Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 382 (2007) 262–279

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the complexity of working set selection[☆]

Hans Ulrich Simon

Fakultät für Mathematik, Ruhr-Universität Bochum, D-44780 Bochum, Germany

Abstract

The decomposition method is currently one of the major methods for solving the convex quadratic optimization problems being associated with Support Vector Machines (SVM-optimization). A key issue in this approach is the policy for working set selection. We would like to find policies that realize (as well as possible) three goals simultaneously: "(fast) convergence to an optimal solution", "efficient procedures for working set selection", and "a high degree of generality" (including typical variants of SVM-optimization as special cases). In this paper, we study a general policy for working set selection that has been proposed in [Nikolas List, Hans Ulrich Simon, A general convergence theorem for the decomposition method, in: Proceedings of the 17th Annual Conference on Computational Learning Theory, 2004, pp. 363–377] and further analyzed in [Nikolas List, Hans Ulrich Simon, General polynomial time decomposition algorithms, in: Proceedings of the 17th Annual Conference on Computational Learning Theory, 2005, pp. 308–322]. It is known that it efficiently approaches feasible solutions with minimum cost for any convex quadratic optimization problem. Here, we investigate its computational complexity when it is used for SVM-optimization. It turns out that, for a variable size of the working set, the general policy poses an NP-hard working set selection problem. But a slight variation of it (sharing the convergence properties with the original policy) can be solved in polynomial time. For working sets of fixed size 2, the situation is even better. In this case, the general policy coincides with the "rate certifying pair approach" (introduced by Hush and Scovel). We show that maximum rate certifying pairs can be found in linear time, which leads to a quite efficient decomposition method with a polynomial convergence rate for SVM-optimization.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Support vector machines; Decomposition method; Working set selection; Approximation algorithms

1. Introduction

We begin with some remarks that clarify why the decomposition method is used for the purpose of SVM-optimization. Although (more or less) the same remarks are found in other papers that deal with the decomposition method (like [25] for example), we repeat them here for sake of convenience.

Support vector machines (SVMs), as introduced by Vapnik and co-workers [1,34], are a promising technique for classification, function approximation, and other key problems in statistical learning theory. In this paper, we consider the optimization problems that are induced by SVMs, which are special cases of convex quadratic optimization.

[☆] This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. This work was furthermore supported by the Deutsche Forschungsgemeinschaft Grant SI 498/7-1.

E-mail address: simon@lmi.rub.de.

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.03.041

The difficulty of solving problems of this kind comes from the density of the matrix that represents the “quadratic part” of the cost function. Thus, a prohibitive amount of memory is required to store the matrix, and traditional optimization algorithms (such as Newton, for example) cannot be directly applied. Several authors have proposed (different variants of) a decomposition method to overcome this difficulty [30,11,31,32,27,28,3,14,20,4,29,15,13,21, 22,17,7,18,9,25,24,5,6]. This method keeps track of a current feasible solution, which is iteratively improved. In each iteration, the variable indices are split into a “working set” $I \subseteq \{1, \dots, m\}$ and its complement $J = \{1, \dots, m\} \setminus I$. Then, the subproblem with variables x_i, $i \in I$, is solved, thereby leaving the values for the remaining variables x_j, $j \in J$, unchanged. The success of the method depends in a quite sensitive manner on the policy for the selection of the working set I (whose size is typically much smaller than m). Ideally, the selection procedure should be computationally efficient and, at the same time, effective in the sense that the resulting sequence of feasible solutions converges (with high speed) to an optimal limit point. Clearly, these goals are conflicting in general, and trade-offs are to be expected.

1.1. Our results

We study a general policy for working set selection that has been proposed in [25] and further analyzed in [26]. It is known that it efficiently approaches feasible solutions with minimum cost for any convex quadratic optimization problem. Here, we investigate its computational complexity when it is used for SVM-optimization. As a “test-case”, we discuss one of the most popular and well-studied SVM-optimization problems.[1]

It turns out that, for *variable size of the working set*, the general policy poses an NP-hard working set selection problem (see Section 4). But a slight variation of it (replacing in some sense “Exact Working Set Selection” by “Approximate Working Set Selection”) allows for a convergence proof (valid for convex quadratic optimization in general; see Section 3) while posing a polynomially approximable working set selection problem as far as SVM-optimization is concerned (see Section 5). In particular, we show that any approximation algorithm with performance ratio c for the well-known “Knapsack Problem” can be converted (without much loss of efficiency) into an approximation algorithm with performance ratio $2c$ for “Approximate Working Set Selection”. If we start, for instance, with the greedy heuristic with performance ratio 2 for “Knapsack”, we obtain a heuristic with performance ratio 4 for “Approximate Working Set Selection”. Furthermore, we design a full polynomial time approximation scheme (FPTAS) for “Approximate Working Set Selection”.[2]

For *working sets of fixed size* 2, the situation is even better. In this case, the general policy coincides with the “rate certifying pair approach” (introduced by Hush and Scovel [9]). In Section 6, we design a new algorithm that finds “maximum rate certifying pairs” in linear time (thereby improving on a previous algorithm by Hush and Scovel by factor $\log m$), which leads to a quite efficient decomposition method with a polynomial convergence rate as far as SVM-optimization is concerned.

An extended abstract with (some of) the results of this paper is found in [33].

1.2. Related work on decomposition algorithms

One of the most interesting debates concerning decomposition algorithms in connection with SVM-optimization deals with the question how the “rate certifying pair approach” compares to the “violating pair approach”. The latter is perhaps the most prominent one. It is based on the selection of the maximally KKT-violating pairs as implemented, for example, in SVM^{light} [11] or LIBSVM [4]. A paper by Lin [19] seems to imply the following quite strong result for SVM-optimization (although not stating it explicitly): decomposition algorithms following the approach of maximally KKT-violating pairs are within ε of optimality after only $O(\log 1/\varepsilon)$ iterations.[3] The analysis has to assume the strict convexity of the objective function and some non-degeneracy conditions. The convergence rate is only given in terms of the distance, ε, to the optimal value, whereas the dependence on other problem parameters (like, for example, the

[1] The dual problem to maximum margin classification with the 1-norm soft margin criterion.

[2] Although the time bound for the FPTAS is not satisfactory for practical purposes, this result might be of some interest in its own right, because the problem does not fit into the framework of “independence schemes” introduced by Korte and Schrader [16]. Thus, one cannot design the FPTAS in the style of the well-known FPTAS for “Knapsack”. We come back to this discussion in Section 5.

[3] See [5] for a generalization of this result to similar but more general policies for working set selection.

number of variables) is not clarified. Recent work [9,26] on the "rate certifying pair approach" (including this paper) provides theoretical guarantees that look at least as good as the best known theoretical guarantees for the "violating pair approach". Since the latter approach is empirically quite successful, this might, however, merely indicate that the "violating pair approach" is not yet fully understood from a theoretical point of view.

2. Preliminaries

In the subsequent sections, we shall study a policy for working set selection that was suggested in [25]. For this reason, we introduce in this section some notations that are borrowed from [25].

Throughout this paper,

$$f(x) = \frac{1}{2}x^\top Qx - w^\top x = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} Q_{i,j}x_i x_j - \sum_{i=1}^{m} w_i x_i \tag{1}$$

denotes a convex cost function, where $Q \in \mathbb{R}^{m\times m}$ is a positive semi-definite matrix over the reals. We are interested in optimization problems of the following general form $\mathcal{P}$:

$$\min_x f(x) \text{ s.t. } Ax = b, l \le x \le r. \tag{2}$$

Here, $A \in \mathbb{R}^{k\times m}$, $b \in \mathbb{R}^k$, $l, r \in \mathbb{R}^m$, and $l \le x \le r$ is the short-notation for the "box constraints"

$$\forall i = 1, \dots, m : l_i \le x_i \le r_i .$$

In the sequel,

$$R(\mathcal{P}) = \{x \in \mathbb{R}^m \mid Ax = b, l \le x \le r\}$$

denotes the set of feasible points for $\mathcal{P}$. Note that f is bounded on $R(\mathcal{P})$ because f is continuous and $R(\mathcal{P})$ is compact.

We briefly note that any bounded[4] optimization problem with cost function $f(x)$ and linear equality- and inequality-constraints can be brought into the form (2), because we may convert the linear inequalities into linear equations by introducing non-negative slack variables. By the compactness of the region of feasible points, we may also put a suitable upper bound on each slack variable such that finally all linear inequalities take the form of box constraints.

