SMO Algorithms
for Support Vector Machines without Bias Term

Michael Vogt, 18-Jul-2002

1 SMO Classification without Bias Term

Platt’s SMO algorithm [1] can be modified if the bias term \( b \) of the support vector machine does not need to be computed. This happens for some kernel functions, such as inhomogeneous polynomials or Gaussian bells, that provide an implicit bias. In this case the modification leads to a simpler and faster algorithm.

1.1 The Minimization Problem

The main advantage of an implicit bias is the fact that the summation constraint of the dual optimization problem does not exist, since the derivative of the primal Lagrangian with respect to \( b \) vanishes. The dual problem is then given by

\[
L = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_{ij} - \sum_{i=1}^{N} \alpha_i = \min_{\alpha},
\]

s.t. \( 0 \leq \alpha_i \leq C \) for \( i = 1, \ldots, N \),

where \( \alpha_i \) are the dual variables, i.e., the Lagrange multipliers of the primal problem. \( y_i \in \{-1, 1\} \) are the desired output values, and \( k_{ij} = k(x_i, x_j) \) are the kernel function values. The slack variables \( \xi_i \geq 0 \) do not appear in the dual problem.

In the original algorithm, two parameters have to be optimized per step to ensure that the solution obeys the summation constraint. The algorithm adapted to the problem defined above sequentially optimizes only one parameter \( \alpha_i \) per step, which makes it significantly easier and faster: On the one hand, the analytical solution is easier to compute for a one-dimensional problem. On the other hand, we do not need an (occasionally) time-consuming selection of the second parameter. The parameter to optimize is simply chosen by Platt’s “first choice heuristic”. The absence of the “second choice heuristic” raises the question, if an error cache (or function cache) is needed. However, simulations have shown that a cache speeds up the algorithm, since the cache values can also made use of for checking the KKT conditions.
1.2 The Analytical Solution

In each step of the SMO algorithm, \( L \) is minimized with respect to the chosen parameter, which can be done analytically. Without loss of generality, we assume that \( \alpha_1 \) will be optimized whereas \( \alpha_2, \ldots, \alpha_N \) are fixed. Additionally, it is assumed that \( y_i^2 = 1 \) (since \( y_i \in \{\pm 1\} \)) and \( k_{ij} = k_{ji} \). In that case we have to separate all terms of \( L \) containing \( \alpha_1 \):

\[
L = \frac{1}{2} \sum_{i=1}^{N} \alpha_i y_i \left( \alpha_1 y_i k_{i1} + \sum_{j=2}^{N} \alpha_j y_j k_{ij} \right) - \alpha_1 - \sum_{i=2}^{N} \alpha_i
\]

\[
= \frac{1}{2} \alpha_1 y_1 \sum_{i=1}^{N} \alpha_i y_i k_{i1} + \frac{1}{2} \sum_{i=1}^{N} \alpha_i y_i - \alpha_1 + A_1
\]

\[
= \frac{1}{2} \alpha_1 y_1 \sum_{i=1}^{N} \alpha_i y_i k_{i1} + \frac{1}{2} \alpha_1 y_1 \sum_{j=2}^{N} \alpha_j y_j k_{ij} + \frac{1}{2} \sum_{i=2}^{N} \sum_{j=2}^{N} \alpha_i \alpha_j y_i y_j k_{ij} - \alpha_1 + A_1 + A_2
\]

\[
= \frac{1}{2} \alpha_1^2 y_1^2 k_{11} + \frac{1}{2} \alpha_1 y_1 \sum_{i=2}^{N} \alpha_i y_i k_{i1} + \frac{1}{2} \alpha_1 y_1 \sum_{j=2}^{N} \alpha_j y_j k_{ij} - \alpha_1 + A_1 + A_2
\]

\[
= \frac{1}{2} k_{11} \alpha_1^2 + \left( y_1 \sum_{j=2}^{N} \alpha_j y_j k_{ij} - 1 \right) \alpha_1 + \text{const.}
\]

