Nonlinear Analysis: Modelling and Control, Vol. 25, No. 2, 266–281
https://doi.org/10.15388/namc.2020.25.16517

ISSN: 1392-5113
eISSN: 2335-8963

# Support vector machine parameter tuning based on particle swarm optimization metaheuristic

**Konstantinas Korovkinas**[a], **Paulius Danėnas**[b], **Gintautas Garšva**[a]

[a]Kaunas Faculty, Vilnius University,
Muitinės str. 8, LT-44280, Kaunas, Lithuania
konstantinas.korovkinas@knf.vu.lt; gintautas4garsva@gmail.com

[b]Center of Information Systems Design Technologies,
Department of Information Systems, Kaunas University of Technology,
Studentų str. 50-313a, LT-51368, Kaunas, Lithuania
paulius.danenas@ktu.lt

**Received:** January 14, 2019 / **Revised:** July 15, 2019 / **Published online:** March 2, 2020

**Abstract.** This paper introduces a method for linear support vector machine parameter tuning based on particle swarm optimization metaheuristic, which is used to find the best cost (penalty) parameter for a linear support vector machine to increase textual data classification accuracy. Additionally, majority voting based ensembling is applied to increase the efficiency of the proposed method. The results were compared with results from our previous research and other authors' works. They indicate that the proposed method can improve classification performance for a sentiment recognition task.

**Keywords:** particle swarm optimization, support vector machine, textual data classification.

## 1 Introduction

Textual data analysis is a very challenging area. We need to understand the whole context of the sentence because even a single word can change the polarity of a sentence, and this might have a significant impact on particular domains, such as medicine, stock prediction, etc. A support vector machine (SVM) is one of the most frequently used machine learning algorithms to solve sentiment classification problem in [2, 32]. Its efficiency has been proved to solve difficult tasks in different domains, such as image classification in [25], credit risk evaluation in [9], for sensor multifault diagnosis in [10], monitoring metal-oxide surge arrester conditions in [21], Parkinsonian disorders classification in [14], forecasting stock market movement direction in [38], sentiment analysis in [27, 28, 30], etc. The authors in [44] reported that a linear SVM achieves the best results consistently to SVM with different kernels including SVM-Poly. The authors in [7, 16, 26] also reported the linear SVM efficiency for binary text classification. Unfortunately,

manual hyperparameter selection still remains one of the practical application issues, while recent literature still does not provide any heuristic rules or rules of thumb for this task in [41]. Hence, it still requires training multiple classifiers with different sets of hyperparameters to obtain satisfiable performance. Such hyperparameter optimization is mostly guided by some heuristics, like genetic algorithm in [24] and [6], particle swarm optimization (PSO) in [45] and [17], and colony optimization in [22] and [46]. Simple grid search is one of the most common choices to solve this problem [1] as it often integrated in different machine learning packages, such as LibSVM [5] or scikit-learn [13], which helps to simplify research pipelines. Particle swarm optimization is also a very promising option [19, 23, 29]. One of its strengths is combination with other evolutionary techniques. In [36], the authors proposed an improved quantum behaved particle swarm algorithm based on a mutation operator. In [47], the authors presented the SVM parameter optimization technique based on intercluster distance in the feature space and a hybrid of the barebones particle swarm optimization and differential evolution. In [21], a differential particle swarm optimization to select parameters for support vector machines is applied. There are a number of works, which focus on the combined selection of both features and hyperparameters [31, 42].

Ensembles of classifiers are one of the most challenging areas yet they often result in increased performance compared to single classifiers. In [34], the proposed ensemble method is based on static classifier selection involving a majority voting error and forward search for text sentiment classification. In [35], an ensemble system based on three classifiers, which are combined via a majority voting for the sentiment analysis of textual data is presented. In [4], the authors reported that their ensemble voting algorithm in conjunction with three classifiers performed better on Turkish sentiment classification problem.

Motivated by these improvements, this paper proposes a simple method to improve a linear support vector machine (LSVM) performance for textual data classification. The rest of the paper is organized as follows. Section 2 briefly introduces the algorithms, which were used in the experiment. In Section 3, our method is outlined with evaluation thoroughly described in Section 4 together with a description of datasets, experimental settings, and results. Finally, Section 5 outlines the conclusions and sets guidelines for future work.

## 2 Relevant algorithms

This section describes the algorithms relevant to research presented in this paper: Support Vector Machines [8, 15] and Particle Swarm Optimization [11].

### 2.1 Support vector machines

The early foundations for support vector machines were introduced in [3, 8] and later extensively described in [43]. Basically, they attempt to find the best possible surface to separate positive and negative training samples in supervised manner. In this section, we focus on linear SVM [15], which is optimized for large-scale learning and, therefore, is used in this paper.

Given training vectors $x_i \in \mathbb{R}^n$, $i = 1, \ldots, l$, in two class, and a vector $y \in \mathbb{R}^l$ such that $y_i = \{1, -1\}$, a linear classifier generates a weight vector $w$ as the model. The decision function is

$$\operatorname{sgn}\bigl(w^{\mathrm{T}} x\bigr).$$

L2-regularized L1-loss support vector classifier (SVC) solves the following primal problem:

$$\min_w \frac{1}{2} w^{\mathrm{T}} w + C \sum_{i=1}^{l} \bigl(\max\bigl(0,\, 1 - y_i w^{\mathrm{T}} x_i\bigr)\bigr),$$

whereas L2-regularized L2-loss SVC solves the following primal problem:

$$\min_w \frac{1}{2} w^{\mathrm{T}} w + C \sum_{i=1}^{l} \bigl(\max\bigl(0,\, 1 - y_i w^{\mathrm{T}} x_i\bigr)\bigr)^2. \tag{1}$$

Their dual forms are:

$$\min_\alpha \frac{1}{2} \alpha^{\mathrm{T}} \bar{Q} \alpha - \mathrm{e}^{\mathrm{T}} \alpha,$$

$$\text{subject to} \quad 0 \leqslant \alpha_i \leqslant U,\ i = 1, \ldots, l,$$

where $e$ is the vector of all ones, $\bar{Q} = Q + D$, $D$ is a diagonal matrix, and $Q_{ij} = y_i y_j x_i^{\mathrm{T}} x_j$. For L1-loss SVC, $U = C$ and $D_{ii} = 0$ for all $i$. For L2-loss SVC, $U = \infty$ and $D_{ii} = 1/(2C)$ for all $i$.

