$$
\begin{aligned}
&+\alpha_j s K_{jj} + \alpha_i K_{ij} + (\alpha_i + s\alpha_j)(K_{ij} - K_{jj}) \\
&= y_i((f(x_j) - y_j) - (f(x_i) - y_i)) + \alpha_i \chi
\end{aligned} \tag{10.77}
$$

Substituting the values of γ, χ, and ζ into Lemma 10.3 concludes the proof. ■

10.5.3 Regression

We proceed as in classification. We have the additional difficulty, however, that for each pair of patterns x_i and x_j we have four Lagrange multipliers $\alpha_i, \alpha_i^*, \alpha_j$, and α_j^*. Hence we must possibly consider up to three different pairs of solutions[10]. Let us rewrite the restricted optimization problem in regression as follows

$$
\begin{aligned}
\underset{\alpha_i,\alpha_i^*,\alpha_j,\alpha_j^*}{\text{minimize}} \quad & \tfrac{1}{2} \begin{bmatrix} \alpha_i - \alpha_i^* \\ \alpha_j - \alpha_j^* \end{bmatrix}^\top \begin{bmatrix} K_{ii} & K_{ij} \\ K_{ij} & K_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i - \alpha_i^* \\ \alpha_j - \alpha_j^* \end{bmatrix} \\
& + \begin{bmatrix} c_i \\ c_j \end{bmatrix}^\top \begin{bmatrix} \alpha_i - \alpha_i^* \\ \alpha_j - \alpha_j^* \end{bmatrix} + \varepsilon(\alpha_i + \alpha_i^* + \alpha_j + \alpha_j^*) \\
\text{subject to} \quad & 0 \le \alpha_l \le C_l \text{ and } 0 \le \alpha_l^* \le C_l^* \text{ for all } l \in \{i, j\} \\
& (\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = (\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}) + (\alpha_j^{\text{old}} - \alpha_j^{*\text{old}}) =: \gamma.
\end{aligned} \tag{10.78}
$$

Here c_i, c_j are suitably chosen constants depending solely on the differences $\alpha_i - \alpha_i^*$ and $\alpha_j - \alpha_j^*$. One can check that

$$
c_i = -y_i + (f(x_i) - b - K_{ii}(\alpha_i - \alpha_i^*) - K_{ij}(\alpha_j - \alpha_j^*)) \tag{10.79}
$$

and c_j likewise. We deliberately keep the contribution due to ε separate since this is the only part where sums $\alpha_i + \alpha_i^*$ rather than differences enter the equations.

Eliminating α_j, α_j^*

As in the classification case we begin with the constraints on α_i and α_i^*. Due to the summation constraint in the optimization problem we obtain $(\alpha_i - \alpha_i^*) = \gamma - (\alpha_j - \alpha_j^*)$. Using the additional box constraints on α_i, α_i^* of (10.78) leads to

$$
L := \max(\gamma - C_j, -C_i^*) \le \alpha_i - \alpha_i^* \le \min(\gamma + C_j^*, C_i) =: H. \tag{10.80}
$$

This allows us to eliminate α_j, α_j^* and rewrite (10.78) in terms of $\beta := \alpha_i - \alpha_i^*$,

$$
\begin{aligned}
\underset{\beta}{\text{minimize}} \quad & \tfrac{1}{2}\beta^2(K_{ii} + K_{jj} - 2K_{ij}) + \beta(\gamma(K_{ij} - K_{jj}) + c_i - c_j) \\
& + \varepsilon(|\beta| + |\gamma - \beta|) \\
= \quad & \tfrac{1}{2}\beta^2\chi + \beta((f(x_i) - y_i) - (f(x_j) - y_j) - \chi(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}})) \\
& + \varepsilon(|\beta| + |\gamma - \beta|) \\
\text{subject to} \quad & L \le \beta \le H.
\end{aligned} \tag{10.81}
$$

10. The number of solutions is restricted to four due to the restriction that α_i and α_i^* (or analogously α_j and α_j^*) may never both be nonzero at the same time. In addition, the constraint that $\alpha_i - \alpha_i^* + \alpha_j - \alpha_j^* = \gamma$ rules out one of these remaining four combinations.

