Scientific Research Publishing

Journal of Computer and Communications, 2022, 10, 1-15
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Fast Algorithm for Training Large Scale Support Vector Machines

Mayowa Kassim Aregbesola, Igor Griva

George Mason University, Fairfax, USA
Email: maregbes@gmu.edu, igriva@gmu.edu

How to cite this paper: Aregbesola, M.K. and Griva, I. (2022) A Fast Algorithm for Training Large Scale Support Vector Machines. *Journal of Computer and Communications*, **10**, 1-15.
https://doi.org/10.4236/jcc.2022.1012001

Received: October 12, 2022
Accepted: December 13, 2022
Published: December 16, 2022

Abstract

The manuscript presents an augmented Lagrangian—fast projected gradient method (ALFPGM) with an improved scheme of working set selection, *pWSS*, a decomposition based algorithm for training support vector classification machines (SVM). The manuscript describes the ALFPGM algorithm, provides numerical results for training SVM on large data sets, and compares the training times of ALFPGM and Sequential Minimal Minimization algorithms (SMO) from Scikit-learn library. The numerical results demonstrate that ALFPGM with the improved working selection scheme is capable of training SVM with tens of thousands of training examples in a fraction of the training time of some widely adopted SVM tools.

Keywords

SVM, Machine Learning, Support Vector Machines, FISTA, Fast Projected Gradient, Augmented Lagrangian, Working Set Selection, Decomposition

1. Introduction

A support vector machine (SVM) [1] [2] is a supervised machine learning technique used for classification and regression [3]. SVM were initially designed for binary classification and have been expanded to multiclass classification that can be implemented by combining multiple binary classifiers using the pairwise coupling or one class against the rest methods [4]. The main advantage of the support vector machines is in their ability to achieve accurate generalization and their foundation on well developed learning theory [2].

Training dual soft margin SVM requires solving a large scale convex quadratic problem with a linear constraint and simple bounds for variables. To solve large scale SVM problems, decomposition methods such as sequential minimal optimization (SMO) [5] gained popularity due to their efficiency and low memory usage requirement. Each iteration the SMO keeps all the variables except two of them fixed and solves a small two-dimensional subproblem. In an effort to investigate a performance of decomposition based algorithms with larger working sets, one of the previous manuscripts [6] presented an augmented Lagrangian fast projected gradient method (ALFPGM) with working set selection for finding the solution to the SVM problem. The ALFPGM used larger working sets and trained SVM with a size up to 19020 training examples. The current study extends the previous results by presenting an updated working selection method $pWSS$ and testing the improved algorithm on larger data sets with up to tens of thousands of training examples.

The paper is organized as follows. The next section describes the SVM training problem, the ALFPGM and SMO algorithms. Section 3 describes a new working set selection principle responsible for an efficient implementation of ALFPGM. Section 4 provides some details of the calculations. Section 5 provides numerical results. Section 6 contains concluding remarks and Section 7 discusses some future directions of further improving the efficiency of the developed algorithm.

2. SVM Problem

To train the SVM, one needs to find $\boldsymbol{\alpha}^* = \left(\alpha_1^*, \cdots, \alpha_m^*\right)^{\mathrm{T}}$ that minimizes the following objective problem:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & -\sum_{i=1}^{m} \alpha_i + \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{K}\left(\mathbf{x}_i, \mathbf{x}_j\right) \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, i = 1, \cdots, m. \end{aligned} \tag{1}$$

Here $\left\{\left(\mathbf{x}_1, y_1\right), \cdots, \left(\mathbf{x}_m, y_m\right)\right\}$ is a set of m labeled data points where $\mathbf{x}_i \in \mathbb{R}^n$ is a vector of features and $y_i \in \{-1, 1\}$ represent the label indicating the class to which $\mathbf{x}_i$ belongs. The Matrix Q with the elements $q_{ij} = y_i y_j \mathbf{K}\left(\mathbf{x}_i, \mathbf{x}_i\right)$ is positive semi-definite but usually dense.

The sequential minimal optimization (SMO) algorithm developed by Platt [7] is one of the most efficient and widely used algorithms to solve (1). It is a low memory usage algorithm that iteratively solves two-dimensional QP subproblems. The QP subproblems are solved analytically. Many popular SVM solvers are based on SMO.

There have been several attempts to further speed up the SMO [8] [9] [10]. In particular, there have been attempts on parallelizing SMO [10]. However, selection of the working set in SMO depends on the violating pairs, which are constantly changed each iteration, making it challenging to develop a parallelized algorithm for the SMO [6]. Other attempts have focused on parallelizing selections of larger working sets of more than one pair. The approach presented here follows the latter path.

2.1. SVM Decomposition

Matrix Q is dense and it is inefficient and often memory prohibitive to populate and store Q in the operating memory for large training data. Therefore decomposition methods that consider only a small subset of variables per iteration [11] [12] are used to train an SVM with a large number of training examples.

Let B be the subset of selected l variables called the working set. Since each iteration involves only l rows and columns of the matrix Q, the decomposition methods use the operating memory economically [11].

The algorithm repeats the *select working set then optimize* process until the global optimality conditions are satisfied. While B denotes the working set with l variables, N denotes the non-working set with $(m-l)$ variables. Then, $\boldsymbol{\alpha}$, y and Q can be correspondingly written as:

$$\boldsymbol{\alpha}=\begin{bmatrix}\boldsymbol{\alpha}_B\\ \boldsymbol{\alpha}_N\end{bmatrix},\quad \mathbf{y}=\begin{bmatrix}\mathbf{y}_B\\ \mathbf{y}_N\end{bmatrix},\quad Q=\begin{bmatrix}\mathbf{Q}_{BB} & \mathbf{Q}_{BN}\\ \mathbf{Q}_{NB} & \mathbf{Q}_{NN}\end{bmatrix}.$$

