Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 5255346, 9 pages
https://doi.org/10.1155/2022/5255346

*Research Article*

# Gaussian Pyramid for Nonlinear Support Vector Machine

**Rawan Abo Zidan**[1] **and George Karraz**[1,2,3]

[1]*PhD Program, Syrian Virtual University, Damascus, Syria*
[2]*Faculty of Engineering, Al-Sham Private University, Damascus, Syria*
[3]*Faculty of Engineering, Damascus University, Damascus, Syria*

Correspondence should be addressed to Rawan Abo Zidan; rawanaz.179@gmail.com

Received 19 March 2022; Revised 2 May 2022; Accepted 11 May 2022; Published 31 May 2022

Academic Editor: Manikandan Ramachandran

Support vector machine (SVM) is one of the most efficient machine learning tools, and it is fast, simple to use, reliable, and provides accurate classification results. Despite its generalization capability, SVM is usually posed as a quadratic programming (QP) problem to find a separation hyperplane in nonlinear cases. This needs huge quantities of computational time and memory for large datasets, even for moderately sized ones. SVM could be used for classification tasks whose number of samples is limited but does not scale well to large datasets. The idea is to solve this problem by a smoothing technique to get a new smaller dataset representing the original one. This paper proposes a fast and less time and memory-consuming algorithm to solve the problems represented by a nonlinear support vector machine tool, based on generating a Gaussian pyramid to minimize the size of the dataset. The reduce operation between dataset points and the Gaussian pyramid is reformulated to get a smoothed copy of the original dataset. The new dataset points after passing the Gaussian pyramid will be closed to each other, and this will minimize the degree of nonlinearity in the dataset, and it will be 1/4 of the size of the original large dataset. The experiments demonstrate that our proposed techniques can reduce the classical SVM tool complexity, more accurately, and are applicable in real time.

## 1. Introduction

Support vector machine is used for classification and regression purposes. SVM offers very high accuracy, and it aims to find the best hyperplane (also called decision boundary) with the largest amount of margin. SVM finds an optimal hyperplane which helps in classifying new data points. In other words, SVM allows for maximizing the generalization ability of a model [1].

The SVM algorithm was originally proposed to construct a linear classifier in 1963 by Vapnik [2]. At that time, the algorithm was in its early stages, and the only possibility is to draw hyperplanes for a linear classifier.

An alternative use for SVM is the kernel trick for nonlinear classifiers, which was introduced In 1992, by Boser et al. [3] which enables us to model higher dimensional space, and it converts nonlinear separable problems to linear separable problems by adding more dimensions to it.

The classical way to incorporate nonlinearity into SVM is to derive the dual formulation (quadratic programming problem) and employ the kernel method [4]. Moreover, dual problems are generally expensive to solve [5].

The standard SVM faces some disadvantages such as SVM for large datasets, due to its excessive computational cost because the training kernel matrix grows in quadratic form with the size of the dataset, which provokes that training of SVM on large datasets is a very slow process [1].

The training dataset may contain up to several thousands of samples, and this implies that training time complexity and space complexities are $O(n^3)$ and $O(n^2)$, where n is the number of points in the dataset. It is thus computationally infeasible on very large datasets.

Nowadays, the biggest challenge is to develop efficient and scalable learning algorithms to deal with “big data.” To solve this challenge, try to reduce the problem size and computations by considering fewer parameters or fewer instances during each iteration of the learning algorithm [6].

The main aim of this paper is to apply the Gaussian pyramid to nonlinear SVM to improve the training of weak SVM classifiers. The reason behind choosing the Gaussian pyramid is that the core of the Gaussian pyramid is a convolutional smoothing operation [7], which is used in image processing that breaks down an image into successively smaller groups of pixels to blur it, but in our case, we will use the Gaussian pyramid on the large dataset to smooth it and make it smaller and then apply the linear SVM on it. The new technique reduces time and space complexities and can handle much larger datasets than existing scale-up methods.

The outcome of the Gaussian pyramid algorithm generates a linear separating surface that depends on 1/4 of the original dataset size only, instead of the conventional nonlinear kernel surface which would depend on the entire points. This is very important for large datasets such as those used in fraud detection.

The rest of this paper is organized as follows. We briefly introduce the related work of SVM methods and different learning models in Section 2, problem description in Section 3, followed by motivation and objective in Section 4. Then, we explain the details of our proposed framework in Section 5, followed by comprehensively reporting the designed experiments in Section 6. Conclusion is given in Section 7. Finally, we discuss the future studies in Section 8.

## 2. Related Work

In this section, we review previous SVM methods. Generally, the state-of-the-art approaches address SVM classification problems.

Platt [8] proposed sequential minimal optimization (SMO) which is an algorithm for solving the quadratic programming (QP) problem that appears in SVM.

While SMO has been shown to be effective on sparse datasets and especially fast for linear SVMs, the algorithm can be extremely slow on nonsparse datasets and on problems that have many support vectors.

It also suffers from limitations in generalizing because it depends on reducing the problem to smaller problems.

