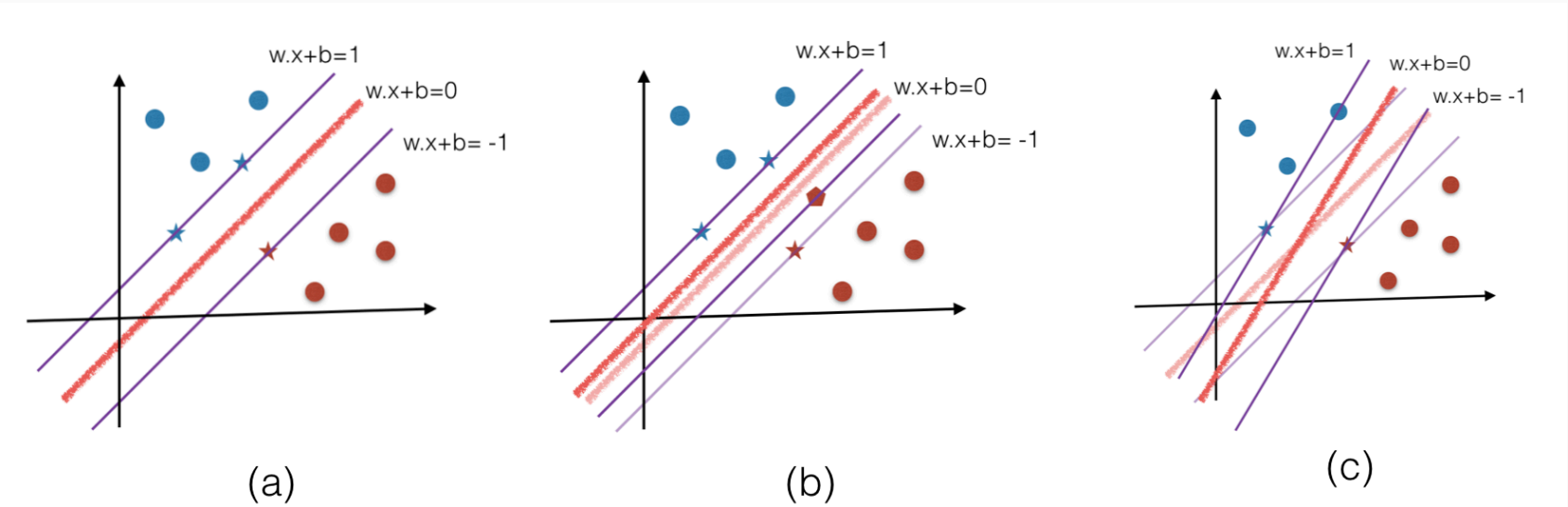
Support Vector Machine

Machine Learning Algorithm in C++

Xinqiao Wei, Minhe Ren

Basic SVM

An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The best hyperplane for an SVM means the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points.



Binary Classification

- Label space: $Y = \{-1, +1\}$
- Feature space: $X \in \mathbb{R}^2$
- Training Set: $D = (x_1, y_1), ..., (x_n, y_n)$
- Family of classifiers: $H = \{h(x) = sign(w^T x + b), w \in \mathbb{R}^2, b \in \mathbb{R}\}$

Equation for hyperplane in $\mathbb{R}^2$ is: $w^T + b = 0$. We need to find the optimum hyperplane, which is the maximum margin separating hyperplane.
Support vectors are the vectors on the boundary:

$$y_j(w^T x_j + b) = 1$$

Dual optimization problem

α is Lagrange multipliers.

$$w^{SVM} = \sum_{j=1}^{n} \alpha_j^{SVM} y_j x_j$$

$$b^{SVM} = y_i - (w_{SVM})^T x_j$$

for any $\alpha_j^{SVM} > 0$ and there exist at least one such $\alpha_j^{SVM} > 0$.

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

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.

Sequential minimal optimization

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 α_j the smallest possible problem involves two such multipliers. Then, for any two multipliers α_1 and α_2,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.