The optimization subproblem can be rewritten as

$$\begin{aligned}\underset{\alpha_B}{\text{minimize}}\quad & \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_{\mathrm{B}}^{\mathrm{T}} & \boldsymbol{\alpha}_{\mathrm{N}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\mathbf{Q}_{BB} & \mathbf{Q}_{BN}\\ \mathbf{Q}_{NB} & \mathbf{Q}_{NN}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{\mathrm{B}}\\ \boldsymbol{\alpha}_{\mathrm{N}}\end{bmatrix}-\begin{bmatrix}\mathbf{e}_{\mathrm{B}}^{\mathrm{T}} & \mathbf{e}_{\mathrm{N}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{\mathrm{B}}\\ \boldsymbol{\alpha}_{\mathrm{N}}\end{bmatrix}\\ \text{subject to}\quad & \boldsymbol{\alpha}_{\mathrm{B}}^{\mathrm{T}}y_{\mathrm{B}}+\boldsymbol{\alpha}_{\mathrm{N}}^{\mathrm{T}}y_N=0\\ & 0\le\boldsymbol{\alpha}_{\mathrm{B}}\le C.\end{aligned}\tag{2}$$

with a fixed $\boldsymbol{\alpha}_{\mathrm{N}}$. The problem (2) can be rewritten as

$$\begin{aligned}\underset{\alpha_B}{\text{minimize}}\quad & \frac{1}{2}\boldsymbol{\alpha}_{\mathrm{B}}^{\mathrm{T}}\mathbf{Q}_{\mathrm{BB}}\boldsymbol{\alpha}_{\mathrm{B}}+\boldsymbol{\alpha}_{\mathrm{B}}^{\mathrm{T}}\chi\\ \text{subject to}\quad & \boldsymbol{\alpha}_{\mathrm{B}}^{\mathrm{T}}y_{\mathrm{B}}+G_k=0\\ & 0\le\boldsymbol{\alpha}_{\mathrm{B}}\le C.\end{aligned}\tag{3}$$

where $\chi=\mathbf{Q}_{\mathrm{BN}}\boldsymbol{\alpha}_N-\mathbf{e}_B=q-\mathbf{e}_B$ and $G_k=\boldsymbol{\alpha}_N^{\mathrm{T}}y_N$.

The reduced problem (3) can be solved much faster than the original problem (1). The resulting algorithm is shown as Algorithm 1.

Algorithm 1 Decomposition method

```
1: while global minimum not reached do
2:     find working set $B$
3:     find $\boldsymbol{\alpha}_B$ for working set $B$ using a quadratic problem solver.
4:     Update $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}_B$
5: end while
```

2.2. Augmented Lagrangian—Fast Projected Gradient Method for SVM

A relatively simple and efficient algorithm capable of training medium size SVM with up to a few tens of thousands of data points was proposed in [13]. The algorithm takes advantage of two methods: one is a fast projected gradient method for solving a minimization problem with simple bounds on variables [14] and the second is an augmented Lagrangian method [15] [16] employed to satisfy the only linear equality constraint. The method projects the primal variables onto the "box-like" set: $0 \le \alpha_i \le C, i \in \mathrm{B}$.

Using the following definitions

$$\begin{aligned} f(\boldsymbol{\alpha}) &= \sum_{i\in \mathrm{B}} \alpha_i (q_i - 1) + \frac{1}{2} \sum_{i,j\in \mathrm{B}} \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \\ g(\boldsymbol{\alpha}) &= \sum_{i\in \mathrm{B}} y_i \alpha_i + G_k \end{aligned} \tag{4}$$

and the bounded set:

$$Box = \{\boldsymbol{\alpha} \in \mathbb{R}^l : 0 \le \alpha_i \le C, i \in \mathrm{B}\}, \tag{5}$$

the optimization problem (3) can be rewritten as follows:

$$\begin{aligned} &\underset{\alpha}{\text{minimize}} \;\; f(\boldsymbol{\alpha}) \\ &\text{subject to} \;\; g(\boldsymbol{\alpha}) = 0 \\ &\qquad\qquad \boldsymbol{\alpha} \in Box \end{aligned} \tag{6}$$

The augmented Lagrangian is defined as

$$\mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda) = f(\boldsymbol{\alpha}) - \lambda g(\boldsymbol{\alpha}) + \frac{\mu g(\boldsymbol{\alpha})^2}{2}, \tag{7}$$

where $\lambda \in \mathbb{R}$ the Lagrange multiplier that corresponds to the only equality constraint and $\mu > 0$ is the scaling parameter.

The augmented Lagrangian method is a sequence of inexact minimizations of $\mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda)$ in $\boldsymbol{\alpha}$ on the Box set:

$$\hat{\boldsymbol{\alpha}} \approx \boldsymbol{\alpha}(\lambda) = \underset{\alpha\in Box}{\arg\min} \, \mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda) \tag{8}$$

followed by updating the Lagrange multiplier λ:

$$\hat{\lambda} = \lambda - \mu g(\hat{\boldsymbol{\alpha}}). \tag{9}$$

The stopping criteria for (8) uses the following function that measures the violation of the first order optimality conditions of (8):

$$\nu(\boldsymbol{\alpha}, \lambda) = \max_{1\le i\le m} \nu_i(\boldsymbol{\alpha}, \lambda), \tag{10}$$

where

$$\nu_i(\boldsymbol{\alpha}, \lambda) = \begin{cases} \left|\nabla_\alpha \mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda)_i\right| & 0 < \alpha_i < C \\ \max\{0, -\nabla_\alpha \mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda)_i\} & \alpha_i = 0 \\ \max\{0, \nabla_\alpha \mathcal{L}_\mu(\boldsymbol{\alpha}, \lambda)_i\} & \alpha_i = C \end{cases} \tag{11}$$

The minimization (8) (the inner loop) is performed by the fast projected gradient method (FPGM). The inner loop exits when $\nu(\boldsymbol{\alpha}, \lambda) < \varepsilon$. The final stopping criteria for the augmented Lagrangian method uses $\mu(\boldsymbol{\alpha}, \lambda) = \max\{\nu(\boldsymbol{\alpha}, \lambda), |g(\boldsymbol{\alpha})|\}$, which measures the violation of the optimality conditions for (6).

Algorithm 2 describes the ALFPGM (see [6] for more detail). The convergence of Algorithm 2 is established in [17].