L1-regularization generates a sparse solution $w$. L1-regularized L2-loss SVC solves the following primal problem:

$$\min_w \|w\|_1 + C \sum_{i=1}^{l} \bigl(\max\bigl(0,\, 1 - y_i w^{\mathrm{T}} x_i\bigr)\bigr)^2, \tag{2}$$

where $\|\cdot\|_1$ denotes the 1-norm, $C > 0$ is a penalty parameter; see [15].

### 2.2 Particle swarm optimization

Particle swarm optimization was introduced in [11]. Let $a_i(t)$ denote the position of particle $i$ in the search space at time step $t$; unless otherwise stated, $t$ denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position, i.e.

$$a_i(t+1) = a_i(t) + v_i(t+1) \tag{3}$$

with $a_i(0) \sim U(a_{\min}, a_{\max})$. Velocity vector reflects both the experiential knowledge of the particle and socially exchanged information from the particle's neighborhood.

For Global Best PSO, the velocity of particle $i$ is calculated as

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\bigl[b_{ij}(t) - a_{ij}(t)\bigr] + c_2 r_{2j}(t)\bigl[\hat{b}_j(t) - a_{ij}(t)\bigr], \tag{4}$$

where $v_{ij}(t)$ is the velocity of particle $i$ in dimension $j = 1, \ldots, n_a$ at time step $t$, $a_{ij}(t)$ is the position of particle $i$ in dimension $j$ at time step $t$, $c_1$ and $c_2$ are positive acceleration constants used to scale the contribution of the cognitive and social components, respectively, and $r_{1j}(t)$, $r_{2j}(t) \sim U(0,1)$ are random values in the range $[0,1]$, sampled from a uniform distribution. These random values introduce a stochastic element to the algorithm.

The personal best position, $b_i$, associated with particle $i$ is the best position the particle has visited since the first time step. Considering minimization problems, the personal best position at the next time step, $t+1$, is calculated as in [12].

$$b_i(t+1) = \begin{cases} b_i(t) & \text{if } f(a_i(t+1)) \geqslant f(b_i(t)), \\ a_i(t+1) & \text{if } f(a_i(t+1)) < f(b_i(t)), \end{cases} \tag{5}$$

where $f : \mathbb{R}^{n_a} \to \mathbb{R}$ is the fitness function. As with evolutionary algorithms, the fitness function measures how close the corresponding solution is to the optimum, i.e. the fitness function quantifies the performance or quality of a particle (or solution) [12]. The global best position, $\hat{b}(t)$, at time step $t$, is defined as

$$\hat{b}(t) \in \{b_0(t), \ldots, b_{n_s}(t)\}, \quad f(\hat{b}(t)) = \min\{f(b_0(t)), \ldots, f(b_{n_s}(t))\}, \tag{6}$$

where $n_s$ is the total number of particles in the swarm. $\hat{b}$ is the best position discovered by any of the particles so far – it is usually calculated as the best personal best position. The global best position can also be selected from the particles of the current swarm, in which case [12, 48]

$$\hat{b}(t) = \min\{f(x_0(t)), \ldots, f(a_{n_s}(t))\}. \tag{7}$$

## 3 The proposed method

The main goal of the proposed method (further as LSVM$^{\text{PSO}}$) is to select penalty (cost) parameter of the error term $C$ for linear SVM training to increase the accuracy of the method presented in [27]. The starting $C$ value is defined by using cross-validated grid-search over a predefined grid of possible $C$ values. Figure 1 and Algorithm 1 present a modified method, initially introduced in [27], with additional steps (particularly, step 3). LSVM$^{\text{PSO}}$ is depicted in the region bounded by a red rectangle. The dashed line in the figure represents steps for additional calculations executed before the concrete step of the main algorithm before it is joined to it.

List of parameters used in algorithms:

| | |
|---|---|
| $class_i$ | certain category of sentiment assigned to the text in dataset |
| $td_i$ | set of testing data subsets |
| $Subset_{size}$ | size of testing data subset is divided into |
| $D_{train}$ | set of training data |
| $D_{test}$ | set of testing data |
| $Train_{count}$ | count of text instances of the certain class should be selected from dataset; $Train_{count} = (1/k) \cdot Subset_{size} \cdot \lvert D_{train}\rvert / \lvert D_{test}\rvert$ |
| $k$ | number of different classes |

| | |
|---|---|
| $\lvert D_{train}\rvert$ | number of instances in $D_{train}$ |
| $\lvert D_{test}\rvert$ | number of instances in $D_{test}$ |
| $R_{\text{LSVM}}$ | set of LSVM results |
| $LSVM\,sent$ | class of sentiment |

Algorithm 2 contains a pseudo code of the proposed LSVM$^{\text{PSO}}$ method. The main idea of the method presented in [27] is based on the selection of the training data size subject to the subset of split testing data. Thus, the testing data is split into equal subsets, and the size of training data is calculated on the basis of the size of the first subset. This is done with intent that a smaller training dataset would significantly reduce time and computational effort to train the classifier, which would provide similar or slightly smaller accuracy. This approach is also extended to majority voting based ensemble (further it is referred to as CL$\{n\}$_LSVM$^{\text{PSO}}$, $n$ – number of classifiers), which will be shown to improve classification performance for LSVM$^{\text{PSO}}$ as well. However, it introduces additional challenges, such as a decision on the number of used classifiers (LSVM$^{\text{PSO}}$). The proposed method should be applicable for both binary and multi-class tasks. The algorithm and diagram of the proposed method are presented in Algorithm 3 and Fig. 2.