With

\[
\alpha_i = \alpha_i^{\text{old}} \quad \text{for} \quad i = 2, \ldots, N
\]

and the cached SVM output values

\[
f_i = \sum_{j=1}^{N} \alpha_j^{\text{old}} y_j k_{ij}
\]

follows

\[
\sum_{j=2}^{N} \alpha_j y_j k_{ij} = \sum_{j=2}^{N} \alpha_j^{\text{old}} y_j k_{ij} = f_i - \alpha_1^{\text{old}} y_1 k_{11}
\]

and therefore

\[
L = \frac{1}{2} k_{11} \alpha_1^2 + \left( y_1 f_i - k_{11} \alpha_1^{\text{old}} - 1 \right) \alpha_1 + \text{const.}
\]

\( L \) is minimal with respect to \( \alpha_1 \) if \( \partial L / \partial \alpha_1 = 0 \):

\[
\frac{\partial L}{\partial \alpha_1} = k_{11} \alpha_1 + (y_1 f_i - k_{11} \alpha_1^{\text{old}} - 1) \frac{1}{k_{11}} \quad \text{if} \quad \alpha_1^{\text{old}} \neq 0
\]

\[
k_{11} \left( \alpha_1 - \alpha_1^{\text{old}} \right) = 1 - y_1 f_i
\]

\[
\alpha_1 = \alpha_1^{\text{old}} + \frac{1 - y_1 f_i}{k_{11}}
\]

Since we want to use an error cache containing the error values \( E_i = f_i - y_i \) instead of the function values \( f_i \), we can exploit the equality

\[
1 - y_1 f_i = y_i^2 - y_i f_i = -y_i (f_i - y_i) = -y_i E_i
\]
(which relies on the fact that \( y_i \in \{ \pm 1 \} \) and therefore \( y_i^2 = 1 \)) to express the solution of the unconstrained minimization problem as
\[
\alpha_1 = \alpha_1^{\text{old}} - \frac{y_i E_i}{k_1},
\]
The solution of the constrained problem is found by “clipping” \( \alpha_1 \) to the interval \([0, C]\):
\[
\alpha_1^{\text{new}} = \min \{ \max \{ \alpha_1, 0 \}, C \}
\]

### 1.3 Checking the KKT Conditions

The Karush-Kuhn-Tucker conditions can be exploited to check the optimality of the current solution. Before each optimization step, the KKT conditions for \( \alpha_i \) are evaluated to decide if an update of \( \alpha_i \) is needed. This check takes only very few computation time if an error cache or function cache is used. The conditions for \( \alpha_i \) are
\[
(y_i f_i + \xi_i - 1) \cdot \alpha_i = 0 \\
(C - \alpha_i) \cdot \xi_i = 0,
\]
where \( \xi_i \) are the slack variables of the primal problem. At the optimal solution, the KKT conditions can be fulfilled in three different ways:

- **Case 1:** \( \alpha_i = 0 \) \( \Rightarrow \) \( C - \alpha_i = C \neq 0 \) \( \Rightarrow \) \( \xi_i = 0 \)
  \[
  \Rightarrow (y_i f_i + \xi_i - 1) \cdot \alpha_i = 0 \Rightarrow y_i f_i - 1 \geq 0
  \]
  The “\( \geq \)” sign in the last step follows from the constraints of the primal problem.

- **Case 2:** \( 0 < \alpha_i < C \) \( \Rightarrow \) \( C - \alpha_i > 0 \) \( \Rightarrow \) \( \xi_i = 0 \)
  \[
  \Rightarrow (y_i f_i + \xi_i - 1) \cdot \alpha_i = 0 \Rightarrow y_i f_i - 1 = 0
  \]

- **Case 3:** \( \alpha_i = C \) \( \Rightarrow \) \( C - \alpha_i = 0 \) \( \Rightarrow \) \( \xi_i \geq 0 \)
  \[
  \Rightarrow (y_i f_i + \xi_i - 1) \cdot \alpha_i = 0 \Rightarrow y_i f_i - 1 = -\xi_i \leq 0
  \]
  The “\( \geq \)” sign in the first step follows from the constraints of the primal problem.

