Hindawi
Complexity
Volume 2021, Article ID 9384318, 10 pages
https://doi.org/10.1155/2021/9384318

Research Article

From Linear Programming Approach to Metaheuristic Approach: Scaling Techniques

Elsayed Badr,[1,2] **Mustafa Abdul Salam**,[3] **Sultan Almotairi**,[4] **and Hagar Ahmed**[1]

[1]*Scientific Computing Department, Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt*
[2]*Higher Technological Institute, 10th of Ramadan City, Egypt*
[3]*Artificial Intelligence Department, Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt*
[4]*Department of Natural and Applied Sciences, Community College, Majmaah University, Al-Majmaah 11952, Saudi Arabia*

Correspondence should be addressed to Elsayed Badr; badrgraph@gmail.com and Sultan Almotairi; almotairi@mu.edu.sa

Received 3 April 2020; Revised 11 September 2020; Accepted 31 January 2021; Published 11 February 2021

Academic Editor: Roberto Natella

The objective of this work is to propose ten efficient scaling techniques for the Wisconsin Diagnosis Breast Cancer (WDBC) dataset using the support vector machine (SVM). These scaling techniques are efficient for the linear programming approach. SVM with proposed scaling techniques was applied on the WDBC dataset. The scaling techniques are, namely, arithmetic mean, de Buchet for three cases ($p = 1, 2, \text{and} \infty$), equilibration, geometric mean, IBM MPSX, and L_p-norm for three cases ($p = 1, 2, \text{and} \infty$). The experimental results show that the equilibration scaling technique overcomes the benchmark normalization scaling technique used in many commercial solvers. Finally, the experimental results also show the effectiveness of the grid search technique which gets the optimal parameters (C and gamma) for the SVM classifier.

1. Introduction

Scaling techniques play an important role in the convergence speed of machine learning algorithms, specially in classification and regression tasks. Using an efficient scaling technique makes training of algorithms faster. There is an integrative relationship between the linear programming approach and metaheuristic approach according to scaling techniques. A scaling technique is defined as a mathematical formula which makes these elements have similar magnitudes. In linear programming, the scaling techniques are applied on the objective function, the coefficient matrix of the inequalities, and the coefficient of constants. On the contrary, in the metaheuristic approach, the scaling techniques are applied on the matrix in which its rows represent the observations and its columns are the attributes of the dataset.

A dataset contains mostly nonzero elements which are of different values. Such representation is called a bad representation for this matrix. Scaling techniques can be used to handle this issue. Scaling techniques are used before applying the classifier in order to improve the classification accuracy on the dataset.

The comparison among the following scaling techniques, Curtis and Reid [1] scaling technique, arithmetic mean scaling technique, Wolfe [2] scaling technique, geometric mean scaling technique, and equilibration scaling technique, was proposed by Tomlin [3] on 6 test linear programming problems of different sizes. Another study was proposed by Larsson [4] by proposing and comparing entropy, de Buchet scaling technique [5], and L_p-norm scaling technique [6] on one-hundred thirty-five randomly generated problems of different dimensions. He deduced that the entropy scaling method outperforms the other scaling techniques. Elble and Sahinidis [7] proposed new experimental results from the comparison among the following scaling techniques: IBM MPSX, entropy, arithmetic mean, binormalization, geometric mean, L_p-norm, equilibration, and de Buchet on benchmark problems from Netlib. Scaling and solution times, the number of iterations for the solution, and the maximum condition number were the evaluation metrics for their study. They deduced that the equilibration method

outperformed other techniques. Ploskas and Samaras [8] introduced experimental results for three algorithms: MATLAB's revised simplex method, exterior point simplex method, and interior point algorithm using geometric mean scaling technique, equilibration scaling technique, and arithmetic mean scaling technique. They deduced that the equilibration scaling technique overcame the other techniques and that the effectiveness of scaling is important to both the interior point algorithm and revised simplex method; on the contrary, the exterior point simplex method is scaling invariant [9]. Ploskas and Samaras [10] proposed new experimental results comparing arithmetic mean, de Buchet for three cases ($p = 1, 2, \text{and} \infty$), equilibration, geometric mean, IBM MPSX, and L_p-norm for three cases ($p = 1, 2, \text{and} \infty$). Ploskas and Samaras [10] deduced that arithmetic mean, equilibration, and geometric mean overcame the other scaling techniques according to the execution time. In [11], Ploskas and Samaras in the chapter "Scaling Techniques" present a complete list of the scaling techniques plus illustrative examples. Ploskas and Samaras [12] state clearly that MATLAB's GPU environment (in 2014) did not offer sparse utilities. Also, they were the first to present a GPU-based simplex implementation that showed speedups in benchmark instances. In their implementation, they used the most efficient scaling techniques.

In this work, ten efficient scaling techniques were proposed for the Wisconsin Diagnosis Breast Cancer (WDBC) dataset using the support vector machine (SVM). The SVM with proposed scaling techniques was applied on the WDBC dataset. The experimental results show that the equilibration scaling technique overcomes the benchmark normalization scaling technique.

The rest of this paper is organized as follows. The support vector machine classifier is described in section 2. In section 3, detailed descriptions of new scaling techniques are presented. The experimental design which has data description, experimental setup, measure for performance evaluation, and grid search method is introduced in section 4. In section 5, experimental results and discussions are discussed. In section 6, conclusions and future works are introduced.

2. Support Vector Machine Classifier

The support vector machine (SVM) is considered as a machine learning model originally developed by Vapnik [13, 14]. The SVM is based on the Vapnik–Chervonenkis (VC) theory and structural risk minimization (SRM) principle [13, 15]. The main objective of the SVM is finding a hyperplane in an N-dimensional space (N: the number of features) that distinctly classifies the data points, as shown in Figure 1. The convex quadratic programming is used for the SVM in order to avoid the local minima [13, 16].

