*Optimization Methods and Software*
Vol. 00, No. 00, Month 200x, 1–23

# A low-cost interior-point algorithm for continuous knapsack problems with application to Support Vector Machines.

Maria D. Gonzalez-Lima[a]* and Esnil Guevara[b]

[a] *Universidad Militar Nueva Granada, Departamento de Matemáticas, Colombia and Universidad Simón Bolívar, Departamento de Cómputo Científico y Estadística, Venezuela*; [b] *Universidad de Carabobo, Departamento de Matemáticas, Venezuela.*

*(Received 00 Month 200x; In final form 00 Month 200x)*

In this paper we present an algorithm for solving convex quadratic optimization problems with a single linear equality constraint and box restrictions. Our proposal is based on a modification of the primal-dual interior-point method for quadratic programming. The primal-dual iterates generated by the algorithm have non negative components and the search direction is obtained by approximating the Hessian of the objective function by a multiple of the identity matrix, using the Barzilai-Borwein spectral length. As a result, there is no need to solve a linear system per iteration and only matrix-vector multiplications are computed. Therefore, the algorithm proposed is particularly well suited for optimization problems where the Hessian of the objective function is large and dense. Problems of this type arise, for example, in the context of Support Vector Machines (SVM) that is a technique based on supervised learning developed to classify data. To improve the efficiency of our method, we include an identification procedure. We show numerical results of our algorithm when applied to real life SVM problems and compare with the ASL method by Gonzalez-Lima, Hager, and Zhang (2011).

*Keywords:* primal-dual interior-point methods for quadratic programming, Support Vector Machines (SVM), continuous knapsack problems.

## 1 Introduction

In this paper we consider the convex quadratic programming problem with continuous knapsack constraints

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^tQx - e^tx \\ \text{subject to} & y^tx = b, \qquad 0 \leq x_i \leq C,\ i = 1, ..., n \end{array} \tag{1}$$

where $x, y \in \mathbb{R}^n$, $b, C \in \mathbb{R}$, $Q \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix, and

[a] Corresponding author. Email: mlima@usb.ve

$e$ is the $n$-vector of all ones.

Throughout this paper, we denote the components of a vector by subscripts and the iteration counters of the algorithms by superscripts. As shorthand, $p \leq q$, $\min(p)$, and $\min(p/q)$, for any two real $n$ vectors, stand for, respectively, $p_i \leq q_i$, $\min_i(p_i)$, and $\min_i(p_i/q_i)$ for all $1 \leq i \leq n$ (similar when using max). We use $(p, q)$ as a notation for the vector $(p^t, q^t)^t$ and $||.||$ is the standard Euclidean norm of a vector. To simplify notation we suppress the evaluation points, and the dimension of the zero vector, when it is clear from the context.

We are interested in the case when the Hessian matrix $Q$ of the objective function is large and dense (and possibly bad conditioned). Problems as this one arise in the context of Support Vector Machines (SVM). This is a classification technique based on the supervised learning theory developed by V. Vapnik [8, 33] used to separate data into two or more different classes in order to classify a new object. SVM has become a very popular data mining technique, and it is being used for solving many real life applications as, for example, isolated handwritten digit recognition ( [8], [29], [30], [5], [4]), object recognition [3], speaker identification [28], face detection images ( [25], [26]), text categorization [18], and some nonlinear least squares problems as the inverse density estimation problem [34].

Traditional methods for quadratic programming do not seem appropriated for the SVM applications since they usually need to store $Q$ or to solve a large linear system per iteration, as is the case for the path-following primal-dual interior-point methods. There are, however, some attempts in the literature to use these interior methods for solving SVM problems [16], [15], [35], [20]. Efficient low cost approaches that are not path following interior algorithms have also been developed, as LIBSVM [14], SOM [27], GPDT [31, 32, 37], $SVM^{light}$ [18, 22], ASL [17]. All these approaches combine the decomposition of the problem into smaller optimization problems of the same structure and low cost algorithms to solve the small problems.

The low cost algorithm proposed in this paper is a modification of the general path-following primal-dual interior point method (for a description we refer to [24]). Inspired by the ASL method proposed by Gonzalez-Lima et al in [17], we approximate the Hessian at each iteration by the identity matrix multiplied by the Barzilai and Borwein spectral length [1] such that the method does not explicitly use second order information and, contrary to ASL, there is no need to solve a nonlinear equation at each iteration. Moreover, the generation of primal and dual variables simultaneously in our method allows us to include an identification procedure that reduces significantly the time required for convergence. The algorithm obtained is primal feasible and its performance depends on the dual infeasibility error.

The structure of this paper is the following. In the next section we present some preliminaries related with the quadratic programming problem to solve and the algorithm is described. We introduce to the Support Vector Machines (SVM) theory in Section 3. The following section contains the identification procedure added to the method. Section 5 shows the numerical performance of our algorithm when solving real life SVM problems. There, we compare how the identification procedure reduces the CPU running time of the algorithm when applied to these problems while keeping the generalization errors. The section ends with comparisons with the ASL algorithm. The last section contains remarks and conclusions.

## 2 The Algorithm

The dual problem associated to (1) can be stated as

$$\begin{aligned} \underset{u,s}{\text{maximize}} \quad & -\frac{1}{2}x^tQx + bs - Cu^te \\ \text{subject to} \quad & Qx - t - ys + u = e \\ & u, t, x \geq 0. \end{aligned} \tag{2}$$

A primal and dual feasible point is a primal-dual solution if and only if the gap between the objective functions of (1) and its dual problem (2) is equal zero. This is, $x^tt + u^tw = (\frac{1}{2}x^tQx - e^tx) - (-\frac{1}{2}x^tQx + bs - Cu^te) = 0$ with $w = C - x$. Therefore, $x^tt + u^tw$ is called the duality gap.

This condition, together with the primal and dual feasibilities, are the well known first-order optimality (or Karush-Kuhn-Tucker) conditions:

$$F(x,t,u,w,s) = \begin{pmatrix} Qx - e - sy - t + u \\ y^tx - b \\ C - w - x \\ XTe \\ UWe \end{pmatrix} = 0; \;\; (x,t,u,w) \geq 0. \tag{3}$$

Here, $X, T, U, W$ are diagonal matrices with the vectors $x, t, u, w$ in the diagonal, respectively.

The search direction $\Delta$ used in primal-dual interior-point methods is a perturbed and damped Newton direction found by solving the linear system $F'\Delta = -F_\mu = -F + \mu(0,0,0,e,e)$ with $F'$ denoting the Jacobian of $F$.

This is,

$$\begin{pmatrix} Q & -y & -I & I & 0 \\ y^t & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & -I \\ T & 0 & X & 0 & 0 \\ 0 & 0 & 0 & W & U \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta t \\ \Delta u \\ \Delta w \end{pmatrix} = - \begin{pmatrix} Qx - e - sy - t + u \\ y^t x - b \\ C - w - x \\ TXe - \mu e \\ UWe - \mu e \end{pmatrix} = - \begin{pmatrix} r_e \\ r_b \\ r_c \\ r_{tx} \\ r_{uw} \end{pmatrix} \tag{4}$$

Note that a point $(x, t, s, u, w) > 0$ is primal feasible if $r_b = r_c = 0$, and dual feasible if $r_e = 0$. The point is said to be at the central path if it also satisfies $r_{tx} = r_{uw} = 0$. Path-following primal-dual methods attempt to generate iterates close to the central path that converge to the solution set while gradually reduce $\mu$ to zero.

In our approach, at each iteration we substitute the Hessian $Q$ in the matrix of system (4), by $\hat{\lambda} I$ for some $\hat{\lambda} > 0$ and $I$ the identity matrix. Once the direction is obtained, the customary fashion for primal-dual methods is used in order to find the step lengths and the centering parameter $\mu$. This is, the step lengths are found such that the iterates $(x, t, u, w)$ have all positive components at each iteration, and $\mu = \sigma \frac{x^t t + u^t w}{2n}$ for some $\sigma \in (0, 1)$. It is easy to see that the search direction $\Delta$ of our proposal solves the system

$$F' \Delta = -F_\mu + \begin{pmatrix} (Q - \hat{\lambda} I)\Delta x \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5}$$

Using direct calculations we can obtain from (5) the search direction without solving a linear system of equations:

$$\Delta s = \frac{v^t b_z + r_b}{-v^t y} \tag{6}$$

$$\Delta x_i = ((b_z)_i + y_i \Delta s)/d_i, \quad i = 1, .., n \tag{7}$$

with

$$b_z = -r_e - X^{-1} r_{tx} + W^{-1}(r_{uw} + U r_c), \tag{8}$$

$$d = De = (\hat{\lambda} I + X^{-1} T + W^{-1} U)e, \text{ and } v = D^{-1} y. \tag{9}$$

Also,

$$\Delta t = X^{-1}\left(-r_{tx} - T\Delta x\right) \tag{10}$$

$$\Delta w = r_c - \Delta x \tag{11}$$

$$\Delta u = W^{-1}\left(-r_{uw} - U\Delta w\right) \tag{12}$$

It is clear that the choice of $\hat{\lambda}$ is key for the performance of the method. Given $\Delta x$, we seek to choose this parameter such that it minimizes the Euclidean norm of the residual term $(Q - \hat{\lambda} I)\Delta x$ in order to preserve the good behavior of the standard primal-dual methods. As a safeguard, we also ask for this parameter to be greater than a given small number $\lambda_{min} > 0$. Hence,

$$\hat{\lambda} = \arg\min_{\lambda \geq \lambda_{min}} ||(Q - \lambda I)\Delta x|| = \max\{\lambda_{min}, \frac{(\Delta x)^t Q \Delta x}{(\Delta x)^t \Delta x}\}. \tag{13}$$

Observe that this choice of $\hat{\lambda}$ is the one that minimizes the secant equation $||Q(x^+ - x) - \lambda(x^+ - x)||$ for $x^+ = x + \alpha\Delta x$, used in the context of Quasi-Newton methods (e.g. Barzilai and Borwein [1]). It is a Rayleigh quotient bounded by the lower and upper eigenvalues of $Q$, so we will call it the spectral length.

With all these considerations, the Low-Cost interior point algorithm proposed follows.

*Algorithm 2.1* Low-Cost (LC) Primal-Dual Interior-Point Algorithm

Given $(x^0, t^0, u^0, w^0) > 0, s^0, (\lambda^0, \lambda_{min}) > 0$, $\tau^k \in [0.9, 1)$. For $k = 0, 1, 2...$ until convergence.

1 Compute $r_e^k, r_b^k, r_c^k, r_{tx}^k$ and $r_{uw}^k$ defined by (4) with $\mu^k = \sigma^k \frac{(x^k)^t t^k + (u^k)^t s^k}{2n}$ some $\sigma^k \in (0, 1)$.

