Particle swarm optimization for linear support vector machines based classifier selection

Gintautas Garšva, Paulius Danėnas

Department of Informatics, Kaunas Faculty, Vilnius University
Muitinės str. 8, LT-44280 Kaunas, Lithuania
gintautas.garsva@khf.vu.lt; paulius.danenas@vukhf.lt

Received: 7 November 2012 / **Revised:** 23 September 2013 / **Published online:** 25 November 2013

Abstract. Particle swarm optimization is a metaheuristic technique widely applied to solve various optimization problems as well as parameter selection problems for various classification techniques. This paper presents an approach for linear support vector machines classifier optimization combining its selection from a family of similar classifiers with parameter optimization. Experimental results indicate that proposed heuristics can help obtain competitive or even better results compared to similar techniques and approaches and can be used as a solver for various classification tasks.

Keywords: particle swarm optimization, linear SVM, support vector machines, machine learning, classification.

1 Introduction

Novel machine learning techniques play a very important role in solving various analysis and forecasting problems in various domains. Computational finance is one of such fields, which involves many researchers working in subfields such as financial forecasting, development of advanced techniques for credit risk modelling and evaluation to evaluate ratings, classify debtors by their risk level, predict bankruptcies. Various financial institutions and authorities such as banking sector, investors, governing authorities pay a lot more attention to these techniques as they prove to overcome limitations of previously applied techniques or tend to show competitive results in terms of accuracy or precision. As Balthazar refers in [1], machine learning techniques such as support vector machines (SVM) based models are already successfully applied to solve real world problems in Standard & Poor's rating company. Support vector machines is widely adopted classification technique with performance comparable to neural network classifiers; yet, they help to avoid some of their problems such as overtraining, overfitting, local minimas. SVM is applied to solve various classification problems in different domains, including bioinformatics and computational biology [2, 3], document classification [4, 5] image

recognition [6, 7] etc. as well as bankruptcy prediction [8–10]. Least squares SVM (LS-SVM), developed by Suykens and Vandevalle [11], reformulates SVM as quadratic programming problem, solved by a set of linear equations. Lai et al., Zhou and Lai used LS-SVM to develop approaches for credit risk evaluation [12, 13]. One of main challenges in SVM adoption for practical real world problems is parameter selection task – multiple SVMs with different parameters have to be computed in order to find SVM which results in best classification performance. This is also stated in book by Steinwart and Christmann [14]. Various heuristic and evolutionary optimization techniques are used to solve this task. Grid search is used as an option by various researchers such as Chen et al. [8], Yun et al. [15]; it is also implemented by default in some SVM packages such as LibSVM [16]. Papers which describe adoption of genetic algorithm report its benefits and increase in overall classification performance [17, 18]. Particle swarm optimization (abbr. as PSO) algorithm introduced by Kennedy [19] and based on behaviour of flock of birds has also been reported by various researchers as an effective tool for parameter selection [15, 20–23], feature selection [24] or combination of both [25–27].

Linear support vector machines are not widely applied to solve classification problems in credit risk domain, mainly because of their inflexibility in modelling. They are more applicable for large scale classification, whereas related surveys [28, 29] indicated that research of insolvency prediction or ratings analysis mostly involved mostly less than 1000 instances. This is related mainly to limited availability of related data; however, increasing amount of various financial data available online offers possibilities for new insights, as well as advantages of larger scale research. The authors of linear SVM (particularly its implementations in LIBLINEAR software) showed that in some cases it is able to produce competitive results to nonlinear SVMs while resulting in less complexity and reduced computational time required to train classifier [30–32]. Similar results were also obtained in our previous works [33, 34] where we worked with comparatively large number of instances. Wu [35] demonstrated that linear SVM classifiers can be very sensitive regarding their cost parameters; therefore their selection is necessary, yet subtle task. To deal with this problem, we previously introduced a hybrid technique based on PSO and linear SVM called PSO-LinSVM with capability to select linear SVM classifier algorithm together with its parameters. In this work we concentrated mainly on its adoption on credit risk evaluation problem, applying real-value based PSO algorithm for hybrid search space with one discrete dimension [36, 37]. Although the results were promising, the nature of real-valued PSO indicated its possible improvements regarding particle movement in discrete dimension. Therefore, this paper extends this work by proposing enhancement of PSO-LinSVM algorithm for hybrid search space.

2 Description of techniques used in research

2.1 Support vector machines algorithms

Support vector machines (SVM) technique is based on statistical learning theory, developed by Vapnik and Chervonenkis, and structural risk minimization (SRM), which minimizes an upper bound on the generalization error and enables good generalization

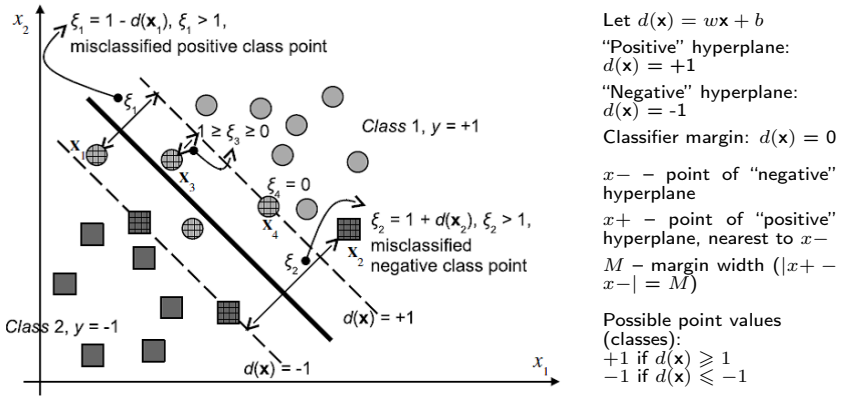

Fig. 1. Linear support vector machine illustration. (Source: adopted from [38], using comments by the authors.)

capabilities; it is described in detail in [39]. SVM performs data discrimination by mapping the input space to a high-dimensional feature space using kernel functions.