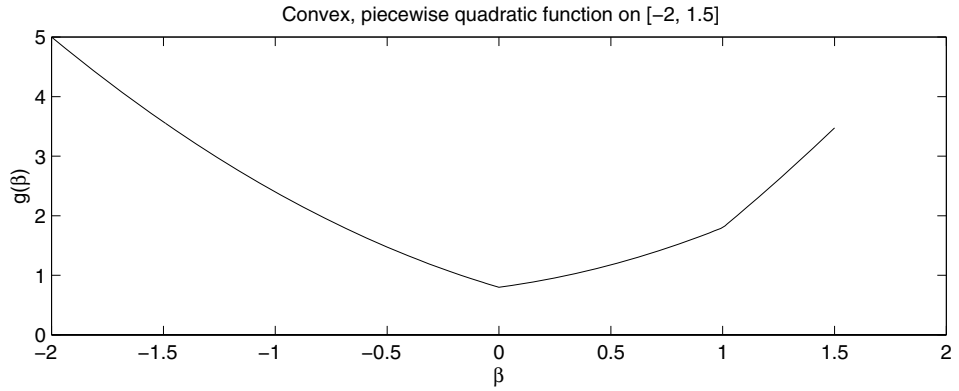

Figure 10.6 The minimum of this function occurs at $\beta = 0$ due to the change in $\varepsilon|\beta|$.

Here we used the $\chi = K_{ii} + K_{jj} - 2y_i y_j K_{ij}$, as in classification. The objective function is convex and piecewise quadratic on the three intervals

$$I_- := [L, \min(0,\gamma)],\ I_0 := [\min(0,\gamma), \max(0,\gamma)],\ I_+ := [\max(0,\gamma), H] \tag{10.82}$$

(for $\gamma = 0$ the interval I_2 vanishes). An example of such a function is given in Figure 10.6. One can check that the *unconstrained* minimum of the quadratic objective function (10.81), as defined on the intervals I_-, I_0, I_+, would be given by

Effect of Piecewise Convex Function

$$\left.\begin{matrix}\beta_- \\ \beta_0 \\ \beta_+\end{matrix}\right\} = (\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}) + \frac{1}{\chi}((f(x_j) - y_j) - (f(x_i) - y_i)) + \frac{\varepsilon}{\chi}\left\{\begin{matrix} 2 & \text{if } \beta \in I_- \\ 0 & \text{if } \beta \in I_0 \\ -2 & \text{if } \beta \in I_+ \end{matrix}\right. \tag{10.83}$$

For $\chi = 0$ the same considerations as in classification apply; the optimum is found on one of the interval boundaries. Furthermore, since (10.78) is convex all we now have to do is match up the solutions β_i with the corresponding intervals I_i.

For convenience we start with β_0 and I_0. If $\beta_0 \in I_0$ we have found the optimum. Otherwise we must continue our search in the direction in which β_0 exceeds I_0. Without loss of generality assume that this is I_+. Again, if $\beta_+ \in I_+$ we may stop. Otherwise we simply "clip" β_+ to the interval boundaries of I_+. Now we have to reconstruct α from β. Due to the box constraints and the fact that $\sum_i(\alpha_i - \alpha_i^*) = 0$ we obtain

Update is Independent of b

$$\begin{matrix} \alpha_i & = & \max(0,\beta), & \alpha_j & = & \max(0,\gamma-\beta) \\ \alpha_i^* & = & \max(0,-\beta), & \alpha_j^* & = & \max(0,-\gamma+\beta). \end{matrix} \tag{10.84}$$

In order to arrive at a complete SV regression or classification algorithm, we still need a way of selecting the patterns x_i, x_j and a method specifying how to update the constant offset b efficiently. Since most pattern selection methods use b as additional information to select patterns we will start with b.