2 Compute the search direction $\Delta = (\Delta x^k, \Delta t^k, \Delta s^k, \Delta u^k, \Delta w^k)$ by using (6) - (12).

3 Compute the step lengths $\alpha_{primal}^k, \alpha_{dual}^k$ such that $(x^{k+1}, t^{k+1}, u^{k+1}, w^{k+1}) > 0$:

$$\alpha_{primal}^k = \left[\max(1, -\Delta x^k / x^k, -\Delta w^k / w^k)\right]^{-1}$$

$$\alpha_{dual}^k = \left[\max(1, -\Delta t^k / t^k, -\Delta u^k / u^k)\right]^{-1}$$

4 Update

$$(x^{k+1}, w^{k+1}) = (x^k, w^k) + \tau^k \alpha_{primal}^k (\Delta x^k, \Delta s^k),$$

$$(s^{k+1}, t^{k+1}, u^{k+1}) = (s^k, t^k, u^k) + \tau^k \alpha_{dual}^k (\Delta s^k, \Delta t^k, \Delta u^k)$$

5 Compute the spectral length $\lambda^{k+1} = \max\{\lambda_{min}, \frac{(x^{k+1}-x^k)^t Q(x^{k+1}-x^k)}{(x^{k+1}-x_k)^t(x^{k+1}-x^k)}\}$

In the following section we study some theoretical properties of the LC algorithm and state conditions under which it converges.

## 2.1 *Properties*

It is easy to check that the LC is a primal feasible algorithm, i.e. if $r_b^k = r_c^k = 0$ then $r_b^{k+1} = r_c^{k+1} = 0$. The convergence performance will depend on the value of $r_e$ which measures the dual infeasibility. Next propositions establish relationships of the primal direction $\Delta x$ and the dual infeasibility $r_e$. We assume that the point has positive entries $(x, t, u, w)$.

Proposition 2.1 *Let $r_b = 0$. The direction $\Delta x = D^{-\frac{1}{2}} P D^{-\frac{1}{2}} b_z$ where $D$ is defined in (9) and $P = I - \frac{(D^{-\frac{1}{2}}y)(D^{-\frac{1}{2}}y)^t}{(D^{-\frac{1}{2}}y)^t(D^{-\frac{1}{2}}y)}$ is a projection matrix.*

The proof follows straightforward from (9). This proposition says that $\Delta x$ is a projection of vector $b_z$ onto the primal feasible region of problem (1). Then if $||\Delta x||$ is small and $b_z$ is not a multiple of $y$, it should also have small norm.

Observe that $b_z = r_e$ when $r_b = r_{tx} = r_{uw} = 0$. Therefore, if the evaluation point is primal feasible and the values of $t_i x_i$ and $u_i w_i$ are balanced, close to the gap, the norm of the primal direction should be bounded by the norm of the dual infeasibility, as is shown next.

Proposition 2.2 *Let us consider a positive, primal feasible point. If $||X^{-1}r_{tx}|| \leq ||F||/2$ and $||W^{-1}r_{uw}|| \leq ||F||/2$ then $||\Delta x|| \leq \frac{2}{\max(\lambda_{min}, \lambda^Q_{min})}||F||$ with $\lambda^Q_{min}$ the lower eigenvalue of the matrix $Q$ and $\lambda_{min}$ as in (13).*

*Proof* First, observe that $b_z = -r_e + r$ with $r = -X^{-1}r_{tx} + W^{-1}r_{uw}$. And, $||r_e|| \leq ||F||$. Then, $||\Delta x|| = ||D^{-\frac{1}{2}} P \hat{D}^{-\frac{1}{2}}(-r_e + r)||$. Since $D$ is diagonal and $P = P^t = PP$ is a projection matrix, $||\Delta x|| \leq \frac{1}{\hat{\lambda} + min(\frac{t}{x} + \frac{u}{w})}|| - r_e + r||$. Since $min(\frac{t}{x} + \frac{u}{w}) > 0$, $\lambda_{min} \leq \hat{\lambda}$, $\lambda^Q_{min} \leq \hat{\lambda}$, and $|| - r_e + r|| \leq ||r|| + || - r_e||$ we have that $||\Delta x|| \leq \frac{1}{max(\lambda_{min}, \lambda^Q_{min})}(||F|| + ||r||)$. The proof concludes by using the assumption. □

The following proposition will be used for our convergence result. It shows that under some assumptions, the LC search direction is descent for the merit function $\frac{1}{2}||F||^2$.

Proposition 2.3 *Let $f = \frac{1}{2}||F||^2 \neq 0$. If $F'$ is non singular and $||(Q - \hat{\lambda}I)\Delta x|| \leq \hat{\eta}||F||$ with $\sigma + \hat{\eta} \in (0, 1)$, the search direction $\Delta$ computed in the Step (2) satisfies that $\Delta^t \nabla f < 0$*

*Proof* Since $||r_e|| \leq ||F||$, it suffices to prove the proposition if $||(Q - \hat{\lambda}I)\Delta x|| \leq \eta ||F||$. From (5), observe that $F'\Delta = -F + \mu(0,0,0,e,e) + (Q - \lambda I)(\Delta x, 0,0,0,0)$ and the gradient $\nabla f = (F')^t F$. Then, $(\nabla f)^t \Delta = (\nabla f)^t - (F')^{-1}F + (F')^{-1}(0,0,0,\mu e, \mu e) + (F')^{-1}(Q - \lambda I)(\Delta x, 0,0,0,0) = -F^t F + F^t(0,0,0,\mu e,\mu e) + F^t(\Delta x,0,0,0,0)$. This says that $(\nabla f)^t\Delta = -||F||^2 + \sigma\frac{((x^t)t+(u^t)w)^2}{2n} + r_e^t(Q - \lambda I)\Delta x$. Since $\frac{(x^t)t+(u^t)w}{\sqrt{2n}} \leq ||F||$, $r_e^t(Q - \lambda I)\Delta x \leq ||r_e|| ||(Q - \lambda I)\Delta x||$ (by Cauchy-Schwarz inequality), we get that $(\nabla f)^t\Delta \leq (-1 + \sigma + \eta)||F||^2 < 0$, using the assumption. □

The same result is obtained if $||(Q - \hat{\lambda}I)\Delta x|| \leq \hat{\eta}||r_e||$ or $||(Q - \hat{\lambda}I)\Delta x|| \leq \frac{\hat{\eta}((x^t)t+(u^t)w)}{\sqrt{2n}}$, since $||r_e|| \leq ||F||$ and $\frac{(x^t)t+(u^t)w}{\sqrt{2n}} \leq ||F||$.

Next result shows that the assumption of the previous proposition is satisfied in some cases. The proof follows from Proposition 2.1.

Proposition 2.4 *If* $\lambda^Q_{max} - \hat{\lambda} < \frac{1-\sigma}{2}\max(\lambda^Q_{min}, \lambda_{min})$ *with* $\lambda^Q_{max}$ *the upper eigenvalue of* $Q$, $||X^{-1}r_{tx}|| \leq ||F||/2$ *and* $||W^{-1}r_{uw}|| \leq ||F||/2$. *Then,* $||(Q - \hat{\lambda}I)\Delta x|| \leq \eta||F(z)||$ *with* $\sigma + \eta \in (0,1)$.

Because of the substitution of $Q$, the LC algorithm is a dual infeasible algorithm. It can be seen as an inexact Newton method with residual term $r^k = ((Q - \hat{\lambda}^k I)\Delta x^k, 0, 0, \mu^k e, \mu^k e)$ at each iteration $k$. Following Eisenstat and Walker [11] we could prove global convergence of the algorithm under the assumption that $||r^k|| \leq \eta^k||F^k||$ with $\eta^k \in (0,1)$ and there is enough decrease of the merit function $\frac{1}{2}||F||^2$ when considering the search direction as $\alpha^k\Delta^k$ with $\alpha^k = \tau^k \min(\alpha^k_{primal}, \alpha^k_{dual})$. Therefore, the boundedness away from zero of the step lengths $\alpha^k$ is key for the proof of convergence. For primal-dual interior-point methods, it is customary to prove that the step lengths $\alpha^k$ are never zero at the limit by assuming that the iterates are confined in some neighborhood of the central path, as well as some boundedness conditions are satisfied. Our next theorem encompasses these ideas, it follows Bellavia [2]. We will use the following notation $G(z^k) = (r_e(z^k), r_b(z^k), r_c(z^k))$ for all $k \geq 0$. And, $z(\alpha) = z + \alpha\Delta z = (x,t,s,u,w) + \alpha(\Delta x, \Delta t, \Delta s, \Delta u, \Delta w)$.

Theorem 2.5 *Let us consider the LC algorithm with chosen data* $z^0 = (x^0, t^0, s^0, u^0, w^0)$, $\eta_{max}$, $\beta$, $\hat{\theta} \in (0,1)$. *Let us denote* $\hat{\tau}_1 = \min(T_0X_0e)/[(t_0^t x_0)/n]$, $\hat{\tau}_2 = \min(U_0W_0e)/[(u_0^t w_0)/n]$, $\hat{\tau}_3 = (t_0^t x_0)/||G(z_0)||$, $\hat{\tau}_4 = (u_0^t w_0)/||G(z_0)||$. *Additionally, choose* $\gamma^0 \in [1/2, 1)$ *and* $\gamma^k \in [1/2, \gamma^{k-1})$ *such that* $\max(\hat{\tau}_1, \hat{\tau}_2)\gamma^k < 1$ *for all* $k \geq 0$.

*Consider the LC algorithm and choose the step lengths in the following way*

- *Choose* $\hat{\alpha_1}^k, \hat{\alpha_2}^k, \hat{\alpha_3}^k, \hat{\alpha_4}^k$ *such that*

$$M(x,t,\hat{\tau}_1,\gamma^k;\hat{\alpha_1}^k), M(u,w,\hat{\tau}_1,\gamma^k;\hat{\alpha_2}^k) \geq 0$$

*with*

$$M(x,t,\tau,\gamma;\alpha) = \min(T(\alpha)X(\alpha)e) - \tau\gamma t(\alpha)^t x(\alpha)/n.$$

*And,*

$$D(x,t,\hat{\tau_3},\gamma^k;\hat{\alpha_3}^k), D(u,w,\hat{\tau_4},\gamma^k;\hat{\alpha_4}^k) \geq 0$$

*with*

$$D(x,t,\tau,\gamma;\alpha) = t(\alpha)^t x(\alpha) - \tau\gamma||G(z(\alpha))||.$$

*Let* $\hat{\alpha}^k = \min(\hat{\alpha_1}^k, \hat{\alpha_2}^k, \hat{\alpha_3}^k, \hat{\alpha_4}^k)$.

