11

Computational Aspects

Overview. *This chapter presents techniques to compute decision functions $f_{D,\lambda}$ of SVMs and mentions some software tools. In addition, we discuss how to choose suitable hyperparameters used by the regularization term, the kernel, and the loss.*

Prerequisites. *We need Chapter 2 on loss functions, Chapter 4 on kernels, Chapter 8 on SVMs for classification, and Chapter 9 on SVMs for regression problems. Some results from convex analysis, in particular from Section A.6.5 on convex programs and Section 6.5 on oracle inequalities for the purpose of parameter selection, are used.*

This chapter is concerned with the question of how to compute a decision function $f_{D,\lambda}$ of support vector machines for a given data set D. No attempt is made to describe in detail all relevant computational aspects regarding SVMs. From our point of view, there exists such a large body of literature on this topic that even the main research approaches published during the last decade would probably fill a textbook of their own. Here we only show some facets regarding computational aspects of SVMs. More precisely, we will concentrate in this chapter on addressing the following questions.

i) How can we compute the empirical SVM decision function $f_{D,\lambda}$?
ii) Are there algorithms to compute $f_{D,\lambda}$ that work well even for large sample sizes?
iii) Are there loss functions L such that the numerical problem of computing $f_{D,\lambda}$ can be solved especially fast?
iv) Are there loss functions such that $f_{D,\lambda}$ can efficiently be computed and have good robustness properties in the sense of Chapter 10?

After a short introduction, we show in Section 11.1 that empirical SVM solutions $f_{D,\lambda}$ are solutions of specific convex programs. SVMs based on the hinge loss and the logistic loss for classification purposes, ϵ-insensitive loss and least squares loss for regression, and pinball loss function for kernel-based quantile regression are treated as special cases. Some computational aspects of computing $f_{D,\lambda}$ efficiently for large sample sizes n are considered in Section 11.2, where special consideration is given to the hinge loss. SVMs depend in general not only on the loss function L and the kernel k but also on the determination of hyperparameters such as $\lambda > 0$, kernel parameters such as γ^2 for the Gaussian RBF kernel, or the parameter ϵ for the ϵ-insensitive loss

function. These hyperparameters can have a substantial impact on the quality of $f_{\mathrm{D},\lambda}$ and $\mathcal{R}_{L,\mathrm{D}}(f_{\mathrm{D},\lambda})$ in applications. Section 11.3 lists some techniques to determine suitable choices of these hyperparameters. Section 11.4 mentions a few software tools available to compute empirical SVMs.

11.1 SVMs, Convex Programs, and Duality

In this section, it will be shown that the decision function $f_{\mathrm{D},\lambda}$ of SVMs for a given data set D with n data points is a solution of a certain *finite-dimensional* convex program. Throughout this section, we make the following assumptions if not otherwise stated.

Assumption 11.1 *Let $L : X \times Y \times \mathbb{R} \to [0,\infty)$ be a convex loss function, H a reproducing kernel Hilbert space over a non-empty convex set X with positive definite kernel k and canonical feature map $\Phi(x) := k(\cdot, x)$, $x \in X$. Furthermore, let $\lambda > 0$ be the regularization parameter and $D = \{(x_i, y_i), i = 1,\ldots,n\} \subset X \times Y$ be a fixed data set with corresponding empirical measure $\mathrm{D} = \frac{1}{n}\sum_{i=1}^{n} \delta_{(x_i,y_i)}$.*

We know by the representer theorem (see Theorem 5.5) that there exists a unique empirical SVM solution $f_{\mathrm{D},\lambda} \in H$ satisfying

$$\mathcal{R}^{reg}_{L,\mathrm{D},\lambda}(f_{\mathrm{D},\lambda}) = \inf_{f\in H} \mathcal{R}_{L,\mathrm{D}}(f) + \lambda\|f\|_H^2 \,. \tag{11.1}$$

In addition, there exists a vector $\bar{\alpha} = (\bar{\alpha}_1,\ldots,\bar{\alpha}_n) \in \mathbb{R}^n$, which in general is not uniquely determined, such that the empirical SVM solution has the representation

$$f_{\mathrm{D},\lambda} = \sum_{j=1}^{n} \bar{\alpha}_j \Phi(x_j) \in H_{|X'} \,, \tag{11.2}$$

where $H_{|X'} := \operatorname{span}\{\Phi(x_j) : j = 1,\ldots,n\}$. For notational convenience, let us denote for each $\alpha \in \mathbb{R}^n$ the corresponding function in $H_{|X'}$ by $w(\alpha)$,

$$w(\alpha) = \sum_{j=1}^{n} \alpha_j \Phi(x_j).$$

Note that we have

$$\|w(\alpha)\|_H^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j k(x_i,x_j) = \alpha^{\mathsf{T}} K \alpha, \tag{11.3}$$

where $K \in \mathbb{R}^{n\times n}$ is the symmetric kernel matrix (or Gram matrix) with coefficients $K_{i,j} := k(x_i,x_j)$ and α^{T} denotes the transpose of the vector α. Furthermore, we obtain

$$f_{\mathrm{D},\lambda}(x) = \langle w(\bar{\alpha}), \Phi(x) \rangle_H = \sum_{j=1}^{n} \bar{\alpha}_j k(x, x_j), \quad x \in X, \tag{11.4}$$

and

$$f_{\mathrm{D},\lambda}(x_i) = \bar{\alpha}_i^{\mathsf{T}} K e_i = e_i^{\mathsf{T}} K \bar{\alpha}_i, \quad i = 1, \ldots, n. \tag{11.5}$$

Note that $f_{\mathrm{D},\lambda} \in H_{|X'}$. Hence it is sufficient to optimize over $H_{|X'}$. Therefore, if we plug (11.3) and (11.4) into (11.1), we obtain

$$\begin{aligned} \mathcal{R}^{reg}_{L,\mathrm{D},\lambda}(f_{\mathrm{D},\lambda}) &= \inf_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L\big(x_i, y_i, f(x_i)\big) + \lambda \|f\|_H^2 \\ &= \min_{w \in H_{|X'}} \frac{1}{n} \sum_{i=1}^{n} L\big(x_i, y_i, \langle w, \Phi(x_i) \rangle_H\big) + \lambda \|w\|^2_{H_{|X'}} \\ &= \min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} L\Big(x_i, y_i, \sum_{j=1}^{n} \alpha_j k(x_i, x_j)\Big) + \lambda\, \alpha^{\mathsf{T}} K \alpha\,. \end{aligned} \tag{11.6}$$

A finite sum of convex functions all defined on the same convex set is of course also a convex function. Therefore, the convexity of the loss function yields the convexity of the first term in (11.6). Furthermore, the kernel k is by Assumption 11.1 positive definite in the sense of Definition 4.15. This shows that the quadratic form $\|w(\alpha)\|_H^2 = \alpha^{\mathsf{T}} K \alpha$ is convex with respect to $\alpha \in \mathbb{R}^n$ whenever the symmetric matrix K is positive semi-definite; see (A.51). Combining these results, we conclude that the *decision function* $f_{\mathrm{D},\lambda}$ *is the solution of a finite-dimensional convex program* (see Definition A.6.22) if the Assumption 11.1 is valid. This is somewhat astonishing because the dimension of the reproducing kernel Hilbert space H is not assumed to be finite. Now recall that L is non-negative. Thus we can rewrite (11.6) in the form

$$\min_{\alpha,\, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \, \|w(\alpha)\|_H^2 \tag{11.7}$$

$$\text{s.t.} \quad \xi_i \geq L\big(x_i, y_i, \langle w(\alpha), \Phi(x_i) \rangle_H\big)\,, \quad i = 1, \ldots, n. \tag{11.8}$$

Hence many classical results about convex programs with constraints such as Lagrange multipliers, determination of saddle points, and algorithms to solve convex programs are applicable for empirical SVMs and will be used in the following considerations; see Section A.6.5 for details on convex programs.

Now we consider the convex programs for $f_{\mathrm{D},\lambda}$ corresponding to several loss functions often used for classification and regression purposes.

Example 11.2 (Margin-based loss). Assume that L is a margin-based loss function in the sense of Definition 2.24; i.e., $L(x, y, t) = \varphi(yt)$ for $y \in \{-1, +1\}$ and $t \in \mathbb{R}$. Hence the convex program (11.7) and (11.8) for $f_{\mathrm{D},\lambda}$ simplifies to

$$\min_{\alpha,\,\xi\in\mathbb{R}^n} \quad \frac{1}{n}\sum_{i=1}^n \xi_i + \lambda \left\|w(\alpha)\right\|_H^2 \tag{11.9}$$

$$\text{s.t.} \quad \xi_i \geq \varphi\big(y_i \left\langle w(\alpha), \Phi(x_i)\right\rangle_H\big), \quad i = 1,\ldots,n. \tag{11.10}$$

The decision function is $f_{\mathrm{D},\lambda} = \sum_{i=1}^n \bar{\alpha}_i \Phi(x_i)$, where $\bar{\alpha} = (\bar{\alpha}_1,\ldots,\bar{\alpha}_n)$ solves (11.9) and (11.10). ◁

Example 11.3 (Classification based on hinge loss). The hinge loss for classification is given by $L_{\mathrm{hinge}}(y,t) := \max\{0, 1-yt\}$ for $y \in \{-1,+1\}$ and $t \in \mathbb{R}$; see Example 2.27. The convex program for $f_{\mathrm{D},\lambda}$ is therefore given by

$$\min_{\alpha,\,\xi\in\mathbb{R}^n} \quad C\sum_{i=1}^n \xi_i + \frac{1}{2}\left\|w(\alpha)\right\|_H^2 \tag{11.11}$$

$$\text{s.t.} \quad \xi_i \geq 0,\ \ \xi_i \geq 1 - y_i \left\langle w(\alpha), \Phi(x_i)\right\rangle_H, \quad i = 1,\ldots,n, \tag{11.12}$$

where $C := 1/(2n\lambda)$ and $w(\alpha) = \sum_{j=1}^n \alpha_j y_j \Phi(x_j)$. The corresponding Lagrangian L^* is given by

$$C\sum_{i=1}^n \xi_i + \frac{1}{2}\left\|w(\alpha)\right\|_H^2 + \sum_{i=1}^n \alpha_i\big(1 - y_i \left\langle w(\alpha), \Phi(x_i)\right\rangle_H - \xi_i\big) - \sum_{i=1}^n \eta_i \xi_i\,,$$

where $\alpha := (\alpha_1,\ldots,\alpha_n) \in [0,\infty)^n$ and $\eta := (\eta_1,\ldots,\eta_n) \in [0,\infty)^n$. The corresponding dual program is found by differentiating L^* with respect to α and $\xi = (\xi_1,\ldots,\xi_n)$ imposing stationarity,

$$\nabla_\alpha L^*(\alpha,\xi) = w(\alpha) - \sum_{i=1}^n \alpha_i y_i \Phi(x_i) = 0, \tag{11.13}$$

$$\frac{\partial L^*}{\partial \xi_i} = C - \alpha_i - \eta_i = 0, \tag{11.14}$$

and resubstituting these relations into the primal problem. We obtain the dual program

$$\max_{\alpha\in[0,C]^n} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j k(x_i,x_j), \tag{11.15}$$

where the box constraint $\alpha \in [0,C]^n$ results from $\alpha_i \in [0,\infty)$, $\eta_i \in [0,\infty)$, and $\alpha_i = C - \eta_i$ due to (11.14). Therefore, $f_{\mathrm{D},\lambda}$ is the unique solution of even a *quadratic program with box constraints* if the hinge loss is used. We will see later on that this fact allows us to construct fast algorithms because quadratic problems can often be solved faster than general convex problems. For the case of an additional offset term b (i.e., we are computing $(f_{\mathrm{D},\lambda}, b_{\mathrm{D},\lambda}) \in H \times \mathbb{R}$) we have to replace $\langle w(\alpha), \Phi(x_i)\rangle_H$ by $\langle w(\alpha), \Phi(x_i)\rangle_H + b$ and add the additional constraint $\sum_{i=1}^n \alpha_i y_i = 0$ in (11.15); see Exercise 11.2. ◁

Example 11.4 (Classification based on logistic loss). The logistic loss for classification is given by $L_{\text{c-logist}}(y,t) = \varphi(yt) = \ln\bigl(1+\exp(-yt)\bigr)$, $y \in \{-1,+1\}$, $t \in \mathbb{R}$; see Example 2.29. Let us define the function $g : \mathbb{R} \to \mathbb{R}$, $g(r) := \varphi(-r) = \ln(1+e^r)$, $r \in \mathbb{R}$, and $C := 1/(2n\lambda)$. The empirical decision function based on this loss is thus given by

$$f_{\mathrm{D},\lambda} = \arg\min_{\alpha\in\mathbb{R}^n} C\sum_{i=1}^{n} g\bigl(-y_i \langle w(\alpha), \Phi(x_i)\rangle_H\bigr) + \frac{1}{2}\|w(\alpha)\|_H^2 . \tag{11.16}$$

Note that $w(\alpha) = \sum_{j=1}^n \alpha_j y_j \Phi(x_j)$. With $\xi_i := -\sum_{j=1}^n \alpha_j y_i y_j k(x_i,x_j)$ for $i,j \in \{1,\ldots,n\}$, it follows that

$$f_{\mathrm{D},\lambda} = \arg\min_{\alpha\in\mathbb{R}^n} C\sum_{i=1}^{n} g(\xi_i) + \frac{1}{2}\|w(\alpha)\|_H^2 \tag{11.17}$$

and $\|w(\alpha)\|_H^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j k(x_i,x_j)$. The Lagrangian for this problem is given by

$$L^*(\alpha,\xi) = C\sum_{i=1}^{n} g(\xi_i) + \frac{1}{2}\|w(\alpha)\|_H^2 - \sum_{i=1}^{n} \alpha_i\bigl(\xi_i + y_i \langle w(\alpha), \Phi(x_i)\rangle_H\bigr).$$

Note that $g'(r) = e^r/(1+e^r) = 1/\bigl(1+e^{-r}\bigr)$, $r \in \mathbb{R}$, equals the cumulative distribution function of the logistic distribution and that its inverse is the logit function $(g')^{-1}(u) = \ln(u/(1-u))$, $u \in (0,1)$. Computing the partial derivatives of the Lagrangian, we obtain the optimality conditions

$$\nabla_\alpha L^* = w(\alpha) - \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \;=\; 0, \tag{11.18}$$

$$\frac{\partial L^*}{\partial \xi_i} = Cg'(\xi_i) - \alpha_i \;=\; 0, \qquad\qquad i = 1,\ldots,n. \tag{11.19}$$

Define the function $G(u) = u\ln(u) + (1-u)\ln(1-u)$, $u \in (0,1)$. Hence $G'(u) = (g')^{-1}(u)$. It follows from (11.19) that $g'(\xi_i) = \alpha_i/C$ and $\xi_i = (g')^{-1}(\alpha_i/C) = G'(\alpha_i/C)$. We obtain the dual program for kernel-based logistic regression:

$$\min_{\alpha\in(0,C)^n} C\sum_{i=1}^{n} G\Bigl(\frac{\alpha_i}{C}\Bigr) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j k(x_i,x_j), \tag{11.20}$$

which is a finite-dimensional convex program; see also Exercise 11.3. ◁

Example 11.5 (Classification based on least squares loss). The least squares loss function for classification is given by $L_{\mathrm{LS}}(y,t) = (1-yt)^2 = (y-t)^2$, $y \in \{-1,+1\}$, $t \in \mathbb{R}$, considered in Example 2.26. The decision function $f_{\mathrm{D},\lambda}$ based on this loss function solves the convex programming problem

$$\begin{aligned} \min_{\alpha,\, r \in \mathbb{R}^n} \quad & \frac{C}{2} \sum_{i=1}^{n} r_i^2 \, + \frac{1}{2} \, \|w(\alpha)\|_H^2 \\ \text{s.t.} \quad & r_i = 1 - y_i \, \langle w(\alpha), \Phi(x_i) \rangle_H \,, \quad i = 1, \ldots, n, \end{aligned}$$

where $r = (r_1, \ldots, r_n)$ and $C := 1/(\lambda n)$. The Lagrangian is given by

$$L^*(\alpha, r) = \frac{C}{2} \sum_{i=1}^{n} r_i^2 \, + \frac{1}{2} \, \|w(\alpha)\|_H^2 + \sum_{i=1}^{n} \alpha_i \big(1 - y_i \, \langle w(\alpha), \Phi(x_i) \rangle_H - r_i\big).$$

After computing the partial derivatives of L^*, we get the following conditions for optimality:

$$\begin{aligned} w(\alpha) &= \sum_{j=1}^{n} \alpha_j y_j \Phi(x_j), \\ \alpha_i \quad &= C r_i \,, \qquad\qquad\qquad\qquad i = 1, \ldots, n, \\ r_i \quad &= 1 - y_i \, \langle w(\alpha), \Phi(x_i) \rangle_H \,, \quad i = 1, \ldots, n; \end{aligned} \tag{11.21}$$

see Exercise 11.4. This is a system of linear equations with respect to α. Note that sparseness of $f_{\mathrm{D},\lambda}$ is lost due to $\alpha_i = C r_i$ in (11.21). ◁

Example 11.6 (Distance-based loss). Assume that L is a distance-based loss function in the sense of Definition 2.32 (i.e., $L(x, y, t) = \psi(y - t)$, $y, t \in \mathbb{R}$) with $\psi : \mathbb{R} \to \mathbb{R}$ some suitable function. The convex program (11.7) and (11.8) for $f_{\mathrm{D},\lambda}$ simplifies to

$$\min_{\alpha,\, \xi \in \mathbb{R}^n} \quad \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \, \alpha^{\mathsf{T}} K \alpha \tag{11.22}$$

$$\text{s.t.} \quad \xi_i \geq \psi\Big(y_i - \sum_{j=1}^{n} \alpha_j k(x_i, x_j)\Big), \quad i = 1, \ldots, n. \tag{11.23}$$

