Numerical Optimization Project

Jaeik Jeon

May 2020

1. Binary hard-margin support vector machine

We first discuss a formulation of the hard-margin support vector machine.

Let $\mathcal{D}\{(\mathbf{x}_i, y_i)\}_{i=1}^m \in \mathbb{R}^m \times \{-1, 1\}$ be a training set, and we will define a hyperplane by a set:

$$H_{\mathbf{w},b} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}^T\mathbf{x} + b = 0\} \tag{1}$$

with some $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

We also assume that the training data is linearly separable, which implies

$$\exists \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R} \text{ such that } y_i(\mathbf{w}^T\mathbf{x}_i + b) > 0, \;\; i = 1, \dots, m \tag{2}$$

and the corresponding decision boundary $H_{\mathbf{w},b}$ is called a separating hyperplane.

The distance $\gamma_\mathbf{x}$ of a point $\mathbf{x}$ from a hyperplane $H_{\mathbf{w},b}$ is given by

$$\gamma_\mathbf{x} = \frac{|\langle \mathbf{x}, \mathbf{w}\rangle + b|}{||\mathbf{w}||}. \tag{3}$$

This is because, given a plane $\mathbf{w} \cdot \mathbf{y} + b = 0$ with a point $\mathbf{x}$, normed vector is given by $\mathbf{w}$, and the vector $\mathbf{y}$ from the plane to point $\mathbf{x}$ is given by $\boldsymbol{\gamma} = \mathbf{x} - \mathbf{y}$. By projecting $\boldsymbol{\gamma}$ onto $\mathbf{w}$ gives

$$\frac{|\langle \gamma, \mathbf{w}\rangle|}{||\mathbf{w}||} = \frac{|\langle \mathbf{x} - \mathbf{y}, \mathbf{w}\rangle|}{||\mathbf{w}||} = \frac{|\langle \mathbf{x}, \mathbf{w}\rangle + b|}{||\mathbf{w}||} \tag{4}$$

gives the result.

If a hyperplane $H_{\mathbf{w},b}$ separates the training set $\mathcal{D}$, we define its margin as

$$\gamma_\mathcal{D}(\mathbf{w}, b) := \min_{i=1}^m \gamma_{\mathbf{x}_i}, \tag{5}$$

and the margin of a point is defined as $\frac{\langle \mathbf{x}, \mathbf{w}\rangle + b}{||\mathbf{w}||}$, which can be positive or negative.

Given a training data $\mathcal{D}$, the best decision boundary would be the one that maximizes the margin $\gamma_\mathcal{D}(\mathbf{w}, b)$.

In order to achieve the separating hyperplane (the decision boundary) with the maximum margin, we can solve the following optimal problem:

$$\gamma_\mathcal{D} = \max_{\mathbf{w},b} \min_{1 \le i \le m} \left\{ \frac{y_i(\langle \mathbf{w}, \mathbf{x}\rangle + b)}{||\mathbf{w}||} : y_j(\langle \mathbf{w}, \mathbf{x}_j + b) > 0, j = 1, \dots, m \right\} \tag{6}$$

However, note that the decision boundary, parametrized by $\mathbf{w}, b$ is not unique (i.e. $\langle \mathbf{w}, \mathbf{x}\rangle + b = 0$ and $3\langle \mathbf{w}, \mathbf{x}\rangle + 3b = 0$ gives the same hyperplane).

Hence, we will choose $||\mathbf{w}||$ that makes $\gamma_\mathcal{D}(\mathbf{w}, b) = \frac{1}{||\mathbf{w}||}$. Then, the separating hyperplane now can be written as:

$$\begin{aligned}
\gamma_\mathcal{D} &= \max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} : \min_i \{y_i(\mathbf{w}^T\mathbf{x}_i + b)\} = 1, y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 0 \right\} \\
&= \max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} : y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1 \right\} \\
&= \frac{1}{\min_{\mathbf{w},b} \{||\mathbf{w}|| : y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1\}}
\end{aligned} \tag{7}$$

Hence, we have the following optimization problem to achieve the optimum margin:

$$\begin{aligned} &\min_{\mathbf{w},b}\frac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{such that } y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1,\quad i=1,\dots,m \end{aligned} \tag{8}$$