The algorithm was applied to a database consisting of 32562 records as shown in Table 1.

Table 1: Running times of different databases to train an SVM using SMO.

| #Samples of the database | Test time (s) |
|---|---|
| 2265 | 0.9 |
| 3185 | 1.8 |
| 4781 | 3.6 |
| 6414 | 5.5 |
| 11221 | 17.0 |
| 16101 | 35.3 |
| 22697 | 85.7 |
| 32562 | 163.6 |

Zareapoor et al. [9] presented a hybrid system where a supervised deep belief network that has multiple hidden layers is trained to select generic features and a kernel-based SVM is trained from the features learned by the DBN.

In this hybrid model, the researchers substituted linear kernels for nonlinear ones (due to a large number of classes) without loss of accuracy, and this gives significant gains on a real-world dataset with varying numbers of dimensions and records, (from 500 to 10,000 records). To evaluate the impact of data size on the performance, the researchers used operating characteristic (ROC) curve and the corresponding area under the curve (AUC) as shown in Table 2.

This model shows some drawbacks in terms of scalability to the size of datasets, so different patterns of data led to a decrease in performance.

As the previous table shows, the percentage of area under the curve in the private database is low, which is the largest among the databases in size.

Sadrfaridpour et al. [10] introduced novel methods of multilevel frameworks for efficient and effective training of nonlinear SVM classifiers, the framework inspired by the algebraic multigrid.

The proposed multilevel frameworks are particularly effective on imbalanced datasets; this problem occurs when the number of instances of one class (negative or majority class) is substantially larger than the number of instances that belong to the other class (positive or minority class).

The computation time of the proposed multilevel frameworks exhibits a significant improvement and can generate several classifiers at different coarse-grained resolutions in one complete training iteration which also helps to interpret these classifiers qualitatively.

Overall, the complexity of the entire framework is linear in the number of data points.

Table 3 shows some of the experimental results of the proposed model.

However, the model suffers from classification problems. For example, on the ISOLET instance, the tool computes a model that puts all data points on a single side [11], while the purpose of the support vector machines is to separate the data by a decision boundary.

Chen et al. [12] improved the projection twin support vector machine (PTSVM) algorithm to a novel nonparallel classifier, termed V-PTSVM.

V-PTSVM is equipped with a more theoretically sound parameter $V$, which can be used to control the bounds of a fraction of both support vectors and margin-error instances.

As the researchers mentioned in their scientific paper that the algorithm still suffers from problems with the increasing number of data, the developed algorithm has proven effective when applied to the NDC database as shown in Table 4.

Li et al. [13] proposed to overcome the defect of PSVM on feature selection, and the purpose of PSVM is to generate a pair of nonparallel hyperplanes for classification.

The paper introduced $\ell_0$-norm regularization in PSVM which enables PSVM to select important features and remove redundant features simultaneously for classification.

The effectiveness of $\ell_0$-PSVM to obtain sparse classifiers, an alternating scheme based on DCA, is proposed by using a nonconvex continuous function.

Table 2: Running times and AUC of databases using a hybrid system.

| Database | #Samples | Train time (s) | Test time (s) | AUC |
|---|---|---|---|---|
| OAR | 2551 | 1, 04 | $10^{-3}\times 1, 6$ | %90, 5 |
| HAR | 10299 | 1, 12 | $10^{-3}\times 1, 5$ | %91, 9 |
| Private database | 398,690 | 1, 48 | $10^{-2}\times 1, 8$ | %80, 4 |

Table 3: Performance measures and computational time of databases using multilevel frameworks.

| Databases | #Samples | Computational time (s) | ACC (%) | SN | SP |
|---|---|---|---|---|---|
| Advertisement | 3 279 | 91 | 0.94 | 0.96 | 0.80 |
| Buzz | 140 707 | 957 | 0.94 | 0.96 | 0.87 |
| Clean (Musk) | 6 598 | 6 | 1.00 | 1.00 | 0.99 |
| Cod-rna | 59 535 | 92 | 0.94 | 0.97 | 0.95 |
| ISOLET | 6 919 | 64 | 0.99 | 1.00 | 0.85 |

Table 4: Computational results of V-PTSVM.

| Database NDC | Time (s) | ACC (%) |
|---|---|---|
| $(100\times 32)$ | 0.0402 | 80.59 |
| $(500\times 32)$ | 0.1046 | 82.84 |
| $(1k\times 32)$ | 5.6840 | 85.49 |
| $(3k\times 32)$ | 39.252 | 86.57 |
| $(5k\times 32)$ | 317.58 | 86.24 |
| $(5k\times 32)$ | 3628.4 | 85.97 |

DCA is an efficient descent method with linear convergence, which has been widely used in numerous nonconvex optimization problems [14].

Table 5 shows benchmark datasets used in experiments and comparative results in terms of the average classification accuracy, the average number of selected features, the average training time, and AUC.

The paper observed that PSVM, sPSVM, and $\ell_0$-PSVM are time-consuming classifiers since their training time increases rapidly as the number of features increases.