Let us now consider for the sake of simplicity a convex distance-based loss function of the type

$$L(x, y, t) = \psi(y - t) = \tilde{\psi}\big(\max\big\{0, |y - t| - \epsilon\big\}\big),$$

where $\tilde{\psi} : [0, \infty) \to [0, \infty)$ and $\epsilon \geq 0$. We additionally assume that ψ has a continuous first derivative on $\mathbb{R} \backslash \{-\epsilon, +\epsilon\}$. We obtain

$$\psi(y - t) = \tilde{\psi}\big(\max\{0, y - t - \epsilon\}\big) + \tilde{\psi}\big(\max\{0, t - y - \epsilon\}\big), \quad y, t \in \mathbb{R}.$$

The convex program for $f_{\mathrm{D},\lambda}$ can then be rewritten as

$$\begin{aligned}\min_{\alpha,\,\xi^+,\,\xi^-\in\mathbb{R}^n}\quad & \frac{1}{n}\sum_{i=1}^{n}\bigl(\xi_i^+ + \xi_i^-\bigr) + \lambda\,\alpha^\mathsf{T} K\alpha \\ \text{s.t.}\quad & \xi_i^+ \geq \tilde{\psi}\Bigl(\max\Bigl\{0, y_i - \sum_{j=1}^{n}\alpha_j k(x_i,x_j) - \epsilon\Bigr\}\Bigr),\ i=1,\ldots,n, \\ & \xi_i^- \geq \tilde{\psi}\Bigl(\max\Bigl\{0, \sum_{j=1}^{n}\alpha_j k(x_i,x_j) - y_i - \epsilon\Bigr\}\Bigr),\ i=1,\ldots,n;\end{aligned}$$

see also Exercise 11.5. ◁

Example 11.7 (Regression based on ϵ-insensitive loss). Let $\epsilon > 0$. The ϵ-insensitive loss for regression uses $\psi(y-t) = \max\bigl\{0, |y-t| - \epsilon\bigr\}$ for $y, t \in \mathbb{R}$, considered in Example 2.42. This loss function is obviously an example of the distance-based symmetric loss functions considered in Example 11.6. The convex program for $f_{\mathrm{D},\lambda}$ can be rewritten as

$$\begin{aligned}\min_{\alpha,\,\xi^+,\,\xi^-\in\mathbb{R}^n}\quad & C\sum_{i=1}^{n}\bigl(\xi_i^+ + \xi_i^-\bigr) + \frac{1}{2}\,\alpha^\mathsf{T} K\alpha \\ \text{s.t.}\quad & \xi_i^+ \geq 0,\ \xi_i^+ \geq y_i - \sum_{j=1}^{n}\alpha_j k(x_i,x_j) - \epsilon,\ i=1,\ldots,n, \\ & \xi_i^- \geq 0,\ \xi_i^- \geq \sum_{j=1}^{n}\alpha_j k(x_i,x_j) - y_i - \epsilon,\ i=1,\ldots,n,\end{aligned}$$

where $C := 1/(2n\lambda)$. The Lagrangian is thus given by

$$\begin{aligned}L^*(\alpha,\xi^+,\xi^-) = C\sum_{i=1}^{n}\bigl(\xi_i^+ + \xi_i^-\bigr) \quad & + \frac{1}{2}\,\alpha^\mathsf{T} K\alpha \\ & + \sum_{i=1}^{n}\alpha_i^+\Bigl(y_i - \sum_{j=1}^{n}\alpha_j k(x_i,x_j) - \epsilon - \xi_i^+\Bigr) - \sum_{i=1}^{n}\eta_i^+\xi_i^+ \\ & + \sum_{i=1}^{n}\alpha_i^-\Bigl(\sum_{j=1}^{n}\alpha_j k(x_i,x_j) - y_i - \epsilon - \xi_i^-\Bigr) - \sum_{i=1}^{n}\eta_i^-\xi_i^-,\end{aligned}$$

where the Lagrange multipliers α_i^+, α_i^-, η_i^+, and η_i^-, $i=1,\ldots,n$, have to be non-negative. It follows from the saddle point condition of convex programs (see Theorem A.6.26) that the partial derivatives of L^* with respect to the primal variables $w(\alpha)$, ξ^+, and ξ^- have to vanish for optimality; i.e.,

$$\nabla_\alpha L^* = w(\alpha) - \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-)\Phi(x_i) = 0, \tag{11.24}$$

$$\frac{\partial L^*}{\partial \xi_i^+} = C - \alpha_i^+ - \eta_i^+ = 0\,, \qquad i = 1, \ldots, n,$$

$$\frac{\partial L^*}{\partial \xi_i^-} = C - \alpha_i^- - \eta_i^- = 0\,, \qquad i = 1, \ldots, n.$$

Note that we can easily eliminate η_i^+ and η_i^- from L^* because $\eta_i^+ = C - \alpha_i^+$ due to (11.25) and $\eta_i^- = C - \alpha_i^-$ due to (11.25), $i = 1, \ldots, n$. Substituting (11.24) and (11.25) into the formula for the Lagrangian L^* yields the dual program (see Exercise 11.6)

$$\max_{\alpha^+, \alpha^- \in \mathbb{R}^n} \; \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) y_i - \epsilon \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) - \frac{1}{2}(\alpha_i^+ - \alpha_i^-)^\mathsf{T} K (\alpha_i^+ - \alpha_i^-)$$
$$\text{s.t.} \qquad \alpha_i^+,\, \alpha_i^- \in [0, C], \quad i = 1, \ldots, n\,. \tag{11.25}$$

Due to (11.24), we thus obtain $f_{\mathrm{D},\lambda} = \sum_{i=1}^n \alpha_i \Phi(x_i)$ with $\alpha_i = \alpha_i^+ - \alpha_i^-$, $i = 1, \ldots, n$. The ϵ-insensitive loss and the hinge loss hence share the nice property that $f_{\mathrm{D},\lambda}$ is the unique solution of a *quadratic program with box constraints*.

For the case of an additional offset term b (i.e., we compute $\big(f_{\mathrm{D},\lambda}, b_{\mathrm{D},\lambda}\big) \in H \times \mathbb{R}$) we have to replace $\langle w(\alpha), \Phi(x_i)\rangle_H$ by $\langle w(\alpha), \Phi(x_i)\rangle_H + b$ and to add the constraint $\sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0$ in (11.25) of the dual program. ◁

Example 11.8 (Regression based on least squares loss). The least squares loss function for regression is given by $L_{\mathrm{LS}}(y, t) = (y - t)^2$, $y, t \in \mathbb{R}$. The decision function $f_{\mathrm{D},\lambda}$ based on this loss function solves the convex programming problem

$$\min_{\alpha,\, r \in \mathbb{R}^n} \; \frac{C}{2} \sum_{i=1}^n r_i^2 \, + \frac{1}{2}\, \alpha^\mathsf{T} K \alpha$$
$$\text{s.t.} \quad r_i = y_i - \sum_{j=1}^n \alpha_j k(x_i, x_j)\,, \quad i = 1, \ldots, n,$$

where $r = (r_1, \ldots, r_n)$ and $C := 1/(\lambda n)$. The Lagrangian is given by

$$L^*(\alpha, r) = \frac{C}{2} \sum_{i=1}^n r_i^2 \, + \frac{1}{2}\, \alpha^\mathsf{T} K \alpha + \sum_{i=1}^n \alpha_i \Big(y_i - \sum_{j=1}^n \alpha_j k(x_i, x_j) - r_i \Big)\,.$$

After computing the partial derivatives of L^*, we get the following conditions for optimality:

$$\begin{aligned} w(\alpha) &= \sum_{i=1}^{n} \alpha_i \Phi(x_i), \\ \alpha_i &= C r_i\,, \qquad\qquad i = 1, \ldots, n, \\ r_i &= y_i - \langle w(\alpha), \Phi(x_i) \rangle_H\,, \quad i = 1, \ldots, n, \end{aligned}$$

see Exercise 11.7. This is a system of linear equations with respect to α. Note that one loses the sparseness property due to the conditions $\alpha_i = C r_i$ for $i = 1, \ldots, n$. ◁

Example 11.9 (Quantile regression based on the pinball loss). As our final example in this section, we consider the pinball loss function, which is suitable for kernel based quantile regression (see Example 2.43). This distance-based loss function uses $\psi_\tau(y-t) = (\tau-1)(y-t)$ for $y-t < 0$ and $\psi_\tau(y-t) = \tau(y-t)$ for $y - t \geq 0$, where $\tau \in (0,1)$ specifies the desired quantile level. The convex programming problem to determine $f_{\mathrm{D},\lambda}$ can be rewritten as

$$\min_{\alpha,\, \xi^+,\, \xi^- \in \mathbb{R}^n} \quad C \sum_{i=1}^{n} \bigl(\tau \xi_i^+ + (1-\tau)\xi_i^-\bigr) + \frac{1}{2}\, \alpha^{\mathsf{T}} K \alpha \tag{11.26}$$

$$\text{s.t.} \qquad \xi_i^+ \geq 0, \quad \xi_i^+ \geq y_i - \sum_{j=1}^{n} \alpha_j k(x_i, x_j),\ i = 1, \ldots, n, \tag{11.27}$$

$$\xi_i^- \geq 0, \quad \xi_i^- \geq \sum_{j=1}^{n} \alpha_j k(x_i, x_j) - y_i\,,\ i = 1, \ldots, n, \tag{11.28}$$

where $C = 1/(2n\lambda)$. Using the Lagrangian approach in a way similar to Example 11.7, we obtain the dual program

$$\max_{\alpha \in \mathbb{R}^n} \quad \alpha^{\mathsf{T}} y - \alpha^{\mathsf{T}} K \alpha \tag{11.29}$$

$$\text{s.t.} \quad C(\tau - 1) \leq \alpha_i \leq C\tau, \quad i = 1, \ldots, n\,, \tag{11.30}$$

where $y := (y_1, \ldots, y_n)$ and $C := 1/(2n\lambda)$ (see Exercise 11.8). Note that the box constraints in (11.30) are only symmetric around 0 for $\tau = \frac{1}{2}$; i.e., for median regression. It follows from (11.29) and (11.30) that $f_{\mathrm{D},\lambda}$ based on the pinball loss is the unique solution of even a *quadratic program with box constraints*.

For the case of an additional offset term b (i.e., we compute $\bigl(f_{\mathrm{D},\lambda}, b_{\mathrm{D},\lambda}\bigr) \in H \times \mathbb{R}$), we have to replace $\sum_{j=1}^{n} \alpha_j k(x_i, x_j)$ by $\sum_{j=1}^{n} \alpha_j k(x_i, x_j) + b$ and to add the constraint $\sum_{i=1}^{n} \alpha_i = 0$ in (11.30).

Suppose that we want to estimate $m \geq 2$ conditional quantile functions with quantile levels $0 < \tau_1 < \ldots < \tau_m < 1$ on the same data set D. Let us denote the corresponding empirical SVM solutions by $f_{\mathrm{D},\lambda,\tau_h}$, $h = 1, \ldots, m$. Then it can occur that for some $x \in X$ and some pair (h_1, h_2) with $1 \leq h_1 < h_2 \leq m$, the conditional quantile estimates are in reversed order; i.e.,

$$f_{\mathrm{D},\lambda,\tau_{h_1}}(x) > f_{\mathrm{D},\lambda,\tau_{h_2}}(x).$$

This undesired phenomenon is called the *crossing problem* and is not specific to SVMs based on the pinball loss but can also occur in classical parametric quantile regression. The reason why this phenomenon can occur is that the conditional quantile functions are independently estimated. One technique to overcome this problem is to fit all m conditional quantile functions simultaneously and to add constraints that enforce that the crossing problem cannot occur at ℓ points $\{x_j \in X : j = 1, \ldots, \ell\}$. Let us write the empirical SVM solution for the quantile level τ_h as $f_{\mathrm{D},\lambda,\tau_h}(x) = \langle w(\alpha_h), \Phi(x)\rangle_H$, where $w(\alpha_h) \in H_{|X'}$ for $j = 1, \ldots, m$, $x \in X$. The non-crossing constraints can be specified as linear constraints,

$$\langle w(\alpha_h), \Phi(x_j)\rangle_H \le \langle w(\alpha_{h+1}), \Phi(x_j)\rangle_H, \quad 1 \le h \le m-1,\ 1 \le j \le \ell,$$

in $H_{|X'}$. The primal optimization problem becomes

$$\begin{aligned}
\min_{w(\alpha_h),\,\xi_h^+,\,\xi_h^-,\,1\le h\le m-1} \quad & \sum_{h=1}^{m}\left(C\sum_{i=1}^{n}\Big(\tau_h\xi_{h,i}^+ + (1-\tau_h)\xi_{h,i}^-\Big) + \frac{1}{2}\|w(\alpha_h)\|_H^2\right) \\
\text{s.t.} \quad & \xi_{h,i}^+ - \xi_{h,i}^- = y_i - \langle w(\alpha_h), \Phi(x_i)\rangle_H, \\
& \qquad 1 \le h \le m,\ 1 \le i \le n, \\
& \langle w(\alpha_{h+1}), \Phi(x_j)\rangle_H - \langle w(\alpha_h), \Phi(x_j)\rangle_H \ge 0, \qquad (11.31) \\
& \qquad 1 \le h \le m-1,\ 1 \le j \le \ell.
\end{aligned}$$

Using the Lagrangian approach, we obtain the corresponding dual problem,

$$\begin{aligned}
\max_{\alpha_h,\,\beta_h\in\mathbb{R}^n,\,1\le h\le m-1} \quad & \sum_{h=1}^{m}\Big(\alpha_h^{\mathsf{T}} y - \frac{1}{2}\alpha_h^{\mathsf{T}} K \alpha_h - \alpha_h^{\mathsf{T}}\tilde{K}(\beta_{h-1} - \beta_h) \\
& \qquad - \frac{1}{2}(\beta_{h-1} - \beta_h)^{\mathsf{T}}\bar{K}(\beta_{h-1} - \beta_h)\Big) \\
\text{s.t.} \quad & C(\tau_h - 1) \le \alpha_{h,i} \le C\tau_h, \quad 1 \le h \le m,\ 1 \le i \le n, \\
& \beta_{h,j} \ge 0, \quad 1 \le h \le m,\ 1 \le j \le \ell,
\end{aligned}$$

where $\beta_{h,j}$ is the Lagrange multiplier from (11.31), $\tilde{K} \in \mathbb{R}^{n\times\ell}$ with entries $\tilde{K}_{i,j} = k(x_i, x_j)$, $\bar{K} \in \mathbb{R}^{\ell\times\ell}$ with entries $\bar{K}_{i,j} = k(x_i, x_j)$, and $\beta_h = (\beta_{h,1}, \ldots, \beta_{h,\ell}) \in \mathbb{R}^\ell$ for $h = 1, \ldots, m$. The empirical SVM solution for the conditional quantile function for quantile level τ_h is given by

$$f_{\mathrm{D},\lambda,\tau_h} = \sum_{i=1}^{n}\alpha_{h,i}\Phi(x_i) + \sum_{j=1}^{\ell}(\beta_{h-1,i} - \beta_{h,i})\Phi(x_j)\,. \qquad \triangleleft$$

11.2 Implementation Techniques

We saw in the previous section that SVM decision functions $f_{\mathrm{D},\lambda}$ are determined via the solution of convex programs. Hence classical results about

convex programs such as Lagrange multipliers, saddle points, and algorithms to solve convex programs can be used to compute $f_{\mathrm{D},\lambda}$. For details on convex programs and related topics, we refer to Section A.6.5.

There are several standard numerical methods to solve convex programs. The Nelder-Mead search (Nelder and Mead, 1965) is a versatile optimization algorithm that can even be used to minimize continuous but non-differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$.[1] The Nelder-Mead algorithm does not compute or approximate gradients or Hessian matrices. It is based on the iterative construction of a simplex with $(n+1)$ points, and the function values at the points of this simplex are computed. Then the points of the simplex are iteratively changed by contraction, reflection, or expansion in an adaptive way. The Nelder-Mead search is in general not fast because many function values have to be evaluated before the simplex contracts to a point (i.e., before the algorithm converges) but it is numerically stable and easy to use.

The idea of gradient descent algorithms is to start with an initial vector $\alpha^{(0)}$ for α. Then a sequence of vectors $\alpha^{(\ell)}$, $\ell \in \mathbb{N}$, is iteratively computed such that $\alpha^{(\ell+1)}$ is evaluated on the basis of $\alpha^{(\ell)}$ and $\alpha^{(\ell+1)}$ is in the direction where the gradient of $g\big(\alpha^{(\ell)}\big)$ has steepest descent.

From a numerical point of view, algorithms such as Newton-Raphson are inefficient to solve convex programs for large sample sizes n that need to store the kernel matrix K into the RAM of the computer or use matrix inversions of K. The main reason is that K is an $n \times n$ matrix and in general not sparse. Hence, even if one takes the symmetry of K into account, one has to store approximately $n(n+1)/2$ coefficients. If 8 bytes are needed to store a single coefficient in double precision, one needs approximately $4n(n+1)$ bytes to store K on a computer. Table 11.1 clearly shows that the storage space needed to store K increases substantially if n increases. For large data sets, say with at least a million data points (x_i, y_i), it is not possible to store K in the RAM of current standard PCs or on a small cluster of workstations. Data sets with more than a million data points are now not unusual in bioinformatics or data mining projects (see also Chapter 12). This is one reason why subsampling strategies such as robust learning from bites (see Section 10.5) can be useful for data sets of this size.

If the sample size n is small to moderate, *interior point algorithms* belong to the most reliable and accurate optimization techniques. The main idea of interior point algorithms when used to compute an empirical SVM solution $f_{\mathrm{D},\lambda}$ is to solve the primal and the dual programs simultaneously. This is done by gradually enforcing the Karush-Kuhn-Tucker conditions to iteratively find a feasible solution. A vector α is called a *feasible solution* of a convex program if α is an element of the set over which the optimization is carried out and if α satisfies the constraints of the convex program. The *duality gap* between the objective functions of the primal and the dual programs is used to determine the quality of the current set of variables and to check whether the stopping

[1] There are variants of the Nelder-Mead algorithm that can deal with constraints.

Table 11.1. Relationship between sample size and space needed to store K.

Sample Size n	Storage Space $4n(n+1)$
100	40 KB
1000	4 MB
10000	400 MB
100000	40 GB
1000000	4 TB

criteria are fulfilled. For large-sized optimization problems, it is sometimes necessary to use approximations of interior point algorithms.

