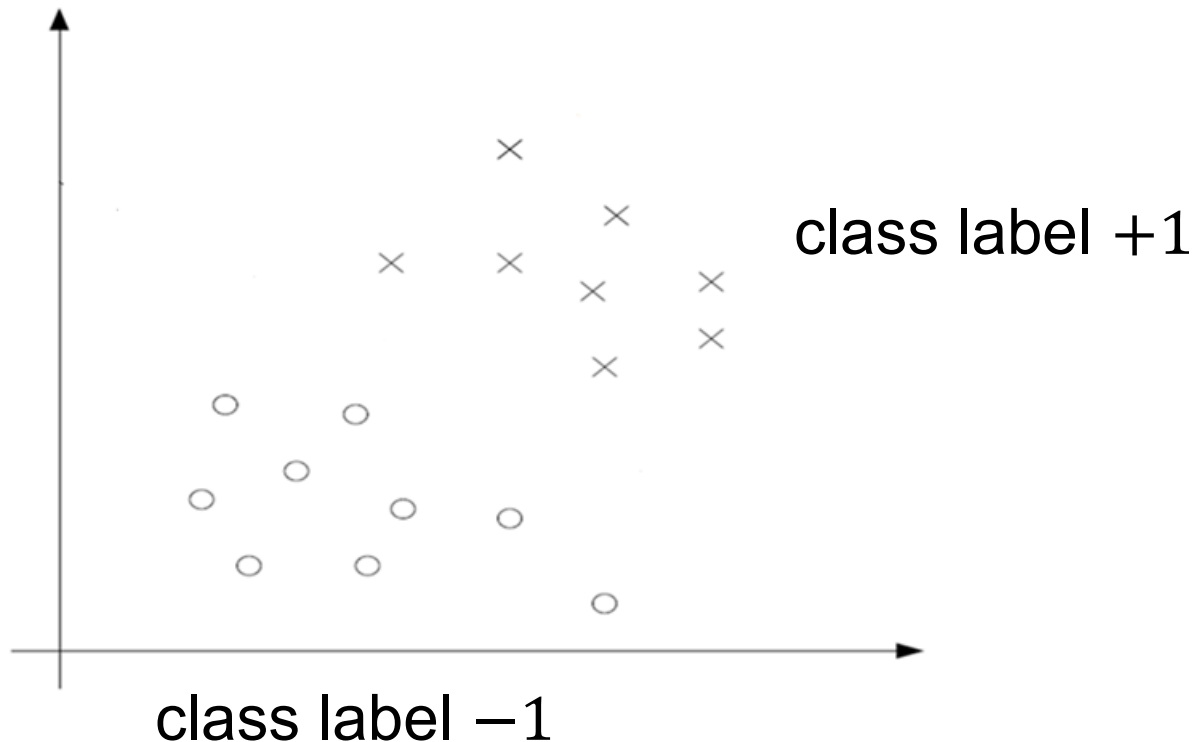


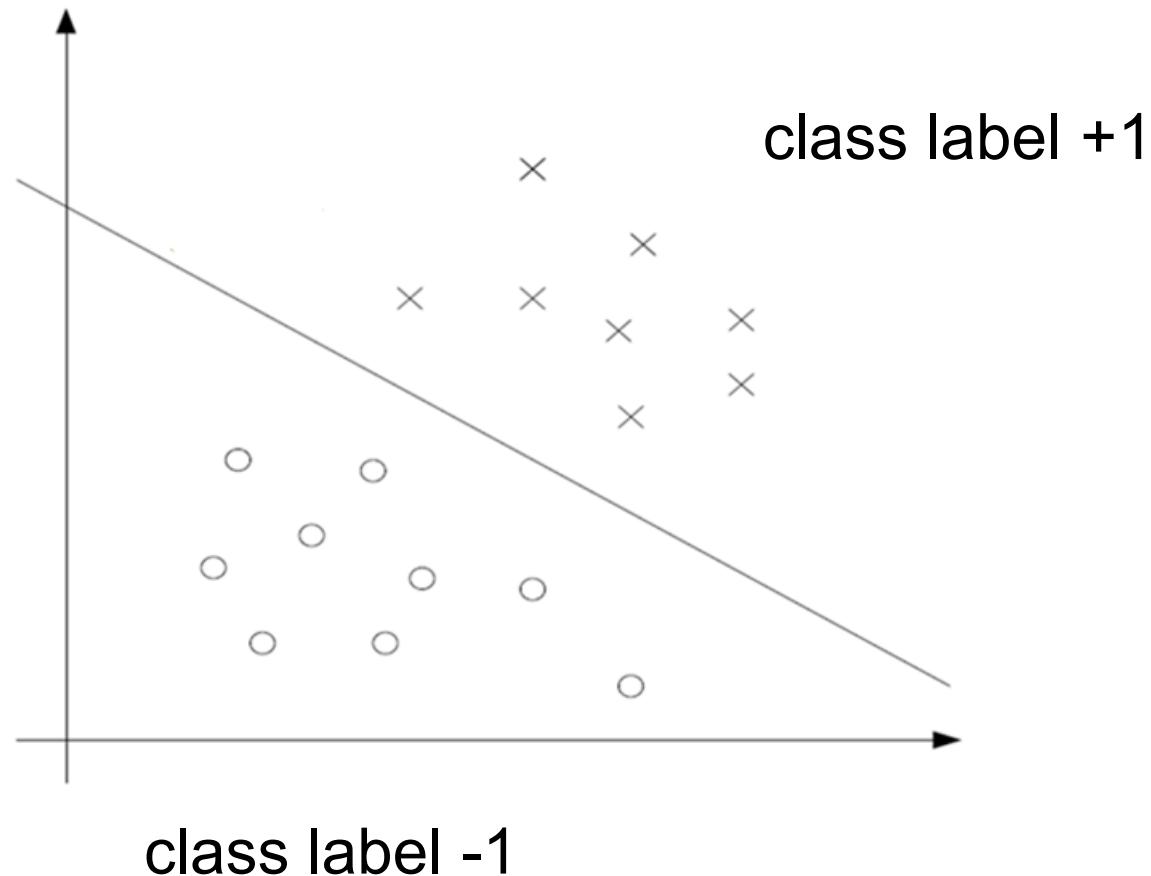
5. Support Vector Machines

5.1 Motivation