Example 1. A popular (and actually possibly the most well studied) variant of Support Vector Machines leads to an optimization problem of the following form $\mathcal{P}_0$:

$$\min_x f(x) \text{ s.t. } y^\top x = 0\ ,\ l \le x \le r. \tag{3}$$

Here, $y \in \{-1, 1\}^m$ is a vector whose components represent binary classification labels. Note that we may "normalize" $\mathcal{P}_0$ by substituting $y_i x_i$ for x_i. This leads to a problem of the same form as $\mathcal{P}_0$, but with $y = \bar{1}$, where $\bar{1}$ denotes the "all-ones" vector. (An analogous notational convention is used for any other constant.) The main difference between $\mathcal{P}_0$ and the general problem $\mathcal{P}$ is that $\mathcal{P}_0$ has only the single equality constraint $y^\top x = 0$.[5]

The *decomposition method* with working sets of size at most q proceeds iteratively as follows: given a feasible solution $x \in R(\mathcal{P})$ (chosen arbitrarily in the beginning), a so-called working set $I \subseteq \{1, \dots, m\}$ of size at most q is selected. Then x is updated by the optimal solution for the subproblem with variables x_i, $i \in I$ (leaving the values x_j with $j \notin I$ unchanged). The policy for working set selection is a critical issue that we briefly discuss in the next section.

[4] Here, "bounded" means that the region of feasible points is compact (or can be made compact without changing the smallest possible cost).

[5] As far as the SVM-application is concerned, we could also set $w = \bar{1}$, $l = \bar{0}$, and $r = \bar{C}$ for some constant $C > 0$. Since these settings do not substantially simplify the problem, we prefer the more general notation.

3. Working set selection and convergence

Consider the situation where we attack a problem $\mathcal{P}$ of the general form (2) by means of the decomposition method. As explained below, the selection of a working set $I \subseteq \{1, \dots, m\}$ of size at most q can be guided by a function family $C_I(x)$. For instance, the following family (with h ranging over $\mathbb{R}^k$) was considered in [25]:

$$C_I(x) := \inf_h \left(\sum_{i \in I} (x_i - l_i) \max\{0, (\nabla f(x))_i - A_i^\top h\} + (r_i - x_i) \max\{0, A_i^\top h - (\nabla f(x))_i\} \right). \qquad (4)$$

Here, A_i denotes the i'th column of A and $A_i^\top$ its transpose.

The following results, being valid for any optimization problem $\mathcal{P}$ of the form (2), were shown in [25]:

- The function family $C_I(x)$ defined in (4) satisfies the following conditions:

(C1) For each $I \subseteq \{1, \dots, m\}$ such that $|I| \le q$, $C_I(x)$ is continuous on $R(\mathcal{P})$.

(C2) If $|I| \le q$ and x' is an optimal solution for the subproblem induced by the current feasible solution x and working set I, then $C_I(x') = 0$.

(C3) If x is not an optimal solution for $\mathcal{P}$, then there exists an $I \subseteq \{1, \dots, m\}$ such that $|I| \le q$ and $C_I(x) > 0$ provided that $q \ge k + 1$.

- Let $q \ge k + 1$ (where k is the number of equality constraints in $\mathcal{P}$). Assume that $Q \in \mathbb{R}^{m \times m}$ (the matrix from the definition of the cost function in (1)) has positive definite sub-matrices $Q_{I,I}$ for all subsets $I \subseteq \{1, \dots, m\}$ of size at most q.[6] Let $C_I(x)$ be a family of functions[7] that satisfies conditions (C1), (C2), (C3). Let a *decomposition method induced by* $C_I(x)$ be an algorithm that, given the current feasible solution x, always chooses a working set $I \subseteq \{1, \dots, m\}$ of size at most q that maximizes $C_I(x)$. Then this method leads to a feasible solution of smallest cost in the limit.

Remark 2. It is interesting to note that the decomposition method induced by the family $C_I(x)$ from (4) coincides with the decomposition method that follows the "rate certifying set approach" (a generalization of the "rate certifying pair approach") that was discussed and analysed in [26]. We will come back to this issue at the end of Section 6.

We will show in Section 4 that the decomposition method induced by the family $C_I(x)$ from (4) leads to an NP-hard (variable size) working set selection problem already for the (comparatively simple) case of SVM-optimization. On the other hand, we will design a polynomially time-bounded algorithm that, given the current feasible solution x, always finds a working set $I \subseteq \{1, \dots, m\}$ of size at most q that maximizes $C_I(x)$ up to factor 4. This raises the question of whether convergence can still be granted when the selection of a working set that maximizes $C_I(x)$ (referred to as "Exact Working Set Selection") is replaced by the selection of a working set that maximizes $C_I(x)$ only up to a constant factor (referred to as "Approximate Working Set Selection"). It turns out, fortunately, that this is true:

Theorem 3. *Let $\mathcal{P}$ be the optimization problem given by* (2) *and* (1)*, and let $q \ge k + 1$. Assume that $Q \in \mathbb{R}^{m \times m}$ is a matrix with positive definite sub-matrices $Q_{I,I}$ for all subsets $I \subseteq \{1, \dots, m\}$ of size at most q. Let $C_I(x)$ be a family of functions that satisfies conditions* (C1), (C2), (C3). *Let an* approximate decomposition method induced by *$C_I(x)$ be an algorithm that, given the current feasible solution x, always chooses a working set $I \subseteq \{1, \dots, m\}$ of size at most q that maximizes $C_I(x)$ up to a constant factor $c \ge 1$. Then this method computes a sequence of feasible solutions whose costs approach the cost of an optimal solution in the limit.*

Proof. The proof is similar to the proof of the main theorem in [25] (although we have to cope here with the weaker assumption of "Approximate Working Set Selection").

The first part of the proof applies to every policy for working set selection. Let I^n denote the working set chosen in iteration n of the decomposition method, and x^n the optimal feasible solution for the corresponding subproblem.

[6] This assumption does not follow automatically from the positive semi-definiteness of Q (though it would follow from positive definiteness of Q). It is however (at least for SVM-applications) often satisfied. For some kernels like, for example, the RBF-kernel, it is certainly true; for other kernels it is typically satisfied provided that q is sufficiently small. See also the discussion of this point in [20].

[7] Not necessarily identical to the family defined in (4).

Since f is bounded on $R(\mathcal{P})$ and $f(x^n)$ is decreasing with n, it converges to a limit even if x^n does not converge to a limit point. However, since $R(\mathcal{P})$ is compact, there exists an infinite set S of non-negative integers such that $(x^s)_{s\in S}$ converges to a feasible limit point x^∞. Clearly, $f(x^\infty) = \lim_{n\to\infty} f(x^n)$. In [25], it is shown that, for every $\delta > 0$, there exists $s_0 \geq 1$ such that

$$\|x^\infty - x^s\| < \delta \text{ and } \|x^\infty - x^{s+1}\| < \delta$$

holds for all $s \in S$ provided that $s \geq s_0$.[8] Thanks to condition (C1), we can proceed with a continuity argument and conclude that, for every $\varepsilon > 0$, there exists $s_0 \geq 1$ such that

$$|C_I(x^\infty) - C_I(x^s)| < \varepsilon \text{ and } |C_I(x^\infty) - C_I(x^{s+1})| < \varepsilon \tag{5}$$

holds for every working set $I \subseteq \{1, \dots, m\}$ of size at most q and for all $s \in S$, provided that $s \geq s_0$.
The final part of the proof is tailored to the particular policy of the approximate decomposition method induced by $C_I(x)$. Assume for sake of contradiction that x^∞ is not an optimal solution for $\mathcal{P}$. According to (C3), there exists a working set $I \subseteq \{1, \dots, m\}$ such that $|I| \leq q$ and

$$\epsilon_0 := C_I(x^\infty) > 0.$$

In the sequel, we will apply (5) several times with $\varepsilon := \epsilon_0/(3c) \leq \epsilon_0/3$, respectively. Assume that $s \in S$ is sufficiently large such that, according to (5),

$$C_I(x^s) > \frac{2\epsilon_0}{3}.$$

Thus, the working set I^{s+1} returned by the decomposition method in iteration $s+1$ satisfies

$$C_{I^{s+1}}(x^s) > \frac{2\epsilon_0}{3c}.$$

Another application of (5) leads to

$$C_{I^{s+1}}(x^\infty) > \frac{\epsilon_0}{3c}.$$

Again from (5), we get

$$C_{I^{s+1}}(x^{s+1}) > 0.$$

Since x^{s+1} is an optimal solution for the subproblem induced by I^{s+1}, we may however infer from (C2) that

$$C_{I^{s+1}}(x^{s+1}) = 0.$$

We arrived at a contradiction. $\square$

Theorem 3 provides a justification to pass from the "Exact Working Set Selection" to the "Approximate Working Set Selection".

4. Complexity of the exact working set selection

The function family $C_I(x)$ from (4) can be expressed in a simpler fashion when we consider the special optimization problem $\mathcal{P}_0$ from (3) in its "normalized version" (with $y = \bar{1}$):

$$C_I(x) = \inf_{t\in\mathbb{R}} \left(\sum_{i\in I} (x_i - l_i)\max\{0, (\nabla f(x))_i - t\} + (r_i - x_i)\max\{0, t - (\nabla f(x))_i\} \right).$$

[8] The proof in [25] is easily seen to be valid for *every* policy for working set selection.