Since we prefer an *error* cache (instead of a function cache), an optimal \( \alpha_i \) has to obey one of the following three conditions
\[
\alpha_i = 0 \quad \land \quad y_i E_i \geq 0 \\
\text{or} \quad 0 < \alpha_i < C \quad \land \quad y_i E_i = 0 \\
\text{or} \quad \alpha_i = C \quad \land \quad y_i E_i \leq 0.
\]
In other words, \( \alpha_i \) needs to be updated if

\[
\alpha_i < C \land y_i E_i < 0 \\
\text{or} \quad \alpha_i > 0 \land y_i E_i > 0
\]

SMO assumes \( \alpha_i \) to be optimal, if the KKT conditions are fulfilled with some precision \( \tau \). A typical value for classification tasks is \( \tau = 10^{-3} \). Therefore SMO performs an update only if

\[
\alpha_i < C \land y_i E_i < -\tau \\
\text{or} \quad \alpha_i > 0 \land y_i E_i > \tau
\]

### 1.4 Updating the Cache

As described above, a cache should be used to speed up the checking of the KKT conditions. If the parameter \( \alpha_1 \) has been optimized, a function cache can be updated in the following way:

\[
f_i = \sum_{j=1}^{N} \alpha_i y_j k_{ij} = \alpha_1 y_1 k_{i1} + \sum_{j=2}^{N} \alpha_j^{\text{old}} y_j k_{ij}
\]

With the abbreviation \( t_1 = (\alpha_1 - \alpha_1^{\text{old}}) y_1 \) (which is independent of \( i \)) the update rule for \( f_i \) can be written as

\[
f_i = f_i^{\text{old}} + t_1 k_{i1} \quad \text{for} \quad i = 1, \ldots, N.
\]

Since \( y_j \) is constant, the update of an error cache works in the same way:

\[
E_i = f_i - y_i = f_i^{\text{old}} + t_1 k_{i1} - y_i = E_i^{\text{old}} + y_i + t_1 k_{i1} - y_i = E_i^{\text{old}} + t_1 k_{i1}
\]

Platt’s original SMO algorithm updates only those \( E_i \) (or \( f_i \), respectively) with \( 0 < \alpha_i < C \), because these are the only parameters involved in the “second choice heuristic” [1]. When SMO enters the \texttt{takeStep} procedure only these parameters are retrieved form the cache for the KKT check – all other values are re-computed. This trick is not such crucial for the simplified algorithm, since there is no “second choice heuristic”. It provides only a small acceleration of the algorithm.

### 2 SMO Regression without Bias Term

SMO regression works similar to SMO classification. As the main difference, each data point requires two slack variables \( \xi_i \) and \( \xi_i^+ \) measuring the distance above and below the \( \varepsilon \)-tube [2]. Only one of both parameters is different from zero. Consequently, the solution has \( 2N \) parameters but at least half of them are zero. As in the classification case, the standard algorithm can be simplified if the use of implicit bias kernel functions makes an explicit bias dispensable.
2.1 The Minimization Problem

Also for regression an implicit bias eliminates the summation constrained of the dual problem, which is now given by

\[ L = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) k_{ij} + \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{N} y_i (\alpha_i - \alpha_i^*) = \min_{\alpha_i, \alpha_i^*}. \]

s.t. \[ 0 \leq \alpha_i, \alpha_i^* \leq C \quad \text{for} \quad i = 1, \ldots, N \]

with the dual variables \( \alpha_i \) and \( \alpha_i^* \). Also for the dual variables the equation \( \alpha_i \cdot \alpha_i^* = 0 \) holds true, \( y_i \) are the desired output values, and \( k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \) are the kernel function values. The primal slack variables \( \xi_i \) and \( \xi_i^* \) only appear in the KKT conditions but not in the dual problem.

