Support Vector Machines

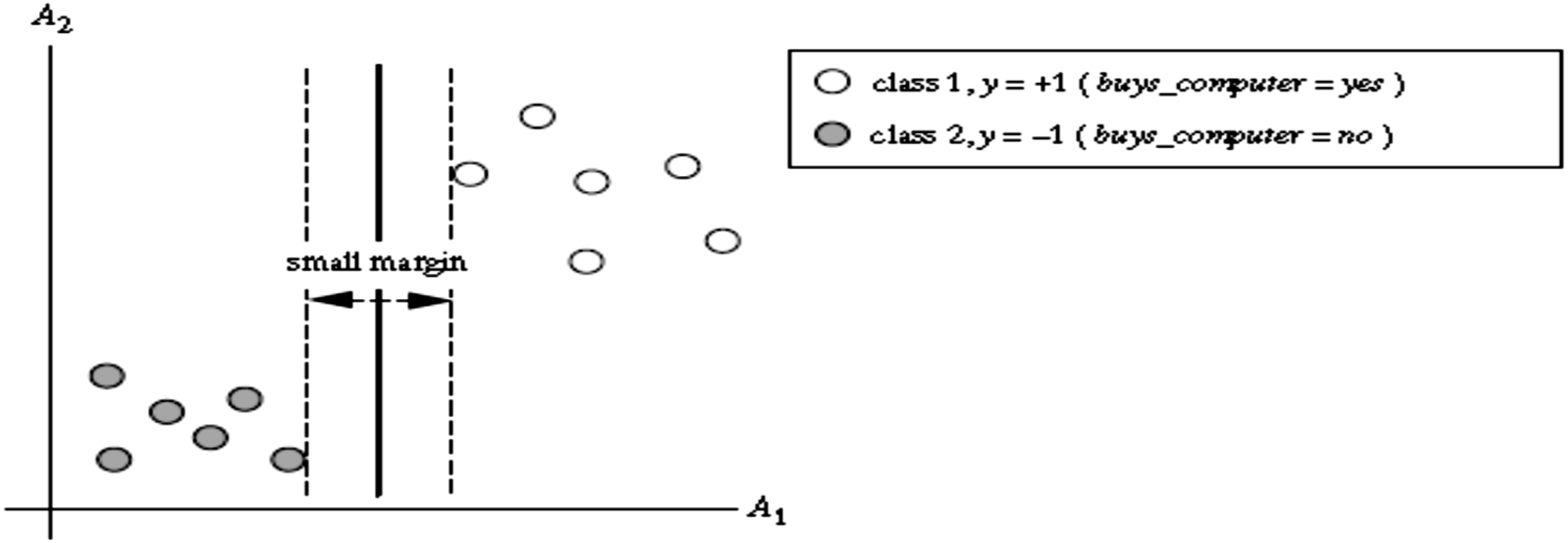
Dr. Priyank Thakkar
Associate Professor
CSE Department
Institute of Technology
Nirma University

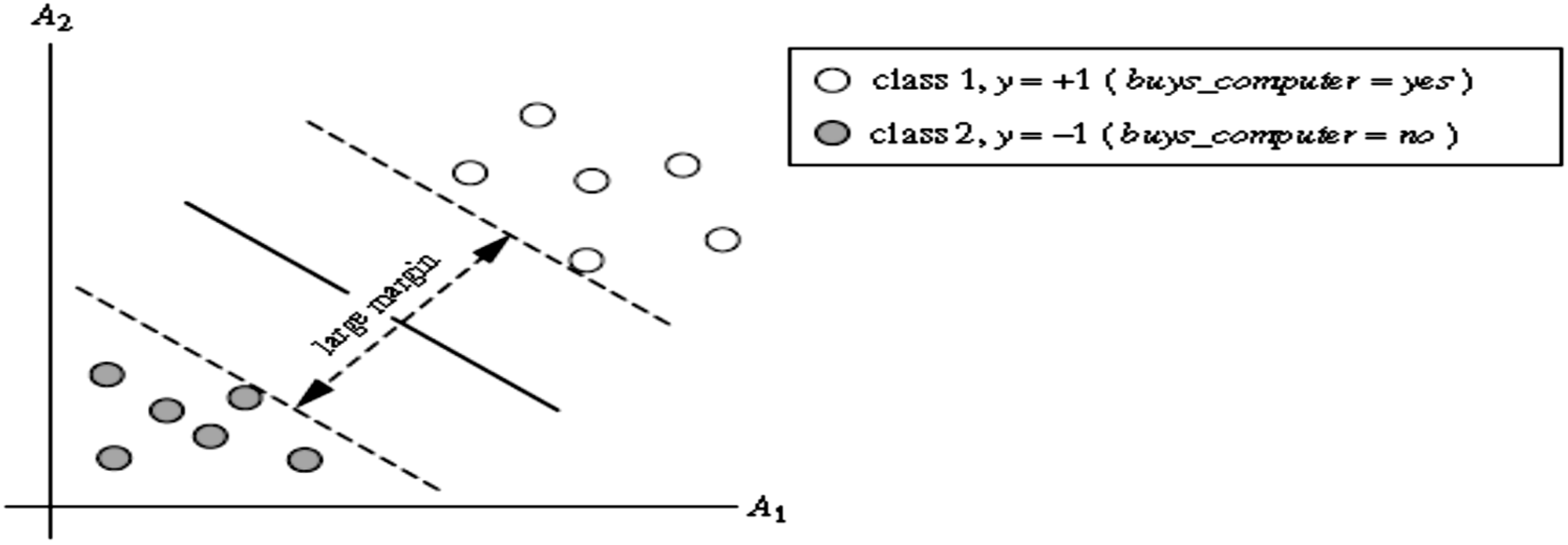
Some Facts about SVM

- Linear Classifier
  - So, can't it learn non-linear decision boundary?
- Binary Classifier
  - What about multiple classes?

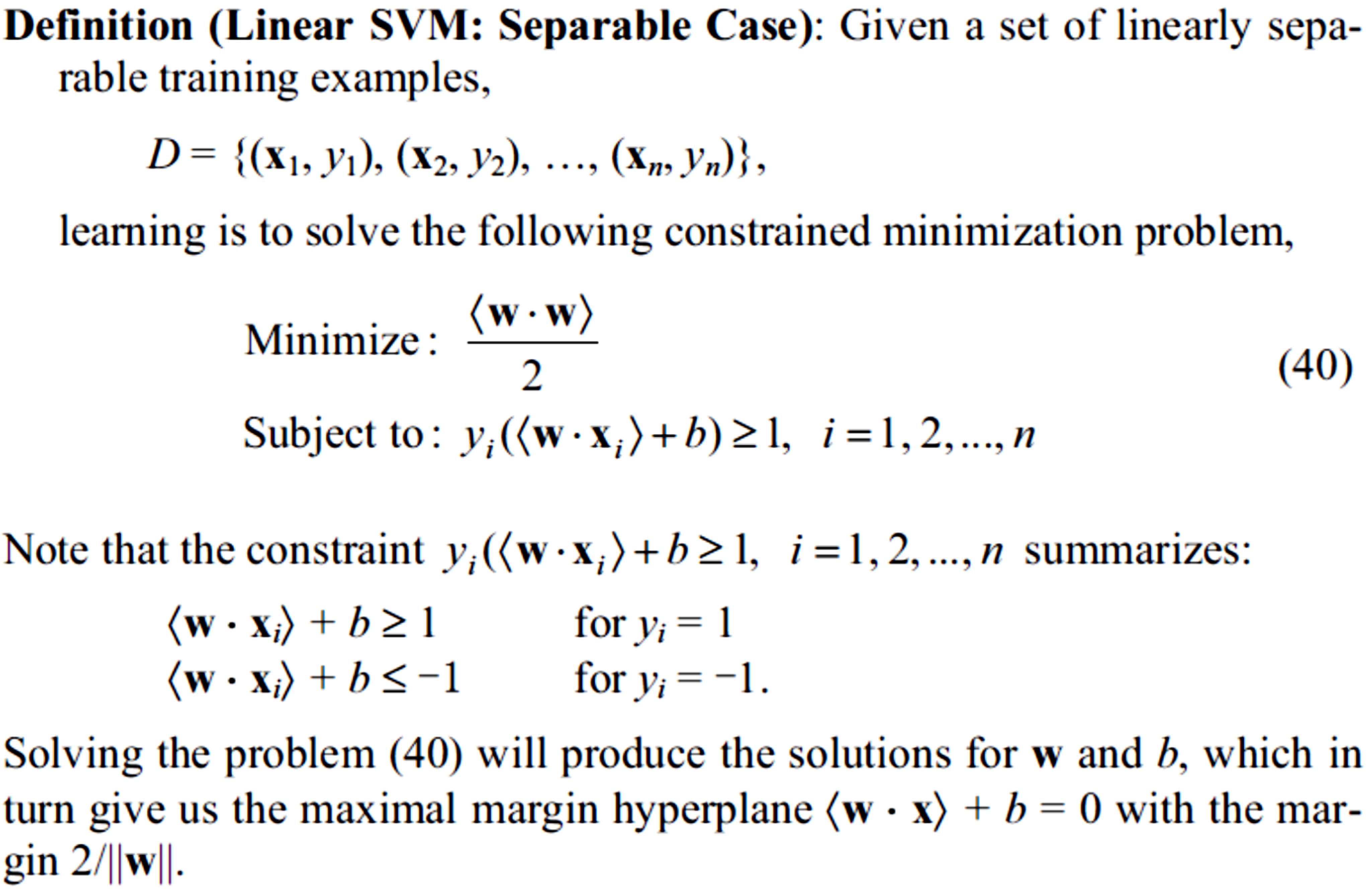
SVM

- Which hyper plane should be selected as decision boundary?


A_2
○ class 1, $y = +1$ (*buys_computer = yes*)
● class 2, $y = -1$ (*buys_computer = no*)
small margin
A_1


A_2
○ class 1, $y = +1$ (*buys_computer = yes*)
● class 2, $y = -1$ (*buys_computer = no*)
large margin
A_1

SVM



Definition (Linear SVM: Separable Case): Given a set of linearly separable training examples,

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\},$$

learning is to solve the following constrained minimization problem,

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \qquad (40)$$

$$\text{Subject to}: y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, \ldots, n$$

Note that the constraint $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1, \quad i = 1, 2, \ldots, n$ summarizes:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 \qquad \text{for } y_i = 1$$

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 \qquad \text{for } y_i = -1.$$

Solving the problem (40) will produce the solutions for **w** and b, which in turn give us the maximal margin hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ with the margin $2/\|\mathbf{w}\|$.

