EE6550 Machine Learning, Spring 2016

A Supplement of SMO Algorithm for HW#2 Programming Problem

Professor Chung-Chin Lu
Department of Electrical Engineering, National Tsing Hua University
April 27, 2016

This article is a detailed description of the sequential minimal optimization (SMO) algorithm to help students implement this efficient algorithm in HW# 2 programming problem for a kernel-based SVM-learning algorithm $\mathbb{A}$ for binary classification.

The Kuhn-Tucker conditions :

On page 58 of the Lecture 4: Support Vector Machines, any feasible point $(\mathbf{w}, b, \eta)$, i.e.,

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \eta_i, \ \forall\ i \in [1, m], \tag{1}$$
$$\eta_i \geq 0, \ \forall\ i \in [1, m], \tag{2}$$

which satisfies the Kuhn-Tucker necessary conditions

$$\mathbf{w} = \sum_{i=1}^{m} \lambda_i c(\mathbf{x}_i)\mathbf{x}_i, \tag{3}$$
$$0 = \sum_{i=1}^{m} \lambda_i c(\mathbf{x}_i), \tag{4}$$
$$C = \lambda_i + \mu_i, \ \forall\ i \in [1, m], \tag{5}$$
$$\lambda_i(c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \eta_i) = 0, \ i \in [1, m], \tag{6}$$
$$\mu_i \eta_i = 0, \ \forall\ i \in [1, m], \tag{7}$$
$$\lambda_i \geq 0, \ \forall\ i \in [1, m], \tag{8}$$
$$\mu_i \geq 0, \ \forall\ i \in [1, m]. \tag{9}$$

is a global minimum solution of the primal problem of SVM.

Consequences of the Kuhn-Tucker conditions (1)-(9) : From (5), (8) and (9), there are three cases to consider:

1. If $\lambda_i = 0$, then $\mu_i = C - \lambda_i = C$ by (5) and then $\eta_i = 0$ by (7) so that

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \ \text{ by (1).}$$

2. If $0 < \lambda_i < C$, then by (6),

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \eta_i = 0.$$

Since $\mu_i = C - \lambda_i > 0$ by (5), we have $\eta_i = 0$ by (7) and then

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0.$$

3. If $\lambda_i = C$, by (6),

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \eta_i = 0.$$

Since $\mu_i = C - \lambda_i = 0$ by (5), we have $\eta_i \geq 0$ by (2) and then

$$c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \leq 0.$$

The quantity $c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1$ can be computed as

$$R_i = c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - c(\mathbf{x}_i)^2 = c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b - c(\mathbf{x}_i)) = c(\mathbf{x}_i)E_i \tag{10}$$

where $E_i = \mathbf{w} \cdot \mathbf{x}_i + b - c(\mathbf{x}_i)$ is the prediction error. To summarize, the Kuhn-Tucker conditions imply:

$$\begin{aligned} \lambda_i = 0 &\Rightarrow R_i \geq 0, \\ 0 < \lambda_i < C &\Rightarrow R_i = 0, \\ \lambda_i = C &\Rightarrow R_i \leq 0. \end{aligned}$$

Thus in the following two cases, the Kuhn-Tucker conditions are violated:

1. $\lambda_i < C$ and $R_i < 0$,
2. $\lambda_i > 0$ and $R_i > 0$.

Checking Kuhn-Tucker conditions (1)-(9) without using the shift b : As the Lagrangian dual problem of SVM does not solve for the shift b directly, the improvement proposed by Keerthi et al.[1] avoids the use of the shift b in checking the Kuhn-Tucker conditions. The quantity $c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1$ can also be written as

$$R_i = c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = c(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i - c(\mathbf{x}_i) + b) = c(\mathbf{x}_i)(F_i + b)$$

where

$$F_i = \mathbf{w} \cdot \mathbf{x}_i - c(\mathbf{x}_i). \tag{11}$$

Note that $E_i = F_i + b$ and then

$$E_i - E_j = F_i - F_j. \tag{12}$$

Now consequences of the Kuhn-Tucker conditions can be rewritten as

$$\begin{aligned} \lambda_i = 0 &\Rightarrow c(\mathbf{x}_i)(F_i + b) \geq 0, \\ 0 < \lambda_i < C &\Rightarrow c(\mathbf{x}_i)(F_i + b) = 0, \\ \lambda_i = C &\Rightarrow c(\mathbf{x}_i)(F_i + b) \leq 0. \end{aligned}$$

