Supplementary Materials for
Two-variable Block Dual Coordinate Descent Methods for
Large-scale Linear Support Vector Machines

I. Implementation Issues in Solving the Sub-problem (22)

In Algorithm 3, the most expensive operations are on calculating

\[ \nabla_i f(\alpha) = y_i w^T x_i - 1, \quad (i) \]
\[ \nabla_j f(\alpha) = y_j w^T x_j - 1, \quad (ii) \]
\[ Q_{ij} = y_i y_j x_i^T x_j. \quad (iii) \]

For sparse data, we find that (iii) is more expensive than either (i) or (ii). Because \( w \) is a
dense vector, in implementing (i) or (ii), we go through \( x_i \)'s or \( x_j \)'s non-zero entries and
conduct products between them and \( w \)'s corresponding entries. That is,

\[ w^T x = \sum_{t: x_i \neq 0} w_t x_t. \]

In contrast, for (iii) we must go through both \( x_i \)'s and \( x_j \)'s non-zero entries:

\[ x_i^T x_j = \sum_{t_1, t_2} (x_i)_{t_1} (x_j)_{t_2}, \]

where the summation is over the following set.

\[ \{(t_1, t_2) | (x_i)_{t_1} \neq 0, (x_j)_{t_2} \neq 0, t_1 = t_2\}. \]

This is clearly more complicated. To reduce the number of accesses of data, we find that
combining both (i) and (ii) with (iii) can reduce the computational time. The following
setting computes (i)-(iii) all together.

\[ \textbf{while } t_1 < n \textbf{ or } t_2 < n \textbf{ then} \]
  \[ \textbf{if } t_1 = t_2 \textbf{ then} \]
  \[ w^T x_i \leftarrow w^T x_i + w_{t_1}(x_i)_{t_1} \]
  \[ w^T x_j \leftarrow w^T x_j + w_{t_2}(x_j)_{t_2} \]
  \[ x_i^T x_j \leftarrow x_i^T x_j + (x_1)_{t_1} (x_j)_{t_2} \]
  \[ t_1 \leftarrow \text{next}(x_i), t_2 \leftarrow \text{next}(x_j) \]
  \[ \textbf{else if } t_1 > t_2 \textbf{ then} \]
  \[ w^T x_j \leftarrow w^T x_j + w_{t_2}(x_j)_{t_2} \]
  \[ t_2 \leftarrow \text{next}(x_j) \]
  \[ \textbf{else} \]
  \[ w^T x_i \leftarrow w^T x_i + w_{t_1}(x_i)_{t_1} \]
\( t_1 \leftarrow \text{next}(x_i) \)
\end{array}
\end{equation}

where \( \text{next}(x) \) is the next non-zero index, and \( n \) is the number of features. Clearly, \( x_i \) and \( x_j \) are now accessed only once.

II. Some Differences Between CD Methods for Linear and Kernel SVM

We point out a difference in solving (53) between the CD method for linear SVM. For kernel, Chang and Lin (2011), as mentioned in Section 2.4, consider a greedy working set selection by using the gradient information, so their selected set satisfies

\[ -y_i y_j \nabla_i f(\alpha) + \nabla_j f(\alpha) \neq 0. \tag{iv} \]

Thus, even if

\[ Q_{ii} - 2y_i y_j Q_{ij} + Q_{jj} = 0, \tag{v} \]

the minimum of (54)

\[ \frac{-y_i y_j \nabla_i f(\alpha) + \nabla_j f(\alpha)}{Q_{ii} - 2y_i y_j Q_{ij} + Q_{jj}} = \infty \text{ or } -\infty \tag{vi} \]

can be easily handled under the IEEE floating-point standard. However, for linear SVM, because of a random selection, (iv) does not hold and \( 0/0 \) may occur. It can be easily seen that if

\[ -y_i y_j \nabla_i f(\alpha) + \nabla_j f(\alpha) = 0, \]

then the minimum of (54) is

\[ d_j = 0. \]

Therefore, the selected pair is not useful to reduce the function value. We can conduct a simple check on (iv) before solving the two-variable sub-problem.

III. Experiments on \( l_1 \)-loss SVM

We present results in Figures 1 and 2 by using \( C = 1 \) and 8,192, respectively. Similar observations to those for \( l_2 \)-loss SVM can be made.

References

Figure 1: A comparison between one-variable and two-variable CD for $l_1$ loss with $C = 1$. For each set, $x$-axis in the upper sub-figure is the number of CD steps, while $x$-axis in the lower sub-figure is the running time (in seconds).
Figure 2: A comparison between one-variable and two-variable CD for $l_1$ loss with $C = 8,192$. For each set, $x$-axis in the upper sub-figure is the number of CD steps, while $x$-axis in the lower sub-figure is the running time (in seconds).