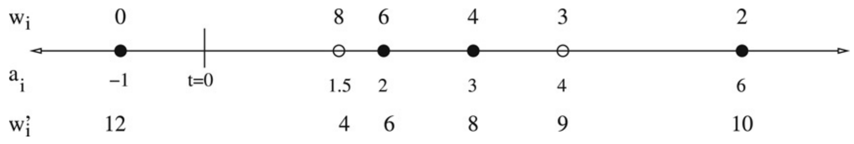

Fig. 1. Illustration of the game induced by the parameters in (6).

We will cast the problem of working set selection in the following (more abstract) purely combinatorial form[9]:
Exact Working Set Selection (EWSS): Given numbers $a_1 \le \cdots \le a_m$, non-negative weight parameters $w_1, w'_1, \ldots, w_m, w'_m$ and a non-negative number q, find a "working set" $I \subseteq \{1, \ldots, m\}$ of size at most q that maximizes the "gain"

$$G_I := \inf_{t\in\mathbb{R}} \left(\overbrace{\sum_{i\in I: a_i<t} w'_i(t-a_i) + \sum_{i\in I:a_i>t} w_i(a_i-t)}^{=:G_I(t)} \right).$$

When we consider EWSS as a decision problem, we assume that the input contains an additional bound B. The question is whether there exists a working set that achieves a gain of at least B.
Conventions:

(1) We set $a_0 := -\infty$ and $a_{m+1} := \infty$ such that $\mathbb{R}$ partitions into intervals of the form $[a_i, a_{i+1}]$ for $i = 0, \ldots, m$.
(2) We assume that the following conditions are satisfied[10]:
 (a) At least one of the parameters $w_1, \ldots, w_m$ is strictly positive.
 (b) At least one of the parameters $w'_1, \ldots, w'_m$ is strictly positive.
 (c) For every $i \in \{1, \ldots, m\}$, at least one of the parameters w_i, w'_i is strictly positive.

We may think of $a_1, \ldots, a_m$ as points on the real line that are ordered from left to right. EWSS corresponds to a game between two players:

(1) Player 1 selects $q' \le q$ points. Intuitively, this means that she commits herself to a working set $I \subseteq \{1, \ldots, m\}$ of size q'.
(2) Player 2 selects a threshold t and pays an amount of $\sum_{i\in I:a_i<t} w'_i(t-a_i) + \sum_{i\in I:a_i>t} w_i(a_i - t)$ to player 1.

Example 4. Consider the following sequence a_i of numbers and the corresponding weight parameters w_i, w'_i:

a_i	−1	1.5	2	3	4	6
w_i	0	8	6	4	3	2
w'_i	12	4	6	8	9	10

(6)

Let $q = 4$. The game between two players is visualized in Fig. 1, where the numbers a_i appear as points on the real line being marked by w_i (top marking) and w'_i (bottom marking). As indicated in the figure by the bold points, player 1 selects the $q' = q = 4$ numbers $-1, 2, 3, 6$ and player 2 chooses threshold $t = 0$. The gain achieved by player 1 evaluates to

$$(0-(-1)) \cdot 12 + (2-0) \cdot 6 + (3-0) \cdot 4 + (6-0) \cdot 2 = 48.$$

After this illustrative example, we continue with the general discussion. Given I (the move of player 1), the optimal move of player 2 is easy to determine because the following holds:

(1) $G_I(t)$ is a continuous function.

[9] The correspondence is as follows. Assume that $(\nabla f(x))_1 \le \cdots \le (\nabla f(x))_m$ (after re-indexing if necessary). Now, a_i represents $(\nabla f(x))_i$, w_i represents $x_i - l_i$, w'_i represents $r_i - x_i$.
[10] Instances that violate one of the following conditions are either trivial or can be simplified.

(2) $G_I(t)$ is a piecewise linear function that is linear in the interval from a_j to a_{j+1} for every $j \in \{0, \dots, m\}$. Moreover,

$$\forall a_j < t < a_{j+1} : G'_I(t) = \sum_{i \in I : i \le j} w'_j - \sum_{i \in I : i \ge j+1} w_i .$$

In particular:
(a) $G'_I(t) = -\sum_{i \in I} w_i < 0$ for $a_0 < t < a_1$.
(b) $G'_I(t) = +\sum_{i \in I} w'_i > 0$ for $a_m < t < a_{m+1}$.
(c) $G'_I(t) < G'_I(t) + w_j + w'_j = G_I(t')$ for $a_{j-1} < t < a_j < t' < a_{j+1}$.

From these observations, we immediately obtain the following result:

Lemma 5. *Let $j' \in \{1, \dots, m\}$ be minimal such that $G'_I(t) \ge 0$ on interval $(a_{j'}, a_{j'+1})$. Then the following holds:*

(1) *If $G'_I(t) > 0$ on interval $(a_{j'}, a_{j'+1})$, then $t = a_{j'}$ is the unique optimal move for player 2.*
(2) *If $G'_I(t) = 0$ on interval $(a_{j'}, a_{j'+1})$, then the set of optimal moves for player 2 equals the closed interval $[a_{j'}, a_{j'+1}]$.*

Moreover, t is an optimal move of player 2 iff it satisfies

$$-\sum_{i \in I : a_i = t} w'_i \le \sum_{i \in I : a_i < t} w'_i - \sum_{i \in I : a_i > t} w_i \le \sum_{i \in I : a_i = t} w_i . \quad (7)$$

For ease of later reference, we say that threshold t is in a *balanced position* if it satisfies (7). If $t \notin \{a_1, \dots, a_m\}$, then (7) collapses to $\sum_{i \in I : a_i < t} w'_i - \sum_{i \in I : a_i > t} w_i = 0$. It is easy to check that, in the setting illustrated in Fig. 1, threshold $t = 0$ is in a balanced position.

Clearly, Lemma 5 implies the following

Corollary 6. *Given I, a threshold $t \in \{a_1, \dots, a_m\}$ in a balanced position is found in linear time.*

We now turn to the problem of finding a best move for player 1. Let $S > 0$. We say that t is an *S-unifier* if, for $i = 1, \dots, m$, the following holds:

$$a_i < t \Rightarrow S = w'_i(t - a_i) \quad \text{and} \quad a_i > t \Rightarrow S = w_i(a_i - t).$$

It is easy to see that, in the setting illustrated in Fig. 1, threshold $t = 0$ is a 12-unifier.

Lemma 7. *If there exists an S-unifier, say t_0, then*

$$\max_{I \subseteq \{1,\dots,m\} : |I| \le q} \inf_{t \in \mathbb{R}} \left(\sum_{i \in I : a_i < t} w'_i(t - a_i) + \sum_{i \in I : a_i > t} w_i(a_i - t) \right) \le qS$$

with equality iff there exists a working set I of size q that puts t_0 in a balanced position.

Proof. If player 2 chooses threshold t_0, then the gain for player 1 is $|I|S \le qS$ (no matter which working set I was chosen). Thus, qS is the largest gain player 1 can hope for. If she finds a working set I of size q that puts t_0 in a balanced position, the gain qS is achieved (since player 2 will find no threshold superior to t_0). If not, player 2 can reduce the gain from qS to a strictly smaller value by shifting threshold t away from t_0 in the direction of the "heavier" side. □

Recall that "Subset Sum" is the following NP-complete problem:
Given non-negative integers $s_1 \le \dots \le s_m \le S$, decide whether there exists an $I \subseteq \{1, \dots, m\}$ such that $\sum_{i \in I} s_i = S$.

We consider here a variant of "Subset Sum" where the input contains an additional parameter $q \le m$, and the question is whether there exists an $I \subseteq \{1, \dots, m\}$ such that $|I| = q$ and $\sum_{i \in I} s_i = S$. We refer to this problem as "Exact Cardinality Constrained Subset Sum".

Lemma 8. *"Exact Cardinality Constrained Subset Sum" is NP-complete.*

Proof. The problem clearly belongs to NP. NP-hardness with respect to Turing-reductions is obvious because of the straightforward Turing-reduction from "Subset Sum" to "Exact Cardinality Constrained Subset Sum". □

We briefly note that the problem is also NP-hard with respect to the usual many-one reductions, which can be proven by slightly modifying Karp's [12] polynomial reduction from "3-dimensional Matching" to "Partition". Since the required modifications are quite straightforward, we omit the details.

Theorem 9. *"Exact Working Set Selection" (viewed as a decision problem) is NP-complete.*

Proof. Since the optimal threshold t for a given working set I can be found efficiently (and the appropriate working set can be guessed), the problem clearly belongs to NP. We complete the proof by presenting a polynomial reduction from "Exact Cardinality Constrained Subset Sum" to EWSS. From the input parameters $s_1 \le \cdots \le s_m \le S$ and q of "Exact Cardinality Constrained Subset Sum", we derive the following input parameters of EWSS:

- $a_0 = -1$ and $a_i = S/s_i$ for $i = 1, \ldots, m$.
- $(w_0, w'_0) = (0, S)$ and $(w_i, w'_i) = (s_i, S - s_i)$ for $i = 1, \ldots, m$.
- Parameter $q' = q + 1$ is considered as the bound on the size of the working set and $B = q'S$.

