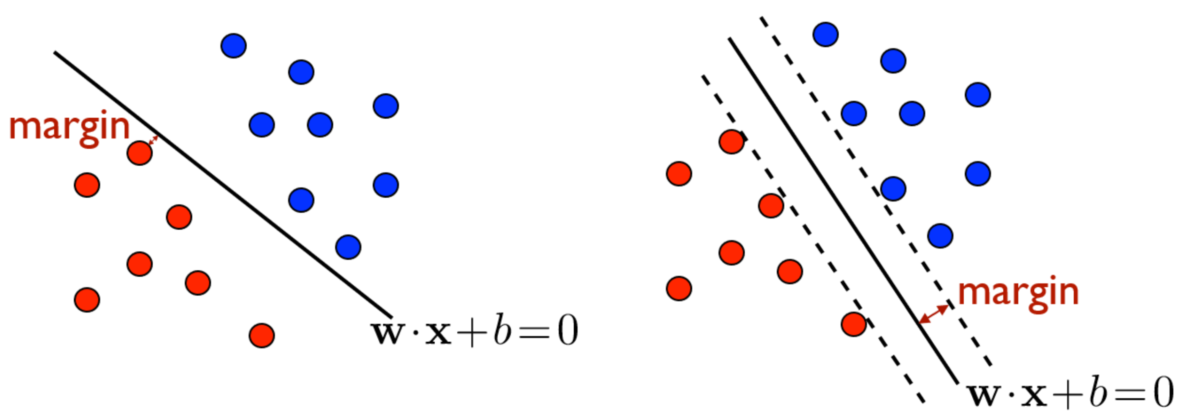
SVM Problem with SVO Solver

By Minghe Ren, Xinqiao Wei

EC500 (HPC) Project

1. SVM problem Introduction

In machine learning algorithm, SVM (support vector machine) is often used in supervised classification. The idea is, giving a data set, the algorithm trying to find the best hyperplane that separating the data. For example, in two dimension case, suppose you have a dataset which has two different labels. And the data points has two features (two dimension). The general idea can be represented as figure below, W·x + b = 0 represents the hyperplane.



In our project, we decided to write a two dimensional SVM classifier in C++. That is, writing our SVM solver code for two dimensional dataset to find the best w and b the separate data, which is the hyperplane.

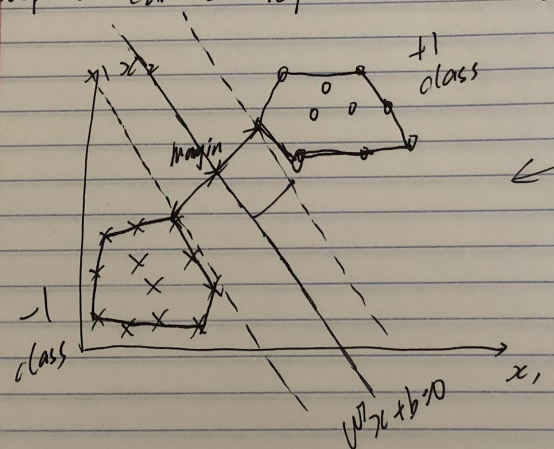
The idea can be shown as:

data feature: $x = \mathbb{R}^d$ In this case $d=2$ $x = (x_1, x_2)$
feature1 feature2

label: $y = \{-1, +1\}$

Data $= \{ (x_1, y_1), (x_2, y_2) \cdots\cdots (x_n, y_n) \}$
at least one sample from each class

hyperplane can be represented as $W^T x + b = 0$



For Data is linearly separable case

Distance of x to hyperplane (W^T, b) can be represented:

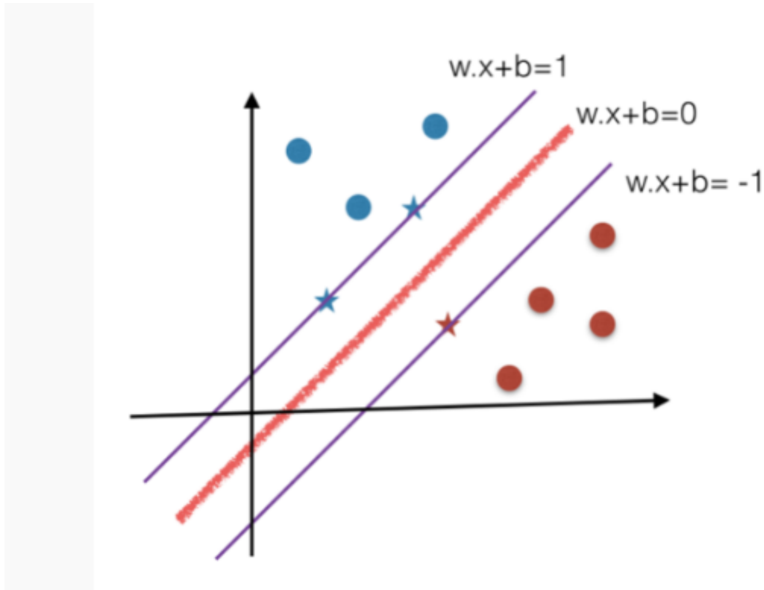
Distance $= \dfrac{|W^T x + b|}{\|W\|}$, Margin $= \min(\text{Distance}) = \min \dfrac{|W^T x + b|}{\|W\|}$