Algorithm 2 Augmented Lagrangian-Fast Projected Gradient Method

```
Initialization
 1: $\alpha_0 \in \mathbb{R}^m$ (starting point, not necessarily feasible) $\alpha_0 = 0$
 2: $\lambda_0 \in \mathbb{R}$ (initial "guestimate" of Lagrange multiplier vector) $\lambda_0 = 0$
 3: $\mu_0 > 0$ (initial value of scaling parameter)
 4: $\epsilon > 0$ (Required accuracy)
 5: $0 < \theta < 1$
 6: rec $= \max\{\nu(\alpha, \lambda), |g(\alpha)|\}$
Run the Augmented Lagragian Algorithm
 1: while $rec > \epsilon$ do
 2:     find $\alpha : \nu(\alpha, \lambda) \leq \theta * rec$ using FPGM
 3:     $\lambda = \lambda - \mu g(\alpha)$
 4:     $rec = \min(rec, \nu(\alpha, \lambda))$
 5: end while
Run the FPGM
 1: procedure FPGM
 2:     Input $(\alpha_0, \lambda)$
 3:     Set $k := 0$, $\bar{\alpha}_k = \alpha_k, t_k = 1$. Select $L > 0$
 4:     while $\nu(\alpha_k, \lambda) > \theta * rec$ do
 5:         $\alpha_{k+1} := P_{Box}\left(\bar{\boldsymbol{\alpha}}_k - \frac{1}{L}\nabla_\alpha \mathcal{L}_\mu(\bar{\boldsymbol{\alpha}}_k, \lambda)\right)$.
 6:         $t_{k+1} := \frac{1}{2} + \frac{\sqrt{1+4t_k^2}}{2}$
 7:         if $(\bar{\boldsymbol{\alpha}}_k - \boldsymbol{\alpha}_{k+1})^T(\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k) \geq 0$ then
 8:             $\bar{\boldsymbol{\alpha}}_{k+1} = \boldsymbol{\alpha}_{k+1}$.
 9:         else
10:             $\bar{\boldsymbol{\alpha}}_{k+1} := \boldsymbol{\alpha}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k)$
11:         end if
12:         $k := k + 1$
13:     end while
14:     Output $\boldsymbol{\alpha}_k$.
15: end procedure
 1: procedure $P_{Box}$
 2:     for $i = 1, \ldots, m$ do
 3:         if $\alpha_i < 0$ then
 4:             $\alpha_i = 0$
 5:         end if
 6:         if $\alpha_i > C$ then
 7:             $\alpha_i = C$
 8:         end if
 9:     end for
10: end procedure
```

3. *PWSS* Working Set Selection

The flow chart describing *pWSS* is shown in **Figure 1**. An important issue of any decomposition method is how to select the working set *B*. First, a working set selection (WSS) method WSS_{2nd} that uses the second order information [18] is described.

Consider the sets:

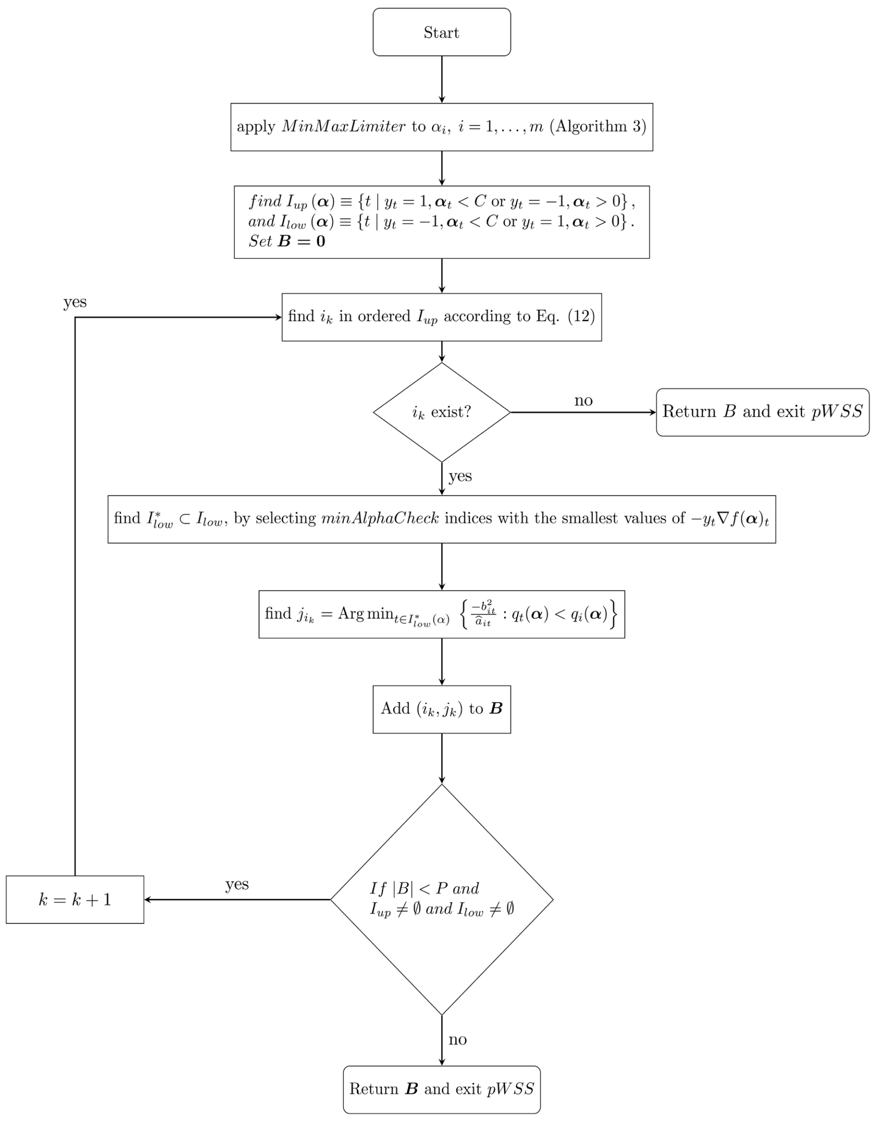
$$I_{up}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = 1, \boldsymbol{\alpha}_t < C \text{ or } y_t = -1, \boldsymbol{\alpha}_t > 0\},$$
$$I_{low}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = -1, \boldsymbol{\alpha}_t < C \text{ or } y_t = 1, \boldsymbol{\alpha}_t > 0\}.$$

Then let us define

$$q_t(\boldsymbol{\alpha}) = -y_t \nabla f(\boldsymbol{\alpha})_t.$$

In WSS_{2nd}, one selects

$$i \in \arg\max_{t \in I_{up}(\boldsymbol{\alpha})} \{q_t(\boldsymbol{\alpha})\}, \tag{14}$$



Start
apply $MinMaxLimiter$ to α_i, $i = 1, \dots, m$ (Algorithm 3)
$find\ I_{up}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = 1, \boldsymbol{\alpha}_t < C \text{ or } y_t = -1, \boldsymbol{\alpha}_t > 0\}$, $and\ I_{low}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = -1, \boldsymbol{\alpha}_t < C \text{ or } y_t = 1, \boldsymbol{\alpha}_t > 0\}$. $Set\ \boldsymbol{B} = \boldsymbol{0}$
find i_k in ordered I_{up} according to Eq. (12)
i_k exist?
no
Return B and exit $pWSS$
yes
find $I^*_{low} \subset I_{low}$, by selecting $minAlphaCheck$ indices with the smallest values of $-y_t \nabla f(\boldsymbol{\alpha})_t$
find $j_{i_k} = \operatorname{Arg\,min}_{t \in I^*_{low}(\alpha)} \left\{ \frac{-b_{it}^2}{\hat{a}_{it}} : q_t(\boldsymbol{\alpha}) < q_i(\boldsymbol{\alpha}) \right\}$
Add (i_k, j_k) to $\boldsymbol{B}$
$If\ |B| < P\ and$ $I_{up} \neq \emptyset\ and\ I_{low} \neq \emptyset$
yes
$k = k + 1$
yes
no
Return $\boldsymbol{B}$ and exit $pWSS$