10.5.4 Computing the Offset b and Optimality Criteria

We can compute b by exploiting the KKT conditions (see Theorem 6.21). For instance in classification; at the solution, the margin must be exactly 1 for Lagrange multipliers for which the box constraints are inactive. We obtain

$$y_i f(x_i) = y_i(\langle \mathbf{w}, \Phi(x)\rangle + b) = 1 \text{ for } \alpha_i \in (0, C_i) \tag{10.85}$$

Computing b via KKT Conditions

and likewise for regression

$$f(x_i) = \langle \mathbf{w}, \Phi(x)\rangle + b = y_i - \varepsilon \text{ for } \alpha_i \in (0, C_i) \tag{10.86}$$

$$f(x_i) = \langle \mathbf{w}, \Phi(x)\rangle + b = y_i + \varepsilon \text{ for } \alpha_i^* \in (0, C_i^*). \tag{10.87}$$

Hence, if all the Lagrange multipliers α_i were optimal, we could easily find b by picking any of the unconstrained α_i or α_i^* and solving (10.85), (10.86), or (10.87).

Unfortunately, during training, not all Lagrange multipliers will be optimal, since, if they were, we would already have obtained the solution. Hence, obtaining b by the aforementioned procedure is not accurate. We resort to a technique suggested by Keerthi et al. [291, 289] in order to overcome this problem.

Sets of KKT Violation and Satisfaction

For the sake of simplicity we start with the classification setting; we first split the patterns X into the following five sets:

$$\begin{aligned} I_0 &= \{i | \alpha_i \in (0, C_i)\} \quad I_{+,0} = \{i | \alpha_i = 0, y_i = +1\} \quad I_{+,C} = \{i | \alpha_i = C_i, y_i = +1\} \\ &\qquad\qquad I_{-,0} = \{i | \alpha_i = 0, y_i = -1\} \quad I_{-,C} = \{i | \alpha_i = C_i, y_i = -1\} \end{aligned}$$

Moreover we define

$$\begin{aligned} e_{\text{hi}} &:= \min_{i \in I_0 \cup I_{+,0} \cup I_{-,C}} f(x_i) - y_i \\ e_{\text{lo}} &:= \max_{i \in I_0 \cup I_{-,0} \cup I_{+,C}} f(x_i) - y_i. \end{aligned} \tag{10.88}$$

Since the KKT conditions have to hold for a solution we can check that this corresponds to $e_{\text{hi}} \geq 0 \geq e_{\text{lo}}$. For I_0 we have already exploited this fact in (10.85). Formally we can always satisfy the conditions for e_{hi} and e_{lo} by introducing two thresholds: $b_{\text{hi}} = b - e_{\text{hi}}$ and $b_{\text{lo}} = b - e_{\text{lo}}$. Optimality in this case corresponds to $b_{\text{hi}} \leq b_{\text{lo}}$. Additionally, we may use $\frac{1}{2}(b_{\text{up}} + b_{\text{lo}})$ as an improved estimate of b.

Choose Large Discrepancy with Large Possible Updates

The real benefit, however, comes from the fact that we may use e_{hi} and e_{lo} to choose patterns to focus on. The largest contribution to the discrepancy between e_{hi} and e_{lo} stems from that pair of patterns (i, j) for which

$$\text{discrepancy}(i, j) := (f(x_i) - y_i) - (f(x_j) - y_j) \text{ where } \begin{array}{l} i \in I_0 \cup I_{-,0} \cup I_{+,C} \\ j \in I_0 \cup I_{+,0} \cup I_{-,C} \end{array} \tag{10.89}$$

is largest. This is a reasonable strategy for the following reason: from Proposition 10.4 we conclude that the potential change in the variables α_i, α_j is largest if the discrepancy $(f(x_i) - y_i) - (f(x_j) - y_j)$ is largest. The only modification is that i and j are not chosen arbitrarily any more.