The simplified algorithm chooses a sample \( i \) by Platt’s “first choice heuristic” and then performs a joint optimization of \( \alpha_i \) and \( \alpha_i^* \). An algorithm based on the original SMO method would have to optimize four parameters in one step to obey the summation constraint, as shown by Schölkopf and Smola [2].

2.2 The Analytical Solution

In contrast to classification (where \( \alpha_i^* \) does not exist), \( L \) is minimized with respect to both \( \alpha_i \) and \( \alpha_i^* \) in each step. Again, without loss of generality, we assume that \( \alpha_1 \) and \( \alpha_1^* \) are the variables to be optimized, and separate them in \( L \):

\[
L = \frac{1}{2} \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \left( (\alpha_1 - \alpha_1^*) k_{i1} + \sum_{j=2}^{N} (\alpha_j - \alpha_j^*) k_{ij} \right) + \varepsilon (\alpha_1 + \alpha_1^*) - y_1 (\alpha_1 - \alpha_1^*) - \sum_{i=2}^{N} y_i (\alpha_i - \alpha_i^*)
\]

\[
= \frac{1}{2} (\alpha_1 - \alpha_1^*) \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k_{i1} + \frac{1}{2} \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \sum_{j=2}^{N} (\alpha_j - \alpha_j^*) k_{ij} + \varepsilon (\alpha_1 + \alpha_1^*) - y_1 (\alpha_1 - \alpha_1^*) + A_1 + A_2
\]

\[
= \frac{1}{2} (\alpha_1 - \alpha_1^*)^2 k_{11} + \frac{1}{2} (\alpha_1 - \alpha_1^*) \sum_{i=2}^{N} (\alpha_i - \alpha_i^*) k_{i1} + \frac{1}{2} (\alpha_1 - \alpha_1^*) \sum_{j=2}^{N} (\alpha_j - \alpha_j^*) k_{1j} + \varepsilon (\alpha_1 + \alpha_1^*) - y_1 (\alpha_1 - \alpha_1^*) + A_1 + A_2\]

\[
= \frac{1}{2} (\alpha_1 - \alpha_1^*)^2 \left( \sum_{i=2}^{N} (\alpha_i - \alpha_i^*) k_{i1} - y_1 \right) (\alpha_1 - \alpha_1^*) + \varepsilon (\alpha_1 + \alpha_1^*) + A_1 + A_2 + A_3.
\]

With

\[ \alpha_i = \alpha_i^\text{old} \quad \text{and} \quad \alpha_i^* = \alpha_i^\text{old}^* \quad \text{for} \quad i = 2, \ldots, N \]
and the cached SVM output values

\[ f_i = \sum_{j=1}^{N} (\alpha_j^{\text{old}} - \alpha_j^{*\text{old}})k_{ij} \]

follows

\[ \sum_{j=2}^{N} (\alpha_j - \alpha_j^{*})k_{ij} = \sum_{j=2}^{N} (\alpha_j^{\text{old}} - \alpha_j^{*\text{old}})k_{ij} = f_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} \]

and therefore

\[
L = \frac{1}{2}k_{11}(\alpha_1 - \alpha_1^{*})^2 + (f_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - y_1)(\alpha_1 - \alpha_1^{*}) + \varepsilon(\alpha_1 + \alpha_1^{*}) + \text{const.}
\]

\[
= \frac{1}{2}k_{11}\alpha_1^2 + (f_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - y_1 + \varepsilon)\alpha_1
\]

\[
+ \frac{1}{2}k_{11}\alpha_1^{*2} - (f_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - y_1 - \varepsilon)\alpha_1^{*} + \text{const.}
\]