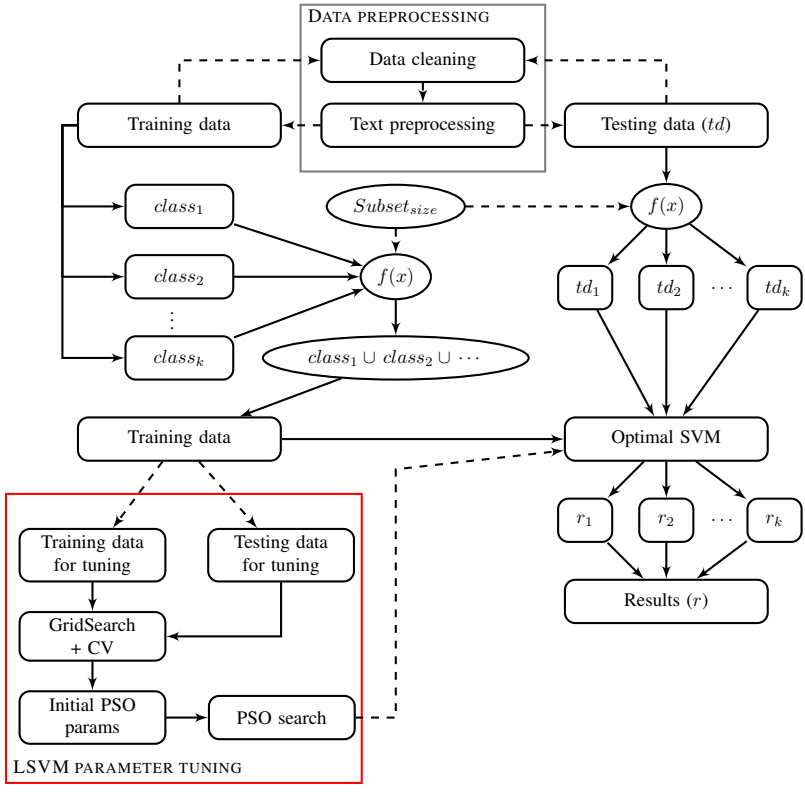


**Figure 1.** LSVM$^{\text{PSO}}$.

**Algorithm 1.** LSVM_$SpeedUP$

1. Run data preprocessing on Training and Testing data.
2. Randomly select data of each presented class in Training data.
$class_1 \leftarrow (\text{random.sample}(class_1, Train_{count}))$
$class_2 \leftarrow (\text{random.sample}(class_2, Train_{count}))$
$class_k \leftarrow (\text{random.sample}(class_k, Train_{count}))$
$D_{train} \leftarrow class_1 \cup class_2 \cup \cdots \cup class_k$
3. Split $D_{train}$ into Training data for tuning and Testing data for tuning.
$C \leftarrow \text{LSVM}^{\text{PSO}}$
4. Train LSVM($C$) with $D_{train}$
5. Split testing data in subsets and run on LSVM.
$k \leftarrow 0$
**for** $i = 1$: $\text{trunc}(\text{len}(D_{test})/Subset_{size})$ **do**
  $td_i \leftarrow D_{test}[(k+1) : (Subset_{size} * i),]$
  $LSVMsent_i \leftarrow \text{EVALUATELSVM}(td_i)$
  $\boldsymbol{R}_{\text{LSVM}} \leftarrow \boldsymbol{R}_{\text{LSVM}} \cup \{LSVMsent_i\}$
  $k \leftarrow (Subset_{size} * i)$
**end for**
**if** $\text{len}(D_{test} \,\%\, Subset_{size}) > 0$ **then**
  $td_{i+1} \leftarrow D_{test}[(k+1) : (\text{len}(D_{test})),]$
  $\boldsymbol{LSVMsent_{i+1}} \leftarrow \text{EVALUATELSVM}(td_{i+1})$
  $\boldsymbol{R}_{\text{LSVM}} \leftarrow \boldsymbol{R}_{\text{LSVM}} \cup \{LSVMsent_{i+1}\}$
**end if**
**Output** : $Results \leftarrow \boldsymbol{R}_{\text{LSVM}}$

**Algorithm. 2** $\text{LSVM}^{\text{PSO}}$

**Require:** a set of examples $\{\boldsymbol{x}, y\}$, $y \in \mathbb{Z}$
1. Find initial LSVM $C$ value using standard grid search based cross-validation procedure
$C_{Gs} \leftarrow \text{GRIDSEARCHCV}(Train\_data, param\_grid = (1, 2, \ldots, m))$
$acc_{Gbest}, C_{Gbest} \leftarrow \text{TRAINLSVM}(Train\_data, C_{Gs})$
2. Declare variables.
$C_{\min} \leftarrow C_{Gbest} - R$
$C_{\max} \leftarrow C_{Gbest} + R$
3. Initialize particles with population size $nPs$.
$\boldsymbol{C} \sim U(C_{\min}, C_{\max})$
$\boldsymbol{v} \leftarrow 0$
**for** $k = 1$: $nPs$ **do**
  $\boldsymbol{acc}[k] \leftarrow \text{EVALUATELSVM}(\boldsymbol{C}[k], Tuning\_data)$
**end for**
$\boldsymbol{acc_{best}} \leftarrow$ **acc**
$\boldsymbol{C_{best}} \leftarrow$ **C**
$acc_{Gbest} \leftarrow \max(acc_{Gbest}, \max(\boldsymbol{acc_{best}}))$
$C_{Gbest} \leftarrow \max(C_{Gbest}, \max(\boldsymbol{C_{best}}))$
4. Run PSO algorithm.
**for** $j = 1$: $nIteration$ **do**
  $\boldsymbol{r_1} \sim U(0, 1)$
  $\boldsymbol{r_2} \sim U(0, 1)$
  $\boldsymbol{v} \leftarrow w \times \boldsymbol{v} + c_1 \times \boldsymbol{r_1} \times (\boldsymbol{C_{best}} - \boldsymbol{C}) + c_2 \times \boldsymbol{r_2} \times (C_{Gbest} - \boldsymbol{C})$
  $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{v}$
  **for** $k = 1$: $nPs$ **do**
    **if** $\boldsymbol{C}[k] > 0$ **then**
      $\boldsymbol{acc}[k] \leftarrow \text{EVALUATELSVM}(\boldsymbol{C}[k], Tuning\_data)$
      $\boldsymbol{acc_{best}}[k] \leftarrow \max(\boldsymbol{acc_{best}}[k], \boldsymbol{acc}[k])$