It is easily noticed that this problem is a convex quadratic objective with linear constraint.
Now, in order to derive the dual problem, we will breifly review the basic convex optimization theory.
Consider the following optimization problem:

$$\min f(\mathbf{x}) \text{ such that } f_i(\mathbf{x})\leq 0,\ i=1,\dots,n, \tag{9}$$

with $\mathbf{x} in \mathbb{R}^m$ and $f:\mathbb{R}^m\rightarrow\mathbb{R}$, and the inequality constraint functions $f_i:\mathbb{R}^m\rightarrow\mathbb{R},\ i=1,\dots,n.$ We will call $\mathbf{x}^*$ is optimal if it is a solution of the above optimization problem.
The Lagrangian of the above optimization problem (9) is

$$L(\mathbf{x},\boldsymbol{\alpha})=f(\mathbf{x})+\sum_{i=1}^{n}\alpha_i f_i(\mathbf{x}),\quad \alpha_i\geq 0, \tag{10}$$

with $\boldsymbol{\alpha}=(\alpha_1,\dots,\alpha_m)^T$, a Lagrange multiplier.
The Lagrangian dual function is given as

$$g(\boldsymbol{\alpha})=\inf_{\mathbf{x}}L(\mathbf{x},\boldsymbol{\alpha}), \tag{11}$$

and one can show that

$$g(\boldsymbol{\alpha})\leq f^* \tag{12}$$

Hence, we can again construct the Lagrange dual optimization problem:

$$\max g(\boldsymbol{\alpha}) \text{ such that } \alpha_i\geq 0\forall i=1,\dots n. \tag{13}$$

From the saddle point theorem, we have that: if a pair $(\mathbf{x}^*,\boldsymbol{\alpha}^*)$ exists such that

$$L(\mathbf{x}^*,\boldsymbol{\alpha})\leq L(\mathbf{x}^*,\boldsymbol{\alpha}^*)\leq L(\mathbf{x},\boldsymbol{\alpha}^*), \tag{14}$$

and satisfies the constraints, then the $\mathbf{x}^*$ is a solution to the optimization problem.
Then, since

$$f^*=L(\mathbf{x}^*,\boldsymbol{\alpha}^*)=\inf_{\mathbf{x}}L(\mathbf{x},\boldsymbol{\alpha}^*)\leq\max_{\boldsymbol{\alpha}\geq 0}\inf_{\mathbf{x}}L(\mathbf{x},\boldsymbol{\alpha})=\max_{\boldsymbol{\alpha}\geq 0}g(\boldsymbol{\alpha}), \tag{15}$$

combining with (12) gives $f^*=g^*$. Furthermore, we can easily derive the following the complementary slackness condition:

$$\alpha_i^* f_i(\mathbf{x}^*)=0,\ i=1,\dots,n, \tag{16}$$

from the identity

$$f(\mathbf{x}^*)=f(\mathbf{x}^*)+\sum_{i=1}^{n}\alpha_i^* f_i(\mathbf{x}^*),\quad \alpha_i^*\geq 0,\text{ and } f_i(\mathbf{x}^*)\leq 0 \tag{17}$$

From the above, we also see that $\mathbf{x}^*$ minimizes $L(\mathbf{x},\boldsymbol{\alpha}^T)$, which gives

$$\nabla_{\mathbf{x}}L(\mathbf{x}^*,\boldsymbol{\alpha})=\mathbf{0} \tag{18}$$

Then, by collecting the above derived conditions, we now state a set of optimality conditions (a.k.a KKT condition): If the optimization problem is convex then any points that satisfies the following conditions that have zero duality gap

$$\begin{aligned} \mathbf{f}(\mathbf{x}^*)&\leq 0 \\ \boldsymbol{\alpha}^*&\geq 0 \\ \alpha_i^* f_i(\mathbf{x}^*)&=0,\quad 1\leq i\leq n \\ \nabla_{\mathbf{x}}f(\mathbf{x}^*)+\sum_{i=1}^{n}\alpha_i^*\nabla_{\mathbf{x}}f_i(\mathbf{x}^*)&=\mathbf{0}. \end{aligned} \tag{19}$$

We will now derive the dual problem for the problem (8)

Constructing the Lagrangian gives:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \tag{20}$$

The dual form can be found as

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}, b, a) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \\ \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \end{aligned} \tag{21}$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{22}$$

By substituting this into the Lagrangian,

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i)^T \mathbf{x}_j - b\sum_{i=1}^{n} \alpha_i y_i. \tag{23}$$

From (22),

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i)^T \mathbf{x}_j \\ &= \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T D \boldsymbol{\alpha}, \quad D_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j \end{aligned} \tag{24}$$

Combining (24) with constraint $\alpha_i \geq 0$, we have the followubg dual optimization problem:

$$\begin{aligned} &\max_\alpha W(\alpha) = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T D \boldsymbol{\alpha} \\ &\text{such that } \alpha_i \geq 0, \quad i = 1, \ldots, n \\ &\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{25}$$

From the complementary slackness condition, we know that $\alpha_i^* > 0$ implies $f_i(w^*) = 0$.