- *Let* $p^k = \hat{\alpha}^k \Delta^k, \eta^k = 1 - \hat{\alpha}^k(1-\hat{\eta}^k)$. *While* $||F(z^k + p^k)|| > (1 - \beta(1 - \eta^k))||F(z^k)||$, *set* $p^k = \hat{\theta} p^k$ *and* $\eta^k = 1 - \hat{\theta}(1-\eta^k)$. *Update* $z^{k+1} = z^k + p^k$. *Return to Step 1.*

*Let us assume*

*(i)* $||(Q - \hat{\lambda}^k I)\Delta x^k|| \leq \hat{\eta}^k ||F(z^k)||$ *with* $\sigma^k + \hat{\eta}^k \in (0, \eta_{max})$.
*(ii) The iteration sequence* $\{z^k\}$ *is bounded.*
*(iii)* $\sigma^k > \max(\hat{\tau_3}, \hat{\tau_4})\gamma^k \hat{\eta}^k$

*Then, the sequence* $\{\hat{\alpha}^k\}$ *is bounded away from zero and* $\{||F(z^k)||\}$ *converges to zero.*

*Proof* The proof follows [2] observing that in our algorithm $\hat{r_k}^{(1)} = (Q - \hat{\lambda}^k I)\Delta x^k$ and $\hat{r_k}^{(2)} = 0$. And, that the assumptions (A1)-(A4) from [2] are satisfied for the convex, quadratic problem considered. □

We are particularly interested in applying the LC algorithm to the problems from Support Vector Machines to be introduced in the following section.

## 3 Support Vector Machines

In this section, we give a brief introduction to Support Vector Machines (SVM).

Let us consider the given data set of points $\{Z_i \in \mathbb{R}^m \quad i = 1, \ldots, n\}$. Each point belongs to a class identified by a corresponding given value $y_i = 1$ or $y_i = -1$ (in this paper we are only considering two classes). The dimension $m$ denotes the number of " attributes" of the data. The aim for SVM is to find a way to separate the points in different classes and to use this separation in order to classify a new given point. Hence, the ultimate goal is to classify any point by using the separation obtained with the given initial data set named training set. If the training set is linearly separable (that is, there exists an hyperplane that separates the points into two different classes), the SVM theory seeks to find the hyperplane with normal vector $v$ that maximizes the separation margin between classes constrained to which class each point belongs. The corresponding optimization problem can be written as

$$\begin{array}{l} \underset{v,d}{\text{minimize}} \;\; \frac{1}{2}\|v\|^2 \\ \text{subject to } y_i(v^t Z_i + d) \geq 1 \;\; \forall\, i = 1, ..., n, \end{array} \tag{14}$$

where $v \in \mathbb{R}^m$ and $d \in \mathbb{R}$.

The support vectors are the ones from the training data set where at least one of the constraints in the feasible set of (14) is satisfied as equality.

If the data set is linearly nonseparable (this is, that there does not exist a solution of problem (14)) then two variants are considered. Perturbation variables are included in order to relax the constraints so that a margin of error in the classification is accepted. Additionally, in the nonseparable case, the nonlinear decision surface is computed by mapping the input variable $Z$ into a higher dimensional "feature space", through a function $\Phi$, and by working with linear classification in that space. So, in the non separable case, the primal problem is obtained by substituting the variable $Z$ with the new "feature vector" $\Phi(Z)$. In practice, an explicit description of $\Phi$ is not needed since the solution can be found by solving the dual problem. What is really needed is to find a function that preserves, in higher dimensional spaces, the properties of the inner product. These are the kernel functions (we refer to [4] for the interested reader). Formally, a kernel function $K$ is a real function on $\mathbb{R}^m \times \mathbb{R}^m$ such that for some map $\Phi$ from $\mathbb{R}^m$ into a Hilbert space, we have $K(Z, h) = \Phi(Z)^t \Phi(h)$.

The dual problem, after relaxing the constraints and considering the higher dimensional feature space, is given by

$$\begin{array}{l} \underset{x}{\text{maximize}} \;\; e^t x - \frac{1}{2} x^t Q x \\ \text{subject to } y^t x = 0 \\ \qquad\qquad\;\; 0 \leq x_i \leq C \quad \text{for } i = 1, \ldots, n \end{array} \tag{15}$$

where $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix with positive diagonal, defined as $Q_{ij} = y_i y_j K(Z_i, Z_j)$.

One advantage of solving (15) is that constraints are simpler than in the original problem but more importantly, the dimension of the feature space for the classification can be increased without increasing the dimension of the optimization problem to solve.

Let $x^*$ be a solution of (15). Because of the optimality relations between primal and dual SVM problems, the hyperplane that separates the data in the high dimensional space, determined by the normal vector $v^*$ and the intersection with the axis, $d^*$, satisfies $v^* = \sum_{i=1}^n x_i^* y_i \Phi(Z_i)$ and $d^* = 1 - \max_{y_j=1} (v^*)^t \Phi(Z_j)$. Hence, the function used to classify a new point $Z$ (according to the side of the hyperplane where it falls) can be written as

$$gen(Z) = \text{sign}(\sum_{i=1}^{n} y_i x_i^* K(Z, Z_i) + d^*).$$

This function is called the decision or generalization function. Observe that in this sum, only the $x_i^* \neq 0$ are important. Because of the strict complementarity optimality conditions, these components correspond to the $Z_i$ support vectors. This is saying that the classification of a new point can be made just by selecting a (hopefully small) group of points from the large original data. Therefore, the main objective for SVM is to find these vectors: the support vectors.

In the SVM literature, the classification (or generalization) error means the percentage of new points that are correctly classified by the decision function. Usually, the data set is separated in a group of training points and a group of test points such that this classification error can be calculated. In practice, different kernel functions are known and they are used for SVM problems. Experimental knowledge and data distribution may suggest the kernel function to use. But to find the right kernel for each application is a difficult task that goes beyond our work. In our numerical experimentation we use the kernels suggested in the literature.

We refer to the book of Cristianini and Shawe-Taylor [9] for details on the precedent discussion.

Observe that the size of the Hessian of the objective function in (15) is equal to $n \times n$, with $n$ the number of training samples, therefore it is usually large for real life applications. Besides, the entries are rarely zero so the matrix is dense. The rank of the matrix $Q$ depends on the dimension of the feature space given by the kernel. For instance if the data is linearly separable, $\text{rank}(Q) = m$ if $m < n$ and $Q$ is singular. Larger this dimension of the feature space, larger is the rank of $Q$ and better conditioned is the matrix. This may impact the algorithm used to solve the problem.

Next, we will introduce an identification procedure added to our algorithm such that the variables that are not zero at the solution can be identified in advance. This procedure allows us to reduce the size of the problem and to improve the time required for solving it.

## 4 Identification procedure

There are in the literature several efficient approaches in the context of primal-dual interior point methods for identification of zero and nonzero variables at the solution (for example, El-Bakry et al [12], Facchinei et al [13]). Jung et al [19] proposed an approach that fits in the SVM environment [20] and we combine this approach with the one in [12].

The basic idea is to look at $\Theta_i = \frac{t_i}{x_i} + \frac{u_i}{w_i}$. If $0 < x_i < C$ at solution, $i$ corresponds to a support vector. Because of the optimality conditions (see (3)) when $0 < x_i$ then $t_i = 0$. And, if $0 < w_i$ then $u_i = 0$. Hence, at optimality, $\Theta_i = 0$ (or equivalently, $\Theta_i^{-1} = \frac{w_i x_i}{w_i t_i + u_i x_i}$ goes to infinity). So, let us consider an iterate $(x, t, w, u, s)$ close to the solution set. We estimate the number $a$ of support vectors by finding lower and upper bounds ($a_L$ and $a_U$, respectively) for $a$. Then, the position of the $a$ smaller values of the vector $\Theta$ correspond to the predicted support vectors.

The selection of the upper bound $a_U$ is based on the primal Tapia indicator sequence $\{x_{i+1}^k / x_i^k\}$. It can be seen from [12] that, under standard assumptions, this sequence converges to 1 if $x_i > 0$ at solution, and converges to 0 if not. Therefore we consider $a_U = |\{i : x_{i+1}/x_i > 0.9\}|$.

In order to set the lower bound $a_L$ we look to identify the larger values of $\Theta^{-1}$ when the iterates are close to the solution. We follow a heuristic used in [20]. Since in primal-dual interior point methods, under the strict complementarity assumption, for all $i$, $\{(\Theta_i^k)^{-1}\}$ diverges to infinity as fast as $\mu^{-1}$ or converges to zero as $\mu$, the components of this vector are said to be large if they are greater than a multiple $\theta$ of $\sqrt{\mu}$. In this way, a significant gap between the large and small values of this vector can be established when close to the solution set. In our experimentation we consider $\theta = 100$ so $a_L = |\{i : \Theta_i^{-1} \geq 100\sqrt{\mu}\}|$.

The number of predicted support vectors is set to $a = \max(a_L, \min(\lceil \rho n \rceil, a_U))$ where $n$ is the size of the data set and $\rho = \mu^{\frac{1}{4}}$ a heuristic used to reduce the size of $a_U$ according to the closeness to the solution.

Finally, the identified candidates for support vectors are the ones corresponding to the $a$ smaller values of $\Theta$ which form the set $\mathbf{A}(\Theta, a) = \{A | A \subseteq \{1, ..., n\}, |A| = a \quad \text{and} \quad \Theta_i \leq \Theta_j \quad \forall i \in A, j \notin A\}$.

Because there are data in two classes, the predicted support vectors should belong to both in order to avoid miss classification if these vectors belong only (or mainly) to one class. Therefore the procedure looks for numbers $a^+$ and $a^-$ (and corresponding support vectors) for each class in an analogous way that described above. That is, $a^+ + a^- = a$ where

$$a_L^+ = |\{i : \Theta_i^{-1} \geq \theta\sqrt{\mu} \quad ; \quad y_i = +1\}|$$
$$a_L^- = |\{i : \Theta_i^{-1} \geq \theta\sqrt{\mu} \quad ; \quad y_i = -1\}|$$

and

$$a^+ = \max\left(a_L^+, \min\left(\left\lceil \frac{\min(\lceil \rho n \rceil, a_U)}{2} \right\rceil, n^+\right)\right)$$
$$a^- = \max\left(a_L^-, \min\left(\left\lceil \frac{\min(\lceil \rho n \rceil, a_U)}{2} \right\rceil, n^-\right)\right)$$

Here, $a_U$ is defined as before, $n^+$ and $n^-$ are the number of elements corresponding to $y_i = 1$ or $-1$. Then, $\mathbf{A}^+(\Theta, a^+)$ and $\mathbf{A}^-(\Theta, a^-)$ stand for the components of the vector $\Theta$ with smaller $a^+$ and $a^-$ values, respectively.