In the linear classification, the hyperplane is placed in the largest distance between two vectors. In case of the nonlinear classification, it is mapped to the linear classification problem in a high-dimensional space [17], as shown in Figure 2.

Let us consider a binary classification task: suppose that $(x_1, y_1), \ldots, (x_n, y_n)$: $x_i \in R^d$ and $y_i \in (-1, 1)$ is a labeled

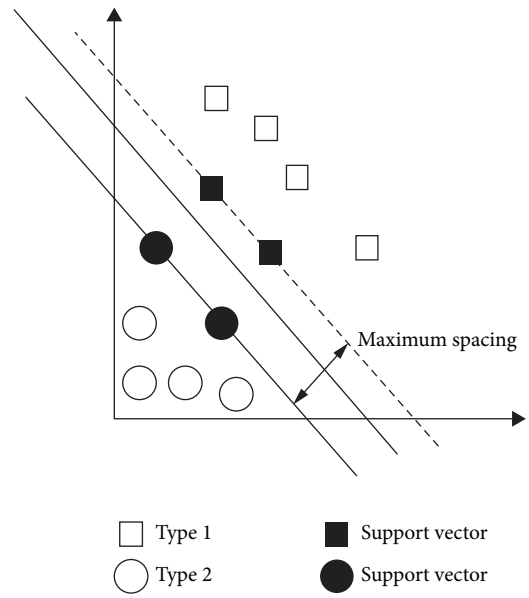

Figure 1: The SVM hyperplane in the linear classification.

training dataset such that x_i is a representation of the feature vector and y_i is the class label (negative or positive) of a training compound i. The optimal hyperplane can then be defined as follows:

$$wx^T + b = 0. \tag{1}$$

Such that w is the weight vector, x is the input feature vector, and b is the bias. w and b would satisfy both inequality (2) and inequality (3) for all elements of the training set:

$$wx_i^T + b \geq 1, \quad \text{if } y_i = 1, \tag{2}$$

$$wx_i^T + b \leq 1, \quad \text{if } y_i = -1. \tag{3}$$

The aim of training an SVM classifier is to determine w and b so that the hyperplane separates the data and maximizes the margin $1/\|w\|^2$. Vectors, x_i for which $|y_i|(wx_i^T + b) = 1$, will be termed the support vector.

There are cases in which we can linearly separate between the two classes, and there are other cases in which we cannot linearly separate between the two classes. We can overcome this problem by transforming the original input space into some higher-dimensional feature space where the two classes can be linearly separable. An alternative use for the SVM is the kernel method, which enables us to model higher-dimensional, nonlinear models [18]. In a nonlinear problem, a kernel function could be used to add additional dimensions to the raw data and thus making it a linear problem in the resulting higher-dimensional space. On the contrary, the kernel functions could help do certain calculations faster which would need computations in the high-dimensional space. There are many kernel functions, for example, but not limited to, the linear kernel and the Gaussian kernel, which are defined as shown in the following equations:

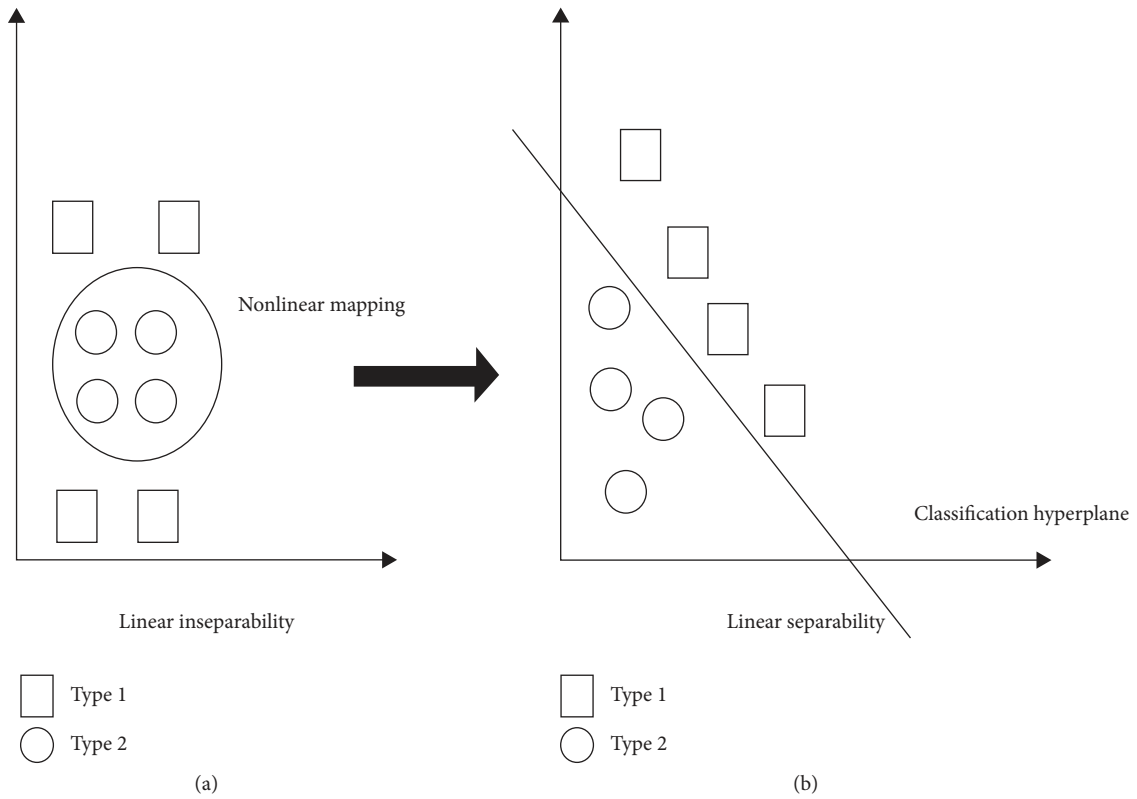

Figure 2: Mapping from nonlinear to linear classification.

$$K\left(x_i, x_j\right) = \left(1 + x_i^T x_j\right)^p, \tag{4}$$

$$K\left(x_i, x_j\right) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \tag{5}$$

where p is the order of polynomial and γ is the predefined parameter controlling the width of the Gaussian kernel. The SVM classification accuracy is improved by the proper model parameters' setting [19]. It is important to choose the parameters in advance. These parameters are C, (γ or p), and kernel function.