Let

$$\begin{aligned} I_0 &= \{i \in [1, m] \mid 0 < \lambda_i < C\}, \\ I_1 &= \{i \in [1, m] \mid c(\mathbf{x}_i) = +1,\ \lambda_i = 0\}, \\ I_2 &= \{i \in [1, m] \mid c(\mathbf{x}_i) = -1,\ \lambda_i = C\}, \\ I_3 &= \{i \in [1, m] \mid c(\mathbf{x}_i) = +1,\ \lambda_i = C\}, \\ I_4 &= \{i \in [1, m] \mid c(\mathbf{x}_i) = -1,\ \lambda_i = 0\}. \end{aligned}$$

[1] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy, "Improvements to platt's SMO algorithm for SVM classifier design," *Neural Computation*, 13(3):637–649, March 2001.

Then we have

$$i \in I_0 \cup I_1 \cup I_2 \Rightarrow F_i \geq b$$

and

$$i \in I_0 \cup I_3 \cup I_4 \Rightarrow F_i \leq b$$

which imply that

$$F_i \geq F_j \ \forall \ i \in I_0 \cup I_1 \cup I_2 \text{ and } \forall \ j \in I_0 \cup I_3 \cup I_4 \tag{13}$$

if the Kuhn-Tucker conditions hold. Define

$$\begin{aligned} B_{up} &= \min\{F_i : i \in I_0 \cup I_1 \cup I_2\}, \\ B_{low} &= \max\{F_j : j \in I_0 \cup I_3 \cup I_4\}. \end{aligned}$$

Then the Kuhn-Tucker conditions imply

$$B_{up} \geq B_{low}, \tag{14}$$

$$F_i \geq B_{low}, \ \forall \ i \in I_0 \cup I_1 \cup I_2, \tag{15}$$

$$F_j \leq B_{up}, \ \forall \ j \in I_0 \cup I_3 \cup I_4. \tag{16}$$

These comparisons do not use the shift b.

The SMO algorithm : The sequential minimal optimization (SMO) algorithm proposed by Platt[2] is an efficient iterative algorithm commonly used to solve the Lagrangian dual problem for kernel-based SVM:

$$\begin{array}{ll} \text{Maximize} & \theta(\lambda) = \sum_{i=1}^m \lambda_i - \frac{1}{2}\sum_{i,j=1}^m \lambda_i\lambda_j c(\omega_i)c(\omega_j)K(\omega_i,\omega_j) \\ \text{Subject to} & \lambda_i \geq 0, i = 1,\ldots,m \\ & C - \lambda_i \geq 0, i = 1,\ldots,m \\ & \sum_{i=1}^m \lambda_i c(\omega_i) = 0 \\ & \lambda \in \mathbb{R}^m. \end{array} \tag{17}$$

In each iteration, the SMO algorithm solves the Lagrangian dual problem which involves only two Lagrangian multipliers λ_i and λ_j, by fixing the values of other Lagrangian multipliers $\lambda_k, k \neq i, j$ to their most recently updated values, to update the values of the two multiplier λ_i and λ_j. We next describe the update rules of the SMO algorithm.

The update rules of the SMO algorithm : Let $\lambda_1^*, \ldots, \lambda_m^*$ be the most recently updated values of the Lagrangian multipliers $\lambda_1, \ldots, \lambda_m$. By fixing $\lambda_k, k \neq i, j$, to their most recently updated values $\lambda_k^*, k \neq i, j$, the dual problem in (17) becomes

$$\begin{array}{ll} \text{Maximize} & \Psi_1(\lambda_i,\lambda_j) = \lambda_i + \lambda_j - \frac{1}{2}\lambda_i^2 K_{ii} - \frac{1}{2}\lambda_j^2 K_{jj} - \lambda_i\lambda_j s_{ij} K_{ij} \\ & -\lambda_i c(\omega_i) v_i^* - \lambda_j c(\omega_j) v_j^* \\ \text{Subject to} & 0 \leq \lambda_i, \lambda_j \leq C \ \text{ and } \ \lambda_i + s_{ij}\lambda_j = \gamma_{ij}^*, \end{array} \tag{18}$$

where $K_{ln} = K(\omega_l, \omega_n) \ \forall \ l, n \in [1, m]$, $s_{ij} = c(\omega_i)c(\omega_j)$, $v_i^* = \sum_{k \neq i,j} \lambda_k^* c(\omega_k) K_{ik}$, $v_j^* = \sum_{k \neq i,j} \lambda_k^* c(\omega_k) K_{jk}$ and $\gamma_{ij}^* = \lambda_i^* + s_{ij}\lambda_j^*$. By substituting $\lambda_i = \gamma_{ij}^* - s_{ij}\lambda_j$, the dual problem in (18) becomes