SVM

$$\text{Minimize: } \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2}$$

$$\text{Subject to: } y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., n$$

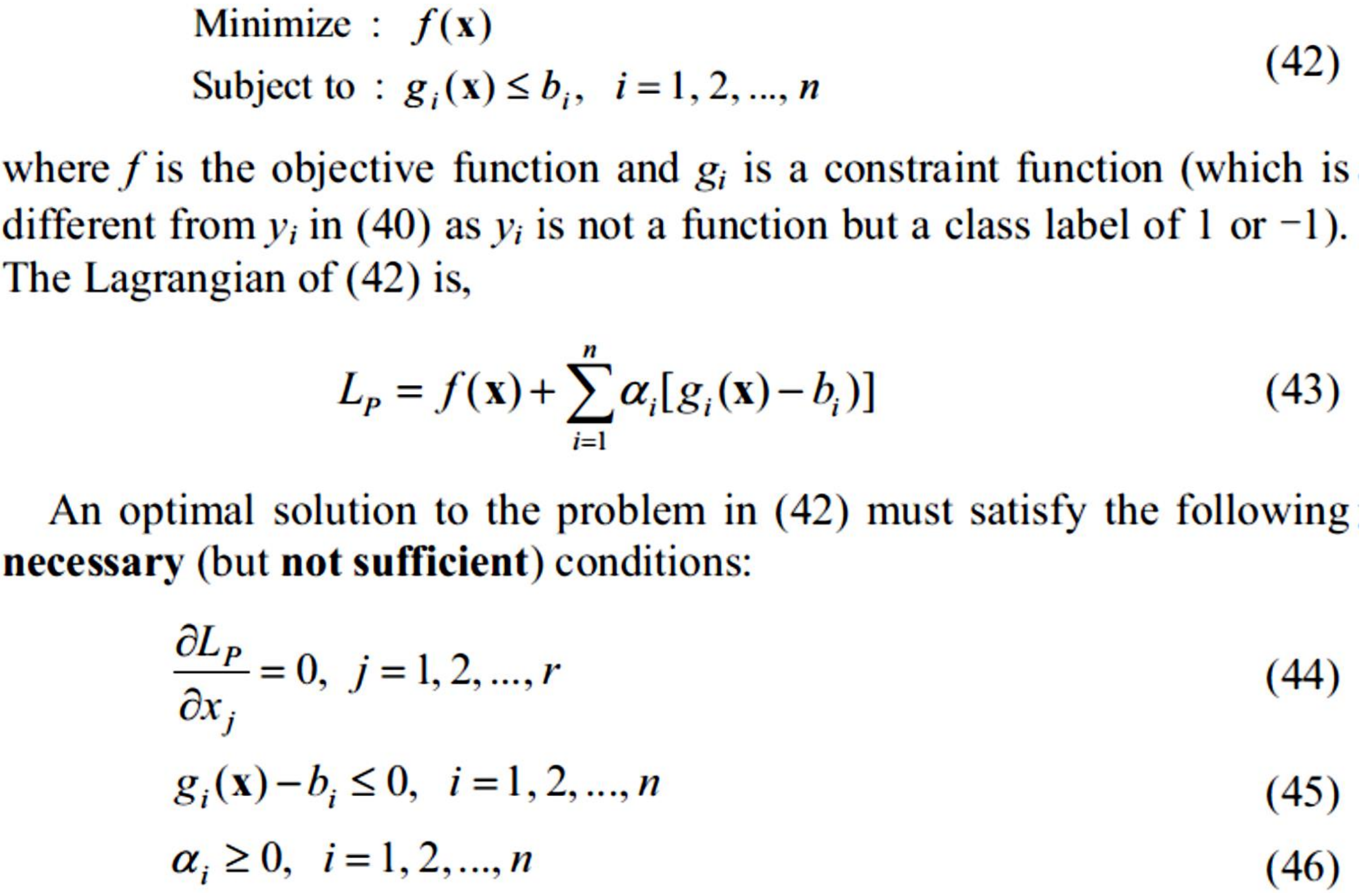
Since the objective function is quadratic and convex and the constraints are linear in the parameters **w** and b, we can use the standard Lagrangian multiplier method to solve it.

Instead of optimizing only the objective function (which is called unconstrained optimization), we need to optimize the Lagrangian of the problem, which considers the constraints at the same time. The need to consider constraints is obvious because they restrict the feasible solutions. Since our inequality constraints are expressed using "≥", the **Lagrangian** is formed by the constraints multiplied by positive Lagrange multipliers and subtracted from the objective function, i.e.,

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1] \qquad (41)$$

where $\alpha_i \geq 0$ are the **Lagrange multipliers**.

SVM (Kuhn-Tucker Conditions)



$$\begin{aligned}&\text{Minimize} \ : \ f(\mathbf{x}) \\ &\text{Subject to} \ : \ g_i(\mathbf{x}) \le b_i, \quad i = 1, 2, ..., n\end{aligned} \qquad (42)$$

where f is the objective function and g_i is a constraint function (which is different from y_i in (40) as y_i is not a function but a class label of 1 or −1). The Lagrangian of (42) is,

$$L_P = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i [g_i(\mathbf{x}) - b_i)] \qquad (43)$$

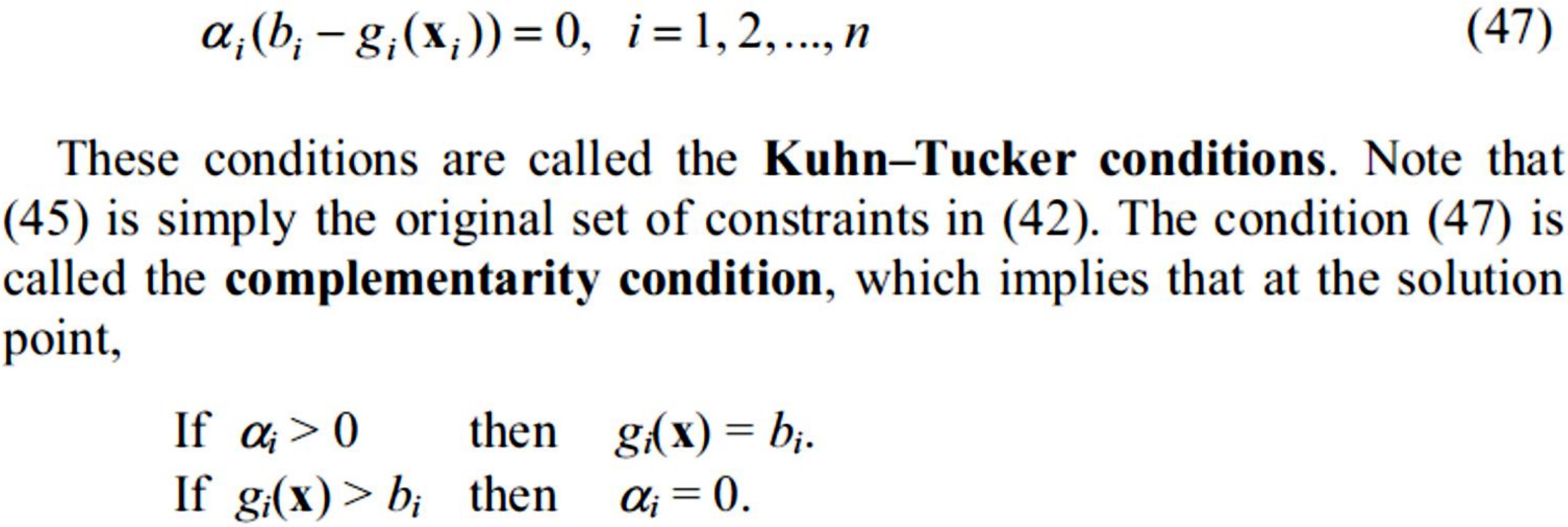
An optimal solution to the problem in (42) must satisfy the following necessary (but **not sufficient**) conditions:

$$\frac{\partial L_P}{\partial x_j} = 0, \ \ j = 1, 2, ..., r \qquad (44)$$

$$g_i(\mathbf{x}) - b_i \le 0, \ \ i = 1, 2, ..., n \qquad (45)$$

$$\alpha_i \ge 0, \ \ i = 1, 2, ..., n \qquad (46)$$

SVM (Kuhn-Tucker Conditions)



$$\alpha_i(b_i - g_i(\mathbf{x}_i)) = 0, \quad i = 1, 2, \ldots, n \tag{47}$$

These conditions are called the **Kuhn–Tucker conditions**. Note that (45) is simply the original set of constraints in (42). The condition (47) is called the **complementarity condition**, which implies that at the solution point,

$$\begin{array}{lll} \text{If } \alpha_i > 0 & \text{then} & g_i(\mathbf{x}) = b_i. \\ \text{If } g_i(\mathbf{x}) > b_i & \text{then} & \alpha_i = 0. \end{array}$$

SVM

Kuhn-Tucker Conditions

$$L_P = \frac{1}{2}\langle \mathbf{w}\cdot\mathbf{w}\rangle - \sum_{i=1}^{n}\alpha_i[y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1] \tag{41}$$

Let us come back to our problem. For the minimization problem (40), the Kuhn–Tucker conditions are (48)–(52):

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i\alpha_i x_{ij} = 0, \quad j = 1, 2, \ldots, r \tag{48}$$

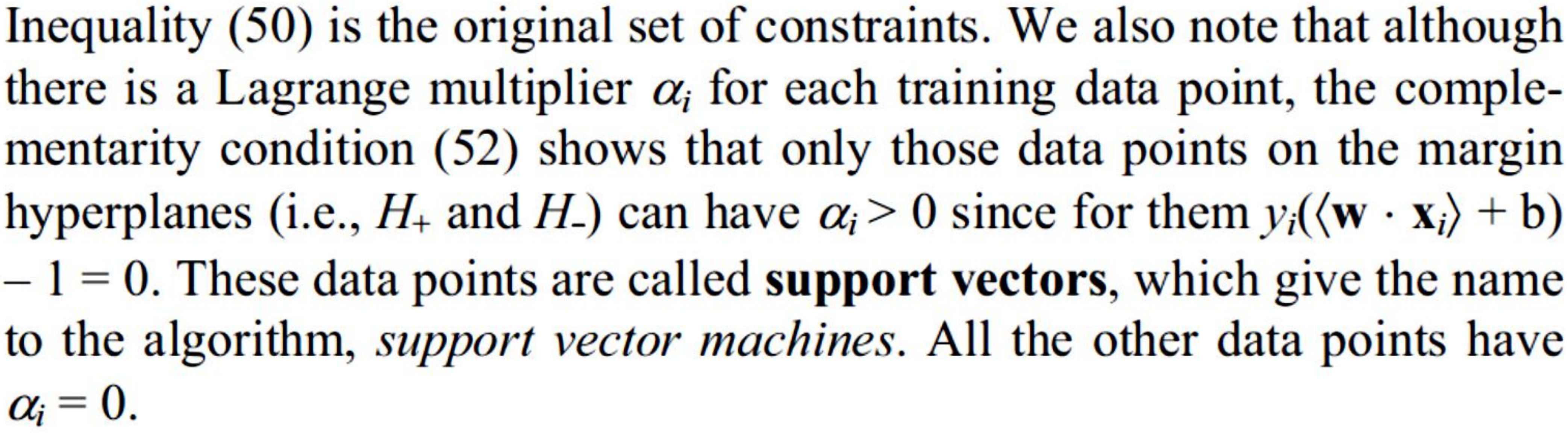
$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i\alpha_i = 0 \tag{49}$$

$$y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1 \geq 0, \quad i = 1, 2, \ldots, n \tag{50}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, n \tag{51}$$