$\boldsymbol{C_{best}}[k] \leftarrow \max(\boldsymbol{C_{best}}[k], \boldsymbol{C}[k])$
**end if**
**end for**
$\boldsymbol{acc_{best}} \leftarrow \boldsymbol{acc}$
$\boldsymbol{C_{best}} \leftarrow \boldsymbol{C}$
$acc_{Gbest} \leftarrow \max(acc_{Gbest}, \max(\boldsymbol{acc_{best}}))$
$C_{Gbest} \leftarrow \max(C_{Gbest}, \max(\boldsymbol{C_{best}}))$
$w \leftarrow w \times 0.99$
**end for**
**Output**: $C_{Gbest}$

List of parameters used in algorithms:

| | |
|---|---|
| $nPs$ | the number of particles in the swarm |
| $nIteration$ | number of iterations |
| $C$ | vector with LSVM cost parameters used in each PSO iteration |
| $R$ | the radius; (neighborhood) of $C_{Gs}$ where PSO search is performed |
| $C_{Gs}$ | vector with cost $C$ values obtained using grid search with cross validation |
| $acc$ | vector of accuracy obtained with LSVM procedure |
| $velocity$ | vector of the direction and magnitude of the particle movement |
| $C_{best}$ | vector with values of optimal $C$ values for each particle |
| $acc_{best}$ | vector with values of the best individual accuracy for each particle |
| $acc_C$ | vector with values of the optimal $C$ value for each particle |
| $acc_{Gbest}$ | best global accuracy |
| $C_{Gbest}$ | best global $C$ value |
| $c_1$, $c_2$ | acceleration coefficients |
| $r_1$, $r_2$ | random vectors |
| $w$ | coefficient of inertia |
| $param\_grid$ | grid of parameters to sequences of allowed values |

Further, we present the ensemble extension; here, the main parameters are as follows:

| | |
|---|---|
| $nLSVM$ | number of LSVM$^{\text{PSO}}$ classifiers obtained |
| $\mathcal{C}$ | number of classes |

**Algorithm 3.** Ensemble LSVM$^{\text{PSO}}$

**Require:** a set of training data $D_{train} = \{Train\_d_1, Train\_d_2, \ldots, Train\_d_k\}$
1. Train LSVM$^{\text{PSO}}$ ensemble.
**for** $i = 1$: $|D_{train}|$ **do**
LSVM$_i^{\text{PSO}} \leftarrow$ LSVM$^{\text{PSO}}$($Train\_d_i$)
**end for**
2. Run all LSVM$^{\text{PSO}}$ ensembles on testing data.
**for** $k = 1$: $nLSVM$ **do**
$d_k \leftarrow$ EVALUATELSVM(LSVM$_k^{\text{PSO}}$, $Testing\_data$)
**end for**
**Output** : $\arg\max_{z \in 1 \ldots \mathcal{C}} \sum_{i=1,\ldots,nLSVM} d_i = z$

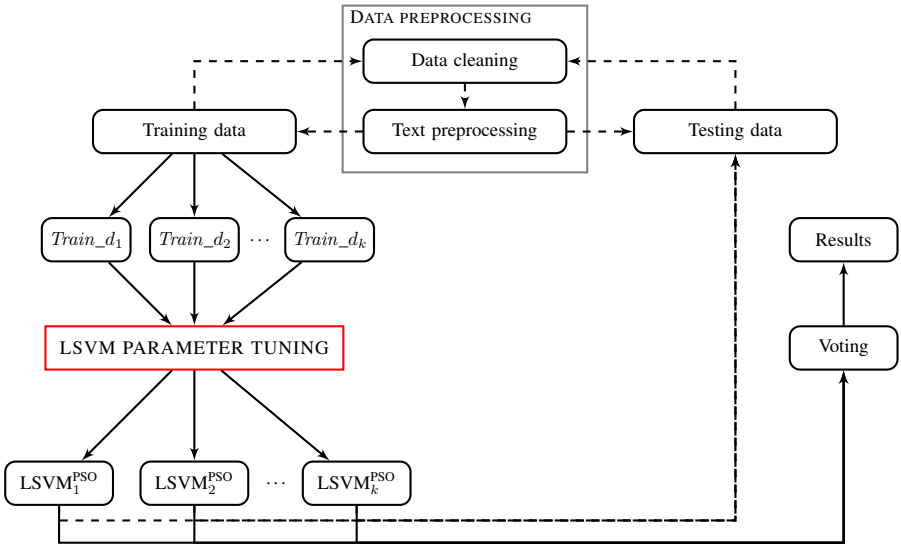


**Figure 2.** Ensemble $LSVM^{PSO}$.

# 4 Experiments and results

## 4.1 Dataset

To measure the performance of the introduced method it is evaluated on the largest labeled datasets available: the Stanford Twitter sentiment corpus dataset[1] introduced in [18], Amazon customer reviews dataset[2], and the Amazon[3] product data dataset introduced in [33]. We consider Books, Electronics, Kindle Store, Cell Phones, and Accessories datasets from the Amazon product data in particular. A brief description of the datasets is presented in Table 1.