That is, $y_i(\mathbf{w}^T\mathbf{x}_i)$ + b = 1, and the condition is "active". We will call these vectors as "support vectors" and the bias b^* can be calculated as

$$b^* = y_i - \mathbf{x}_i^T \mathbf{w}^* \tag{26}$$

with any support vector $\mathbf{x}_i$.

2. **Binary soft-margin support vector machine**

When the data set is not linearly separable, we can introduce the hinge loss, so that the optization problem is now become

$$\min_{\mathbf{w}, b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} \max\{0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\}, \tag{27}$$

where the scalar C is a tunable parameter that balance between margin and the loss.

Then, by introducing the slack variable ξ_i, the above problem is equivalent to

$$\begin{aligned} &\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} \xi_i \\ &\text{such that } \xi_i \geq 0, \quad 1 - y_i(w^T x_i + b) \leq \xi_i, \quad i = 1, \ldots, m \end{aligned} \tag{28}$$

Again, the problem is differentiable convex with the affine constraints.

The Lagrangian of the problem is

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \gamma_i\xi_i, \tag{29}$$

with some Lagrangian multiplier $\alpha_i \geq 0$, $\gamma_i \geq 0$.

Then, by exploiting the KKT condition above, we have

$$\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}^*) &= \mathbf{w}^* - \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i = 0 \\
\frac{\partial}{\partial b} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}^*) &= -\sum_{i=1}^{m} \alpha_i^* y_i = 0 \\
\frac{\partial}{\partial \xi_i} L(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}^*) &= C - \alpha_i^* - \gamma_i^* = 0, \quad i = 1, \dots, m.
\end{aligned} \tag{30}$$

Again, by plugging back to the Lagrangian and combining with the KKT conditions gives

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T D \boldsymbol{\alpha} \\
&\text{such that } \ \boldsymbol{\alpha}^T \mathbf{y} = 0, \quad 0 \leq \boldsymbol{\alpha} \leq C\mathbf{1}.
\end{aligned} \tag{31}$$

Note that only change by introducing the slack variable is in the constraint $0 \leq \boldsymbol{\alpha} \leq C\mathbf{1}$.

Hence, we can combine the following conditions:

$$\begin{aligned}
&\alpha_i^*(y_i(\mathbf{x}_i^T \mathbf{w}^* + b_-^* 1 + \xi_i^*) = 0 \\
&\gamma_i^* \xi_i^* = 0 \\
&\gamma_i^* = C - \alpha_i^* > 0 \quad i = 1, \dots, m,
\end{aligned} \tag{32}$$

so that we can calculate

$$b^* = y + i - \mathbf{x}_i^T \mathbf{w}^* \tag{33}$$

with any support vectors with $0 < \alpha_i^* < C$.

3. **Binary soft-margin kernel support vector machine**

We can use the kernel trick in order for nonlinear classifier.

We first reformulate the optimal hyperplane $H_{\mathbf{w}^*,b}$ as:

$$\begin{aligned}
H_{\mathbf{w}^*,b} &= \mathbf{w}^{*T}\mathbf{x} + b^* \\
&= \left(\sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i\right)^T \mathbf{x} + b^* \\
&= \sum_{i=1}^{n} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b^*.
\end{aligned} \tag{34}$$

Here, we can see that we do not need the exact data points, instead we just need their inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Accordingly, the matrix D in the equation (24) also only depends on the inner products of data points.

Thus, we can map the data to a more high dimensional space and formulate the decision bounday in this high dimensional space.

For example, let $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ be a map such that $(x, y) \mapsto (t_1, x_2, t_3)^T = (x^2, \sqrt{2}xy, y^2)^T$ Then, our decision boundary in $\mathbb{R}^3$ will be of the form

$$\begin{pmatrix} w_1 & w_2 & w_3 \end{pmatrix} \begin{pmatrix} x^2 \\ \sqrt{2}xy \\ y^2 \end{pmatrix} = 0, \tag{35}$$

which is an ellipse equation.