Ma et al. [15] proposed a new robust loss function called adaptive capped $\ell_\theta\varepsilon$-loss and proposed a new robust distance metric induced by correntropy (CIM) that is based on the Laplacian kernel.

The researchers applied the $\ell_\theta\varepsilon$-loss and CIM to a twin support vector machine (TWSVM) and developed an adaptive robust learning framework, namely, adaptive robust twin support vector machine (ARTSVM).

The proposed ARTSVM not only inherits the advantages of TWSVM but also improves the robustness and accuracy of classification problems.

All experiments on large-scale databases with 0% (without noise), 10%, and 30% label noises are presented in Table 6.

The proposed ARTSVM has shown good robustness to feature noise and outliers in most cases, but from the perspective of time consumption, ARTSVM is undoubtedly inferior to the other algorithms in terms of learning time.

This is because the ARTSVM algorithm requires a lot of time to perform iterative calculations and needs to remove outliers during the training process.

## 3. Problem Description

In nonlinear kernel enables us to model higher dimensional space, for a given binary classification problem, If $x \in \mathfrak{R}n$ is an input point, let $\phi(x)$ be the corresponding feature point with $\phi$ a mapping from $\mathfrak{R}n$ to certain space called feature space.

This is very computationally expensive, especially if the mapping is to a high-dimensional space. But many works of literature show that kernel function can be used to accomplish the same result efficiently.

The kernel is a function $k(x_i.x_j)$ that given two vectors in input space returns the dot product of their images in feature space.

$$k(x_i.x_j) = \phi(x_i)\cdot\phi(x_j). \tag{1}$$

By computing the dot product directly using a kernel function, one avoids the mapping $\phi\ (x)$.

This is desirable because $Z$ has possibly infinite dimensions and $\phi(x)$ can be tricky or impossible to compute. Using a kernel function, one does not need to explicitly know what $\phi(x)$ is. By using a kernel function, an SVM that operates in infinite-dimensional space can be constructed. Also, the decision function will be

$$f(x) = \sum_{i=1}^{m} a_i\, y_i K(x_i\,.x_j) + b, \tag{2}$$

where $x = [x_1, x_2, \ldots, x_l]$ represent the input data, $\alpha$ is the Lagrange multiplier, $b$ is the offset, and $y_i$ is the output label.

But the kernel function still requires the inversion of the $n\times n$ matrix, which needs huge quantities of computational time and memory for large datasets; therefore, the training complexity of SVM is highly dependent on the size of a dataset.

Large datasets impose heavy computational time and storage requirements during training, sometimes rendering SVM even slower than ANN. For this reason, support vector set cardinality may be a problem when online prediction requires real-time performance on platforms with limited computational and power supply capabilities, such as mobile devices [16].

## 4. Motivation and Objective

The motivation behind this model is to improve the conventional SVM algorithm shortage by the following:

(1) Get rid of the process of processing large datasets, and this drawback is essentially related to the necessity to store and manipulate large, dense, and unstructured kernel matrices [17].

(2) Kernel SVM always brings additional parameters, and one may need to pay lots of effort to tune the parameters for better performance [18, 19]. Improper setting of the hyperparameters often brings overfitting or underfitting problems. Last but not least, even though we can use kernel trick to project the data into high-dimensional space, there is a

Table 5: Results for $\ell_0$-PSVM.

| Database | #Samples | #Features | #Features after selection | Accuracy (%) | Time | AUC |
|---|---|---|---|---|---|---|
| WPBC | 198 | 33 | 33 | 76.13 | 0.669 | 0.84 |
| Ionosphere | 351 | 34 | 34 | 83.13 | 0.856 | 0.71 |
| Spambase | 4601 | 57 | 57 | 62.43 | 0.031 | 0.73 |
| Heart | 270 | 13 | 12 | 86.29 | 0.466 | 0.86 |
| Glass1 | 214 | 9 | 8 | 68.45 | 0.140 | 0.82 |
| Vehicle1 | 846 | 18 | 17 | 77.70 | 0.343 | 0.80 |
| Vehicle3 | 846 | 18 | 17 | 74.95 | 0.637 | 0.78 |

Table 6: Performance measures of ARTSVM.

| Database | $R$ (%) (noise ratio) | #Samples | ACC |
|---|---|---|---|
| Spam | 0 | $(4601 \times 57)$ | 85.04 |
| Abalone | 0 | $(4177 \times 8)$ | 86.11 |
| Spam | 10 | $(4601 \times 57)$ | 83.26 |
| Abalone | 10 | $(4177 \times 8)$ | 83.55 |
| Spam | 30 | $(4601 \times 57)$ | 81.05 |
| Abalone | 30 | $(4177 \times 8)$ | 80.19 |

possibility it cannot be linearly separable after using kernel trick. Specific kernel with specific hyper-parameters may fail on some datasets [20].

(3) Fast and applicable in real-time, because we are dealing with a huge volume of data we always need less time-consuming methods.