Fig. 1 illustrates this construction. It shows the input instance of EWSS that results from the input instance

$$(s_5, \ldots, s_1) = (2, 3, 4, 6, 8)\ ,\ S = 12\ ,\ q = 3$$

of "Exact Cardinality Constrained Subset Sum".
Note that threshold $t = 0$ is an S-unifier. Thus, the largest possible gain for player 1 is $B = q'S$. This gain can be achieved iff there exists a working set I of size q' that puts threshold 0 in a balanced position. Since $0 \notin \{a_0, a_1, \ldots, a_m\}$, 0 is in a balanced position iff

$$\sum_{i \in I : a_i < 0} w'_i = \sum_{i \in I : a_i > 0} w_i . \tag{8}$$

According to the choice of the input parameters for EWSS, condition (8) implies that $\sum_{i \in I : a_i < 0} w'_i = w'_0 = S$ and $\sum_{i \in I : a_i > 0} w_i = \sum_{i \in I : a_i > 0} s_i$, where the latter sum contains q terms.
This discussion can be summarized as follows: player 1 can achieve a gain of at least $B = q'S$ iff there exist q terms from $s_1, \ldots, s_n$ that sum-up to S. This shows that "Exact Cardinality Constrained Subset Sum" can be polynomially reduced to EWSS. □

5. Complexity of approximate working set selection

Recall that "Knapsack" is the following NP-complete problem:
Given a finite set M of items, a "weight" $w(i) > 0$ and a "value" $v(i) > 0$ for each $i \in M$, a "weight bound" $W > 0$ and a "value goal" $V > 0$, decide whether there exists a subset $\mathcal{I} \subseteq M$ such that

$$W_{\mathcal{I}} := \sum_{i \in \mathcal{I}} w(i) \le W \quad \textit{and} \quad V_{\mathcal{I}} := \sum_{i \in \mathcal{I}} v(i) \ge V.$$

In order to view "Knapsack" as optimization problem, we may drop the value constraint $V_{\mathcal{I}} \ge V$, and ask for a subset $\mathcal{I}$ that leads to the largest possible total value $V_{\mathcal{I}}$ subject to the weight constraint $W_{\mathcal{I}} \le W$.

We consider here the problem "Cardinality Constrained Knapsack". It is a variant of "Knapsack" where the input contains an additional parameter $q \le n$ and the constraint $|\mathcal{I}| \le q$ is put on top of the weight constraint. Since this problem contains "Knapsack" as a special case (where $q = |M|$), it is clearly NP-hard. The following is known about this problem:

Lemma 10 (*[2]*). (1) *There exists an algorithm for "Cardinality Constrained Knapsack" that runs in time $O(|M|)$ and outputs a solution whose total value is within a factor 2 of optimality.*

(2) *"Cardinality Constrained Knapsack" can be solved by a full polynomial time approximation scheme within $O(|M| + kq^2)$ steps, i.e., there exists an algorithm that, on input (M, w, v, W, q, k), runs for $O(|M| + kq^2)$ steps and outputs a (feasible) solution $\mathcal{I}$ whose total value $V_{\mathcal{I}}$ is within factor $1 + 1/k$ of optimality.*

The main result of this section is as follows:

Theorem 11. *Any algorithm that solves "Cardinality Constrained Knapsack" within $T(|M|, q)$ steps and outputs a solution whose total value is within factor c of optimality can be converted into an algorithm that solves "Approximate Working Set Selection" within $O(qmT(m, q))$ steps and outputs a solution that is within factor $2c$ of optimality.*

Proof. Let I_* denote a working set that leads to the largest possible gain, say g_*, for player 1 and t_* the corresponding threshold (being in a balanced position w.r.t. I_*). According to Lemma 5, we may assume without loss of generality that $t_* \in \{a_1, \ldots, a_m\}$. Define

$$\begin{aligned} M &= \{1, \ldots, m\}, & M_*^= &= \{i \in M \mid a_i = t_*\}, \\ M'_* &= \{i \in M \mid a_i < t_*\}, & M''_* &= \{i \in M \mid a_i > t_*\}, \end{aligned}$$

and

$$I_*^= = M_*^= \cap I_*,\ I'_* = M'_* \cap I_*,\ I''_* = M''_* \cap I_*.$$

Note that $g_* = g'_* + g''_*$ where

$$g'_* = \sum_{i \in I'_*} w'_i (t_* - a_i) \text{ and } g''_* = \sum_{i \in I''_*} w_i (a_i - t_*).$$

Consider the following non-deterministic[11] algorithm:

(1) Guess $j \in M$ such that $t_* = a_j$, compute the partition of M into $M_*^=$, M'_*, M''_* (assuming that a_j can play the role of t_*), guess whether $g'_* \geq g''_*$, and guess $0 \leq q' \leq q$.[12]
(2) If $g''_* \geq g'_*$, then proceed as follows:
  (a) Pick a set $I' \subseteq M'_* \cup M_*^=$ of size q' such that the parameters w'_i with $i \in I'$ are the q' largest weight parameters in $(w'_i)_{i \in M'_* \cup M_*^=}$. Compute the total weight $W' = \sum_{i \in I'} w'_i$.
  (b) For every $i \in M''_*$, set $v_i = w_i(a_i - t_*)$. Intuitively, v_i represents the gain that results from putting i into the working set. Apply the given algorithm, say A, for "Cardinality Constrained Knapsack" to the following problem instance:
    - M''_* is the set of items.
    - w_i is the weight and v_i the value of item $i \in M''_*$.
    - W' is the weight bound, and $q'' = q - q'$ is the bound on the number of items.

    Let I'' be the set of indices that is returned by A.
(3) If $g''_* < g'_*$, then apply the analogous procedure.
(4) Output working set $I = I' \cup I''$.

Let's see how to remove the initial non-deterministic guesses. In a deterministic implementation, the algorithm performs an exhaustive search instead of guessing: it loops through all $j \in M$ and $q' \in \{0, \ldots, q\}$, and it explores both possible values (TRUE and FALSE) of the Boolean expression $g''_* \geq g'_*$. Clearly, all but one runs through these loops are based on "wrong guesses" (including the possibly wrong computation of $M_*^=$, M'_*, M''_* when the algorithm uses an $a_j \neq t_*$ in the role of t_*). A wrong guess may even lead to an inconsistency. (For example, there might be less than q' elements in $M'_* \cup M_*^=$.) Whenever such an inconsistency is detected, the algorithm aborts the current iteration and proceeds with the next. Each non-aborted iteration (including the iteration with the correct guesses) leads to a candidate working set. A candidate set leading to the highest gain for player 1 is finally chosen.
Observe now that I''_* is a feasible solution for the knapsack Problem, since it is a subset of M'' of size at most $q'' = q - q'$ and

$$\sum_{i \in I''_*} w_i \leq \sum_{i \in I'_* \cup I_*^=} w'_i \leq W'.$$

[11] We will later remove non-determinism by means of an exhaustive search.
[12] If $g''_* \geq g'_*$, then we should intuitively identify q' with $|I_*^= + I'_*|$.

The first inequality holds because t_* is in a balanced position; the second-one holds because W' is the total weight of the q' "heaviest" points from $M'_* \cup M^=_*$. Let's now argue that the solution $I = I' \cup I''$ obtained in the iteration with the correct guesses leads to a gain $g \geq g_*/(2c)$. Let g' and g'' denote the gains represented by I' and I'', respectively (such that $g = g' + g''$). For reasons of symmetry, we may assume that $g''_* \geq g'_*$. Then, the following holds:

$$g \geq g'' \geq \frac{1}{c} \max \left\{ \sum_{i \in \mathcal{I}} v_i \;\middle|\; \mathcal{I} \subseteq M'', |\mathcal{I}| \leq q'', \sum_{i \in \mathcal{I}} w_i \leq W' \right\}$$
$$\geq \frac{1}{c} \sum_{i \in I''_*} v_i = \frac{g''_*}{c} \geq \frac{g_*}{2c}.$$

Note that, within the max-expression, $\mathcal{I}$ runs through all feasible solutions for the knapsack problem instance. The second inequality holds for the following reasons:

- By assumption, A has performance ratio c.
- Although $I = I' \cup I''$ does not necessarily put $t = a_j$ in a balanced position, only the left side might perhaps be "too heavy".[13] Thus $v_i = w_i(a_i - t_*)$ underestimates the true gain values contributing to g'' (or, at least, does not overestimate them).