Training and testing data has been preprocessed and cleaned before it was passed as the input of LSVM algorithm. It included removing redundant tokens, such as hashtag symbols `@`, numbers, "http" for links, punctuation symbols, etc. After cleaning was performed, all datasets were checked, and empty strings were removed.

**Table 1.** The description of datasets.

| Dataset | Num. of reviews | Num. of classes |
|---|---|---|
| Stanford Twitter sentiment corpus | 1,600,000 | 2 |
| Amazon customer reviews | 4,000,000 | 2 |
| Books | 22,507,155 | 5 |
| Electronics | 7,824,482 | 5 |
| Kindle Store | 3,205,467 | 5 |
| Cell Phones and Accessories | 3,447,249 | 5 |

[1] `http://help.sentiment140.com/`
[2] `https://www.kaggle.com/bittlingmayer/amazonreviews/`
[3] `http://jmcauley.ucsd.edu/data/amazon/`

### 4.2 Experiments

Two experiments are performed to evaluate our proposed method with our methods (Subset30K and kmLSVM) presented in earlier works [27, 28]. Ensembles of three (CL3_LSVM$^{PSO}$) and five (CL5_LSVM$^{PSO}$) classifiers, Stanford Twitter sentiment corpus, and Amazon customer reviews datasets are used in these experiments. During the experiments, the training data is randomly selected from the whole dataset, while the remaining part is used for testing. To obtain more objective results, 10 iterations of these experiments were performed, and the results were averaged.

Next, four experiments were done to compare the results with other authors' works. The datasets (Books, Electronics, Kindle Store, Cell Phones, and Accessories) used in [20, 37, 40] were selected. The descriptions of these datasets are presented in Table 1 (see Section 4.1). Although the Amazon reviews come in 5-star rating, the aforementioned authors used only two classes of the presented datasets. According to them, 3-star ratings are considered as neutral reviews meaning neither positive nor negative, hence instances with this class were discarded from datasets. The remaining classes were converted to binary as follows: reviews receiving 1- or 2-star ratings were labeled as "0", whereas reviews receiving 4 and 5 stars received a label "1". As in the first two experiments, the training data is randomly selected from the whole dataset, and the remaining part is used for testing. Additionally for Electronics, Cell Phones, and Accessories datasets, we perform experiments where linear SVM is used together with splitting dataset into training (70%) and testing (30%) subsets. It can also be observed that all experiments were performed 10 times to get more accurate results, and the average is taken as the final result.

Python programming language and scikit-learn [13] library for machine learning were used to implement and evaluate the proposed method. Linear SVC module, implemented in terms of LibLinear (a Library for Large Linear Classification[4]), was used to implement LSVM functionality. The data was converted to a matrix of TF-IDF (term frequency, inverse document frequency) features.

Table 2 shows the sizes of training and testing data for LSVM input. In experimental settings, it is assumed that the testing subset is 30%, therefore, the training data should

**Table 2.** Dataset splits in experiments.

| Dataset | Training data | Testing data |
|---|---|---|
| Stanford Twitter sentiment corpus | 70,000 | 480,000 |
| Amazon customer reviews | 70,000 | 1,200,000 |
| Books | 70,000 | 20,037,414 |
| Electronics | 70,000 | 7,117,716 |
| Kindle Store | 70,000 | 2,828,300 |
| Cell Phones and Accessories | 70,000 | 3,025,090 |
| Ordinary dataset splits for additional experiments | | |
| Electronics | 5,031,400 | 2,156,316 |
| Cell Phones and Accessories | 2,166,563 | 928,527 |

[4] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

be 70%, and subset size should be 30,000 instances (30%), then training data calculated dependently on subset size is 70,000 instances (70%). Then we should split all testing data into subsets containing 30,000 instances (the last subset is the remainder, and it could contain less than 30,000 instances) and run them separately one by one on LSVM. Parameter $R$, neighborhood for obtained $C_{Gs}$, where PSO search is performed, is set to 0.1.

For experiments, there is used computer with processor Intel(R) Core(TM)i7-4712MQ CPU @ 2.30 GHz and 16.00 GB installed memory (RAM).

### 4.3 Performance evaluation

Effectiveness is measured using statistical measures, which are used often for similar tasks, particularly, accuracy ($ACC$), precision (positive predictive value $PPV$ and negative predictive value $NPV$), recall (true positive rate $TPR$ and true negative rate $TNR$), harmonic mean of $PPV$ and $TPR$ ($F_1score$). Formulas are presented below [39]:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \qquad PPV = \frac{TP}{TP + FP},$$
$$NPV = \frac{TN}{TN + FN}, \qquad TPR = \frac{TP}{TP + FN},$$
$$TNR = \frac{TN}{TN + FP}, \qquad F_1score = \frac{2}{\frac{1}{PPV} + \frac{1}{TPR}},$$

where $TP$ – count of correctly classified "positive" sentiments, $TN$ – count of correctly classified "negative" sentiments. $FP$ – count of incorrectly classified "positive" sentiments. $FN$ – count of incorrectly classified "negative" sentiments. Area under the Receiver Operating Characteristics ($AUC$) is also used to measure the quality of the model predictions.

### 4.4 Results

All experiments described in Section 4.2 are performed, and the results are compared. Table 3 gives average results for the proposed method in comparison with Subset30K and kmLSVM when Stanford Twitter sentiment corpus and the Amazon customer reviews datasets were used. The distribution of results per iteration is visually depicted in Figs. 3 and 4.

The quality measure of the model's predictions ($AUC$) clearly shows that the proposed method and its ensembles outperform the previously proposed methods (Subset30K and kmLSVM) on both datasets: the Stanford Twitter sentiment corpus dataset and Amazon customer reviews dataset.

Other metrics in terms of – accuracy, $PPV$, $NPV$, $TPR$, $TNR$, $F_1score$ – also show the superiority of the proposed method compare with Subset30K on both datasets.