If the sample size n is rather large, *decomposition methods* are often helpful to compute $f_{\mathrm{D},\lambda}$. As explained before, the kernel matrix $K = \big(k(x_i, x_j)\big) \in \mathbb{R}^{n\times n}$ is often fully dense and may be too large to be stored in the RAM of a computer for large values of n. Decomposition methods are designed to handle this difficulty by breaking the optimization problem into smaller and manageable subproblems and solving these in an iterative manner. This technique to tackle the numerical problem is commonly referred to as *chunking* (Vapnik, 1982) or as *subset selection*. In other words, in contrast to many optimization methods for convex problems, where the whole vector $\alpha \in \mathbb{R}^n$ of the dual problem is iteratively updated in each step, decomposition methods modify only a subset of α in each iteration step. This subset, which is generally denoted as the *working set*

$$B := \{\alpha_j : j \in J \subset \{1, \ldots, n\}, |J| = q\},$$

leads to a relatively small subproblem to be minimized in each iteration step.

The idea of *sequential minimal optimization* proposed by Platt (1999) is to use the decomposition method in an extreme manner: the optimization step is done for only $q = 2$ points at each iteration step. At first sight, this approach might look too simple to be useful, but the opposite is true. SMO can be very effective for SVMs because the optimization problem for only two points can often be calculated analytically. This eliminates calling an iterative convex program optimizer at each iteration step, which can therefore save a lot of computation time. To describe this method, let us consider the convex problem

$$\min_{\alpha\in(0,C)^n} \quad C\sum_{i=1}^{n} G\Big(\frac{\alpha_i}{C}\Big) + \frac{1}{2}\alpha^{\mathsf{T}} K\alpha \tag{11.32}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{11.33}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$, G is some real-valued convex function, and $K = \big(k(x_i, x_j)\big) \in \mathbb{R}^{n\times n}$ denotes a positive definite kernel matrix. This problem

is typical for SVMs; see Example 11.4 and Exercise 11.3 on SVMs based on the logistic loss for classification problems. The basic algorithm for an SMO-type decomposition with $|B| = q = 2$ is then given by the following four steps.

Sequential Minimal Optimization Algorithm (SMO)

i) Find an initial feasible solution $\alpha^{(1)} \in \mathbb{R}^n$ fulfilling the constraints and set $\ell = 1$.

ii) If $\alpha^{(\ell)}$ solves the dual program with the desired numerical precision, then stop. Otherwise, find a working set $B^{(\ell)} \subset \{1, \ldots, n\}$ with $|B^{(\ell)}| = 2$.

iii) Define $A^{(\ell)} := \{\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n : \alpha_j = \alpha_j^{(\ell)} \text{ for } j \notin B^{(\ell)}\}$. Solve the following subproblem with respect to the two variables α_j, $j \in B^{(\ell)}$:

$$\begin{aligned} \min_{\alpha \in A^{(\ell)}} \quad & C \sum_{i=1}^{n} G\Big(\frac{\alpha_i}{C}\Big) + \frac{1}{2} \alpha^{\mathsf{T}} K \alpha \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

This optimization problem is equivalent to

$$\begin{aligned} \min_{\alpha \in A^{(\ell)}} \quad & C \sum_{i \in B^{(\ell)}} G(\alpha_i/C) + \frac{1}{2} \sum_{i \in B^{(\ell)}} \sum_{j \in B^{(\ell)}} \alpha_i \alpha_j k(x_i, x_j) + c_*^{(\ell)} + \sum_{i \in B^{(\ell)}} \alpha_i c_i^{(\ell)} \\ \text{s.t.} \quad & \textstyle\sum_{i \in B^{(\ell)}} \alpha_i y_i = c_{**}^{(\ell)}, \end{aligned}$$

where

$$c_*^{(\ell)} = C \sum_{i \notin B^{(\ell)}} G(\alpha_i/C) + \frac{1}{2} \sum_{i \notin B^{(\ell)}} \sum_{j \notin B^{(\ell)}} \alpha_i \alpha_j k(x_i, x_j), \tag{11.34}$$

$$c_i^{(\ell)} = \sum_{j \notin B^{(\ell)}} \alpha_j k(x_i, x_j), \quad i \in B^{(\ell)}, \tag{11.35}$$

$$c_{**}^{(\ell)} = \sum_{j \notin B^{(\ell)}} \alpha_j y_j \,. \tag{11.36}$$

iv) Set $\alpha^{(\ell+1)}$ to be an optimal solution of the optimization problem given in step *iii)*. If the stopping rules of the algorithm are not yet met, increase ℓ to $\ell + 1$ and go to step *ii)*.

Of course, two problems are left open in this SMO-type decomposition algorithm: how to select the working sets and how to specify stopping rules. However, before we address these questions, let us first give an example for the SMO-type decomposition for the special case of the hinge loss. Let us further assume that there is a bias term $b \in \mathbb{R}$ to be estimated. The dual problem is thus

$$\max_{\alpha\in\mathbb{R}^n} \ \alpha^{\mathsf{T}}1 - \frac{1}{2}\alpha^{\mathsf{T}}\tilde{K}\alpha \tag{11.37}$$

$$\text{s.t.} \quad \alpha \in [0,C]^n \text{ and } \alpha^{\mathsf{T}}y = 0, \tag{11.38}$$

where C is the upper bound, $1 := (1,\ldots,1) \in \mathbb{R}^n$, $y = (y_1,\ldots,y_n) \in \{-1,+1\}^n$, and $\tilde{K} := \big(y_i y_j k(x_i,x_j)\big) \in \mathbb{R}^{n\times n}$ (see Exercise 11.2). Let $B \subset \{1,\ldots,n\}$ and denote by 1_B and 1_{B^c} vectors with $|B|$ and $|B^c|$ coefficients, respectively, all being equal to one. The first algorithm for an SMO-type decomposition is then given by the following algorithm.

Sequential Minimal Optimization Algorithm 1 (ALG1)

i) Find $\alpha^{(1)} \in \mathbb{R}^n$ as an initial feasible solution. Set $\ell = 1$.

ii) If $\alpha^{(\ell)}$ solves the dual program up to the desired numerical precision, stop. Otherwise, find a working set $B := \{i,j\} \subset \{1,\ldots,n\}$ with $|B| = 2$. Define the complement of B by $B^c = \{1,\ldots,n\}\backslash B$ and $\alpha_B^{(\ell)}$ and $\alpha_{B^c}^{(\ell)}$ being the subvectors of $\alpha^{(\ell)}$ with index sets B and B^c, respectively.

iii) Solve the following subproblem with the dual variable α_B, where we use obvious matrix notation:

$$\begin{aligned}
\max_{\alpha_B\in\mathbb{R}^2} \quad & \begin{bmatrix}\alpha_B^{\mathsf{T}} & (\alpha_{B^c}^{(\ell)})^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}1_B\\ 1_{B^c}\end{bmatrix} - \frac{1}{2}\begin{bmatrix}\alpha_B^{\mathsf{T}} & (\alpha_{B^c}^{(\ell)})^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\tilde{K}_{B,B} & \tilde{K}_{B,B^c}\\ \tilde{K}_{B^c,B} & \tilde{K}_{B^c,B^c}\end{bmatrix}\begin{bmatrix}\alpha_B\\ \alpha_{B^c}^{(\ell)}\end{bmatrix} \\
&= \big(1_B - \tilde{K}_{B,B^c}\alpha_{B^c}^{(\ell)}\big)\alpha_B - \frac{1}{2}\alpha_B^{\mathsf{T}}\,\tilde{K}_{B,B}\,\alpha_B \ + \ c \\
&= \big(1_B - \tilde{K}_{B,B^c}\alpha_{B^c}^{(\ell)}\big)\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} - \frac{1}{2}[\alpha_i \ \ \alpha_j]\begin{bmatrix}\tilde{K}_{i,i} & \tilde{K}_{i,j}\\ \tilde{K}_{j,i} & \tilde{K}_{j,j}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + c \\
\text{s.t.} \quad & \alpha_B \in [0,C]^2, \ \ \alpha_i y_i + \alpha_j y_j = -\sum\nolimits_{i\in B^c}\alpha_i^{(\ell)} y_i\,,
\end{aligned}$$

where c is some constant independent of α_B and

$$\begin{bmatrix}\tilde{K}_{B,B} & \tilde{K}_{B,B^c}\\ \tilde{K}_{B^c,B} & \tilde{K}_{B^c,B^c}\end{bmatrix}$$

is a permutation of K where rows and columns of K are permuted such that $\tilde{K}_{B,B} \in \mathbb{R}^{2\times 2}$ contains the elements of K with indexes $(i,j) \in B\times B$. The submatrices $\tilde{K}_{B,B^c}$, $\tilde{K}_{B^c,B}$, and $\tilde{K}_{B^c,B^c}$ are analogously constructed.

iv) Set $\alpha_B^{(\ell+1)}$ to be an optimal solution of the optimization problem given in step *iii)* and set $\alpha_{B^c}^{(\ell+1)} := \alpha_{B^c}^{(\ell)}$. If the stopping rules of the algorithm are not yet met, then increase ℓ to $\ell+1$ and go to step *ii)*.

Note that we denoted the working set by B instead of $B^{(\ell)}$ at every iteration step because misunderstandings are unlikely, although the working set changes from one iteration to another.

Now some results concerning the *working set selection* will be given. The decomposition method clearly has the advantage compared to some other

techniques to solve convex programs that this method can be used for data sets with large sample sizes n without storing the kernel matrix $K \in \mathbb{R}^{n\times n}$ or $\tilde{K} \in \mathbb{R}^{n\times n}$ into the RAM of the computer. However, since only q components of $\alpha \in \mathbb{R}^n$ with $q \ll n$ are updated per iteration, the decomposition method can suffer from slow convergence if n is large. In other words, even if any single iteration step can be done fast, the overall computation time can be long if many iteration steps are necessary. Hence it is essential to choose the working sets in a suitable way to reduce the number of iterations and the computation time. Some methods rely on a violation of the optimality condition for the subproblems. Several gradient-based methods belong to this class of methods. Recent research indicates that using a second-order approximation of the objective function to be optimized in the subproblems generally leads to faster overall convergence.

We will now describe an algorithm for SVMs based on the hinge loss. The dual program given in (11.37) and (11.38) is a quadratic program. Therefore, a second-order approximation

$$g : \mathbb{R}^n \to \mathbb{R}, \quad g(\alpha) := \frac{1}{2}\alpha^\mathsf{T}\tilde{K}\alpha - \alpha^\mathsf{T}1, \quad \alpha \in \mathbb{R}^n, \tag{11.39}$$

of the convex objective function to be minimized in the subproblems directly relates to a decrease of the objective function. Let us denote the gradient of $g(\alpha)$ by $\nabla g(\alpha) = \tilde{K}\alpha - 1$. To shorten the notation, we will often write $\nabla g(\alpha)_i$ instead of $\big(\nabla g(\alpha)\big)_i$, $i = 1, \ldots, n$.

One way to select the working set B is via a maximal violating pair that will formally be defined in Definition 11.10. Often we will write B instead of $B^{(\ell)}$ if misunderstandings are unlikely. The *maximal violating pair algorithm* (Keerthi *et al.*, 2001) can be described as follows.

Maximal Violating Pair Algorithm (ALG2)

i) Select two indexes $i, j \in \{1, \ldots, n\}$ such that

$$\begin{aligned} i &\in \arg\max\big\{-y_h \nabla g(\alpha^{(\ell)})_h : h \in I_{up}(\alpha^{(\ell)})\big\}, \\ j &\in \arg\min\big\{-y_h \nabla g(\alpha^{(\ell)})_h : h \in I_{low}(\alpha^{(\ell)})\big\}, \end{aligned}$$

where

$$\begin{aligned} I_{up}(\alpha) &:= \big\{h \in \{1, \ldots, n\} : \alpha_h > 0, y_h = -1 \text{ or } \alpha_h < C, y_h = +1\big\}, \\ I_{low}(\alpha) &:= \big\{h \in \{1, \ldots, n\} : \alpha_h > 0, y_h = +1 \text{ or } \alpha_h < C, y_h = -1\big\}. \end{aligned}$$

ii) Choose the working set $B := \{i, j\}$.

This technique to choose B is motivated by the Karush-Kuhn-Tucker conditions (see Section A.6.5): a vector $\alpha \in \mathbb{R}^n$ is a stationary point of the dual program (11.37) and (11.38) if and only if there exists a number $b \in \mathbb{R}$ and two vectors $\eta = (\eta_1, \ldots, \eta_n) \in [0, \infty)^n$ and $\mu = (\mu_1, \ldots, \mu_n) \in [0, \infty)^n$ such that

$$\begin{aligned} \nabla g(\alpha) + by &= \eta - \mu\,, \\ \eta_i \alpha_i &= 0, \\ \mu_i (C - \alpha_i) &= 0, \quad i = 1, \ldots, n. \end{aligned}$$

Note that we can rewrite this condition as

$$\begin{aligned} \nabla g(\alpha)_i + by_i &\geq 0, \text{ if } \alpha_i < C, \\ \nabla g(\alpha)_i + by_i &\leq 0, \text{ if } \alpha_i > 0, \end{aligned}$$

which is equivalent to

$$\begin{aligned} -y_i \nabla g(\alpha)_i &\leq b, \text{ if } i \in I_{up}(\alpha), \\ -y_i \nabla g(\alpha)_i &\geq b, \text{ if } i \in I_{low}(\alpha). \end{aligned}$$

Using these relations together with $Y = \{-1, +1\}$, we see that a feasible vector $a \in \mathbb{R}^n$ is a stationary point of (11.37) and (11.38) if and only if

$$m(\alpha) := \max_{i \in I_{up}(\alpha)} -y_i \nabla g(\alpha)_i \leq \min_{i \in I_{low}(\alpha)} -y_i \nabla g(\alpha)_i =: M(\alpha)\,. \tag{11.40}$$

Note that the maximum and the minimum in (11.40) are well-defined except for the case where $\sum_{i=1}^n y_i \in \{-n, +n\}$. For these special cases, the vector $(0, \ldots, 0) \in \mathbb{R}^n$ is the only feasible solution, and the iterative decomposition method already stops at the first step, provided we initialize with $(0, \ldots, 0)$.

Definition 11.10. *Consider a support vector machine based on the hinge loss with dual problem (11.37) and (11.38).*

i) A pair of indexes $\{i, j\} \subset \{1, \ldots, n\}$ with $i \neq j$ is called a ***violating pair*** *if $i \in I_{up}(\alpha)$, $j \in I_{low}(\alpha)$, and $y_i \nabla g(\alpha)_i < y_j \nabla g(\alpha)_j$.*

ii) A ***maximal violating pair*** *is a violating pair such that the difference $y_j \nabla g(\alpha)_j - y_i \nabla g(\alpha)_i$ is maximized over all violating pairs.*

A maximal violating pair is clearly a plausible choice of the working set B. Hush and Scovel (2003) showed for $\tilde{K}$ positive semi-definite that the sequence $\big(g(\alpha^{(\ell)})\big)_{\ell \in \mathbb{N}}$ strictly decreases for SMO-type methods if and only if the working set $B^{(\ell)}$ is a violating pair in each iteration. Unfortunately, having a violating pair even a strict decrease of $\big(g(\alpha^{(\ell)})\big)_{\ell \in \mathbb{N}}$ does not guarantee the convergence to a stationary point for $\ell \to \infty$. There exist counterexamples[2] if the working set B is a violating pair but not a maximal violating pair.

In the following we will show that the maximal violating pair is related to a *first-order approximation* of the objective function $g(\alpha)$ for an empirical SVM solution for classification based on the hinge loss having the dual problem (11.37) and (11.38). More precisely, the pair $\{i, j\}$ selected by the maximal violating pair algorithm satisfies

[2] See, e.g., Chen *et al.* (2006, pp. 895ff.).

$$\{i, j\} = \arg\min_B \mathrm{Sub}(B), \tag{11.41}$$

where the subproblem $\mathrm{Sub}(B)$ is defined by

$$\begin{aligned}
\mathrm{Sub}(B) := \min_{d_B} & \big(\nabla g(\alpha^{(\ell)})\big)_B^{\mathsf{T}} d_B && (11.42)\\
\text{s.t. } & d_B^{\mathsf{T}} y_B = 0, && (11.43)\\
& d_h \geq 0, \text{ if } \alpha_h^{(\ell)} = 0,\ h \in B, && (11.44)\\
& d_h \leq 0, \text{ if } \alpha_h^{(\ell)} = C,\ h \in B, && (11.45)\\
& d_h \in [-1, +1],\ h \in B, && (11.46)
\end{aligned}$$

the minimization of $\mathrm{Sub}(B)$ is over all subsets $B \subset \{1, \ldots, n\}$ with $|B| = 2$, and $\alpha_B^{(\ell)}, d_B, y_B \in \mathbb{R}^2$ are the subvectors of $\alpha^{(\ell)}, d, y \in \mathbb{R}^n$, respectively, where only coefficients with indexes in B are considered.

