. Matlab 2013a and Sipina 3.3 were served as the software platform. The data set contained 6000 emails, of which 3000 were labeled “spam” and the other 3000 were labeled “nonspam” manually, collected during year 2012.
3.1 Effect of capital-run-length Features

In order to prove the effectiveness of three capital-run-length related features, a two sample Kolmogorov-Smirnov hypothesis test [33] was made to get an objective evaluation. The Kolmogorov-Smirnov hypothesis test quantified a distance between the empirical distribution functions of two samples. The null distribution of this statistic was calculated under the null hypothesis that the samples were drawn from the same distribution. The alternate hypothesis was that they are from different continuous distribution [34]. Table 6 lists the \( p \) value of the hypothesis test, in which the first sample \( x_1 \) was the distribution from spam and the second sample \( x_2 \) was the distribution from nonspam.

Table 6 \( p \) value of hypothesis test (The first sample \( x_1 \) is the distribution from spam, the second sample \( x_2 \) is the distribution from nonspam)

<table>
<thead>
<tr>
<th>Feature</th>
<th>( H_0 (x_1 = x_2) )</th>
<th>( H_1 (x_1 \neq x_2) )</th>
<th>( H_0 (x_1 &lt; x_2) )</th>
<th>( H_1 (x_1 &gt; x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Length</td>
<td>( 3.0882 \times 10^{-192} ) (( &lt; 0.001 ))</td>
<td>1</td>
<td>1.5441 \times 10^{-192} (( &lt; 0.001 ))</td>
<td></td>
</tr>
<tr>
<td>Longest Length</td>
<td>( 1.4574 \times 10^{-203} ) (( &lt; 0.001 ))</td>
<td>1</td>
<td>7.2871 \times 10^{-204} (( &lt; 0.001 ))</td>
<td></td>
</tr>
<tr>
<td>Total Length</td>
<td>( 5.5808 \times 10^{-157} ) (( &lt; 0.001 ))</td>
<td>1</td>
<td>2.7904 \times 10^{-157} (( &lt; 0.001 ))</td>
<td></td>
</tr>
</tbody>
</table>

3.2 Comparison with other Meta-heuristics Algorithms

The parameter \( \alpha \) in formula (1) was set 7, viz., we gave 7 times cost to FP error compared with FN error. The termination tolerance was set 0.001. To demonstrate the improvements on spam detection achieved by the MBPSO approach, it was compared to the other six methods such as Iterated Local Search (ILS) [35, 36], GA [37], restarted SA (RSA) [38], Ant Colony Optimization (ACO) [39], PSO [40], and BPSO [31]. Recall that decision trees were used in combination with C4.5 algorithm in the wrapper methodology. Within that procedure, the 15 cross validation was applied to obtain the average cost of each method. The results are presented in Table 7.

Table 7 Algorithm Performance Comparison (\( \alpha = 7 \))

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Features</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std.</td>
<td>Mean</td>
<td>Std.</td>
</tr>
<tr>
<td>No-FS</td>
<td>57</td>
<td>0</td>
<td>88.42</td>
<td>3.19</td>
</tr>
<tr>
<td>ILS</td>
<td>10</td>
<td>4</td>
<td>88.84</td>
<td>3.64</td>
</tr>
<tr>
<td>GA</td>
<td>11</td>
<td>3</td>
<td>89.77</td>
<td>4.21</td>
</tr>
<tr>
<td>RSA</td>
<td>14</td>
<td>4</td>
<td>85.36</td>
<td>3.90</td>
</tr>
<tr>
<td>ACO</td>
<td>11</td>
<td>3</td>
<td>89.81</td>
<td>3.94</td>
</tr>
<tr>
<td>PSO</td>
<td>9</td>
<td>3</td>
<td>90.05</td>
<td>3.67</td>
</tr>
<tr>
<td>BPSO</td>
<td>8</td>
<td>2</td>
<td>90.64</td>
<td>3.81</td>
</tr>
<tr>
<td>MBPSO</td>
<td>7</td>
<td>2</td>
<td>91.02</td>
<td>4.18</td>
</tr>
</tbody>
</table>

The goal of the next experiment was to compare the algorithms with respect to computation time. Here all algorithms ran 10 times, and the computation time of ILS, GA, RSA, ACO, PSO, BPSO, and MBPSO are shown in Figure 7. The average time for ILS, GA, RSA, ACO, PSO, BPSO, and MBPSO were 8600s, 14500s, 22700s, 15000s, 14200s, 11400s, and 11600s, respectively.
The features chosen by most runs of each algorithm are listed in Table 8. It showed that the selected features by different algorithms overlapped to some extent, and the character feature was seldom selected. Several critical words (like “address”, “internet”, “people”, etc.) with average and total capital-run-lengths were selected commonly.