Then, mapping data by ϕ will result:

$$H_{\mathbf{w}^*,b} = \sum_{i=1}^{n} \alpha_i^* y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b^*, \quad D_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \tag{36}$$

However, it turns out that we don't actually need to know ϕ in the dual representation. Instead, we can use kernel function $K : \mathbb{R}^n \mathbb{R}^n \to \mathbb{R}$ associated to a feature map such that

$$K(x, y) = \langle \phi(x), \phi(y) \rangle, \quad x, y \in \mathbb{R}^n \tag{37}$$

Then, using the kernel trick, the optimization problem (25) can be reformulated as:

$$\begin{aligned} &\max_\alpha W(\alpha) = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T D \boldsymbol{\alpha}, \quad D_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{such that } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n \end{aligned} \tag{38}$$

where the prediction can be made by:

$$\hat{y} = sign\left(\sum_{i=1}^{n} \alpha_i^* y_i k(\mathbf{x}_i, \mathbf{x}) + b^* \right) \tag{39}$$

In this project, we will consider two kernel function i.e. RBF and polynomial kernel:

$$k_{poly}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^d, \quad k_{RBF}(\mathbf{x}, \mathbf{y}) = \exp(\gamma ||\mathbf{x} - \mathbf{y}||^2) \tag{40}$$

4. **Optimization algorithms for binary soft-margin kernel support vector machine**

   (a) **Interior Point Method with the Barrier Method.**
   We now describe, for the first optimization method for the binary soft-margin kernel support vector machine, the barrier method.
   We first consider the problem

$$\min t f_0(\mathbf{x}) + \phi(\mathbf{x}) \text{ subject to } A\mathbf{x} = \mathbf{b}, \tag{41}$$

   where $\phi(x) = -\sum_{i=1}^{m} \log(-f_i(x))$. Then, the optimal $\mathbf{x}^*(t)$ can be derived by:

Algorithm 1 Barrier method

```
Given $\mathbf{x} \in \mathcal{D}$, $t = t^{(0)} > 0,\ \mu > 1,\ \epsilon > 0$
while $\frac{m}{t}$ do
    Centering Step:
    Compute $x^*(t)$ by minimizing $f_0 + \phi$ subject to $Ax = b$, starting at x
    Update $x := x^*(t)$
    $t := \mu t$
end while
```

Then the centering step can be done by Newton step $\triangle_{nt}$, and associated dual variable by solving the following linear equations:

$$\begin{pmatrix} t\nabla^2 f_0(\mathbf{x}) + \nabla^2 \phi(\mathbf{x}) & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \triangle_{nt} \\ v_{nt} \end{pmatrix} = - \begin{pmatrix} t\nabla f_0(\mathbf{x}) + \nabla \phi(\mathbf{x}) \\ 0 \end{pmatrix} \tag{42}$$

Then, the exploiting the Newton's method with the line search by backtracking:
Applying the above barrier method to our optimization problem (38) we have:

$$\begin{aligned} &\min_{\alpha} F = \frac{1}{2} t \boldsymbol{\alpha}^T D \boldsymbol{\alpha} - t \boldsymbol{\alpha}^T \mathbf{1} - \sum_{i=1}^{m} \log \alpha_i - \sum_{i=1}^{m} \log(C\mathbf{1} - \mathbf{w}) \\ &\text{such that } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n \end{aligned} \tag{43}$$

Algorithm 2 Newton's method

```
Given a starting point $\mathbf{x} \in \mathcal{D}$, tol $< \epsilon$
while Stopping Criterian do
    Line search: choose the step size $t$ by backtracking line search
    Compute the Newton step $\triangle_{nt}$
    Update $x := x + t\triangle_{nt}$
end while
```

Then, the Newton step $\triangle_{nt}$ is:

$$\begin{pmatrix}\triangle_{nt}\\ v_{nt}\end{pmatrix} = -\begin{pmatrix}H & \mathbf{y}\\ \mathbf{y}^T & 0\end{pmatrix}^{-1}\begin{pmatrix}\nabla F\\ 0\end{pmatrix}, \tag{44}$$

where

$$\nabla F = tD\boldsymbol{\alpha} - t\mathbf{1} - \left(\frac{1}{\boldsymbol{\alpha}} + \frac{1}{\boldsymbol{\alpha} - C\mathbf{1}}\right), H = tD + \begin{pmatrix}\frac{1}{\alpha_1^2} & 0 & \dots & 0\\ 0 & \frac{1}{\alpha_2^2} & 0 & \vdots\\ \vdots & 0 & \ddots & 0\\ 0 & \dots & 0 & \frac{1}{\alpha_m^2}\end{pmatrix} \tag{45}$$

(b) **Simplified Sequential Minimal Optimization**

We will now briefly describe how the Simplified Sequential Minimal Optimization(SMO) algorithm works.