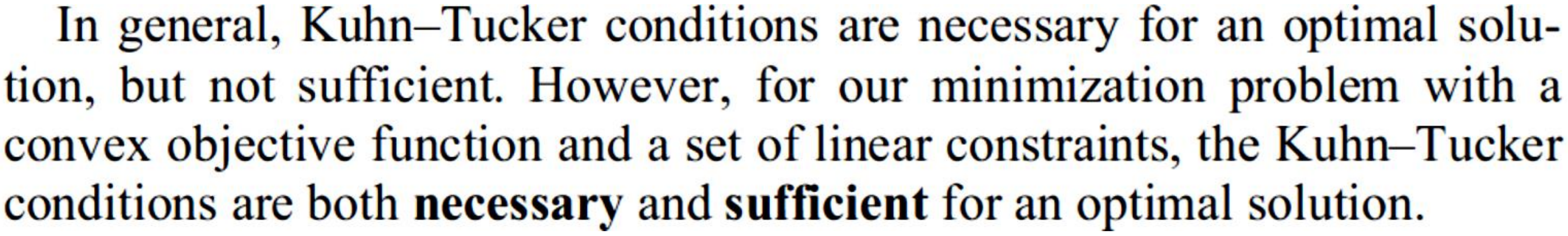
$$\alpha_i(y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1) = 0, \quad i = 1, 2, \ldots, n \tag{52}$$

SVM



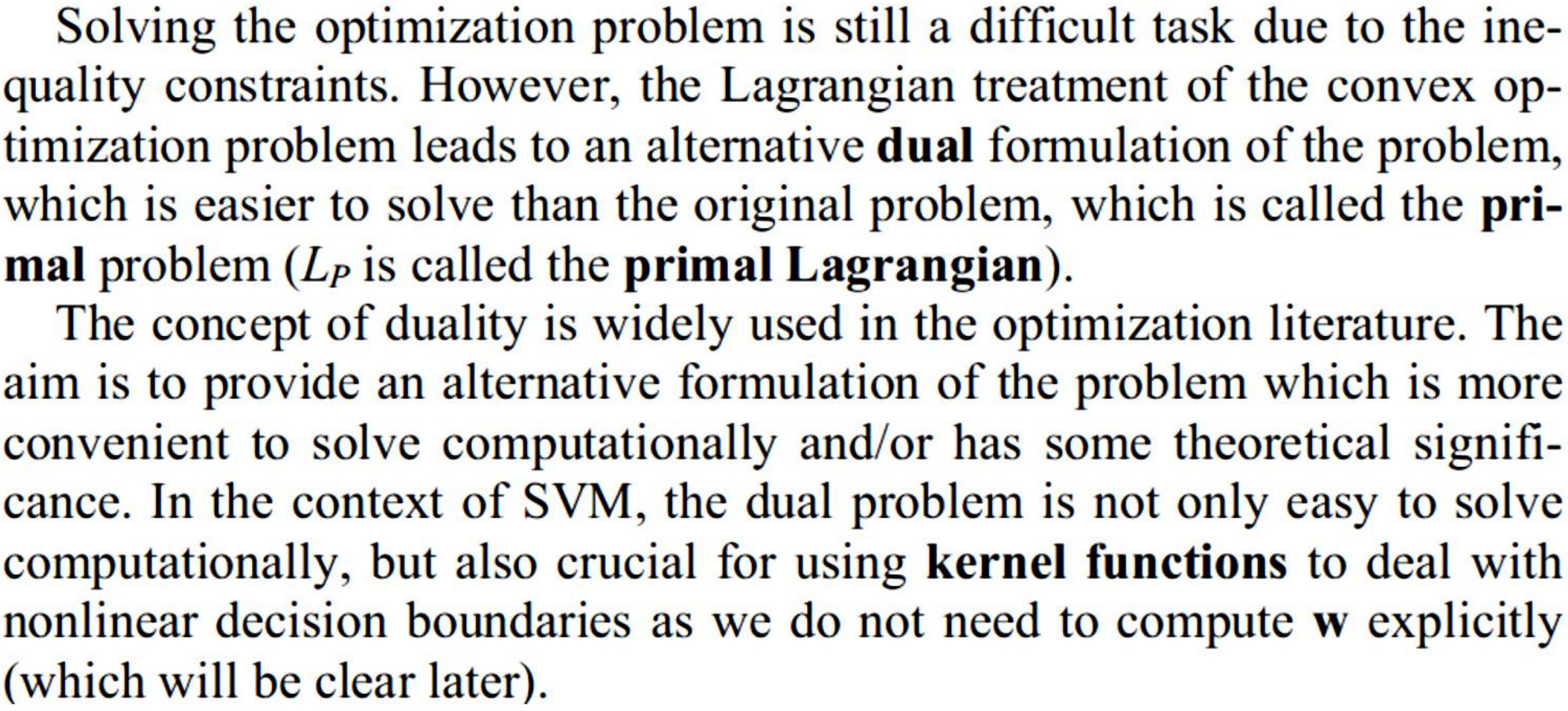
Inequality (50) is the original set of constraints. We also note that although there is a Lagrange multiplier α_i for each training data point, the complementarity condition (52) shows that only those data points on the margin hyperplanes (i.e., H_+ and H_-) can have $\alpha_i > 0$ since for them $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + \mathrm{b}) - 1 = 0$. These data points are called **support vectors**, which give the name to the algorithm, *support vector machines*. All the other data points have $\alpha_i = 0$.

SVM



In general, Kuhn–Tucker conditions are necessary for an optimal solution, but not sufficient. However, for our minimization problem with a convex objective function and a set of linear constraints, the Kuhn–Tucker conditions are both **necessary** and **sufficient** for an optimal solution.

SVM



Solving the optimization problem is still a difficult task due to the inequality constraints. However, the Lagrangian treatment of the convex optimization problem leads to an alternative **dual** formulation of the problem, which is easier to solve than the original problem, which is called the **primal** problem (L_P is called the **primal Lagrangian**).

The concept of duality is widely used in the optimization literature. The aim is to provide an alternative formulation of the problem which is more convenient to solve computationally and/or has some theoretical significance. In the context of SVM, the dual problem is not only easy to solve computationally, but also crucial for using **kernel functions** to deal with nonlinear decision boundaries as we do not need to compute **w** explicitly (which will be clear later).

SVM

Transforming from the primal to its corresponding dual can be done by setting to zero the partial derivatives of the Lagrangian (41) with respect to the **primal variables** (i.e., **w** and *b*), and substituting the resulting relations back into the Lagrangian. This is to simply substitute (48), which is

$$w_j = \sum_{i=1}^{n} y_i \alpha_i x_{ij}, \quad j = 1, 2, \ldots, r \tag{53}$$

and (49), which is $$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{n} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1]$$

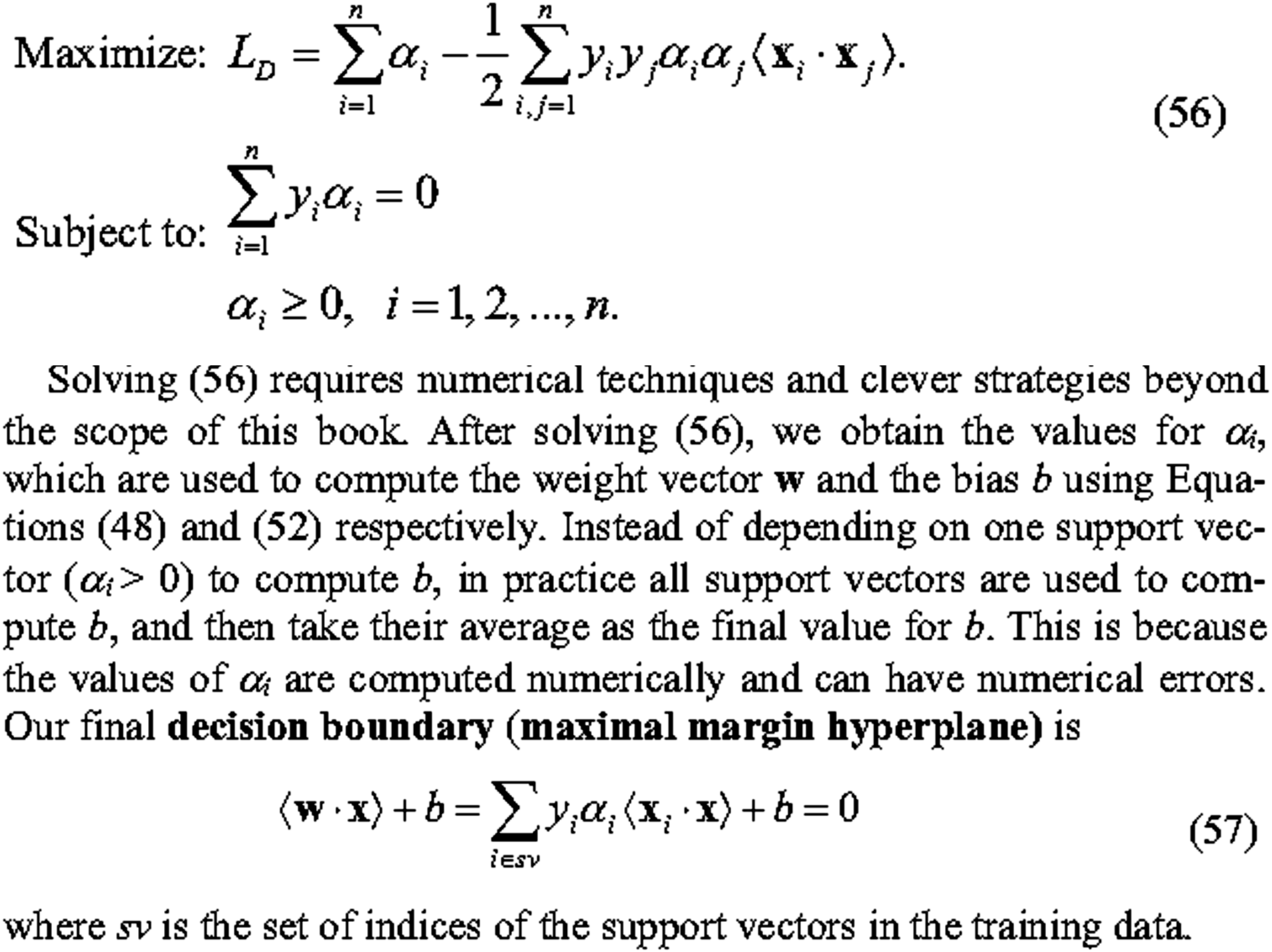
$$\sum_{i=1}^{n} y_i \alpha_i = 0, \tag{54}$$

into the original Lagrangian (41) to eliminate the primal variables, which gives us the dual objective function (denoted by L_D),