Finally, we obtain another stopping criterion. Instead of requiring that the violation of the KKT condition is smaller than some tolerance Tol we may require that $e_{\text{lo}} \leq e_{\text{hi}}$ holds with some tolerance; $e_{\text{lo}} \leq e_{\text{hi}} - 2\,\text{Tol}$. In addition, we will not consider patterns where discrepancy$(i, j) < 2\,\text{Tol}$. See [290] for more details and pseudocode of their implementation.

To adapt these ideas to regression we have to modify the sets I slightly. The change is needed since we have to add or subtract ε in a way that is very similar to our treatment of the classification case, where $y_i \in \{\pm 1\}$.

1. If $\alpha_i = 0$ at optimality we must have $f(x_i) - (y_i - \varepsilon) \geq 0$.
2. For $\alpha_i \in (0, C_i)$ we must have $f(x_i) - (y_i - \varepsilon) = 0$.
3. For $\alpha_i = C_i$ we get $f(x_i) - (y_i - \varepsilon) \leq 0$.

Analogous inequalities hold for α_i^*. As before we split the patterns X into several sets according to

$$I_0 = \{i|\alpha_i \in (0, C_i)\} \quad I_{+,0} = \{i|\alpha_i = 0\} \quad I_{+,C} = \{i|\alpha_i = C_i\}$$
$$I_0^* = \{i|\alpha_i^* \in (0, C_i^*)\} \quad I_{-,0} = \{i|\alpha_i^* = 0\} \quad I_{-,C} = \{i|\alpha_i^* = C_i^*\}$$

Computing b for Regression

and introduce $e_{\text{hi}}, e_{\text{lo}}$ by

$$e_{\text{hi}} := \min\left(\min_{i \in I_0 \cup I_{+,0}} f(x_i) - (y_i - \varepsilon), \min_{i \in I_0^* \cup I_{-,C}} f(x_i) - (y_i + \varepsilon)\right) \tag{10.90}$$

$$e_{\text{lo}} := \max\left(\max_{i \in I_0 \cup I_{+,C}} f(x_i) - (y_i - \varepsilon), \max_{i \in I_0^* \cup I_{-,0}} f(x_i) - (y_i + \varepsilon)\right). \tag{10.91}$$

The equations for computing a more robust estimate of b are identical to the ones in the classification case. Note that (10.90) and (10.91) are equivalent to the ansatz in [494], the only difference being that we sacrifice a small amount of numerical efficiency for a somewhat simpler definition of the sets I (some of them are slightly larger than in [494]) and the rules regarding which e_{hi} and e_{lo} are obtained (the cases $\alpha_i = 0, \alpha_i^* = C_i^*$ and $\alpha_i^* = 0, \alpha_i = C_i$ are counted twice).

Without going into further details, we may use a definition of a discrepancy like (10.89) and then choose patterns (i, j) for optimization where this discrepancy is largest. See the original work [494] for more details. Below we give a simpler (and slightly less powerful) reasoning.

10.5.5 Selection Rules

The previous section already indicated some ways to pick the indices (i, j) such that the decrease in the objective function is maximized. We largely follow the reasoning of Platt [409, Section 12.2.2]. See also the pseudocode (Algorithms 10.3 and 10.4).

We choose a two loop approach to maximizing the objective function. The outer loop iterates over all patterns violating the KKT conditions, or possibly over those where the threshold condition of the previous section (using e_{hi} and e_{lo}) is violated.

Usually we first loop only over those with Lagrange multipliers neither on the upper nor lower boundary. Once all of these are satisfied we loop over all patterns violating the KKT conditions, to ensure self consistency on the complete dataset. This solves the problem of choosing the index i.

Full Sweep for Noisy Data

It is sometimes useful, especially when dealing with noisy data, to iterate over the complete KKT violating dataset before complete self consistency on the subset has been achieved. Otherwise considerable computational resources are spent making subsets self consistent that are not globally self consistent. The trick is to perform a full sweep through the data once only less than, say, 10% of the non bound variables change[11].