SMO is an algorithm that motivated from the coordinate ascent algorithm. The main difference is that we update two α_i and α_j simultaneously. Hence, the general optimization procedure is going to be starting with selecting some two pairs α_i and α_j and optimize $W(\alpha)$ w.r.t α_i and α_j while fixing $\alpha_k,\ k \neq i, j$.

There are many heuristics on choosing α_i and α_j in SMO, yet we iterate over all α_i, $i = 1, \dots, n$, until the KKT conditions are satisified. If there is no changes in α, we stop iterations.

Note that by simplifying the heuristics, global optimum is not gauranteed. We now state the update rules.

First, we find bounds L, H so that $L \leq \alpha_j \leq H$ making $0 \leq \alpha_j \leq C$. This can be done by:

$$\begin{cases} L = \max(0, \alpha_j - \alpha_i),\ H = min(C, C + \alpha_j - \alpha_i) & \text{if } y_i \neq y_j\\ L = \max(0, \alpha_j + \alpha_i - C),\ H = min(C, \alpha_j + \alpha_i) & \text{if } y_i = y_j\end{cases} \tag{46}$$

Then, α_j is updated by:

$$\alpha_j = \alpha_j - \frac{y_j - (E_i - E_j)}{\eta}, \tag{47}$$

where

$$\begin{aligned} E_k &= f(\mathbf{x}_k) - y_k\\ \eta &= 2\langle \mathbf{x}_i, \mathbf{x}_j\rangle - \langle \mathbf{x}_i, \mathbf{x}_i\rangle - \langle \mathbf{x}_j, \mathbf{x}_j\rangle \end{aligned} \tag{48}$$

which will be clipped so that it is $L \leq \alpha \leq H$:

$$\alpha_j = \begin{cases} H, & \alpha_j > H\\ \alpha_j, & L \leq \alpha_j \leq H\\ L, & \alpha_j < H\end{cases} \tag{49}$$

and update α_i by:

$$\alpha_i = \alpha_i + y_i y_j(\alpha_j^{(old)} - \alpha_j) \tag{50}$$

Finally, we update b as

$$b_i = \begin{cases} b_1, & 0 < \alpha_i < C\\ b_2, & 0 < \alpha_j < C\\ \frac{b_1 + b_2}{2}, & \text{otherwise,}\end{cases} \tag{51}$$

where

$$b_1 = b - E_i - y_i(\alpha_i = \alpha_i^{(old)})\langle \mathbf{x}_i^{(i)}, \mathbf{x}_j\rangle - y_j(\alpha_j - \alpha_j^{(old)}\langle \mathbf{x}_i, \mathbf{x}_j\rangle \tag{52}$$

$$b_2 = b - E_j - y_i(\alpha_i = \alpha_i^{(old)})\langle \mathbf{x}_i^{(i)}, \mathbf{x}_i\rangle - y_i(\alpha_j - \alpha_j^{(old)}\langle \mathbf{x}_j, \mathbf{x}_i\rangle \tag{53}$$

Algorithm 3 Simplified SMO (Ng, 2000)

```
Given C, tol, max_passes, $\mathcal{D}$, initialize $\boldsymbol{\alpha} = \mathbf{0}$, $b = 0$
while passes< max_passes do
  num_changed_alphas = 0
  for i in range(m) do
    Calculate $E_i$ (48)
    if $\{y_i E_i < -$tol and $\alpha_i < C\}$ or $\{y_i E_i > tol$ and $\alpha_i > 0\}$ then
      select $j \neq i$ randomly.
      Calculate $E_j$ (48)
      old_alpha_i, old_alpha_j = alpha[i], alpha[j]
      Compute L and H (46)
      if $L == H$ then
        Continue
      end if
      Calculate $\eta$ (48)
      if $\eta \geq 0$ then
        Comtinue
      end if
      Compute (47), (49)
      if |old_alpha_j - $\alpha_j$| < $1e-5$ then
        Continue
      end if
      Compute (50)
      Compute (52) and (53)
      Compute (51)
      num_changed_alphas + = 1
    end if
  end for
  if  num_changed_alphas == 0 then
    passes + = 1
  else
    passes = 0
  end if
end while
```

5. **Experiment with the MNIST binary hand-written digits data**

In this project, we will consider the MNIST binary hand-written digits data. In specific, we will find a separating hyperplane with maximum margin that separates digit zero and one.