The third inequality holds because I''_* is a feasible solution for the knapsack problem instance. The final inequality is valid since $g''_* \geq g'_*$.
The time bound $O(qmT(m, q))$ is obvious, since there are basically $O(qm)$ calls of the given algorithm A with time bound T. $\square$

Combining the first statement in Lemma 10 with Theorem 11, we get

Corollary 12. *There exists an algorithm for "Approximate Working Set Selection" that runs in time $O(qm^2)$ and finds a working set whose gain is within factor 4 of optimality.*

Although a time bound $O(qm^2)$ is a polynomial of small degree, it is still not satisfactory for practical purposes. It would be interesting to know whether there are faster approximation algorithms that avoid this detour through the knapsack problem.

For a large class of maximization problems (dealing with so-called "independence systems"), Korte and Schrader [16] characterized all problems that admit a full polynomial time approximation scheme (FPTAS). It turns out that, for any such problem with an FPTAS, one can apply a variant of the algorithm by Ibarra and Kim [10] that was originally designed for "Knapsack". The problem "Approximate Working Set Selection" does however *not* fit into the framework of independence systems. The following result is therefore *not* implied by the results of Korte and Schrader, and the FPTAS that we design for "Approximate Working Set Selection" is *not* a variant of Ibarra and Kim's algorithm (although it is also based on "rounding and scaling"):

Theorem 13. (1) *There exists a pseudo-polynomial algorithm for "Working Set Selection" that finds an optimal solution in $O(q^3 m^2 w_{\max}^2)$ steps where $w_{\max} := \max\{w_1, w'_1 \ldots, w_m, w'_m\}$.*
(2) *There exists a full polynomial time approximation scheme for "Approximate Working Set Selection" which runs for $O(k^2 q^7 m^3)$ steps and outputs a solution that is optimal up to factor $1 - 1/k$.*

The proof of Theorem 13 is technically involved, and therefore it is postponed to the Appendix A.

6. Linear time construction of a rate certifying pair

Throughout this section, we restrict ourselves to the optimization problem $\mathcal{P}_0$ from (3).

[13] This follows from the greedy selection of the q' indices that are included in I'.

Let $x \in R(\mathcal{P}_0)$ be a feasible solution and $x_* \in R(\mathcal{P}_0)$ an optimal feasible solution. Define

$$\sigma(x) := \sup_{x' \in R(\mathcal{P}_0)} \left((\nabla f(x))^\top (x - x') \right) \tag{9}$$

$$\sigma(x|i_1, i_2) := \sup_{x' \in R(\mathcal{P}_0): x'_i = x_i \text{ for } i \neq i_1, i_2} \left((\nabla f(x))^\top (x - x') \right). \tag{10}$$

As already noted by Hush and Scovel [9], the following holds[14]:

$$f(x) - f(x_*) \leq (\nabla f(x))^\top (x - x_*) \leq \sigma(x). \tag{11}$$

(i_1, i_2) is called an *α-rate certifying pair* for x if

$$\sigma(x|i_1, i_2) \geq \alpha(f(x) - f(x_*)). \tag{12}$$

An *α-rate certifying algorithm* is a decomposition algorithm for $\mathcal{P}_0$ that always includes an α-rate certifying pair for the current feasible solution in the working set. Hush and Scovel [9] have shown that any α-rate certifying algorithm has a polynomial rate of convergence. Furthermore, they have provided a decomposition algorithm that finds an $(1/m^2)$-rate certifying pair for the current feasible solution within $O(m \log m)$ steps.

In this section, we describe how to retrieve an α-rate certifying pair for the current feasible solution in linear time, where α is always as large as possible (at least $1/m^2$).

To this end, we introduce the following notions:

- A pair (i_1, i_2) that maximizes $\sigma(x|i_1, i_2)$ is called a *maximum-rate certifying pair* for x.
- Let DS(x) denote the data structure (associated with the current feasible solution x) that consists of the following two sorted lists of length m, respectively:
  . The first list contains the items $(i, x_i - l_i)$ for $i = 1, \ldots, m$.
  . The second list contains the items $(i, u_i - x_i)$ for $i = 1, \ldots, m$.

  Both lists are sorted non-increasingly according to the *second* component of their items.

Note that data structure DS(x) can be updated in $O(m)$ steps, because two subsequent feasible solutions differ in only constantly many components (assuming constant size of the working set). The main result of this section reads as follows:

Theorem 14. *Given x, $\nabla f(x)$, and DS(x), a maximum-rate certifying pair can be computed in $O(m)$ steps.*

Proof. There are only m possible values for the quantities

$$\mu_i^+ := x_i - l_i \quad \text{and} \quad \mu_i^- := r_i - x_i,$$

respectively (because i ranges from 1 to m). The key to the proof is the observation that the inspection of m^2 candidate pairs (i_1, i_2) can be circumvented by considering all possible

$$\mu \in M := \{x_i - l_i, r_i - x_i \,|\, i = 1, \ldots, m\},$$

one at a time, and by efficiently retrieving a kind of "conditioned maximum-rate certifying pair" (i_1, i_2) subject to $\mu_{i_1}^+ \geq \mu$ and $\mu_{i_2}^- \geq \mu$. Details follow.
Without loss of generality, we may assume that $\mathcal{P}_0$ is normalized, i.e., y in (3) is the all-ones vector. It is not hard to see that the following holds:

Claim 1. If $(\nabla f(x))_{i_1} \geq (\nabla f(x))_{i_2}$ then

$$\sigma(x|i_1, i_2) = \left((\nabla f(x))_{i_1} - (\nabla f(x))_{i_2}\right) \min\{\mu_{i_1}^+, \mu_{i_2}^-\}.$$

[14] The first inequality in (11) is equivalent to $f(x) + (\nabla f(x))^\top (x_* - x) \leq f(x_*)$, which is satisfied by all convex functions. The second inequality in (11) is trivial.

Proof of Claim 1. According to (10), we may choose x' such that $\bar{1}^\top x' = 0$, $l \le x' \le r$, $x'_i = x_i$ for every $i \ne i_1, i_2$, and

$$\sigma(x|i_1, i_2) = (\nabla f(x))_{i_1}(x_{i_1} - x'_{i_1}) + (\nabla f(x))_{i_2}(x_{i_2} - x'_{i_2}).$$

Recall that x (as a member of $R(\mathcal{P}_0)$) also satisfies $\bar{1}^\top x = 0$ and $l \le x \le r$. Thus,

$$0 = \bar{1}^\top (x - x') = (x_{i_1} - x'_{i_1}) + (x_{i_2} - x'_{i_2}),$$

which implies that

$$d := x_{i_1} - x'_{i_1} = x'_{i_2} - x_{i_2} \tag{13}$$

and

$$\sigma(x|i_1, i_2) = d \cdot \left((\nabla f(x))_{i_1} - (\nabla f(x))_{i_2}\right).$$

Since, by assumption, $(\nabla f(x))_{i_1} - (\nabla f(x))_{i_2} \ge 0$, it follows that x' was chosen so as to maximize d from (13) without violating the box-constraints. The largest possible value for d is obviously

$$d = \min\{x_{i_1} - l_{i_1}, r_{i_2} - x_{i_2}\} = \min\{\mu^+_{i_1}, \mu^-_{i_2}\},$$

which concludes the proof of Claim 1.

Claim 2. For $\mu \in M = \{x_i - l_i, r_i - x_i \mid i = 1, \ldots, m\}$, define

$$\sigma_\mu(x) = \left(\max_{i_1:\mu^+_{i_1} \ge \mu} (\nabla f(x))_{i_1} - \min_{i_2:\mu^-_{i_2} \ge \mu} (\nabla f(x))_{i_2}\right)\mu.$$

Then, the following holds:

$$\max_{i_1,i_2} \sigma(x|i_1, i_2) = \max_\mu \sigma_\mu(x).$$

Proof of Claim 2. We first show that

$$\max_{i_1,i_2} \sigma(x|i_1, i_2) \le \max_\mu \sigma_\mu(x).$$

According to Claim 1, we may choose i_1^*, i_2^* such that

$$0 \le \max_{i_1,i_2} \sigma(x|i_1, i_2) = \left((\nabla f(x))_{i_1^*} - (\nabla f(x))_{i_2^*}\right)\min\{\mu^+_{i_1^*}, \mu^-_{i_2^*}\}.$$

Setting

$$\mu^* := \min\{\mu^+_{i_1^*}, \mu^-_{i_2^*}\},$$

we may proceed as follows:

$$\begin{aligned}\max_{i_1,i_2} \sigma(x|i_1, i_2) &= \left((\nabla f(x))_{i_1^*} - (\nabla f(x))_{i_2^*}\right)\mu_* \\ &\le \max_\mu \left(\max_{i_1:\mu^+_{i_1} \ge \mu} (\nabla f(x))_{i_1} - \min_{i_2:\mu^-_{i_2} \ge \mu} (\nabla f(x))_{i_2}\right).\end{aligned}$$

The last inequality is valid because $\mu := \mu^*$ and $i_1 := i_1^*, i_2 := i_2^*$ is among the possible choices for μ, i_1, i_2. The reverse inequality

$$\max_\mu \sigma_\mu(x) \le \max_{i_1,i_2} \sigma(x|i_1, i_2)$$

can be shown is a similar fashion. Choose μ^* and i_1^*, i_2^* such that

$$\mu^+_{i_1^*} \geq \mu^* \text{ and } \mu^-_{i_2^*} \geq \mu^*$$

and

$$\begin{aligned}\max_{\mu} \sigma_\mu(x) &= \Big((\nabla f(x))_{i_1^*} - (\nabla f(x))_{i_2^*}\Big)\,\mu_* \\ &\leq \Big((\nabla f(x))_{i_1^*} - (\nabla f(x))_{i_2^*}\Big)\min\{\mu^+_{i_1^*}, \mu^-_{i_2^*}\} \\ &\leq \max_{i_1,i_2} \sigma(x|i_1, i_2).\end{aligned}$$

This concludes the proof of Claim 2.