Figure 1. $pWSS$ Working set selection.

and

$$j \in \underset{t \in I_{low}(\boldsymbol{\alpha})}{\arg\min} \left\{ \frac{-b_{it}^2}{\hat{a}_{it}} : q_t(\boldsymbol{\alpha}) < q_i(\boldsymbol{\alpha}) \right\}. \tag{15}$$

with

$$a_{ij} = K_{ii} + K_{jj} - 2K_{ij} > 0,$$
$$\hat{a}_{ij} = \begin{cases} a_{ij}, \text{ if } a_{ij} > 0, \\ \tau \end{cases}$$
$$b_{ij} = q_i(\alpha) - q_j(\alpha) > 0,$$

where τ is a small positive number.

3.1. Limiting the Search Space for I_{up} and I_{low}

One of the challenges of the previously developed dual decomposition WSS_{2nd}

is that a particular data set point can be selected multiple times in different iterations of decomposition in the working set, often increasing the SVM training time. The previously developed algorithm [6] is improved, and the changes to *pWSS* are made by limiting the search space of I_{up} and I_{low} as described below.

3.2. *MinMaxLimiter* Algorithm

The *MinMaxLimiter* algorithm with the following changes to *pWSS* is proposed:

• Some elements of $\boldsymbol{\alpha}$ that no longer change after many iterations, are eliminated: $\left\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\right\| < \delta_{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha}_{t-1}$, $\boldsymbol{\alpha}_t \in B$ are two consecutive iterates, and $\delta_{\boldsymbol{\alpha}}$ is a user defined threshold.

• Another introduced parameter is *minAlphaOpt*, which is the minimum number of times a data index is used in working set. After this threshold, if the value of $\boldsymbol{\alpha}_i, i = 1, \cdots, m$ is 0 or *C*, then that data point index is no longer considered

• The last introduced parameter is *maxAlphaOpt*, which is the maximum number of times a training examples is used in the decomposition rounds.

$minAlphaOpt = \infty$ and $maxAlphaOpt = \infty$ will result in considering every $\boldsymbol{\alpha}$ in every iteration.

Algorithm 3 $MinMaxLimiter$

```
1: $p$ = number of pairs
2: $B_J = \emptyset$
3: if $(\alpha_t - \alpha_{t-1} > \delta_\alpha)$ & $(IterCount(\alpha_t) < maxAlphaOpt)$ then
4:     if $(IterCount(\alpha_t) < minAlphaOpt)$ then
5:         $I_{up}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = 1, \alpha_t < C \text{ or } y_t = -1, \alpha_t > 0\}$.
6:         $I_{low}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = -1, \alpha_t < C \text{ or } y_t = 1, \alpha_t > 0\}$..
7:         $\alpha_{t-1}$ is the previous iteration of $\alpha$.
8:     end if
9: end if
```

Algorithm 4 $WSS_{minAlphaCheck}$

```
1: $p$ = number of pairs
2: $B_J = \emptyset$
3: $I_{up}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = 1, \alpha_t < C \text{ or } y_t = -1, \alpha_t > 0\}$.
4: $I_{low}(\boldsymbol{\alpha}) \equiv \{t \mid y_t = -1, \alpha_t < C \text{ or } y_t = 1, \alpha_t > 0\}$.
5: for $k = 0, 1, 2, \ldots, p$ do
6:     find $i_k = \text{Arg}\max_{t \in I_{up}(\boldsymbol{\alpha})} q_t(\boldsymbol{\alpha})$
7:     find $I^*_{low} \subset I_{low}$, by selecting $minAlphaCheck$ indices with the smallest values of $-y_t \nabla.f(\boldsymbol{\alpha})_t$
8:     find $j_{i_k} = \text{Arg}\min_{t \in I^*_{low}(\alpha)} \left\{ \frac{-b^2_{it}}{\bar{a}_{it}} : q_t(\boldsymbol{\alpha}) < q_i(\boldsymbol{\alpha}) \right\}$.
9:     if $j_{i_k} \neq \emptyset$ then
10:        $B = B \cup i_k \cup j_{i_k}$
11:    end if
12: end for
```

3.3. Optimizing *j* Search Using *MinAlphaCheck*

Another modification to WSS_{2nd} is the introduction of parameter *minAlphaCheck*. This is used to reduce the search space of *j* in (13) to the first *minAlphaCheck* in the sorted I_{low} set. The experimental results show that this converges and is faster than WSS_{2nd}.

4. Implementation

4.1. Kernel Matrix Computation

The cache technique is used to store previously computed rows to access them when the same kernel value needs to be recomputed. The amount of cached kernel is a selectable parameter. The same **L**east **R**ecently **U**sed (LRU) LRU strategy that SMO implemented in Scikit-learn (Sklearn) was used, and it is a good basis for comparing the results obtained using the SVM training methods. Previous discussions [8] [10] have been held on the ineffectiveness of the LRU strategy. However, that is beyond the scope of this work.

To compute the kernel matrix, Intel MKL is used. The kernel matrix can be computed efficiently using $\|x_i\|^2$ and the elements of matrix $\boldsymbol{XX'}$, where $\boldsymbol{X}$ is the matrix with the rows made of the data vectors x_i. Intel MKL's cblas_dgemv is used to compute $f = \frac{1}{2}\alpha^T Q\alpha - \mathrm{e}^T\alpha$.

4.2. Computing the Lipschitz Constant *L*

The Fast Projected Gradient Method (FPGM) requires estimation of the Lipschitz constant *L*. Since $\mathcal{L}_\mu$ is of quadratic form with respect to $\boldsymbol{\alpha}$,

$$L = \left\|\nabla^2_{\alpha\alpha}\mathcal{L}_\mu(\cdot)\right\| = \left\|\mathbf{Q_B} + \mu\mathbf{y_B}\mathbf{y_B^T}\right\|,$$

where the matrix spectral norm is the largest singular value of a matrix.

To estimate *L* first $\|\mathbf{Q_B}\|$ was estimated. Since $\mathbf{Q_B}$ is symmetric and positive semidefinite, $\|\mathbf{Q_B}\| = \lambda_1$ is the largest eigenvalue of $\mathbf{Q_B}$.