Our goal: Find the best hyperplane which has the largest margin

$$(w, b) = \underset{w,b}{\operatorname{argmax}} \left[\min_{1 \le j \le n} \left(\frac{|W^T x_j + b|}{\|W\|} \right) \right]$$

The hyperplane is the classifier. Above it, it’s class +1. Below it, it’s class -1.

If we let the min|W·x + b | = 1 , we can represent the margin as:



By introduce Laplacian Multipliers and KKT condition, we actually are solving this dual optimization problem:

- **Lagrangian**: for all $\mathbf{w}, b, \alpha_i \geq 0,$

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1].$$

- **KKT conditions**:

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \iff \boxed{\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i.}$$

$$\nabla_b L = -\sum_{i=1}^{m} \alpha_i y_i = 0 \iff \boxed{\sum_{i=1}^{m} \alpha_i y_i = 0.}$$

$$\boxed{\forall i \in [1, m], \ \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0.}$$

Substitude expression for w back into Lagrangian

We get $$\mathcal{L} = \frac{1}{2}\Big\|\sum_{j=1}^{n} \alpha_j y_j x_j\Big\|^2 + \sum_{j=1}^{n} \alpha_j \Big(1 - y_j\big(x_j^T\sum_{k=1}^{n} \alpha_k y_k x_k + b\big)\Big)$$

$$= \frac{1}{2}\Big\|\sum_{j=1}^{n} \alpha_j y_j x_j\Big\|^2 + \sum_{j=1}^{n} \alpha_j - b\sum_{j=1}^{n} \alpha_j y_j$$

$$- \underbrace{\sum_{j=1}^{n} \alpha_j y_j \sum_{k=1}^{n} \alpha_k y_k (x_j^T x_k)}$$

$$- \underbrace{\sum_{j=1}^{n}\sum_{k=1}^{n} \alpha_j \alpha_k y_j y_k (x_j^T x_k)}_{\|w\|^2 = w^T w}$$

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \sum_{j=1}^{n}\alpha_j - \underbrace{\cancel{b\sum_{j=1}^{n}\alpha_j y_j}}_{=0} - \|w\|^2$$

$$= \Big(\sum_{j=1}^{n}\alpha_j\Big) - \frac{1}{2}\|w\|^2$$

$$= \sum_{j=1}^{n}\alpha_j - \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \alpha_j\alpha_k y_j y_k (x_j^T x_k)$$

Dual optimation problem

$\max_{\alpha \geq 0} f(\alpha)$, $\sum \alpha_j y_j = 0$ | $\alpha^{svm} = \arg\max_{\alpha} \Big(\sum_{j=1}^{n}\alpha_j - \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j\alpha_k y_j y_k (x_j^T x_k)\Big)$

$\forall j \;\; \alpha_j \geq 0$, $\sum_{j=1}^{n}\alpha_j y_j \neq 0$

$$\alpha^{svm} = \underset{\alpha \geq 0}{argmax} \Big[\underbrace{\sum_{j=1}^{n} \alpha_j}_{\sum_{j=1}^{n} \alpha_j y_j = 0} - \frac{1}{2} \underbrace{\sum_{j=1}^{n}\sum_{k=1}^{n} \alpha_j \alpha_k y_j y_k (x_j^T x_k)}_{= \| \sum_{j=1}^{n} \alpha_j y_j x_j \|^2}\Big]$$

May not be unique.

This Dual Optimazation problem is solved via an iterative numerical method, eg - Quadratic Program (QP)
- Sequential Minimal Optimization (SMO)

Complementary Conditions

$$\forall j \quad \alpha_j^{svm}\big(1 - y_j((w^{svm})^T x_j + b^{svm})\big) = 0$$

Support Vector (SV)

A feature x_j for which $\alpha_j^{svm} > 0$ is called a support vector

$\Rightarrow$ if x_j is a SV then

$$1 - y_j((w^{svm})^T x_j + b^{svm}) = 0$$

$$\Rightarrow (w^{svm})^T x_j + b^{svm} = y_j$$

$\Rightarrow$ all SV's are closest to the svm hyperplane.