In order to compute the pair (i_1, i_2) that maximizes $\sigma(x|i_1, i_2)$, we may apply Claim 2 and proceed as follows: Consider the values $\mu \in M$ in decreasing order, thereby keeping track of the index $i_1(\mu)$ that maximizes $(\nabla f(x))_{i_1}$ subject to $\mu^+_{i_1} \geq \mu$, and of the index $i_2(\mu)$ that minimizes $(\nabla f(x))_{i_2}$ subject to $\mu^-_{i_2} \geq \mu$. Finally, pick the value μ_* that maximizes $\big((\nabla f(x))_{i_1(\mu)} - (\nabla f(x))_{i_2(\mu)}\big)\,\mu$ and output $(i_1(\mu_*), i_2(\mu_*))$. Clearly, this procedure can be implemented within $O(m)$ steps by means of DS(x). □

In the final two remarks, we would like to point the reader's attention to some new results that are connected to the results in this paper (and that were found during the review period):

Remark 15. As for SVM-optimization, there always exists a $1/m$-rate certifying pair [26] or (the latest improvement) even a $1/(m-1)$-rate certifying pair [8]. In [8], the interested reader will also find further discussion of (and experiments with) our linear time procedure for the retrieval of a maximum-rate certifying pair (in combination with procedures that follow the “violating-pair approach”).

Remark 16. List and Simon [26] have generalized the “rate certifying pair approach” such that it becomes applicable to arbitrary convex quadratic optimization problems (not just SVM-optimization). To this end, they have defined a function $\sigma(x|I)$, where I denotes a working set of size at most q. This function plays the same role for arbitrarily large working sets that function $\sigma(x|i_1, i_2)$ is playing for working sets of size 2. Furthermore, List [23] has shown that the function $C_I(x)$ from Eq. (4) in this paper coincides with a function $\sigma(x|I)$ from [26]. Looking back, it follows that Theorems 14 and 9 (about SVM-optimization) nicely complement each-other:

- Finding a maximum-rate certifying pair can be done in linear time.
- Finding a maximum-rate certifying set (of variable size) is NP-hard.

7. Future research

A central object of our future research is the design of policies that lead to a fast convergence to the optimum and, at the same time, can be efficiently implemented for a wide range of optimization problems. Furthermore we would like to gain a better theoretical understanding of the good empirical performance of SVM-optimization algorithms that follow the “violating pair approach”.

Acknowledgements

The very useful remarks and suggestions of two anonymous referees are gratefully acknowledged.

Appendix A. Proof of Theorem 13

The first statement of the theorem is proven in Section A.1, and the second one in Section A.2.

A.1. A pseudo-polynomial algorithm for "Working Set Selection"

For technical reasons, we consider a slightly more general problem where an input instance is augmented by "gain values" $v_1, \dots, v_m, v'_1, \dots, v'_m$ (which may be functions of a threshold $t \in \{a_1, \dots, a_m\}$ chosen by player 2).[15] It will still be the case that player 1 obtains gain v_i if $a_i > t$, gain v'_i if $a_i < t$ and no gain if $a_i = t$.[16] For any $1 \le j \le m$, consider a 4-dimensional table

$$G_j[m', q', l, r] = \max_I \left(\sum_{i \in I : a_i < a_j} v'_i + \sum_{i \in I : a_i > a_j} v_i \right) \tag{A.1}$$

subject to $I \subseteq \{1, \dots, m'\}$, $|I| \le q'$, and

$$l - \sum_{i \in I : a_i = a_j} w'_i \le \sum_{i \in I : a_i < a_j} w'_i - \sum_{i \in I : a_i > a_j} w_i \le r + \sum_{i \in I : a_i = a_j} w_i .$$

A set I leading to the maximum in (A.1) and satisfying the subsequent constraints is called a "witness" for "gain" $G_j[m', q', l, r]$. From the tables G_j, we can infer the largest possible gain for player 1:

- $G_j[m, q, 0, 0]$ is the largest gain for player 1 subject to the condition that she selects a set $I \subseteq \{1, \dots, m\}$ of size at most q that puts threshold $t = a_j$ in a balanced position (such that player 2 is forced to select threshold $t = a_j$).
- $\max_{1 \le j \le m} G_j[m, q, 0, 0]$ represents the largest possible gain for player 1.

Assuming

$$1 \le m' \le m, \ 1 \le q' \le q, \ -q w_{\max} \le l, r \le q w_{\max},$$

table G_j is of size $q^3 m w_{\max}^2$. We adopt the convention that G_j evaluates to $-\infty$ when one of the parameters m', q', l, r is outside its range. For $m' = 1$, the entries of G_j are computed according to

$$G_j[1, q', l, r] = \begin{cases} v'_1 \text{ if } a_1 < a_j \text{ and } l \le +w'_1 \le r \\ v_1 \text{ if } a_1 > a_j \text{ and } l \le -w_1 \le r \\ 0 \text{ otherwise.} \end{cases}$$

It is easy to see that, for $m' \ge 2$, the following holds:

- If $a_{m'} < a_j$, then $G_j[m', q', l, r]$ equals
$$\max\{G_j[m'-1, q', l, r] \ , \ v'_{m'} + G_j[m'-1, q'-1, l - w'_{m'}, r - w'_{m'}]\}.$$
- If $a_{m'} > a_j$, then $G_j[m', q', l, r]$ equals
$$\max\{G_j[m'-1, q', l, r] \ , \ v_{m'} + G_j[m'-1, q'-1, l + w_{m'}, r + w_{m'}]\}.$$
- If $a_{m'} = a_j$, then $G_j[m', q', l, r]$ equals
$$\max\{G_j[m'-1, q', l, r] \ , \ G_j[m'-1, q'-1, l - w'_{m'}, r + w_{m'}]\}.$$

It is furthermore easy to see that, without introducing much additional overhead, we can maintain a data structure that allows us to efficiently retrieve (on demand) a witness I for an entry of a table. Thus, we can first compute the largest possible gain for player 1 and then the index set $I \subseteq \{1, \dots, m\}$ representing the best move of player 1. Since there are m tables of size $q^3 m w_{\max}^2$, we arrive at a pseudo-polynomial algorithm with time bound $O(q^3 m^2 w_{\max}^2)$. This completes the proof of the first statement in Theorem 13. $\square$

[15] It is precisely this functional dependence that makes our problem not fit into the framework of independence systems.
[16] Thus, the special setting $v_i = w_i(a_i - t)$ and $v'_i = w'_i(t - a_i)$ leads us back to our original problem.

A.2. An FPTAS for “Approximate Working Set Selection”

“Rounding and scaling” is a standard technique that often allows one to convert a pseudo-polynomial algorithm into a full polynomial time approximation scheme (FPTAS). As we will show below, this is also the case for “Approximate Working Set Selection”. The basic idea is to apply the pseudo-polynomial algorithm from the previous subsection with gain values $v_i = w_i(a_i - t)$, $v'_i = w'_i(t - a_i)$ and with weight parameters $\tilde{w}_i, \tilde{w}'_i$ obtained from w_i, w'_i by means of rounding and scaling. We will, however, land into a difficulty that does not exist for problems fitting into the framework of independence systems: since we are not dealing with the true weight parameters, we will not be able to check whether a selection of $I \subseteq \{1, \dots, m\}$ puts a threshold $t = a_j$ into a balanced position. Instead, we can guarantee only that the position is “almost balanced”. The total gain value that we associate with the selection of I may therefore be too optimistic. Player 2 can make use of the (slight) unbalance, shift his threshold away from a_j and destroy some fraction of the “quasi-gain” that would result from $t = a_j$. We will, however, be able to show that a fraction $1 - 1/k - 1/k^2$ of the quasi-gain can be saved. To this end, the scaling factor used to bring the weight parameters into a smaller range must be carefully chosen. We *cannot* simply make it proportional to $w_{\max}$, because the resulting rounding error would give player 2 too much flexibility for choosing his threshold. In fact, we will be forced to exhaustively search through a collection of possible scaling factors which will result in a larger collection of tables. We will, furthermore, distinguish between non-degenerate and degenerate solutions, show how to approximate the best non-degenerate solution, and discuss the case of degenerate solutions separately. Details follow.