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{55}$$

L_D contains only **dual variables** and must be maximized under the simpler constraints, (48) and (49), and $\alpha_i \geq 0$. Note that (48) is not needed as it has already been substituted into the objective function L_D. Hence, the **dual** of the primal Equation (40) is

SVM



Maximize: $$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{56}$$

Subject to: $$\sum_{i=1}^{n} y_i \alpha_i = 0$$

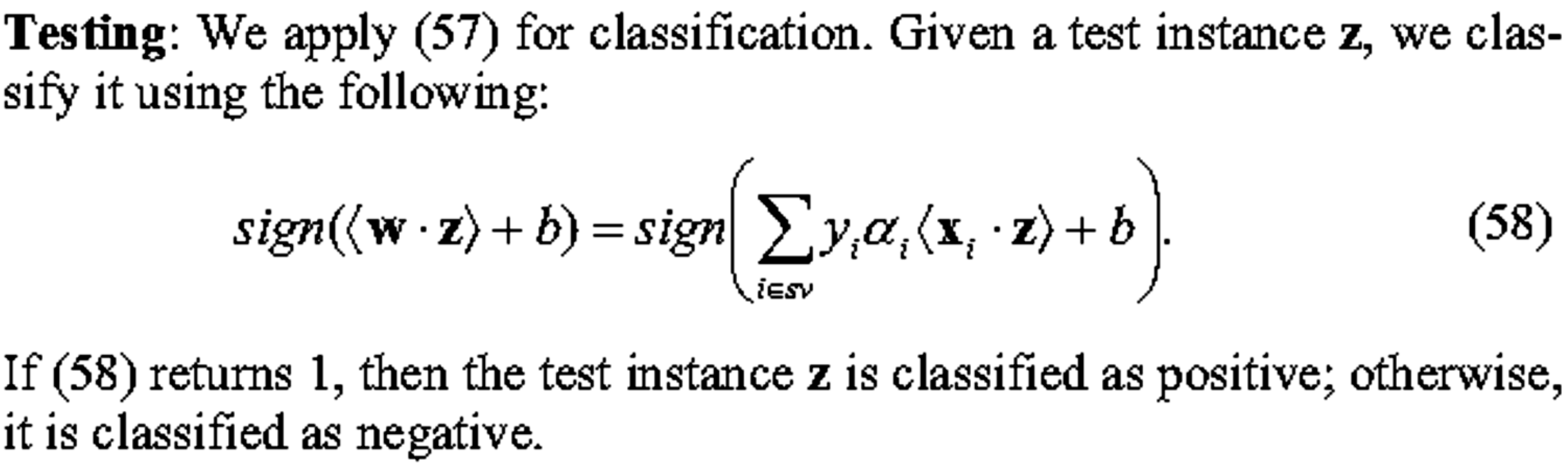
$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, n.$$

Solving (56) requires numerical techniques and clever strategies beyond the scope of this book. After solving (56), we obtain the values for α_i, which are used to compute the weight vector **w** and the bias *b* using Equations (48) and (52) respectively. Instead of depending on one support vector ($\alpha_i > 0$) to compute *b*, in practice all support vectors are used to compute *b*, and then take their average as the final value for *b*. This is because the values of α_i are computed numerically and can have numerical errors. Our final **decision boundary (maximal margin hyperplane)** is

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \tag{57}$$

where *sv* is the set of indices of the support vectors in the training data.

SVM

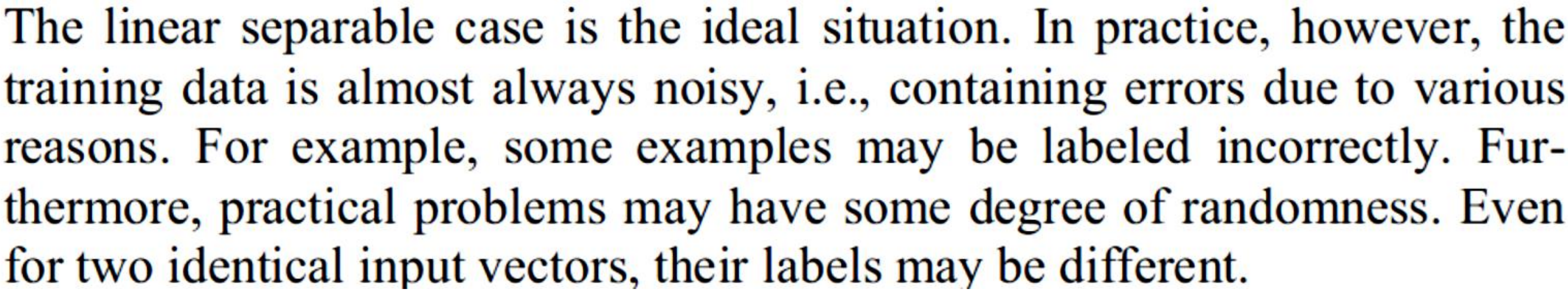


Testing: We apply (57) for classification. Given a test instance **z**, we classify it using the following:

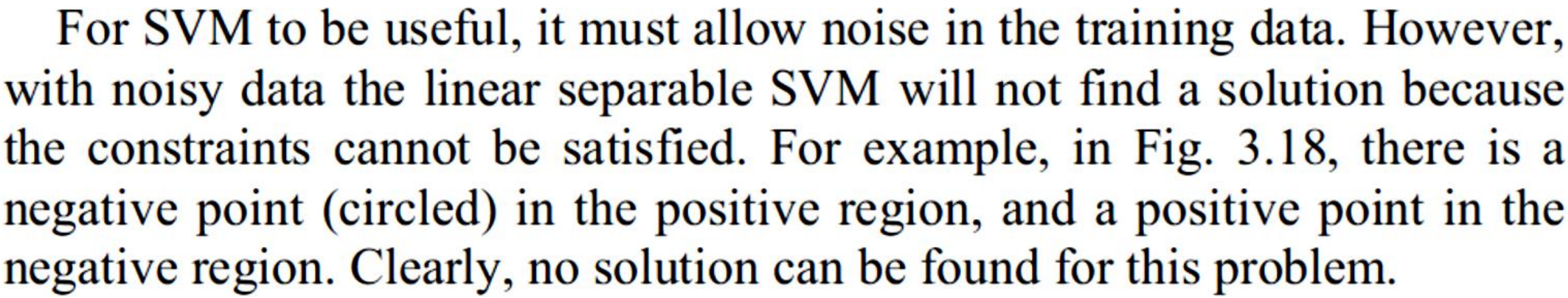
$$sign(\langle \mathbf{w}\cdot\mathbf{z}\rangle + b) = sign\left(\sum_{i\in sv} y_i\alpha_i\langle \mathbf{x}_i\cdot\mathbf{z}\rangle + b\right). \tag{58}$$

If (58) returns 1, then the test instance **z** is classified as positive; otherwise, it is classified as negative.

SVM - Inseparable Case

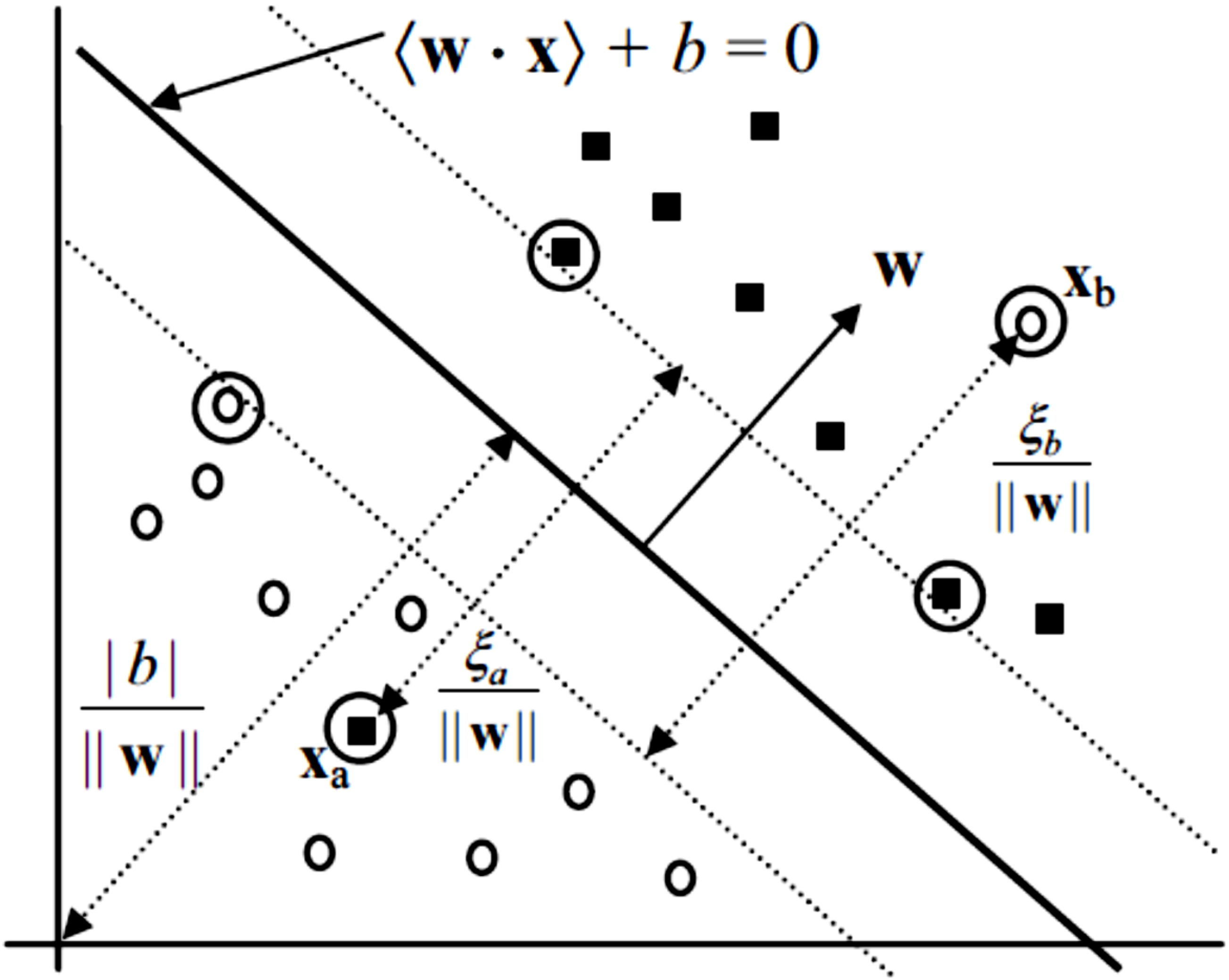


The linear separable case is the ideal situation. In practice, however, the training data is almost always noisy, i.e., containing errors due to various reasons. For example, some examples may be labeled incorrectly. Furthermore, practical problems may have some degree of randomness. Even for two identical input vectors, their labels may be different.