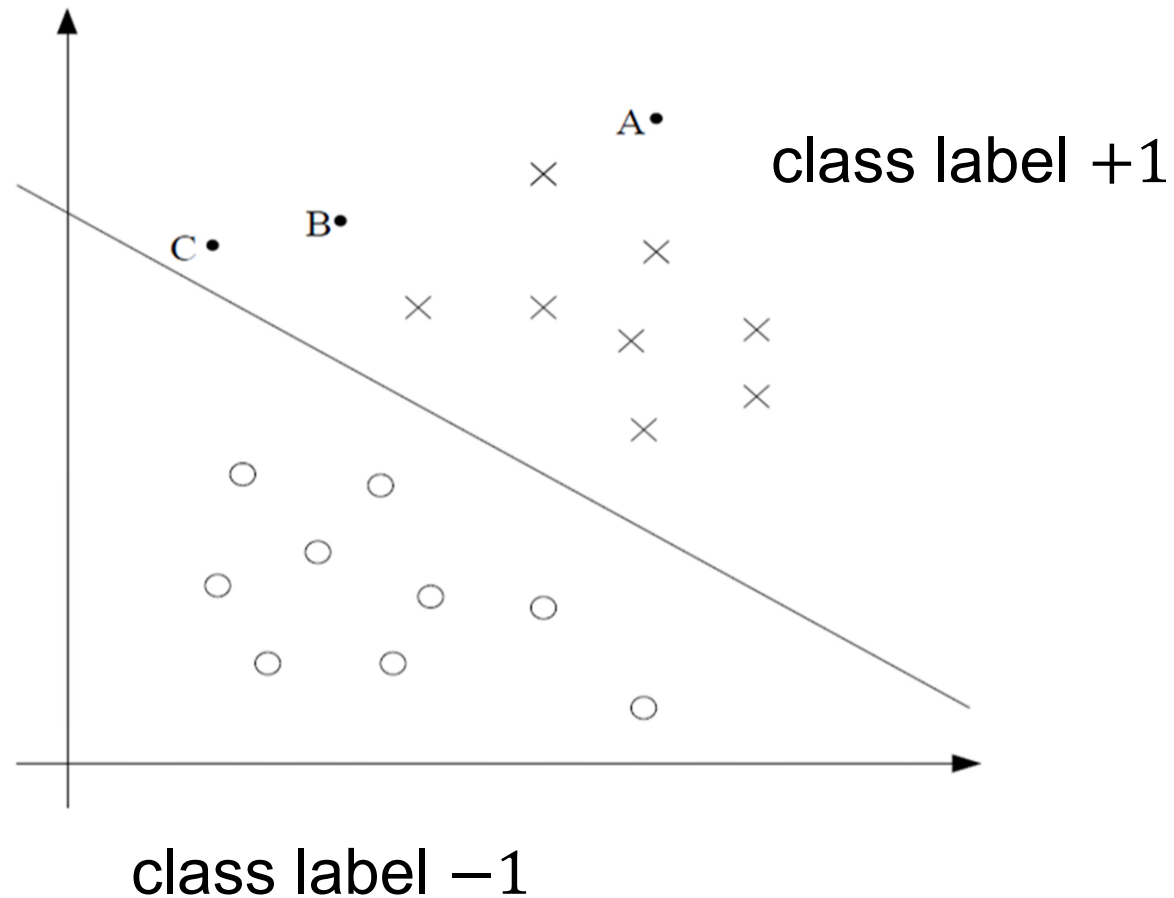
Consider N training samples from two classes, where class label d takes value of $+1$ or -1 , respectively