It is known that the predictor-corrector version (introduced by Mehrotra [23]) reduces the number of iterations for convergence of primal-dual interior point methods. Our first attempt was to consider the predictor-corrector directions in our algorithm. However preliminary experimentation showed that with this choice, the sequences $\{x_i^k t_i^k\}$ and $\{w_i^k u_i^k\}$ converge very fast to zero, much before the iterates have reached dual feasibility. Therefore, when using these directions, the iterates get stuck in the wrong faces without obtaining convergence. However, this bad behavior is good for the identification since the components of $\Theta$ goes to zero or infinity very fast. Therefore, we modify the Low-Cost algorithm from Section 2 by including the use of predictor-corrector directions (and a related centering parameter) but only when the iterates are far from being feasible. Once the duality gap is close to zero we perform the identification. Algorithm 4.1 describes the whole procedure.

We perform the identification only once. With the selected variables (sets $\mathbf{A}^+(\Theta, a^+)$ and $\mathbf{A}^-(\Theta, a^-)$) a new problem is formed and solved by using the Low-Cost algorithm (Algorithm 2.1).

---

*Algorithm 4.1* Low-Cost Identification (LC-I) Primal-Dual Interior-Point Algorithm

---

Given $(x^0, t^0, u^0, w^0) > 0$, $s^0, (\lambda^0, \lambda_{min}) > 0$, and $\eta^k \in [0.9, 1)$. For $k = 0, 1, 2...$ until convergence.

```
1 Compute $r_e^k, r_b^k, r_c^k, r_{tx}^k$ and $r_{uw}^k$ defined by (4) with $\mu^k = 0$.
2 Compute the predictor step $(\Delta x^k, \Delta t^k, \Delta s^k, \Delta w^k, \Delta u^k)$ by solving (6) - (12).
```

```
3  Compute the step length
       $$\alpha^k = \eta^k \left[\max(1, -\Delta x^k/x^k, -\Delta w^k/w^k, -\Delta t^k/t^k, -\Delta u^k/u^k)\right]^{-1}$$
4  Compute $\mu^k = \frac{(x^k)^t t^k + (u^k)^t w^k}{2n}$ and $\mu_a^k$ as
       $$\mu_a^k = \frac{(x^k + \alpha^k \Delta x^k)^t (t^k + \alpha^k \Delta t^k) + (u^k + \alpha^k \Delta u^k)^t (w^k + \alpha^k \Delta w^k)}{2n}$$
5  Compute the centering parameter $\sigma^k = \left(\frac{\mu_a^k}{\mu^k}\right)^3$
6  **If** $||r_c^k|| > 10^{-5}$ and $||r_b^k|| > 10^{-5}$ **then**
       Compute the centering direction $(\Delta x^k, \Delta t^k, \Delta s^k, \Delta u^k, \Delta w^k)$ by using (6) - (12)
       with $r_{tx}^k = T^k X^k e - \sigma^k \mu^k + \Delta X^k \Delta T^k e$ and $r_{uw}^k = U^k W^k e - \sigma^k \mu^k + \Delta U^k \Delta W^k e$.
   **else**
       Compute the centering direction $(\Delta x^k, \Delta t^k, \Delta s^k, \Delta u^k, \Delta w^k)$ by using (6) - (12)
       with $r_{tx}^k = T^k X^k e - \sigma^k \mu^k$ and $r_{uw}^k = U^k W^k e - \sigma^k \mu^k$.
7  Compute the step lengths $\alpha_{primal}^k, \alpha_{dual}^k$ such that $(x^{k+1}, t^{k+1}, u^{k+1}, w^{k+1}) > 0$:
       $$\alpha_{primal}^k = \left[\max(1, -\Delta x^k/x^k, -\Delta w^k/w^k)\right]^{-1}$$
       $$\alpha_{dual}^k = \left[\max(1, -\Delta t^k/t^k, -\Delta u^k/u^k)\right]^{-1}$$
8  Update
       $$(x^{k+1}, w^{k+1}) = (x^k, w^k) + \eta^k \alpha_{primal}^k (\Delta x^k, \Delta w^k),$$
       $$(s^{k+1}, t^{k+1}, u^{k+1}) = (s^k, t^k, u^k) + \eta^k \alpha_{dual}^k (\Delta s^k, \Delta t^k, \Delta u^k)$$
9  Compute the spectral length $\lambda^{k+1} = \max(\lambda_{min}, \frac{(x^{k+1}-x^k)^t Q (x^{k+1}-x^k)}{(x^{k+1}-x^k)^t (x^{k+1}-x^k)})$.
10 **If** $sigfig > 1$ (see (16)) compute $a^+, a^-$ **then**
       if $a^+ \neq 0$ and $a^- \neq 0$ then compute $A^+(\Theta, a^+)$ and $A^-(\Theta, a^-)$. Form the new
       problem using $A^+, A^-$ and solve it with Algorithm 2.1.
       else, go to Step 1.
```

## 5 Numerical Experimentation

In this section we study the performance of Algorithms 2.1 and 4.1 (Low-Cost interior point algorithm without and with identification). The experiments were done on a personal computer with an i3 processor, 4 Gb of Ram memory, and operating system Ubuntu 12.04 LTS. The algorithms were implemented in C/C++ language and they were called from the main file of the version 3.14 code from the LIBSVM library [7] for SVM (available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm). The code to identify the variables was implemented on Matlab.

The behavior of primal-dual interior point methods is very much affected by the choice of the initial point. This is why in our approach we start with an infeasible primal and dual point, with a value depending on $C$ such that it balances the component wise product of the vectors $x$ and $t$ as well as $w$ and $u$. We choose $x^0 = Ce, w^0 = x^0; t^0 = (1/C)e, u^0 = t^0$, $s^0 = 0$. The initial spectral length is chosen as $\lambda^0 = \max(\lambda_{min}, ||Qx^0 - e||_\infty)$ and $\lambda_{min}$ equal to $10^{-5}$. We use the damped parameter $\eta^k = 0.95$ for all $k$.

We say that the algorithm converges if an iterate $(x, s, t, u, w)$ generated by the algorithm satisfies approximately the KKT optimality conditions (3). This is, if for tolerance $\epsilon = 10^{-5}$ it satisfies $(\frac{\|y^t x - b\|}{\|b\|+1} \& \frac{\|C-x-w\|}{\|C\|+1} \& \frac{\|Qx-e-sy-t+u\|}{\|e\|+1} < \epsilon)$. and any of the following conditions is satisfied

$$\left| \frac{x^t t + u^t w}{2n} \right| < \epsilon \text{ or } sigfig > 6.$$

This condition measures if the gap is small by computing the relative difference among the objective values of the primal and dual problem or their significant figures, since

$$sigfig = \max\left( -\log_{10} \frac{|\, primal_{obj} - dual_{obj} \,|}{|\, primal_{obj} \,| + 1}, 0 \right), \qquad (16)$$

where $primal_{obj}$ is the objective function value of problem (1) and $dual_{obj}$ the objective function value of the dual problem (2).

In our initial experiments we wanted to test the effectiveness of the LC algorithm solving the SVM problems. We did not use any decomposition scheme so we solved medium size problems from the LIBSVM library and UCI repository [21]. The dimensions of the tested problems are included in the tables.

The kernel functions (see Section 3) considered in our experimentation are

- Linear: $K(Z_i, Z_j) = Z_i^t Z_j$;
- Polynomial: $K(Z_i, Z_j) = (\frac{1}{n} Z_i^t Z_j)^3$.
- Gaussian: $K(Z_i, Z_j) = exp(-\frac{1}{n}||Z_i - Z_j||^2)$.

The Low-Cost algorithm (without identification, named LC) is able to solve all the problems tested. We use in all the tests, the centering parameter $\sigma^k = 0.5$. The iteration sequence generated by the algorithm becomes primal feasible very fast and it is clear from all the extensive experimentation that the quotient $\frac{\|(Q-\lambda^k I)\Delta x^k\|}{||F^k||}$ is less than one for all (or an infinite subset) iterations $k$. The dual feasibility is, of course, the most difficult to reach and the algorithm can take many iterations trying to reduce the dual infeasibility. Therefore, improvements of the algorithm should address this issue.

We compare the performance of the LC with the ASL algorithm by running the ASL-1.1 C code publicly available at http://www.math.ufl.edu/˜hager to the same problems. When there is a symbol "-" for ASL, it means that the algorithm is not able to solve the problem, but the code does not specify if this is because of memory limitations or maximum number of iterations exceeded. It is important to say that the option for decomposition that the ASL-1.1 code considers, is not activated for our tests, so fair comparisons can be done. However, the ASL-1.1 code incorporates a special treatment of the linear kernel which we were not able to deactivate and it seems to give advantage to ASL. We decide to compare with ASL for two reasons. On one hand, LC and ASL share the interior point nature and the use of an approximation of the Hessian based on the spectral length. On the other hand, in [17], ASL is compared with two state of the art algorithms: LIBSVM [7] and GPDT [31,32,37]. Numerical results showed that ASL is competitive with these algorithms, so to be able to compete with ASL speaks in favor of the performance of the algorithms here proposed.

The support vectors found by LC are mostly the same than when using ASL, as well as the optimal objective values. Therefore, the generalization errors are roughly the same. In terms of efficiency, for most tests, ASL requires less time to solve the problems than LC. However, the running times are comparable when solving problems with much more attributes than dimension. Even when the running time for LC is 100 % more than ASL in some problems tested, in absolute time most of them are solved very fast, which represents, for practical purposes, low cost. As an illustration see Tables 3 and 4. There, we compare the CPU times, the optimal objective values and the number of support vectors for the three kernels and values of $C = 1, 5, 10$. Observe the problems "Arcene.scale, Dorothea, Dexter" where LC has a better performance than ASL. We believe that these results may be related with the fact that, for these problems, the Hessians are well conditioned (see Table 1) and the LC algorithm is more affected by the ill-conditioning of the Hessian than ASL. Also observe that LC converges for some instances where ASL does not, as in problems "Farm-ads, News20, Rev1".