For SVM to be useful, it must allow noise in the training data. However, with noisy data the linear separable SVM will not find a solution because the constraints cannot be satisfied. For example, in Fig. 3.18, there is a negative point (circled) in the positive region, and a positive point in the negative region. Clearly, no solution can be found for this problem.

SVM - Inseparable Case


$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
$\mathbf{w}$
$\mathbf{x}_b$
$\frac{\xi_b}{\|\mathbf{w}\|}$
$\frac{|b|}{\|\mathbf{w}\|}$
$\frac{\xi_a}{\|\mathbf{w}\|}$
$\mathbf{x}_a$

The non-separable case: $\mathbf{x}_a$ and $\mathbf{x}_b$ are error data points

SVM - Inseparable Case

Recall that the primal for the linear separable case was:

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \qquad (59)$$

$$\text{Subject to}: y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., n.$$

To allow errors in data, we can relax the margin constraints by introducing **slack** variables, ξ_i (≥ 0) as follows:

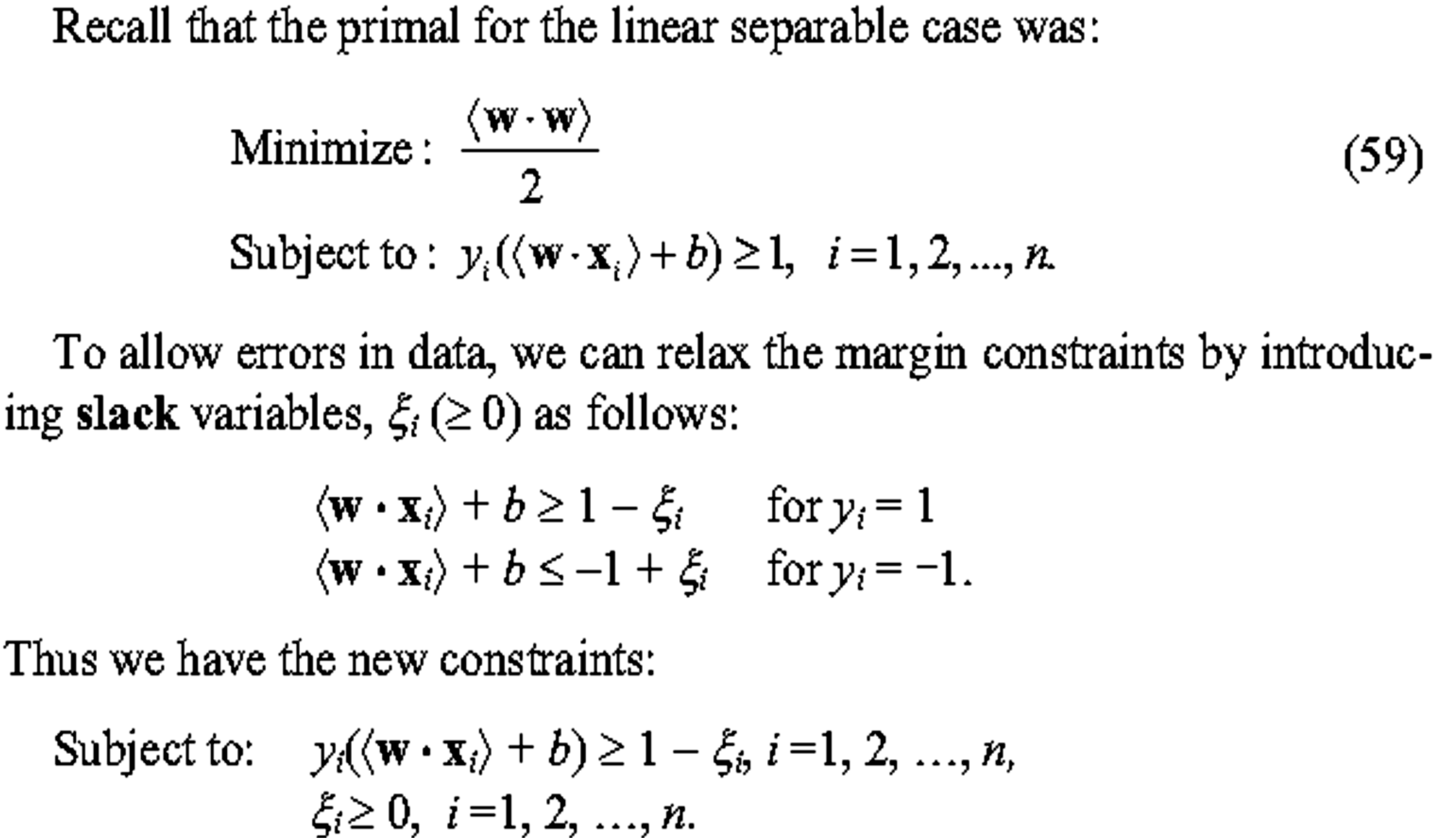
$$\begin{aligned} \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i \quad & \text{for } y_i = 1 \\ \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 + \xi_i \quad & \text{for } y_i = -1. \end{aligned}$$

Thus we have the new constraints:

$$\begin{aligned} \text{Subject to:} \quad & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \ i = 1, 2, \ldots, n, \\ & \xi_i \geq 0, \ i = 1, 2, \ldots, n. \end{aligned}$$

max (0, 1 - $y_i f(x_i)$)

SVM - Inseparable Case



Recall that the primal for the linear separable case was:

$$\text{Minimize}: \ \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \qquad (59)$$

$$\text{Subject to}: \ y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., n.$$

To allow errors in data, we can relax the margin constraints by introducing **slack** variables, ξ_i (≥ 0) as follows:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i \qquad \text{for } y_i = 1$$

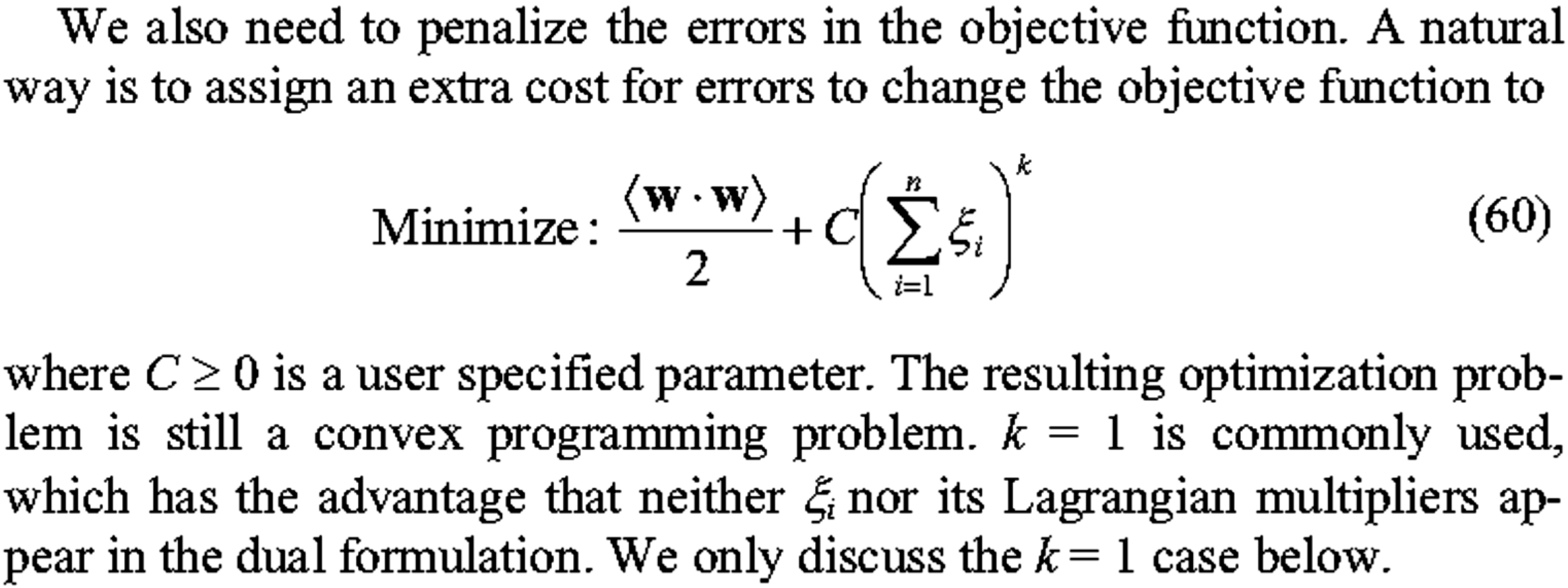
$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 + \xi_i \qquad \text{for } y_i = -1.$$

Thus we have the new constraints:

$$\text{Subject to:} \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \ i = 1, 2, \ldots, n,$$

$$\xi_i \geq 0, \ \ i = 1, 2, \ldots, n.$$

SVM - Inseparable Case



We also need to penalize the errors in the objective function. A natural way is to assign an extra cost for errors to change the objective function to

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\left(\sum_{i=1}^{n} \xi_i\right)^k \qquad (60)$$

where $C \geq 0$ is a user specified parameter. The resulting optimization problem is still a convex programming problem. $k = 1$ is commonly used, which has the advantage that neither ξ_i nor its Lagrangian multipliers appear in the dual formulation. We only discuss the $k = 1$ case below.