Now to select j: To make a large step towards the minimum, one looks for large steps in α_i. Since it is computationally expensive to compute χ for all possible pairs (i, j) one chooses a heuristic to maximize the change in the Lagrange multipliers α_i and thus to maximize the absolute value of the numerator in the expressions (10.72) and (10.83). This means that we are looking for patterns with large differences in their relative errors $f(x_i) - y_i$ and $f(x_j) - y_j$. The index j corresponding to the maximum absolute value is chosen for this purpose.

Second Choice Hierarchy

If this heuristic happens to fail, in other words if little progress is made by this choice, all other indices j are looked at (this is what is called "second choice hierarchy" in [409]) in the following way.

1. All indices j corresponding to non-bound examples are looked at, searching for an example to make progress on.
2. In the case that the first heuristic was unsuccessful, all other samples are analyzed until an example is found where progress can be made.
3. If both previous steps fail, SMO proceeds to the next index i.

For a more detailed discussion and further modifications of these heuristics see [409] and [494, 291].

10.6 Iterative Methods

Many training algorithms for SVMs or similar estimators can be understood as iterative methods. Their main advantage lies in the simplicity with which they can be implemented. While not all of them provide the best performance (plain gradient descent in Section 10.6.1) and some may come with restrictions on the scope of applications (Lagrangian SVM in Section 10.6.2 can be used only for quadratic soft-margin loss), the algorithms presented in this section will allow practitioners to obtain first results in a very short time. Finally, Section 10.6.3 indicates how Support Vector algorithms can be extended to online learning problems.

11. This modification is not contained in the pseudocodes, however, its implementation should not pose any further problems. See also [494, 291] for further pseudocodes.

Algorithm 10.3 Pseudocode for SMO Classification

```
function TakeStep(i, j)
  if i = j then return 0
  s = y_i y_j
  if s = 1 then
    L = max(0, α_i + α_j − C_j)
    H = min(C_i, α_i + α_j)
  else
    L = max(0, α_i − α_j)
    H = min(C_i, C_j + α_i − α_j)
  end if
  if L = H then return 0
  χ = K_ii + K_jj − 2K_ij
  if χ > 0 then
    ᾱ = α_i + χ^{-1} y_i((f(x_j) − y_j) − (f(x_i) − y_i))
    ᾱ = min(max(ᾱ, L), H)
  else if y_i((f(x_j) − y_j) − (f(x_i) − y_i)) < 0 then
    ᾱ = H
  else
    ᾱ = L
  end if
  if |α_i − ᾱ| < ε(ε + ᾱ + α_i) then return 0
  α_j += s(α_i − ᾱ) and α_i = ᾱ (note: x += y means x = x + y)
  Update b
  Update f(x_1), ..., f(x_m)
  return 1
end function

function ExamineExample(i)
  KKT_i = H(α_i) max(0, y_i f(x_i) − 1) + H(1 − α_i) max(0, 1 − y_i f(x_i))
  if KKT_i > Tol then
    if Number of nonzero and non bound α_i > 1 then
      Find j with second choice heuristic
      if TakeStep(i, j) = 1 then return 1
    end if
    for all α_j > 0 and α_j < C_j (start at random point) do
      if TakeStep(i, j) = 1 then return 1
    end for
    for all remaining α_j do
      if TakeStep(i, j) = 1 then return 1
    end for
  end if
  return 0
end function

main SMO Classification(k, X, Y, ε)
  Initialize α_i, α_i^* = 0 and b = 0, make X, Y, α global variables
  ExamineAll = 1
  while NumChanged > 0 or ExamineAll = 1 do
    NumChanged = 0
    if ExamineAll = 1 then
      for all α_i do NumChanged += ExamineExample(i)
    else
      for all α_i > 0 and α_i < C_i do NumChanged += ExamineExample(i)
    end if
    if ExamineAll = 1 then
      ExamineAll = 0
    else if NumChanged = 0 then
      ExamineAll = 1
    end if
  end while
end main
```

.