This result is achieved because only one of the variables \(\alpha_1\) and \(\alpha_1^{*}\) can be different from zero, which implies \(\alpha_1 \cdot \alpha_1^{*} = 0\). Again, as we usually want to employ an error cache instead of a function cache, \(L\) reads as

\[
L = \frac{1}{2}k_{11}\alpha_1^{2} + \left( E_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} + \varepsilon \right)\alpha_1
\]

\[
+ \frac{1}{2}k_{11}\alpha_1^{*2} - \left( E_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - \varepsilon \right)\alpha_1^{*} + \text{const.}
\]

The solution of the unconstrained problem is found by evaluating the conditions \(\partial L / \partial \alpha_1 = 0\), and \(\partial L / \partial \alpha_1^{*} = 0\), respectively. Since there are no “cross terms”, \(\alpha_1\) and \(\alpha_1^{*}\) can be determined independently:

\[
\frac{\partial L}{\partial \alpha_1} = k_{11}\alpha_1 + \left( E_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} + \varepsilon \right) \overset{!}{=} 0
\]

\[
k_{11}\alpha_1 = -E_1 + (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - \varepsilon
\]

\[
\alpha_1 = \alpha_1^{\text{old}} - \alpha_1^{*\text{old}} - \frac{E_1 + \varepsilon}{k_{11}}
\]

\(\alpha_1^{*}\) is computed in the same way:

\[
\frac{\partial L}{\partial \alpha_1^{*}} = k_{11}\alpha_1^{*} - \left( E_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - \varepsilon \right) \overset{!}{=} 0
\]

\[
k_{11}\alpha_1^{*} = E_1 - (\alpha_1^{\text{old}} - \alpha_1^{*\text{old}})k_{11} - \varepsilon
\]

\[
\alpha_1^{*} = -(\alpha_1^{\text{old}} - \alpha_1^{*\text{old}}) + \frac{E_1 - \varepsilon}{k_{11}} = -\alpha_1 - \frac{2\varepsilon}{k_{11}}
\]

As mentioned above, \(L\) does not contain any “cross terms” which implies that the minimum of the constrained optimization problem can be found again by clipping \(\alpha_1\) and \(\alpha_1^{*}\) to the interval \([0, C]\):

\[
\alpha_1^{\text{new}} = \min \{ \max \{ \alpha_1, 0 \}, C \}
\]

\[
\alpha_1^{*\text{new}} = \min \{ \max \{ \alpha_1^{*}, 0 \}, C \}
\]
2.3 Checking the KKT Conditions

Like in the classification case, the Karush-Kuhn-Tucker conditions are exploited for an optimality check. Here we have to consider the conditions for $\alpha_i$

\[
(\varepsilon + \xi_i + f_i - y_i) \cdot \alpha_i = 0 \\
(C - \alpha_i) \cdot \xi_i = 0
\]

and for $\alpha_i^*$

\[
(\varepsilon + \xi_i^* - f_i + y_i) \cdot \alpha_i^* = 0 \\
(C - \alpha_i^*) \cdot \xi_i^* = 0
\]

Starting with $\alpha_i$ we have again to consider three different ways to obey the KKT conditions in the optimum:

- **Case 1**: $\alpha_i = 0 \quad \Rightarrow \quad C - \alpha_i = C \neq 0 \quad \Rightarrow \quad \xi_i = 0$
  
  \[
  \Rightarrow \quad (\varepsilon + \xi_i + f_i - y_i) \cdot \alpha_i = 0 \quad \Rightarrow \quad \varepsilon + f_i - y_i \geq 0
  \]

  The “$\geq$” sign in the last step follows from the constraints of the primal problem.

- **Case 2**: $0 < \alpha_i < C \quad \Rightarrow \quad C - \alpha_i > 0 \quad \Rightarrow \quad \xi_i = 0$
  
  \[
  \Rightarrow \quad (\varepsilon + \xi_i + f_i - y_i) \cdot \alpha_i = 0 \quad \Rightarrow \quad \varepsilon + f_i - y_i = 0
  \]

- **Case 3**: $\alpha_i = C \quad \Rightarrow \quad C - \alpha_i = 0 \quad \Rightarrow \quad \xi_i \geq 0$
  
  \[
  \Rightarrow \quad (\varepsilon + \xi_i + f_i - y_i) \cdot \alpha_i = 0 \quad \Rightarrow \quad \varepsilon + f_i - y_i = -\xi_i \leq 0
  \]

  The “$\geq$” sign in the first step follows from the constraints of the primal problem.