C parameter is considered a regularization or generalization parameter. It governs the trade-off between having a minimum training error and minimizing the weight's norm. C parameter tuning is a very important step in optimizing of the SVM. The parameter C imposes an upper bound on the weight's norm, which implies that there are multiple hypothesis classes indexed by C. Increasing the C parameter leads to increasing the complexity of the hypothesis class. If we increase C slightly, we can still form all of the linear models [19]. Determining how to set C is not very well developed, so most researchers use cross-validation.

3. Scaling Techniques

Here, we introduce the mathematical notations of ten scaling techniques in addition to the normalization scaling techniques with ranges [0, 1)] and [−1, 1]. First of all, we introduce the following mathematical preliminaries, as shown in Table 1.

The scaled matrix is expressed as RAS, such that $R = \text{diag}(r_1, \ldots, r_m)$ and $S = \text{diag}(s_1, \ldots, s_n)$. All scaling techniques proposed in this section first apply row scaling and after that column scaling. Then, the matrix after full scaling (row and column) is given by the following:

$$\begin{aligned} A^R &= RA, \\ A^{\text{RS}} &= A^R S. \end{aligned} \tag{6}$$

3.1. Arithmetic Scaling Technique [11]. First, equation (7) represents the rows' scaling such that each row (instance) is divided by the arithmetic mean of the absolute value of the nonzero elements in that row (instance):

$$r_i = \frac{n_i}{\sum_{j \in N_i} \left| a_{ij} \right|} : a_{ij} \neq 0. \tag{7}$$

Second, equation (8) represents the columns' scaling such that each column (attribute) is divided by the arithmetic mean of the absolute value of the nonzero elements in that column (attribute):

$$s_j = \frac{m_j}{\sum_{i \in M_j} \left| a_{ij}^R \right| \left| a_{ij}^R \right|} \neq 0. \tag{8}$$

3.2. de Buchet Scaling Technique [4]. Equation (9) formulates the de Buchet scaling method which is based on the relative divergence:

Table 1: Mathematical preliminaries for scaling techniques.

Symbol	Description
$A(a_{ij})$:	$m \times n$ matrix (with m rows (observations) and n columns (attributes))
r_1:	The scaling agent of row i
s_j:	The scaling agent of column j
R:	Diagonal matrix, such that $R = \text{diag}(r_1, \ldots, r_m)$
S:	Diagonal matrix, such that $S = \text{diag}(r_1, \ldots, r_n)$
N_i:	$N_i = \{j\colon A_{ij} \neq 0\}$, such that $1 \leq i \leq m$ and $M_j = \{i\colon A_{ij} \neq 0\}$, such that $1 \leq j \leq n$
M_j: n_i:	The number of elements for the set N_i
m_j:	The number of elements for the set M_j
$A^R(a_{ij}^R)$:	The scaled matrix by the row R scaling agent
$A^{\text{RS}}(a_{ij}^{\text{RS}})$:	The final scaled matrix

$$\min_{r,s>0} \left(\sum_{i,j \in \overline{Z}} \left\{ \frac{a_{ij} r_i s_j + 1}{a_{ij} r_i s_j} \right\}^p \right)^{1/p}, \tag{9}$$

where the number of the nonzero elements of A is dented by $\overline{Z}$ and the parameter p is a positive integer. Here, there are the following three cases:

Case $p = 1$: in this case, equation (9) approaches to the following equation:

$$\min_{r,s>0} \sum_{i,j \in \overline{Z}} \left\{ \frac{a_{ij} r_i s_j + 1}{a_{ij} r_i s_j} \right\}. \tag{10}$$

Equation (11) represents the row scaling factor of the matrix A:

$$r_i = \left\{ \frac{\left(\sum_{j \in N_i} 1/\left| a_{ij} \right| \right)}{\left(\sum_{j \in N_i} \left| a_{ij} \right| \right)} \right\}^{1/2}. \tag{11}$$

Equation (12) represents the column scaling factor of the scaled matrix A by r_i:

$$s_j = \left\{ \frac{\left(\sum_{i \in M_j} 1/\left| a_{ij}^R \right| \right)}{\left(\sum_{i \in M_j} \left| a_{ij}^R \right| \right)} \right\}^{1/2}. \tag{12}$$

Case $p = 2$: in this case, equation (9) approaches to the following equation:

$$\min_{r,s>0} \left(\sum_{i,j \in \overline{Z}} \left\{ \frac{a_{ij} r_i s_j + 1}{a_{ij} r_i s_j} \right\}^2 \right)^{1/2}. \tag{13}$$

Equation (14) represents the row scaling factor of the matrix A:

$$r_i = \left\{ \frac{\left(\sum_{j \in N_i} \left(1/\left| a_{ij} \right| \right)^2 \right)}{\left(\sum_{j \in N_i} \left(\left| a_{ij} \right| \right)^2 \right)} \right\}^{1/4}. \tag{14}$$

Equation (15) represents the column scaling factor of the scaled matrix A by r_i:

$$s_j = \left\{ \frac{\left(\sum_{i \in M_j} \left(1/\left| a_{ij}^R \right| \right)^2 \right)}{\left(\sum_{i \in M_j} \left(\left| a_{ij}^R \right| \right)^2 \right)} \right\}^{1/4}. \tag{15}$$

Case $p = \infty$: in this case, equation (9) approaches to the following equation:

$$\min_{r,s>0} \max_{i,j \in \overline{Z}} \left| \log\left(a_{ij} r_i s_j \right) \right|. \tag{16}$$

Equation (17) represents the row scaling factor of the matrix A:

$$r_i = \frac{1}{\left\{ \left(\max_{j \in N_i} \left| a_{ij} \right| \right) \left(\min_{j \in N_i} \left| a_{ij} \right| \right) \right\}^{1/2}}. \tag{17}$$

Equation (18) represents the column scaling factor of the scaled matrix A by r_i:

$$s_j = \frac{1}{\left\{ \left(\max_{i \in M_j} \left| a_{ij}^R \right| \right) \left(\min_{i \in M_j} \left| a_{ij}^R \right| \right) \right\}^{1/2}}. \tag{18}$$

The last case of the de Buchet ($p = \infty$) scaling technique is equivalent to the geometric mean scaling method that will be introduced later.