Algorithm 10.4 Pseudocode for SMO Regression

function TakeStep(i, j)
  if $i = j$ **then** return 0
  $\gamma = (\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*)$
  $L = \max(\gamma - C_j, -C_i^*)$ and $H = \min(\gamma + C_j^*, C_i)$
  if $L = H$ **then** return 0
  $l = \min(\gamma, 0)$ and $h = \max(\gamma, 0)$
  $\chi = K_{ii} + K_{jj} - 2K_{ij}$
  if $\chi > 0$ **then**
    $\beta_0 = (\alpha_i - \alpha_i^*) + \chi^{-1}((f(x_i) - y_i) - (f(x_j) - y_j))$
    $\beta_+ = \beta_0 - 2\frac{\varepsilon}{\chi}$ and $\beta_- = \beta_0 + 2\frac{\varepsilon}{\chi}$.
    $\beta = \max(\min(\beta_0, h), l)$ (clip β_0 to I_0)
    if $\beta = h$ **then** $\beta = \max(\min(\beta_+, H), h)$
    if $\beta = l$ **then** $\beta = \max(\min(\beta_-, l), L)$
  else if $(f(x_i) - y_i) - (f(x_j) - y_j) < 0$ **then**
    $\beta = h$
    if $(f(x_i) - y_i) - (f(x_j) - y_j) + 2\varepsilon < 0$ **then** $\beta = H$
  else
    $\beta = l$
    if $(f(x_i) - y_i) - (f(x_j) - y_j) - 2\varepsilon > 0$ **then** $\beta = L$
  end if
  if $|\beta - (\alpha_i - \alpha_i^*)| < \varepsilon(\varepsilon + |\beta| + \alpha_i + \alpha_i^*)$ **then** return 0
  $\alpha_i = \max(\beta, 0)$, $\alpha_i^* = \max(-\beta, 0)$, and $\alpha_j = \max(0, \gamma - \beta)$, $\max(0, -\gamma + \beta)$
  Update b
  Update $f(x_1), \ldots, f(x_m)$
  return 1
end function

function ExamineExample(i)
$$\overline{\mathrm{KKT}}_i = \mathrm{H}(\alpha_i)\max(0, f(x_i) - (y_i - \varepsilon)) + \mathrm{H}(\alpha_i^*)\max(0, (y_i + \varepsilon) - f(x_i)) + \mathrm{H}(C_i - \alpha_i)\max(0, (y_i - \varepsilon) - f(x_i)) + \mathrm{H}(C_i^* - \alpha_i^*)\max(0, f(x_i) - (y_i + \varepsilon))$$
  if $\overline{\mathrm{KKT}}_i > \mathrm{Tol}$ **then**
    if Number of nonzero and non bound $\alpha_i > 1$ **then**
      Find j with second choice heuristic
      if TakeStep(i, j) = 1 **then** return 1
    end if
    for all $\alpha_j > 0$ **and** $\alpha_j < C_j$ (start at random point) **do**
      if TakeStep(i, j) = 1 **then** return 1
    end for
    for all remaining α_j **do**
      if TakeStep(i, j) = 1 **then** return 1
    end for
  end if
  return 0
end function

main SMO Regression(k, X, Y, ε)
  Initialize $\alpha_i, \alpha_i^* = 0$ and $b = 0$
  ExamineAll = 1
  while NumChanged > 0 **or** ExamineAll = 1 **do**
    NumChanged = 0
    if ExamineAll = 1 **then**
      for all α_i **do** NumChanged += ExamineExample(i)
    else
      for all $\alpha_i > 0$ **and** $\alpha_i < C_i$ **do** NumChanged += ExamineExample(i)
    end if
    if ExamineAll = 1 **then**
      ExamineAll = 0
    else if NumChanged = 0 **then**
      ExamineAll = 1
    end if
  end while
end main

.