$$\begin{array}{ll} \text{Maximize} & \Psi_2(\lambda_j) = \gamma_{ij}^* - s_{ij}\lambda_j + \lambda_j - \frac{1}{2}(\gamma_{ij}^* - s_{ij}\lambda_j)^2 K_{ii} - \frac{1}{2}\lambda_j^2 K_{jj} \\ & -(\gamma_{ij}^* - s_{ij}\lambda_j)\lambda_j s_{ij} K_{ij} - (\gamma_{ij}^* - s_{ij}\lambda_j) c(\omega_i) v_i^* - \lambda_j c(\omega_j) v_j^* \\ \text{Subject to} & 0 \leq \lambda_j \leq C \ \text{ and } \ 0 \leq \gamma_{ij}^* - s_{ij}\lambda_j \leq C. \end{array} \tag{19}$$

[2] J.C. Pratt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Research Technical Report MSR-TR-98-14, April 21, 1998.

Without the inequality constraints in (19), the λ_j^{new} which maximizes Ψ_2 is

$$\lambda_j^{new} = \lambda_j^* + c(\omega_j)\frac{F_i^* - F_j^*}{\eta_{ij}}, \tag{20}$$

where

$$\begin{aligned} F_i^* &= \sum_{k=1}^{m} \lambda_k^* c(\omega_k) K(\omega_k, \omega_i) - c(\omega_i), & (21)\\ F_j^* &= \sum_{k=1}^{m} \lambda_k^* c(\omega_k) K(\omega_k, \omega_j) - c(\omega_j), & (22)\\ \eta_{ij} &= K_{ii} + K_{jj} - 2K_{ij}. & (23) \end{aligned}$$

Since the kernel is PDS, we have $\eta_{ij} \geq 0$ and we have assumed that $\eta_{ij} > 0$ here. When $s_{ij} = c(\omega_i)c(\omega_j) = 1$, the inequality constraints in the optimization problem in (19) are

$$0 \leq \lambda_j \leq C \text{ and } \gamma_{ij}^* - C \leq \lambda_j \leq \gamma_{ij}^* \Leftrightarrow L = \max(0, \gamma_{ij}^* - C) \leq \lambda_j \leq \min(C, \gamma_{ij}^*) = H. \tag{24}$$

And when $s_{ij} = c(\omega_i)c(\omega_j) = -1$, the inequality constraints in the optimization problem in (19) are

$$0 \leq \lambda_j \leq C \text{ and } -\gamma_{ij}^* \leq \lambda_j \leq C - \gamma_{ij}^* \Leftrightarrow L = \max(0, -\gamma_{ij}^*) \leq \lambda_j \leq \min(C, C - {}^*\gamma_{ij}) = H. \tag{25}$$

Now the newly updated value of λ_j, i.e., the solution of the optimization problem with inequality constraints in (19), should be

$$\lambda_j^{new,clip} = \begin{cases} \lambda_j^{new}, & \text{if } L \leq \lambda_j^{new} \leq H, \\ L, & \text{if } \lambda_j^{new} < L, \\ H, & \text{if } \lambda_j^{new} > H, \end{cases} \tag{26}$$

which is called the "clipped" value of λ_j^{new} in (20). When $\eta_{ij} = K_{11} + K_{22} - 2K_{12} = 0$, the dual problem in (19) becomes

$$\begin{array}{ll} \text{Maximize} & \Psi_3(\lambda_j) = c(\omega_j)(F_i^* - F_j^*)\lambda_j \\ \text{Subject to} & 0 \leq \lambda_j \leq C \text{ and } 0 \leq \gamma_{ij}^* - s_{ij}\lambda_j \leq C. \end{array} \tag{27}$$

In this case, we have

$$\lambda_j^{new,clip} = \begin{cases} L, & \text{if } \Psi_3(L) \geq \Psi_3(H), \\ H, & \text{if } \Psi_3(L) < \Psi_3(H). \end{cases} \tag{28}$$

Now the newly updated value λ_i^{new} of λ_i together with $\lambda_j^{new,clip}$ is the solution of the optimization problem with inequality and equality constraints in (18) and must satisfy the equality

$$\lambda_i^{new} + s_{ij}\lambda_j^{new,clip} = \gamma_{ij}^* = \lambda_i^* + s_{ij}\lambda_j^*.$$

Thus the newly updated value of λ_i is

$$\lambda_i^{new} = \lambda_i^* + s_{ij}(\lambda_j^* - \lambda_j^{new,clip}). \tag{29}$$