3.3. Equilibration Scaling Technique [11]. The largest element in the absolute value is the corner stone for this scaling method. Each row of the matrix A is divided by the largest element in the absolute value in that row. Then, each column of the scaled matrix A by the row factor is divided by the largest element in the absolute value in that column. The range of the final scaled matrix A is (–1, 1).

3.4. Geometric Mean Scaling Technique [11]. First, equation (19) represents the rows' scaling such that each row (instance) is divided by the geometric mean of the absolute value of the nonzero elements in that row (instance):

$$r_i = \left(\max_{j \in N_i} \left| a_{ij} \right| \min_{j \in N_i} \left| a_{ij} \right| \right)^{-1/2}. \quad (19)$$

Second, equation (20) represents the columns' scaling such that each column (attribute) is divided by the geometric mean of the absolute value of the nonzero elements in that column (attribute):

$$s_j = \left(\max_{j \in M_j} \left| a_{ij}^R \right| \min_{j \in M_j} \left| a_{ij}^R \right| \right)^{-1/2}. \quad (20)$$

3.5. IBM MPSX Scaling Technique [11]. The IBM MPSX scaling method is a combination between the geometric mean and the equilibration scaling methods. First, the geometric mean is performed four times or until the relation (21) holds true:

$$\frac{1}{|\overline{Z}|} \left(\frac{\sum_{i,j \in \overline{Z}} \left(a_{ij} \right)^2 - \left(\sum_{i,j \in \overline{Z}} \left(a_{ij} \right)^2 \right)^2}{|\overline{Z}|} \right) < \varepsilon, \quad (21)$$

where the number of the nonzero elements of A is denoted by $\overline{Z}$ and the parameter $\varepsilon$is a positive integer (which is often $\varepsilon < 10$). Then, the equilibration scaling method is applied. The IBM MPSX scaling method was introduced by Benichou et al. [20].

3.6. L_p-Norm Scaling Technique [11]. Equation (22) formulates the L_p-norm scaling method:

$$\min_{r,s>0} \sum_{i,j \in \overline{Z}} \left(\left| \log \left(a_{ij} r_i s_j \right) \right|^p \right)^{1/p}, \quad (22)$$

where the number of the nonzero elements of A is dented by $\overline{Z}$. Here, there are the following three cases:

Case $p = 1$: in this case, equation (22) approaches to the following equation:

$$\min_{r,s>0} \sum_{i,j \in \overline{Z}} \left| \log \left(a_{ij} r_i s_j \right) \right|. \quad (23)$$

Equation (24) represents the row scaling factor of the matrix A:

$$r_i = \frac{1}{\text{median} \left\{ a_{ij} | j \in N_i \right\}}. \quad (24)$$

Similarly, equation (25) represents the column scaling factor of the matrix A:

$$s_j = \frac{1}{\text{median} \left\{ a_{ij}^R | i \in M_j \right\}}. \quad (25)$$

Case $p = 2$: in this case, equation (22) approaches to the following equation:

$$\min_{r,s>0} \sum_{i,j \in \overline{Z}} \left(\left| \log \left(a_{ij} r_i s_j \right) \right|^2 \right)^{1/2}. \quad (26)$$

Equation (27) represents the row scaling factor of the matrix A:

$$r_i = \frac{1}{\prod_{j \in N_i} \left(a_{ij} \right)^{1/n_i}}. \quad (27)$$

Similarly, equation (28) represents the column scaling factor of the matrix A:

$$s_j = \frac{1}{\prod_{i \in M_j} \left(a_{ij}^R \right)^{1/m_j}}. \quad (28)$$

Case $p = \infty$, the last case of the L_p-norm ($p = \infty$) scaling technique is equivalent to the geometric mean scaling method.

3.7. Normalization Scaling Technique [−1, 1] [21]. Equation (29) is used for the normalization scaling method with range [−1, 1] such that a, a', $\max_k$, and $\min_k$ are the original value, the scaled value, the maximum value, and the minimum value of feature k, respectively:

$$a' = 2 \left(\frac{a - \min_k}{\max_k - \min_k} \right) - 1. \quad (29)$$

The normalization scaling method avoids the numerical difficulties during the calculation.

3.8. Normalization Scaling Technique [0, 1] [21]. Another normalization scaling technique is formulated from the updated equation (29) as follows:

$$a' = \frac{a - \min_k}{\max_k - \min_k}. \quad (30)$$

4. Experimental Design

In this section, we introduce data description, measure for performance evaluation, and the grid search method.

4.1. Data Description. In this work, we have run the proposed model on the Wisconsin Diagnosis Breast Cancer (WDBC) dataset that is available on the UCI Machine Learning Repository [22]. The dataset consists of 569 instances divided into two classes. The two classes malignant and benign have 357 and 212 cases, respectively. Every observation in the database has thirty-three attributes. These thirty-three attributes differ between benign and malignant samples.

The MATLAB platform is used to implement the SVM diagnostic system. On the contrary, we use LIBSVM that is developed by Chang and Lin [23]. Table 2 describes the computing environment.

