Optimization Letters manuscript No.
(will be inserted by the editor)

Heuristic Approaches for Support Vector Machines with the Ramp Loss

Emilio Carrizosa · Amaya Nogales-Gómez · Dolores Romero Morales

the date of receipt and acceptance should be inserted later

Abstract Recently, Support Vector Machines with the ramp loss (RLM) have attracted attention from the computational point of view. In this technical note, we propose two heuristics, the first one based on solving the continuous relaxation of a Mixed Integer Nonlinear formulation of the RLM and the second one based on the training of an SVM classifier on a reduced dataset identified by an integer linear problem. Our computational results illustrate the ability of our heuristics to handle datasets of much larger size than those previously addressed in the literature.

Keywords Support Vector Machines · ramp loss · Mixed Integer Nonlinear Programming · heuristics.

This work has been partially supported projects MTM2012-36163, Ministerio de Economía y Competitividad, Spain, and FQM-329 of Junta de Andalucía, Spain, both with EU ERD Funds.

E. Carrizosa
Departamento de Estadística e Investigación Operativa
Facultad de Matemáticas
Universidad de Sevilla
E-mail: ecarrizosa@us.es

A. Nogales-Gómez
Departamento de Estadística e Investigación Operativa
Facultad de Matemáticas
Universidad de Sevilla
E-mail: ecarrizosa@us.es

D. Romero Morales
Saïd Business School
University of Oxford
United Kingdom
E-mail: dolores.romero-morales@sbs.ox.ac.uk

1 Introduction

In *Supervised Classification*, [16,17,27], we are given a set of objects Ω partitioned into classes and the aim is to build a procedure for classifying new objects. In its simplest form, each object $i \in \Omega$ has associated a pair (x_i, y_i), where the predictor vector x_i takes values on a set $X \subseteq \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ is the class membership of object i. See [2–4,10,14,15,19,20] for successful applications of Supervised Classification.

Support Vector Machines (SVM), [12,24,25], have proved to be one of the state-of-the-art methods for Supervised Classification. The SVM aims at separating both classes by means of a hyperplane, $\omega^\top x + b = 0$, found by solving the following optimization problem:

$$\min_{\omega \in \mathbb{R}^d,\, b \in \mathbb{R}} \omega^\top \omega/2 + \frac{C}{n} \sum_{i=1}^{n} g((1 - y_i(\omega^\top x_i + b))^+),$$

where n is the size of the sample used to build the classifier, $(1 - y_i(\omega^\top x_i + b))^+ = \max\{1 - y_i(\omega^\top x_i + b), 0\}$, C is a nonnegative parameter, and g a nondecreasing function in $\mathbb{R}_+$, the so-called *loss* function. The most popular choices for g, namely the *hinge* loss, $g(t) = t$, and the squared hinge loss, $g(t) = t^2$, are convex functions. These convex loss functions yield smooth convex optimization problems, in fact, convex quadratic, which have been addressed in the literature by a collection of competitive algorithms. See [8] for a recent review on Mathematical Optimization and the SVMs. More challenging optimization problems arise when solving the SVM with non-convex loss functions, see e.g., [11,22,23,28].

In this technical note, we are interested in the SVM with the so-called *ramp loss* function, $g(t) = (\min\{t, 2\})^+$, [11,23]. From the computational perspective, a first attempt to study the SVM with the ramp loss is presented in [18], where the problem is formulated as a Mixed Integer Nonlinear Programming (MINLP) problem, and datasets with up to $n = 100$ objects are solved with a commercial software package. The state-of-the-art algorithm is given in [6], where the ramp loss model, (RLM), is formulated as the following MINLP problem

$$\min_{\omega, b, \xi, z} \omega^\top \omega/2 + \frac{C}{n} \left(\sum_{i=1}^{n} \xi_i + 2 \sum_{i=1}^{n} z_i \right)$$

s.t. (RLM)