The objective of linear classification is to **find a decision surface in the form of hyperplane** to separate the data in the two classes as shown below.



Assuming that we have three test data points A, B, C. We now predict the class labels for the three data points.

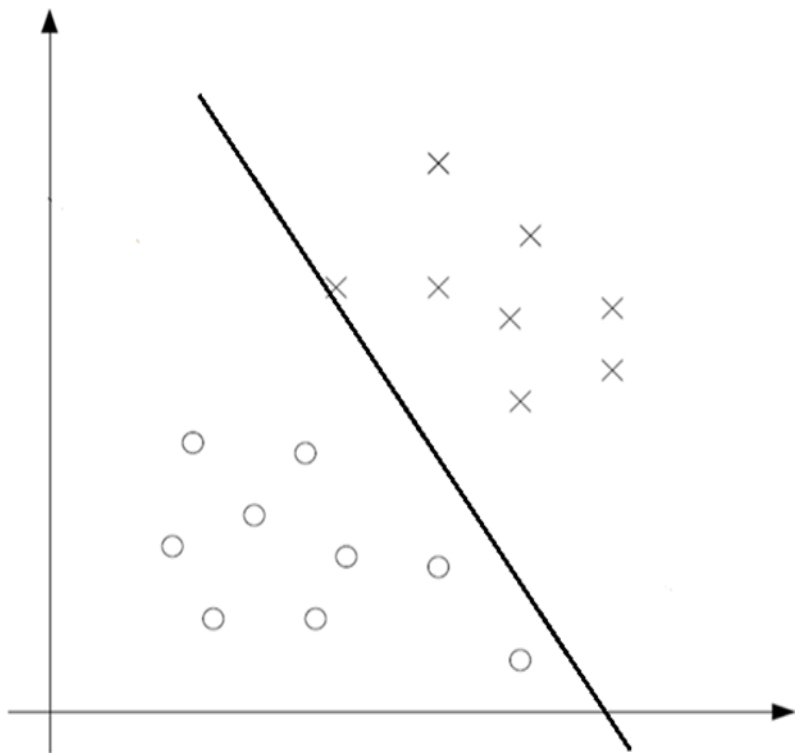


It is observed that

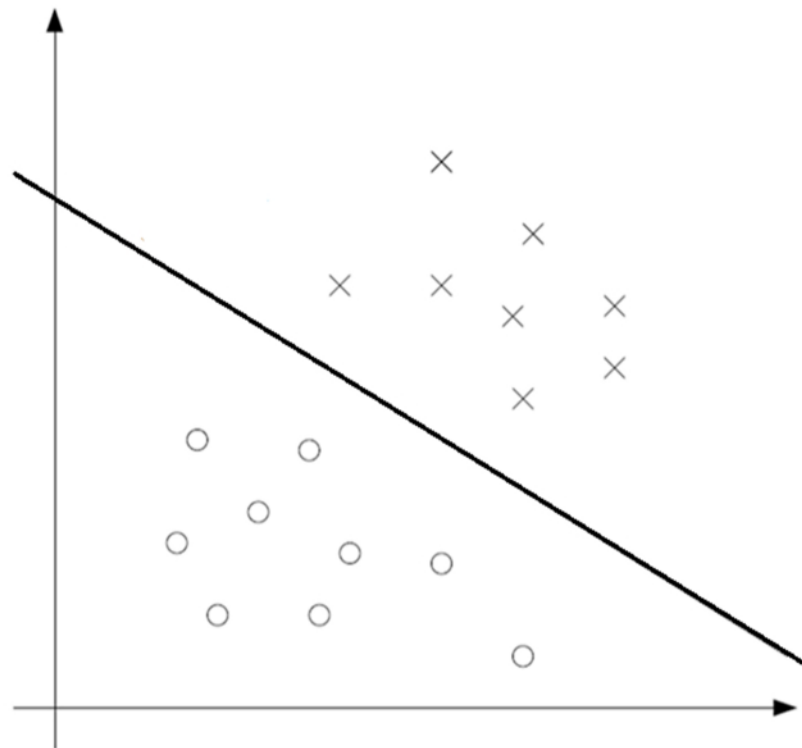
- (1) Point A is far from the separating hyperplane. We are quite confident that its label is $+1$.
- (2) Point C is very close to the hyperplane. While we would predict $+1$, it seems likely that just a small change to the separating plane could easily cause our prediction to be -1 .