Table 2: Description of the computing environment.

CPU	Intel (R) core (TM) i5-7200 U CPU@ 2.70 GHz
RAM size	4 GB RAM
MATLAB version	R2018a (9.4.0.813654)

Salzberg [24] introduced the k-fold CV which is used to guarantee the valid results. In this paper, set k as 10, i.e., the data consists of 10 subsets. The most commonly used (default) value is $k = 10$ in k-fold CV, which is often a good choice [25]. Each time, one of the 10 subsets is utilized as the test set and the remaining 9 subsets are utilized as a training set.

4.2. Measure for Performance Evaluation. In order to test the performance of the SVM model, we use accuracy (ACC). Table 3 shows the confusion matrix. TP, FN, TN, and FP are the number of true positives, the number of false negatives, the number of true negatives, and the number of false positives, respectively. According to the confusion matrix, the total classification accuracy (ACC) is defined as follows: $\text{ACC} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN}) \times 100\%$.

4.3. Grid Search Method. In order to test the performance of the SVM system, we use the grid search method. The grid search method is used to determine the optimal parameters C and γ. Figure 3 shows the flowchart of the SVM training using the grid search. We utilize the searching space of C and γ as follows: $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\{2^{-15}, 2^{-13}, \ldots, 2^{1}\}$, respectively.

5. Experimental Results and Discussion

Here, the experimental results were applied, and an attempt is made to prove the validation of the proposed scaling techniques. The experiments were done on the WBCD dataset using the SVM to estimate the efficiency of the proposed scaling techniques for the breast cancer.

Table 4 shows the effectiveness of the grid search method which gets the best parameters C and γ for the SVM. The accuracy of normalization scaling techniques (S1) is better than that without the scaling technique (S0). Also, using scaling techniques (S1) speeds up the search and achieves dramatic decrease of CPU time. So, this result shows the effectiveness of the grid search method using the scaling technique (S1).

Tables 5 and 6 show the average classification accuracy rates and CPU time of the SVM with four scaling techniques. These techniques are normalization between (−1, 1) (S2), the equilibration scaling (S3), the geometric mean scaling (S4), and the arithmetic mean scaling (S5). One can notice easily that S3 achieved the best accuracy with 98.95% outperforming the compared scaling techniques. S3 also achieved lowest CPU time with about 10.2 seconds.

Tables 7 and 8 show the average accuracy rates of the SVM with the de Buchet scaling technique with $p = 1$ (S6), de Buchet scaling technique with $p = 2$ (S7), and the IBM MPSX scaling technique (S8). It is clear that both S6 and S8 scaling techniques have the same accuracy of 98.59% which is better than the accuracy of S7 scaling technique.

Table 3: confusion matrix for breast cancer diagnosis.

		Predictive positive	Predictive negative
Actual	Positive	(TP)	(FN)
	Negative	(FP)	(TN)

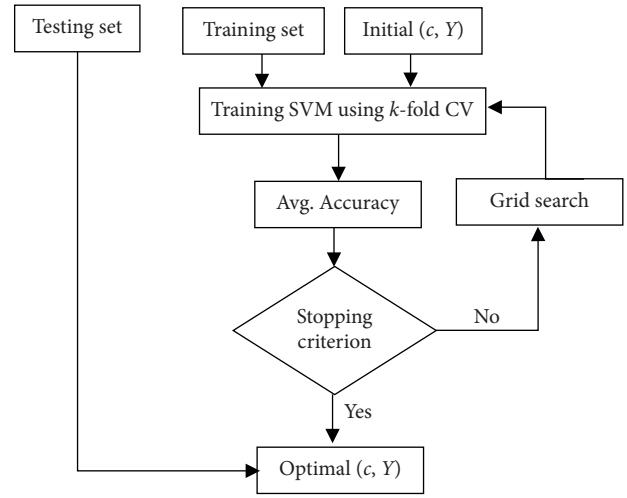

Figure 3: SVM using grid search.

Table 9 shows the average classification accuracy rates of the SVM with the L_p-norm scaling technique with $p = 1$ (S9) and L_p-norm scaling technique with $p = 2$ (S10). One can notice that S9 achieved 98.25 accuracy outperforming S10 scaling technique. But, S10 CPU time is slightly lower than S9.

Tables 10 and 11 summarize the accuracy and CPU time of all compared scaling techniques. The equilibration scaling technique (S3) achieved the best accuracy and the lowest CPU time outperforming all compared scaling techniques. Figures 4 and 5 show the superiority of S3 according to the accuracy rate and CPU time, respectively.

Figure 6 shows the superiority of equilibration scaling technique (S3) and achieved best accuracy in all 10-fold cross-validation.

6. Conclusions

In this work, we proposed ten efficient scaling techniques for the Wisconsin Diagnosis Breast Cancer (WDBC) dataset using the support vector machine (SVM). These scaling techniques can enhance classification accuracy, reduce CPU time, and make training faster. Also, grid search is used to select best free parameters of the SVM (C, gamma). Simulation results showed that equilibration scaling technique (S3) achieved best accuracy with 98.95% outperforming all compared scaling techniques. S3 also achieved lowest CPU time with about 10.2 seconds. Eight efficient scaling techniques outperformed the two benchmark scaling techniques according to the accuracy rate. These techniques are S3, S4,

Table 4: Accuracy of the WBCD database using the SVM with C and γ which were calculated by the grid search technique (without scaling, normalization scaling [0, 1], and normalization scaling [−1, 1]).