Linear SVM is illustrated in Fig. 1. The main objective is to find a hyperplane which minimizes margin error and is described as a set of support vectors. Finding these vectors from training data is formulated as quadratic optimization problem [39]

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i\big(\boldsymbol{w}^{\mathrm{T}}\phi(x_i) + b\big) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \end{aligned} \tag{1}$$

where C is a regularization (also referred as cost [16]) parameter that determines the trade-off between the maximum margin and the minimum classification error. The decision function is defined as [39]

$$\operatorname{sgn}\big(\phi(\boldsymbol{x}) \cdot \boldsymbol{w}\big). \tag{2}$$

If training vectors are not linearly separable, they can be represented in a larger (probably infinite) dimensional space by using kernel function $K(x_i, x_j) \equiv \phi(x_i)^{\mathrm{T}}\phi(x_j)$. SVM is solved using dual formulation [16, 38]

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^{l}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{l}\alpha_i y_i \quad \forall i\colon 0 \leqslant \alpha_i \leqslant C, \end{aligned} \tag{3}$$

where the number of training examples is denoted by l, training vectors $x_i \in \mathbb{R}$, $i = 1 \ldots l$, and a vector $y \in \mathbb{R}^l$ such that $y \in \{-1, 1\}$. $\boldsymbol{\alpha}$ is a vector of l Lagrange multipliers, where each α_i corresponds to a training example (x_i, y_i). According to [38], parameters

Table 1. Linear SVM classification algorithms and their formulations.

Algorithm	Minimization problem
L2-regularized logistic regression	$\min_w \frac{1}{2}w^{\mathrm{T}}w + C\sum_{i=1}^{l}\log(1+e^{y_l w^{\mathrm{T}}x_l})$
L2-regularized L2-loss SVC (dual)	$\min_\alpha \frac{1}{2}\alpha^{\mathrm{T}}Q\alpha - \boldsymbol{e}^{\mathrm{T}}\alpha, \quad 0 \leqslant \alpha_i \leqslant C,\ i = 1\dots l$
L2-regularized L2-loss SVC (primal)	$\min_w \frac{1}{2}w^{\mathrm{T}}w + C\sum_{i=1}^{l}(\max(0, 1-y_l w^{\mathrm{T}}x_l))^2$
L2-regularized L1-loss SVC	$\min_w \frac{1}{2}w^{\mathrm{T}}w + C\sum_{i=1}^{l}\max(0, 1-y_l w^{\mathrm{T}}x_l)$
L1-regularized L2-loss SVC	$\min_w \frac{1}{2}\|w\|_1 + C\sum_{i=1}^{l}(\max(0, 1-y_l w^{\mathrm{T}}x_l))^2$
L1-regularized logistic regression	$\min_w \frac{1}{2}\|w\|_1 + C\sum_{i=1}^{l}\log(1+e^{y_l w^{\mathrm{T}}x_l})$
L2-regularized logistic regression (dual)	$\min_\alpha \frac{1}{2}\alpha^{\mathrm{T}}Q\alpha + \sum_{i:\alpha_i>0}\alpha_i\log\alpha_i$ $-\sum_{i:\alpha_i<C}(C-\alpha_i)\log(C-\alpha_i) - \sum_{i=1}^{l}C\log C$

for optimal hyperplane $\boldsymbol{w}_0$ and b_0 are obtained using

$$\boldsymbol{w}_0 = \sum_{i=1}^{l}\alpha_{0i}y_i\boldsymbol{x}_i, \qquad b_0 = \frac{1}{N_{SV}}\sum_{s=1}^{N_{SV}}\big(y_s - \boldsymbol{x}_s^{\mathrm{T}}\boldsymbol{w}_0\big), \quad s = 1\dots N_{SV}, \tag{4}$$

where N_{SV} is the number of support vectors. Support vectors are instances which have nonzero α_{0i} and support forming the decision function which becomes [16]

$$\operatorname{sgn}\left(\sum_{i=1}^{l}y_i\alpha_i K(x_i, \boldsymbol{x}) + b\right). \tag{5}$$

Linear SVM. A linear SVM classifier is defined as follows [40]: given training vectors $x_i \in \mathbb{R}^n$, $i = 1\dots l$, in two class, and a vector $y \in \mathbb{R}^l$ such that $y \in \{-1, 1\}$, a linear classifier generates a weight vector $\boldsymbol{x}$ using a decision function

$$\operatorname{sgn}\big(w^{\mathrm{T}}x\big). \tag{6}$$

LIBLINEAR includes a family of linear SVM and logistic regression classifiers for large-scale SVM classification. These classifiers have several advantages over nonlinear SVM implementations (such as LibSVM or SVM$^{\mathrm{Light}}$) as absence of kernel functions results in reduced complexity and training time. In some cases, the discriminant function of the classifier includes a bias term b. LIBLINEAR handles this term by augmenting the vector $\mathbf{w}$ and each instance x_i with an additional dimension $\mathbf{w}^{\mathrm{T}} \leftarrow \big\lfloor \mathbf{w}^{\mathrm{T}}, \mathbf{b}\big\rfloor$, $\mathbf{x_i}^{\mathrm{T}} \leftarrow \big\lfloor \mathbf{x_i}^{\mathrm{T}}, \mathbf{b}\big\rfloor$ using constant B, which is specified by the user as bias term (further B will be referred as bias parameter) [40]. According to [40], L1-SVM and L2-SVM are solved using coordinate descent method [40, 41]; for logistic regression and L2-SVM, a trust region Newton method [40, 42] is implemented.

For the research, LIBLINEAR classifiers listed in Table 1 were used; for more information refer to [30, 40].

Least squares SVM. Least squares SVM (abbr. as LS-SVM) aims to solve a set of linear equations instead of convex quadratic programming performed by using standard SVM.

LS-SVM is also referred as kernel Fisher discriminant analysis [11]. The problem can be formulated as

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2}w^{\mathrm{T}}w + \frac{1}{2}\gamma\sum_{i=1}^{l}e_i^2 \\ \text{subject to} \quad & y_i\big(w^{\mathrm{T}}\phi(x_i)+b\big) = 1 - e_i, \quad i = 1\ldots l, \end{aligned} \tag{7}$$

with l denoting the number of training instances. Lagrangian for LS-SVM is defined as [11]

$$L(w,b,e,\alpha) = \frac{1}{2}w^{\mathrm{T}}w + \frac{1}{2}\gamma\sum_{i=1}^{l}e_i^2 - \sum_{k=1}^{l}\alpha_k\big\{y_k\big[\big(w^{\mathrm{T}}\phi(x_k)+b\big)\big] - 1 + e_k\big\} \tag{8}$$

with α_k as Lagrangian multipliers.