Sequential minimal optimization Algorithm

Here are the algorithm:

The algorithm proceeds as follows:

1. Find a Lagrange multiplier α_1 that violates the Karush–Kuhn–Tucker (KKT) conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair (α_1, α_2).
3. Repeat steps 1 and 2 until convergence.

When all the Lagrange multipliers satisfy the KKT conditions (within a user-defined tolerance), the problem has been solved. Although this algorithm is guaranteed to converge, heuristics are used to choose the pair of multipliers so as to accelerate the rate of convergence. This is critical for large data sets since there are $n(n-1)/2$ possible choices for α_i and α_j.

Sequential minimal optimization Algorithm Implementation

The linear classifier $f(x) = w^T x + b$ can be express as:

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

The KKT condition for this problem:

$$\alpha_i = 0 \rightarrow y^i(w^T x_i + b) \geq 1 \tag{2}$$

$$\alpha_i = C \rightarrow y^i(w^T x_i + b) \leq 1 \tag{3}$$

$$0 < \alpha_i < C \rightarrow y^i(w^T x_i + b) = 1 \tag{4}$$

Sequential minimal optimization Algorithm Implementation

Optimizing α_i and α_j

First, find the bounds L and H such that $L \leq \alpha_j \leq H$ for the constrain $0 \leq \alpha_j \leq C$:

$$\text{If } y_i \neq y_j,\ L = max(0, \alpha_j - \alpha_i),\ H = min(C, C + \alpha_j - \alpha_i) \tag{5}$$

$$\text{If } y_i = y_j,\ L = max(0, \alpha_i + \alpha_j - C),\ H = min(C, \alpha_i + \alpha_j) \tag{6}$$

The optimal α_j is given by:

$$\alpha_j := \alpha_j - \frac{y_i(E_i - E_j)}{\eta} \tag{7}$$

where $E_k = f(x_k) - y_k$. It is the error between the SVM output on the kth example and the true label y_k.

And $\eta = 2 < x_i, x_j > - < x_i, x_i > - < x_j, x_j >$

Sequential minimal optimization Algorithm Implementation

Optimizing α_i and α_j

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

$$\alpha_j := \begin{cases} H & \text{if } \alpha_j > H \\ \alpha_j & \text{if } L \leq \alpha_j \leq H \\ L & \text{if } \alpha_j < L \end{cases} \tag{8}$$

Then we solve for α_i:

$$\alpha_i := \alpha_i + y_i y_j (\alpha_j^{old} - \alpha_j) \tag{9}$$

Sequential minimal optimization Algorithm Implementation

Optimizing α_i and α_j

Select the threshold b satisfying KKT for x_i and x_j. If α_i is not at the bounds then b_1 is valid:

$$b_2 = b - E_i - y_i(a_i - a_i^{old}) < x_i, x_i > -y_i(\alpha_j - \alpha_j^{old}) < x_i, x_j > \quad (10)$$

Similarly, if $0 < \alpha_j < C$, b_2 is valid:

$$b_2 = b - E_j - y_i(a_i - a_i^{old}) < x_i, x_j > -y_j(\alpha_j - \alpha_j^{old}) < x_j, x_j > \quad (11)$$

If both b_1 and b_2 are both valid, the value would be the same. If both αs are at the bounds then the $b := (b_1 + b_2)/2$ would meet the KKT condition.