In order to improve the efficiency of the Low-Cost algorithm, we modify it by including the identification procedure and as a result we get the Low-Cost Identification algorithm (Algorithm 4.1, named LC-I). Tables 5,6, 7 and 8 show the results obtained when algorithms LC and LC-I are applied to the problems in Table 2. The running time for LC-I includes the time needed

| Problem | Linear | Polynomial | Gaussian |
|---|---|---|---|
| Arcene.scale | 2370 | 505.64 | 1479 |
| Dorothea | $1.59e^6$ | $1.42e^3$ | $5,91e^4$ |
| Dexter | $6.20e^4$ | $2e^4$ | $1.12e^5$ |

Table 1. Condition numbers of the Hessians for problems Arcene.scale, Dorothea, and Dexter

to form the new problem and the identification process, which was coded in Matlab. Improvements of the LC-I algorithm will consider this implementation in C++. It can be seen that there is a significant decrease of the CPU time of the LC algorithm when the identification procedure is used. It is clear that there is a difference in the number of support vectors found by the LC and LC-I algorithms but the generalization errors, in most problems, stay very close. Moreover, for some problems the generalization error is better for LC-I (see for example problems "Heart-scale" and "Live.disorders-scale", for Polynomial kernel and $C = 10$, as well as "Svmguide3" for Linear kernel and $C = 10$ ), which speaks in favor of the identification procedure. These results support that for SVM problems there does not seem to be needed to work with all the data but to use vectors that give information on the distribution of the whole training data set. We would like to cite a recent paper by Camelo et al [6] where it is shown that considering random samples and notions of closeness with the support vectors, the SVM problems can be solved without degrading too much the generalization error; and the paper by Xu et al [36] that studies the robustness of these problems.

For a convenient presentation of the comparison among the LC, ASL and LC-I algorithms we use the following definitions (following [10]):

The Efficiency Index ($E_A$) is defined as $E_A = \frac{\sum_{i=1}^{k} e_i}{a_A}$ with

$$e_i = \begin{cases} 0; & \text{if the algorithm A fails at problem i} \\ 1; & \text{if } t_{ib} = 0 \text{ and } t_{iA} = 0 \\ t_{ib}/t_{iA}; & \text{if } t_{iA} \neq 0 \end{cases}$$

where $t_{iA}$ is the running time for algorithm $A$ for solving problem $i$, $t_{ib} = \min_A\{t_{iA}\}$, this is, the best result among all the algorithms for problem $i$, $a_A$ is the number of problems solved by the algorithm $A$, and $k$ is the total of problems to solve.

The Robustness Index ($R_A$) represents the percentage of cases where the algorithm finds a solution and it is defined as $R_A = a_A/r$ where $r$ is the greater number of solved problems by any of the tested algorithms.

The Combination of Robustness and Efficiency ($E_{RA}$): $E_{RA} = E_A \times R_A$.

These profiles are included in the Figures 1, 2, 3. For these tests, the problems considered are the ones included in Table 2 where the number of training data is larger than the attributes. Each problem is solved for the three kernels and the values of $C = 1, 5, 10$.

Overall, the graphs show that the LC and LC-I algorithm are competitive with ASL specially for larger values of $C$.

## 6 Concluding Remarks

In this paper we propose a novel approach to solve the quadratic convex continuous knapsack problems. It only involves matrix-vector multiplications so it is appropriate for the case when the Hessian of the objective function of the optimization problem is large and dense, as the ones arising in the context of Support Vector Machines. The method is simple and it solves problems with singular or ill-conditioned Hessian, even though experiments showed that it may be sensitive to this bad conditioning. An identification procedure was added to the algorithm exploiting its primal-dual nature which contributes to greatly reduce the running time with almost no reduction on the generalization error. Extensive numerical experimentation was performed showing the effectiveness of the algorithm. We compare with the ASL algorithm by using the code available from its authors, showing that the Low-Cost Interior Point algorithm (with or without identification) is competitive specially for nonlinear kernels. Future research will attempt to improve the performance of the algorithm for ill-conditioning Hessians.

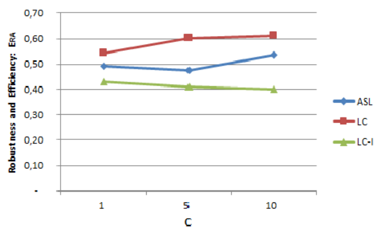

Figure 1. Linear kernel & $E_{RA}$

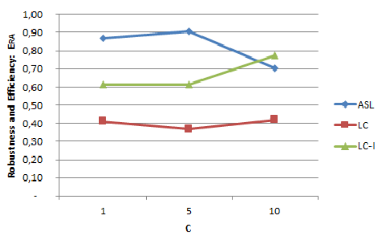

Figure 2. Polynomial kernel & $E_{RA}$

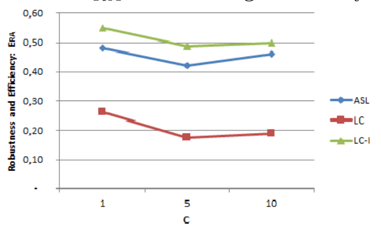

Figure 3. Gaussian kernel & $E_{RA}$

| Problem | # Training data | # Attributes |
|---|---|---|
| a1a | 1605 | 123 |
| a2a | 2265 | 123 |
| a3a | 3185 | 123 |
| a4a | 4781 | 123 |
| australian-scale | 552 | 14 |
| breast.cancer-scale | 547 | 10 |
| diabetes-scale | 615 | 8 |
| fourclass-scale | 690 | 2 |
| german.numer-scale | 800 | 24 |
| heart-scale | 216 | 13 |
| ionosphere-scale | 281 | 34 |
| liver.disorders-scale | 276 | 6 |
| splice-scale | 1000 | 60 |
| sonar-scale | 167 | 60 |
| svmguide1-scale | 3809 | 4 |
| svmguide3 | 1243 | 21 |
| w1a | 2477 | 300 |
| w3a | 3470 | 300 |
| w4a | 4912 | 300 |
| gisette-scale | 3470 | 300 |
| mushrooms | 4912 | 300 |

Table 2. Problems tested with much less attributes than number of training data

## References

[1] J. Barzilai and J. M. Borwein. Two point step size gradient methods. *IMA*, 8:141–148, 1988.

[2] S. Bellavia. Inexact interior-point method. *Journal of Optimization Theory and Applications*, 96(1):109–121, 1998.

[3] V. Blanz, B. Schölkopf, H. Bülthoffand, C. Burges, V. N. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In J.C. Vorbrüggen, C. von der Malsburg, W. von Seelen and B. Sendhoff, editors, *Artificial Neural Networks, ICIANN'96, pages 251-256, Berlin 1996. Springer Lecture Notes in Computer Science, Vol. 1112.* 1996.

[4] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. In U. Fayyad, editor, *Data Mining on Knowledge Discovery*, pages 121–167. Kluwer Academic Publishers, Netherlands, 1998.

[5] C.J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In M. Jordan M. Mozer and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381. MIT Press, Cambridge, MA, 1997.

[6] S.A. Camelo, M.D. Gonzalez-Lima, and A.J. Quiroz. Nearest neighbors methods for support vector machines. *Annals of Operations Research*, 235:85–101, 2015.

[7] C. Chang and C. Lin. Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27, 2011.

[8] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[9] N. Cristianini and J.Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, United Kingdom, 2000.

[10] J. Dominguez and M. Gonzalez-Lima. A primaldual interior-point algorithm for quadratic programming. *Numer. Algor.*, 42:1–30, 2006.

[11] S. Eisenstat and W. Walker. Globally convergent inexact newton methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.

[12] A. S. El-Bakry, R. A. Tapia, and Y. Zhang. A study of indicators for identifying zero variables in interior point methods. *SIAM Review*, 36(1):45–72, 1994.

[13] F. Facchinei, A. Fischer, and C. Kanzow. On the identification of zero variables in an interior-point framework. *SIAM Journal on Optimization*, 10:1058–1078, 2000.

[14] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[15] M. C. Ferris and T.S. Munson. Interior point methods for massive support vector machines. *SIAM J. Optim.*, 13(3):783–804, 2002.

[16] E. M. Gertz and J. D. Griffin. Using an iterative linear solver in an interior-point method for generating support vector machines. *COAP*, 47:431–450, 2010.

[17] M. D. Gonzalez-Lima, W. W. Hager, and H. Zhang. An affine-scaling interior-point method for continuous knapsack constraints with application to support vector machines. *SIAM Journal on Optimization*, 21(1):361–390, 2011.

[18] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pages 137–142. Springer, April 1998.

[19] J. H. Jung, D. P. O Leary, and A. L. Tits. Adaptive constraint reduction for convex quadratic programming. *COAP*, 51:125–157, 2012.

[20] J.H. Jung, D.P. O Leary, and A.L. Tits. Adaptive constraint reduction for training support vector machines. *Electron. Trans. Numer. Anal.*, 31:156–177, 2008.

[21] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[22] Chih-Jen Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, pages 1288–1298, 2001.

[23] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2(4):575601, 1992.

[24] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[25] E.E. Osuna, R. Freund, and F. Girosi. Support Vector Machines: Training and Applications. Technical Report A.I. Memo No. 1602, C.B.C.L. Paper No. 144, Massachusetts Institute of Technology, Cambridge, MA, USA, 1997.

[26] E.E. Osuna, R. Freund, and F. Girosi. Training support vector vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136. 1997.

[27] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods. Support Vector Learning*, pages 41–65, Cambridge, MA, 1998. MIT Press.

[28] M. Schmidt. Identifying speaker with support vector networks. In *Interface'96 Proceedings.* Sydney, Australia, 1996.

[29] B. Scholkopf, C. Burges, and V. N. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings First International Conference on Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1995.

[30] B. Scholkopf, C. Burges, and V. N. Vapnik. Incorporating invariances in support vector learning machines. In J.C. Vorbrüuggen C. von der Malsburg, W. von Seelen and B. Sendhoff, editors, *Artificial Neural Networks, ICIANN'96, pages 47-52, Berlin 1996. Springer Lecture Notes in Computer Science, Vol. 1112.* 1996.

[31] T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20:353–378, 2003.

[32] T. Serafini and L. Zanni. On the working set selection in gradient-based descomposition techniques for support vector machines. *Optimization Methods and Software*, 20:586–593, 2005.

[33] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[34] J. Weston, A. Gammerman, M.O. Stitson, V. Vapnik, V. Vork, and C. Watkins. Density estimation using support vector machines. Technical Report CSD-TR-97-23, Royal Holloway College, 1997.

[35] K. Woodsend and J. Gondzio. Exploiting separability in large scale linear support vector machine training. *Computational Optimization and Applications*, 49:241–269, 2011.

[36] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009.