Using (as usual) the error values $E_i = f_i - y_i$ we can summarize the conditions for an optimal solution $\alpha_i$ by

\[
\alpha_i = 0 \quad \land \quad \varepsilon + E_i \geq 0
\]

or

\[
0 < \alpha_i < C \quad \land \quad \varepsilon + E_i = 0
\]

or

\[
\alpha_i = C \quad \land \quad \varepsilon + E_i \leq 0.
\]

In the same way we find the conditions for $\alpha_i^*$ as

\[
\alpha_i^* = 0 \quad \land \quad \varepsilon - E_i \geq 0
\]

or

\[
0 < \alpha_i^* < C \quad \land \quad \varepsilon - E_i = 0
\]

or

\[
\alpha_i^* = C \quad \land \quad \varepsilon - E_i \leq 0.
\]
As $\alpha_i$ and $\alpha_i^*$ are jointly optimized, the solution is sub-optimal with respect to both variables if

$$
\alpha_i < C \quad \land \quad \varepsilon + E_i < 0 \\
\text{or} \quad \alpha_i > 0 \quad \land \quad \varepsilon + E_i > 0 \\
\text{or} \quad \alpha_i^* < C \quad \land \quad \varepsilon - E_i < 0 \\
\text{or} \quad \alpha_i^* > 0 \quad \land \quad \varepsilon - E_i > 0
$$

SMO assumes $\alpha_i$ and $\alpha_i^*$ to be optimal, if the KKT conditions are fulfilled with some precision $\tau$. In contrast to classification tasks (where the output values are $\pm 1$), a fixed $\tau$ is not suitable for regression problems. It should rather be dependent from the magnitude of the output values, e.g., $\tau = \max \{|y_1|, \ldots, |y_N|\} \cdot 10^{-3}$. SMO performs a joint update of $\alpha_i$ and $\alpha_i^*$ if

$$
\alpha_i < C \quad \land \quad \varepsilon + E_i < -\tau \\
\text{or} \quad \alpha_i > 0 \quad \land \quad \varepsilon + E_i > \tau \\
\text{or} \quad \alpha_i^* < C \quad \land \quad \varepsilon - E_i < -\tau \\
\text{or} \quad \alpha_i^* > 0 \quad \land \quad \varepsilon - E_i > \tau
$$

### 2.4 Updating the Cache

Also for regression problems a cache should be used for quickly checking the KKT conditions. The update of the function values works in the following way:

$$
f_i = \sum_{j=1}^{N} (\alpha_j - \alpha_j^*) k_{ij} = (\alpha_1 - \alpha_1^*) k_{1i} + \sum_{j=2}^{N} (\alpha_j^{\text{old}} - \alpha_j^{\text{old}}) k_{ij}
$$

$$
= (\alpha_1 - \alpha_1^*) k_{1i} + (f_i^{\text{old}} - (\alpha_1^{\text{old}} - \alpha_1^{\text{old}}) k_{1i}) = f_i^{\text{old}} + (\alpha_1 - \alpha_1^* - \alpha_1^{\text{old}} + \alpha_1^{*\text{old}}) k_{1i}
$$

With the auxiliary variable $t_1 = \alpha_1 - \alpha_1^* - \alpha_1^{\text{old}} + \alpha_1^{*\text{old}}$, which is independent from $i$, the update equation can be written as

$$
f_i = f_i^{\text{old}} + t_1 k_{1i} \quad \text{for} \quad i = 1, \ldots, N,
$$

or, if an error cache is employed, as

$$
E_i = E_i^{\text{old}} + t_1 k_{1i} \quad \text{for} \quad i = 1, \ldots, N.
$$

### References