### Table 8 Selected Features

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Common Words</th>
<th>characters</th>
<th>capital-run-length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILS</td>
<td>address; order; people; email; hp; labs; re; edu</td>
<td>Average; Longest</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>address; internet; email; you; labs; cs; re; edu; table</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>address; our; internet; order; people; email; you; hp; labs; cs; table</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>ACO</td>
<td>address; internet; order; people; you; hp; labs; parts</td>
<td>Longest; Total</td>
<td></td>
</tr>
<tr>
<td>PSO</td>
<td>our; order; people; addresses; email; re; edu</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>BPSO</td>
<td>address; our; internet; people; you; re</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>MBPSO</td>
<td>address; our; internet; order; 2006</td>
<td>Average; Total</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3 Selected Features

3.3 Selected Features

The features chosen by most runs of each algorithm are listed in Table 8. It showed that the selected features by different algorithms overlapped to some extent, and the character feature was seldom selected. Several critical words (like “address”, “internet”, “people”, etc.) with average and total capital-run-lengths were selected commonly.

### Table 8 Selected Features

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Common Words</th>
<th>characters</th>
<th>capital-run-length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILS</td>
<td>address; order; people; email; hp; labs; re; edu</td>
<td>Average; Longest</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>address; internet; email; you; labs; cs; re; edu; table</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>address; our; internet; order; people; email; you; hp; labs; cs; table</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>ACO</td>
<td>address; internet; order; people; you; hp; labs; parts</td>
<td>Longest; Total</td>
<td></td>
</tr>
<tr>
<td>PSO</td>
<td>our; order; people; addresses; email; re; edu</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>BPSO</td>
<td>address; our; internet; people; you; re</td>
<td>Average; Total</td>
<td></td>
</tr>
<tr>
<td>MBPSO</td>
<td>address; our; internet; order; 2006</td>
<td>Average; Total</td>
<td></td>
</tr>
</tbody>
</table>

### 3.4 Comparison with Other FS Algorithms

In this experiment, the proposed MBPSO was compared to the conventional feature selection methods: SFS and SBS. The results are shown in Table 9. Although both SFS and SBS were deterministic, they still needed to run 15 times because of the cross validation technique. Averagely, SFS selected 13 features, and SBS selected 44 features. The sensitivity, specificity, and accuracy of SFS were 88.66, 95.28, and 91.97, respectively. The sensitivity, specificity, and accuracy of SBS were 88.42, 95.02, and 91.72, respectively.

### Table 9 Comparison with Other FS Algorithms (α = 7)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Features</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Accuracy</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std.</td>
<td>Mean</td>
<td>Std.</td>
<td>Mean</td>
</tr>
<tr>
<td>SFS</td>
<td>13</td>
<td>1</td>
<td>88.66</td>
<td>3.99</td>
<td>95.28</td>
</tr>
<tr>
<td>SBS</td>
<td>44</td>
<td>2</td>
<td>88.42</td>
<td>3.84</td>
<td>95.02</td>
</tr>
<tr>
<td>MBPSO</td>
<td>7</td>
<td>2</td>
<td>91.02</td>
<td>4.18</td>
<td>97.51</td>
</tr>
</tbody>
</table>

### 3.5 Comparison with other Spam Detection Algorithms

In this experiment, the proposed method was compared to the other spam detection algorithms such as ANN and SVM. The ANN structure was chosen as 7-5-1. The transfer function was chosen as the sigmoid function. The number of input neurons was equal to the number of selected features by the proposed method (decision tree trained by MBPSO). The weights/biases of ANN were trained by Levenberg-Marquardt method. The kernel of SVM was chosen as Gaussian Radial Basis function with $\sigma = 1$. The parameters of SVM were trained by quadratic programming method [41].

The results are shown in Table 10. It showed that the sensitivity, specificity, and accuracy of ANN were
91.08, 97.68, and 94.38, respectively. The sensitivity, specificity, and accuracy of SVM were 91.14, 97.70, and 94.42, respectively.