[37] L. Zanni. An improved gradient projection-based decomposition techniques for support vector machines. *Computational Management Science*, 3:131–145, 2006.

| **Problem # training/ # attributes** | **Kernel** | **C** | **Time (seconds)** | | **Obj. Func.** | | **# SV** | |
|---|---|---|---|---|---|---|---|---|
| **adwords1** | | | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.004 | 0.01 | -2.098 | -2.098 | 50 | 50 |
| | | 5 | 0.01 | 0.01 | -10.1 | -10.1 | 50 | 50 |
| | | 10 | 0.01 | 0.01 | -20.1 | -20.1 | 50 | 50 |
| 52/4702 | Polynomial | 1 | 0.008 | 0.01 | -45.99 | -45.28 | 52 | 52 |
| | | 5 | 0.008 | 0.01 | -229.9 | -229.8 | 52 | 52 |
| | | 10 | 0.004 | 0.01 | -459.7 | -459.7 | 52 | 52 |
| | Gaussian | 1 | 0.008 | 0.01 | -43.01 | -43.01 | 52 | 51 |
| | | 5 | 0.008 | 0.01 | -155.5 | -155.5 | 52 | 51 |
| | | 10 | 0.008 | 0.01 | -213.3 | -213.3 | 51 | 51 |
| **adwords2** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.004 | 0.01 | -2.097 | -2.097 | 49 | 49 |
| | | 5 | 0.008 | 0.01 | -10.1 | -10.1 | 49 | 49 |
| | | 10 | 0.01 | 0.01 | -20.1 | -20.1 | 49 | 49 |
| 52/3721 | Polynomial | 1 | 0.004 | 0.01 | -45.99 | -45.81 | 52 | 52 |
| | | 5 | 0.008 | 0.01 | -229.9 | -229.9 | 52 | 52 |
| | | 10 | 0.004 | 0.01 | -459.5 | -459.5 | 52 | 52 |
| | Gaussian | 1 | 0.008 | 0.01 | -42.06 | -42.06 | 52 | 51 |
| | | 5 | 0.008 | 0.01 | -136.3 | -136.3 | 52 | 51 |
| | | 10 | 0.008 | 0.01 | -180.4 | -180.4 | 51 | 51 |
| **adwords3** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.001 | 0.01 | -4.17 | -4.17 | 47 | 47 |
| | | 5 | 0.001 | 0.01 | -47.46 | -4.746 | 47 | 47 |
| | | 10 | 0.001 | 0.01 | -4.746 | -4.746 | 47 | 47 |
| 52/242 | Polynomial | 1 | 0.001 | 0.01 | -45.99 | -45.28 | 52 | 52 |
| | | 5 | 0.001 | 0.01 | -230 | -229.9 | 52 | 52 |
| | | 10 | 0.001 | 0.01 | -459.9 | -459.8 | 52 | 52 |
| | Gaussian | 1 | 0.001 | 0.01 | -43.83 | -43.83 | 51 | 51 |
| | | 5 | 0.001 | 0.01 | -175.7 | -175.7 | 51 | 51 |
| | | 10 | 0.001 | 0.01 | -257.2 | -257.2 | 51 | 51 |
| **adwords4** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.001 | 0.01 | -4.248 | -4.248 | 47 | 47 |
| | | 5 | 0.001 | 0.01 | -4.831 | -4.831 | 48 | 48 |
| | | 10 | 0.001 | 0.01 | -4.831 | -4.831 | 48 | 48 |
| 52/229 | Polynomial | 1 | 0.001 | 0.01 | -45.99 | -45.28 | 42 | 52 |
| | | 5 | 0.001 | 0.01 | -230.0 | -229.9 | 52 | 52 |
| | | 10 | 0.001 | 0.01 | -459.9 | -459.8 | 52 | 52 |
| | Gaussian | 1 | 0.001 | 0.01 | -43.65 | -43.63 | 52 | 49 |
| | | 5 | 0.001 | 0.01 | -171.3 | -171.3 | 52 | 50 |
| | | 10 | 0.001 | 0.01 | -246.1 | -246.1 | 49 | 49 |
| **arcene.scale** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.22 | 0.20 | -0.033 | -0.034 | 75 | 76 |
| | | 5 | 0.23 | 0.19 | -0.033 | -0.034 | 75 | 76 |
| | | 10 | 0.21 | 0.19 | -0.033 | -0.034 | 75 | 76 |
| 80/10000 | Polynomial | 1 | 0.18 | 0.19 | -61.48 | -63.25 | 76 | 77 |
| | | 5 | 0.18 | 0.19 | -204.5 | -210.8 | 77 | 79 |
| | | 10 | 0.18 | 0.19 | -266.3 | -273.4 | 78 | 79 |
| | Gaussian | 1 | 0.20 | 0.19 | -55.71 | -58.20 | 80 | 75 |
| | | 5 | 0.20 | 0.19 | -105.6 | -161.8 | 80 | 78 |
| | | 10 | 0.20 | 0.18 | -105.8 | -181.7 | 80 | 77 |
| **duke** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.02 | 0.02 | -0.003 | -0.003 | 27 | 27 |
| | | 5 | 0.02 | 0.02 | -0.003 | -0.003 | 27 | 27 |
| | | 10 | 0.02 | 0.02 | -0.003 | -0.003 | 27 | 27 |
| 29/7129 | Polynomial | 1 | 0.01 | 0.01 | -20.91 | -20.91 | 29 | 29 |
| | | 5 | 0.02 | 0.02 | -69.51 | -69.51 | 29 | 29 |
| | | 10 | 0.02 | 0.02 | -95.41 | -95.41 | 27 | 29 |
| | Gaussian | 1 | 0.02 | 0.01 | -16.59 | -16.59 | 29 | 29 |
| | | 5 | 0.02 | 0.01 | -19.10 | -19.10 | 29 | 29 |
| | | 10 | 0.02 | 0.01 | -19.10 | -19.10 | 29 | 29 |
| **leukemia** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.02 | 0.02 | -0.002 | -0.002 | 29 | 27 |
| | | 5 | 0.02 | 0.03 | -0.002 | -0.002 | 29 | 27 |
| | | 10 | 0.02 | 0.03 | -0.002 | -0.002 | 29 | 27 |
| 38/7129 | Polynomial | 1 | 0.02 | 0.03 | -14.58 | -14.58 | 35 | 35 |
| | | 5 | 0.02 | 0.03 | -29.59 | -29.59 | 36 | 36 |
| | | 10 | 0.02 | 0.03 | -30.76 | -30.76 | 37 | 37 |
| | Gaussian | 1 | 0.03 | 0.03 | -12.53 | -12.53 | 38 | 38 |
| | | 5 | 0.02 | 0.02 | -13.83 | -13.83 | 38 | 38 |
| | | 10 | 0.03 | 0.03 | -13.83 | -13.83 | 38 | 38 |

Table 3. ASL vs. LC for problems with high number of attributes.

| **Problem # training/ # attributes** | **Kernel** | **C** | **Time (seconds)** | | **Obj. Func.** | | **# SV** | |
|---|---|---|---|---|---|---|---|---|
| **colon-cancer** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.008 | 0.01 | -0.010 | -0.010 | 26 | 26 |
| | | 5 | 0.01 | 0.01 | -0.010 | -0.010 | 26 | 26 |
| | | 10 | 0.01 | 0.01 | -0.010 | -0.010 | 26 | 26 |
| 40/2000 | Polynomial | 1 | 0.008 | 0.01 | -15.55 | -15.55 | 39 | 39 |
| | | 5 | 0.008 | 0.01 | -20.25 | -20.25 | 38 | 38 |
| | | 10 | 0.01 | 0.01 | -20.25 | -20.25 | 38 | 38 |
| | Gaussian | 1 | 0.01 | 0.01 | -15.39 | -15.39 | 39 | 38 |
| | | 5 | 0.008 | 0.02 | -17.31 | -17.31 | 38 | 38 |
| | | 10 | 0.01 | 0.01 | -17.31 | -17.31 | 38 | 38 |
| **farm-ads** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | - | 74.65 | - | -28.76 | - | 1383 |
| | | 5 | - | 227.4 | - | -52.76 | - | 1381 |
| | | 10 | - | 299.0 | - | -82.76 | - | 1380 |
| 3315/54877 | Polynomial | 1 | 16.89 | 19.88 | -3092 | -3080 | 3315 | 3315 |
| | | 5 | 16.90 | 20.82 | -15460 | -15457 | 3315 | 3315 |
| | | 10 | 16.93 | 20.83 | -30920 | -30917 | 3315 | 3315 |
| | Gaussian | 1 | 18.54 | 22.41 | -2899 | -2898 | 3117 | 3117 |
| | | 5 | 18.70 | 25.22 | -11413 | -11413 | 2824 | 2825 |
| | | 10 | 18.83 | 27.66 | -19601 | -19599 | 2520 | 2521 |
| **news20** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | - | 1014 | - | -2237 | - | 7610 |
| | | 5 | - | 1318 | - | -2670 | - | 7486 |
| | | 10 | - | 1392 | - | -2911 | - | 7357 |
| 15997/1355191 | Polynomial | 1 | 561.2 | 1059 | -15991 | -15613 | 15997 | 15997 |
| | | 5 | 526.2 | 902.9 | -79970 | -79985 | 15997 | 15997 |
| | | 10 | 565.7 | 880.1 | -159940 | -159970 | 15997 | 15997 |
| | Gaussian | 1 | 798.5 | 865.7 | -15996 | -15997 | 15997 | 15997 |
| | | 5 | 823.7 | 885.6 | -80027 | -79977 | 15996 | 15997 |
| | | 10 | 844.1 | 1047 | -160267 | -159940 | 15996 | 15997 |
| **rcv1** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | - | 647.8 | - | -1746 | - | 4819 |
| | | 5 | - | 896.7 | - | -2761 | - | 4257 |
| | | 10 | - | 944.4 | - | -3159 | - | 4165 |
| 20242/47236 | Polynomial | 1 | 279.2 | 383.9 | -19500 | -19466 | 20242 | 20242 |
| | | 5 | 265.20 | 396.88 | -97510 | -97494 | 20242 | 20242 |
| | | 10 | 267.8 | 397.6 | -195020 | -195012 | 20242 | 20242 |
| | Gaussian | 1 | 336.6 | 439.2 | -19491 | -19474 | 20150 | 19510 |
| | | 5 | 339.8 | 451.5 | -97243 | -96867 | 19620 | 19506 |
| | | 10 | 376.1 | 440.2 | -193950 | -192463 | 19582 | 19508 |
| **dorothea** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.74 | 1.93 | -3.959 | -3.959 | 225 | 225 |
| | | 5 | 0.70 | 2.10 | -3.959 | -3.959 | 225 | 225 |
| | | 10 | 0.70 | 2.09 | -3.959 | -3.959 | 225 | 225 |
| 641/937 | Polynomial | 1 | 1.15 | 1.21 | -121.7 | -121.7 | 410 | 399 |
| | | 5 | 1.10 | 1.23 | -549.5 | -549.5 | 405 | 404 |
| | | 10 | 1.18 | 1.31 | -998.3 | -998.3 | 405 | 403 |
| | Gaussian | 1 | 1.14 | 1.30 | -114.9 | -114.9 | 282 | 281 |
| | | 5 | 1.35 | 1.34 | -458.6 | -458.7 | 294 | 299 |
| | | 10 | 1.36 | 1.38 | -682.4 | -682.8 | 301 | 302 |
| **dexter** | **Kernel** | **C** | **ASL** | **LC** | **ASL** | **LC** | **ASL** | **LC** |
| | Linear | 1 | 0.1 | 1.22 | -0.568 | -0.568 | 237 | 237 |
| | | 5 | 0.1 | 1.23 | -0.568 | -0.568 | 237 | 237 |
| | | 10 | 0.1 | 1.14 | -0.568 | -0.568 | 237 | 237 |
| 240/19999 | Polynomial | 1 | 1.3 | 1.14 | -239.4 | -239.4 | 240 | 240 |
| | | 5 | 1.33 | 1.13 | -1186 | -1186 | 240 | 240 |
| | | 10 | 1.3 | 1.13 | -2343 | -2343 | 240 | 240 |
| | Gaussian | 1 | 1.52 | 1.14 | -237.1 | -236.9 | 240 | 240 |
| | | 5 | 1.35 | 1.14 | -1127 | -1122 | 240 | 240 |
| | | 10 | 1.4 | 1.14 | -2109 | -2089 | 240 | 240 |