The largest eigenvalue λ_1 can be efficiently computed using the power method. After estimating λ_1 with a few power iterations, the upper bound estimates *L* as follows:

$$L \approx \lambda_1 + m\mu. \tag{16}$$

5. Experimental Results

For testing, 11 binary classification training data sets (shown in **Table 1**) were selected with a number of training examples ranging from 18201 to 98528 from University of California, Irvine (UCI) Machine Learning Repository [19] and https://www.openml.org/. One of the tests, Test 2, was the largest test used in the previous work [6]. All simulations were done using a desktop with dual Intel Xeon 2.50 GHz, 12 Cores processors sharing 64 GB of computer memory. In all cases, $C = 100$ was used for all tests. The data was normalized and radial basis kernel was used with γ shown in **Table 1**. The scaling parameter μ used for all the experiments is $\mu = 0.1$ as suggested in [13]. The parameters in *pWSS* were taken as $\delta_\alpha = 10^{-6}$, $minAlphaOpt = 5$, $maxAlphaOpt = 20$. The accuracy of solution was selected $\varepsilon = 10^{-2}$. The SVM training times by ALFPGM were compared with results obtained using SMO implemented in Scikit-learn (Sklearn) [20] for the SVM training. The same data inputs were used in SMO and ALFPGM.

Table 1. Data used for testing.

	Test	m	n	# -1	# 1	γ	Description
1	Riccardo	18201	4295	3201	15000	0.500	https://www.openml.org/d/41161
2	Magictelescope	19020	11	12332	6573	0.500	Data are MC generated to simulate registration of high energy gamma particles in an atmospheric Cherenkov telescope.
3	2d_planes	26714	10	13369	13345	2.000	https://www.openml.org/d/727
4	Nomao	32062	118	22251	9811	5.000	The dataset has been enriched during the Nomao Challenge: organized along with the ALRA workshop.
5	Webdata_wXa	36974	123	8874	28100	1.000	https://www.openml.org/d/350
6	Fried	40768	10	20341	20427	5.000	https://www.openml.org/d/901
7	Bank-marketing	45211	16	5289	39922	2.000	https://www.openml.org/d/1461. The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed.
8	Electricity	45312	8	19237	26075	5.000	This data was collected from the Australian New South Wales Electricity Market. In this market, prices are not fixed and are affected by demand and supply of the market. https://www.openml.org/d/151
9	Run_or_walk_information	88588	6	44365	44223	1.000	This dataset describes whether a person is running or walking based on deep neural networks and sensor data collected from iOS devices.
10	Numerai28.6	96320	21	48658	47662	1.000	Encrypted Stock Market Training Data from Numer.ai.
11	VehicleNorm	98528	101	49264	49264	1.000	Vehicle classification in distributed sensor networks. https://www.openml.org/d/1242.

ALFPGM—Investigating of Using Different Number of Pairs *p*

The optimal number of pairs p needed for the training with ALFPGM is determined with numerical experiments using training Tests 2, 6 and 10, which are representatives of small, medium size, and large training sets. In all cases, the $pWSS$ method was employed for the pairs selection. The results are presented in Figures 2-4. The results suggest that for ALFPGM the optimum number of pairs is $50 \le p \le 100$.

There are several factors that come into play when varying the number of pairs p.

1) The size of p determines the amount of time it takes to find the size of the working set. The larger the size of p, the longer it takes to find the working set.

2) The smaller the size of p, the more the number of decompositions needed to solve the SVM problem.

3) The larger the size of p, the longer it takes to solve the SVM subproblems.

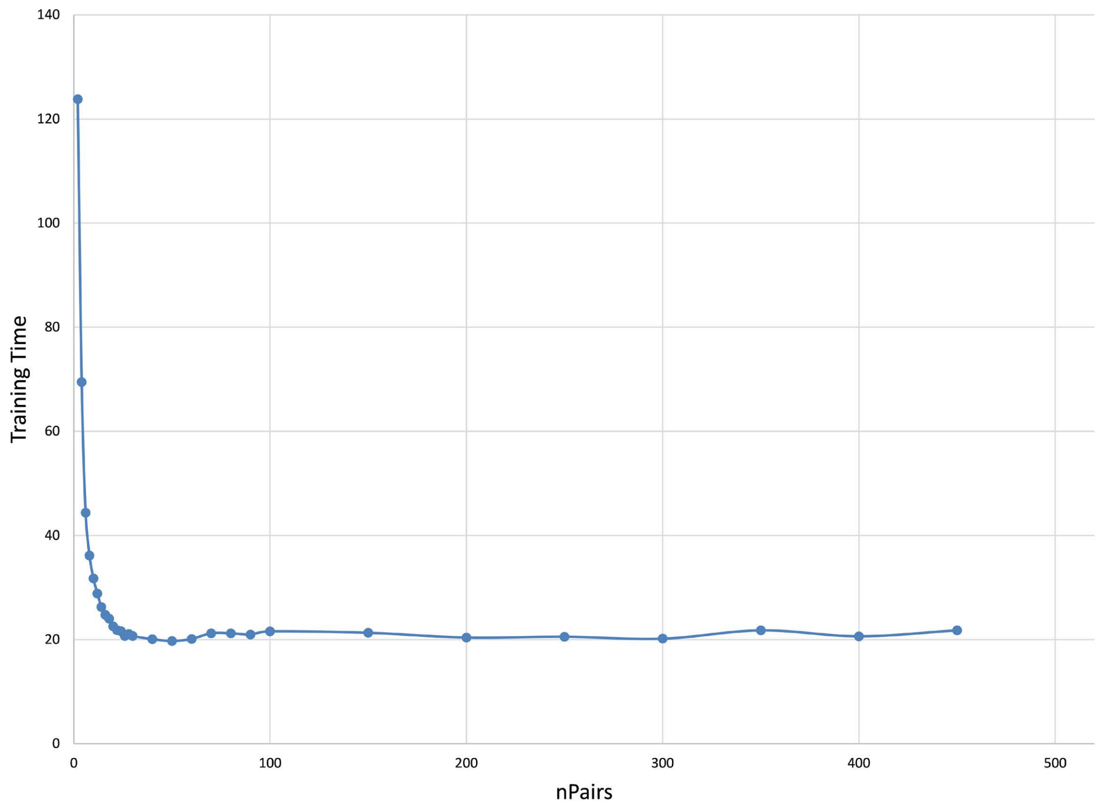


Figure 2. ALFPGM—Classification time for different number of pairs p Test 2.