Now we will show that a maximal violating pair solves (11.41), provided that there exists at least one violating pair. Let us define $d := [d_B,\, 0_{B^c}] \in \mathbb{R}^n$. Then the objective function in (11.42) is obtained as a first-order approximation of $g(\alpha^{(\ell)} + d)$ because

$$g(\alpha^{(\ell)} + d) \;\approx\; g(\alpha^{(\ell)}) + \big(\nabla g(\alpha^{(\ell)})\big)^{\mathsf{T}} d \;=\; g(\alpha^{(\ell)}) + \big(\nabla g(\alpha^{(\ell)})\big)_B^{\mathsf{T}} d_B\,. \tag{11.47}$$

The constraint in (11.43) is from $(\alpha^{(\ell)} + d)^{\mathsf{T}} y = 0$ and $(\alpha^{(\ell)})^{\mathsf{T}} y = 0$. The condition $\alpha \in [0, C]^n$ leads to the inequalities (11.44) and (11.45), and (11.46) avoids that the value of the objective function approaches $-\infty$ because the function in (11.42) is linear. It is not necessary to consider all $n(n-1)/2$ possible subsets of $\{1, \ldots, n\}$ having two elements to find the optimal subset because the maximal violating pair algorithm solves (11.41) in $\mathcal{O}(n)$ steps, as can be seen as follows. For any set $\{i, j\} \subset \{1, \ldots, n\}$ with $i \neq j$, define $\hat{d}_i := y_i d_i$ and $\hat{d}_j := y_j d_j$ in (11.42)–(11.46). The objective function becomes

$$\big(-y_i \nabla g(\alpha^{(\ell)})_i + y_j \nabla g(\alpha^{(\ell)})_j\big)\hat{d}_j\,. \tag{11.48}$$

As $d_i = d_j = 0$ is feasible for (11.42)–(11.46), the minimum of (11.48) is less than or equal to zero. If $y_i \nabla g(\alpha^{(\ell)})_i < y_j \nabla g(\alpha^{(\ell)})_j$, then the term in (11.48) is negative if and only if $\hat{d}_j < 0$ and $\hat{d}_i > 0$ because $\hat{d}_i + \hat{d}_j = 0$. From the definition of $I_{low}(\alpha)$ and $I_{up}(\alpha)$ and (11.44) and (11.45), this corresponds to $i \in I_{up}(\alpha^{(\ell)})$ and $j \in I_{low}(\alpha^{(\ell)})$. Furthermore, the minimum occurs at $\hat{d}_i = -\hat{d}_j = 1$. Similar relations hold for $y_i \nabla g(\alpha^{(\ell)})_i > y_j \nabla g(\alpha^{(\ell)})_j$. Hence, solving (11.41) is essentially the same as solving

$$\begin{aligned}
&\min\Big\{\min\big\{0, y_i \nabla g(\alpha^{(\ell)})_i - y_j \nabla g(\alpha^{(\ell)})_j\big\} : i \in I_{up}(\alpha^{(\ell)}), j \in I_{low}(\alpha^{(\ell)})\Big\}\\
&= \min\Big\{0, -\max_{i \in I_{up}(\alpha^{(\ell)})} \big(-y_i \nabla g(\alpha^{(\ell)})_i\big) + \min_{i \in I_{low}(\alpha^{(\ell)})} \big(-y_i \nabla g(\alpha^{(\ell)})_i\big)\Big\}\,.
\end{aligned}$$

If we take (11.40) into account, we obtain that a maximal violating pair solves (11.41), provided that there exists at least one violating pair.

We continue with considering the dual program (11.37) and (11.38) for $f_{\mathrm{D},\lambda}$ based on the hinge loss. The objective function g in (11.39) is quadratic, and

$$\begin{aligned} g(\alpha^{(\ell)} + d) - g(\alpha^{(\ell)}) &= \big(\nabla g(\alpha^{(\ell)})\big)^{\mathsf{T}} d + \frac{1}{2} d^{\mathsf{T}} \nabla^2 g(\alpha^{(\ell)}) d \\ &= \big(\nabla g(\alpha^{(\ell)})\big)_B^{\mathsf{T}} d_B + \frac{1}{2} d_B^{\mathsf{T}} \big(\nabla^2 g(\alpha^{(\ell)})\big)_{B,B} d_B \end{aligned} \tag{11.49}$$

equals the reduction of the value of the objective function. Therefore, we obtain a selection method for the working set, taking the result of a second-order approximation into account if we replace the objective function in (11.42) by (11.49). We obtain the following *second-order approximation algorithm*

$$\{i, j\} = \arg\min_B \mathrm{Sub}(B), \tag{11.50}$$

where the subproblem $\mathrm{Sub}(B)$ is given by

$$\mathrm{Sub}(B) := \min_{d_B} \frac{1}{2} d_B^{\mathsf{T}} \big(\nabla^2 g(\alpha^{(\ell)})\big)_{B,B} d_B + \big(\nabla g(\alpha^{(\ell)})\big)_B^{\mathsf{T}} d_B \tag{11.51}$$

$$\text{s.t. } d_B^{\mathsf{T}} y_B = 0, \tag{11.52}$$

$$d_h \geq 0, \text{ if } \alpha_h^{(\ell)} = 0,\ h \in B, \tag{11.53}$$

$$d_h \leq 0, \text{ if } \alpha_h^{(\ell)} = C,\ h \in B, \tag{11.54}$$

the minimum is taken over all sets $B \subset \{1, \ldots, n\}$ with $|B| = 2$, and $\alpha_B^{(\ell)}, d_B, y_B \in \mathbb{R}^2$ are the subvectors of $\alpha^{(\ell)}, d, y \in \mathbb{R}^n$, respectively, where only coefficients with indexes in B are considered. The box constraints from (11.46) are removed because it will later be shown that the optimal value does not converge to $-\infty$. One hopes that (11.51)–(11.54) outperforms (11.42)–(11.46), but there seems to be no way that avoids considering all working sets having two elements to apply the second-order approximation algorithm. Hence the following heuristic method (Fan *et al.*, 2005) for an implementation of a second-order approximation is promising because not all possible working sets are considered.

Working Set Selection Algorithm (ALG3)

i) Select $i \in \arg\max_h \big\{-y_h \nabla g(\alpha^{(\ell)})_h : h \in I_{up}(\alpha^{(\ell)})\big\}$.
ii) Consider $\mathrm{Sub}(B)$ as defined in (11.51)–(11.54) and select

$$j \in \arg\min_h \big\{\mathrm{Sub}(\{i, h\}) : h \in I_{low}(\alpha^{(\ell)}), y_h \nabla g(\alpha^{(\ell)})_h > y_i \nabla g(\alpha^{(\ell)})_i\big\}. \tag{11.55}$$

iii) Choose $B := \{i, j\}$ as the working set.

If we use the same index i as in ALG2, we only have to check $\mathcal{O}(n)$ possible candidates for the working set B to choose the index j. An alternative is to choose $j \in \arg M(\alpha^{(\ell)})$ and search for the index i by a way similar to (11.55).

The following theorem shows that one can analytically solve (11.51)–(11.54) such that the working set selection method ALG3 does not cost much more than the algorithm ALG2. Recall that $K = \big(k(x_i, x_j)\big) \in \mathbb{R}^{n\times n}$ denotes the kernel matrix and that $\tilde{K} := \big(y_i y_j k(x_i, x_j)\big) \in \mathbb{R}^{n\times n}$. We denote the (i,j)-elements of these matrices by $K_{i,j}$ and $\tilde{K}_{i,j}$, respectively.

Theorem 11.11. *If L is the hinge loss, $B = \{i, j\}$ is a violating pair, and $b_{i,j} := K_{i,i} + K_{j,j} - 2K_{i,j} > 0$, then (11.51)–(11.54) have the optimal value*

$$-\frac{\big(y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i\big)^2}{2\, b_{i,j}}$$

of the objective function.

Proof. Define $\hat{d}_i := y_i d_i$ and $\hat{d}_j := y_j d_j$. From (11.52), we obtain $\hat{d}_i = -\hat{d}_j$, and the objective function in (11.51) becomes

$$\begin{aligned} &\frac{1}{2}\begin{bmatrix} d_i & d_j \end{bmatrix} \begin{bmatrix} \tilde{K}_{i,i} & \tilde{K}_{i,j} \\ \tilde{K}_{j,i} & \tilde{K}_{j,j} \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} + \begin{bmatrix} \nabla g(\alpha^{(\ell)})_i & \nabla g(\alpha^{(\ell)})_j \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} \\ &= \frac{1}{2}(K_{i,i} + K_{j,j} - 2K_{i,j})\hat{d}_j^2 + \big(y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i\big)\hat{d}_j\,. \end{aligned} \tag{11.56}$$

Since $K_{i,i} + K_{j,j} - 2K_{i,j} > 0$ and B is a violating pair, we define the positive constants

$$b_{i,j} := K_{i,i} + K_{j,j} - 2K_{i,j} \quad \text{and} \quad c_{i,j} := y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i\,. \tag{11.57}$$

Thus, the function in (11.56) has a minimum at

$$\hat{d}_j = -\hat{d}_i = -\frac{c_{i,j}}{b_{i,j}} < 0, \tag{11.58}$$

and the value of the objective function $\mathrm{Sub}(B)$ in (11.51) equals $-c_{i,j}^2/(2b_{i,j})$. Moreover, $\hat{d}_i$ and $\hat{d}_j$ fulfill the constraints (11.53) and (11.54). Indeed, we obtain for the case $j \in I_{low}(\alpha^{(\ell)})$ that $\alpha_j^{(\ell)} = 0$ implies $y_j = -1$ and hence $d_j = y_j \hat{d}_j > 0$. This condition is required by (11.53). The other cases can be treated in a similar way. Thus, $\hat{d}_i$ and $\hat{d}_j$ from (11.58) are optimal for (11.51)–(11.54). □

If the kernel matrix K is strictly positive definite, then the condition $K_{i,i} + K_{j,j} - 2K_{i,j} > 0$ of Theorem 11.11 is valid for any pair $\{i, j\}$ with $i \neq j$. Note that Theorem 11.11 enables us to write (11.55) as

$$j \in \arg\min_h \Big\{ -\frac{c_{i,h}^2}{b_{i,h}} \,:\, h \in I_{low}(\alpha^{(\ell)}),\, y_h \nabla g(\alpha^{(\ell)})_h > y_i \nabla g(\alpha^{(\ell)})_i \Big\},$$

where $b_{i,h}$ and $c_{i,h}$ are the constants defined in (11.57). If K is not strictly positive definite, $b_{i,j} = 0$ can occur. The following algorithm will address this situation.

Note that (11.42)–(11.46) and (11.51)–(11.54) are only used to select the working set B. Hence they do not have to fulfill the feasibility condition $\alpha_i^{(\ell)} + d_i \in [0, C]$ for all $i \in B$. However, feasibility must hold for the subproblem in step *iii)* of ALG1 used to compute $\alpha^{(\ell+1)}$ after the working set is determined.

Now we will consider a general working set algorithm (Chen *et al.*, 2006).

Working Set Selection Algorithm (ALG4)

i) Let $h^* : \mathbb{R} \to \mathbb{R}$ be a measurable function that is strictly increasing on the interval $[0, \infty)$ and fulfills $h^*(0) = 0$ and $h^*(z) \leq z$ for $z \in [0, \infty)$.

ii) Select $B = \{i, j\}$ as the working set if $i \in I_{up}(\alpha^{(\ell)})$, $j \in I_{low}(\alpha^{(\ell)})$, and

$$y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i \geq h^*\big(m(\alpha^{(\ell)}) - M(\alpha^{(\ell)})\big) > 0. \tag{11.59}$$

A special case of ALG4 is obtained by choosing $h(z) := \sigma z$, $z \in \mathbb{R}$, where $\sigma \in (0, 1]$ is fixed. In other words, ALG4 uses in this case a "constant-factor" violating pair as the working set.

Let us now consider the case where the kernel matrix K may not be strictly positive definite; e.g., $b_{i,j} = 0$ can happen for a linear kernel. The following algorithm (Chen *et al.*, 2006) for an SMO-type decomposition for the case of classification based on the hinge loss consists of four steps. The main idea of this approach is to slightly modify the subproblems if necessary (see (11.61)) to obtain an algorithm with nice properties.

Working Set Selection Algorithm (ALG5)

The steps *i)*, *ii)*, and *iv)* are the same as those in ALG1, but step *iii)* is replaced by the following:

iii′) Let τ be a small positive number being constant for all iteration steps, and define

$$\tilde{b}_{i,j} := \tilde{K}_{i,i} + \tilde{K}_{j,j} - 2 y_i y_j \tilde{K}_{i,j}\,. \tag{11.60}$$

If $\tilde{b}_{i,j} > 0$, then solve the subproblem from step *iii)* of the algorithm ALG1 and set $\alpha_B^{(\ell+1)}$ to be the optimal point of this subproblem. If $\tilde{b}_{i,j} \leq 0$, then solve the modified subproblem

$$\begin{aligned} \max_{(\alpha_i, \alpha_j) \in \mathbb{R}^2} \quad & \big(1_B - \tilde{K}_{B,B^c} \alpha_{B^c}^{(\ell)}\big)^{\mathsf{T}} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} - \frac{1}{2} [\alpha_i \;\; \alpha_j] \begin{bmatrix} \tilde{K}_{i,i} & \tilde{K}_{i,j} \\ \tilde{K}_{j,i} & \tilde{K}_{j,j} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ & - \frac{\tau - \tilde{b}_{i,j}}{4} \Big((\alpha_i - \alpha_i^{(\ell)})^2 + (\alpha_j - \alpha_j^{(\ell)})^2 \Big) \\ \text{s.t.} \quad & (\alpha_i, \alpha_j) \in [0, C]^2, \quad \alpha_i y_i + \alpha_j y_j = -\sum\nolimits_{i \in B^c} \alpha_i^{(\ell)} y_i\,, \end{aligned} \tag{11.61}$$

and set $\alpha_B^{(\ell+1)}$ to be the optimal point of this subproblem.

Let us first show how the subproblem from step *iii′)* of ALG2 can be solved. Using $d_i = -d_j$, we obtain

$$\frac{\tau - \tilde{b}_{i,j}}{4} \left\| \alpha_B - \alpha_B^{(\ell)} \right\|^2 = \frac{\tau - \tilde{b}_{i,j}}{2} d_j^2 \,. \tag{11.62}$$

Define

$$\tilde{b}^*_{i,j} := \begin{cases} \tilde{b}_{i,j} & \text{if } \tilde{b}_{i,j} > 0 \\ \tau & \text{otherwise,} \end{cases} \tag{11.63}$$

and

$$\tilde{c}^*_{i,j} := y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i > 0. \tag{11.64}$$

Hence (11.61) is essentially a strictly convex optimization problem of the form

$$\begin{aligned} \min_{d_j} \quad & \frac{1}{2} \tilde{b}^*_{i,j} d_j^2 + \tilde{c}^*_{i,j} d_j \\ \text{s.t.} \quad & c_{low} \le d_j \le c_{up} \,, \end{aligned} \tag{11.65}$$

where $\tilde{b}^*_{i,j}, \tilde{c}^*_{i,j} > 0$, $c_{low} < 0$, and $c_{up} \ge 0$. The optimum of the quadratic objective function is

$$\bar{d}_j \;=\; \max\bigl\{c_{low}, -\tilde{c}^*_{i,j}/\tilde{b}^*_{i,j}\bigr\} \;<\; 0. \tag{11.66}$$

Therefore, once $\tilde{b}^*_{i,j}$ is defined in (11.63), both subproblems can easily be solved no matter whether $\tilde{b}^*_{i,j}$ is positive or not.

Let us now check whether the value of the objective function actually decreases if ℓ increases. Some calculations using $\bar{d}_j < 0$, $\tilde{b}^*_{i,j}\bar{d}_j + \tilde{c}^*_{i,j} \ge 0$ due to (11.66) and $\|\alpha^{(\ell+1)} - \alpha^{(\ell)}\|_2^2 = 2\bar{d}_j^2$ yield that $g(\alpha) - g(\alpha^{(\ell)})$ equals

$$\frac{\bar{d}_j^2}{2}\bigl(\tilde{K}_{i,i} + \tilde{K}_{j,j} - 2y_i y_j \tilde{K}_{i,j}\bigr)^2 + \bigl(y_j \nabla g(\alpha^{(\ell)})_j - y_i \nabla g(\alpha^{(\ell)})_i\bigr) d_j \tag{11.67}$$

and

$$\begin{aligned} g(\alpha^{(\ell+1)}) - g(\alpha^{(\ell)}) = g(\bar{d}_j) = \frac{1}{2}\tilde{b}^*_{i,j}\bar{d}_j^2 + \tilde{c}^*_{i,j}\bar{d}_j \;=\; (\tilde{b}^*_{i,j}\bar{d}_j + \tilde{c}^*_{i,j})\bar{d}_j - \frac{\tilde{b}^*_{i,j}}{2}\,\bar{d}_j^2 \\ \le -\frac{\tilde{b}^*_{i,j}}{2}\,\bar{d}_j^2 \;=\; -\frac{\tilde{b}^*_{i,j}}{4}\left\|\alpha^{(\ell+1)} - \alpha^{(\ell)}\right\|_2^2, \quad \ell \in \mathbb{N}. \end{aligned}$$

We thus obtain the following result.

Lemma 11.12. *Suppose that L is the hinge loss and that the working set $B^{(\ell)}$ in algorithm ALG5 is a violating pair for all $\ell \in \mathbb{N}$, and let $\bigl(\alpha^{(\ell)}\bigr)_{\ell\in\mathbb{N}}$ be the sequence generated by this algorithm. Then*

$$g(\alpha^{(\ell+1)}) < g(\alpha^{(\ell)}) - \delta\left\|\alpha^{(\ell+1)} - \alpha^{(\ell)}\right\|_2^2, \quad \ell \in \mathbb{N}, \tag{11.68}$$

holds with

$$\delta := \frac{1}{4}\min\bigl\{\tau, \min\{\tilde{b}_{i,j} : \tilde{b}_{i,j} > 0\}\bigr\}. \tag{11.69}$$

Theorem 11.13. *Let $L = L_{\text{hinge}}$, and assume that $\tilde{K} \in \mathbb{R}^{n\times n}$ is positive semi-definite, the working set of algorithm ALG5 is a violating pair, and $\tau \leq \frac{2}{C}$. If $\tilde{b}_{i,j} = 0$, then the optimal solution of the subproblem in step iii′) from ALG5 is the same as the optimal solution of the subproblem in step iii) from ALG1.*

Proof. Suppose $\tilde{b}_{i,j} = 0$. From (11.67) and $\tilde{c}^*_{i,j} > 0$, it follows that the subproblem from step *iii)* has the optimum at $\bar{d}_j = c_{low}$. For the subproblem from step *iii′)* and the problem in (11.65), it follows from $\tilde{b}_{i,j} = 0$ that $\tilde{b}^*_{i,j} = \tau$. Since $\tilde{K}$ is positive definite by assumption, we obtain

$$0 = \tilde{b}_{i,j} = \tilde{K}_{i,i} + \tilde{K}_{j,j} - 2y_iy_j\tilde{K}_{i,j} = \|\Phi(x_i) - \Phi(x_j)\|^2_H\,,$$

which gives $\Phi(x_i) = \Phi(x_j)$. Hence $k(x_i, x_h) = k(x_j, x_h)$ for all $h = 1, \ldots, n$ implies

$$\begin{aligned} &y_j\nabla g(\alpha^{(\ell)})_j - y_i\nabla g(\alpha^{(\ell)})_i \\ &= \sum_{h=1}^{n} y_h k(x_j, x_h)\alpha_h^{(\ell)} - y_j - \sum_{h=1}^{n} y_h k(x_i, x_h)\alpha_h^{(\ell)} + y_i \;=\; y_i - y_j\,. \end{aligned}$$

Now $\{i, j\}$ is a violating pair. Hence $y_i - y_j > 0$ implies $\tilde{c}^*_{i,j} = y_i - y_j = 2$. As $\tau \leq \frac{2}{C}$ by assumption, (11.66) implies that $\bar{d}_j = c_{low}$ is the solution for the step *iii′)*. Hence both subproblems have the same optimal point. □

Before we can state a result concerning the asymptotic convergence of ALG5, we need the following result.