2.2 Heuristic optimization algorithms applied in researh

Simulated annealing (SA) method is described as "based on the metaphor of molecules cooling into a crystalline pattern after being heated. In a molten metal the molecules move chaotically, and as the metal cools they begin to find patterns of connectivity with neighbouring molecules, until they cool into a nice orderly pattern – an optimum" [19]. According to Kennedy et al., simulated annealing extends hill climbing with a stochastic decision and a cooling schedule [19]. Application of this technique has been proved to show good performance and obtain good results in relatively small number of iterations. The main idea of the algorithm is as follows: at each iteration, a set of possible solutions is generated and a random successor v is chosen. If $f(v) < f(u)$, it is considered as optimal, otherwise cost function value increases, then the new set is accepted with a certain probability and a random step is taken to obtain the new variable set where r is selected according to

$$r < \mathrm{e}^{(f(u)-f(v))/T}, \tag{9}$$

here r is random value from uniform distribution, i.e., $r \sim U(0,1)$. After a certain number of iterations, the new variable sets do not minimize the costs, thus the procedure is stopped after temperature T converges to 0 (i.e., $T \leqslant \epsilon$).

Particle swarm optimization (PSO). Particle swarm optimization (abbr. as PSO) algorithm was introduced by Kennedy [19]. This technique is based on behaviour of flock of birds which search for food randomly in some area, knowing only the distance from the food. In PSO, each possible solution is represented as this bird and is called a particle, and its location relative to the object which is searched (food in this example) is defined by the fitness value. Thus all the particles have one fitness value defined by a function which is optimized, and each particle has one velocity to determine its flying direction and distance. All the particles perform search in the solution space by following currently the most optimal particle. PSO is initialized to be a group of random particles and iteratively find the optimal solution. In each iteration each particle is updated itself by two extremes that are tracked. The first extreme is the optimal solution found by the particle itself

($pbest$), the other is the optimal solution found by the whole swarm ($gbest$). At each step of the algorithm, particles are displaced from their current position by applying a velocity vector to them. The magnitude and direction of their velocity at each step is influenced by their velocity in the previous iteration of the algorithm, simulating momentum, and the location of a particle relative to the location of its $pbest$ and the $gbest$. At each step a particle is stochastically accelerated towards its previous best position and towards a neighbourhood (global) best position, thereby forcing particles to continually search in the most-promising regions found so far in the solution space.

Velocity equation includes such components [43]:

- The previous velocity (also referred as inertia or momentum) $v_p(t)$, representing memory of the previous movement direction, i.e. movement in the immediate past. This prevents the particle from drastically changing direction, and to bias towards the current direction.
- Global optima value $\hat{y}$.
- The cognitive component $c_1 r_1(y_p - x_p)$, representing confidence in solutions of individual particle which is relative to past performances and encourages the particle to return to their own best positions; r_1 is a random value with a uniform distribution, i.e., $r_1 \sim U(0, 1)$.
- The social component $c_2 r_2(\hat{y} - x_p)$ representing confidence on solutions by the neighbours of particle and representing common standard that individuals want to obtain; $r_2 \sim U(0, 1)$. The particle is able to head to the best position found by the particle's neighbourhood.
- Positive acceleration constants (also referred as learning factors) c_1 and c_2 used to scale the contribution of the cognitive and social components.

3 Proposed classification approach

As Table 1 shows, linear SVM based classifiers, although having different formulations, share several common parameters. Therefore, selection problem can be approached as selection of classifier together with its parameters instead of parameter selection for each classifier from this set. This lead to a metaheuristic approach proposed in this paper. According to the techniques which are used for its development (particle swarm optimization and linear SVM) it is further referred as PSO-LinSVM.

Definition 1. Each particle $P = \langle p_1; p_2; p_3 \rangle$ in PSO-LinSVM is represented as follows:

- p_1 – non-negative integer value, which represents the algorithm used for classification;
- p_2 – real value, cost parameter C;
- p_3 – real value, which represents bias term B.

Note that p_1 value itself does not play an important role in obtaining position value, as p_1 is initialized randomly in whole search space, and optimization is done according to performance of SVM classifier represented by this particle. However, scattered

values may influence particle velocity; therefore, it is required that p_1 values are non-negative successive integers (i.e., given $cl_{min} \leqslant P_{i1} \leqslant cl_{max}$, $S(i) = i + h$ for each P_{i1}). Although it is possible that it can be used with other h values, it is not reasonable computationally, as corresponding population initialization and velocity equations would require modifications by replacing round operations with operators which ensure that P_{i1} and velocity values stay valid and require additional operations. Thus $h = 1$ was used in the experiments. The results can depend the number of particles used in optimization – the larger number of particles is used, the better coverage of search space is obtained, but the larger is the demand for computational resources. Another factor which can influence final results arises from initialization of particles which are initialized randomly in search space; thus cl sequential ordering as well as implementation of random number generator used in the implementation of this algorithm can also have impact on the results.

Such inner representation is used in further research (according to $-s$ value for training command in LIBLINEAR library which defines the linear SVM classifier):

- 0 – L2-regularized logistic regression (primal);
- 1 – L2-regularized L2-loss support vector classification (dual);
- 2 – L2-regularized L2-loss support vector classification (primal);
- 3 – L2-regularized L1-loss support vector classification (dual);
- 4 – L1-regularized L2-loss support vector classification;
- 5 – L1-regularized logistic regression;
- 6 – L2-regularized logistic regression (dual).