We start with the definition of degenerate and non-degenerate index sets. For any $I \subseteq \{1, \dots, m\}$ and any $1 \le j \le m$, define

$$w_+(I, j) = \max\{w_i, w'_i \mid i, i' \in I, a_i < a_j, a_{i'} > a_j\},$$
$$w_+^=(I, j) = \max\{w_i, w'_i \mid i, i' \in I, a_i = a_{i'} = a_j\}.$$

Let $I \subseteq \{1, \dots, m\}$, $|I| \le q$, be an index set that puts threshold $t = a_j$ be in a balanced position. We say that I is *non-degenerate* if $w_+^=(I, j) \le q w_+(I, j)$. Let $G_j^{\max}$ denote the maximal gain for player 1 provided that she selects a non-degenerate index set that puts threshold $t = a_j$ into a balanced position, and let $G^{\max} := \max_{1 \le j \le m} G_j^{\max}$. In the sequel, we show how to approximate $G^{\max}$ in polynomial time. Thereafter, we briefly sketch how to approximate the best degenerate solution.

For any $i_+, j \in \{1, \dots, m\}$ such that $a_{i_+} \ne a_j$ and for

$$w_+ := \begin{cases} w'_{i_+} & \text{if } a_{i_+} < a_j \\ w_{i_+} & \text{if } a_{i_+} > a_j, \end{cases} \tag{A.2}$$

consider the 4-dimensional table of “quasi-gains”

$$G_{i_+, j}[m', q', l, r] = \max_I \left(\sum_{i \in I : a_i < a_j} v'_i + \sum_{i \in I : a_i > a_j} v_i \right) \tag{A.3}$$

subject to $i_+ \notin I$, $I \subseteq \{1, \dots, m'\}$, $|I| \le q'$,

$$w_+(I, j) \le w_+ \,, \; w_+^=(I, j) \le q w_+ \tag{A.4}$$

and

$$l - \sum_{i \in I : a_i = a_j} \tilde{w}'_i \le \sum_{i \in I : a_i < a_j} \tilde{w}'_i - \sum_{i \in I : a_i > a_j} \tilde{w}_i \le r + \sum_{i \in I : a_i = a_j} \tilde{w}_i$$

where

$$\tilde{w}_i := \left\lfloor \frac{2qkw_i}{w_+} \right\rfloor \quad \text{and} \quad \tilde{w}'_i := \left\lfloor \frac{2qkw'_i}{w_+} \right\rfloor .$$

A set I leading to the maximum in (A.3) and satisfying the subsequent constraints is called a “witness” for “quasi-gain” $G_{i_+, j}[m', q', l, r]$. The role of the parameter i_+ will be clarified later.

In order to compute table $G_{i_+, j}$, we proceed as in the pseudo-polynomial algorithm, except that index i_+ and indices

leading to a violation of (A.4) are out of the game. Note that condition (A.4) makes sure that the parameters $\tilde{w}_i$ and $\tilde{w}'_{i'}$, $i, i' \in I, a_{i'} \le a_j, a_i \ge a_j$, do not exceed $2q^2k$. Substituting $2q^2k$ for $w_{\max}$ in the time bound of the pseudo-polynomial algorithm and taking into account that we have m-times as many tables as before, we conclude that all tables can be computed within $O(k^2q^7m^3)$ steps (even when an entry of our choice is provided with a witness).
We move on and show how $G^{\max}$ can be efficiently approximated by means of the tables $G_{i_+,j}$. To this end, we consider "special entries" of the form

$$\hat{G}_{i_+,j} := \begin{cases} G_{i_+,j}[m,\ q-1,\ \overbrace{-\tilde{w}_+ - q}^{=l},\ \overbrace{-\tilde{w}_+ + q}^{=r}] \text{ if } a_{i_+} < a_j \\ G_{i_+,j}[m,\ q-1,\ \underbrace{\tilde{w}_+ - q}_{=l},\ \underbrace{\tilde{w}_+ + q}_{=r}] \text{ if } a_{i_+} > a_j. \end{cases}$$

Given a special entry (quasi-gain) $\hat{G}_{i_+,j}$ and a corresponding witness I, we should intuitively think of $I_+ := I \cup \{i_+\}$ as the set chosen by player 1. This explains why we have chosen parameter $q-1$ instead of q, and why we find "set-offs" $\pm\tilde{w}_+$ in the specification of the parameters l and r, respectively. The additional set-off-terms $\pm q$ will account for rounding errors.
Let

$$\tilde{G}_{i_+,j} := \begin{cases} v'_{i_+} + \hat{G}_{i_+,j} \text{ if } a_{i_+} < a_j \\ v_{i_+} + \hat{G}_{i_+,j} \text{ if } a_{i_+} > a_j, \end{cases}$$

$\tilde{G}_j := \max_{i_+ \ne j} \tilde{G}_{i_+,j}$ and $\tilde{G}_* := \max_{1 \le j \le m} \tilde{G}_j$.

Claim 1. $\tilde{G}_j \ge G_j^{\max}$ and (therefore) $\tilde{G}_* \ge G_*^{\max}$.

Proof of the Claim. Let I_+ be a non-degenerate index-set that puts threshold $t = a_j$ in a balanced position and leads to gain $G_j^{\max}$ for player 1. Let $w_+ := w_+(I, j)$. Choose i_+ such that w_+, i_+ satisfy (A.2). In the sequel, we focus on the case $a_{i_+} < a_j$ (and note that the case $a_{i_+} > a_j$ can be treated analogously). Finally, let $I := I_+ \setminus \{i_+\}$. Since I_+ puts threshold $t = a_j$ in a balanced position, we know that

$$-\sum_{i \in I_+ : a_i = a_j} w'_i \le \sum_{i \in I_+ : a_i < a_j} w'_i - \sum_{i \in I_+ : a_i > a_j} w_i \le \sum_{i \in I_+ : a_i = a_j} w_i.$$

Since scaling and rounding are applied to at most q terms, we obtain

$$-q - \sum_{i \in I_+ : a_i = a_j} \tilde{w}'_i \le \sum_{i \in I_+ : a_i < a_j} \tilde{w}'_i - \sum_{i \in I_+ : a_i > a_j} \tilde{w}_i \le q + \sum_{i \in I_+ : a_i = a_j} \tilde{w}_i, \tag{A.5}$$

which, since $a_{i_+} < a_j$, can be rewritten as

$$-\tilde{w}_+ - q - \sum_{i \in I : a_i = a_j} \tilde{w}'_i \le \sum_{i \in I : a_i < a_j} \tilde{w}'_i - \sum_{i \in I : a_i > a_j} \tilde{w}_i \le -\tilde{w}_+ + q + \sum_{i \in I : a_i = a_j} \tilde{w}_i. \tag{A.6}$$

Index set I witnesses that

$$\hat{G}_{i_+,j} = G_{i_+,j}[m,\ q-1,\ -\tilde{w}_+ - q,\ -\tilde{w}_+ + q] \ge G_j^{\max} - v'_{i_+}.$$

Thus $\tilde{G}_j \ge \tilde{G}_{i_+,j} = v'_{i_+} + \hat{G}_{i_+,j} \ge G_j^{\max}$. □

Let $j \in \{1, \ldots, m\}$ be arbitrary but fixed. Choose i_+ such that $\tilde{G}_j = \tilde{G}_{i_+,j}$. Let I be a witness of the special entry $\hat{G}_{i_+,j}$. We can view $\tilde{G}_j = \tilde{G}_{i_+,j}$ as the gain for player 1 provided that she selects $I_+ := I \cup \{i_+\}$, and provided that player 2 chooses the (perhaps suboptimal) threshold $t = a_j$. The crucial question is how much this gain is diminished when player 2 chooses an optimal threshold. Here is an answer to this question:

Claim 2: If player 1 chooses I_+ and player 2 chooses an optimal threshold, the resulting gain for player 1 is not smaller than $(1 - 1/k - 1/k^2)\tilde{G}_j$.