It would be nice if we manage to find a separating plane that allows us to make predictions

- (1) correctly and
- (2) confidently (meaning far away from the separating hyperplane).



(a)



(b)

Hyperplane in (b) is preferred because the samples are farther from separating hyperplane in (b) than in (a).

5.2 Problem formulation

Mathematically, the equation of the hyperplane is given as follows:

$$\mathbf{w}^T \mathbf{x} + b = 0$$

Where \mathbf{x} is the feature vector, \mathbf{w} is the adjustable weight vector, and b is the bias.

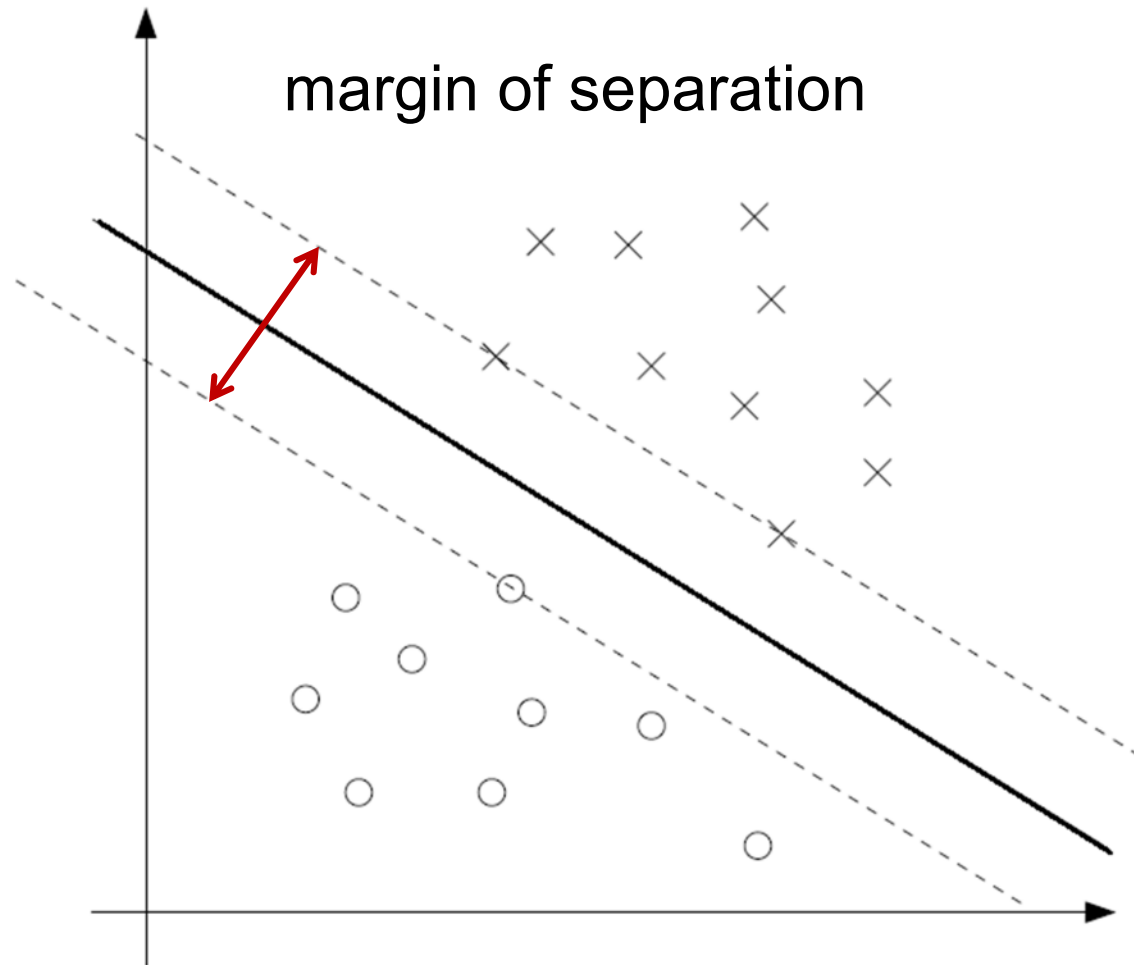
(a) For samples with class label $d = +1$