The main objective of this algorithm is to maximize fitness function defined as sum of TPR values for each class

$$f_{fitness} = \sum_{i=1}^{N_C} TPR_i = \sum_{i=1}^{N_C} \frac{TP_i}{FN_i + TP_i}, \tag{10}$$

where N_C is the number of classes, TPR_i – TPR value for ith class. Alternatively, it can be defined as minimization problem where it is aimed to minimize the difference between "ideal" performance (i.e., when TPR value for all classes is equal to 1) and performance obtained by the classifier

$$f_{fitness} = N_C - \sum_{i=1}^{N_C} TPR_i = N_C - \sum_{i=1}^{N_C} \frac{TP_i}{FN_i + TP_i}. \tag{11}$$

Accuracy or error ratio for fitness evaluation is chosen often [21–23]; however, in case of imbalanced learning, accuracy is not the best option (it is possible to obtain high classification accuracy, if the classifier correctly recognizes most of "majority" instances, but fails to identify most of "minority" instances), so sum of TP rate values is selected in our approach. These evaluations are obtained by performing k-fold cross-validation training; the number of folds can be selected according to the size of training dataset. As the formula shows, an ideal solution can be obtained only in case of perfect classification;

as this happens very rarely, the main goal is to find satisfactory solution. Therefore, this technique ism also adjusted in order to terminate search after no further improvements in performance are observed. This technique also comprises such aspects as velocity clamping, where $V_{max,j}$ represents maximum allowed velocity in dimension j. According to Engelbrecht [43], large values of $V_{max,j}$ facilitate global exploration, while smaller values encourage local exploitation. It is often computed as a fraction σ of search space and selected empirically, according to the problem which is solved. Therefore, in proposed technique it is calculated as

$$\sigma_j = \frac{R_{max,j}}{|R_{min,j}| + |R_{max,j}|} \times 0.8, \tag{12}$$

where $R_{min,j}$ is denoted as minimum of search space for jth dimension, $R_{max,j}$ – as its maximum.

The proposed approach for linear SVM classifier selection based on these principles is presented as Algorithm 1. The algorithm is defined as a solver for $f_{fitness}$ minimization problem, as defined in Eq. (11), although it can be easily adapted to solve maximization problem in Eq. (10).

Algorithm 1. PSO-LinSVM algorithm.

function PSO-LINSVM(n, $c1$, $c2$, $terminate\_iterations$, $max\_iterations$, $rangeC$, $rangeBias$)

Input: a set of examples $\{\mathbf{x}, y\}, y \in \mathbb{Z}$

Output: SVM with a set of optimal parameters y_p

$k \longleftarrow 3$ ▷ number of dimensions in particle, representing linear SVM classifiers according to Definition 1
$perf \longleftarrow []$
$global\_fitness \longleftarrow 0$
$term\_iterations \longleftarrow 0$
$t \longleftarrow 0$ ▷ number of iterations
$P \longleftarrow Init(n)$ ▷ Initialize a 3-dimentional swarm
for $\forall p_x \in P$ **do**
  $p_{x1} \longleftarrow cl_{min} + \text{ROUND}(\text{rand}(0,1) \times (cl_{max} - cl_{min}))$
  $p_{x2} \longleftarrow C_{min} + \text{rand}(0,1) \times (C_{max} - C_{min})$
  $p_{x3} \longleftarrow b_{min} + \text{rand}(0,1) \times (b_{max} - b_{min})$
  $y_p \longleftarrow P$
repeat
  if $no\_iterations = max\_iterations$ **then return** SVM(y_p)
  for $\forall p_x \in P$ **do**
    $f(x_p) \longleftarrow \text{EVALSVM}(p_{x1}, p_{x2}, p_{x3})$
    if $f(x_p) < \hat{y}(t)$ **then**
      $y_p \longleftarrow x_p$
      $term\_iterations \longleftarrow 1$
    else
      $term\_iterations \longleftarrow term\_iterations + 1$
  if $f(y_p) < f(\hat{y})$ **then** $\hat{y} = y_p$
  for $\forall p_x \in P$ **do**
    for $j \longleftarrow 1, k$ **do**
      $V_{max} \longleftarrow \sigma_j \times (R_{max,j} - R_{min,j})$

Algorithm 1. (*Continued.*)

if $j = 1$ **then**
$V_{max} \longleftarrow \text{ROUND}(V_{max})$
$v_{pj}(t+1) \longleftarrow v_{pj}(t) + \text{ROUND}(c_1 \times \text{rand}(0,1) \times (y_{pj}(t) - x_{pj}(t)) + c_2 \times \text{rand}(0,1) \times (\hat{y}_j(t) - x_{pj}(t)))$
else
$v_{pj}(t+1) \longleftarrow v_{pj}(t) + c_1 \times \text{rand}(0,1) \times (y_{pj}(t) - x_{pj}(t)) + c_2 \times \text{rand}(0,1) \times (\hat{y}_j(t) - x_{pj}(t))$
if $v_{pj}(t+1) \geqslant V_{max}$ **then** $v_{pj}(t+1) \longleftarrow V_{max}$
$x_p(t+1) \longleftarrow x_p(t) + v_p(t+1)$
$y_p(t+1) \longleftarrow \begin{cases} y_p(t) & \text{if} f(x_p(t+1)) \leqslant f(y_p(y)) \\ y_p(t+1) & \text{if} f(x_p(t+1)) > f(y_p(y)) \end{cases}$
if $x_{p1}(t+1) \geqslant cl_{max}$ **then** $x_{p1}(t+1) \longleftarrow cl_{min}$
if $x_{p2}(t+1) < C_{min}$ **then** $x_{p2}(t+1) \longleftarrow C_{min}$
$\hat{y}(t) \longleftarrow \min(f(y_0(t)), \ldots, f(y_n(t)))$
$t \longleftarrow t + 1$
until $term\_iterations < terminate\_iterations$

Such parameters are defined for the proposed algorithm:

- n – size of swarm.
- $c1$ – PSO coefficient for cognitive component.
- $c2$ – PSO coefficient for social component.
- $cl \longleftarrow \{i \mid cl_{min} \leqslant i \leqslant cl_{max},\ cl_{min} \in \mathbb{Z},\ i \in \mathbb{Z},\ cl_{max} \in \mathbb{Z}\}$ – a set of classifiers, represented by inner encodings.
- $rangeC = [C_{min}; C_{max}]$ – range of cost parameters which is considered (note that $C \geqslant 0$).
- $rangeBias = [b_{min}; b_{max}]$ – range of B (bias term) parameters which is considered in optimization.
- $terminate\_iteration$ (optional) – parameter which defines the number of iterations after which PSO optimization should be terminated if no further improvement is observed.
- $max\_iterations$ (optional) – maximum number of iterations for PSO optimization. It is also optional and if it is not given, the procedure loops until $terminate\_iteration$ criteria is satisfied. This can be considered if a fast convergence to optimal solution is known to occur.
- $\hat{y}(t)$ is the global best position obtained at iteration t.