Proof of the Claim: Again, we focus on the case $a_{i_+} < a_j$ (and note that the case $a_{i_+} > a_j$ can be treated analogously). Since I is a witness for table entry $\hat{G}_{i_+,j} = G_{i_+,j}[m,\ q-1,\ -\tilde{w}_+ - q,\ -\tilde{w}_+ + q]$, we know that (A.5) holds. Rescaling (dividing by $2qk/w_+$) and taking into account the rounding errors, we get (after some straightforward algebraic manipulations)

$$\sum_{i\in I_+:a_i=a_j} w'_i - \frac{w_+}{k} \le \sum_{i\in I_+:a_i<a_j} w'_i - \sum_{i\in I_+:a_i>a_j} w_i \le \sum_{i\in I_+:a_i=a_j} w_i + \frac{w_+}{k}. \tag{A.7}$$

If player 2 shifts the threshold d units away from a_j (such that it comes into a balanced position), he can diminish the gain at most by dw_+/k. On the other hand,

$$\min\left\{\sum_{i\in I_+:a_i=a_j} w'_i + \sum_{i\in I_+:a_i<a_j} w'_i\ ,\ \sum_{i\in I_+:a_i=a_j} w_i + \sum_{i\in I_+:a_i>a_j} w_i\right\} \ge \left(1-\frac{1}{k}\right) w_+,$$

because, term w_+ occurs in $\sum_{i\in I_+:a_i<a_j} w'_i$ and (A.7) implies that

$$\sum_{i\in I_+:a_i=a_j} w_i + \sum_{i\in I_+:a_i>a_j} w_i \ge \sum_{i\in I_+:a_i<a_j} w'_i - \frac{w_+}{k} \ge w_+ - \frac{w_+}{k}.$$

Thus, no matter which direction the threshold is shifted to, the remaining gain is at least $d(1-1/k)w_+$. It follows that player 2 can diminish the quasi-gain $\tilde{G}_j$ at most by fraction $(1-(1/k))/k = 1/k - 1/k^2$. Thus, the resulting gain for player 1 is at least $(1-1/k-1/k^2)\tilde{G}_j$. □

It is now easy to describe how $G^{\max}$ can be approximated. First, choose indices i_*, j_* as maximizers for $\tilde{G}_{i_+,j}$. Then choose a witness I for the special table entry $\hat{G}_{i_*,j_*}$. Finally let player 1 choose $I_* := I \cup \{i_*\}$. According to Claim 2, the resulting gain is not smaller than $(1-1/k-1/k^2)\tilde{G}_*$. According to Claim 1, this is not smaller than $(1-1/k-1/k^2)G_*^{\max}$.

Of course, the best non-degenerate solution could be much worse than the best degenerate solution. Therefore, we have to finally describe how to approximate the best gain for player 1, provided that she selects a degenerate index set I. This approximation can be performed in an even simpler manner for the following reasons:

- We can distinguish between three types of degeneracies:
  (1) $\max_{i\in I:a_i=a_j} w'_i > qw_+(I,j)$ but $\max_{i\in I:a_i=a_j} w_i \le qw_+(I,j)$.
  (2) $\max_{i\in I:a_i=a_j} w_i > qw_+(I,j)$ but $\max_{i\in I:a_i=a_j} w'_i \le qw_+(I,j)$.
  (3) $\max_{i\in I:a_i=a_j} w'_i > qw_+(I,j)$ and $\max_{i\in I:a_i=a_j} w_i > qw_+(I,j)$.
- Each type of degeneracy leads to a simplified dynamic programming scheme. Consider, for instance, the degeneracy of the first type. In this case, we will replace condition $w_\mp^=(I,j) \le qw_+$ in (A.4) by $\max_{i\in I:a_i=a_j} w'_i > qw_+$ and $\max_{i\in I:a_i=a_j} w_i \le qw_+$. Thus, I automatically satisfies the balance condition (7). Consequently, parameter l of the 4-dimensional table now becomes superfluous. Similar modifications (basically simplifications, since the table becomes lower-dimensional) take place for degeneracies of the second or third type.

We omit the straightforward but tedious details. □

References

[1] Bernhard E. Boser, Isabelle M. Guyon, Vladimir N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, 1992, pp. 144–152.

[2] Alberto Caprara, Hans Kellerer, Ulrich Pferschy, David Pisinger, Approximation algorithms for knapsack problems with cardinality constraints, European Journal of Operational Research 123 (2) (2000) 333–345.

[3] Chih-Chung Chang, Chih-Wei Hsu, Chih-Jen Lin, The analysis of decomposition methods for support vector machines, IEEE Transactions on Neural Networks 11 (4) (2000) 248–250.

[4] Chih-Chung Chang, Chih-Jen Lin, LIBSVM: A library for support vector machines, 2001. Available from: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Pai-Hsuen Chen, Rong-En Fan, Chih-Jen Lin, Training support vector machines via SMO-type decomposition methods, in: Proceedings of the 16th International Conference on Algorithmic Learning Theory, 2005, pp. 45–62.

[6] Pai-Hsuen Chen, Rong-En Fan, Chih-Jen Lin, A study on SMO-type decomposition methods for support vector machines, IEEE Transactions on Neural Networks 17 (4) (2006) 893–908.

[7] Chih-Wei Hsu, Chih-Jen Lin, A simple decomposition method for support vector machines, Machine Learning 46 (1–3) (2002) 291–314.

[8] Don Hush, Patrick Kelly, Clint Scovel, Ingo Steinwart, QP algorithms with guaranteed accuracy and run time for support vector machines, Journal of Machine Learning Research 7 (2007) 733–769.

[9] Don Hush, Clint Scovel, Polynomial-time decomposition algorithms for support vector machines, Machine Learning 51 (1) (2003) 51–71.

[10] Oscar H. Ibarra, Chul E. Kim, Fast approximation algorithms for knapsack and sum of subset problem, Journal of the Association on Computing Machinery 22 (4) (1975) 463–468.

[11] Thorsten Joachims, Making large scale SVM learning practical, in: Bernhard Schölkopf, Christopher J.C. Burges, Alexander J. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, 1998, pp. 169–184.

[12] Richard M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, James W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.

[13] S. Sathiya Keerthi, E.G. Gilbert, Convergence of a generalized SMO algorithm for SVM classifier design, Machine Learning 46 (1–3) (2002) 351–360.

[14] S. Sathiya Keerthi, ShirishKrishnaj Shevade, Chiranjib Bhattacharyya, Karaturi Radha Krishna Murthy, Improvements to SMO algorithm for SVM regression, IEEE Transactions on Neural Networks 11 (5) (2000) 1188–1193.

[15] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, Karaturi Radha Krishna Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Neural Computation 13 (3) (2001) 637–649.

[16] Bernhard Korte, Rainer Schrader, On the existence of fast approximation schemes, in: Olvi L. Mangasarian, R.R. Meyer, Stephen M. Robinson (Eds.), Nonlinear Programming, vol. 4, Academic Press, 1981, pp. 415–437.

[17] Pavel Laskov, Feasible direction decomposition algorithms for training support vector machines, Machine Learning 46 (1–3) (2002) 315–349.

[18] Shuo-Peng Liao, Hsuan-Tien Lin, Chih-Jen Lin, A note on the decomposition methods for support vector regression, Neural Computation 14 (6) (2002) 1267–1281.

[19] Chih-Jen Lin, Linear convergence of a decomposition method for support vector machines, 2001. Available from: http://www.csie.ntu.edu.tw/~cjlin/papers/linearconv.pdf.

[20] Chih-Jen Lin, On the convergence of the decomposition method for support vector machines, IEEE Transactions on Neural Networks 12 (6) (2001) 1288–1298.

[21] Chih-Jen Lin, Asymptotic convergence of an SMO algorithm without any assumptions, IEEE Transactions on Neural Networks 13 (1) (2002) 248–250.

[22] Chih-Jen Lin, A formal analysis of stopping criteria of decomposition methods for support vector machines, IEEE Transactions on Neural Networks 13 (5) (2002) 1045–1052.

[23] Nikolas List, Personal communication.

[24] Nikolas List, Convergence of a generalized gradient selection approach for the decomposition method, in: Proceedings of the 15th International Conference on Algorithmic Learning Theory, 2004, pp. 338–349.

[25] Nikolas List, Hans Ulrich Simon, A general convergence theorem for the decomposition method, in: Proceedings of the 17th Annual Conference on Computational Learning Theory, 2004, pp. 363–377.

[26] Nikolas List, Hans Ulrich Simon, General polynomial time decomposition algorithms, in: Proceedings of the 17th Annual Conference on Computational Learning Theory, 2005, pp. 308–322.

[27] Olvi L. Mangasarian, David R. Musicant, Successive overrelaxation for support vector machines, IEEE Transactions on Neural Networks 10 (5) (1999) 1032–1037.

[28] Olvi L. Mangasarian, David R. Musicant, Active support vector machine classification, in: Advances in Neural Information Processing Systems, vol. 12, MIT Press, 2000, pp. 577–583.

[29] Olvi L. Mangasarian, David R. Musicant, Lagrangian support vector machines, Journal of Machine Learning Research 1 (2001) 161–177.

[30] Edgar E. Osuna, Robert Freund, Federico Girosi, Training support vector machines: An application to face detection, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 130–136.

[31] John C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Bernhard Schölkopf, Christopher J.C. Burges, Alexander J. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, 1998, pp. 185–208.

[32] Craig Saunders, Mark O. Stitson, Jason Weston, Leon Bottou, Bernhard Schölkopf, Alexander J. Smola, Support vector machine reference manual, Technical Report CSD-TR-98-03, Royal Holloway, University of London, Egham, UK, 1998.

[33] Hans Ulrich Simon, On the complexity of working set selection, in: Proceedings of the 15th International Conference on Algorithmic Learning Theory, 2004, pp. 324–337.

[34] Vladimir Vapnik, Statistical learning theory, in: Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control, John Wiley & Sons, 1998.