$$\begin{aligned} y_i(\omega^\top x_i + b) &\geq 1 - \xi_i - M z_i \quad \forall i = 1, \ldots, n \\ 0 \leq \xi_i &\leq 2 \quad \forall i = 1, \ldots, n \\ z &\in \{0,1\}^n \\ \omega &\in \mathbb{R}^d \\ b &\in \mathbb{R}, \end{aligned} \tag{1}$$

with M a big enough constant. See [6] for further details on this formulation, called there the SVMIP1(ramp), and [13,26] for related models. This MINLP

formulation has a quadratic term of dimension d and n binary variables, and is therefore challenging from the computational point of view. In [6], datasets with up to 500 objects are solved to optimality. In this technical note, we propose two heuristics for the RLM that can handle datasets of larger size, and compare them to the state-of-the-art algorithm. The first one is based on the continuous relaxation of the RLM, therefore, it is a cheap heuristic (its computation time is comparable to training an SVM). The second heuristic is based on training an SVM on a reduced dataset identified by an integer linear problem. At the expense of higher running times, and as our computational results will illustrate, this procedure behaves much better in terms of classification accuracy than the other two.

The remainder of the note is organized as follows. In Section 2, the two heuristics and the state-of-the-art algorithm for the RLM are described in more detail. In Section 3, we report our computational results using both synthetic and real-life datasets. We end this note in Section 4 with some conclusions and topics for future research.

2 Heuristic Approaches

In this section, we describe three heuristics, Heuristics 1, 2, 3, whose descriptions can be found in Figures 2–4, respectively. Heuristics 1 and 2 are our proposals while Heuristic 3 is the state-of-the-art algorithm with a time limit [6]. Before describing the heuristics, we present in Figure 1 a procedure that exploits the nature of the RLM formulation to build feasible solutions starting from a partial solution (ω, b). In short, the RLM can be written as a two-stage problem, in which we first choose the classifier, defined by ω and b, and then fill in the variables ξ and z.

Therefore, in the rest of this section, when describing the three heuristics, we will concentrate on explaining how to obtain the partial solution (ω, b) defining the classifier. The corresponding ξ and z will be derived using the fill-in procedure in Figure 1.

Step 1. Let a classifier be defined by ω and b.
Step 2. For each i, fill in variables ξ_i and z_i as follows:
Case I. If $y_i(\omega^\top x_i + b) > 1$, then set $\xi_i = 0,\ z_i = 0$.
Case II. If $-1 \leq y_i(\omega^\top x_i + b) \leq 1$, then set $\xi_i = 1 - y_i(\omega^\top x_i + b),\ z_i = 0$.
Case III. If $y_i(\omega^\top x_i + b) < -1$, then set $\xi_i = 0,\ z_i = 1$.
Step 3. $(\omega,\ b,\ \xi,\ z)$ is a feasible solution for the RLM.

Fig. 1: Fill-in Procedure for the RLM

We now describe Heuristic 1, see Figure 2. In Heuristic 1, we first construct the continuous relaxation of the RLM, where the integrality constraints (1) are replaced by $z \in [0,1]^n$, and return as (partial) solution (ω, b) to the RLM the optimal classifier of this relaxation. This is a cheap and naïve heuristic, which consists of solving a convex quadratic problem with linear constraints, and our computational results show that it performs well in the datasets tested.