Several modifications, compared to original PSO $gbest$ algorithm, can be excluded here:

- This algorithm performs search in mixed search space (one of dimensions is mapped to integer space, while other two are represented by subsets of real-valued space), thus both initialization procedure and velocity equation are modified to meet these requirements.

- The search space is constrained by several constraints ($p_1 \in \{i \mid cl_{min} \leqslant i \leqslant cl_{max},\ cl_{min}, i, cl_{max} \in \mathbb{Z}\};\ p_2 > 0$) which cannot be violated, i.e., none of obtained parameters can be outside of these constraints. To deal with these constraints, particle "teleportation" principle is applied (if particle reaches upper boundary for first dimension it is moved back to the lower boundary), while infeasible value for p_2 is replaced by minimal value of cost parameter C.

4 Experimental results

An experiment to verify classification efficiency of PSO-LinSVM algorithm defined in Algorithm 1 was performed using Australian and German datasets; they both can be accessed in the UCI repository. These datasets were chosen because of their popularity and wide adoption for similar experiments as well as the credit risk domain context; therefore, the results can be used in benchmarking to compare with similar algorithms. Two variations of German credit dataset are provided: the original dataset which contains categorical/symbolic attributes and one for algorithms that need numerical attributes. The latter file has been edited and several indicator variables have been added to make it suitable for algorithms which cannot cope with categorical (nominal) variables. Thus the final German credit dataset consists of 1000 instances (700 instances labeled as "Class 1" and 300 instances labeled as "Class 2") with 24 numerical attributes. Main specification of numerical German credit dataset is given in Table 2.

Australian credit dataset concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. Numerical version of this dataset has 690 instances with 14 attributes; of these 690 instances, 383 instances are labeled as "Class 1" and 307 instances are labeled as "Class 2".

The proposed technique was compared with LibSVM and LS-SVM implementations. The experiment was performed using MATLAB 2010b environment, LibSVM 3.12, LibLINEAR 1.8 and LS-SVMlab 1.8 toolboxes. Their parameter selection was implemented using simulated annealing algorithm in MATLAB's optimization toolbox, while PSO

Table 2. Main characteristics of German dataset.

Attribute	Min. value	Max. value	Mean	Std. deviation	Attribute	Min. value	Max. value	Mean	Std. deviation
A1	1	4	2.577	1.258	A13	1	2	1.155	0.362
A2	4	72	20.903	12.059	A14	1	2	1.404	0.491
A3	0	4	2.545	1.083	A15	1	2	1.037	0.189
A4	2	184	32.711	28.253	A16	0	1	0.234	0.424
A5	1	5	2.105	1.58	A17	0	1	0.103	0.304
A6	1	5	3.384	1.208	A18	0	1	0.907	0.291
A7	1	4	2.682	0.708	A19	0	1	0.041	0.198
A8	1	4	2.845	1.104	A20	0	1	0.179	0.384
A9	1	4	2.358	1.05	A21	0	1	0.713	0.453
A10	19	75	35.546	11.375	A22	0	1	0.022	0.147
A11	1	3	2.675	0.706	A23	0	1	0.2	0.4
A12	1	4	1.407	0.578	A24	0	1	0.63	0.483

optimization was developed using PSO toolbox for MATLAB by Sam Chen[1]. Two approaches for fitness evaluation were applied, in order to obtain a classifier with best classification performance capabilities in both balanced and unbalanced classification conditions:

- Accuracy, obtained using k-fold cross-validation (further referred as CV optimization).
- The sum of TP ratios also obtained using k-fold cross-validation (this principle further referred as balanced CV optimization). This is an approach used in [36, 37].

In the experiment, $k = 5$ was selected (although if dataset is large, $k = 2$ or $k = 3$ can be considered as a computationally more efficient choice). Fig. 2 and Fig. 3 present result visualizations obtained after iteratively performing classification tasks in bounded search spaces, for both accuracy and unbalanced classification optimization. In order enable comparison of various optimization approaches, a direct parameter search procedure was run, using seven linear SVM classifiers in LIBLINEAR 1.8 toolbox (this procedure is further referred as LIBLINEAR+DS). These figures show performance results in terms of accuracy, where each point is represented as (C; $bias$; $\max(acc_i)$), with and acc_i as accuracy obtained by best performing classifier from the set of LIBLINEAR classifiers at particular search space point. C parameter change step was set to 5, whereas bias parameter was set to change by 1. Such representation can be used to visualize search surface and can be used to identify core parameters for optimization procedure. As an example, these figures help to identify that best performing classifier had relatively large C (somewhere between 30 and 70) and bias parameters for the case of German dataset, whereas regularization parameter is relatively small for Australian credit dataset. This information can be considered during initialization of particles, for e.g., to initialize larger part of swarm in particular regions in order to enable faster convergence or increase possibility to obtain best possible solution. In this research, default PSO initialization (with its modifications for discrete dimension) is considered. Notably, unbalanced classification resulted in total domination of single classifiers with different C and $bias$ parameters.

According to Engelbrecht, larger social coefficient is a better option for search spaces with smooth surface while larger cognitive coefficient is preferred for problem spaces which have many global and local optimas and therefore result in rough search space [43]. As visualizations of linear SVM search space show, SVM selection problem can be considered as the latter case, therefore, in further experiments $c_1 > c_2$ will be used. These figures also show that accuracy selected as evaluation metrics resulted in wide range of classifiers, i.e., none of the classifiers could be identified as the most effective solution whereas in case of balanced cross-validation (using sum of true positive rate values) based evaluation single classifiers (dual L2-regularized L1-loss support vector classification and dual L2-regularized logistic regression) dominated. Note that although such results might indicate optimal classifiers, they are not necessary among the visualized points, as direct search was performed in discrete and limited search subspace, thus this representation is very approximate.