Sequential minimal optimization Algorithm Implementation

Pseudo Code

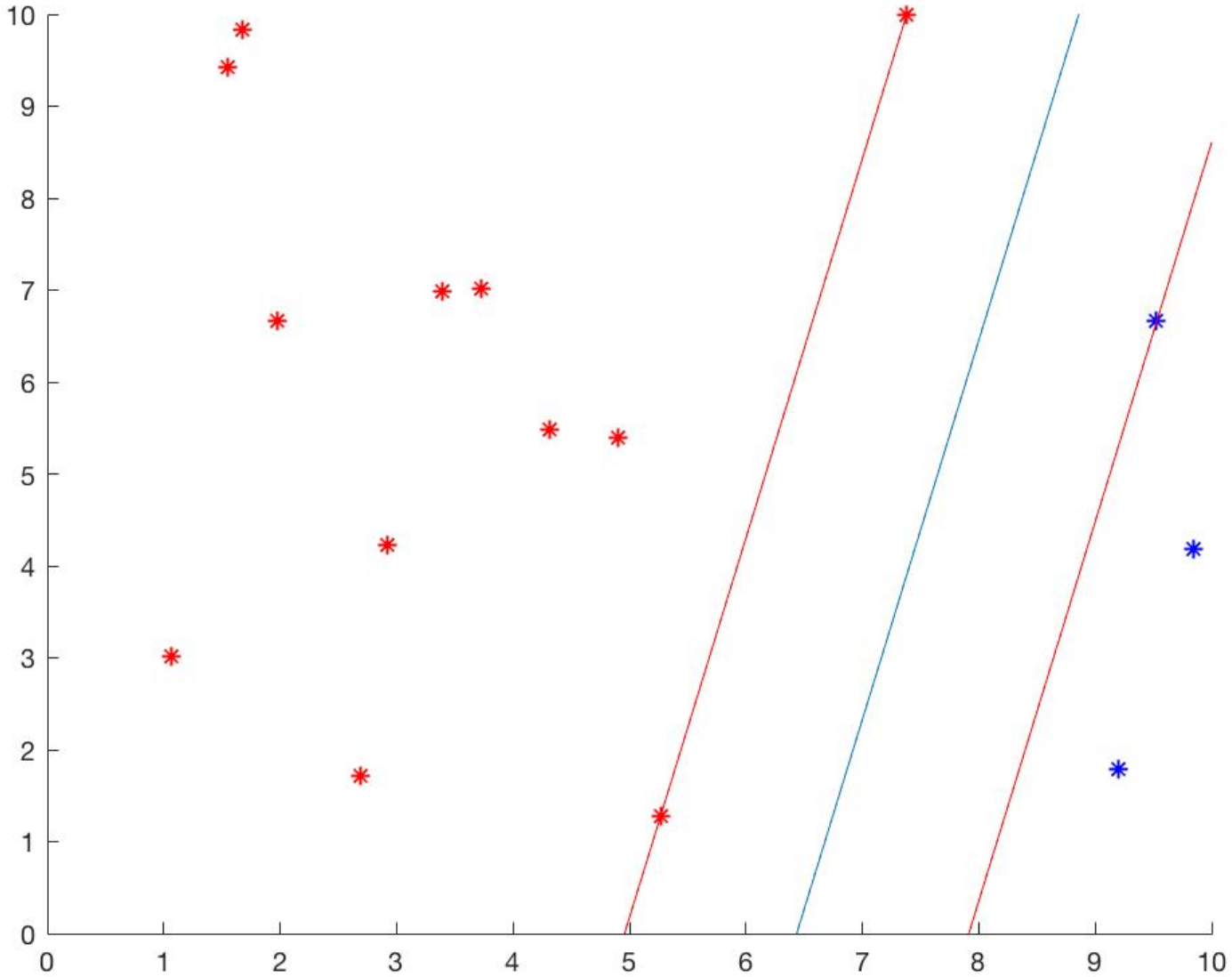
Initial: $\alpha_i = 0, \forall i,\ b = 0,\ passes = 0$
while $passes < max\_passes$ **do**
    $num\_changed\_alphas = 0;$
    for $i = 1, ..., m$ **do**
        Calculate $E_i = f(x^i) - y^i$
        if $((y^i E_i < -tol \&\& a_i < C) \| (y^i E_i > tol \&\& a_i > 0))$ **then**
            Select $j \neq i$ 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.