Table 4. Cont. ASL vs. LC for problems with high number of attributes.

| **Problem** | | | **Time (seconds)** | | **# SV** | | **Generalization error** | |
|---|---|---|---|---|---|---|---|---|
| **a1a** | **Kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 5.1 | 0.57 | 591 | 186 | 83.70% | 83.91% |
| | | 5 | 9.48 | 2.56 | 577 | 377 | 83.87% | 83.29% |
| | | 10 | 19.24 | 3.58 | 571 | 338 | 83.76% | 83.00% |
| | Polynomial | 1 | | 0.32 | | 300 | | 75.94 % |
| | | 5 | 0.55 | 0.5 | 880 | 660 | 75.94% | 75.94% |
| | | 10 | 0.56 | 0.5 | 830 | 742 | 75.94% | 75.94% |
| | Gaussian | 1 | 0.91 | 0.27 | 745 | 287 | 83.58% | 83.55% |
| | | 5 | 1.45 | 0.43 | 660 | 173 | 84.32% | 84.02% |
| | | 10 | 1.73 | 0.46 | 645 | 189 | 84.42% | 83.70% |
| **a2a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 12.45 | 1.11 | 882 | 263 | 84.28% | 84.44% |
| | | 5 | 25.89 | 8.11 | 887 | 541 | 84.02% | 83.69% |
| | | 10 | 50.31 | 13.32 | 879 | 578 | 84.02% | 83.28% |
| | Polynomial | 1 | | 0.41 | | 406 | | 76 % |
| | | 5 | 1.09 | 0.67 | 1189 | 963 | 76.00% | 76.00% |
| | | 10 | 1.18 | 0.75 | 1190 | 858 | 76.00% | 76.00% |
| | Gaussian | 1 | 1.88 | 0.48 | 1068 | 297 | 83.93% | 83.78% |
| | | 5 | 2.9 | 0.83 | 965 | 183 | 84.53% | 84.32% |
| | | 10 | 4.82 | 0.91 | 937 | 182 | 84.64% | 84.34% |
| **a3a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 20 | 3.54 | 1184 | 518 | 84.33% | 84.33% |
| | | 5 | 86.66 | 16.54 | 1154 | 707 | 84.09% | 83.81% |
| | | 10 | 157.53 | 36.21 | 1155 | 760 | 84.06% | 83.79% |
| | Polynomial | 1 | | 0.65 | | 605 | | 75.93 % |
| | | 5 | 2.29 | 0.97 | 1593 | 1422 | 75.93% | 75.93% |
| | | 10 | 1.93 | 1.39 | 3158 | 1348 | 75.93% | 75.93% |
| | Gaussian | 1 | 3.23 | 1.06 | 1510 | 377 | 83.84% | 83.53% |
| | | 5 | 6.8 | 1.62 | 1252 | 210 | 84.48% | 84.22% |
| | | 10 | 9.31 | 3.86 | 1229 | 1031 | 84.46% | 84.43% |
| **a4a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 69.79 | 7.99 | 1756 | 665 | 84.27% | 84.56% |
| | | 5 | 228.93 | 58.09 | 1736 | 1199 | 84.45% | 84.42% |
| | | 10 | 382.54 | 73.81 | 1740 | 1151 | 80.03% | 84.26% |
| | Polynomial | 1 | | 1.21 | | 949 | | 76.05 % |
| | | 5 | 3.97 | 2.25 | 2473 | 1621 | 76.05% | 76.05% |
| | | 10 | 5.35 | 3.26 | 2585 | 2087 | 76.18% | 80.44% |
| | Gaussian | 1 | 9.27 | 2.22 | 2006 | 387 | 83.98% | 83.44% |
| | | 5 | 18.26 | 4.46 | 2473 | 1042 | 84.39% | 84.29% |
| | | 10 | 30.75 | 8.80 | 2585 | 1413 | 84.52% | 84.36% |
| **australian_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 1.53 | 0.07 | 178 | 16 | 84.78% | 84.78% |
| | | 5 | 5.27 | 1.03 | 180 | 174 | 84.78% | 84.78% |
| | | 10 | 7.36 | 0.85 | 189 | 208 | 84.78% | 84.78% |
| | Polynomial | 1 | 0.11 | 0.06 | 245 | 86 | 84.78% | 84.78% |
| | | 5 | 0.24 | 0.06 | 210 | 53 | 89.13% | 86.23% |
| | | 10 | 0.28 | 0.07 | 204 | 73 | 87.68% | 86.23% |
| | Gaussian | 1 | 0.25 | 0.07 | 200 | 48 | 84.78% | 84.78% |
| | | 5 | 0.39 | 0.10 | 193 | 77 | 89.13% | 84.05% |
| | | 10 | 0.59 | 0.22 | 194 | 144 | 87.68% | 86.23% |
| **breast-cancer_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.07 | 0.05 | 47 | 22 | 97.79% | 97.79% |
| | | 5 | 0.15 | 0.08 | 43 | 28 | 97.79% | 97.79% |
| | | 10 | 0.17 | 0.08 | 44 | 16 | 97.79% | 97.79% |
| | Polynomial | 1 | 0.05 | 0.04 | 80 | 43 | 98.52% | 98.52% |
| | | 5 | 0.06 | 0.05 | 55 | 47 | 98.52% | 98.52% |
| | | 10 | 0.07 | 0.04 | 58 | 27 | 98.52% | 98.52% |
| | Gaussian | 1 | 0.06 | 0.04 | 63 | 32 | 98.52% | 98.52% |
| | | 5 | 0.1 | 0.06 | 55 | 33 | 98.52% | 98.52% |
| | | 10 | 0.12 | 0.08 | 61 | 44 | 97.79% | 97.79% |

Table 5. LC vs. LC-I

| **Problem** | | | **Time (seconds)** | | **# SV** | | **Generalization error** | |
|---|---|---|---|---|---|---|---|---|
| **diabetes_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.2 | 0.08 | 337 | 34 | 76.47% | 76.47% |
| | | 5 | 0.54 | 0.19 | 326 | 150 | 76.47% | 77.77% |
| | | 10 | 0.9 | 0.29 | 325 | 16 | 75.81% | 75.81% |
| | Polynomial | 1 | 0.03 | 0.05 | 434 | 166 | 67.32% | 69.28% |
| | | 5 | 0.08 | 0.06 | 396 | 162 | 77.12% | 72.54% |
| | | 10 | 0.13 | 0.07 | 376 | 125 | 76.47% | 73.85% |
| | Gaussian | 1 | 0.1 | 0.06 | 369 | 103 | 76.47% | 74.50% |
| | | 5 | 0.23 | 0.1 | 328 | 133 | 75.81% | 73.20% |
| | | 10 | 0.38 | 0.18 | 326 | 212 | 76.47% | 75.16% |
| **fourclass_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.18 | 0.07 | 392 | 49 | 84.88% | 83.72% |
| | | 5 | 0.6 | 0.21 | 389 | 51 | 84.88% | 83.13% |
| | | 10 | 1.04 | 0.31 | 386 | 150 | 84.88% | 84.88% |
| | Polynomial | 1 | 0.17 | 0.15 | 407 | 119 | 80.23% | 80.23% |
| | | 5 | 0.47 | 0.39 | 356 | 279 | 83.13% | 85.46% |
| | | 10 | 0.5 | 0.33 | 348 | 214 | 85.46% | 83.72% |
| | Gaussian | 1 | 0.28 | 0.07 | 353 | 54 | 86.62% | 86.04% |
| | | 5 | 0.58 | 0.18 | 332 | 175 | 87.20% | 83.13% |
| | | 10 | 0.87 | 0.44 | 320 | 42 | 90.11% | 87.20% |
| **german.number_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 1.4 | 0.29 | 429 | 30 | 78.50% | 79% |
| | | 5 | 3.85 | 1.73 | 428 | 272 | 77.5% | 76% |
| | | 10 | 6.05 | 3.05 | 428 | 299 | 78% | 76.5% |
| | Polynomial | 1 | 0.13 | 0.08 | 506 | 188 | 73% | 73% |
| | | 5 | 0.27 | 0.15 | 475 | 212 | 77.50% | 78% |
| | | 10 | 0.57 | 0.27 | 462 | 263 | 78% | 77.5% |
| | Gaussian | 1 | 0.28 | 0.1 | 482 | 132 | 75% | 75% |
| | | 5 | 0.45 | 0.19 | 461 | 222 | 79.50% | 77.5% |
| | | 10 | 0.84 | 0.39 | 452 | 295 | 77.50% | 77% |
| **heart_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.04 | 0.02 | 85 | 20 | 85.18% | 83.33% |
| | | 5 | 0.12 | 0.05 | 82 | 53 | 85.18% | 83.33% |
| | | 10 | 0.16 | 0.11 | 82 | 70 | 85.18% | 85.18% |
| | Polynomial | 1 | 0.01 | 0.009 | 155 | 50 | 83.33% | 83.33% |
| | | 5 | 0.02 | 0.02 | 117 | 57 | 83.33% | 83.33% |
| | | 10 | 0.02 | 0.02 | 110 | 60 | 81.48% | 85.18% |
| | Gaussian | 1 | 0.01 | 0.01 | 116 | 38 | 83.33% | 83.33% |
| | | 5 | 0.02 | 0.01 | 99 | 44 | 83.33% | 83.33% |
| | | 10 | 0.04 | 0.02 | 100 | 69 | 85.18% | 85.18% |
| **ionosphere_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.1 | 0.05 | 78 | 52 | 85.71% | 84.28% |
| | | 5 | 0.17 | 0.10 | 60 | 45 | 85.71% | 87.14% |
| | | 10 | 0.21 | 0.11 | 57 | 37 | 85.71% | 82.85% |
| | Polynomial | 1 | 0.02 | 0.09 | 205 | 91 | 64.28% | 64.28% |
| | | 5 | 0.03 | 0.02 | 189 | 85 | 84.28% | 84.28% |
| | | 10 | 0.04 | 0.03 | 168 | 81 | 85.71% | 81.42% |
| | Gaussian | 1 | 0.04 | 0.02 | 116 | 72 | 92.85% | 90% |
| | | 5 | 0.07 | 0.05 | 83 | 61 | 98.57% | 94.28% |
| | | 10 | 0.07 | 0.05 | 74 | 60 | 98.57% | 97.14% |
| **liver.disorders_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.02 | 0.02 | 230 | 61 | 63.76% | 56.52% |
| | | 5 | 0.07 | 0.04 | 215 | 85 | 66.66% | 66.66% |
| | | 10 | 0.1 | 0.08 | 210 | 26 | 66.66% | 65.21% |
| | Polynomial | 1 | 0.01 | 0.01 | 237 | 127 | 57.97% | 43.47% |
| | | 5 | 0.01 | 0.02 | 233 | 77 | 59.42% | 65.21% |
| | | 10 | 0.02 | 0.03 | 233 | 112 | 63.76% | 66..66% |
| | Gaussian | 1 | 0.02 | 0.02 | 237 | 75 | 63.76% | 59.42% |
| | | 5 | 0.04 | 0.03 | 222 | 66 | 68.11% | 68.11% |
| | | 10 | 0.06 | 0.04 | 217 | 84 | 69.56% | 66.66% |