Lemma 11.14. *Let L be the hinge loss, and assume that the working set in each iteration of ALG5 is a violating pair. If a subsequence $\left(\alpha^{(\ell)}\right)_{\ell\in J}$, $J \subset \mathbb{N}$, converges to $\bar{\alpha}$, then for any given $s \in \mathbb{N}$, the sequence $\left(\alpha^{(\ell+s)}\right)_{\ell\in J}$ converges to $\bar{\alpha}$ as well.*

Proof. For the subsequence $\left(\alpha^{(\ell+1)}\right)_{\ell\in J}$ from $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ of Lemma 11.12 and the fact that the sequence $\left(g(\alpha^{(\ell)})\right)_{\ell\in\mathbb{N}}$ is bounded and decreasing, we obtain

$$\begin{aligned} \lim_{\ell\in J,\,\ell\to\infty} \|\alpha^{(\ell+1)} - \bar{\alpha}\| \;&\leq\; \lim_{\ell\in J,\,\ell\to\infty} \left(\|\alpha^{(\ell+1)} - \alpha^{(\ell)}\| + \|\alpha^{(\ell)} - \bar{\alpha}\|\right) \\ \leq \lim_{\ell\in J,\,\ell\to\infty} &\left(\delta^{-1/2}\left(g(\alpha^{(\ell)}) - g(\alpha^{(\ell+1)})\right)^{1/2} + \|\alpha^{(\ell)} - \bar{\alpha}\|\right) \;=\; 0. \end{aligned}$$

This yields $\alpha^{(\ell+1)} \to \bar{\alpha}$ for $\ell \to \infty$ and $\ell \in J$. By induction, we obtain the convergence $\alpha^{(\ell+s)} \to \bar{\alpha}$, $\ell \to \infty$, and $\ell \in J$ for any $s \in \mathbb{N}$. □

Note that the feasible region of (11.37) and (11.38) is compact due to Heine-Borel's Theorem A.2.4. Hence there exists a convergent subsequence of $\left(\alpha^{(\ell)}\right)$. The limit point of any convergent subsequence is a stationary point of (11.37) and (11.38), as the following result by Lin (2001) and Chen *et al.* (2006) shows.

Theorem 11.15. *Let L be the hinge loss and $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ be the infinite sequence generated by the SMO-type algorithm ALG5 using ALG4. Then any limit point of $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ is a stationary point of (11.37) and (11.38).*

Proof. Assume that $\bar{\alpha}$ is the limit point of a convergent subsequence $\left(\alpha^{(\ell)}\right)_{\ell\in J}$, $J \subset \mathbb{N}$. If $\bar{\alpha}$ is not a stationary point of (11.37) and (11.38), then it can not satisfy the optimality condition (11.40). Hence a maximal violating pair $(\bar{i}, \bar{j})$ exists such that

$$\bar{i} \in \arg\max_{i\in I_{up}(\alpha)} -y_i \nabla g(\alpha)_i\,, \quad \bar{j} \in \arg\min_{i\in I_{low}(\alpha)} -y_i \nabla g(\alpha)_i \tag{11.70}$$

and

$$\Delta := y_{\bar{j}} \nabla g(\bar{\alpha})_{\bar{j}} - y_{\bar{i}} \nabla g(\bar{\alpha})_{\bar{i}} \;>\; 0. \tag{11.71}$$

Let us further define the positive constant

$$\Delta' = \min\Big\{\Delta, \frac{1}{2}\min\big\{\big|y_s \nabla g(\bar{\alpha})_s - y_t \nabla g(\bar{\alpha})_t\big| : y_s \nabla g(\bar{\alpha})_s \neq y_t \nabla g(\bar{\alpha})_t\big\}\Big\}. \tag{11.72}$$

Lemma 11.14, the continuity of $\nabla g(\alpha)$, and $h^*(\Delta'/2) > 0$ imply that, for any given positive integer r, there exists an index $\bar{\ell} \in J$ such that, for all $\ell \in J$ with $\ell \geq \bar{\ell}$, the following relationships are valid (we will only give details for the derivation of (11.73) and (11.79), because the proofs to show the validity of (11.74)–(11.78) are somewhat similar):

$$y_{\bar{j}} \nabla g(\alpha^{(\ell+u)})_{\bar{j}} - y_{\bar{i}} \nabla g(\alpha^{(\ell+u)})_{\bar{i}} > \Delta' \qquad \text{if } u = 0, \ldots, r, \tag{11.73}$$

$$i \in I_{up}(\alpha^{(\ell)}), \ldots, i \in I_{up}(\alpha^{(\ell+r)}) \quad \text{if } i \in I_{up}(\bar{\alpha}), \tag{11.74}$$

$$i \in I_{low}(\alpha^{(\ell)}), \ldots, i \in I_{low}(\alpha^{(\ell+r)}) \qquad \text{if } i \in I_{low}(\bar{\alpha}), \tag{11.75}$$

$$\begin{gathered} y_j \nabla g(\alpha^{(\ell+u)})_j - \tfrac{\Delta'}{\sqrt{2}} > y_i \nabla g(\alpha^{(\ell+u)})_i \text{ for } u = 0, \ldots, r, \\ \text{if } y_j \nabla g(\bar{\alpha})_j > y_i \nabla g(\bar{\alpha})_i\,, \end{gathered} \tag{11.76}$$

$$\begin{gathered} \big|y_j \nabla g(\alpha^{(\ell+u)})_j - y_i \nabla g(\alpha^{(\ell+u)})_i\big| < h(\Delta') \text{ for } u = 0, \ldots, r, \\ \text{if } y_j \nabla g(\bar{\alpha})_j = y_i \nabla g(\bar{\alpha})_i\,, \end{gathered} \tag{11.77}$$

$$\begin{gathered} (\tau - \hat{b}_{i,j})\|\alpha^{(\ell+u+1)} - \alpha^{(\ell+u)}\| \leq \Delta' \text{ for } u = 0, \ldots, r-1, \text{ where} \\ \hat{b}_{i,j} := \min\{\tilde{K}_{i,i} + \tilde{K}_{j,j} - 2y_i y_j \tilde{K}_{i,j} : \tilde{K}_{i,i} + \tilde{K}_{j,j} - 2y_i y_j \tilde{K}_{i,j} < 0\}, \end{gathered} \tag{11.78}$$

$$\begin{gathered} i \notin I_{up}(\alpha^{(\ell+u+1)}) \text{ or } j \notin I_{low}(\alpha^{(\ell+u+1)}), \\ \text{if } y_j \nabla g(\bar{\alpha})_j > y_i \nabla g(\bar{\alpha})_i \text{ and } \{i, j\} \text{ is the working set at the} \\ (\ell+u)\text{-iteration for } u = 0, \ldots, r-1. \end{gathered} \tag{11.79}$$

We will now derive (11.73). Lemma 11.14 shows that the sequences $\left(\alpha^{(\ell+u)}\right)_{\ell\in J}$ for $u = 0, \ldots, r$ all converge to $\bar{\alpha}$. Now we use the continuity of $\nabla g(\alpha)$ and

(11.72) to obtain for any fixed $u \in \{0,\dots,r\}$ and the corresponding subsequence $\left(\alpha^{(\ell+u)}\right)_{\ell\in J}$ the existence of a constant $k_u \in \mathbb{N}$ such that (11.73) is satisfied for all values $k \in J$ with $k \geq k_u$. Define $\bar{\ell} := \max\{k_u : u \in \{0,\dots,r\}\}$. As r is finite, the validity of (11.73) follows.

Let us now consider (11.79). Similar to (11.40) for the problem (11.37) and (11.38), we obtain for the subproblem at the $(\ell+u)$-th iteration, if the dual subproblem (11.61) is considered and α_B is a stationary point, that

$$\begin{aligned}&\max_{t\in I_{up}(\alpha_B)} -y_t\left(\nabla g\left(\begin{bmatrix}\alpha_B\\ \alpha_{B^c}^{(\ell+u)}\end{bmatrix}\right)\right)_t - \frac{y_t(\tau-\tilde{b}_{i,j})}{2}\big(\alpha_t-\alpha_t^{(\ell)}\big)\\ \leq &\min_{t\in I_{low}(\alpha_B)} -y_t\left(\nabla g\left(\begin{bmatrix}\alpha_B\\ \alpha_{B^c}^{(\ell+u)}\end{bmatrix}\right)\right)_t - \frac{y_t(\tau-\tilde{b}_{i,j})}{2}\big(\alpha_t-\alpha_t^{(\ell)}\big)\,.\end{aligned}$$

Now $B=\{i,j\}$ and $\alpha_B^{(\ell+u+1)}$ is a stationary point of the subproblem that fulfills the inequality above. Suppose that

$$i \in I_{up}(\alpha^{(\ell+u+1)}) \quad \text{and} \quad j \in I_{low}(\alpha^{(\ell+u+1)}).$$

Then it follows from (11.78) and (11.62) that

$$\begin{aligned}&y_i\nabla g(\alpha^{(\ell+u+1)})_i\\ \geq\; &y_j\nabla g(\alpha^{(\ell+u+1)})_j - \frac{y_i(\tau-\tilde{b}_{i,j})}{2}\big(\alpha_i^{(\ell+u+1)}-\alpha_i^{(\ell+u)}\big)\\ &+\frac{y_j(\tau-\tilde{b}_{i,j})}{2}\big(\alpha_j^{(\ell+u+1)}-\alpha_j^{(\ell+u)}\big)\\ \geq\; &y_i\nabla g(\alpha^{(\ell)})_i - \frac{\tau-\tilde{b}_{i,j}}{\sqrt{2}}\left\|\alpha^{(\ell+u+1)}-\alpha^{(\ell+u)}\right\|\\ \geq\; &y_j\nabla g(\alpha^{(\ell)})_j - \frac{\Delta'}{\sqrt{2}}\,.\end{aligned} \tag{11.80}$$

However, this is a contradiction to (11.76) because $y_j\nabla g(\bar{\alpha})_j > y_i\nabla g(\bar{\alpha})_i$ implies (11.76) for $\alpha^{(\ell+u+1)}$. If $\tilde{b}_{i,j}>0$ and the subproblem of step *iii)* of the algorithm ALG1 is considered, then (11.80) has no term $\Delta'/\sqrt{2}>0$ which immediately gives the desired contradiction.

For notational convenience, let us now reorder the indexes of $\bar{\alpha}$ such that

$$y_1\nabla g(\bar{\alpha})_1 \geq \ldots \geq y_n\nabla g(\bar{\alpha})_n\,. \tag{11.81}$$

Further, we define

$$S_{up}(\ell) := \sum_{i\in I_{up}(\alpha^{(\ell)})} i \quad \text{and} \quad S_{low}(\ell) := \sum_{i\in I_{low}(\alpha^{(\ell)})} (n-i), \tag{11.82}$$

which gives

$$n \le S_{low}(\ell) + S_{up}(\ell) \le n(n-1). \tag{11.83}$$

Fix $u \in \{0, \ldots, r\}$. If the pair $\{i, j\}$ is selected at the $(\ell+u)$-th iteration, then it will be shown that

$$y_j \nabla g(\bar{\alpha})_j > y_i \nabla g(\bar{\alpha})_i \,. \tag{11.84}$$

Note that $y_j \nabla g(\bar{\alpha})_j < y_i \nabla g(\bar{\alpha})_i$ is impossible because $y_j \nabla g(\alpha^{(\ell+u)})_j < y_i \nabla g(\alpha^{(\ell+u)})_i$ from (11.76) then violates (11.59). Equality in (11.84) would give

$$\begin{aligned} & y_j \nabla g(\alpha^{(\ell+u)})_j - y_i \nabla g(\alpha^{(\ell+u)})_i & (11.85) \\ & < h^*(\Delta') < h^*\big(y_{\bar{j}} \nabla g(\alpha^{(\ell+u)})_{\bar{j}} - y_{\bar{i}} \nabla g(\alpha^{(\ell+u)})_{\bar{i}}\big) \\ & \le h^*\big(m(\alpha^{(\ell+u)}) - M(\alpha^{(\ell+u)})\big). & (11.86) \end{aligned}$$

Here we used that h^* is strictly increasing and (11.73) and (11.77) to obtain the first two inequalities, and the last inequality in (11.86) results from $\bar{i} \in I_{up}(\bar{\alpha})$, $\bar{j} \in I_{low}(\bar{\alpha})$, (11.74), and (11.75). However, (11.86) is a contradiction to (11.59), which shows that (11.84) is valid.

Now we will use a counting procedure that gives a contradiction to (11.70) and (11.71). Combining (11.84) and (11.79) shows that $i \notin I_{up}(\alpha^{(\ell+1)})$ or $j \notin I_{low}(\alpha^{(\ell+1)})$ if we consider the iteration step from ℓ to $\ell+1$. If $i \notin I_{up}(\alpha^{(\ell+1)})$, then (11.74) implies $i \notin I_{up}(\bar{\alpha})$ and hence $i \in I_{low}(\bar{\alpha})$. From (11.75) and the rule (11.59) to select the working set, it follows that $i \in I_{low}(\alpha^{(\ell)}) \cap I_{up}(\alpha^{(\ell)})$. Thus we have

$$i \in I_{low}(\alpha^{(\ell)}) \cap I_{up}(\alpha^{(\ell)}) \quad \text{and} \quad i \notin I_{up}(\alpha^{(\ell+1)}).$$

Note that $j - i \le -1$ due to (11.81). Therefore, we obtain with $j \in I_{low}(\alpha^{(\ell)})$ the inequalities

$$S_{up}(\ell+1) \le S_{up}(\ell) + j - i \le S_{up}(\ell) - 1 \;\text{ and }\; S_{low}(\ell+1) \le S_{low}(\ell). \tag{11.87}$$

In a similar manner, we get for the case $j \notin I_{low}(\alpha^{(\ell+1)})$ that

$$j \in I_{low}(\alpha^{(\ell)}) \cap I_{up}(\alpha^{(\ell)}) \quad \text{and} \quad j \notin I_{low}(\alpha^{(\ell+1)}).$$

For $i \in I_{up}(\alpha^{(\ell)})$, we have that

$$S_{up}(\ell+1) \le S_{up}(\ell) \text{ and } S_{low}(\ell+1) \le S_{low}(\ell) + (\ell - i) - (\ell - j) \le S_{low}(\ell) - 1. \tag{11.88}$$

The same arguments can be used to go from iteration step $(\ell+1)$ to $(\ell+2)$ because (11.79) can be used, as (11.84) holds for working sets selected during the iteration steps ℓ to $\ell + r$. Now (11.87) and (11.88) show that the term $S_{low}(\ell) + S_{up}(\ell)$ can be reduced to zero in $r := n(n-1)$ iterations, which gives the desired contradiction to (11.83). Hence, the assumptions (11.70) and (11.71) were wrong, which gives the assertion. □

The next result (Chen *et al.*, 2006) shows that the sequence $\big(\alpha^{(\ell)}\big)$ is even globally convergent under mild conditions.

Corollary 11.16. *Let L be the hinge loss function. If the matrix $\tilde{K}$ is strictly positive definite, then $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ globally converges to the unique maximum of the dual problem (11.37) and (11.38).*

Proof. Since $\tilde{K}$ is strictly positive definite, there exists a unique solution, say $\bar{\alpha}$, of the dual problem (11.37)-(11.38). Suppose that the sequence $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ does not globally converge to $\bar{\alpha}$. Then there exists a constant $\varepsilon > 0$ and an infinite subset $J \subset \mathbb{N}$ such that $\|\alpha^{(\ell)} - \bar{\alpha}\| \geq \varepsilon$ for all $\ell \in J$. Since $\{\alpha^{(\ell)} : \ell \in J\}$ is in a compact set, there exists a further subsequence that converges to some point, say α^*, with $\|\alpha^* - \bar{\alpha}\| \geq \varepsilon$. We know by Theorem 11.15 that α^* is an optimal solution of (11.37)-(11.38). This gives the desired contradiction because $\bar{\alpha}$ is the unique global maximum. □

The preceding corollary is quite useful for practical purposes, and we would like to give an example. Suppose that we consider a data set with $x_i \neq x_j$ for all $1 \leq i < j \leq n$. Further, let us assume that a Gaussian RBF kernel is used. Then the matrix $\tilde{K}$ is strictly positive definite and the sequence $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ converges globally to the unique maximum of the dual problem.

The following result (Chen *et al.*, 2006) shows that one can improve Theorem 11.15 if the matrix $\tilde{K}$ is positive definite.

Theorem 11.17. *Let L be the hinge loss, and $\tilde{K}$ be positive definite.*

i) If $\bar{\alpha} \neq \hat{\alpha}$ are any two optimal solutions of (11.37) and (11.38), then

$$y_i\big(\nabla g(\bar{\alpha})\big)_i = y_i\big(\nabla g(\hat{\alpha})\big)_i, \quad i = 1, \ldots, n, \tag{11.89}$$

and

$$m(\bar{\alpha}) = M(\bar{\alpha}) = m(\hat{\alpha}) = M(\hat{\alpha}). \tag{11.90}$$

ii) If there is an optimal solution $\bar{\alpha}$ satisfying $m(\bar{\alpha}) < M(\bar{\alpha})$, then $\bar{\alpha}$ is the unique optimal solution of (11.37) and (11.38).

iii) The following set is independent of any optimal solution $\bar{\alpha}$:

$$I := \{i \in \{1, \ldots, n\} : -y_i\big(\nabla g(\bar{\alpha})\big)_i > M(\bar{\alpha}) \;\text{ or }\; -y_i\big(\nabla g(\bar{\alpha})\big)_i < m(\bar{\alpha})\}. \tag{11.91}$$

Moreover, the problem (11.37) and (11.38) has a unique and bounded optimal solution at α_i, $i \in I$.