Step 1. `Solve the continuous relaxation of the RLM, yielding a (partial) solution` (ω, b).
Step 2. `Fill in` (ω, b) `as described in Figure 1.`

Fig. 2: Description of Heuristic 1

Heuristic 2 is based on the optimization of an integer linear problem, easier to solve than the RLM since neither the quadratic term $\omega^\top \omega/2$ nor the variables ξ are present. Let us consider the Linear Separability Problem (LSP), which aims to find the minimum number of objects to be taken off to make the sets $\{x_i,\, y_i = 1\}$ and $\{x_i,\, y_i = -1\}$ linearly separable. For each object i, let us define the binary variable α_i taking the value 1 if object i is removed, and 0 otherwise. Now, the LSP can be formulated as:

$$\min_{\omega,b,\alpha} \quad \sum_{i=1}^{n} \alpha_i$$

s.t. (LSP)

$$\begin{aligned} y_i(\omega^\top x_i + b) &\geq 1 - M\alpha_i \quad \forall i = 1, \ldots, n \\ \alpha &\in \{0,1\}^n \\ \omega &\in \mathbb{R}^d \\ b &\in \mathbb{R}, \end{aligned}$$

where M is a big enough constant.

Heuristic 2 works as shown in Figure 3. First, a solution to the LSP is used to define a reduced set, in which only objects with $\alpha_i = 0$ are considered. Second, an SVM is trained on the reduced set, yielding a (partial) solution (ω, b) to the RLM. Third, (ω, b) is used as initial solution in a branch and bound ($B\&B$) procedure for the RLM, which is truncated by a time limit.

To end, we describe Heuristic 3, see Figure 4. This heuristic is based on solving the RLM by a $B\&B$ procedure, enhanced with an initial solution, and possibly at each node with a heuristic procedure. In [6], two initial solutions are proposed, the so-called *zero solution* ($ini^{\omega=0}$) and the so-called *zero error solution* ($ini^{z=0}$). (Please note that there is a third heuristic solution in [6]. Since it is computationally very expensive, and therefore, impractical for large

Run for $tlim_2$ seconds:

Step 1. Solve the LSP and let (ω', b', α') be the solution vector obtained.
Step 2. The solution to the LSP is used to define a reduced set, in which only objects with $\alpha'_i = 0$ are considered.
Step 3. An SVM is trained on the reduced set, yielding a (partial) solution (ω, b) to the RLM.
Step 4. Fill in (ω, b) as described in Figure 1.
Step 5. (ω, b, ξ, z) is used as initial solution in a $B\&B$ procedure.

Fig. 3: Description of Heuristic 2

datasets, it is not considered in our tests.) The *zero solution* is derived by setting $\omega = 0$, and $b = 1$ if $\text{card}\{i, y_i = 1\} > \text{card}\{i, y_i = -1\}$ and $b = -1$ otherwise. The *zero error solution* involves solving a quadratic problem consisting of the RLM with $z_i = 0 \ \ \forall i = 1, \ldots, n$. In [6], a heuristic procedure is applied at each node aiming at improving the best upper bound. This node improvement consists of constructing a feasible solution to the RLM by applying the fill-in procedure in Figure 1 to the classifier returned by the continuous relaxation. If the objective value of this new solution improves the best upper bound, this bound will be updated. Note that our Heuristic 1 is a cheap heuristic where such a fill-in procedure is only applied at the root node. As for Heuristic 2, Heuristic 3 is run until a time limit is reached. Note that the three heuristics are matheuristics, [21], since their procedures require solving nontrivial optimization problems.

Run for $tlim_3$ seconds:

Step 1. Let $NI \in \{0, 1\}$.
Step 2. Let (ω, b) be an initial (partial) solution.
Step 3. Fill in (ω, b) as described in Figure 1.
Step 4. (ω, b, z, ξ) is used as initial solution in a $B\&B$ procedure, where if NI=1, Heuristic 1 is applied in each node with the aim of improving the current best feasible solution.

Fig. 4: Description of Heuristic 3

3 Computational Results

This section is aimed to illustrate the performance of our heuristics, Heuristics 1 and 2, with respect to Heuristic 3. The rest of this section is structured as follows. The datasets used to compare these heuristics are described in Section 3.1. The tuning procedure used to choose the tradeoff parameter C is given in Section 3.2. Finally, the computational results are presented in Section 3.3.

Our experiments have been conducted on a PC with an Intel® Core™ i7 processor, 16 Gb of RAM. We use the optimization engine CPLEX v12.3, [1], for solving all optimization problems.