## 5. Proposed Work

This paper presents a new algorithm based on the Gaussian pyramid to make large-scale training of SVM.

In the Gaussian pyramid, subsequent images of the preceding level of the pyramid are weighted down by means of Gaussian average (or Gaussian blur) and scaled down [21]. The developed algorithm uses the same technique of images but on the dataset and passes the Gaussian pyramid once on the large dataset to shrink its size to make it linearly separable. Furthermore, the shrinking technique of the problem during the training of nonlinear SVM is found particularly effective for large learning tasks.

We assume that (X, Y) the training patterns set, where X = {x1, x2, . . ., xn} is the input dataset Y = {y1, y2, . . ., yn} is the label set, label$y_i \in \{-1, 1\}$. The main steps used to implement the Gaussian pyramid algorithm are summarized as follows.

### 5.1. Overview

*5.1. Overview.* An overview of our algorithm is shown in Algorithm 1. Our algorithm starts by preprocessing the data. Then, we need to calculate the correlation matrix to calculate the correlation among all the features.

The next step is to generate a Gaussian pyramid whose size is user input, but it must be an odd number to have a central element.

Now we have the randomly generated Gaussian pyramid, then we need to pass it on to the large dataset in columns and rows, by multiplying each element from the Gaussian pyramid to its corresponding element from the dataset, and then sum all the multiplications and put it in the new dataset.

This new element represents the previous elements from the large old dataset in which we smooth it bypassing the Gaussian pyramid, and this is called the reduce operation.

To evaluate our algorithm, we need to see the distribution of the new data by calculating the multivariate Gaussian distribution; after that, we can apply linear SVM and find the hyperplane to separate the data.

*5.2. Data Preprocessing.* The following preprocessing has been done to the data:

(1) Oversampling to provide class balancing.

(2) Missing values based on mean, median, or mode; it calculates the imputation based on the other feature values for that sample.

(3) One-hot encoding to convert categorical features to numerical attribute.

*5.3. Correlation Matrix.* The correlation matrix gives the correlation coefficients among all the columns in a given matrix. The most familiar measure of dependence is Pearson's correlation coefficient. It computes the correlation of all the columns with themselves.

The correlation matrix is a symmetrical matrix with all diagonal elements equal to +1 because the correlation of a variable with itself is 1, and the Pearson correlation coefficient formula is

$$r = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum (x_i - \overline{x})^2 \sum (y_i - \overline{y})^2}}, \quad (3)$$

where $r$ = correlation coefficient, $x_i$ = values of the x-variable in a sample, $\overline{x}$ = mean of the values of the x-variable, $y_i$ = values of the y-variable in a sample, and $\overline{y}$ = mean of the values of the y-variable.

*5.4. Gaussian Pyramid.* The algorithm generates a Gaussian pyramid that has the following properties:

(1) Has an odd × odd 2D matrix, so it will have a central element.

(2) Has randomly generated numbers between 0 and 1.

```
Input: $D$: the training dataset $(x_i, y_i)$, $k$: the size of the Gaussian pyramid must be an odd number
Output: find an optimal hyperplane
Start
  Step 1: data preprocessing on $D$
  Step 2: build the correlation matrix of $D$ by equation:
  $r = (\sum (x_i - \overline{x})(y_i - \overline{y}) / \sqrt{\sum (x_i - \overline{x})^2 \sum (y_i - \overline{y})^2})$
  Step 3: randomly generate a Gaussian pyramid as a $k \times k$ matrix between 0 and 1
  Step 4: reduce the size of $D$ using $K$ to produce $D_{\text{new}}$, a new training dataset, by equation:
  $g_l(i.j) = \sum_m \sum_n w(m.n) g_{l-1}(2i + m.2j + n)$
  Step 5: find the multivariate normal distribution of $D_{\text{new}}$ by the following probability density function:
  $y = f(x.\mu.\Sigma) = (1/(2\pi)^{(n/2)} |\Sigma|^{(1/2)}) \exp(-(1/2)(x - \mu)^T \Sigma^{-1} (x - \mu))$
  Step 6: classify $D_{\text{new}}$ to find an optimal hyperplane and build the model to classify new data points.
End
```

Algorithm 1: Overview.

*5.5. Reduce Operation.* The reduce operation in Gaussian pyramids is done according to the relation given below.

$$g_l(i.j) = \sum_m \sum_n w(m.n) g_{l-1}(2i + m.2j + n), \tag{4}$$

where $l$ represents the level and $w(m.n)$ is the generated Gaussian pyramid; in reducing operation, we will reduce half of width and height, bypassing the generated matrix of the Gaussian pyramid on the large dataset.

The newly generated dataset after performing the reduce operation is 1/4 of the size of the previous dataset.