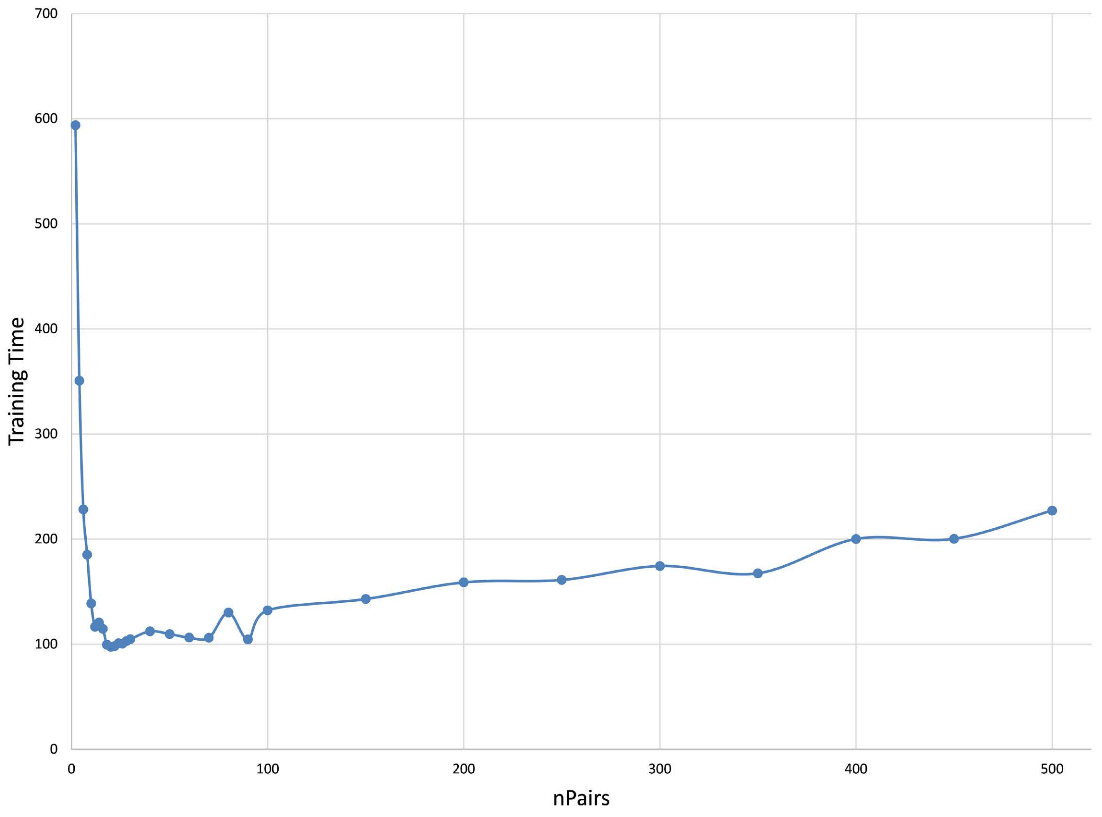


Figure 3. ALFPGM—Classification time for different numbers of pairs p Test 6.

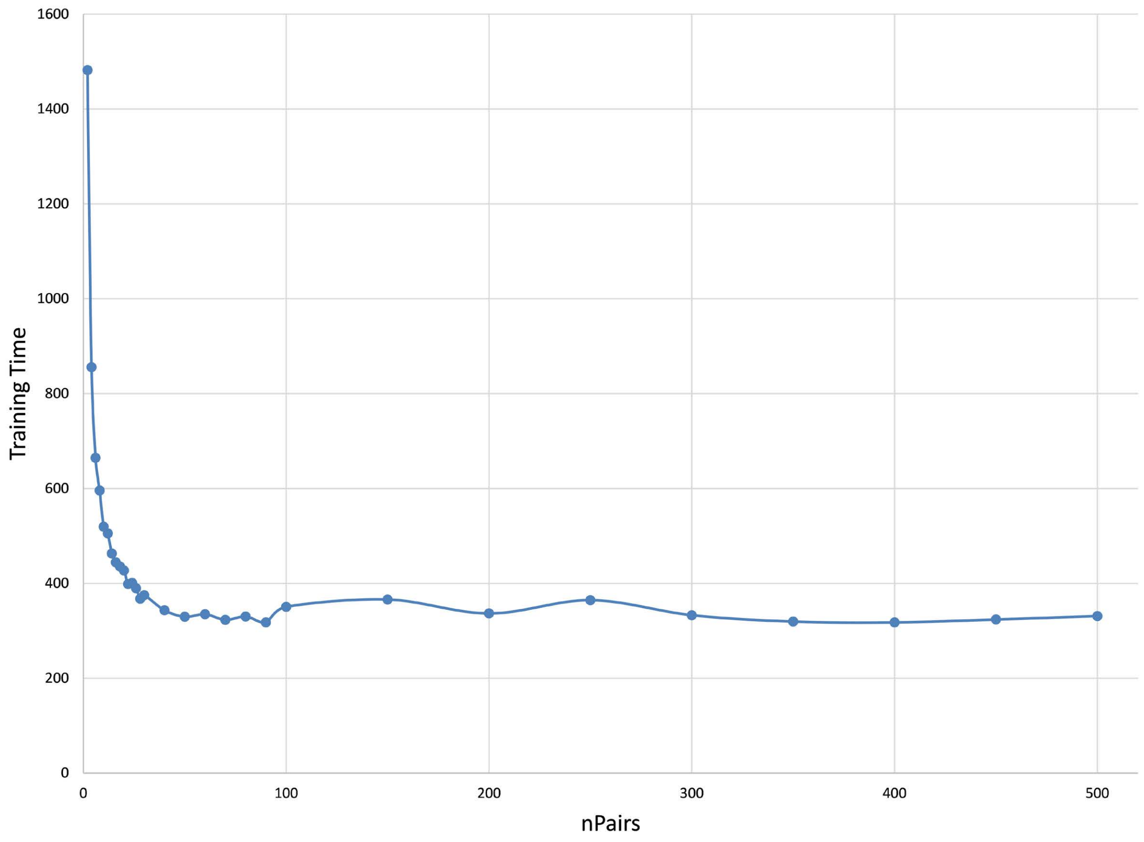


Figure 4. ALFPGM—Classification time for different numbers of pairs p Test 10.

4) The larger the size of p, the greater the probability that the Hessian matrix of the SVM subproblem may be degenerate.

Table 2 provides numerical results for training SVM with the ALFPGM and SMO. For each testing dataset, the table shows the number of training examples, the number of features. Then for both ALFPGM and SMO algorithms the table shows the training errors (the fractions of misclassified training examples), the training times, the number of performed decompositions (working set selections), the number of support vectors, and the optimal objective function values. To calculate training times, each simulation scenario was averaged over 5 runs. The results also are visualized in Figures 5-7. The training sets are arranged in the order of increasing the number of training examples. The figures demonstrate that for most of the cases the ALFPGM outperformed the SMO in training time for similar classification error.