Proof. Since $\tilde{K} \in \mathbb{R}^{n\times n}$ is positive definite, the problem (11.37) and (11.38) is a convex programming problem and $\bar{\alpha}$ and $\hat{\alpha}$ are both global optima. Then

$$g(\bar{\alpha}) = g(\hat{\alpha}) = g(\delta\bar{\alpha} + (1-\delta)\hat{\alpha}), \qquad \text{for all } \delta \in [0,1],$$

implies

$$(\bar{\alpha} - \hat{\alpha})^{\mathsf{T}} \tilde{K} (\bar{\alpha} - \hat{\alpha}) = 0.$$

Now let us factorize $\tilde{K} = UU^{\mathsf{T}}$, which is possible because $\tilde{K}$ is positive definite. We obtain $\|U^{\mathsf{T}}(\bar{\alpha} - \hat{\alpha})\| = 0$ and hence $\tilde{K}\bar{\alpha} = \tilde{K}\hat{\alpha}$. From this we obtain (11.89).

To prove (11.90), we will show that

$$m(\hat{\alpha}) \geq M(\bar{\alpha}) \quad \text{and} \quad m(\bar{\alpha}) \geq M(\hat{\alpha}). \tag{11.92}$$

With the optimality conditions $M(\bar{\alpha}) \geq m(\bar{\alpha})$ and $M(\hat{\alpha}) \geq m(\hat{\alpha})$, the equalities in (11.90) hold. Due to symmetry, it is sufficient to prove the first case of (11.92). If it is false, then $m(\hat{\alpha}) < M(\bar{\alpha})$. We then investigate different cases by comparing $-y_i \nabla g(\bar{\alpha})_i$ with $M(\bar{\alpha})$ and $m(\hat{\alpha})$. If $m(\hat{\alpha}) < M(\bar{\alpha}) \leq -y_i \nabla g(\bar{\alpha})_i$, then $i \notin I_{up}(\hat{\alpha})$ and

$$\hat{\alpha}_i = \begin{cases} 0 & \text{if } y_i = -1 \\ C & \text{if } y_i = +1. \end{cases} \tag{11.93}$$

With $0 \leq \bar{\alpha}_i \leq C$, we obtain

$$y_i(\hat{\alpha}_i - \bar{\alpha}_i) \geq 0. \tag{11.94}$$

If $M(\bar{\alpha}) > m(\hat{\alpha}) \geq -y_i \nabla g(\bar{\alpha})_i$, then $i \notin I_{low}(\bar{\alpha})$ and

$$\bar{\alpha}_i = \begin{cases} C & \text{if } y_i = -1 \\ 0 & \text{if } y_i = +1, \end{cases} \tag{11.95}$$

and (11.94) still holds.

The other indexes are in the set

$$S := \big\{ i \,:\, m(\hat{\alpha}) < -y_i \nabla g(\hat{\alpha})_i = -y_i \nabla g(\bar{\alpha})_i < M(\bar{\alpha}) \big\}.$$

If $i \in S$, then $i \notin I_{up}(\hat{\alpha})$ and $i \notin I_{low}(\bar{\alpha})$. Hence (11.93) and (11.95) yield

$$y_i(\hat{\alpha}_i - \bar{\alpha}_i) = C, \tag{11.96}$$

and thus

$$0 = \sum_{i=1}^{n} y_i \hat{\alpha}_i - \sum_{i=1}^{n} y_i \bar{\alpha}_i = \sum_{i \notin S} y_i(\hat{\alpha}_i - \bar{\alpha}_i) + C|S|.$$

As $C > 0$ and (11.94) implies that each term in the sum above is non-negative, we obtain $|S| = 0$ and $\hat{\alpha}_i = \bar{\alpha}_i$ for all $i \notin S$. Thus, $\bar{\alpha} = \hat{\alpha}$. However, this is a contradiction to the assumption that $\bar{\alpha}$ and $\hat{\alpha}$ are different optimal solutions. Therefore, $m(\hat{\alpha}) < M(\bar{\alpha})$ is wrong and we obtain $m(\hat{\alpha}) \geq M(\bar{\alpha})$ in (11.92), which completes the proof of (11.90).

The second result of the theorem and the validity of the set I follow from (11.90). Moreover, the set I is independent of any optimal solution.

Assume that α is an optimal vector. If $i \in I$ and $m(\alpha) \leq M(\alpha) < -y_i \nabla g(\alpha)_i$, then $i \notin I_{up}(\alpha)$ and α_i is the same as $\hat{\alpha}_i$ in (11.93). Hence, the optimal coefficient α_i is unique and bounded. The case $m(\alpha) > -y_i \nabla g(\alpha)_i$ can be treated in a similar manner. □

Since in general the decomposition method approaches an optimum only after an infinite number of iteration steps, there is a need to specify *stopping criteria* to stop the iteration procedure after a finite number of steps. In general, it is not wise to specify in advance the number of steps to be carried out by the decomposition method because it is unknown how well the approximation of the optimum will be. In general, however, it can be useful to define an upper bound for the number of iteration steps or for the computation time and print an error message if the stopping criteria were not satisfied.

One possible stopping criterion is to specify in advance a small tolerance value, say $\varepsilon > 0$, and stop the iteration process if

$$m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) \leq \varepsilon.$$

This stopping condition is quite plausible and commonly used due to its closeness to the optimality condition (11.40). The following result (Chen *et al.*, 2006) shows that this stopping condition can actually be achieved in a finite number of iteration steps.

Theorem 11.18. *Let L be the hinge loss, $\tilde{K} \in \mathbb{R}^{n\times n}$ be positive definite, and suppose that the SMO-type decomposition method ALG5 using ALG4 generates an infinite sequence $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$. Then*

$$\lim_{\ell\to\infty} m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) = 0. \tag{11.97}$$

Proof. Let us assume that the convergence in (11.97) is wrong. Then there exists an infinite set $\bar{J}$ and a constant $\Delta > 0$ such that

$$\left|m(\alpha^{(\ell)}) - M(\alpha^{(\ell)})\right| \geq \Delta, \quad k \in \bar{J}. \tag{11.98}$$

In the SMO-decomposition method we have $m(\alpha^{(\ell)}) > M(\alpha^{(\ell)})$ for all $\ell \in \mathbb{N}$, and hence (11.98) can be rewritten as

$$m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) \geq \Delta, \quad k \in \bar{J}. \tag{11.99}$$

In the set $\bar{J}$, there exists an infinite subset J such that

$$\lim_{\ell\in J,\, \ell\to\infty} \alpha^{(\ell)} = \bar{\alpha}.$$

Using the assumption that $\tilde{K}$ is positive definite, we obtain the global convergence of $\nabla g(\alpha^{(\ell)})$ by Theorem 11.17

$$\lim_{\ell\to\infty} \nabla g(\alpha^{(\ell)})_i = \nabla g(\bar{\alpha})_i \quad i = 1, \ldots, n. \tag{11.100}$$

Now we will use a counting approach similar to that of Theorem 11.15. First, rewrite (11.99) as

$$m(\alpha^{(\ell)}) \geq M(\alpha^{(\ell)}) + \Delta', \quad k \in \bar{J}, \tag{11.101}$$

where

$$\Delta' := \min\Big\{\Delta, \frac{1}{2}\min\big\{|y_s\nabla g(\bar{\alpha})_s - y_t\nabla g(\bar{\alpha})_t| : y_s\nabla g(\bar{\alpha})_s \neq y_t\nabla g(\bar{\alpha})_t\big\}\Big\} > 0. \tag{11.102}$$

We still require (11.73)–(11.79) but use (11.100) and the definition of Δ' in (11.102) to extend (11.76) and (11.77) for all $\ell \geq \bar{\ell}$ (i.e., not only for $\ell \in J$):

$$y_t\nabla g(\alpha^{(\ell)})_t < y_s\nabla g(\alpha^{(\ell)})_s \quad \text{if } y_t\nabla g(\bar{\alpha})_t < y_s\nabla g(\bar{\alpha})_s\,, \tag{11.103}$$

$$|y_s\nabla g(\alpha^{(\ell)})_s - y_t\nabla g(\alpha^{(\ell)})_t| > \Delta' \quad \text{if } y_t\nabla g(\bar{\alpha})_t \neq y_s\nabla g(\bar{\alpha})_s\,, \tag{11.104}$$

$$|y_s\nabla g(\alpha^{(\ell)})_s - y_t\nabla g(\alpha^{(\ell)})_t| < h^*(\Delta') \quad \text{if } y_t\nabla g(\bar{\alpha})_t = y_s\nabla g(\bar{\alpha})_s\,. \tag{11.105}$$

Then the proof follows Theorem 11.15 except (11.86), in which we need $m(\alpha^{(\ell+u)}) - M(\alpha^{(\ell+u)}) \geq \Delta'$ for all $u \in \{0, \ldots, r\}$. This condition does not follow from (11.101), which holds only for a subsequence. Therefore, our goal is to prove

$$m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) \geq \Delta', \quad \ell \geq \bar{\ell}. \tag{11.106}$$

Assume that there is a positive integer $\ell' \geq \bar{\ell}$ such that $m(\alpha^{(\ell')}) - M(\alpha^{(\ell')}) \in (0, \Delta')$ and that $\{i, j\}$ is the working set at this iteration step. Because $i \in I_{up}(\alpha^{(\ell')})$ and $j \in I_{low}(\alpha^{(\ell')})$ from the selection rule, we have

$$M(\alpha^{(\ell')}) \leq -y_j\nabla g(\alpha^{(\ell')})_j < -y_i\nabla g(\alpha^{(\ell')})_i \leq m(\alpha^{(\ell')}). \tag{11.107}$$

Using (11.104), we obtain that the set $\{i, j\}$ and indexes achieving $m(\alpha^{(\ell')})$ and $M(\alpha^{(\ell')})$ have the same value of $y_t\nabla g(\bar{\alpha})_t$ and are all from the set

$$\big\{t : y_t\nabla g(\bar{\alpha})_t = y_i\nabla g(\bar{\alpha})_i = y_j\nabla g(\bar{\alpha})_j\big\}. \tag{11.108}$$

Note that for elements not in this set, (11.103), (11.104), and (11.107) yield

$$\begin{aligned} &y_t\nabla g(\bar{\alpha})_t < y_i\nabla g(\bar{\alpha})_i \;\text{ implies} \\ &-y_t\nabla g(\alpha^{(\ell')})_t > -y_i\nabla g(\alpha^{(\ell')})_i + \Delta' > m(\alpha^{(\ell')}) \text{ and } t \notin I_{up}(\alpha^{(\ell')}). \end{aligned} \tag{11.109}$$

In a similar way, we obtain that

$$y_t\nabla g(\bar{\alpha})_t > y_i\nabla g(\bar{\alpha})_i \;\text{ implies }\; t \notin I_{low}(\alpha^{(\ell')}). \tag{11.110}$$

Because we have shown that the working set is from the set given in (11.108), other coefficients remain the same from iteration step ℓ' to $\ell' + 1$. Hence, indexes satisfying (11.109) and (11.110) fulfill $t \notin I_{up}(\alpha^{(\ell'+1)})$ and $t \notin I_{low}(\alpha^{(\ell'+1)})$, respectively. Furthermore, indexes in (11.109) have larger values of $-y_t\nabla g(\alpha^{(\ell'+1)})_t$ than others due to (11.103). Hence their values of $-y_t\nabla g(\alpha^{(\ell'+1)})_t$ are greater than $m(\alpha^{(\ell'+1)})$. Similarly, components in (11.110) are smaller than $M(\alpha^{(\ell'+1)})$. Using $m(\alpha^{(\ell'+1)}) > M(\alpha^{(\ell'+1)})$, we see that indexes that achieve $m(\alpha^{(\ell'+1)})$ and $M(\alpha^{(\ell'+1)})$ are again from the set in (11.108), and this is true for all $\ell \geq \ell'$. Now, by (11.105) and the conditions on h^*, we obtain

$$m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) < h^*(\Delta') \leq \Delta', \qquad \ell \geq \ell'.$$

This is the desired contradiction to (11.101), and thus (11.106) holds. □

One can argue that a main advantage of decomposition methods is to allow the computation of $f_{\mathrm{D},\lambda}$ even for large sample sizes n. However, the actual computation time and the number of necessary iteration steps until the stopping conditions are fulfilled can be quite large. Therefore, computational techniques to speed up the computation time or to decrease the number of iteration steps are desirable. Among such techniques, *shrinking* and *caching* have been shown to be successful for SVMs.

Shrinking is based on the idea that if an index $\alpha_i^{(\ell)}$ remains equal to 0 or to C for many iteration steps then it may stay at this value. The size of the optimization problem is reduced in shrinking algorithms without considering some bounded Lagrange multipliers. This has the advantage that the decomposition method then works on a smaller problem and hence a considerable reduction of CPU time is sometimes possible. In addition, less memory is used. Afterward, we have to add shrunken components back and must check whether an optimal solution of the original problem is obtained.

Besides shrinking, a caching strategy can also be helpful to speed up the computation of $f_{\mathrm{D},\lambda}$ for large sample sizes. Since $\tilde{K} \in \mathbb{R}^{n\times n}$ may then be too large to be stored into the RAM of the computer, the elements of $\tilde{K}$ are calculated when they are needed. The idea is to use the cache and the RAM of the computer (which allows relatively fast access to objects in it) to store recently used elements $\tilde{K}_{i,j}$. If in the final iterations only a small subset of columns of $\tilde{K}$ are actually needed and if the cache contains them, the computation of many kernel terms $k(x_i, x_j)$ becomes superfluous. Of course, this is especially interesting for SVMs having sparse solutions (i.e., if many Lagrange multipliers are equal to 0). This is often true for SVMs based on the hinge loss or on the ϵ-insensitive loss function.

The following result was shown by Chen *et al.* (2006).

Theorem 11.19. *Let L be the hinge loss and $\tilde{K} \in \mathbb{R}^{n\times n}$ be positive definite, and assume the SMO-type decomposition method ALG5 using ALG4. Let I be the set of indexes defined in (11.91).*

i) There exists an $\bar{\ell} \in \mathbb{N}$ such that, after $\ell > \bar{\ell}$ iteration steps, every Lagrange multiplier $\alpha_i^{(\ell)}$, $i \in I$, has reached the unique and bounded optimal solution. It remains the same in all subsequent iterations, and $i \in I$ is not *an element of the set*

$$\big\{t \in \{1,\ldots,n\} : M(\alpha^{(\ell)}) \leq -y_t \nabla g(\alpha^{(\ell)})_t \leq m(\alpha^{(\ell)})\big\}. \tag{11.111}$$

ii) If (i) has an optimal solution $\bar{\alpha}$ satisfying $m(\bar{\alpha}) < M(\bar{\alpha})$, then $\bar{\alpha}$ is the unique solution and the decomposition method reaches it in a finite *number of iterations.*

iii) If $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ *is an infinite sequence, then the following two limits exist and are equal:*

$$\lim_{\ell\to\infty} m(\alpha^{(\ell)}) = m(\bar{\alpha}) = \lim_{\ell\to\infty} M(\alpha^{(\ell)}) = M(\bar{\alpha}), \tag{11.112}$$

where $\bar{\alpha}$ *is any optimal solution.*

Proof. i). Suppose that the assertion is wrong. Then there exist an index $\bar{i} \in I$ and an infinite set $\hat{J} \subset \mathbb{N}$ such that

$$\alpha_{\bar{i}}^{(\ell)} \neq \hat{\alpha}_{\bar{i}}\,, \quad \ell \in \hat{J}, \tag{11.113}$$

where $\hat{\alpha}_{\bar{i}}$ is the i-th coefficient of the unique optimal solution according to Theorem 11.17. Using Theorem 11.15, there is a set $J \subset \hat{J}$ such that

$$\lim_{\ell\in J,\,\ell\to\infty} \alpha^{(\ell)} = \bar{\alpha} \tag{11.114}$$

is a stationary point. Furthermore, Theorem 11.17 implies that $\bar{\alpha}_{\bar{i}} = \hat{\alpha}_{\bar{i}}$ for $i \in I$; i.e., these coefficients are optimal and unique.