The results between LSVM$^{\text{PSO}}$ and kmLSVM were also insignificantly better in terms of accuracy, $PPV$, $TNR$ on the Stanford Twitter sentiment corpus dataset, while LSVM$^{\text{PSO}}$ lost slightly in terms of $NPV$, $TPR$, and $F_1score$. On the Amazon customer

**Table 3.** Results of the proposed method.

| Method | $ACC$ | $PPV$ | $NPV$ | $TPR$ | $TNR$ | $F_1 score$ | $AUC$ |
|---|---|---|---|---|---|---|---|
| | Stanford Twitter sentiment corpus dataset | | | | | | |
| Subset30K | 76.88% | 76.64% | 77.11% | 77.32% | 76.43% | 76.98% | 85.09% |
| kmLSVM | 77.81% | 77.07% | **78.59%** | **79.19%** | 76.44% | **78.12%** | 85.72% |
| LSVM$^{PSO}$ | **77.90%** | **77.46%** | 78.35% | 78.70% | **77.09%** | 78.08% | **85.75%** |
| CL3_LSVM$^{PSO}$ | **78.45%** | **77.95%** | **78.97%** | **79.36%** | **77.54%** | **78.64%** | **86.26%** |
| CL5_LSVM$^{PSO}$ | **78.63%** | **78.12%** | **79.16%** | **79.55%** | **77.72%** | **78.83%** | **86.33%** |
| | Amazon customer reviews | | | | | | |
| Subset30K | 87.61% | 87.69% | 87.53% | 87.50% | 87.71% | 87.59% | 95.07% |
| kmLSVM | 88.35% | **88.85%** | 87.85% | 87.70% | **89.00%** | 88.27% | 95.08% |
| LSVM$^{PSO}$ | **88.46%** | 88.63% | **88.28%** | **88.22%** | 88.69% | **88.43%** | **95.29%** |
| CL3_LSVM$^{PSO}$ | **88.98%** | **89.16%** | **88.80%** | **88.75%** | **89.21%** | **88.96%** | **95.60%** |
| CL5_LSVM$^{PSO}$ | **89.11%** | **89.31%** | **88.91%** | **88.85%** | **89.36%** | **89.08%** | **95.65%** |

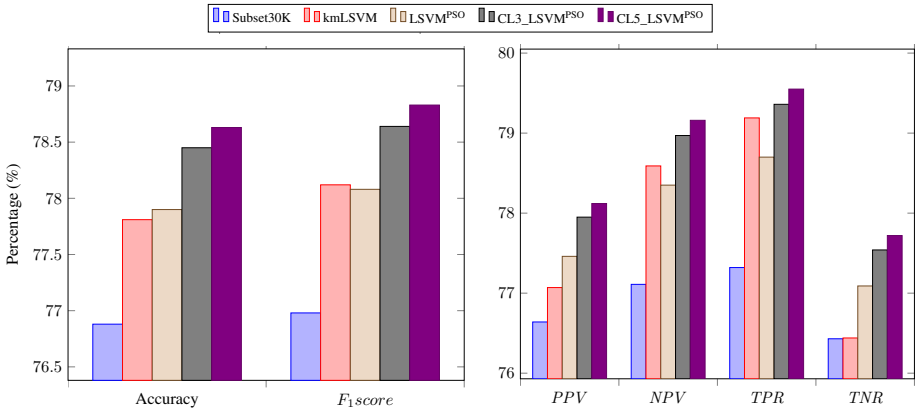


**Figure 3.** Stanford Twitter sentiment corpus dataset.

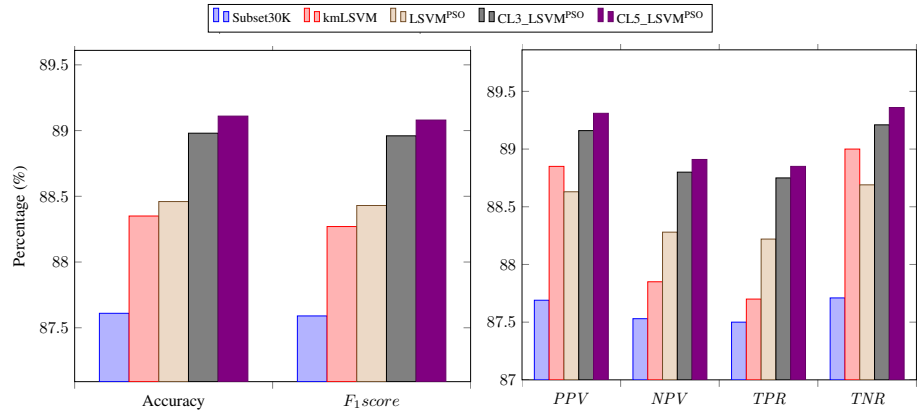


**Figure 4.** Amazon customer reviews dataset.

**Table 4.** Results comparison

| Authors | Dataset | ML method | Accuracy |
|---|---|---|---|
| Rain [37] (2013) | Books | Naïve Bayes | 84.50% |
| Shaikh and Deshpande [40] (2016) | | Naïve Bayes | 80.00% |
| Proposed method | | $\text{LSVM}^{\text{PSO}}$ | **89.50%** |
| | | $\text{CL3\_LSVM}^{\text{PSO}}$ | **89.86%** |
| Rain [37] (2013) | Kindle Store | Naïve Bayes | 87.33% |
| Proposed method | | $\text{LSVM}^{\text{PSO}}$ | **91.27%** |
| | | $\text{CL3\_LSVM}^{\text{PSO}}$ | **91.50%** |
| Haque et al. [20] (2018) | Electronics | LinearSVM | **93.52%** |
| Proposed method | | $\text{LSVM}^{\text{PSO}}$ | 90.14% |
| | | $\text{CL3\_LSVM}^{\text{PSO}}$ | 90.52% |
| | | Ordinary $\text{LSVM}^{\text{PSO}}$ | 93.17% |
| Haque et al. [20] (2018) | Cell Phones | LinearSVM | **93.57%** |
| Wang et al. [44] (2018) | and Accessories | Convolutional Neural Network (CNN-S(+)) | 85.90% |
| | | Contextual Factorization Machine (CFM) | 83.50% |
| | | Position-aware Factorization Machine (PFM) | 84.20% |
| Proposed method | | $\text{LSVM}^{\text{PSO}}$ | 90.57% |
| | | $\text{CL3\_LSVM}^{\text{PSO}}$ | 90.83% |
| | | Ordinary $\text{LSVM}^{\text{PSO}}$ | 93.22% |