Updating $F_k, 1 \leq k \leq m$ and weight vector w after a successful optimization iteration :

$$F_k^{new} = F_k^* + \Delta\lambda_i c(\omega_i) K(\omega_i, \omega_k) + \Delta\lambda_j c(\omega_j) K(\omega_j, \omega_k) \tag{30}$$

where

$$\Delta\lambda_j = \lambda_j^{new,clip} - \lambda_j^*, \tag{31}$$
$$\Delta\lambda_i = \lambda_i^{new} - \lambda_i^* = -s_{ij}\Delta\lambda_j. \tag{32}$$

And if the input space $\mathscr{I} \subseteq \mathbb{R}^N$, we have

$$\mathbf{w}^{new} = \mathbf{w}^* + \Delta\lambda_i c(\mathbf{x}_i)\mathbf{x}_i + \Delta\lambda_j c(\mathbf{x}_j)\mathbf{x}_j.$$

Heuristics for picking two items i and j for joint optimization : As stated in Platt's original paper[3], as long as SMO always optimizes and changes two Lagrange multipliers at every optimization iteration step and at least one of the Lagrange multipliers $\lambda_k, k \in [1, m]$, violated the Kuhn-Tucker conditions before the iteration step, then each optimization iteration step will decrease the objective function according to Osuna et al.'s theorem[4]. Convergence is thus guaranteed.

In order to speed up convergence, Platt provided heuristics to choose which two Lagrange multipliers to jointly optimize as follows:

- There are two separate choice heuristics: one for the first Lagrange multiplier and one for the second.
- The choice of the first heuristic provides the outer loop of the SMO algorithm. The outer loop first iterates over the entire training set, determining whether each item violates the Kuhn-Tucker conditions by checking (14)-(16). If an item violates the Kuhn-Tucker conditions, it is then eligible for optimization.
- After one pass through the entire training set, the outer loop iterates over all items whose Lagrange multipliers are neither 0 nor C (the non-boundary items), i.e., all item $i \in I_0$. Again, each item is checked against the Kuhn-Tucker conditions simply by checking (14)-(16) and violating items are eligible for optimization.
- The outer loop makes repeated passes over the non-bound examples until all of the non-boundary items obey the Kuhn-Tucker conditions (indeed, obey (14)-(16)) within a preset tolerance ϵ.
- The outer loop then goes back and iterates over the entire training set. The outer loop keeps alternating between single passes over the entire training set and multiple passes over the non-boundary subset I_0 until the entire training set obeys the Kuhn-Tucker conditions within ϵ, whereupon the algorithm terminates.
- Once a first Lagrange multiplier λ_i is chosen in the outer loop, SMO chooses the second Lagrange multiplier in the inner loop to tentatively maximize the size of

[3] J.C. Pratt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Research Technical Report MSR-TR-98-14, April 21, 1998.

[4] E. Osuna, R. Freund, and F. Girosi, "Improved training algorithm for support vector machines," *Proc. IEEE NNSP'97*.

the step taken during joint optimization by selecting a non-boundary λ_j which maximizes $|F_j^* - F_i^*|$. If this does not make any positive progress, i.e., makes a zero step size upon joint optimization of the two Lagrange multipliers, it starts a sequential scan through the non-boundary items, starting at a random position; if this fails too, it starts a sequential scan through all the items, also starting at a random position.

Initialization : We will set

- $\lambda_i^{ini} = 0$ for all $i \in [1, m]$;
- $F_i^{ini} = -c(\omega_i)$ for all $i \in [1, m]$.

Then we have

$$\begin{aligned} I_0^{ini} &= \emptyset, \\ I_1^{ini} &= \{i \in [1, m] \mid c(\omega_i) = +1, \lambda_i^{ini} = 0\}, \\ I_2^{ini} &= \emptyset, \\ I_3^{ini} &= \emptyset, \\ I_4^{ini} &= \{i \in [1, m] \mid c(\omega_i) = -1, \lambda_i^{ini} = 0\} \end{aligned}$$

and

$$\begin{aligned} B_{up}^{ini} &= \min\{F_i : i \in I_0 \cup I_1 \cup I_2\} = -1, \\ B_{low}^{ini} &= \max\{F_j : j \in I_0 \cup I_3 \cup I_4\} = +1. \end{aligned}$$

Termination : When the entire training set obeys the three consequences (14)-(16) of the Kuhn-Tucker conditions within the tolerance ϵ, the SMO algorithm terminates.