SVM - Inseparable Case

The new optimization problem becomes:

$$\begin{aligned} &\text{Minimize}: \ \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i-1}^{n} \xi_i \\ &\text{Subject to}: \ y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., n \\ &\qquad\qquad\quad \xi_i \geq 0, \quad i = 1, 2, ..., n. \end{aligned} \tag{61}$$

This formulation is called the **soft-margin SVM**. The primal Lagrangian (denoted by L_P) of this formulation is as follows

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i \tag{62}$$

where α_i, $\mu_i \geq 0$ are the **Lagrange multipliers**. The **Kuhn–Tucker conditions** for optimality are the following:

SVM - Inseparable Case

$$L_P = \frac{1}{2}\langle \mathbf{w}\cdot\mathbf{w}\rangle + C\sum_{i-1}^{n}\xi_i - \sum_{i-1}^{n}\alpha_i[y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1 + \xi_i] - \sum_{i-1}^{n}\mu_i\xi_i \qquad (62)$$

where α_i, $\mu_i \geq 0$ are the **Lagrange multipliers**. The **Kuhn–Tucker conditions** for optimality are the following:

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i\alpha_i x_{ij} = 0, \;\; j = 1, 2, \ldots, r \qquad (63)$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i\alpha_i = 0 \qquad (64)$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \;\; i = 1, 2, \ldots, n \qquad (65)$$

$$y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1 + \xi_i \geq 0, \;\; i = 1, 2, \ldots, n \qquad (66)$$

$$\xi_i \geq 0, \;\; i = 1, 2, \ldots, n \qquad (67)$$

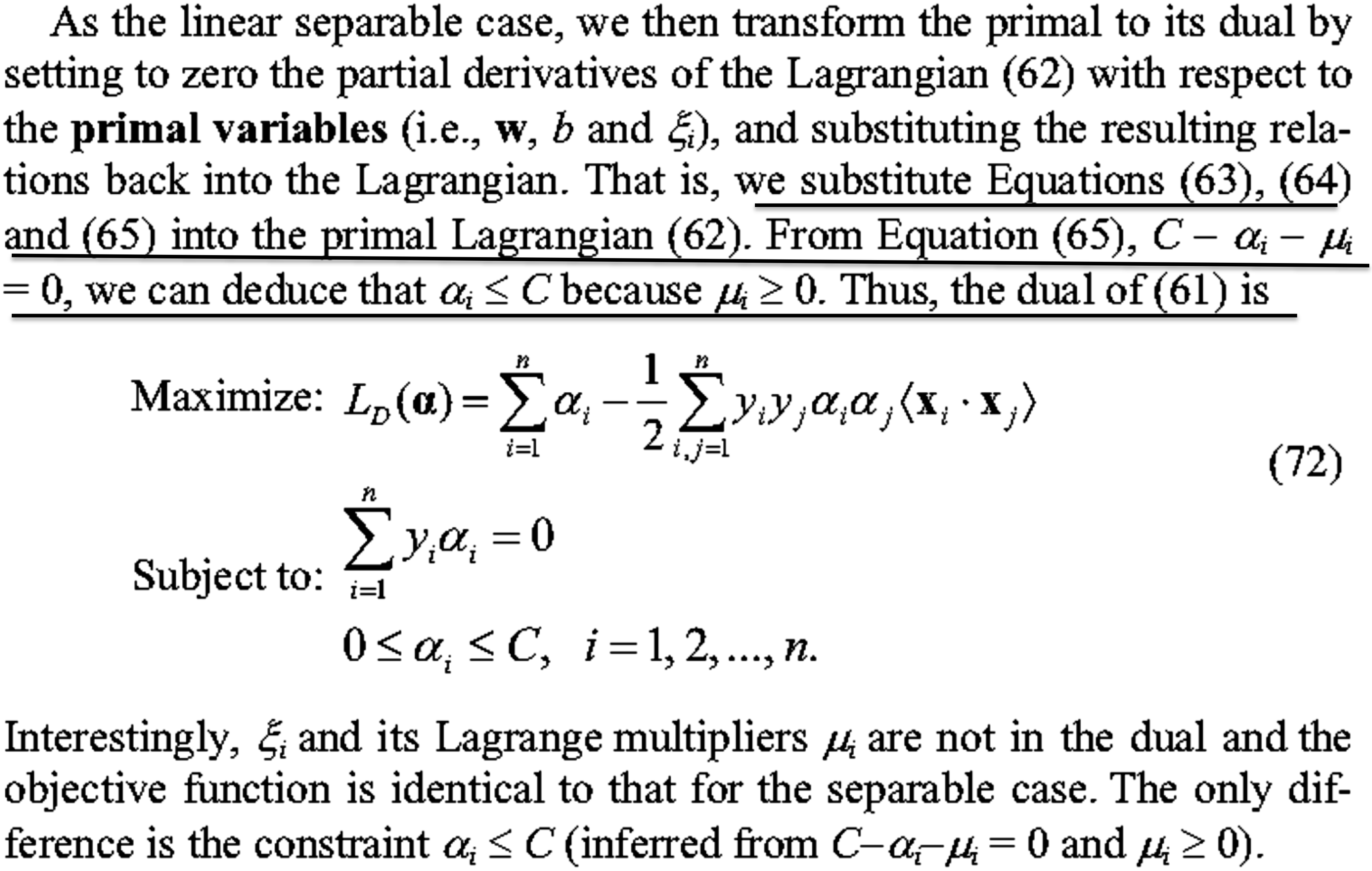
$$\alpha_i \geq 0, \;\; i = 1, 2, \ldots, n \qquad (68)$$

$$\mu_i \geq 0, \;\; i = 1, 2, \ldots, n \qquad (69)$$

$$\alpha_i(y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1 + \xi_i) = 0, \;\; i = 1, 2, \ldots, n \qquad (70)$$

$$\mu_i\xi_i = 0, \;\; i = 1, 2, \ldots, n \qquad (71)$$

SVM - Inseparable Case



As the linear separable case, we then transform the primal to its dual by setting to zero the partial derivatives of the Lagrangian (62) with respect to the **primal variables** (i.e., **w**, b and ξ_i), and substituting the resulting relations back into the Lagrangian. That is, we substitute Equations (63), (64) and (65) into the primal Lagrangian (62). From Equation (65), $C - \alpha_i - \mu_i = 0$, we can deduce that $\alpha_i \leq C$ because $\mu_i \geq 0$. Thus, the dual of (61) is

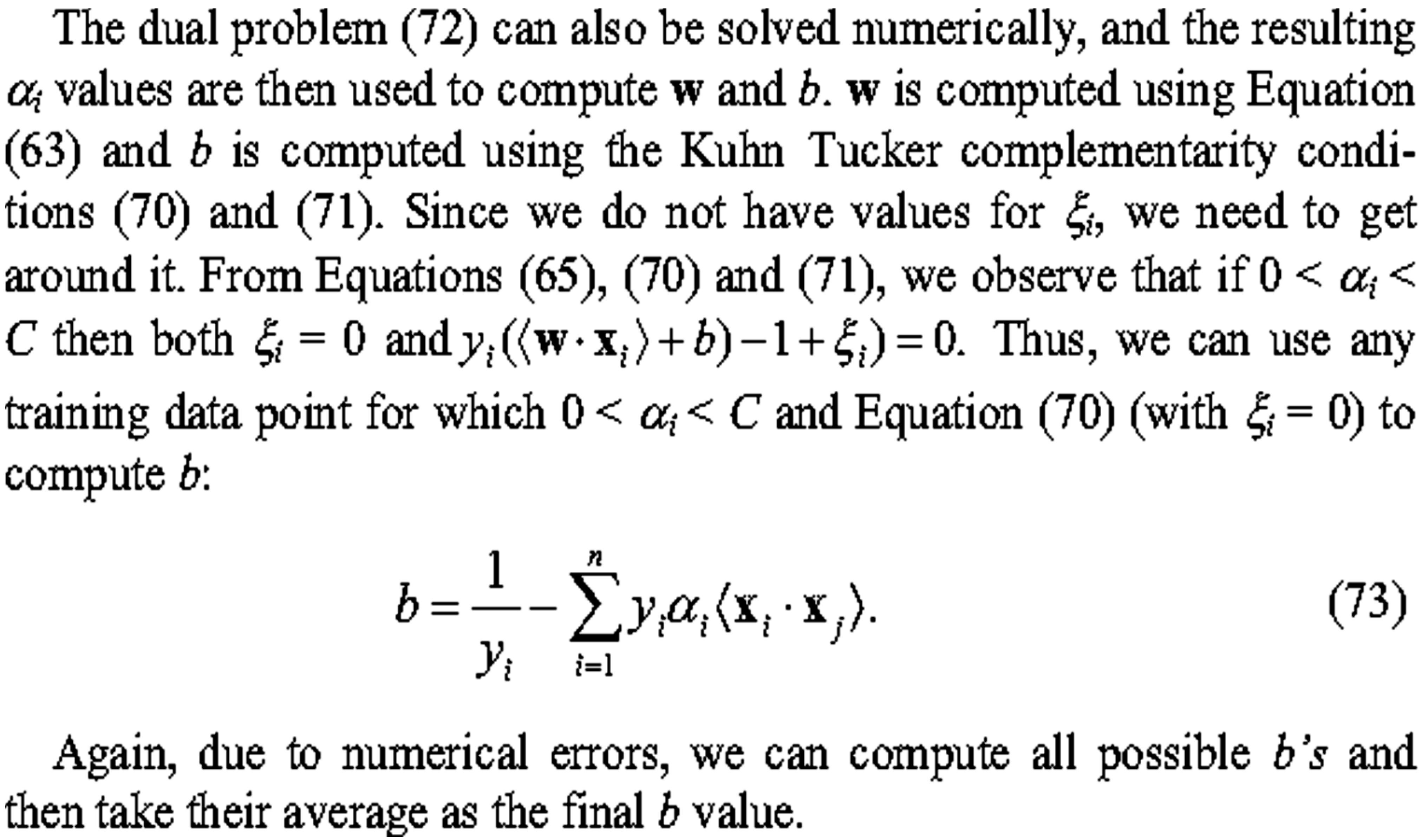
$$\text{Maximize: } L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \tag{72}$$

$$\text{Subject to: } \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n.$$

Interestingly, ξ_i and its Lagrange multipliers μ_i are not in the dual and the objective function is identical to that for the separable case. The only difference is the constraint $\alpha_i \leq C$ (inferred from $C - \alpha_i - \mu_i = 0$ and $\mu_i \geq 0$).