review dataset, $\text{LSVM}^{\text{PSO}}$ performed slightly better in terms of accuracy, $NPV$, $TNR$, and $F_1 score$. In the case of $\text{LSVM}^{\text{PSO}}$ ensembles, the results are better in all metrics to compare with kmLSVM and single $\text{LSVM}^{\text{PSO}}$.

It is difficult to explicitly compare the results obtained with results in other papers due to discrepancy in implementations, parameters, or tasks. Therefore, Table 4 presents the results of comparative analysis based on accuracy when the proposed method is applied to the same datasets and contains the same number of classes. Dataset splits into training and testing data are different and demonstrate that sufficient accuracy can be obtained using a smaller training subset.

Table 4 shows that $\text{LSVM}^{\text{PSO}}$ and its ensemble of three $\text{CL3\_LSVM}^{\text{PSO}}$ resulted in higher accuracy compared to [37] and [40] when applied on the largest Books dataset and Kindle Store dataset [37]. The proposed method and its ensembles also outperform CNN-S(+), CFM, and PFM when they were applied on Cell Phones and Accessories dataset to compare with [44].

However, it performed slightly worse compared to [20] when higher accuracy with Electronics dataset and Cell Phones and Accessories dataset. The introduced method was also slightly outperformed by linear SVM, while testing with Electronics dataset and Cell Phones and Accessories dataset. Yet, this approach proved to be efficient to select the cost parameter $C$ for methods used in [27, 28], as well as ordinary linear SVM classifier, and allowed them to become competitive when compared to the works of other authors.

## 5 Conclusions and future work

This paper explores the application of particle swarm optimization metaheuristics to obtain optimal cost parameter $C$ for linear support vector machines. The main goal was to identify principles to increase accuracy of our methods presented in the previous papers [27, 28]. The results obtained with $\text{LSVM}^{\text{PSO}}$ are comparable with the performance of the aforementioned methods and resulted in improvements in all effectiveness metrics. Further, it is observed that ensembling results of multiple $\text{LSVM}^{\text{PSO}}$ classifiers, obtained with different subsets of dataset, resulted in improved accuracy, compared with a single classifier. It also shown that the proposed method can be applied separately on ordinary linear SVM.

The main advantage of the introduced method is that it can be easily applied on any linear SVM instance for textual data classification tasks on a large datasets and perform faster than ordinary linear SVM when it is used in combination with our method presented in [27]. In this paper, we found that using only 70,000 instances for training instead of more than 20 million (Books dataset) to develop classifier still resulted in performance comparable to [37, 40, 44], and the results obtained are also competitive with state-of-the-art models.

There are several directions to work on the proposed method. First, additional testing would be required to optimize the proposed method for practical applications. This requires a test for an optimal number of classifiers in the ensemble method in terms of trade-off for both performance and training/testing time, optimal subset size, as well as to optimize the implemented particle swarm optimization method for faster convergence to optimal results. Further, it will be tested with multiclass classification tasks, the functionality of built-in SVM implementation.

## References

1. M. Ahmad, S. Aftab, M.S. Bashir, N. Hameed, I. Ali, Z. Nawaz, SVM optimization for sentiment analysis, *Int. J. Adv. Comput. Sci. Appl*, **9**(4), 2018.
2. M. Ahmad, S. Aftab, S.S. Muhammad, S. Ahmad, Machine learning techniques for sentiment analysis: A review, *Int. J. Multidiscip. Sci. Eng*, **8**(3):27–32, 2017.
3. B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, 1992, pp. 144–152.
4. C. Catal, M. Nangir, A sentiment classification model based on multiple classifiers, *Appl. Soft Comput.*, **50**:135–141, 2017.
5. C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, *ACM Trans. Intel. Syst. Technol.*, **2**(3):27, 2011.
6. J.S. Chou, M.Y. Cheng, Y.W. Wu, A.D. Pham, Optimizing parameters of support vector machine using fast messy genetic algorithm for dispute classification, *Expert Syst. Appl.*, **41**(8): 3955–3964, 2014.