As $\bar{i} \in I$, let us first consider the case

$$M(\bar{\alpha}) < -y_{\bar{i}} \nabla g(\bar{\alpha})_{\bar{i}}\,. \tag{11.115}$$

In this situation, we have $\bar{i} \in I_{up}(\bar{\alpha})$, and (11.113) implies

$$\bar{i} \in I_{up}(\alpha^{(\ell)})\,, \quad \ell \in J. \tag{11.116}$$

For each index $j \in \arg M(\bar{\alpha})$, we have $j \in I_{low}(\bar{\alpha})$. It follows from (11.114) that there an integer $\bar{\ell} \in \mathbb{N}$ such that

$$j \in I_{low}(\alpha^{(\ell)})\,, \quad \ell \in J, \ell \geq \bar{\ell}. \tag{11.117}$$

Thus, (11.116) and (11.117) imply

$$m(\alpha^{(\ell)}) - M(\alpha^{(\ell)}) \geq y_j \nabla g(\alpha^{(\ell)})_j - y_{\bar{i}} \nabla g(\alpha^{(\ell)})_{\bar{i}}\,, \quad \ell \in J, \ell \geq \bar{\ell}. \tag{11.118}$$

Now, by (11.114), the continuity of $\nabla g(\alpha)$, and (11.97), and computing the limit on both sides of (11.118), we obtain

$$0 \geq y_j \nabla g(\bar{\alpha})_j - y_{\bar{i}} \nabla g(\bar{\alpha})_{\bar{i}} = -M(\bar{\alpha}) - y_{\bar{i}} \nabla g(\bar{\alpha})_{\bar{i}}\,.$$

However, this inequality violates the inequality in (11.115) which gives the desired contradiction. The proof for the case $m(\bar{\alpha}) > -y_{\bar{i}} \nabla g(\bar{\alpha})_{\bar{i}}$ is similar.

ii). Let us again assume that the assertion is false. Then $\left(\alpha^{(\ell)}\right)_{\ell\in\mathbb{N}}$ is an infinite sequence. It follows from Theorems 11.15 and 11.17 that $\bar{\alpha}$ is the unique optimal solution and $\alpha^{(\ell)}$ globally converges to $\bar{\alpha}$ if $\ell \to \infty$. Define the index sets

$$I_1 := \left\{ i \in \{1, \ldots, n\} : M(\bar{\alpha}) = -y_i \nabla g(\bar{\alpha})_i \right\},$$
$$I_2 := \left\{ i \in \{1, \ldots, n\} : m(\bar{\alpha}) = -y_i \nabla g(\bar{\alpha})_i \right\}.$$

Using part *i)* of the theorem, we see that $\arg m(\alpha^{(\ell)}) \subset I_1 \cup I_2$ and $\arg M(\alpha^{(\ell)}) \subset I_1 \cup I_2$ provided ℓ is sufficiently large. Now, by (11.97), the continuity of $\nabla g(\alpha)$, and the convergence $\lim_{\ell \to \infty} \alpha^{(\ell)} = \bar{\alpha}$, there exists an integer $\bar{\ell} \in \mathbb{N}$ such that for all $\ell \geq \bar{\ell}$

$$\arg m(\alpha^{(\ell)}) \cup \arg M(\alpha^{(\ell)}) \subset I_1 \quad \text{or} \quad \arg m(\alpha^{(\ell)}) \cup \arg M(\alpha^{(\ell)}) \subset I_2 \,. \tag{11.119}$$

Suppose that $\arg m(\alpha^{(\ell)}) \cup \arg M(\alpha^{(\ell)}) \subset I_1$ at the ℓ-th iteration. Then we can use the same argument as in (11.107) and (11.108) to obtain that the working set B is a subset of I_1. The decomposition method maintains feasibility, thus

$$\sum_{i \in B} y_i \alpha_i^{(\ell)} = \sum_{i \in B} y_i \alpha_i^{(\ell+1)} \,. \tag{11.120}$$

From $B \subset I_1$ and the assumption that $m(\bar{\alpha}) < M(\bar{\alpha})$, every $\bar{\alpha}_i$, $i \in B$, satisfies $i \notin I_{up}(\alpha)$. Hence $\bar{\alpha}_i = \hat{\alpha}_i = 0$, if $y_i = -1$ and $i \in B$, and $\bar{\alpha}_i = \hat{\alpha}_i = C$, if $y_i = +1$ and $i \in B$. If we combine this with (11.120), we obtain

$$\begin{aligned}
&\|\alpha^{(\ell+1)} - \bar{\alpha}\|_1 \\
&= \sum_{i \notin B} |\alpha_i^{(\ell+1)} - \bar{\alpha}_i| + \sum_{i \in B,\, y_i = +1} (C - \alpha_i^{(\ell+1)}) + \sum_{i \in B,\, y_i = -1} (\alpha_i^{(\ell+1)} - 0) \\
&= \sum_{i \notin B} |\alpha_i^{(\ell)} - \bar{\alpha}_i| + \sum_{i \in B,\, y_i = +1} (C - \alpha_i^{(\ell)}) + \sum_{i \in B,\, y_i = -1} (\alpha_i^{(\ell)} - 0) \\
&= \|\alpha^{(\ell)} - \bar{\alpha}\|_1 \,.
\end{aligned} \tag{11.121}$$

If $\arg m(\alpha^{(\ell)})$ and $\arg M(\alpha^{(\ell)})$ are both subsets of I_2, the equation (11.121) is still valid. Therefore, $0 \neq \|\alpha^{(\ell)} - \bar{\alpha}\|_1 = \|\alpha^{(\ell+r)} - \bar{\alpha}\|_1$, $r \in \mathbb{N}$, which gives the desired contradiction to the fact that $\left(\alpha^{(\ell)}\right)$ converges to $\bar{\alpha}$. Hence the decomposition method stops after a finite number of iteration steps.

iii). Since $\left(\alpha^{(\ell)}\right)_{\ell \in \mathbb{N}}$ is an infinite sequence, using the result of part *ii)* of the theorem, we see that the dual problem (11.37) and (11.38) has no optimal solution $\bar{\alpha}$ with the property $M(\bar{\alpha}) > m(\bar{\alpha})$. Using Theorem 11.17, this yields

$$M(\bar{\alpha}) = m(\bar{\alpha}) = -y_t \nabla g(\bar{\alpha})_t \,, \quad t \notin I. \tag{11.122}$$

Note that this result is valid for any optimal solution $\bar{\alpha}$. Now, part *i)* of the theorem guarantees the existence of $\bar{\ell} \in \mathbb{N}$ such that, for all $\ell \geq \bar{\ell}$, the index $i \in I$ is not an element of the set in (11.111). Therefore, the set in (11.111) is contained in the index set $I' := \{1, \ldots, n\} \backslash I$ and

$$\min_{i \in I'} -y_i \nabla g(\alpha^{(\ell)})_i \leq M(\alpha^{(\ell)}) < m(\alpha^{(\ell)}) \leq \max_{i \in I'} -y_i \nabla g(\alpha^{(\ell)})_i \,. \tag{11.123}$$

Although the sequence $\left(\alpha^{(\ell)}\right)_{\ell \in \mathbb{N}}$ may not be globally convergent, the sequences $\left(-y_i \nabla g(\alpha^{(\ell)})_i\right)_{\ell \in \mathbb{N}}$, $i = 1, \ldots, n$, are according to (11.89). The limits

of both sides of (11.123) are equal due to (11.122). Hence (11.112) follows and the assertion of part *iii)* is shown, which completes the proof. □

The preceding theorem shows, for SVMs based on the hinge loss, that the SMO-type decomposition method involves only indexes from I' in many iteration steps, which makes caching successful for this loss function. Recall that we know from Chapter 8 that the property of the hinge loss function being equal to zero in a whole interval implies that $f_{\mathrm{D},\lambda} = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$ is usually sparse (i.e., many coefficients α_i are equal to 0) and this implies that caching can be effective. This theorem also illustrates two possible shrinking implementations for SVMs based on the hinge loss function. *(i)* Elements not in the set (11.111) are removed. This is done by the software `LIBSVM` (Chang and Lin, 2004). *(ii)* Any α_i that has stayed at the same bound for a certain number of iterations is removed. This strategy is implemented in $\texttt{SVM}^{light}$ (Joachims, 1999). We also refer to Section 11.4 for additional information regarding these software products. Caching and shrinking can probably offer such a big gain in computing $f_{\mathrm{D},\lambda}$ only for loss functions that allow a sparse representation of $f_{\mathrm{D},\lambda}$.

11.3 Determination of Hyperparameters

In this section, we consider some techniques for determining suitable combinations of the hyperparameters for SVMs. There exists a vast body of literature regarding the choice of hyperparameters for SVMs. Here we will only consider a few facets of how to choose such hyperparameters for classification and regression problems.

The quality of the estimator $\mathcal{R}_{L,\mathrm{D}}(f_{\mathrm{D},\lambda})$ for the unknown risk $\mathcal{R}_{L,\mathrm{P}}(f_{\mathrm{P},\lambda})$ and the precision of predictions $f_{\mathrm{D},\lambda}(x)$ for the unknown values $f_{\mathrm{P},\lambda}(x)$ for unseen $x \in X$ critically depend not only on the data set D used for training purposes, the loss function, and the kernel but also on the choice of the hyperparameters such as the regularizing parameter $\lambda > 0$, kernel parameters, and parameters of the loss function. Examples are thus the value of γ for the Gaussian RBF kernel and ϵ used by the ϵ-insensitive loss function in regression. Unfortunately, choosing these hyperparameters in an optimal way usually requires computing $f_{\mathrm{D},\lambda}$ for many combinations of the hyperparameters. In other words, it is necessary to solve not just one convex problem but a series of them. This increases the computational effort for the use of SVMs in practice.

Let us first consider SVMs for regression based on the ϵ-insensitive loss function. There exists a linear relationship between the noise level of $\mathrm{P}(y|x)$ and the optimal value of ϵ for support vector regression using $L = L_{\epsilon\text{-insens}}$ (Smola *et al.*, 1998). Of course, P is unknown; otherwise we would probably not use $L_{\epsilon\text{-insens}}$, but, for example, the maximum likelihood loss $L\big(x, y, f(x)\big) = -\ln p\big(y - f(x)\big)$ if P has a density function p. There exists

a modification of the SVM based on $L_{\epsilon\text{-insens}}$ called *ν-support vector regression* exploiting this relationship. The idea is to modify (11.1) such that the hyperparameter ϵ becomes a variable of the optimization problem including a specific additional term in the primal objective function that attempts to minimize ϵ. For $L_{\epsilon\text{-insens}}$ and the case with an additional offset term $b \in \mathbb{R}$, the problem (11.1) is thus modified to

$$\inf_{f \in H,\, b \in \mathbb{R},\, \epsilon > 0} \mathbb{E}_{\mathrm{D}} L\big(Y, f(X) + b\big) + \lambda \|f\|_H^2 + \nu\epsilon \tag{11.124}$$

for some $\nu > 0$. Define $C = 1/(2n\lambda)$. Then we obtain the equivalent problem

$$\begin{aligned} \min_{\alpha,\, \xi^+,\, \xi^- \in \mathbb{R}^n,\, b \in \mathbb{R},\, \epsilon > 0} \quad & C \sum_{i=1}^n \big(\xi_i^+ + \xi_i^-\big) + \frac{1}{2}\|w(\alpha)\|_H^2 + Cn\nu\epsilon \\ \text{s.t.} \quad & \xi_i^+ \geq 0,\, \xi_i^+ \geq y_i - \langle w(\alpha), \Phi(x_i)\rangle_H - b - \epsilon, \\ & \xi_i^- \geq 0,\, \xi_i^- \geq \langle w(\alpha), \Phi(x_i)\rangle_H + b - y_i - \epsilon, \quad \forall i. \end{aligned}$$

The dual program becomes

$$\begin{aligned} \max_{\alpha^+,\, \alpha^- \in \mathbb{R}^n} \quad & \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) y_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) k(x_i, x_j) \\ \text{s.t.} \quad & \alpha_i^+,\, \alpha_i^- \in [0, C],\, \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0,\, \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) \leq Cn\nu,\ \ \forall i. \end{aligned}$$

A pendant to ν-support vector regression exists for classification problems.

We will now consider the determination of a suitable combination of hyperparameters as an optimization problem and will summarize empirical results when we compare different numerical methods to solve this optimization problem. We will concentrate on classification and regression problems. A reasonable choice of the hyperparameters depends on the criteria used to measure their quality. One useful criterion is the *accuracy*. In classification problems, the accuracy is often measured by the empirical misclassification rate. One can also use the modification of TV-SVM described in Definition 8.20. In regression problems, the empirical L-risk or the empirical L-risk based on a suitable calibrated loss function are often used as accuracy criteria in regression problems.

Note that $f_{\mathrm{D},\lambda} \in H$ and the predictions $\hat{y} = f_{\mathrm{D},\lambda}(x) \in \mathbb{R}$ depend on the hyperparameters. As the derivative of both target functions on the hyperparameters is usually unknown, the optimal parameters have to be found numerically. The following six methods are often used to determine suitable hyperparameters and will be briefly described: random search, grid search, Nelder-Mead search, cross-validation, a heuristic search, and pattern search.

The simplest version of a *random search* can be described as follows. A random point of the parameter space is chosen, and the value of the objective function is evaluated. This is repeated N times, and the best point is taken

as the result (i.e., this point is considered as a suitable choice of the hyperparameters). Of course, the result of this search strongly depends on the chosen random points and on the number of random points. The random points for which the objective function is evaluated can be drawn, for example, from a multivariate normal distribution with the center of the search space as the mean.

Optimization by the *grid search* is also very simple. After the search space (i.e.; the set of all possible combinations of the hyperparameters) is specified, each search dimension is split into n_i parts. Often these splits are equidistant or geometrically distributed. The intersections of the splits—which form a (multi-)dimensional grid—are the trial points for which the objective function is evaluated. The best point is taken as the result. It is possible to use a two-stage grid search. The first grid covers a broad region of the space of possible hyperparameters, but this grid is relatively rough. The best point of the first grid is used as the center of a second and finer grid, and the best point of the second grid is taken as the result. Properties of grid searches are now relatively well investigated. The danger that the algorithm will only find a local optimum far away from the optimum is relatively small, provided the grid covers a broad region and that the grid is fine enough. Of course, searches with a fine grid for large data sets are very time-consuming, if at all possible.

The *Nelder-Mead algorithm* proposed by Nelder and Mead (1965) constructs a simplex of $m+1$ points for an m-dimensional optimization problem. There are variants of the Nelder-Mead algorithm that allow for constraints. For the determination of hyperparameters for SVMs, we typically have $1 \leq m \leq 4$ for classification and regression problems. The functional values are calculated for the vertices of the simplex, and the worst point is reflected through the opposite side of the simplex. If this trial point is best, the new simplex is expanded further out. If the function value is worse, then the second-worst point of the simplex is contracted. If no improvement at all is found, the simplex is shrunken toward the best point. The iteration terminates if the differences in the function values between the best and worst points are smaller than a pre-specified tolerance value. There is the danger that the algorithm will only find a local optimum.

Cross-validation is also a standard technique for finding a suitable set of hyperparameters, especially for small- to moderate-sized data sets. The data set is randomly divided into ℓ (e.g.; $\ell = 10$) disjoint subsets of equal size, and each subset is used once as a validation set, whereas the other $\ell - 1$ sets are put together to form a training set. In the simplest case, the average accuracy of the ℓ validation sets is used as an estimator for the accuracy of the method. The combination of the hyperparameters with the best performance is chosen. As Schölkopf and Smola (2002), among others, explain, there are some possible disadvantages regarding cross-validation, although it is quite often used in practice. One reason is the obvious danger of overfitting because the training data sets and the validation data sets are related to each other. Another point is that a suitable set of hyperparameters obtained for a data

set of size n may differ from a suitable set of hyperparameters obtained for subsets of this data set of size $(1-1/\ell)n$. Often, the smaller data set used for training purposes needs a slightly stronger regularization (e.g., a larger value of λ) and suitable parameters for the kernel and the loss function also may be slightly different.

Many *heuristic choices* have been proposed for the hyperparameters of SVMs. One approach was proposed by Cherkassky and Ma (2004). Their proposal is based on both theoretical considerations and empirical results. The following suggestions for the regularization parameter C, the width of the ϵ-insensitive loss, and the bandwidth parameter γ of the Gaussian RBF kernel are suited for the case where all input variables are scaled to the interval $[0,1]$. They can easily be adjusted to non-scaled data. Regarding the regularization parameter C, Cherkassky and Ma (2004) agree with the findings of Mattera and Haykin (1999) that C should be chosen according to the range of the values of the response variable in the training data. Since the range is not robust against outliers, Cherkassky and Ma (2004) propose $\epsilon := 3\sigma\sqrt{(\ln n)/n}$ and $C := \max\{|\bar{y} - 3\sigma_y|, |\bar{y} + 3\sigma_y|\}$, where $\bar{y}$ and σ_y denote the mean and the standard deviation of the responses y_i in the training data, respectively. Note that this choice of C does not result in a null sequence (λ_n) if $n \to \infty$. In practice, σ_y will be unknown and must be estimated. To accomplish this, Cherkassky and Ma (2004) proposed a nearest-neighbor regression where the number of neighbors is chosen between 3 and 7. The noise will then be estimated using the residuals of this regression. As Cherkassky and Ma (2004) base all their considerations on the RBF kernel, the kernel parameter γ must also be determined. It is chosen depending on the number of input variables of the regression problem, its dimension d, as $\gamma = \sqrt{2}c^{1/d}$, where c is a some constant between 0.1 and 0.5, for which good SVM performance can be achieved. This heuristic method has the advantage that the choice of the hyperparameters can be accessed directly from the data, which allows relativly fast computation. The authors give several numerical examples that show the power of their approach when used on artificial data. It seems to be unknown, however, whether their heuristic choice of (C, ϵ, γ) is always suitable when applied to real-life data.

Momma and Bennett (2002) proposed the *pattern search* algorithm as a directed search method to determine the hyperparameters for SVMs. It examines points in the parameter space that are arranged in a pattern around the actual optimal point. The pattern depends on the number of parameters in the SVM. For SVMs based on the hinge loss and a Gaussian RBF kernel using the logarithms of the parameter value, the pattern with four elements

$$M = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

can be used to construct a pattern in the parameter space (C, γ). For the three hyperparameters C, ϵ, and γ for an SVM based on the ϵ-insensitive loss and the Gaussian RBF kernel, this pattern can be expanded to

$$M^* = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

The columns of M and M^* describe the change applied to a given parameter vector $q = (C, \gamma)^\mathsf{T}$ or $q^* = (C, \epsilon, \gamma)^\mathsf{T}$. This means that only one parameter is changed at a time. The pattern search algorithm itself works as follows.

i) Initialization. Choose a start pattern center $q^{(0)}$ and compute the value of the function to be minimized $g(q^{(0)})$. Furthermore, choose a factor $\Delta^{(0)}$ that denotes the expansion of the pattern and τ, the expansion at which the algorithm should stop.

ii) Optimization step. Compute $q_i^{(k+1)} := q^{(k)} + \Delta^{(k)} m_i$ for all columns m_i of M and the corresponding $g(q_i^{(k+1)})$. If $\min g(q_i^{(k+1)}) < g(q^{(k)})$, set $q^{(k+1)} := \arg\min g(q_i^{(k+1)})$ and $\Delta^{(k+1)} := \Delta^{(k)}$. Otherwise, set $q^{(k+1)} := q^{(k)}$ and $\Delta^{(k+1)} := \Delta^{(k)}/2$ and proceed to the stopping rule.

iii) Stopping rule. If $\Delta^{(k)} < \tau$, stop the algorithm. Otherwise, perform another optimization step.

The algorithm searches the parameter space pattern-wise and memorizes the best hyperparameter combination it comes across. If the center of the pattern is optimal, the pattern will be made smaller, which corresponds to a finer grid search. If the pattern is small enough, the algorithm will stop. In principle, the pattern search works similar to a grid search, but it only makes calculations for a subset of the grid points. By choosing the direction of the steepest descent among pattern points, it will omit a lot of grid points, which may lead to unsatisfactory results when their respective parameter combinations are applied to the data. Furthermore, a more exhaustive search will be done automatically in the region of interest. This can lead to computational savings, but there is the danger that the algorithm will only find a local optimum.

Besides the accuracy, the *number of evaluations* (i.e., the number of combinations of the hyperparameters which are tested in order to find the best combination of the hyperparameters) is also important for practical purposes when several methods are compared.