Figure 6 shows that the training errors are similar demonstrating that both algorithms produced similar results. The same conclusion can be drawn by observing the similar values of the optimal objective functions for both algorithms in Table 2. Finally, Figure 8 shows the normalized training times *i.e.* the training times divided by the number of training examples. As one can see, the ALFPGM produces a smaller variance in the normalized times than that of the SMO algorithm.

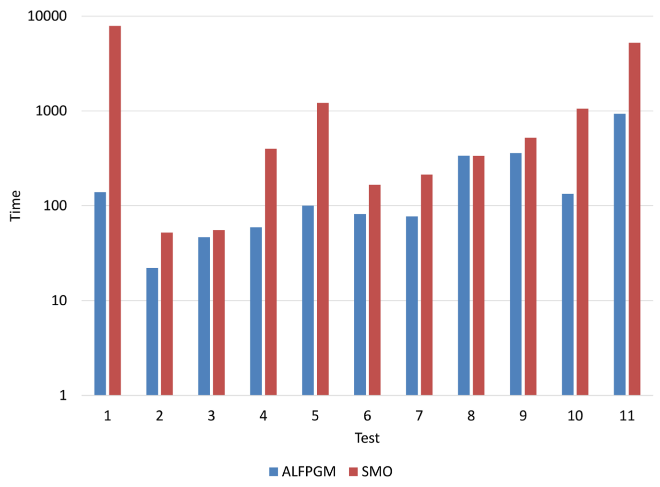


Figure 5. Comparison ALFPGM and SMO results timings.

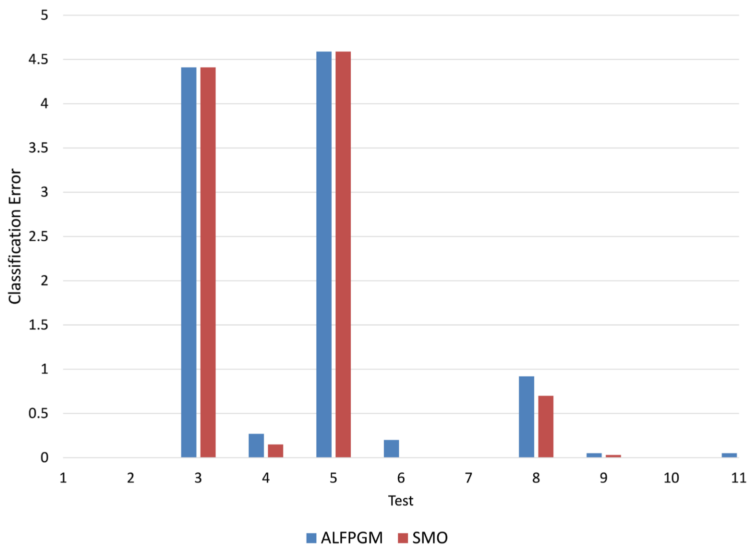


Figure 6. Comparison ALFPGM and SMO results-training classification error.

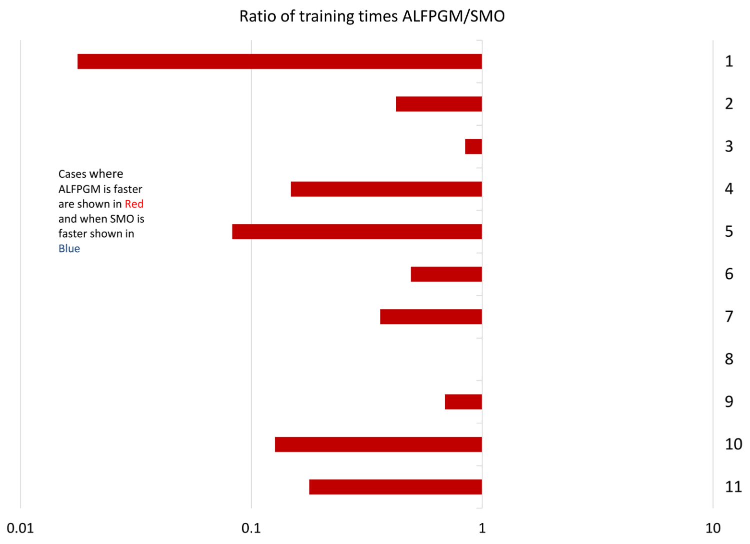


Figure 7. Comparison ALFPGM and SMO results.

Table 2. ALFPGM and SMO comparison.

			ALFPGM					SMO			
Test	numData	numFeat	train_err	wall_time (sec)	nDecomp	nSv	objval	train_err	wall_time	nDecomp	nSv	objval
1	18201	4295	0	139.096	275	18201	−5175.65	0	7870.151	49244	18201	−5175.65
2	19020	11	0	22.13577	294	18905	−8577.31	0	52.3462	62399	18905	−8577.31
3	26714	10	4.41	46.60157	485	26714	−231591	4.41	55.2686	39953	26714	−244605
4	32062	118	0.27	59.22488	630	20998	−15232.9	0.15	399.0199	49785	21016	−20231.4
5	36974	123	4.59	100.483	843	36974	−347874	4.59	1215.305	110026	36974	−350686
6	40768	10	0.2	81.7774	659	11701	−16033.5	0	166.716	79748	11123	−17599.1
7	45211	16	0	77.09719	508	45211	−9340.53	0	213.2611	96930	45211	−9340.53
8	45312	8	0.92	337.4581	2551	32681	−120386	0.7	336.2932	170816	32571	−135219
9	88588	6	0.05	359.1051	1223	43391	−25071.8	0.03	521.7201	122003	43331	−26753.4
10	96320	21	0	133.8112	327	96302	−49869.2	0	1056.651	138749	96304	−49916.9
11	98528	101	0.05	932.8615	1251	98411	−55816.7	0.03	5231.517	166893	98390	−56792.7
			Total time	2289.652					17118.25			

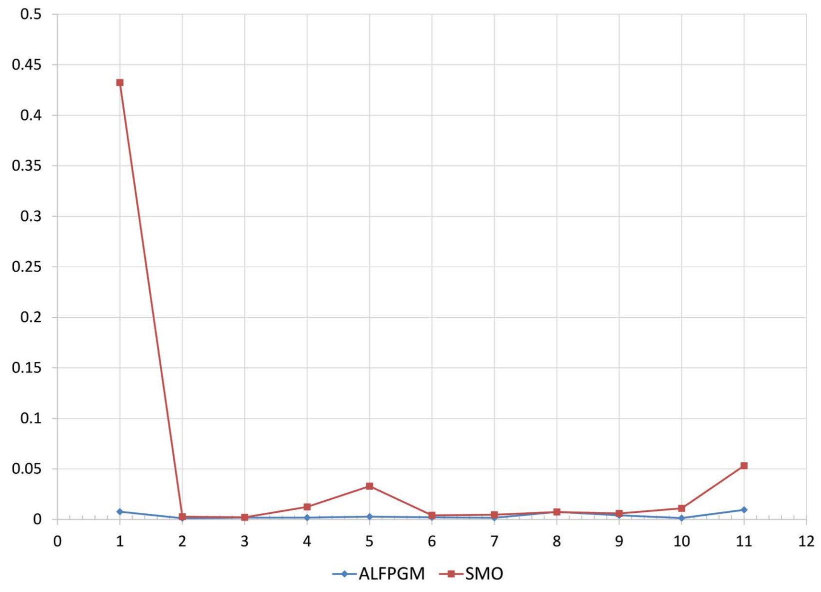


Figure 8. Comparison ALFPGM and SMO normalized time results.