3.1 Datasets

To illustrate the capacity of our heuristics to handle larger dataset sizes than Heuristic 3, we use both synthetic and real-life datasets. In [6], Heuristic 3 was tested on two synthetic datasets obtained using the so-called `TypeA` and `TypeB` generators, and 11 real-life datasets from the UCI repository [5]. We use both the `TypeA` and `TypeB` generators (for d equal to 2, 5 and 10), as well as the dataset `adult`, the only real-life one tested in [6] with a size larger than 5000 objects. In addition, we also report results on three other large UCI datasets, see [5,9].

A description of these datasets can be found in Table 1, whose first three columns give the dataset name, number of attributes (d) and total size of the dataset ($|\Omega|$). As customary in the literature of Supervised Classification, [8], each dataset is partitioned into the so-called, training, testing and validation sets. Large sizes are used for the training set, see the fourth column of Table 1, and the remaining records are equally split between the testing and validation sets. The fifth column of Table 1 reports the class split in the training sample. Finally, the last column reports the mean accuracy for the SVM (with the hinge loss), following the procedure described in the next section for the selection of the tradeoff parameter C.

3.2 Parameters Setting

As customary in Supervised Learning, the RLM contains parameter C in its formulation, which needs to be tuned. Following the usual approach, [8], C is tuned by inspecting a grid of 26 values of the form $C = 2^k$, such that $2^{-13} \leq \frac{C}{n} \leq 2^{13}$. The tuning for a given heuristic works as follows. The dataset is split into three sets, called training, testing and validation set. For each value of C the heuristic is run on the training set, yielding the classifier given by (ω_C, b_C). The different classifiers built in this way are compared according to their accuracy on the testing set. The parameter C^* for which (ω_{C^*}, b_{C^*}) yields the highest accuracy on the testing set is chosen. In the tables below, the accuracy on the validation set of the heuristic for (ω_{C^*}, b_{C^*}) is reported.

Table 1 Datasets

Name	d	$\lvert\Omega\rvert$	n	Class split	SVM
`TypeA`	2	15000	5000	50/50	51.62
`TypeA`	5	15000	5000	50/50	49.21
`TypeA`	10	15000	5000	50/50	48.22
`TypeB`	2	15000	5000	50/50	56.95
`TypeB`	5	15000	5000	50/50	51.73
`TypeB`	10	15000	5000	50/50	50.55
`gamma`	10	19020	10000	32/68	79.28
`adult`	123	30956	15000	24/76	84.88
`cod-rna`	8	59535	20000	33/67	93.87
`ijcnn1`	22	35000	20000	9/91	91.36

In order to make a fair comparison, overall time limits for both Heuristics 2 and 3 should be the same. Heuristic 2 involves solving one LSP plus 26 RLMs, which are aborted when a time limit is exceeded. In our experiments we choose $tlim_{\text{LSP}} = 300$ seconds for the LSP, and for each RLM $tlim_2$ is chosen as the closest integer to $\dfrac{d + \frac{n}{100}}{5}$. Heuristic 3 involves solving 26 RLMs, which are aborted after $tlim_3 = tlim_2 + \frac{tlim_{\text{LSP}}}{26}$.

3.3 Accuracy Results

To obtain sharp estimates for the accuracy of the different heuristics, repeated random subsampling validation is used, where ten instances are run for each dataset. For the synthetic datasets, the ten instances differ in the seed used to generate random data, whereas for the real ones, the seed is used to shuffle the set and then obtain different training, testing and validation sets. For each dataset and for each heuristic, Tables 2 and 3 report the mean validation accuracy across the ten instances, as well as the standard deviation and the median.