SVM - Inseparable Case



The dual problem (72) can also be solved numerically, and the resulting α_i values are then used to compute $\mathbf{w}$ and b. $\mathbf{w}$ is computed using Equation (63) and b is computed using the Kuhn Tucker complementarity conditions (70) and (71). Since we do not have values for ξ_i, we need to get around it. From Equations (65), (70) and (71), we observe that if $0 < \alpha_i < C$ then both $\xi_i = 0$ and $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0$. Thus, we can use any training data point for which $0 < \alpha_i < C$ and Equation (70) (with $\xi_i = 0$) to compute b:

$$b = \frac{1}{y_i} - \sum_{i=1}^{n} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \qquad (73)$$

Again, due to numerical errors, we can compute all possible b's and then take their average as the final b value.

Nonlinear SVM

Example 16: Suppose our input space is 2-dimensional, and we choose the following transformation (mapping):

$$(x_1, x_2) \mapsto ({x_1}^2, {x_2}^2, \sqrt{2}x_1x_2) \qquad (81)$$

The training example ((2, 3), −1) in the input space is transformed to the following training example in the feature space:

$$((4, 9, 8.5), -1).$$

■

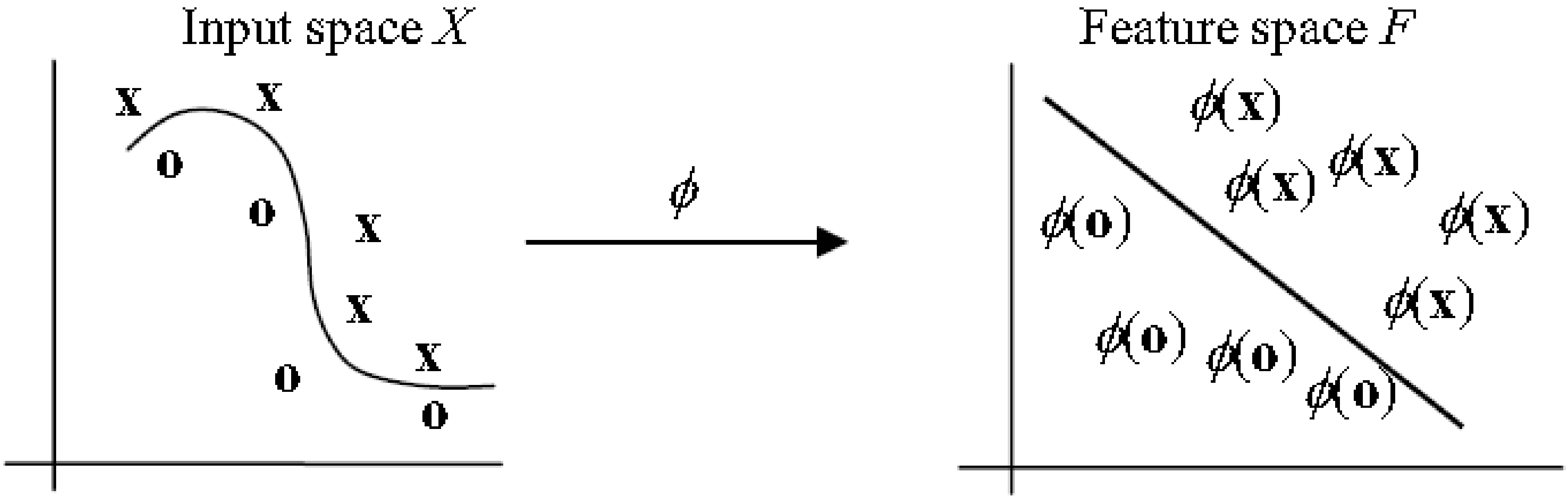
Nonlinear SVM

Thus, the basic idea is to map the data in the input space X to a feature space F via a nonlinear mapping ϕ,

$$\begin{aligned} \phi : X &\rightarrow F \\ \mathbf{x} &\mapsto \phi(\mathbf{x}). \end{aligned} \tag{76}$$

After the mapping, the original training data set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ becomes:

$$\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \ldots, (\phi(\mathbf{x}_n), y_n)\}. \tag{77}$$



Nonlinear SVM

- Consider the following mapping ϕ for an example $\mathbf{x} = \{x_1, \ldots, x_D\}$

$$\phi : \mathbf{x} \rightarrow \{x_1^2, x_2^2, \ldots, x_D^2, , x_1x_2, x_1x_2, \ldots, x_1x_D, \ldots\ldots, x_{D-1}x_D\}$$

- It's an example of a quadratic mapping
  - Each new feature uses a pair of the original features

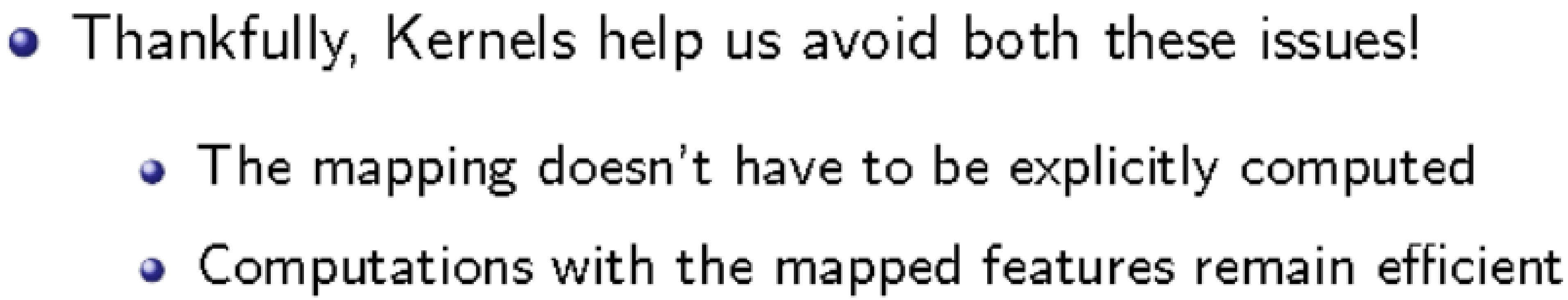
Nonlinear SVM

- Problem: Mapping usually leads to the number of features blow up!
  - Computing the mapping itself can be inefficient in such cases
  - Moreover, *using* the mapped representation could be inefficient too
    - e.g., imagine computing the similarity between two examples: $\phi(\mathbf{x})^\top \phi(\mathbf{z})$

$$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{z} \rangle + b \right).$$

$$\sum_{i=1}^{n} y_i \alpha_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b$$

Nonlinear SVM



- Thankfully, Kernels help us avoid both these issues!
  - The mapping doesn't have to be explicitly computed
  - Computations with the mapped features remain efficient

Nonlinear SVM

With the transformation, the optimization problem in (61) becomes

$$\text{Minimize:}\quad \frac{\langle \mathbf{w}\cdot\mathbf{w}\rangle}{2}+C\sum_{i-1}^{n}\xi_i \tag{78}$$

$$\text{Subject to:}\quad y_i(\langle \mathbf{w}\cdot\phi(\mathbf{x}_i)\rangle+b)\geq 1-\xi_i,\quad i=1,2,\ldots,n$$

$$\xi_i\geq 0,\quad i=1,2,\ldots,n$$

Its corresponding dual is

$$\text{Maximize:}\quad L_D=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}y_iy_j\alpha_i\alpha_j\langle\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)\rangle. \tag{79}$$

$$\text{Subject to:}\quad \sum_{i=1}^{n}y_i\alpha_i=0$$

$$0\leq\alpha_i\leq C,\quad i=1,2,\ldots,n.$$

The final decision rule for classification (testing) is

$$\sum_{i=1}^{n}y_i\alpha_i\langle\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x})\rangle+b \tag{80}$$

Nonlinear SVM

- Thus, if we have a way to compute the dot product $\langle \emptyset(x) \cdot \emptyset(z) \rangle$ in the feature space F using the input vectors x and z directly, then we would not need to know the feature vector $\emptyset(x)$ or even the mapping function Ø tself.

- In SVM, this is done through the use of kernel functions, denoted by K,

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle, \qquad (82)$$

- which are exactly the functions for computing dot products in the transformed feature space using input vectors x and z.

- An example of a kernel function is the polynomial kernel,

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^{d}. \qquad (83)$$

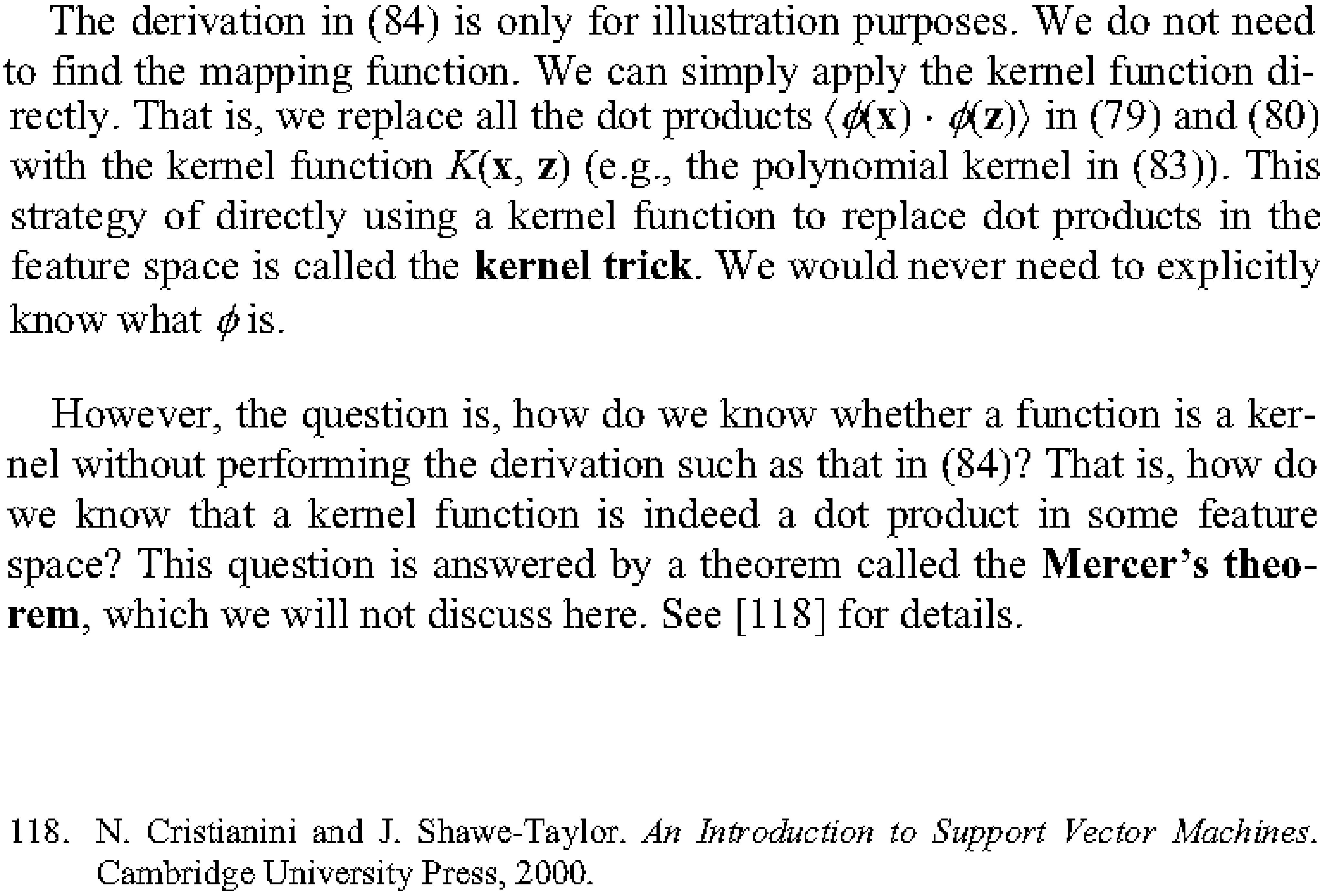
Nonlinear SVM

Example 17: Let us compute this kernel with degree d = 2 in a 2-dimensional space. Let $\mathbf{x} = (x_1, x_2)$ and $\mathbf{z} = (z_1, z_2)$.