7. F. Colas, P. Brazdil, Comparison of SVM and some older classification algorithms in text classification tasks, in *IFIP International Conference on Artificial Intelligence in Theory and Practice*, Springer, 2006, pp. 169–178.
8. C. Cortes, V. Vapnik, Support-vector networks, *Mach. Learn.*, **20**(3):273–297, 1995.
9. P. Danenas, G. Garsva, Selection of support vector machines based classifiers for credit risk domain, *Expert Syst. Appl.*, **42**(6):3194–3204, 2015.
10. F. Deng, S. Guo, R. Zhou, J. Chen, Sensor multifault diagnosis with improved support vector machines, *IEEE Trans. Autom. Sci. Eng.*, **14**(2):1053–1063, 2017.
11. R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995*, IEEE, 1995, pp. 39–43.
12. A.P. Engelbrecht, *Computational Intelligence: An Introduction*, Wiley, 2008.
13. F. Pedregosa et al., Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.*, **12**(Oct): 2825–2830, 2011.
14. R. Morisi et al., Multi-class parkinsonian disorders classification with quantitative MR markers and graph-based features using support vector machines, *Parkinsonism Relat. D.*, **47**:64–70, 2018.
15. R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin, LIBLINEAR: A library for large linear classification, *J. Mach. Learn. Res.*, **9**(Aug):1871–1874, 2008.
16. G. Fei, B. Liu, Social media text classification under negative covariate shift, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2015, pp. 2347–2356.
17. G. Garšva, P. Danenas, Particle swarm optimization for linear support vector machines based classifier selection, *Nonlinear Anal. Model. Control*, **19**(1):26–42, 2014.
18. A. Go, R. Bhayani, L. Huang, Twitter sentiment classification using distant supervision, CS224N Project Report 12, Stanford University, Stanford, 2009.
19. X. Gu, T. Li, Y. Wang, L. Zhang, Y. Wang, J. Yao, Traffic fatalities prediction using support vector machine with hybrid particle swarm optimization, *J. Algorithms Comput. Technol.*, **12**(1): 20–29, 2018.
20. T.U. Haque, N.N. Saber, F.M. Shah, Sentiment analysis on large scale Amazon product reviews, in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, IEEE, 2018, pp. 1–6.
21. T.T. Hoang, M.Y. Cho, M.N. Alam, Q.T. Vu, A novel differential particle swarm optimization for parameter selection of support vector machines for monitoring metal-oxide surge arrester conditions, *Swarm Evol. Comput.*, **38**:120–126, 2018.
22. C.L. Huang, ACO-based hybrid classification system with feature subset selection and model parameters optimization, *Neurocomputing*, **73**(1-3):438–448, 2009.
23. C.L. Huang, J.F. Dun, A distributed PSO–SVM hybrid system with feature selection and parameter optimization, *Appl. Soft Comput.*, **8**(4):1381–1391, 2008.
24. İ. İlhan, G. Tezel, A genetic algorithm–support vector machine method with parameter optimization for selecting the tag SNPs, *J. Biomed. Inf.*, **46**(2):328–340, 2013.

25. D.K. Jain, S.B. Dubey, R.K. Choubey, A. Sinhal, S.K. Arjaria, A. Jain, H. Wang, An approach for hyperspectral image classification by optimizing SVM using self organizing map, *J. Comput. Sci.*, **25**:252–259, 2018.

26. T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in *European Conference on Machine Learning*, Springer, 1998, pp. 137–142.

27. K. Korovkinas, P. Danėnas, G. Garšva, SVM accuracy and training speed trade-off in sentiment analysis tasks, in *International Conference on Information and Software Technologies*, Springer, 2018, pp. 227–239.

28. K. Korovkinas, P. Danėnas, G. Garšva, SVM and $k$-means hybrid method for textual data sentiment analysis, *Baltic Journal of Modern Computing*, **7**(1):47–60, 2019.

29. S.W. Lin, K.C. Ying, S.C. Chen, Z.J. Lee, Particle swarm optimization for parameter determination and feature selection of support vector machines, *Expert Syst. Appl.*, **35**(4):1817–1824, 2008.

30. S. Liu, I. Lee, email sentiment analysis through $k$-means labeling and support vector machine classification, *Cybern. Syst.*, **49**(3):181–199, 2018.

31. Y. Maali, A. Al-Jumaily, A novel partially connected cooperative parallel PSO-SVM algorithm: Study based on sleep apnea detection, in *2012 IEEE Congress on Evolutionary Computation*, IEEE, 2012, pp. 1–8.

32. R. Manikandan, R. Sivakumar, Machine learning algorithms for text-documents classification: A review, *Mach. Learn.*, **3**(2), 2018.

33. J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel, Image-based recommendations on styles and substitutes, in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2015, pp. 43–52.

34. A. Onan, S. Korukoğlu, H. Bulut, A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification, *Expert Syst. Appl.*, **62**:1–16, 2016.

35. I. Perikos, I. Hatzilygeroudis, Recognizing emotions in text using ensemble of classifiers, *Eng. Appl. Artif. Intel.*, **51**:191–201, 2016.

36. Q. Qian, H. Gao, B. Wang, A SVM method trained by improved particle swarm optimization for image classification, in *Chinese Conference on Pattern Recognition*, Springer, Berlin, Heidelberg, 2014, pp. 263–272.

37. C. Rain, Sentiment analysis in Amazon reviews using probabilistic machine learning, Swarthmore College, 2013.

38. R. Ren, D.D. Wu, T. Liu, Forecasting stock market movement direction using sentiment analysis and support vector machine, *IEEE Syst. J.*, 2018.

39. C. Sammut, G.I. Webb, *Encyclopedia of Machine Learning*, Springer, Boston, MA, 2011.

40. T. Shaikh, D. Deshpande, Feature selection methods in sentiment analysis and sentiment classification of Amazon product reviews, *International Journal of Computer Trends and Technology (IJCTT)*, **36**(4), 2016.

41. I. Steinwart, A. Christmann, *Support Vector Machines*, Springer, New York, 2008.

42. Z.A. Sunkad et al., Feature selection and hyperparameter optimization of SVM for human activity recognition, in *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, IEEE, 2016, pp. 104–109.

43. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 2000.
44. S. Wang, M. Zhou, G. Fei, Y. Chang, B. Liu, Contextual and position-aware factorization machines for sentiment classification, 2018, arXiv:1801.06172.
45. C. Yongqi, LS_SVM parameters selection based on hybrid complex particle swarm optimization, *Energy Procedia*, **17**:706–710, 2012.
46. X. Zhang, X. Chen, Z. He, An ACO-based algorithm for parameter optimization of support vector machines, *Expert Syst. Appl.*, **37**(9):6618–6628, 2010.
47. X. Zhang, D. Qiu, F. Chen, Support vector machine with parameter optimization by a novel hybrid method and its application to fault diagnosis, *Neurocomputing*, **149**:641–651, 2015.
48. H. Zhenya, W. Chengjian, Y. Luxi, G. Xiqi, Y. Susu, R.C. Eberhart, Y. Shi, Extracting rules from fuzzy neural network by particle swarm optimisation, in *The 1998 IEEE International Conference on Evolutionary Computation*, IEEE, 1998, pp. 74–77.