We have a whole array of variants of Heuristic 3 depending on the initial solution chosen and whether the node improvement procedure is applied. We can see that the simplest implementation of Heuristic 3, where we start with the *zero solution* and no node improvement is applied, dominates the rest of the variants in terms of mean and median accuracy, except for `TypeA` (`d` = 2) and `gamma`. In any case, Heuristic 2 outperforms any variant of Heuristic 3. Therefore, and unless stated, when referring to Heuristic 3 we will use the results mentioned above.

The following conclusions can be drawn from our computational results. In `TypeA`, Heuristic 2 outperforms Heuristic 3, while Heuristic 3 outperforms Heuristic 1. The increase in mean accuracy shown by Heuristic 2 compared to Heuristic 3 is more pronounced for larger values of d, being equal to 19.68

percentage points for $d = 10$. A similar behaviour is observed for the median accuracy. In `TypeB`, Heuristics 1 and 2 outperform Heuristic 3. Heuristic 2 outperforms Heuristic 1 in terms of median accuracy, where in terms of mean accuracy the same holds for $d = 2, 10$, while for $d = 5$ the mean accuracies are comparable. A closer look reveals that for larger d, any variant of Heuristic 3 has a median accuracy of 50%, the one given by the *zero solution*. Heuristics 1 and 2 have a similar accuracy in the real-life datasets, and they clearly outperform Heuristic 3 in three of the datasets. For the `gamma` dataset, the best mean accuracy of Heuristic 3 is achieved by giving the best among the *zero solution* and the *zero error solution* at the root node, as well as applying the node improvement procedure. For the `gamma` dataset, the increase in mean accuracy (in percentage points) from Heuristic 2 to Heuristic 3 is then equal to 4.62. For the datasets `adult` and `codrna`, the increase in mean accuracy is equal to 5.41 and 12.67, respectively. Similar dominance is observed when using the median accuracy. For the last real-life dataset, `ijcnn1`, the behaviour in terms of accuracy of the three heuristics is similar. Taking a closer look at this dataset, one can observe that classes are highly unbalanced, see Table 1, and therefore the accuracy of the three is very similar to that of the *zero solution*. A finer analysis, taking into account not only the overall accuracy, but also the sensitivity and specificity, reveals significant differences between the heuristics. Indeed, as shown in Table 4, while Heuristic 3 misclassifies all records in Class +1 (the class in minority), our procedures are slightly better for such class. If, instead of the overall average accuracy, we measure the (weighted) geometric mean of the accuracy in both classes, we see that our procedures clearly outperform Heuristic 3.

This shows that our heuristics can address, for a given time limit, larger datasets than Heuristic 3.

4 Conclusions

In this note we show that a quick heuristic, based on solving the continuous relaxation of the RLM, is competitive against the state-of-the-art algorithm for the SVM with the ramp loss, the RLM. Much better results are obtained with our so-called Heuristic 2, which involves solving an integer linear problem and a convex quadratic problem to obtain a good starting solution for the branch and bound procedure.

In order to solve problems of larger size, valid cuts strengthening the formulation would be extremely helpful. In this sense, [6] proposes the so-called geometric cuts, but also mentions that these cuts are only of interest in the trivial case of two-dimensional data. As done in [6], these cuts have not been employed in experiments. New and helpful cuts deserve further study. As done in [6], the algorithms proposed here can be extended to models which use the so-called kernels [8,12]. In [6], two kernels are tested on small datasets. An extension to large datasets requires a careful analysis due to the dramatic increase in computational burden caused by the choice of the kernel parameters,

[7]. Focusing our analysis on the linear kernel has allowed us to measure the performance of our heuristics avoiding the side-effects caused by the kernel choice.