Fold	Without scaling (S0)			Normalization scaling (0, 1) (S1)		
	C	γ	Accuracy %	C	γ	Accuracy %
1	2^{3}	2^{-13}	94.76	2^{13}	2^{-7}	100
2	2^{7}	2^{-15}	91.59	2^{15}	2^{-9}	98.25
3	2^{15}	2^{-13}	100	2^{15}	2^{1}	92.98
4	2^{5}	2^{-13}	97.18	2^{15}	2^{-1}	94.74
5	2^{1}	2^{-11}	96.23	2^{15}	2^{-1}	94.74
6	2^{-1}	2^{-9}	91.29	2^{15}	2^{1}	96.49
7	2^{11}	2^{-15}	97.59	2^{15}	2^{-3}	98.25
8	2^{9}	2^{-15}	98.60	2^{15}	2^{-13}	96.49
9	2^{9}	2^{-15}	97.59	2^{15}	2^{1}	94.74
10	2^{15}	2^{-9}	96.23	2^{15}	2^{1}	98.25
Avg.	6877.9	0.0005	96.10	30310.4	0.91	96.49
CPU time		52.62167			14.558410	

Table 5: Accuracy of the WBCD database using the SVM with C and γ which were calculated by the grid search technique (normalization between [−1, 1] and equilibration scaling).

Fold	Normalization between (−1, 1) (S2)			Equilibration scaling (S3)		
	C	γ	Accuracy%	C	γ	Accuracy %
1	2^{11}	2^{1}	94.64	2^{5}	2^{-1}	100.00
2	2^{15}	2^{1}	92.98	2^{3}	2^{1}	98.25
3	2^{13}	2^{1}	100	2^{5}	2^{-1}	100.00
4	2^{13}	2^{1}	98.25	2^{15}	2^{1}	98.25
5	2^{15}	2^{1}	96.49	2^{1}	2^{-1}	100.00
6	2^{15}	2^{-1}	96.49	2^{9}	2^{-1}	98.25
7	2^{13}	2^{1}	100	2^{15}	2^{1}	100.00
8	2^{15}	2^{1}	96.49	2^{15}	2^{1}	100.00
9	2^{13}	2^{1}	94.74	2^{3}	2^{1}	94.74
10	2^{13}	2^{-1}	96.49	2^{3}	2^{1}	100.00
Avg.	17408	1.7	96.66	9890.6	1.4	98.95
CPU time		19.208797			10.175330	

Table 6: Accuracy of the WBCD database using the SVM with C and γ which were calculated by the grid search technique (de Buchet $p = 1$ scaling, de Buchet $p = 2$ scaling, and IBM MPSX scaling).

Fold	Geometric mean scaling (S4)			Arithmetic mean scaling (S5)		
	C	γ	Accuracy %	C	γ	Accuracy %
1	2^{11}	2^{-9}	100	2^{3}	2^{-7}	100.00
2	2^{11}	2^{-9}	98.25	2^{15}	2^{-9}	98.25
3	2^{11}	2^{-9}	100	2^{9}	2^{-5}	96.49
4	2^{15}	2^{-13}	98.25	2^{-1}	2^{-5}	96.49
5	2^{11}	2^{-9}	96.49	2^{9}	2^{-9}	100.00
6	2^{11}	2^{-9}	100	2^{5}	2^{-5}	98.25
7	2^{11}	2^{-9}	98.25	2^{7}	2^{-7}	98.25
8	2^{5}	2^{-5}	98.25	2^{-1}	2^{-3}	98.25
9	2^{11}	2^{-9}	100	2^{9}	2^{-9}	100.00
10	2^{11}	2^{-9}	96.49	2^{15}	2^{-9}	98.25
Avg.	4918.4	0.0047	98.60	6724.1	0.024	98.42
CPU time		15.143076			12.516496	

Table 7: Accuracy for the WBCD database using the SVM with C and γ which were calculated by the grid search technique (de Buchet $p = 1$ and $p = 2$).

Fold	de Buchet $p = 1$ (S6)			de Buchet $p = 2$ (S7)		
	C	γ	Accuracy %	C	γ	Accuracy %
1	2^{9}	2^{-7}	96.43	2^{15}	2^{-11}	94.46
2	2^{9}	2^{-11}	98.25	2^{5}	2^{-5}	96.49
3	2^{1}	2^{-5}	100	2^{5}	2^{-5}	100
4	2^{9}	2^{-7}	96.49	2^{9}	2^{-7}	100
5	2^{5}	2^{-7}	100	2^{3}	2^{-3}	100
6	2^{9}	2^{-7}	100	2^{3}	2^{-1}	96.49
7	2^{9}	2^{-7}	100	2^{5}	2^{-5}	98.25
8	2^{9}	2^{-7}	100	2^{5}	2^{-5}	100
9	2^{9}	2^{-7}	100	2^{5}	2^{-1}	100
10	2^{13}	2^{-7}	94.74	2^{5}	2^{-5}	100
Avg.	1181	0.0094	98.59	3348.8	0.129	98.57
CPU time		12.99264			13.28168	

Table 8: Accuracy for the WBCD database using the SVM with C and γ which were calculated by the grid search technique (IBM MPSX).

Fold	IBM MPSX (S8)		
	C	γ	Accuracy%
1	2^{15}	2^{-9}	94.64
2	2^{5}	2^{-1}	100
3	2^{13}	2^{-7}	98.25
4	2^{9}	2^{-5}	98.25
5	2^{15}	2^{-9}	100
6	2^{15}	2^{-9}	100
7	2^{5}	2^{1}	100
8	2^{15}	2^{-9}	100
9	2^{15}	2^{-9}	100
10	2^{5}	2^{1}	94.74
Avg.	17264	0.0031	98.59
CPU time		11.516754	

Table 9: Accuracy for the WBCD database using the SVM with C and γ which were calculated by the grid search technique (L_p-norm $p = 1$ and $p = 2$).