[1] Another Particle Swarm Toolbox, `http://www.mathworks.com/matlabcentral/fileexchange/25986-another-particle-swarm-toolbox`

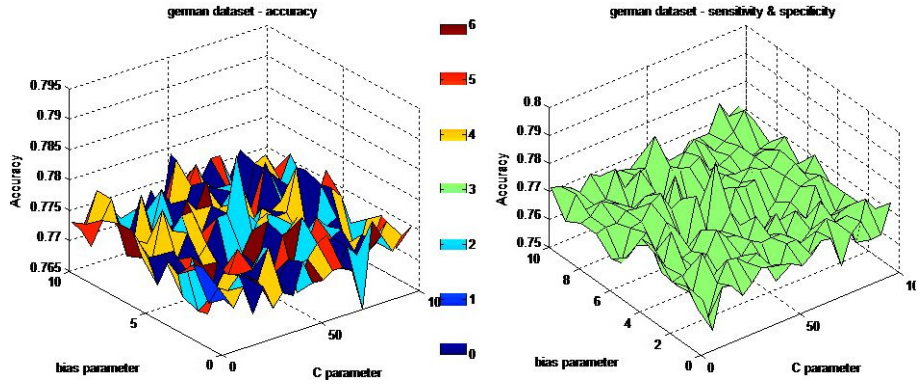

Fig. 2. Linear SVM classifier initial results (German dataset).

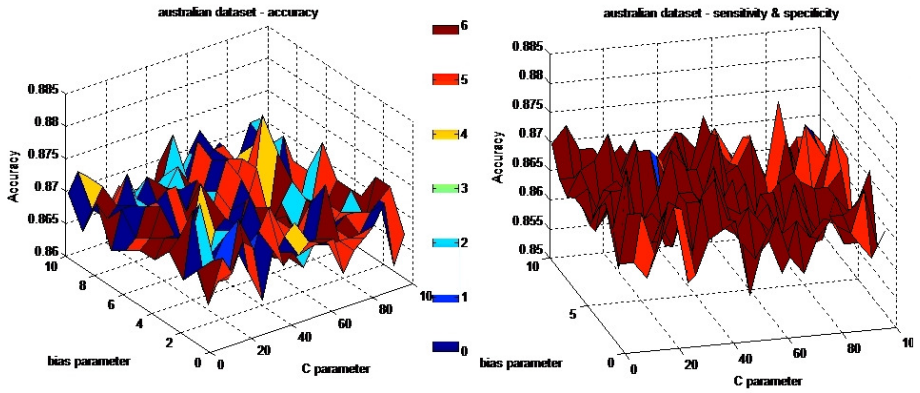

Fig. 3. Linear SVM classifier initial results (Australian dataset).

In order to compare proposed approach with similar classification techniques, similar SVM (particularly LibSVM C-SVC) and LS-SVM classifiers were also developed by performing heuristic parameter selection on their kernel functions using previously described simulated annealing and particle swarm optimization. C-SVC was run using with RBF (further referred as LibSVM$^{\mathrm{RBF}}$) and sigmoid (further referred as LibSVM$^{\mathrm{Sigmoid}}$) kernel functions (polynomial kernel function was not selected of relatively large parameter space and slow performance) whereas LS-SVM classifiers were based on polynomial (further referred as LS-SVM$^{\mathrm{Poly}}$) and RBF (LS-SVM$^{\mathrm{RBF}}$) kernels. Dataset split 7:3 (i.e., 70% of data was selected for classifier training and optimization procedure, the rest 30% were used for testing) widely used in such research was selected.

SA procedure was run using exponential temperature (*temperatureexp*) and *simulannealbnd* functions in 180 iterations whereas PSO implementation was applied with default parameters. LIBLINEAR and PSO based classifier with similar approach to PSO-LinSVM (used in [36]) was also tested; the main difference lies in its design as it is based on real-valued PSO implementation instead of hybrid which is proposed in this paper.

Similar approach is also applied for LIBLINEAR classifier selection using SA; default MATLAB real-valued SA implementation was used for its implementation.

Table 3 and Table 4 present classification results for different SVM based classifiers represented as error rate and true positive rates for each class; core parameters which were obtained during parameter selection such as selected classifier index (in cases of LIBLINEAR based classifiers, as well as PSO-LinSVM) and C parameter (for all classifiers) are also given.

Table 3. German dataset results.

Optimization based on accuracy					
	Linear classifier code	Cost parameter	Error rate	TPR_1	TPR_2
LIBLINEAR+DS	0	46	0.214	0.897	0.527
PSO-LinSVM	3	14.808	**0.187**	0.894	0.634
Particle Swarm Optimization					
LIBLINEAR	5	99.274	0.197	0.894	0.602
LibSVM$^{\text{RBF}}$	–	0.014	0.197	0.903	0.591
LibSVM$^{\text{Sigmoid}}$	–	2.885	0.247	0.889	0.581
LS-SVM$^{\text{Poly}}$	–	8.659	0.33	0.763	0.462
LS-SVM$^{\text{RBF}}$	–	4.674	0.217	0.889	0.548
Simulated Annealing					
LIBLINEAR	6	76.788	0.203	0.884	0.602
LibSVM$^{\text{RBF}}$	–	0.013	0.207	0.889	0.581
LibSVM$^{\text{Sigmoid}}$	–	19.52	0.297	0.966	0.14
LS-SVM$^{\text{Poly}}$	–	9.969	0.357	0.72	0.473
LS-SVM$^{\text{RBF}}$	–	2.198	0.24	0.894	0.462
Optimization based on balanced accuracy					
LIBLINEAR+DS	3	46	0.214	0.897	0.527
PSO-LinSVM	3	14.808	**0.187**	0.894	0.634
Particle Swarm Optimization					
LIBLINEAR	7	96.112	0.233	0.797	0.699
LibSVM$^{\text{RBF}}$	–	0.016	0.217	0.874	0.581
LibSVM$^{\text{Sigmoid}}$	–	11.406	0.38	0.72	0.613
LS-SVM$^{\text{Poly}}$	–	2.859	0.49	0.536	0.452
LS-SVM$^{\text{RBF}}$	–	3.944	0.24	0.87	0.516
Simulated Annealing					
LIBLINEAR	5	85.577	0.2	0.874	0.634
LibSVM$^{\text{RBF}}$	–	0.012	0.2	0.889	0.602
LibSVM$^{\text{Sigmoid}}$	–	10.932	0.273	0.908	0.387
LS-SVM$^{\text{Poly}}$	–	0.138	0.363	0.705	0.484
LS-SVM$^{\text{RBF}}$	–	5.752	0.23	0.87	0.548

Table 4. Australian dataset results.