Each image has a resolution 28×28, so we will flatten the matrix which becomes a vector with 784 features (i.e. $m = 784$. Figure 6 shows the 50 sample image that we are going to classify.

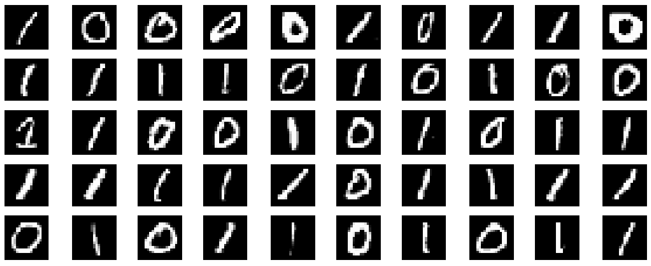

Figure 1: Digits 0 and 1 that we will classify for this project.

(a) **Tunable hyperparameters**

It is important to note that there are plenty of (hyper) parameter that must be tuned. For example, when use the barrier method, we need to choose the initial t, μ, and ρ, c for the line search parameter in Newton's method. The parameters used in Newton's method is chosen with the well-known heuristic: $\rho = 0.5, c = 1e - 4$, and from the experiment we did, it turns out that it doesn't have significant effect to the performance of the algorithm that much. However, we will later see how the parameter μ affects the performance.

Furthermore, we also need to the parameter C which manages the balance between margin and empirical loss, which also significantly affect the SMO algorithm when update $\boldsymbol{\alpha}$.

Last but not least, the hyperparameter in the kernel functions, i.e. γ, d also need to be tuned. We will now see how the optimization worked and report the speed of convergence by comparing two optimization method and the two kernel functions.

For the hyperparameter γ, d in the two kernel functions, $\gamma = 0.00001, d = 2$ was used. These parameters are extremely important since these determine the space that our mapped data lives and hence also determine the space where the function for the hyperplane lives.
The value of the hyperparamter γ might look weird. This is because our flattened data $\mathbf{x}$ mostly consists of 0 and 256. That is, if we standardize the data to -1 to 1, then they γ near 1 would work as well. That is, notice that these values were chosen by inspecting its training and test accuracy that gives high accuracy. Also note that the optimization iteration converges regardless of these hyperparameters.

Hence, these values were empirically chosen by comparing the performance. However, note that we are not going to report their performances by varying them, since this is not our aim of this project.

(b) **Soft-Margin SVM with RBF and Polynomial Kernels**

We first implemented the two methods with $t = 100, \mu = 4$ and $C = 0.1$. Note that by setting $C = 0.1$, we are doing soft-maragin SVM. That is, by simply letting C to ∞, we can easily transform the problem to hard-margin SVM. Below figures shows the value of F^* in (43) at each iteration until convergence. It is important to note that f^* is only stored when the condition for clipping is satisfied (i.e. $((y_iE_i < -tol)\ and(\alpha[i] < C))\ or\ ((y_i * E_i > tol)\ and\ (alpha[i] > 0))$. This is because since we implemented the simplified SMO, which did not consider the choice of which α_i and α_j is chosen in order for optimizing the objective function (38).

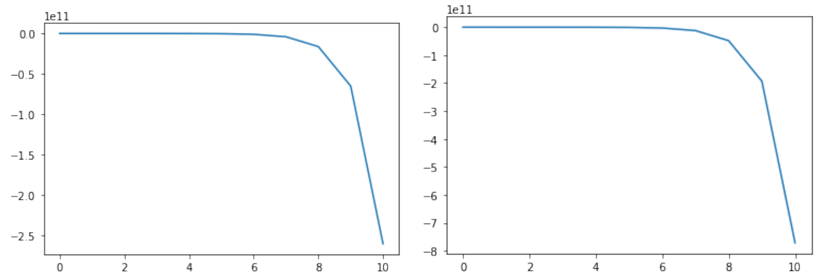

Figure 2: Plot of F^*, where F^* is a objective function (43). The x-axis is number of iteration of the barrier algorithm. The left figure is the one with the RBF kernel, and the rigt figure is the one with the polynomial kernel with $d = 1$.