Fold	L_p-norm $p = 1$ (S9)			L_p-norm $p = 2$ (S10)		
	C	γ	Accuracy %	C	γ	Accuracy %
1	2^{11}	2^{-7}	100	2^{13}	2^{-5}	98.21
2	2^{15}	2^{-15}	98.25	2^{15}	2^{-7}	94.74
3	2^{11}	2^{-9}	100	2^{13}	2^{-5}	96.49
4	2^{9}	2^{-7}	98.25	2^{13}	2^{-5}	92.98
5	2^{15}	2^{-15}	96.49	2^{13}	2^{-5}	96.49
6	2^{15}	2^{-15}	96.49	2^{13}	2^{-5}	100
7	2^{9}	2^{-7}	100	2^{9}	2^{-5}	96.49
8	2^{15}	2^{-15}	100	2^{11}	2^{-5}	100
9	2^{15}	2^{-15}	92.98	2^{13}	2^{-5}	98.25
10	2^{15}	2^{-15}	100	2^{13}	2^{-5}	96.49
Avg.	20172.8	0.0026	98.25	9267.2	0.029	97.01
CPU time		13.217008			12.847815	

Table 10: Accuracy for the WBCD database using the SVM for all scaling techniques.

No.	Symbol	Scaling techniques	Accuracy
1	(S3)	Equilibration scaling	98.95
2	(S4)	Geometric mean scaling	98.60
3	(S8)	IBM MPSX	98.59
4	(S6)	de Buchet $p=1$	98.59
5	(S7)	de Buchet $p=2$	98.57
6	(S5)	Arithmetic mean scaling	98.42
7	(S9)	L_p-norm $p=1$	98.25
8	(S10)	L_p-norm $p=2$	97.01
9	(S2)	Normalization (−1, 1)	96.66
10	(S1)	Normalization scaling (0,1)	96.49
11	(S0)	Without scaling	96.10

Table 11: CPU time for the WBCD database using the SVM for all scaling techniques.

No.	Symbol	Scaling techniques	CPU time
1	(S3)	Equilibration scaling	10.175330
2	(S8)	IBM MPSX	11.516754
3	(S5)	Arithmetic mean scaling	12.516496
4	(S10)	L_p-norm $p=2$	12.847815
5	(S6)	de Buchet $p=1$	12.99264
6	(S9)	L_p-norm $p=1$	13.217008
7	(S7)	de Buchet $p=2$	13.28168
8	(S1)	Normalization scaling (0,1)	14.558410
9	(S4)	Geometric mean scaling	15.143076
10	(S2)	Normalization (−1, 1)	19.208797
11	(S0)	Without scaling	52.62167

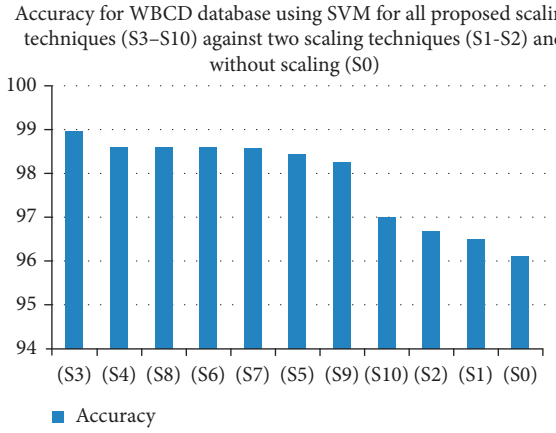

Figure 4: An accuracy comparison among all proposed scaling techniques against the normalization [0, 1], normalization [−1, 1], and without scaling technique S0.

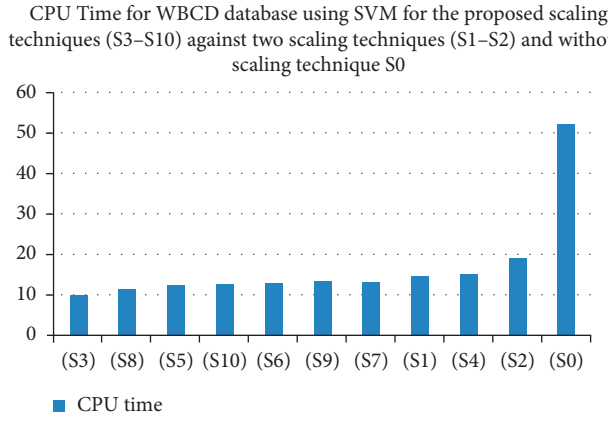

Figure 5: CPU time comparison among all proposed scaling techniques against the normalization [0, 1], normalization [−1, 1], and without scaling technique S0.

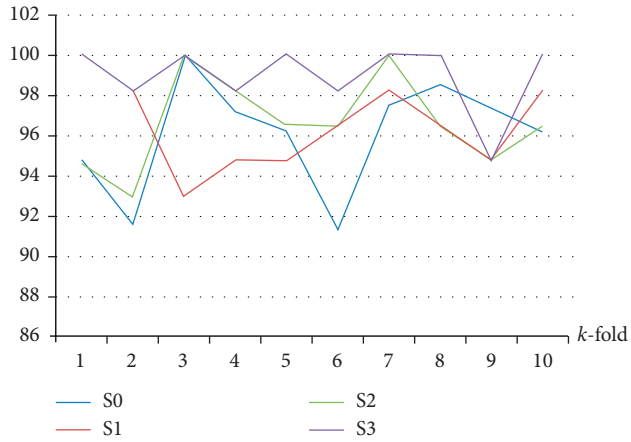

Figure 6: An accuracy comparison among the equilibration scaling technique against the normalization [0, 1], normalization [−1, 1], and without scaling technique S0 according to 10-fold.

S5, S6, S7, and S8. Seven of the ten efficient scaling techniques outperformed two benchmark scaling techniques according to the CPU time. These techniques are S3, S5, S6, S7, and S8.

In the future work, the proposed scaling techniques will be applied on other data sets with other classifiers in order to prove the superiority of these techniques on the benchmark normalization scaling technique that is used in MATLAB SOFTWARE. We can improve this work by using the different metaheuristic algorithms with other mathematical models [26–30]. Also, swarm intelligence techniques will be used to optimize the SVM instead of grid search [31–33].