Table 6. Cont.- LC vs. LC-I

| **Problem** | | | **Time (seconds)** | | **# SV** | | **Generalization error** | |
|---|---|---|---|---|---|---|---|---|
| **splice_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 2.02 | 0.48 | 413 | 80 | 84.78% | 84.22% |
| | | 5 | 6.79 | 0.91 | 412 | 152 | 84.68% | 83.77% |
| | | 10 | 8.96 | 1.37 | 417 | 177 | 83.77% | 84.09% |
| | Polynomial | 1 | 0.2 | 0.06 | 996 | 278 | 53.65% | 57.01% |
| | | 5 | 0.33 | 0.18 | 955 | 587 | 86.06% | 76.55% |
| | | 10 | 0.27 | 0.34 | 928 | 809 | 86.75% | 84.32% |
| | Gaussian | 1 | 0.5 | 0.18 | 614 | 302 | 88.96% | 86.29% |
| | | 5 | 0.67 | 0.42 | 576 | 492 | 90.11% | 89.83% |
| | | 10 | 0.71 | 0.46 | 595 | 582 | 89.56% | 89.70% |
| **sonar_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 0.03 | 0.03 | 76 | 51 | 78.04% | 73.17% |
| | | 5 | 0.08 | 0.05 | 66 | 52 | 75.60% | 80.48% |
| | | 10 | 0.08 | 0.08 | 64 | 49 | 75.60% | 78.04% |
| | Polynomial | 1 | 0.01 | 0.007 | 157 | 44 | 58.53% | 60.97% |
| | | 5 | 0.01 | 0.009 | 138 | 47 | 65.85% | 65.85% |
| | | 10 | 0.01 | 0.01 | 124 | 48 | 70.73% | 63.41% |
| | Gaussian | 1 | 0.01 | 0.008 | 128 | 34 | 82.92% | 65.85% |
| | | 5 | 0.02 | 0.02 | 102 | 44 | 78.04% | 75.60% |
| | | 10 | 0.03 | 0.02 | 93 | 81 | 75.60% | 75.60% |
| **svmguide1.scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 3.74 | 0.92 | 557 | 358 | 48.87% | 48.77% |
| | | 5 | 7.48 | 1.46 | 444 | 278 | 49.05% | 49.17% |
| | | 10 | 12.81 | 1.57 | 410 | 247 | 49.02% | 49.27% |
| | Polynomial | 1 | 1.71 | 0.60 | 768 | 414 | 49.12% | 49.12% |
| | | 5 | 3.19 | 0.74 | 504 | 391 | 49.57% | 49.6% |
| | | 10 | 3.17 | 1.06 | 437 | 347 | 49.6% | 49.67% |
| | Gaussian | 1 | 3.67 | 0.86 | 638 | 424 | 48.85% | 48.77% |
| | | 5 | 6.47 | 1.01 | 435 | 319 | 49.3% | 49.37% |
| | | 10 | 10.46 | 1.44 | 386 | 308 | 49.32% | 49.47% |
| **svmguide3** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 2.97 | 0.29 | 548 | 307 | 17.07% | 51.21% |
| | | 5 | 7.77 | 1.07 | 534 | 58 | 39.02% | 46.34% |
| | | 10 | 12.26 | 7.55 | 526 | 422 | 41.46% | 48.78% |
| | Polynomial | 1 | 0.56 | 0.44 | 594 | 249 | 0% | 0% |
| | | 5 | 0.95 | 0.60 | 582 | 498 | 0% | 0% |
| | | 10 | 1.32 | 0.91 | 576 | 518 | 2.43% | 2.43% |
| | Gaussian | 1 | 1.12 | 0.19 | 573 | 378 | 2.43% | 9.75% |
| | | 5 | 2.17 | 1.01 | 558 | 544 | 12.19% | 17.07% |
| | | 10 | 2.63 | 2.28 | 537 | 528 | 19.51% | 29.26% |
| **w1a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 4.54 | 1.02 | 172 | 170 | 97.74% | 97.71% |
| | | 5 | 11.68 | 2.07 | 170 | 375 | 97.51% | 97.35% |
| | | 10 | 16.77 | 2.33 | 386 | 387 | 97.46% | 97.24% |
| | Polynomial | 1 | 1.02 | 1.64 | 2467 | 2086 | 97.02% | 97.02% |
| | | 5 | 8.74 | 10.50 | 1277 | 1278 | 97.02% | 97.02% |
| | | 10 | 12.16 | 0.15 | 1174 | 72 | 97.02% | 96.86% |
| | Gaussian | 1 | 4.11 | 4.98 | 240 | 240 | 97.02% | 97.02% |
| | | 5 | 4.58 | 0.80 | 236 | 225 | 97.25% | 97.25% |
| | | 10 | 4.42 | 0.55 | 217 | 188 | 97.37% | 97.35% |
| **w3a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 9.43 | 2.13 | 229 | 217 | 98.07% | 98.08% |
| | | 5 | 15.52 | 3.42 | 207 | 200 | 97.80% | 97.71% |
| | | 10 | 29.57 | 5.49 | 201 | 198 | 97.53% | 97.39% |
| | Polynomial | 1 | 2.15 | 2.81 | 3058 | 2994 | 97.03% | 97.03% |
| | | 5 | 16.1 | 8.2 | 1542 | 1492 | 97.03% | 97.03% |
| | | 10 | 11.94 | 0.28 | 1708 | 108 | 97.03% | 3.03% |
| | Gaussian | 1 | 6.7 | 3.53 | 291 | 292 | 97.03% | 97.03% |
| | | 5 | 8.59 | 0.70 | 288 | 225 | 97.22% | 70.70% |
| | | 10 | 8.97 | 1.18 | 295 | 259 | 97.59% | 39.20% |

Table 7. Cont.- LC vs. LC-I

| **Problem** | | | **Time (seconds)** | | **# SV** | | **Generalization error** | |
|---|---|---|---|---|---|---|---|---|
| **w4a** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 22.33 | 5.61 | 281 | 282 | 98.29% | 98.21% |
| | | 5 | 58.82 | 8.50 | 261 | 253 | 98.17% | 98.14% |
| | | 10 | 93.45 | 11.84 | 253 | 270 | 98.01% | 97.96% |
| | Polynomial | 1 | 5.96 | 7.72 | 3100 | 4907 | 98.01% | 97.02% |
| | | 5 | 31.91 | 49.41 | 2134 | 1932 | 97.02% | 97.02% |
| | | 10 | 67.09 | 0.15 | 1757 | 144 | 97.02% | 96.60% |
| | Gaussian | 1 | 13.93 | 1.05 | 365 | 305 | 97.02% | 97.11% |
| | | 5 | 14.53 | 1.95 | 352 | 334 | 97.31% | 97.32% |
| | | 10 | 16.66 | 3.06 | 354 | 342 | 97.70% | 97.73% |
| **gisette_scale** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 530.75 | 44.43 | 1080 | 991 | 97.9% | 97.5% |
| | | 5 | 580.65 | 46.77 | 1080 | 982 | 97.9% | 97.7% |
| | | 10 | 581.50 | 47 | 1080 | 975 | 97.9% | 97.6% |
| | Polynomial | 1 | 519.60 | 12.64 | 1627 | 1156 | 97.8% | 97.2 % |
| | | 5 | 506.28 | 29.95 | 1462 | 1344 | 98% | 97.8 % |
| | | 10 | 517.64 | 43.87 | 1484 | 1466 | 98% | 98% |
| | Gaussian | 1 | 505.55 | 5.38 | 1666 | 1106 | 97.7% | 96.7 % |
| | | 5 | 493.55 | 13.02 | 1363 | 1250 | 98% | 97.5 % |
| | | 10 | 495.16 | 18.02 | 1400 | 1303 | 97.9% | 97.8 % |
| **mushrooms** | **kernel** | **C** | **LC** | **LC-I** | **LC** | **LC-I** | **LC** | **LC-I** |
| | Linear | 1 | 39.57 | 8.48 | 400 | 339 | 100% | 100% |
| | | 5 | 78.26 | 12.10 | 505 | 236 | 100% | 100% |
| | | 10 | 68.79 | 14.40 | 480 | 217 | 100% | 100% |
| | Polynomial | 1 | 36.77 | 47.35 | 3516 | 2227 | 93.71% | 90.20% |
| | | 5 | 45.22 | 41.30 | 1819 | 1397 | 98.83% | 95.75% |
| | | 10 | 47.64 | 37.86 | 1349 | 973 | 99.87% | 98.15% |
| | Gaussian | 1 | 37.91 | 21.45 | 565 | 525 | 99.87% | 99.32 % |
| | | 5 | 60.05 | 31.34 | 292 | 275 | 99.83% | 99.87% |
| | | 10 | 56.59 | 34.5 | 309 | 259 | 100% | 100% |

Table 8. Cont.- LC vs. LC-I