Our problem becomes: a dual optimization problem with boundary condition:

$$\alpha^{SVM} = argmin[\sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{j=1}^{n}\sum_{k=1}^{n} \alpha_j \alpha_k y_j y_k (x_j^T x_k)]$$

The condition is $\sum_{j=1}^{n} \alpha_j y_j = 0$ and $0 \leq \alpha_j \leq C$ (C is hyper parameter). This can be solved by Sequential minimal Optimization.

2. **SMO**

SMO breaks this problem into a series of smallest possible sub-problems, which are then solved analytically. Because of the linear equality constraint involving the Lagrange multipliers αj the smallest possible problem involves two such multipliers. Then, for any two multipliers α1 and α2, the constraints are reduced to:

$$0 \leq \alpha_1, \alpha_2 \leq C$$

$$y_1\alpha_1 + y_2\alpha_2 = k$$

This reduced problem can be solved analytically: one needs to find a minimum of a one-dimensional quadratic function. K is the negative of the sum over the rest of terms in the equality constraint, which is fixed in each iteration.

The linear classifier f(x) = wTx + b can be express as:

$$f(x) = \sum_{i=1}^{m} \alpha_i y^i < x^i, x > +b$$

The KKT condition for this problem:

$$\alpha_i = 0 \rightarrow y^i(w^T x_i + b) \geq 1$$

$$\alpha_i = C \rightarrow y^i(w^T x_i + b) \leq 1$$

$$0 < \alpha_i < C \rightarrow y^i(w^T x_i + b) = 1$$

Next step,optimizing αi and αj. First, find the bounds L and H such that L ≤ αj≤ H for the constraint 0≤αj ≤C:

$$\text{If } y_i \neq y_j,\ L = max(0, \alpha_j - \alpha_i),\ H = min(C, C + \alpha_j - \alpha_i)$$

$$\text{If } y_i = y_j,\ L = max(0, \alpha_i + \alpha_j - C),\ H = min(C, \alpha_i + \alpha_j)$$

The optimal αj is given by:

$$\alpha_j := \alpha_j - \frac{y_i(E_i - E_j)}{\eta}$$

where Ek = f(xk) − Yk . It is the error between the SVM output on the kth example and the true label Yk. And η=2<xi, xj > − <xi, xi > − <xj, xj >

If αj ends up lying outside the bounds L and H, we modify αj:

$$\alpha_j := \begin{cases} H & \text{if } \alpha_j > H \\ \alpha_j & \text{if } L \leq \alpha_j \leq H \\ L & \text{if } \alpha_j < L \end{cases}$$

Then we solve for αi:

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

Select the threshold b satisfying KKT for xi and xj. If αi is not at the bounds then b1 is valid:

$$b_2 = b - E_i - y_i(a_i - a_i^{old}) < x_i, x_i > -y_i(\alpha_j - \alpha_j^{old}) < x_i, x_j >$$

Similarly, if 0 < αj < C, b2 is valid:

$$b_2 = b - E_j - y_i(a_i - a_i^{old}) < x_i, x_j > - y_j(\alpha_j - \alpha_j^{old}) < x_j, x_j >$$

If both b1 and b2 are both valid, the value would be the same. If both αs are at the bounds then the b := (b1 + b2)/2 would meet the KKT condition.

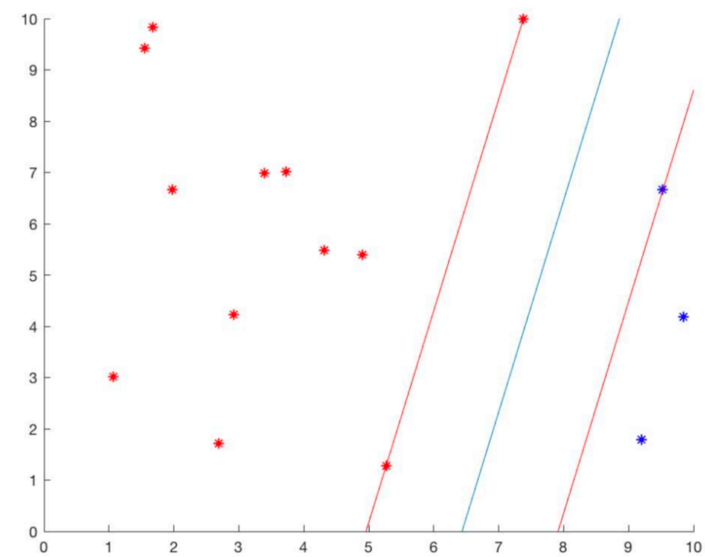
3. Code

Here is our pseudo-code:

Initial: $\alpha_i = 0, \forall i,\ b = 0,\ passes = 0$
while $passes < max\_passes$ **do**
    $num\_changed\_alphas = 0;$
    for $i = 1, ..., m$ **do**
        Calculate $E_i = f(x^i) - y^i$
        if $((y^i E_i < -tol \&\& a_i < C) \| (y^i E_i > tol \&\& a_i > 0))$ **then**
            Select $j \neq i$ randomly.
            Calculate $E_j = f(x^i) - y^j$
            Save old α's: $\alpha_i^{old} = \alpha_i,\ \alpha_j^{old} = \alpha_j$
            Compute L and H.
            if $(L == H)$ **then**
                continue to next i.
            Compute η.
            if $(\eta \geq 0)$ **then**
                continue to next i.
            Compute and clip new value for α_j
            if $(|\alpha_j - \alpha_j^{old}| < 10^{-5})$ **then**
                continue to next i.
            Determine value for α_i
            Compute b_1 and b_2
            Compute b
            $num\_changed\_alphas := num\_changed\_alphas + 1$
        end if
    end for
    if $(num\_changed\_alphas == 0)$ **then**
        $passes := passes + 1$
    else
        $passes := 0$
end while

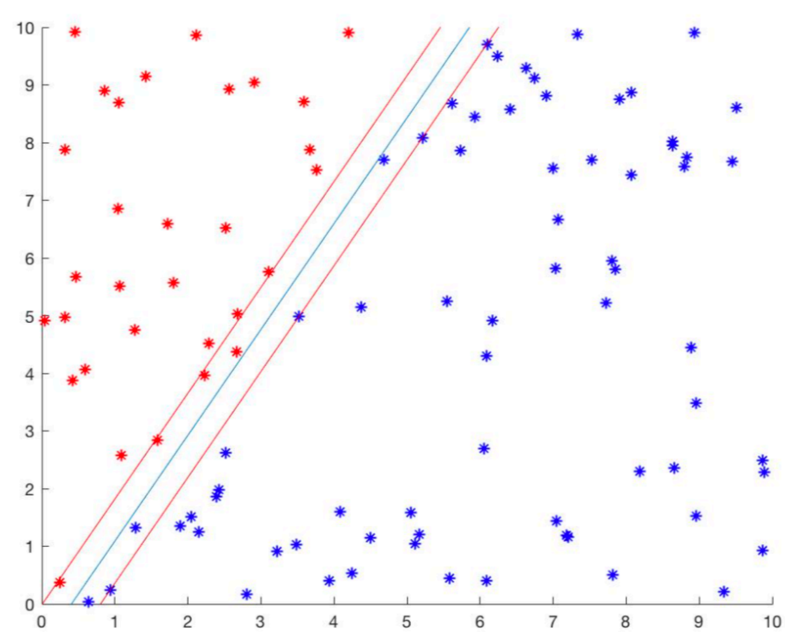
4. Result

For 15 points dataset:



Green line is the hyperplane and the red lines represents the margin. The hyperplane we calculate perfectly separates the data.

For 100 data points:



The green hyperplane also perfectly solve the problem and with some data points living the boundary.

The output of our code shown below, where most of the Laplacian multipliers are zero.

```
The calculated alphas are:
0  0  0  0  0  0  0  0  0  1.6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1.6  0  0.00401323  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  1.14711  0  0  0  0  0  1.6  0  0  1.6  0  1.6  0  0  0  0  0  0  0  0  0  1.6  1.6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0.0158122  0  0  0.433067  0  0  0  0  0
The calculated W is:
2.51426 -1.37197

The calculated b is:
-1.02552

The calculated values of support vector are:
-0.923299 0.374325 1.00948 1 0.520782 -0.849337 -0.325389 -0.949446 0.184743 1.00134 1 [Finished in 1.0s]
```

For 1000 data points, our code can't converge due to some unknown reason.

5. Future work

a. Test on a much bigger dataset like 100,000 points and check the performance.
b. Different algorithms to achieve SVM like Quadratic Programming.
c. Test on non-linear separable dataset and non-separable dataset.
d. Try to parallelize our code

6. Reference:

http://cs229.stanford.edu/materials/smo.pdf

Our Github link:
https://github.com/rmhsawyer/EC500-HPC-Final-Project