<table>
<thead>
<tr>
<th>Classifier Model</th>
<th>Training Algorithm</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean Std.</td>
<td>Mean Std.</td>
<td>Mean Std.</td>
</tr>
<tr>
<td>ANN</td>
<td>Levenberg-Marquardt</td>
<td>91.08 4.49</td>
<td>97.68 1.81</td>
<td>94.38 3.26</td>
</tr>
<tr>
<td>SVM</td>
<td>Quadratic Programming</td>
<td>91.14 3.14</td>
<td>97.70 1.98</td>
<td>94.42 3.35</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>MBPSO</td>
<td>91.02 4.18</td>
<td>97.51 2.14</td>
<td>94.27 3.31</td>
</tr>
</tbody>
</table>

### 3.6 Wrappers vs Filters

In this experiment, we compared the proposed wrapper-based FS with the filter-based FS strategy, as described in section 2.2. For the filter-based FS, the measurement was chosen as Relief-F that was a simple procedure to estimate the quality of features with strong dependencies between features [42]. In practice, Relief-F was commonly applied in data preprocessing as a FS method [43]. The threshold was set as 0.002. The caveat was that the weight parameter $\alpha$ should not be set as 7 since that value was optimized for the best classification for wrapper-based FS. Therefore, it made the comparison fair to let $\alpha=1$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Features</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Accuracy</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean Std.</td>
<td>Mean Std.</td>
<td>Mean Std.</td>
<td>Mean Std.</td>
<td></td>
</tr>
<tr>
<td>filter-based FS</td>
<td>15 2</td>
<td>91.38 4.35</td>
<td>91.16 3.85</td>
<td>91.27 3.64</td>
<td>1100s</td>
</tr>
<tr>
<td>wrapper-based FS</td>
<td>7 3</td>
<td>94.46 3.28</td>
<td>95.63 3.14</td>
<td>95.05 3.19</td>
<td>11600s</td>
</tr>
</tbody>
</table>

### 3.7 Optimal Weight

In aforementioned study, weight parameter $\alpha$ was set as 7, in order to give 7 times cost to FP than FN. The question that arose was why $\alpha$ was chosen to be 7. To address the question, the value was changed from 1 to 15 in step 1 in order to investigate its effect as shown in Figure 8. For the simplified condition when $\alpha=1$, the two types of error were treated equally, so the sensitivity and specificity were nearly equal.

![Figure 8 Effect of weight parameter on (a) The Cost; and (b) Classification Performance.](image-url)
4 Discussions and Analysis

The results from Table 6 showed there were 6 $p$ values less than the significant level (set as 0.001). It demonstrated that the average capital-run-length, the longest capital-run-length, and the total capital-run-length from the nonspam were distinctly less than those from their counterparts (spam). The results validated the conclusion from the spam database in UCI machine learning repository collected in 1999. It also demonstrated that the common words selected in 1999 were still moderately effective in 2012, so the basic characteristics of spam have not changed even though 13 years have passed.

The results from Table 7 showed that the MBPSO obtained the least features as 7 on average. Moreover, the MBPSO performed better than the other six algorithms with the sensitivity as 91.02%, specificity as 97.51%, and accuracy as 94.27%. Using the cost matrix, the specificity of our spam detection task was significantly improved. This was extremely important since the FP error (labeling nonspam as spam) cause more serious trouble than the FN error (labeling spam as nonspam). We can also find that both MBPSO and BPSO were better than PSO, due to their mechanisms that initialized and updated positions were consistent with the binary properties, and PSO does not take into account the binary properties of the positions. Furthermore, the MBPSO embedded the mutation operator that enhanced the diversity of the algorithm, so it obtained better results than BPSO. The classification performance of ILS was not good, which may have arisen from the searching sequence is confined within the attraction basin.

The results from Figure 7 showed that the RSA used the most time as 22700s. The reason lies that it restarted the annealing mechanism that let the temperature move back repeatedly once the algorithm was found to be trapped into local optimal points. ILS cost the least time at were 8600s since it started from a local minimum other than a random point, which could save considerable time. The GA, ACO, and PSO cost nearly the same time. BPSO costs less time because its binary position property made it converge to the global optima in fewer epochs than PSO. MBPSO ranked third, and it was on average 200s slower than BPSO.

The results in Table 8 demonstrated that the common words and capital-run-length features contributed significantly to the classification. However, the preparatory characters feature had little effect. Only RSA selected the character of '{', and ACO selected the character of 'S'. This result dropped a subtle hint about the concept drifting problem [44] that the spam itself was evolving, so the six characters contained in 1999’s dataset (Table 1) may not work properly in the current spam detection problem. From the popular view on the evolving nature of spam, it was suggested to add more “common words” that covered the evolution of spams content, and employed other statistical features to improve the classifier performance. The concept drift problem cannot be avoided for complex phenomena especially those not governed by fixed laws of nature [45]. All processes that arose from human activity, such as socioeconomic processes and biological processes were likely to experience concept drift. Therefore, periodic retraining was necessary. For this spam detection study, it was necessary to update the classifier by the newest samples, or by enforcing an ensemble of classifiers.