Table 2 Validation sample accuracy in %

Name	*Heuristic* 1			*Heuristic* 2			*Heuristic* 3, $NI = 0$								
							$ini^{\omega=0}$			$ini^{z=0}$			$ini^{\omega=0}$ & $ini^{z=0}$		
	mean	*std*	*med*	*mean*	*std*	*med*	*mean*	*std*	*med*	*mean*	*std*	*med*	*mean*	*std*	*med*
`TypeA` ($d = 2$)	68.36	18.62	76.54	*83.78*	0.80	*83.84*	72.03	14.60	82.93	56.11	7.45	51.47	73.95	12.82	83.04
`TypeA` ($d = 5$)	54.86	10.83	53.18	*83.60*	0.85	*83.50*	66.33	8.70	71.46	64.41	9.28	68.18	56.91	7.75	52.52
`TypeA` ($d = 10$)	51.46	10.12	50.26	*79.79*	10.51	*82.98*	60.11	6.83	62.98	57.08	7.24	55.53	56.90	7.05	55.53
`TypeB` ($d = 2$)	70.63	9.29	73.22	*83.39*	1.12	*83.80*	56.09	9.37	51.33	56.09	9.37	51.33	56.09	9.37	51.33
`TypeB` ($d = 5$)	*70.71*	5.41	71.98	70.50	14.44	*76.88*	52.08	5.14	50.00	52.08	5.14	50.00	52.08	5.14	50.00
`TypeB` ($d = 10$)	63.99	8.84	63.84	*77.28*	9.87	*81.24*	51.24	3.35	50.00	51.24	3.35	50.00	51.24	3.35	50.00
`gamma`	79.26	0.45	79.33	*79.55*	0.74	*79.36*	68.30	0.91	68.61	68.30	0.91	68.61	68.30	0.91	68.61
`adult`	*84.91*	0.30	*84.98*	*84.91*	0.29	84.92	79.50	1.62	78.73	-	-	-	79.50	1.62	78.73
`codrna`	*93.88*	0.13	*93.90*	91.11	7.86	93.71	78.44	2.99	78.37	78.44	2.99	78.37	78.41	3.03	78.37
`ijcnn1`	91.35	0.37	91.33	*91.93*	0.44	*91.97*	90.17	0.36	90.03	90.17	0.36	90.03	90.17	0.36	90.03

Table 3 Validation sample accuracy in %

	Heuristic 3, $NI = 1$								
	$ini^{\omega=0}$			$ini^{z=0}$			$ini^{\omega=0}$ & $ini^{z=0}$		
	mean	*std*	*med*	*mean*	*std*	*med*	*mean*	*std*	*med*
`TypeA` ($d = 2$)	54.90	5.68	51.47	56.11	7.45	51.47	54.90	5.68	51.47
`TypeA` ($d = 5$)	54.49	5.84	50.94	54.78	6.31	50.94	54.49	5.84	50.94
`TypeA` ($d = 10$)	53.10	4.75	50.00	53.28	5.04	50.00	53.10	4.75	50.00
`TypeB` ($d = 2$)	55.95	9.45	50.81	56.09	9.37	51.33	55.95	9.45	50.81
`TypeB` ($d = 5$)	51.86	5.18	50.00	52.08	5.14	50.00	51.86	5.18	50.00
`TypeB` ($d = 10$)	50.01	0.02	50.00	51.24	3.35	50.00	50.01	0.02	50.00
`gamma`	64.86	0.64	64.87	74.85	0.67	75.10	74.93	0.72	75.29
`adult`	75.81	0.28	75.90	79.30	1.54	78.73	77.93	0.42	78.09
`codrna`	66.57	0.27	66.62	78.43	3.00	78.37	66.57	0.27	66.62
`ijcnn1`	90.17	0.36	90.03	90.17	0.36	90.03	90.17	0.36	90.03