Optimization based on accuracy					
	Linear classifier code	Cost parameter	Error rate	TPR_1	TPR_2
LIBLINEAR+DS	5	6	**0.122**	0.864	0.896
PSO-LinSVM	1	33.606	**0.126**	0.853	0.901
Particle Swarm Optimization					
LIBLINEAR	6	15.401	0.164	0.905	0.747
LibSVM$^{\text{RBF}}$	–	0.02	0.184	0.914	0.692

Table 4. (*Continued.*)

	Linear classifier code	Cost parameter	Error rate	TPR_1	TPR_2
LibSVM$^{\text{Sigmoid}}$	–	20	0.169	0.905	0.736
LS-SVM$^{\text{Poly}}$	–	3.85	0.43	0.655	0.462
LS-SVM$^{\text{RBF}}$	–	9.972	0.164	0.879	0.78
Simulated Annealing					
LIBLINEAR	7	0.005	0.159	0.905	0.758
LibSVM$^{\text{RBF}}$	–	0.057	0.213	0.888	0.659
LibSVM$^{\text{Sigmoid}}$	–	0.119	0.179	0.897	0.725
LS-SVM$^{\text{Poly}}$	–	0.01	0.164	0.914	0.736
LS-SVM$^{\text{RBF}}$	–	2.656	0.159	0.879	0.791
Optimization based on balanced accuracy					
LIBLINEAR+DS	5	6	**0.122**	0.864	0.896
PSO-LinSVM	1	33.606	**0.126**	0.853	0.901
Particle Swarm Optimization					
LIBLINEAR	6	64.458	0.169	0.862	0.791
LibSVM$^{\text{RBF}}$	–	0	0.15	0.785	0.934
LibSVM$^{\text{Sigmoid}}$	–	9.159	0.159	0.922	0.769
LS-SVM$^{\text{Poly}}$	–	3.201	0.43	0.69	0.418
LS-SVM$^{\text{RBF}}$	–	10.327	0.164	0.879	0.78
Simulated Annealing					
LIBLINEAR	2	45.864	0.155	0.871	0.813
LibSVM$^{\text{RBF}}$	–	0	0.15	0.785	0.934
LibSVM$^{\text{Sigmoid}}$	–	16.945	0.159	0.871	0.89
LS-SVM$^{\text{Poly}}$	–	0.4	0.193	0.897	0.692
LS-SVM$^{\text{RBF}}$	–	4.363	0.145	0.897	0.802

Table 4 presents experimental results obtained with Australian dataset. Again, linear SVM classifiers outperformed other SVM classifiers. LS-SVM$^{\text{RBF}}$ showed similar results, while it outperformed other classifiers in balanced CV based optimization. PSO-LinSVM with L2-regularized L2-loss support vector classification (dual) classifier again proved to be best choice; however, direct search resulted in highest performance. Note that in both experiments PSO-LinSVM obtained the same classifiers in both cases of fitness based on accuracy and balanced CV evaluation. The second case also proved to be a reasonable choice in general – both Table 3 and Table 4 show that application of such this approach often resulted in increased accuracy compared to accuracy-based optimization; this is especially seen with classifiers developed using simulated annealing approach.

This approach identified best parameter sets for classifiers better than or equal to accuracy-based evaluation approach in almost all cases for Australian dataset except LS-SVM$^{\text{Poly}}$ (notable that this classifier also did not perform well with PSO optimization technique for this dataset).

5 Conclusions and future works

Support vector machines techniques are powerful nonparametric techniques which can perform efficient classification and obtain results comparable to neural networks. This

article presents new particle swarm optimization and linear SVM based approach which can be applied for both small scale and large scale classification tasks. This technique uses particle swarm optimization based heuristic to select best performing SVM classifier from a set of linear classifiers with the same parameters. Its comparison with iterative parameter search showed that it is able to obtain SVM configuration resulting in better classification performance in terms of accuracy and identification of each class. An approach for classifier evaluation based on sum of true positive ratios is proposed together with this algorithm; it is more suitable for imbalanced learning as it tries to maximize classification performance instead of often applied accuracy. Empirical results showed that it can produce similar results to accuracy-based parameter selection approach. It is also shown that this approach can be a competitively efficient solution for classification problems, compared to similar SVM based techniques. Future work will concern more detailed investigation of PSO-LinSVM parameters, improvements in PSO algorithm, such as topologies or application of particle clusters [19], possible effects which come from sequential ordering or initialization, together with other possible enhancements.

References

1. L. Balthazar, *From Basel 1 to Basel 3: The Integration of State-of-the-Art Risk Modeling in Banking Regulation*, Palgrave Macmillan, 2006.
2. A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, G. Rätsch, Support vector machines and kernels for computational biology, *PLoS Computational Biology*, **4**(10), 10 pp., 2008.
3. Z.R. Yang, Biological applications of support vector machines, *Brief Bioinformatics*, **5**(4):328–338, 2004.
4. T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998, pp. 137–142.
5. A. Khan, B. Baharudin, L.H. Lee, Kh. Khan, A Review of Machine Learning Algorithms for Text-Documents Classification, *Journal of Advances in Information Technology*, **1**(1):4–20, 2010.
6. O. Chapelle, P. Haffner, V.N. Vapnik, Support vector machines for histogram-based image classification, *IEEE Trans. Neural Networks*, **10**(5):1055–1064, 1999.
7. J.-M. Ramirez-Cortes, P. Gomez-Gil, V. Alarcon-Aquino, D. Baez-Lopez, R. Enriquez-Caldera, A biometric system based on neural networks and SVM using morphological feature extraction from hand-shape images, *Informatica*, **22**(2):225–240, 2011.
8. M.Y. Chen, Bankruptcy prediction in firms with statistical and intelligent techniques and a comparison of evolutionary computation approaches, *Comput. Math. Applications*, **62**(12):4514–4524, 2011.
9. S. Chen, W. Härdle, R.A. Moro, Estimation of default probabilities with support vector machines, SFB 649 Discussion Paper 2006-077, 2006.
10. W.-D. Chen, J.-M. Li, A model based on factor analysis and support vector machine for credit risk identification in small-and-medium enterprises, in: *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, 2009, pp. 913–918.