*5.6. Multivariate Gaussian Distribution.* The multivariate normal distribution is a generalization of the univariate normal distribution to two or more variables. It is a distribution for random vectors of correlated variables, where each vector element has a univariate normal distribution. A vector-valued random variable $X = [X_1 \cdots\cdots X_n]^T$ is said to have a multivariate normal (or Gaussian) distribution if its probability density function (pdf) is given by

$$y = f(x.\mu.\Sigma) = \frac{1}{(2\pi)^{(n/2)} |\Sigma|^{(1/2)}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right), \tag{5}$$

where $\mu$ is the mean vector and $\Sigma$ is the covariance matrix. Diagonal elements contain the variances for each variable, and off-diagonal elements contain the covariance between variables.

Before applying the standard linear SVM, we need to see the distribution of the new data by calculating the multivariate Gaussian distribution; if the data have a narrow Gaussian, this means it is linearly separable.

*5.7. Linear SVM.* Linear support vector machines (SVMs) are originally formulated for binary classification. Given training data and its corresponding labels $(x_n.y_n)$, $n = 1, \ldots,$ N, $x_n \in R^D$, $t_n \in \{-1. + 1\}$, SVMs learning consists of the following constrained optimization:

$$\begin{aligned} &\min_{w.\xi_n} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n, \\ &\text{s.t.} \quad \begin{aligned} &w^T x_n t_n \geq 1 - \xi_n, \ \forall n, \\ &\xi_n \geq 0, \ \forall n, \end{aligned} \end{aligned} \tag{6}$$

where $\xi_n$ are slack variables which penalize data points that violate the margin requirements.

## 6. Experimental Results

This section presents the experimental results to research the utility of our approach.

We used three different sizes of datasets:

(1) Spambase [22] from UCI machine learning repository, with 4601 samples and 58 features.

(2) QSAR biodegradation [23] from UCI machine learning repository, with 1055 samples and 42 features.

(3) Swarm behaviour [24] from Kaggle, with 23309 samples and 2401 features.

The datasets have been split into 70% for training and the remaining 30% for testing, and we will use all features without any selection.

*6.1. Accuracy Measure.* Accuracy of results from the Gaussian pyramid method is commonly measured by the quantity of standard sensitivity, specificity, F1-score, and overall accuracy defined by

Sensitivity (true positive rate): it refers to the probability of a positive test, conditioned on truly being positive.

$$\text{Sensitivity} = \frac{\text{TP}}{(\text{TP} + \text{FN})}. \tag{7}$$

Specificity (true negative rate): it refers to the probability of a negative test, conditioned on truly being negative.

Table 7: Test results for Spambase dataset.

| Kernel size | Sensitivity | Specificity | F1-score | Accuracy | Training time (s) | AUC |
|---|---|---|---|---|---|---|
| $3 \times 3$ | 0.85 | 0.98 | 0.89 | 0.90 | 334.62 | 0.95 |
| $5 \times 5$ | 0.41 | 1.0 | 0.70 | 0.65 | 145.07 | 0.93 |
| $7 \times 7$ | 0.95 | 0.61 | 0.73 | 0.81 | 92.46 | 0.93 |
| $9 \times 9$ | 1.0 | 0.17 | 0.29 | 0.66 | 79.08 | 0.83 |
| $11 \times 11$ | 0.91 | 0.85 | 0.87 | 0.88 | 53.17 | 0.96 |
| $13 \times 13$ | 0.54 | 1.0 | 0.74 | 0.72 | 45.06 | 0.95 |

Table 8: Test results for QSAR biodegradation dataset.

| Kernel size | Sensitivity | Specificity | F1-score | Accuracy | Training time (s) | AUC |
|---|---|---|---|---|---|---|
| $3 \times 3$ | 0.86 | 0.85 | 0.82 | 0.85 | 93.99 | 0.93 |
| $5 \times 5$ | 0.98 | 0.57 | 0.71 | 0.85 | 36.26 | 0.90 |
| $7 \times 7$ | 0.92 | 0.95 | 0.92 | 0.93 | 26.29 | 0.97 |
| $9 \times 9$ | 0.52 | 1.0 | 0.70 | 0.69 | 23.35 | 0.82 |
| $11 \times 11$ | 0.53 | 0.91 | 0.67 | 0.67 | 36.71 | 0.77 |
| $13 \times 13$ | 0.88 | 0 | 0 | 0.6 | 26.36 | 0.77 |

Table 9: Test results for Swarm behaviour dataset.

| Kernel size | Sensitivity | Specificity | F1-score | Accuracy | Training time (s) | AUC |
|---|---|---|---|---|---|---|
| $3 \times 3$ | 0.93 | 0.92 | 0.90 | 0.93 | 39376.87 | 0.98 |
| $5 \times 5$ | 0.95 | 0.97 | 0.94 | 0.96 | 18270.78 | 0.98 |
| $7 \times 7$ | 0.75 | 0.83 | 0.74 | 0.78 | 11076.68 | 0.87 |
| $9 \times 9$ | 0.89 | 0.95 | 0.88 | 0.91 | 7747.42 | 0.98 |
| $11 \times 11$ | 0.66 | 0.85 | 0.70 | 0.73 | 5810.51 | 0.84 |
| $13 \times 13$ | 0.79 | 0.90 | 0.79 | 0.83 | 4724.65 | 0.94 |

$$\text{Specificity} = \frac{\text{TN}}{(\text{TN} + \text{FP})}. \tag{8}$$

F-measure: it is a measure of a test's accuracy.