6. Conclusions

The manuscript presents the results on training support vector machines using an augmented Lagrangian fast projected gradient method (ALFPGM) for large data sets. In particular, the ALFPGM is adapted for training using high performance computing techniques. Highly optimized linear algebra routines were used in the implementation to speed up some of the computations.

A working set decomposition scheme for p pairs selection is developed and optimized. Since working set selection takes a large portion of the overall training time, finding an optimal selection of pairs is the key to using multiple pairs in the decomposition. Numerical results indicate that the optimal choice of the number of pairs in the working set p for ALFPGM is 50 - 100 pairs. It is shown that in some examples the selection of pairs can be truncated earlier without compromising the overall results but with significantly reducing training times.

Finally, the comparison of training performance of the ALFPGM with that of the SMO from the Scikit-learn library demonstrates that the training times with ALFPGM is consistently smaller than those for the SMO for the tested large training sets.

7. Future Work

The results demonstrate that the choice of the number of pairs p is important as it determines the number of decompositions. Even though the numerical experiments determined a critical range of values for the optimal value of pairs p, the $pWSS$ can be further improved for multiple pair selection and the total training time will be further reduced.

One possible direction of achieving that improvement is to determine an optimal dependence of the number of pairs p on the size of the training set. Selecting optimal values of $WSS_{minAlphaCheck}$ and $minAlphaOpt$ used in $pWSS_3$ needs to be further explored. Finding the optimal values of ALFPGM parameters based on the input data can further improve the efficiency of the ALFPGM.

A use of graphics processing units (GPU), which are getting less expensive and with more capabilities than before, should be revisited in future. In the future, the most time consuming parts of the algorithm such as computing similarity kernel measures can be done using the GPU.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Vapnik, V. (1982) Estimation of Dependences Based on Empirical Data (Springer Series in Statistics). Springer-Verlag, New York.

[2] Vapnik, V.N. (1995) The Nature of Statistical Learning Theory. Springer-Verlag, New York. https://doi.org/10.1007/978-1-4757-2440-0

[3] Smola, A.J. and Scholkopf, B. (2004) A Tutorial on Support Vector Regression. *Statistics and Computing*, **14**, 199-222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

[4] Hastie, T. and Tibshirani, R. (1998) Classification by Pairwise Coupling. In: *Proceedings of the* 1997 *Conference on Advances in Neural Information Processing Systems* 10, MIT Press, Cambridge, 507-513. http://dl.acm.org/citation.cfm?id=302528.302744

[5] Keerthi, S. and Gilbert, E. (2002) Convergence of a Generalized SMO Algorithm for SVM Classifier Design. *Machine Learning*, **46**, 351-360.

https://doi.org/10.1023/A:1012431217818

[6] Aregbesola, M. and Griva, I. (2021) Augmented Lagrangian—Fast Projected Gradient Algorithm with Working Set Selection for Training Support Vector Machines. *Journal of Applied and Numerical Optimization*, **3**, 3-20. http://jano.biemdas.com/archives/1224

[7] Platt, J. (1998) Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft Corporation, Tech. Rep.

[8] Lin, C. (2001) On the Convergence of the Decomposition Method for Support Vector Machines. *IEEE Transactions on Neural Networks*, **12**, 1288-1298. https://doi.org/10.1109/72.963765

[9] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C. and Murthy, K.R.K. (2001) Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, **13**, 637-649. https://doi.org/10.1162/089976601300014493

[10] Wei, W., Li, C. and Guo, J. (2018) Improved Parallel Algorithms for Sequential Minimal Optimization of Classification Problems. 2018 *IEEE 20th International Conference on High Performance Computing and Communications*; *IEEE 16th International Conference on Smart City*; *IEEE 4th International Conference on Data Science and Systems* (*HPCC*/*SmartCity*/*DSS*), Exeter, 28-30 June 2018, 6-13. https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00033

[11] Chen, P.-H., Fan, R.-E. and Lin, C.-J. (2006) A Study on SMO-Type Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks*, **17**, 893-908. https://doi.org/10.1109/TNN.2006.875973

[12] Zhang, Q., Wang, J., Lu, A., Wang, S. and Ma, J. (2018) An Improved SMO Algorithm for Financial Credit Risk Assessment—Evidence from China's Banking. *Neurocomputing*, **272**, 314-325. http://www.sciencedirect.com/science/article/pii/S0925231217312328

[13] Bloom, V., Griva, I. and Quijada, F. (2016) Fast Projected Gradient Method for Support Vector Machines. *Optimization and Engineering*, **17**, 651-662. https://doi.org/10.1007/s11081-016-9328-z

[14] Polyak, R.A. (2015) Projected Gradient Method for Non-Negative Least Square. Infinite Products of Operators and Their Applications. *A Research Workshop of the Israel Science Foundation*, Haifa, 21-24 May 2012, 167-179. https://doi.org/10.1090/conm/636/12735

[15] Hestenes, M.R. (1969) Multiplier and Gradient Methods. *Journal of Optimization Theory and Applications*, **4**, 303-320. https://doi.org/10.1007/BF00927673

[16] Powell, M.J.D. (1978) Algorithms for Nonlinear Constraints That Use Lagrangian Functions. *Mathematical Programming*, **14**, 224-248. https://doi.org/10.1007/BF01588967

[17] Griva, I. (2018) Convergence Analysis of Augmented Lagrangian—Fast Projected Gradient Method for Convex Quadratic Problems. *Pure and Applied Functional Analysis*, **3**, 417-428.

[18] Fan, R.-E., Chen, P.-H. and Lin, C.-J. (2005) Working Set Selection Using Second Order Information for Training Support Vector Machines. *Journal of Machine Learning Research*, **6**, 1889-1918. http://dl.acm.org/citation.cfm?id=1046920.1194907

[19] Dua, D. and Gra, C. (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[20] Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.* (2011) Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830.