Sequential minimal optimization Algorithm Implementation

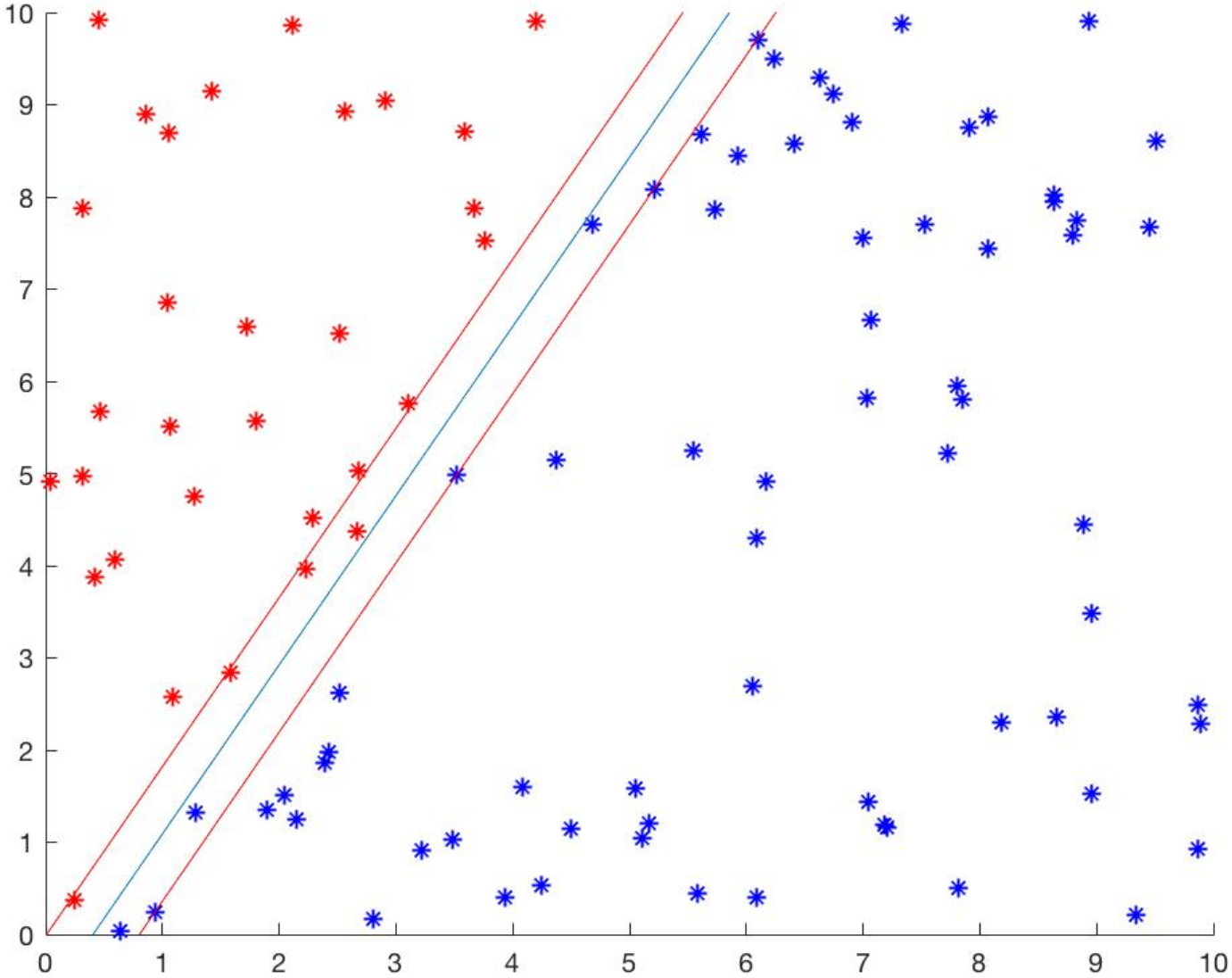
Pseudo Code

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

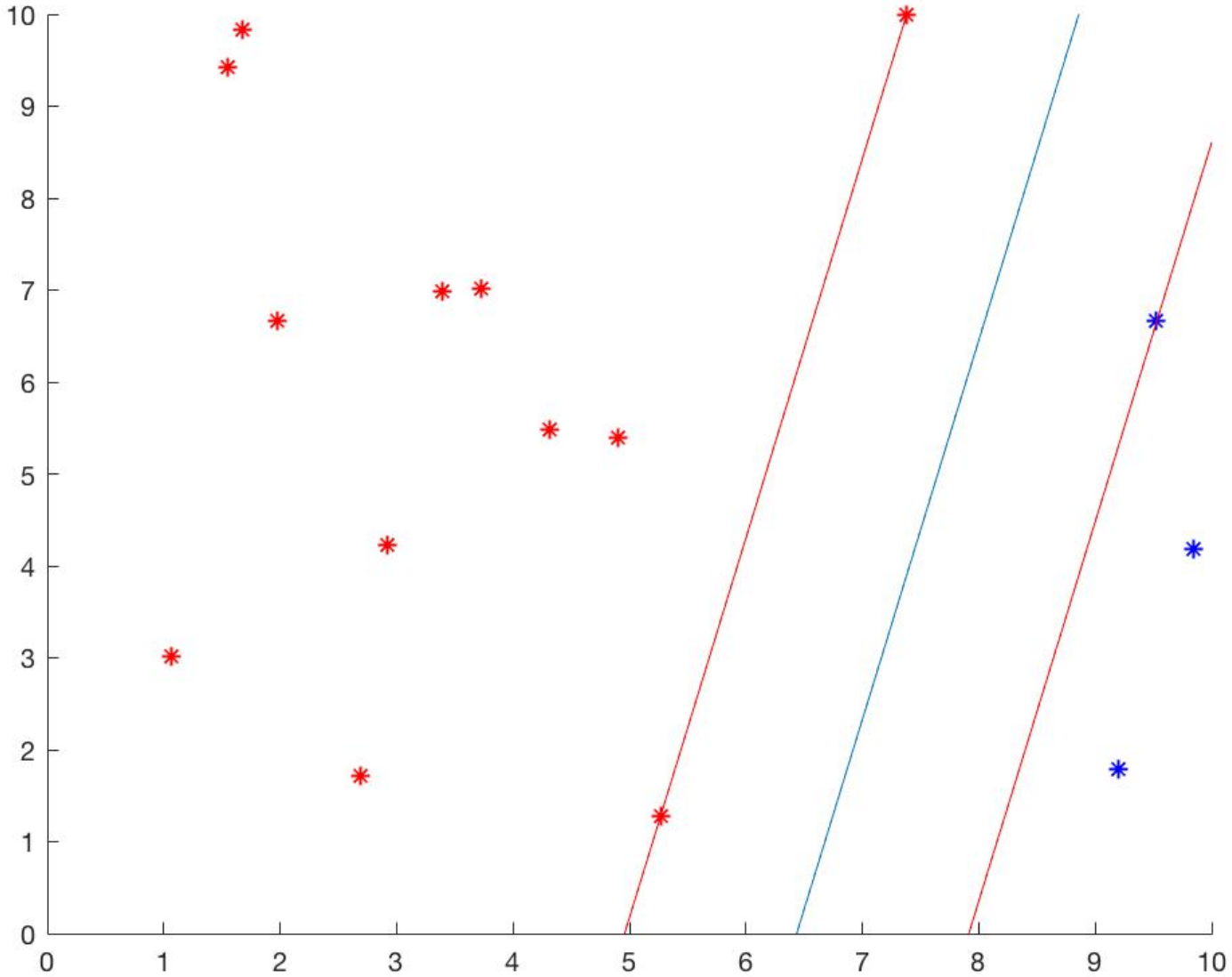
Dataset and Result



Dataset and Result



Dataset and Result



Dataset and Result

```
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  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  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
0.0158122  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]
```

Parallel Implement

- After we successfully solving the SVM problem using SMO algorithm, we also try to parallel this problem.
- We only use small data to test our code and the data points is linear separable, so the runtime is short. The difference between parallel program and sequence program is only about 0.3 second.
- Each slave CPUs calculates the local variables α_i and α_j based on the two variable select algorithms of SMO.
- In the main CPUs check whether α lies in the boundary and update the α

#pragma omp parallel for, mpi_scatter/gather

Future work

- Test on a much bigger dataset like 100,000 points and check the performance
- Different algorithms to achieve SVM like Quadratic Programming.
- Test on non-linear separable dataset and non-separable dataset.

Thanks!