$$F1 - \text{score} = \frac{\text{TP}}{\text{TP} + (1/2)(\text{FP} + \text{FN})}. \tag{9}$$

Accuracy: it indicates the proportion of correct classifications of the total records in the testing set.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{10}$$

where

(i) True positive (TP): anomaly instances correctly classified as an anomaly.
(ii) False positive (FP): normal instances wrongly classified as an anomaly.
(iii) True negative (TN): normal instances correctly classified as normal.
(iv) False negative (FN): anomaly instances wrongly classified as normal.

*6.2. Gaussian Pyramid Performance Analysis.* We implemented the algorithm described in the previous section using Python. Our code uses sklearn and matplotlib. All of our experiments are executed on a machine with an Intel(R) Core(TM) i5-5200U 2.20 GHz processor and 8 GB of RAM.

First, for computing the correlation coefficients among all the features, we calculate the correlation matrix; in the proposed algorithm, we took all the features without any selection; this implies the ability to handle massive problems with a large number of features without any elimination of them.

To evaluate the performance of the algorithm, we generate several Gaussian pyramid sizes and applied the accuracy measures in equations (7)–(10) (see Tables 7–9).

From Table 8, the best prediction accuracy is 0.90% when Gaussian the pyramid size is equal to $3 \times 3$ and the worst prediction pyramid size is $5 \times 5$ with a prediction accuracy of 0.65%. However, the AUC for all Gaussian pyramid sizes gives us good results.

As shown in Table 8, if the Gaussian pyramid has a small odd number, it will be more accurate but more time-consuming than the bigger Gaussian pyramid size, so we need to trade off between time and accuracy in choosing the Gaussian pyramid size.

Table 9 predicted that the processing time of the Gaussian pyramid increases as the data size increases. On the other hand, the processing time of the new algorithm is incomparably low compared to classical nonlinear SVM on large datasets.

The new algorithm is capable of classifying datasets with thousands of samples even millions, and it is a computationally low method that requires nothing more complex than the multiplication of the Gaussian pyramid matrix with the input space, to smooth the dataset to minimize the nonlinearity.

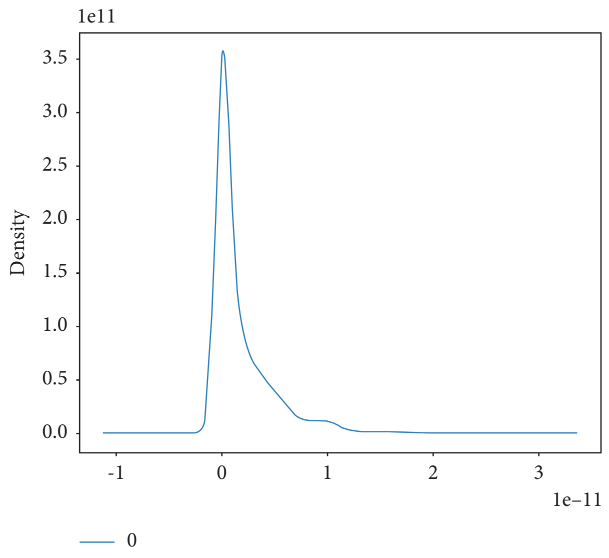

Figure 1: Probability of multivariate Gaussian with Gaussian pyramid $3 \times 3$.

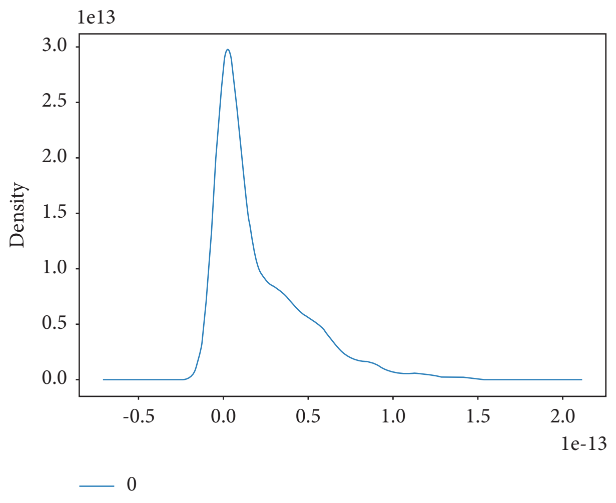

Figure 2: Probability of multivariate Gaussian with Gaussian pyramid $5 \times 5$.

*6.3. Result of Applying Multivariate Gaussian Distribution.* A multivariate Gaussian model can capture the correlations between variables from different dimensions by formulating and calculating a covariance matrix [25].

The result of applying multivariate Gaussian distribution with different Gaussian pyramid sizes on the Spambase dataset is shown in Figures 1–6.