$$\begin{aligned} \langle \mathbf{x}\cdot\mathbf{z}\rangle^2 &= (x_1z_1 + x_2z_2)^2 \\ &= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 \\ &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2)\cdot(z_1^2, z_2^2, \sqrt{2}z_1z_2)\rangle \\ &= \langle \phi(\mathbf{x})\cdot\phi(\mathbf{z})\rangle, \end{aligned} \tag{84}$$

where $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$, which shows that the kernel $\langle\mathbf{x}\cdot\mathbf{z}\rangle^2$ is a dot product in the transformed feature space. The number of dimensions in the feature space is 3. Note that $\phi(\mathbf{x})$ is actually the mapping function used in Example 16. Incidentally, in general the number of dimensions in the feature space for the polynomial kernel function $K(\mathbf{x}, \mathbf{z}) = \langle\mathbf{x}\cdot\mathbf{z}\rangle^d$ is $\binom{r+d-1}{d}$, which is a huge number even with a reasonable number (r) of attributes in the input space. Fortunately, by using the kernel function in (83), the huge number of dimensions in the feature space does not matter. ■

Nonlinear SVM

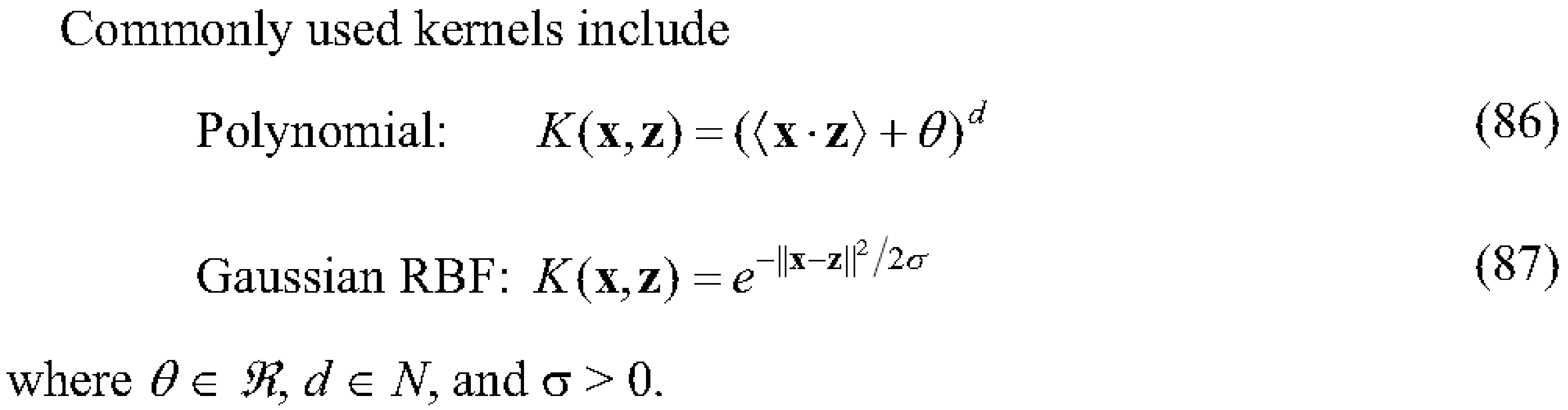


The derivation in (84) is only for illustration purposes. We do not need to find the mapping function. We can simply apply the kernel function directly. That is, we replace all the dot products $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ in (79) and (80) with the kernel function $K(\mathbf{x}, \mathbf{z})$ (e.g., the polynomial kernel in (83)). This strategy of directly using a kernel function to replace dot products in the feature space is called the **kernel trick**. We would never need to explicitly know what ϕ is.

However, the question is, how do we know whether a function is a kernel without performing the derivation such as that in (84)? That is, how do we know that a kernel function is indeed a dot product in some feature space? This question is answered by a theorem called the **Mercer's theorem**, which we will not discuss here. See [118] for details.

118. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Nonlinear SVM



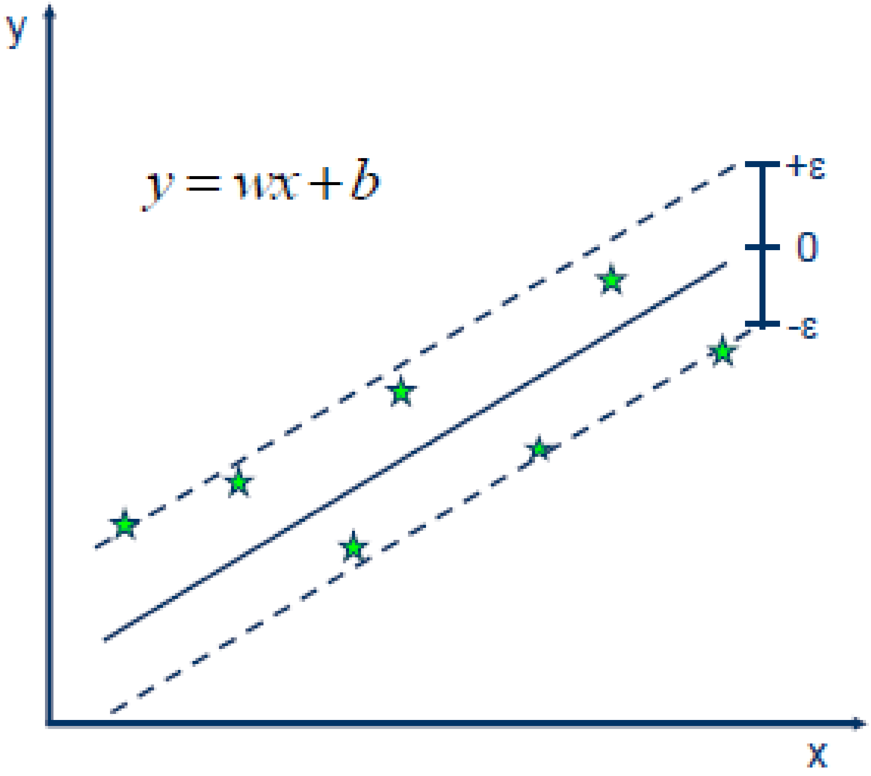
Commonly used kernels include

Polynomial: $$K(\mathbf{x},\mathbf{z}) = (\langle \mathbf{x} \cdot \mathbf{z} \rangle + \theta)^d \tag{86}$$

Gaussian RBF: $$K(\mathbf{x},\mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2 / 2\sigma} \tag{87}$$

where $\theta \in \Re$, $d \in N$, and $\sigma > 0$.

SVR



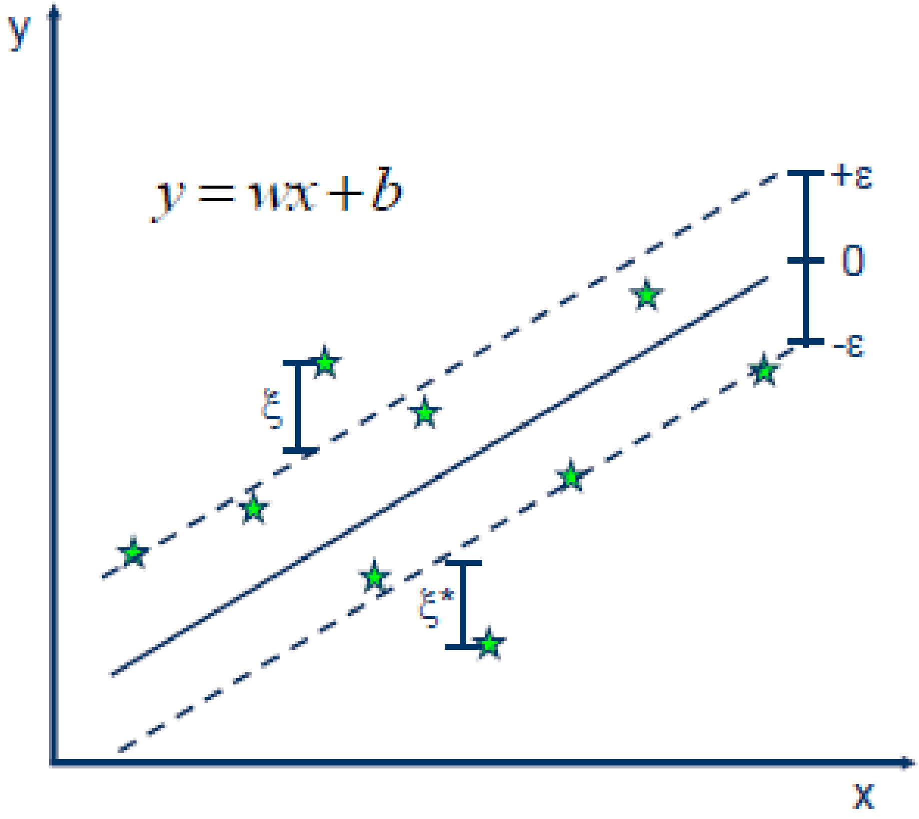
- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$

SVR



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Disclaimer

- Content of this presentation is not original and it has been prepared from various sources for teaching purpose.