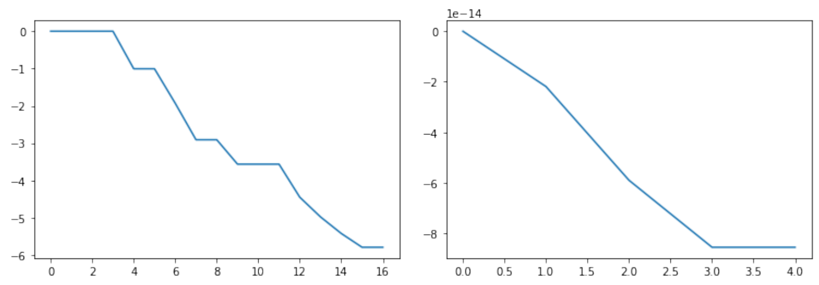

Figure 3: Plot of F^* at each iteration, where F^* is a objective function (38). The x-axis is number of iteration of the SMO algorithm. Note that f^* is only stored when the condition for clpping is satisfied (i.e. $((y_i E_i < -tol)\ and (\alpha[i] < C))\ or\ ((y_i * E_i > tol)\ and\ (alpha[i] > 0))$ The left figure is the one with the RBF kernel, and the right figure is the one with the polynomial kernel with $d = 1$.

From the figure 5 and 4 we see that the barrier method quadratically converged, and the simplified SMO quadratically converged. It is important to notice that, even though the global optimum is not guaranteed for the simplified SMO (due to the ignorance of heuristic to choose α_i, α_j), it seems to converge to the global optimum by referring the training and test accuracy near 100%.

We can further notice that the barrier method didn't affect by the variation of the kernel function, but the simplified SMO is affected by the variation of the kernel function.

From the experiment by varying the hyperparameters in the optimization methods, we could notice that only the parameter μ significantly affect the number of iterations that need to be converged until the stopping criteria. Varying the other parameters such as C in the SMO algorithm did not affect both in the convergence rate and the training/test accuracy. Below we vary the parameter μ.

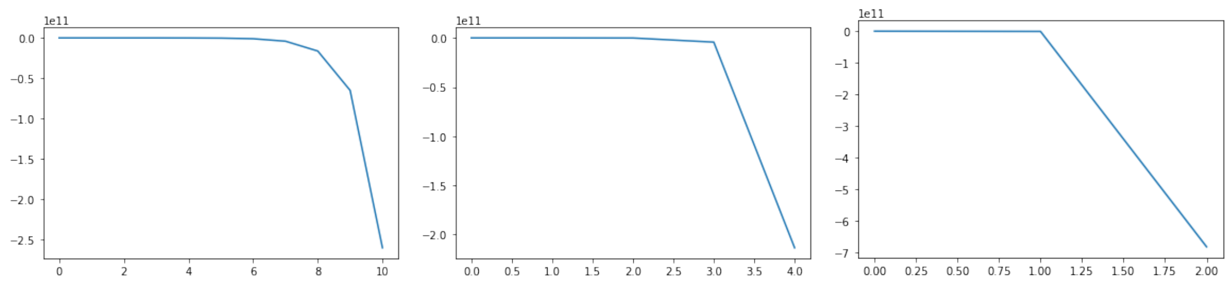

Figure 4: Plot of F^* at each iteration, where F^* is a objective function (38) with the following parameter: (a) $\mu = 4$, (b) $\mu = 100$, (c) $\mu = 1000$

From the figure 4 we can notice that the number of iterations required until the convergence is getting lower as μ increases. This is because the optimal α^* is achieved in the first iteration from Newton's method, and hence the algorithm is continued until $\mu \to \infty$ (i.e. $t \to \infty$). This might be because the problem is simple enough e.g. classifying 0 and 1.

Just for an aside, we can calculate the training and test accuracy, where training accuracy is achieved through the number of correctly classified data that we used for solving the corressponding optimization problem, and the test accuracy is achieved with the unseen data, to see if we actually found the well separating margin correctly.

Below is the result we get from the Jupyter notebook console. Please see the code that is attached in Moodle.

Barrier Method:

```
Training Accuracy of Binary SVM using Barrier method with RBF kernel = 1.0
Test Accuracy of Binary SVM using Barrier method with RBF kernel = 1.0

Training Accuracy of Binary SVM using Barrier method with Polynomial kernel = 1.0
Test Accuracy of Binary SVM using Barrier method with Polynomial kernel = 1.0
```

SMO Method:

```
Training Accuracy of Binary SVM using simplified SMO method with RBF kernel = 0.995
Test Accuracy of Binary SVM using simplfied SMO method with RBF kernel = 0.99

Training Accuracy of Binary SVM using simplified SMO method with Polynomial kernel = 1.0
Test Accuracy of Binary SVM using simplfied SMO method with Polynomial kernel = 1.0
```

(c) **Hard-Margin SVM with RBF Kernel**

We will now implement the hard-margin SVM with RBF kernel. From the above, we saw that there is no significance difference between choosing different kernels. Hence, it is reasonable to implement just RBF kernel for implementing the hard-margin SVM.