Table 4 Validation sample accuracy in % for `ijcnn1`

Heuristic		Sensitivity			Specificity		
		mean	*std*	*med*	*mean*	*std*	*med*
1		18.81	2.29	18.22	99.25	0.17	99.24
2		29.66	5.11	29.80	98.71	0.55	98.92
3,$NI = 0$	$ini^{\omega=0}$	0.00	0.00	0.00	100.00	0.00	100.00
	$ini^{z=0}$	0.00	0.00	0.00	100.00	0.00	100.00
	$ini^{\omega=0}$ & $ini^{z=0}$	0.00	0.00	0.00	100.00	0.00	100.00
3,$NI = 1$	$ini^{\omega=0}$	0.00	0.00	0.00	100.00	0.00	100.00
	$ini^{z=0}$	0.00	0.00	0.00	100.00	0.00	100.00
	$ini^{\omega=0}$ & $ini^{z=0}$	0.00	0.00	0.00	100.00	0.00	100.00

References

1. IBM ILOG CPLEX (2012). Www-01.ibm.com/software/integration/optimization/cplex-optimizer
2. Apte, C.: The big (data) dig. OR/MS Today p. 24 (2003)
3. Baesens, B., Setiono, R., Mues, C., Vanthienen, J.: Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science **49**(3), 312–329 (2003)
4. Bertsimas, D., Bjarnadóttir, M., Kane, M., Kryder, J., Pandey, R., Vempala, S., Wang, G.: Algorithmic prediction of health-care costs. Operations Research **56**(6), 1382–1392 (2008)
5. Blake, C., Merz, C.: UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998). University of California, Irvine, Department of Information and Computer Sciences
6. Brooks, J.P.: Support vector machines with the ramp loss and the hard margin loss. Operations Research **59**(2), 467–479 (2011)
7. Carrizosa, E., Martín-Barragán, B., Romero Morales, D.: Variable neighborhood search for parameter tuning in support vector machines. Tech. rep. (2012)
8. Carrizosa, E., Romero Morales, D.: Supervised classification and mathematical optimization. Computers and Operations Research **40**, 150–165 (2013)
9. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 1–27 (2011)
10. Chaovalitwongse, W.A., Fan, Y.J., Sachdeo, R.C.: Novel optimization models for abnormal brain activity classification. Operations Research **56**(6), 1450–1460 (2008)
11. Collobert, R., Sinz, F., Weston, J., Bottou, L.: Trading convexity for scalability. In: Proceedings of the 23rd International Conference on Machine Learning, ICML06, pp. 201–208. New York, USA (2006)
12. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press (2000)
13. Ertekin, S., Bottou, L., Giles, C.L.: Nonconvex online support vector machines. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(2), 368–381 (2011)
14. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46**, 389–422 (2002)
15. Han, J., Altman, R., Kumar, V., Mannila, H., Pregibon, D.: Emerging scientific applications in data mining. Communications of the ACM **45**(8), 54–58 (2002)
16. Hand, H., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
17. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
18. Liu, Y., Wu, Y.: Optimizing ψ-learning via mixed integer programming. Statistica Sinica **16**, 441–457 (2006)

19. Loveman, G.: Diamonds in the data mine **81**(5), 109–113 (2003)
20. Mangasarian, O., Street, W., Wolberg, W.: Breast cancer diagnosis and prognosis via linear programming. Operations Research **43**(4), 570–577 (1995)
21. Maniezzo, V., Stützle, T., Voss, S. (eds.): Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, *Annals of Information Systems*, vol. 10. Springer (2009)
22. Orsenigo, C., Vercellis, C.: Multivariate classification trees based on minimum features discrete support vector machines. IMA Journal of Management Mathematics **14**(3), 221–234 (2003)
23. Shen, X., Tseng, G.C., Zhang, X., Wong, W.H.: On ψ-learning. Journal of the American Statistical Association **98**, 724–734 (2003)
24. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag (1995)
25. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
26. Wang, L., Jia, H., Li, J.: Training robust support vector machine with smooth ramp loss in the primal space. Neurocomputing **71**(13-15), 3020–3025 (2008)
27. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowledge and Information Systems **14**, 1–37 (2007)
28. Wu, Y., Liu, Y.: Robust truncated hinge loss support vector machines. Journal of the American Statistical Association **102**(479), 974–983 (2007)