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at Majmaah University for funding this work under project no. RGP-2019-29 and Deputyship for Research and Innovation, Ministry of Education in Saudi

Arabia, for funding this research work through the project no. IFP-2020-17.

References

[1] A. R. Curtis and J. K. Reid, "On the automatic scaling of matrices for Gaussian elimination," *IMA Journal of Applied Mathematics*, vol. 10, no. 1, pp. 118–124, 1972.

[2] D. R. Fulkerson and P. Wolfe, "An algorithm for scaling matrices," *SIAM Review*, vol. 4, no. 2, pp. 142–146, 1962.

[3] J. A. Tomlin, "On scaling linear programming problems," *Mathematical Programming Studies*, vol. 4, pp. 146–166, 1975.

[4] T. Larsson, "On scaling linear programs–some experimental results," *Optimization*, vol. 27, pp. 335–373, 1993.

[5] J. De Buchet, "Experiments and statistical data on the solving of large–scale linear programs," in *Proceedings of the Fourth International Conference on Operational Research*, pp. 3–13, Wiley-Interscience, Boston, MA, USA, January 1966.

[6] R. W. Hamming, *Introduction to Applied Numerical Analysis*, McGraw-Hill, New York, NY, USA, 1971.

[7] J. M. Elble and N. V. Sahinidis, "Scaling linear optimization problems prior to application of the simplex method," *Computational Optimization and Applications*, vol. 52, no. 2, pp. 345–371, 2012.

[8] N. Ploskas and N. Samaras, "The impact of scaling on simplex type algorithms," in *Proceedings of the 6th Balkan Conference in Informatics*, pp. 17–22, Thessaloniki, Greece, September 2013.

[9] C. Triantafyllidis and N. Samaras, "Three nearly scaling-invariant versions of an exterior point algorithm for linear programming," *Optimization*, vol. 64, no. 10, pp. 2163–2181, 2014.

[10] N. Ploskas and N. Samaras, "A computational comparison of scaling techniques for linear optimization problems on a graphical processing unit," *International Journal of Computer Mathematics*, vol. 92, no. 2, pp. 319–336, 2015.

[11] N. Ploskas and N. Samaras, *Linear Programming Using MATLAB®*, Vol. 127, Springer, Berlin, Germany, 2017.

[12] N. Ploskas and N. Samaras, "GPU accelerated pivoting rules for the simplex algorithm," *Journal of Systems and Software*, vol. 96, pp. 1–9, 2014.

[13] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A Training Algorithm for Optimal Margin Classifiers*, ACM, New York, NY, USA, 1992.

[15] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, NY, USA, 1998.

[16] M.-W. Huang, C.-W. Chen, W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "SVM and SVM ensembles in breast cancer prediction"" *PLoS One*, vol. 12, no. 1, Article ID e0161501, 2017.

[17] J. Zhao and W. Li, "Improvement intrusion detection based on SVM," in *Information Computing and Applications*, C. Liu, L. Wang, and A. Yang, Eds., vol. 308Berlin, Germany, Springer, 2012.

[18] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[19] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian Kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[20] M. Benichou, J. M. Gauthier, G. Hentges, and G. Ribiere, "The efficient solution of large-scale linear programming problems-some algorithmic techniques and computational results," *Mathematical Programming*, vol. 13, no. 1, pp. 280–322, 1977.

[21] C. W. Hsu, C. C. Chang, and C. J. Lin, *A Practical Guide to Support Vector Classification*, National Taiwan University, Taipei, Taiwan, 2003, http://www.csie.ntu.edu.tw/cjlin/libsvm/.

[22] W. H. Wolberg, *Wisconsin Diagnostic Breast Cancer (WDBC)*, University of Wisconsin School of Computer Science, UCI Machine learning Repository, Madison, WI, USA, 1995.

[23] C. C. Chang and C. J. Lin, *LIBSVM: A Library for Support Vector Machines*, National Taiwan University, Taipei, Taiwan, 2001, http://www.csie.ntu.edu.tw/cjlin/libsvm.

[24] S. L. Salzberg, "On comparing classifiers: pitfalls to avoid and a recommended approach," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 317–328, 1997.

[25] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2010.

[26] E. M. Badr and M. I. Moussa, "An upper bound of radio *k*-coloring problem and its integer linear programming model," *Wireless and Networks*, vol. 26, no. 3, 2019.

[27] E. M. Badr and K. Aloufi, "A robot's response acceleration using the metric dimension problem," *Preprint*, 2019.

[28] E. M. Badr and K. Aloufi, "On a dual direct cosine simplex type algorithm and its computational behavior," *Mathematical Problems in Engineering*, vol. 2020, Article ID 7361092, 8 pages, 2020.

[29] E. Badr, S. Almotairi, A. I. Elrokh, A. Abdel-Hay, and B. Almutairi, "An integer linear programming model for solving radio mean labeling problem," *IEEE Access*, vol. 8, pp. 162343–162349, 2020.

[30] E. M. Badr and H. Elgendy, "A hybrid water cycle–particle swarm optimization for solving the fuzzy underground water confined steady flow," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, 2020.

[31] O. Hegazy, O. Soliman, and M. Abdul Salam, "Optimizing LS-SVM using modified cuckoo search algorithm (MCS) for stock price prediction," *International Journal of Advanced Research in Computer Science and Management Studies*, vol. 3, no. 2, pp. 204–224, 2015.

[32] O. Hegazy, O. Soliman, and M. Abdul Salam, "Comparative study between FPA, BA, MCS, ABC, and PSO algorithms in training and optimizing of LS-SVM for stock market prediction," *International Journal of Advanced Computer Research*, vol. 5, no. 18, pp. 35–45, 2015.

[33] O. Hegazy, O. Soliman, and M. Abdul Salam, "FPA-ELM model for stock market prediction," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 2, pp. 1050–1063, 2015.