Hard-margin SVM can be implemented by letting $C \to \infty$, so that the constraint term $0 \le \boldsymbol{\alpha} \le C\mathbf{1}$ in the equation (31)is become $0 \le \boldsymbol{\alpha}$ which becomes the same optimization problem with the one with no loss function (e.g. hinge loss). Furthermore, it can be noticed that the effect of the term $\boldsymbol{C}$ in the equation (48) is gone, and also in simplified SMO algorithm.

In below, we provide two plots of F^* where F^* is a objective function (43) when we implement the hard-margin SVM by setting C = 10000000 with barrier and simplified SMO method. We can clearly see that there is no difference in rate of convergence, yet we can notice that there is a slight, but not significant, decrease in training and test accuracy in both methods.

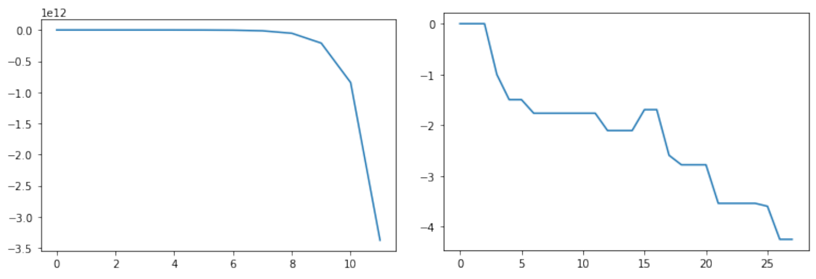

Figure 5: Plot of F^*, where F^* is a objective function (43) when we implement the hard-margin SVM by setting C = 10000000. The x-axis is number of iterations. Both of them uses RBF kernel. The left figure is the one implemented by Barrier method, and the rigt figure is the one implemented by SMO method.

Barrier Method:

```
Training Accuracy of Binary Hard Margin SVM using with RBF kernel = 0.994
Test Accuracy of Binary Hard Margin SVM using with RBF kernel = 0.99
```

SMO Method:

```
Training Accuracy of Binary Hard SVM using simplified SMO method with RBF kernel = 1.0
Test Accuracy of Binary Hard SVM using simplfied SMO method with RBF kernel = 0.98
```

6. **Multi-class soft-margin kernel support vector machine**

   Lastly, we can generalize binary kernel SVM to multi-class simply by building $\frac{n(n-1)}{2}$ binary kernel SVM (i.e. every possible pair of classes). Then, we vote the most chosen class. This method is called one vs one classification.

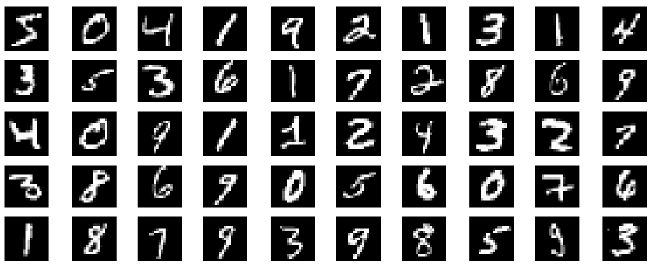

Figure 6: Digits 0 to 9 that we will classify with one vs one method. That is, we construct $\frac{10\times 9}{2}$ binary classifers and make them vote.

   From above, since we noticed that there was no big difference in accuracy between the two algorithms, we choose the simplified SMO method to implement the multi-class soft-margin kernel SVM. We built 45 binary polynomial kernel soft-margin SVM for each possible pair of classes and make them to vote. The hyperparameter is chosen as follows: $C = 0.1$ and $d = 2$.

   In the submitted Jupyter notebook, multiclass kernel svm with polynomial kernel with simplified SMO using one vs one is impleented, and has 0.922 accuracy. Note that there are a plenty of rooms for tuning the appropriate C and d for increasing the accuracy but we leave this for future studies. Please see the `mult_svm.ipynb` notebook in the submission.

References

[1] A. Ng. CS229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

[2] J. Nocedal and S. Wright. *Numerical optimization.* Springer Science &amp; Business Media, 2006.

[3] J. Shawe-Taylor and S. Sun. A review of optimization methodologies in support vector machines. *Neurocomputing*, 74(17):3609–3618, 2011.

[4] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[5] Y. Keshet. Multiclass classification. *sign (M (r, s)*, 50:1, 2014.