To the best of our knowledge, there is currently no practical method known that chooses the hyperparameters of SVMs in an optimal manner for all data sets and is applicable for sample sizes of any size. Nevertheless, a few general results concerning how to find suitable hyperparameters for SVMs based on numerical research[3] for benchmark data sets and simulated data sets may be in order. Although for every fixed combination of hyperparameters we have a convex optimization problem to determine an empirical SVM solution, we are in general faced with a non-convex problem when we optimize over the hyperparameters. Typically, there is no single optimal choice of the hyperparameters but a connected region of close to optimal values. The change in

[3] See Christmann *et al.* (2005).

the level of the target function is sometimes approximately parallel to the input parameters, which seems to be one explanation why a pattern search often performs well. If computationally feasible, a fine grid covering a broad range of the parameter space or a two-stage grid often gives a suitable set of hyperparameters, but at a high computational cost. The Nelder-Mead search sometimes performs very poorly if the parameters for inflation or deflation of this algorithm are inappropriately chosen. Some practitioners do not care too much about possible disadvantages of cross-validation because this method often gives good results even if the resulting hyperparameters are not adjusted.

11.4 Software Packages

There exist well-established numerical packages (e.g., `NAG` and `IMSL`™) that can be used to solve convex or quadratic programs. An advantage of these software products is that they are in general numerically very stable. Some of these packages contain routines that are designed for large sparse systems, but this property is usually not needed to compute $f_{\mathrm{D},\lambda}$ as the kernel matrix K is dense (i.e., most coefficients $K_{i,j} = k(x_i, x_j)$ do not equal zero). One can argue that some commercial packages for general use have the disadvantages of a high price and a relatively large computation time for the determination of $f_{\mathrm{D},\lambda}$ because these programs are not specifically designed for SVMs.

Now we will mention a few implementations that were designed for solving the numerical problems of SVMs. A much longer list of programs to compute SVMs can be found, for example, on the websites `www.kernel-machines.org`[4] and `www.support-vector-machines.org`. Note that it is often helpful to scale all input variables to increase numerical stability, provided the implementation does not automatically scale the data.

`LIBSVM`

Chang and Lin (2004) developed a user-friendly library called `LIBSVM` for the computation of SVMs. This software is able to fit SVMs for classification, regression, and distribution estimation problems, is programmed in `C++` and `Java`, and belongs to the state-of-the-art software tools that are currently available for SVMs. In particular, the use of the hinge loss and the ϵ-insensitive loss is possible, as well as using ν-support vector machines. Updates are regularly available. This software is partially based on Fan *et al.* (2005). `LIBSVM` has the advantage that there are interfaces to several other software tools; e.g., to `R`, `MATLAB`®, `Python`, `Perl`, and the data mining software `Weka`. There exists a bundle of related programs called `LIBSVM Tools` and a graphical interface that is very suitable for demonstrating SVM classification and regression to

[4] In March 2008, there were around 45 software tools for SVMs mentioned on this website.

students. Additionally, there is a useful practical guide for SVM classification available written mainly for beginners.

SVM^{light}

Joachims (1999) developed SVM^{light}, which was one of the first implementations to make SVMs applicable in classification and regression problems for large data sets. It is written in C. In particular, one can use the hinge loss and the ϵ-insensitive loss function. Currently, SVM^{light} is one part of the software SVM^{struct} developed by the same author, which is a collection of SVM algorithms for predicting multivariate or structured outputs. It performs supervised learning by approximating a mapping from the input space X to the output space Y using labeled training examples $(x_1, y_1), \ldots, (x_n, y_n)$. Unlike regular SVMs, however, which consider only univariate predictions as in classification and regression, SVM^{struct} can predict complex objects y such as trees, sequences, or sets. Examples of problems with complex outputs are natural language parsing, sequence alignment in protein homology detection, and Markov models for part-of-speech tagging. The SVM^{struct} algorithm can also be used for linear-time training of binary and multi-class SVMs using a linear kernel. SVM^{struct} can be thought of as an API for implementing different kinds of complex prediction algorithms. SVM^{multiclass} is for multi-class classification problems, SVM^{map} has the goal of learning rankings, and SVM^{perf} is useful for learning a binary classification rule that directly optimizes the area under the receiver operating characteristic (ROC) curve or other criteria.

R

The statistical software package R (R Development Core Team, 2006) can be used to compute SVMs for classification and regression problems provided the function svm developed by D. Meyer from the add-on package e1071 is used. This function is based on LIBSVM and uses methods developed by Fan *et al.* (2005). Together with the graphical routines provided by R, this implementation for SVMs is from our point of view especially appropriate for small to moderate sized data sets, for running simulation studies for small data sets, and can easily be used by students. We also like to mention two other R packages. klarR developed at the Department of Statistics of the University of Dortmund contains an interface to SVM^{light} and the package svmpath developed by T. Hastie from the Stanford University can be used to compute the entire regularization path for an SVM based on the hinge for small data sets.

mySVM

Rüping (2000) developed the implementation mySVM for SVMs for classification, regression, and distribution estimation problems. This implementation

is based on SVM^{light}. The software can also be used for SVMs based on the pinball loss for quantile regression if the options epsilon=0, L+= $1-\tau$, and L-= τ are specified for the pinball loss function $L_{\tau\text{-pin}}$. There exists a Java implementation of mySVM designed to run inside of a database.

myKLR

Keerthi *et al.* (2005) developed a fast SMO-type algorithm for SVMs based on the logistic loss function for classification purposes. The algorithm uses many technical tricks and special properties of this particular loss function to solve the dual problem efficiently. The software myKLR is an implementation of this algorithm and was written by Rüping (2003). This implementation is much faster than quasi-Newton algorithms such as the Broyden-Fletcher-Goldfarb-Shanno algorithm with bound constraints applied to the primal problem for large data sets. Nevertheless, myKLR needs considerably more computation time than comparable SMO algorithms for SVMs based on the hinge loss. This is due to the fact that the empirical solution of SVMs based on the logistic loss is not sparse and that a convex and not (only) a quadratic optimization problem as for the case of the hinge loss must be solved.

LS-SVMlab

LS-SVMlab is a toolbox for SVMs based on the least squares loss function and uses methods described in the textbook by Suykens *et al.* (2002). This software is written in MATLAB® and C and contains besides implementations to solve SVMs in classification and regression problems routines for kernel-based principal component analysis.

11.5 Further Reading and Advanced Topics

Section 11.1, which showed that empirical SVM decision functions $f_{\mathrm{D},\lambda}$ are solutions of special convex or even quadratic programs with constraints, is mainly based on Schölkopf and Smola (2002), Cristianini and Shawe-Taylor (2000), and Smola and Schölkopf (2004). More details on kernel logistic regression can be found in Keerthi *et al.* (2005). We refer to Schölkopf *et al.* (2000) and Smola and Schölkopf (2004) for additional information regarding ν-support vector regression and related topics and to the textbook by Suykens *et al.* (2002) for SVMs based on the least squares loss. For quantile regression, we refer to Koenker and Bassett (1978), He (1997), Koenker (2005), and Takeuchi *et al.* (2006). For problems with monotonicity constraints, we refer to Takeuchi *et al.* (2006).

Section 11.2, on implementation techniques to compute SVMs, is mainly based on Keerthi *et al.* (2001), Fan *et al.* (2005), and Chen *et al.* (2006).

These papers also investigate generalizations of the algorithms given here. Chen *et al.* (2006) also offer results that show that we "only" have linear convergence for decomposition methods based on the algorithm WSS2 for SVMs based on the hinge loss. Many of these results are valid for SVMs based on the ϵ-insensitive loss function and for one-class SVMs, too. We conjecture that this is also true for SVMs based on the pinball loss, but as far as we know, this has not yet been proven. For additional details on decomposition methods, we refer to Osuna *et al.* (1997), Joachims (1999), and Platt (1999). Some improvements for Platt's SMO algorithm for SVM classifiers were proposed by Keerthi *et al.* (2001). The optimization problem (11.41) related to the maximal violating pair algorithm was probably first considered by Joachims (1999). List and Simon (2004) give a general convergence theorem for the decomposition method. Hush and Scovel (2003) propose a polynomial-time decomposition algorithm for SVMs and prove necessary and sufficient conditions for stepwise improvement of their algorithm. For general polynomial time decomposition algorithms, we refer to List and Simon (2007). As far as we know, it is not yet known whether existing SVM algorithms satisfy the conditions, but the authors also provide an algorithm that fulfills the conditions. Let $c(\tilde{K})$ denote the maximum of the norms of the (2×2) submatrices determined by restricting $\tilde{K}$ from the dual program to two indices. If the constant $C = 1/(2n\lambda)$ satisfies $\sqrt{1/2} \le C \le nc(\tilde{K})$, then this algorithm for the computation of an empirical SVM solution based on the hinge loss needs at most $4c(\tilde{K})C^2n^4/\varepsilon$ iterations with a guaranteed precision of ε. For a formal analysis of stopping criteria of decomposition methods for SVMs, we refer also to Lin (2002a), Chen *et al.* (2006), and List *et al.* (2007).

For leave-one-out estimates, we refer to Schölkopf and Smola (2002, Chapter 12), Joachims (2002), and Mukherjee *et al.* (2006). Seeger (2007) proposed cross-validation optimization for large-scale hierarchical classification kernel methods, and the kernel hyperparameters are chosen automatically by maximizing the cross-validation log likelihood in a gradient-based way. Keerthi *et al.* (2007) proposed an efficient method for gradient-based adaptation of the hyperparameters. Davies *et al.* (2008) discussed general nonparametric regression as an example of model choice.

There is a large and rapidly increasing body of literature on implementation techniques for SVMs. Much more information than in Section 11.2 can be found for example in Schölkopf and Smola (2002, Chapter 10) and Cristianini and Shawe-Taylor (2000, Chapter 7). Keerthi *et al.* (2005) proposed a fast dual algorithm for SVMs based on the logistic loss for classification based on an SMO decomposition. This algorithm is implemented in the software `myKLR` (Rüping, 2003). An overview of SVM solvers is given by Bottou and Lin (2006). Joachims (1999), Osuna and Girosi (1999), Platt (1999), Huang *et al.* (2006), and Bottou *et al.* (2007) describe techniques especially designed for making SVMs applicable for large data sets. Joachims (2002) considers fast algorithms for SVMs in the context of text classification. Smola and Schölkopf (2004) describe methods for the numerical computation

of SVMs, with special emphasis on regression problems. For data sets with millions of data points, a subsampling strategy such as robust learning from bites may be useful, too.

Most literature on computational aspects of SVMs currently concentrates on solving the dual optimization problem, but there is increasing interest also in algorithms that solve the primal problem of SVMs. Mangasarian (2002) proposed a finite Newton method for classificaton purposes. Keerthi and DeCoste (2005) proposed an algorithm to solve the primal problem of linear SVMs based on the least squares loss function; see also Suykens *et al.* (2002) for such SVMs. Joachims (2006) developed an algorithm and software to train linear SVMs in *linear* time. Chapelle (2007) argued that the primal problem can often be solved efficiently both for linear and nonlinear SVMs. This also offers the opportunity to investigate new families of algorithms for large-scale SVMs.

Corresponding to the large number of implementation techniques that were proposed for the numerical computation of $f_{\mathrm{D},\lambda}$, there exist many software implementations. A longer list of implementations than the one we gave for the computation of SVMs and related methods can again be found on the websites `www.kernel-machines.org` and `www.support-vector-machines.org`.

If the number of input variables d is very large, *feature selection* can be helpful to increase the precision and to decrease the computational burden of SVMs. Many researchers proposed feature selection methods or compared such methods. A general framework for feature selection is described by Schölkopf and Smola (2002, Chapter 14). We refer to Guyon *et al.* (2002) for recursive feature elimination in the context of gene selection for cancer classification using SVMs. Krishnapuram *et al.* (2004) considered joint feature selection and classifier design in the context of gene expression analysis, and Hochreiter and Obermayer (2004) applied SVMs in the context of gene selection for microarray data. Neumann *et al.* (2005) considered combined SVM-based feature selection and pattern recognition. Their approach is based on additional regularization and embedded nonlinear feature selection and uses difference of convex functions programming from the general framework of non-convex continuous optimization. Cai *et al.* (2007) compared several feature selection and classification algorithms to identify malicious executables and found SVM classifiers to be superior in terms of good prediction accuracy, short training time, and low danger of overfitting. Song *et al.* (2007) investigated supervised feature selection via dependence estimation.

11.6 Summary

The empirical SVM decision function $f_{\mathrm{D},\lambda}$ is defined as the solution of a minimization problem over an infinite-dimensional reproducing kernel Hilbert space H that can have an infinite dimension. Nevertheless, $f_{\mathrm{D},\lambda}$ can be evaluated numerically by solving a finite-dimensional convex program.

Many classical numerical algorithms for solving such convex programs are not well-suited to compute $f_{D,\lambda}$ for large sample sizes n. However, there are algorithms to compute $f_{D,\lambda}$ efficiently even for large values of n. Some of these algorithms are based on sequential minimal optimization (SMO). One main advantage of such algorithms is that it is not necessary to store the $(n \times n)$ matrix $K = (k(x_j, x_i))$ in the memory of the computer.

There are loss functions L such that the numerical problem of computing $f_{D,\lambda}$ can be solved relatively quickly. Among those loss functions are the hinge loss and the least squares loss for classification, the ϵ-insensitive loss function and the least squares loss function for regression, and the pinball loss function for kernel-based quantile regression.

There exist loss functions such that not only $f_{D,\lambda}$ can be computed relatively quickly but it also has good robustness properties in the sense of Chapter 10. Examples are the hinge loss, the ϵ-insensitive loss, and the pinball loss. The least squares loss is not Lipschitz-continuous and yields usually non-robust estimates.

It is not always suitable to use a loss function fulfilling the above-mentioned properties of fast computation and robustness. One counterexample is the hinge loss, which does not allow estimation of the conditional probabilities $P(Y|x)$. In contrast, it is possible to estimate these conditional probabilities based on the Lipschitz-continuous logistic loss function for classification problems. The empirical SVM decision function $f_{D,\lambda}$ based on this loss function offers good robustness properties if used in combination with a bounded universal kernel (e.g., the Gaussian RBF kernel) but has the disadvantage of a substantially higher computation time for large sample sizes compared with the hinge loss.

The SVM decision function $f_{D,\lambda}$ and the corresponding empirical risk $\mathcal{R}_{L,D}(f_{D,\lambda})$ depend critically on hyperparameters such as λ and parameters used by the loss function and the kernel. Currently, there seems to be no easy and computationally fast way to determine these hyperparameters for all data sets in an optimal manner, although different computationally intensive methods can offer a suitable choice.

11.7 Exercises

11.1. Numerical exercise ($\star$)
Compute $f_{D,\lambda}$ and make plots similar to those in Figure 10.11 for the daily milk consumption data set given in Table 10.1 using the ϵ-insensitive loss function and a Gaussian RBF kernel and a polynomial kernel. Use different values of the hyperparameters and study their effect.

Hint: Use, for example, one of the software products `LIBSVM`, $\texttt{SVM}^{light}$, or `mySVM`, or the function `svm` of the `R`-package `e1071`.

11.2. SVM based on hinge loss ($\star$)
Derive the Lagrangian and the dual problem for the computation of $f_{D,\lambda}$

based on the hinge loss function. Furthermore, consider an SVM based on L_{hinge} and the classification problem with an additional offset term $b \in \mathbb{R}$. Derive the primal convex program, the Lagrangian L^*, and the dual program for the computation of $(f_{\text{D},\lambda}, b_{\text{D},\lambda})$.

Hint: Schölkopf and Smola (2002).

11.3. SVM based on logistic classification loss $(\star)$
Derive the Lagrangian and the dual problem for the computation of $f_{\text{D},\lambda}$ based on the logistic loss for classification. Furthermore, consider an SVM based on $L_{\text{c-logist}}$ and the classification problem with an additional offset term $b \in \mathbb{R}$. Derive the primal convex program, the Lagrangian L^*, and the dual program for the computation of $(f_{\text{D},\lambda}, b_{\text{D},\lambda})$.

Hint: Keerthi *et al.* (2005).

11.4. SVM based on least squares loss for classification$(\star)$
Work out the details for Example 11.5. Compute the Lagrangian and its partial derivatives. Show that $f_{\text{D},\lambda}$ is the solution of a set of linear equations. Repeat the calculations for the case of an additional offset term $b \in \mathbb{R}$.

Hint: Suykens *et al.* (2002).

11.5. SVM based on distance-based loss $(\star)$
Work out the details for Example 11.6. Derive L^* and the dual program.

Hint: Smola and Schölkopf (1998) and Schölkopf and Smola (2002).

11.6. SVM based on ϵ-insensitive loss $(\star)$
Derive the Lagrangian and the dual problem for the computation of $f_{\text{D},\lambda}$ based on the ϵ-insensitive loss function. Furthermore, consider an SVM based on $L_{\epsilon\text{-insens}}$ and the regression problem with an additional offset term $b \in \mathbb{R}$. Derive the primal convex program, the Lagrangian L^*, and the dual program for the computation of $(f_{\text{D},\lambda}, b_{\text{D},\lambda})$.

Hint: Smola and Schölkopf (1998).

11.7. SVM based on least squares loss for regression $(\star)$
Consider a regression problem with $X = \mathbb{R}^d$ and $Y = \mathbb{R}$. Derive the primal program, the Lagrangian L^*, and the dual program for the computation of $f_{\text{D},\lambda}$ based on $L_{\text{LS}}(y,t) = (y-t)^2$, $y, t \in \mathbb{R}$. Explain why $f_{\text{D},\lambda}$ can be computed relatively quickly even for large sample sizes n. Furthermore, consider an SVM based on this loss function and a regression problem with an additional offset term $b \in \mathbb{R}$. Derive the primal convex program, the Lagrangian L^*, and the dual program for the computation of $(f_{\text{D},\lambda}, b_{\text{D},\lambda})$.

Hint: Suykens *et al.* (2002).

11.8. SVMs based on the pinball loss for quantile regression $(\star)$
Work out the details for Example 11.9. Compute also the Lagrangian and the dual program.

Hint: Schölkopf and Smola (2002) and Takeuchi *et al.* (2006).