As a result of applying different Gaussian pyramid sizes, we get thin Gaussian after passing the pyramid, and this implies that now we can separate the data linearly and find the hyperplane using linear SVM.

The limitations of the proposed study are it performs better with a large number of features and data samples than smaller datasets, like in the Swarm dataset we get better accuracy and AUC results than QSAR biodegradation and Spambase datasets, which is the largest number of features and data samples in our experiment.

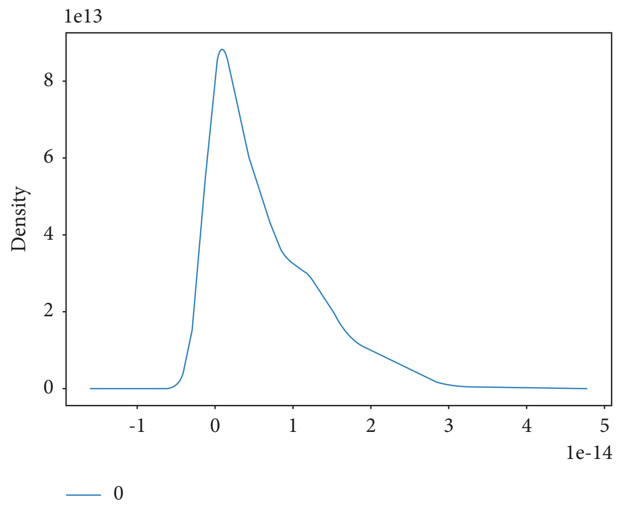

Figure 3: Probability of multivariate Gaussian with Gaussian pyramid $7 \times 7$.

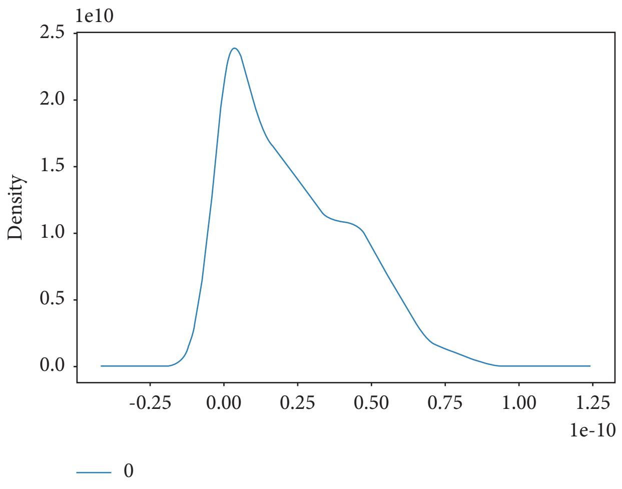

Figure 4: Probability of multivariate Gaussian with Gaussian pyramid $9 \times 9$.

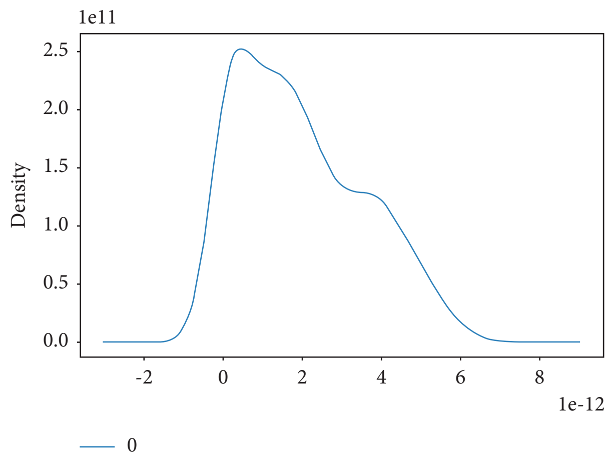

Figure 5: Probability of multivariate Gaussian with Gaussian pyramid $11 \times 11$.

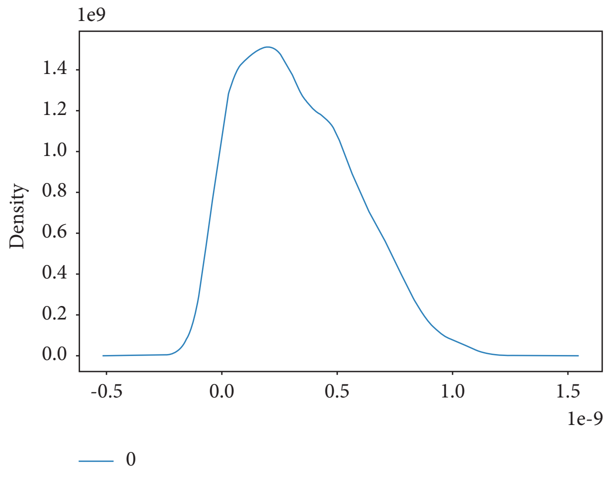

Figure 6: Probability of multivariate Gaussian with Gaussian pyramid $13 \times 13$.