11. J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.*, **9**(3):293–300, 1999.

12. K.K. Lai, L. Yu, L. Zhou, S. Wang, Credit risk evaluation with least square support vector machine, in: *Rough Sets and Knowledge Technology. First International Conference, RSKT 2006, Chongquing, China, July 24–26, 2006. Proceedings*, Lect. Notes Comput. Sci., Vol. 4062, Springer, 2006, pp. 490–495.

13. L. Zhou, K.K. Lai, Weighted LS-SVM credit scoring models with AUC maximization by direct search, in: *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, 2009, pp. 7–11.

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

15. L. Yun, Q. Cao, H. Zhang, Application of the PSO-SVM model for credit scoring, in: *Proceedings of the 7th International Conference on Computational Intelligence and Security*, 2011, pp. 47–51.

16. C.-C.Chang, C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.*, **2**(3), Article 27, 2011.

17. H. Ahn, K. Lee, Global optimization of support vector machines using genetic algorithms for bankruptcy prediction, in: *Neural Information Processing. 13th International Conference, ICONIP 2006, Hong Kong, China, October 3–6, 2006. Proceedings*, Part III, Lect. Notes Comput. Sci., Vol. 4234, Springer, 2006, pp. 420–429.

18. C. Wu, G. Tzeng, Y. Goo, W. Fang, A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy, *Expert Syst. Appl.*, **32**(2):397–408, 2007.

19. J. Kennedy, R.C. Eberhart, Y. Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2001.

20. C. Yongqi, LS_SVM parameters selection based on hybrid complex particle swarm optimization, *Energy Procedia*, **17**:706–710, 2012.

21. L. Li-xia, Z. Yi-qi, X. Liu, Tax forecasting theory and model based on SVM optimized by PSO, *Expert Syst. Appl.*, **38**(1):116–120, 2011.

22. X. Wang. Corporate financial warning model based on PSO and SVM, in: *Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS)*, 2010, 5 pp.

23. F.-L. Chen, F.-C. Li, Combination of feature selection approaches with SVM in credit scoring, *Expert Syst. Appl.*, **37**(7):4902–4909, 2010.

24. A. Unler, A. Murat, R.B. Chinnam, mr2PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification, *Inf. Sci.*, **181**(20):4625–4641, 2011.

25. C.-L. Huang, J.-F. Dun, A distributed PSO–VM hybrid system with feature selection and parameter optimization, *Applied Soft Computing*, **8**(4):1381–1391, 2008.

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

27. Q.-Z. Yao, J. Cai, J.-L. Zhang, Simultaneous feature selection and LS-SVM parameters optimization algorithm based on PSO, in: *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering, Los Angeles, California USA, March 31–April 02*, 2009, pp. 723–727.

28. P. Danenas, G. Garsva, Support vector machines and their application in credit risk evaluation process, *Transformations in Business & Economics*, **8**(3), Suppl. B:57–69, 2009.

29. W.-Y. Lin, Y.-H. Hu, C.-F. Tsai, Machine learning in financial crisis prediction: A survey, *IEEE Trans. Syst. Man Cybern., C*, **42**(4):421–436, 2011.

30. LIBLINEAR – a Library for large linear classification, `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

31. C.-W. Hsu, Ch.-Ch. Chang, Ch.-J. Lin, A practical guide to support vector classification, `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

32. Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, C.-J. Lin, Training and testing low-degree polynomial data mappings via linear SVM, *J. Mach. Learn. Res.*, **11**:1471–1490, 2010.

33. P. Danenas, G. Garsva, Credit risk evaluation using SVM-based classifier, *Business Information Systems Workshops, BIS 2010, Berlin, Germany, May 3–5, 2010. Revised Papers*, Lecture Notes in Business Information Processing, Vol. 57, Springer, Berlin, 2010, pp. 7–12.

34. P. Danenas, G. Garsva, S. Gudas, Credit risk evaluation model development using support vector based classifiers, in: *Procedia Comput. Sci., International Conference on Computational Science, ICCS 2011*, **4**:1699–1707, 2011.

35. J. Wu, A fast dual method for HIK SVM learning, in: *Proceedings of the 11th European conference on Computer vision (ECCV'10)*, Part II, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 552–565.

36. P. Danenas, G. Garsva, Credit risk evaluation modeling using evolutionary linear SVM classifiers and sliding window approach, in: *Procedia Comput. Sci., International Conference on Computational Science, ICCS 2012*, **9**:1324–1333, 2012.

37. P. Danenas, G. Garsva, PSO-based linear SVM classifier selection for credit risk evaluation modeling process, *Proceedings of 14th International Conference on Enterprise Information Systems (ICEIS 2012)*, Vol. 1, SciTePress, 2012, pp. 338–341.

38. T. M. Huang, V. Kecman, I. Kopriva, *Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-Supervised, and Unsupervised Learning*, J. Kacprzyk (Ed.), Stud. Computat. Intell., Vol. 17, Springer, 2006.

39. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2000.

40. R. Fan, K. Chang, C. Hsieh, X. Wang, C. Lin, LIBLINEAR: A library for large linear classification, *J. Mach. Learn. Res.*, **9**:1871–1874, 2008.

41. C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S.S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, 2008, pp. 408–415.

42. C.-J. Lin, R.C. Weng, S.S. Keerthi, Trust region Newton method for large-scale logistic regression, *J. Mach. Learn. Res.*, **9**:627–650, 2008.

43. A.P. Engelbrecht, *Computational Intelligence: An Introduction* 2nd edition, John Wiley & Sons, 2007.