The results from Table 9 show that SFS selected substantially fewer features than SBS. The reason was that SFS started from an empty set, added features sequentially; SBS worked in the opposite direction, started from the full set and removed features sequentially. We also see that SFS performed slightly better than SBS did. The explanation may fall in that SFS performed best especially when the optimal feature set was small and SBS performed best when the optimal feature set was large. In this study, the optimal feature set found by MBPSO was only 7, so SFS was likely to perform better than SBS. Another reason that limited SBS was that SBS cannot reevaluate a feature after discarding it. However, SFS and SBS were both easily trapped into local optima.

Results in Table 10 show ANN and SVM performed somewhat better than the proposed decision tree. The result was consistent with what was expected. The ANN and SVM contained nonlinear components that can map the original feature space to high-dimensional complicated space. This nonlinear mapping improved the classification performance. Nevertheless, the decision tree was chosen as the model, because it can give explicit meaning (Table 8), and easily be transformed to “if-then” rules. For researchers, the decision tree can show what the most important key terms are and how they influence the classification results. Although ANN and SVM cannot deduce physical meanings, their excellent classification performances are attracting in commercial services.

Table 11 indicates that the wrapper-based FS was superior to the filter-based FS overall except the computation time. The filter-based FS cost only about 1100s that is nearly 10.5 times faster than that of wrapper-based FS. The slow execution of wrapper-based FS was worthy considering that it only needs to train once and its classification performance was much better than that of filter-based FS.

Figure 8(a) shows the relationship between the cost and the weight $\alpha$. It was clear that the global minimal point was located at $\alpha = 7$. Therefore, setting $\alpha = 7$ can obtain the minimal cost. To investigate the mechanisms behind it, recall that as the weight $\alpha$ increases, FP error is given more importance, which would lead to the increase of specificity curve at the expense of the decrease of sensitivity and the accuracy curves, where there is exactly a local maximal point at $\alpha = 7$ located. The aforementioned was validated in Figure 8(b). However, the weight tuning work in this study was a coarse search since $\alpha$ was increased by one at each step. Several
improvements will be done in the future experiment, such as reducing step size to get high-precision weight value, and using gradient-based or heuristic optimization algorithm to reduce computation time. Another challenge was in analyzing the robustness of weight \( \alpha \), viz., how the classifier performs if \( \alpha \) is not well-tuned. The weight \( \alpha \) based cost matrix, see formula (1), was effective in distinguishing the two types of errors (FP and FN). It was commonly used in medical applications, because the positive cases are of more important than the negative ones. In this study, it was assumed that the spam detection problem is similar to medical diagnosis, and the experimental results meet the expectations of the experiment correctly.

It is necessary to understand the shortcomings of the proposed method stems from the randomness of MBPSO. It initialized the whole population randomly so the results are not always the global optimal points. The results were obtained by K-fold cross validation, which implied that the classifier may not be trained perfectly for some folds. This conveyed important information that the proposed method had the untapped potential to improve the classification performance, which occurred once all trainings on different folds obtained the global optima. Computation time was another shortcoming that restricted the widespread use of MBPSO. In future research, it will be beneficial to employ other tentative acceleration strategies to reduce the computation time. This may be achieved by introducing adaptive parameter, enhancing the diversity of the population, and etc.

5 Conclusions and Further Study

The main contribution and technical innovation of the paper falls within the following four points: 1) Made a Kolmogorov-Smirnov hypothesis test on capital-run-length related features and having the \( p \) values less than 0.001. 2) Used wrap-based feature selection method that can achieve high classification accuracy meanwhile select important features. 3) Used C4.5 Decision tree as the classifier of the wrapper, and use binary PSO with mutation operator (MBPSO) as the search strategy of the wrapper. 4) Proved that the feature selection results by MBPSO are better than those of ILS, GA, RSA, ACO, PSO, and BPSO. 5) Proved that the capital-run-length related features are effective. 6) Proved that wrappers are more effective than filters with regard to classification performance.

The proposed MBPSO based wrappers can be applied to solve other classification problems, such as web-page classification [46], text classification [47], image classification [48], etc.

Conflict of Interests

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Acknowledgments

We would like to express our gratitude to Nanjing Normal University Research Foundation for Talented Scholars (No. 2013119XG00061) and National Natural Science Foundation of China (No. 610011024).

References