## 7. Conclusion

In this paper, a Gaussian pyramid approach has been proposed for the nonlinear support vector machine. The enhanced algorithm has been applied to different sizes of datasets. Achieved results can be summarized as follows:

(1) We proposed a new method for solving the problems represented by a nonlinear support vector machine tool, by using a Gaussian pyramid in a large dataset.

(2) We performed an analysis of the proposed method. We showed its good performance over the classical nonlinear SVM.

(3) The new approach performs better with a large number of samples and features like Swarm behaviour dataset versus smaller ones.

In contrast to previous approaches, our Gaussian pyramid can shrink the input size to 1/4 size of the original dataset by generating a Gaussian pyramid and passing it on the large dataset and calculating the multivariate Gaussian distribution, and the newly generated dataset after passing the pyramid could be linearly separable using the standard linear SVM.

## 8. Future Studies

In futures studies, we need to

(1) Study the relation between the number of features and samples of the dataset and the Gaussian pyramid size.

(2) Apply the new Gaussian pyramid algorithm on multiclass SVM.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.

[2] V. Vapnik, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, vol. 24, pp. 774–780, 1963.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, Pittsburgh, Pennsylvania, USA, July 1992.

[4] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond*, MIT Press, Cambridge, MA, USA, 2018.

[5] V. Mácha, L. Adam, and V. Šmídl, "Nonlinear classifiers for ranking problems based on Kernelized SVM," 2020, https://arxiv.org/abs/2002.11436.

[6] V. K. Chauhan, K. Dahiya, and A. Sharma, "Problem formulations and solvers in Linear SVM: a review," *Artificial Intelligence Review*, vol. 52, pp. 803–855, 2019.

[7] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, "Beyond Gaussian pyramid: multi-skip feature stacking for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015.

[8] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," 1998.

[9] M. Zareapoor, P. Shamsolmoali, D. K. Jain, H. Wang, and J. Yang, "Kernelized support vector machine with deep learning: an efficient approach for extreme multiclass dataset," *Pattern Recognition Letters*, vol. 115, pp. 4–13, 2018.

[10] E. Sadrfaridpour, T. Razzaghi, and I. Safro, "Engineering fast multilevel support vector machines," *Machine Learning*, vol. 108, pp. 1879–1917, 2019.

[11] S. Schlag, M. Schmitt, and C. Schulz, "Faster support vector machines," *Journal of Experimental Algorithmics*, vol. 26, pp. 1–21, 2021.

[12] W. J. Chen, Y. H. Shao, C. N. Li, M. Z. Liu, Z. Wang, and N. Y. Deng, "$\nu$-projection twin support vector machine for pattern classification," *Neurocomputing*, vol. 376, pp. 10–24, 2020.

[13] G. Li, L. Yang, Z. Wu, and C. Wu, "DC programming for sparse proximal support vector machines," *Information Sciences*, vol. 547, pp. 187–201, 2021.

[14] H. A. Le Thi and T. P. Dinh, "DC programming and DCA: thirty years of developments," *Mathematical Programming*, vol. 169, pp. 5–68, 2018.

[15] J. Ma, L. Yang, and Q. Sun, "Adaptive robust learning framework for twin support vector machine classification," *Knowledge-Based Systems*, vol. 211, Article ID 106536, 2021.

[16] M. Awad and R. Khanna, "Support vector machines for classification," *Efficient Learning Machines*, vol. 65, pp. 39–66, 2015.

[17] S. Cipolla and J. Gondzio, "Training very large scale nonlinear svms using alternating direction method of multipliers coupled with the hierarchically semi-separable kernel approximations," 2021, https://arxiv.org/abs/2108.04167.

[18] V. Sharma, D. Baruah, D. Chutia, P. L. N. Raju, and D. K. Bhattacharya, "An assessment of support vector machine kernel parameters using remotely sensed satellite data," in *Proceedings of the 2016 IEEE International Conference on*

*Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, May 2016.

[19] K. H. Thung, C. Y. Wee, P. T. Yap, and D. Shen, "Identification of progressive mild cognitive impairment patients using incomplete longitudinal MRI scans," *Brain Structure and Function*, vol. 221, pp. 3979–3995, 2016.

[20] F. Nie, W. Zhu, and X. Li, "Decision tree SVM: an extension of Linear SVM for non-linear classification," *Neurocomputing*, vol. 401, 2020.

[21] S. T. M. Ataky, J. de Matos, A. D. S. Britto, L. E. Oliveira, and A. L. Koerich, "Data augmentation for histopathological images based on Gaussian-Laplacian pyramid blending," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, UK, July 2020.

[22] "Spambase dataset," https://archive.ics.uci.edu/ml/datasets/Spambase.

[23] "QSAR biodegradation dataset," https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation.

[24] "Swarm behaviour dataset," https://www.kaggle.com/deepcontractor/swarm-behaviour-classification.

[25] Y. An and D. Liu, "Multivariate Gaussian-based false data detection against cyber-attacks," *IEEE Access*, vol. 7, pp. 119804–119812, 2019.