$$\mathbf{w}^T \mathbf{x} + b > 0$$

(b) For samples with class label $d = -1$

$$\mathbf{w}^T \mathbf{x} + b < 0$$

The separation between the hyperplane and the closest data point is called the **margin of separation**.

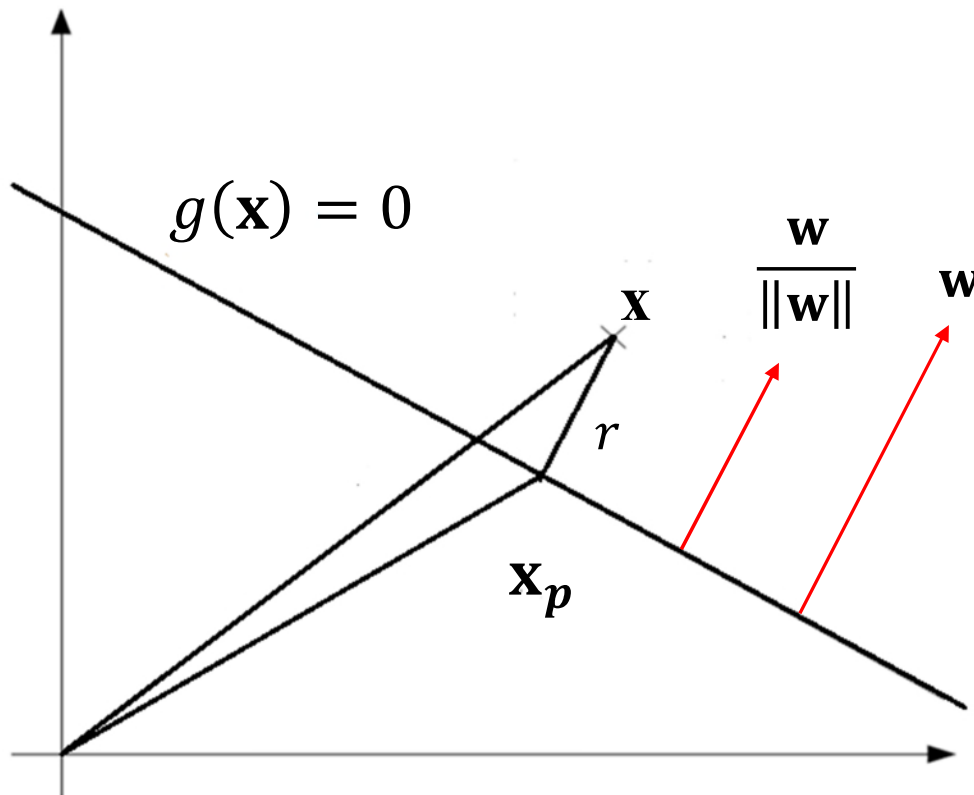


The goal of a **support vector machine** (SVM) is to find the particular hyperplane for which **the margin of separation is maximized**.

Under this condition, the decision surface is referred to as the **optimal hyperplane**.

Assume \mathbf{x}_p is the normal projection of \mathbf{x} onto the hyperplane, r is the distance from \mathbf{x} to the hyperplane, then:

$$\mathbf{x} = \mathbf{x}_p + (\mathbf{x} - \mathbf{x}_p) = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



$$\begin{aligned}
g(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b \\
&= \mathbf{w}^T \left(\mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b \\
&= \mathbf{w}^T \mathbf{x}_p + b + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \\
&= g(\mathbf{x}_p) + r \|\mathbf{w}\|
\end{aligned}$$

Since \mathbf{x}_p is on the separating hyperplane, we have:

$$g(\mathbf{x}_p) = 0$$

Then we have:

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

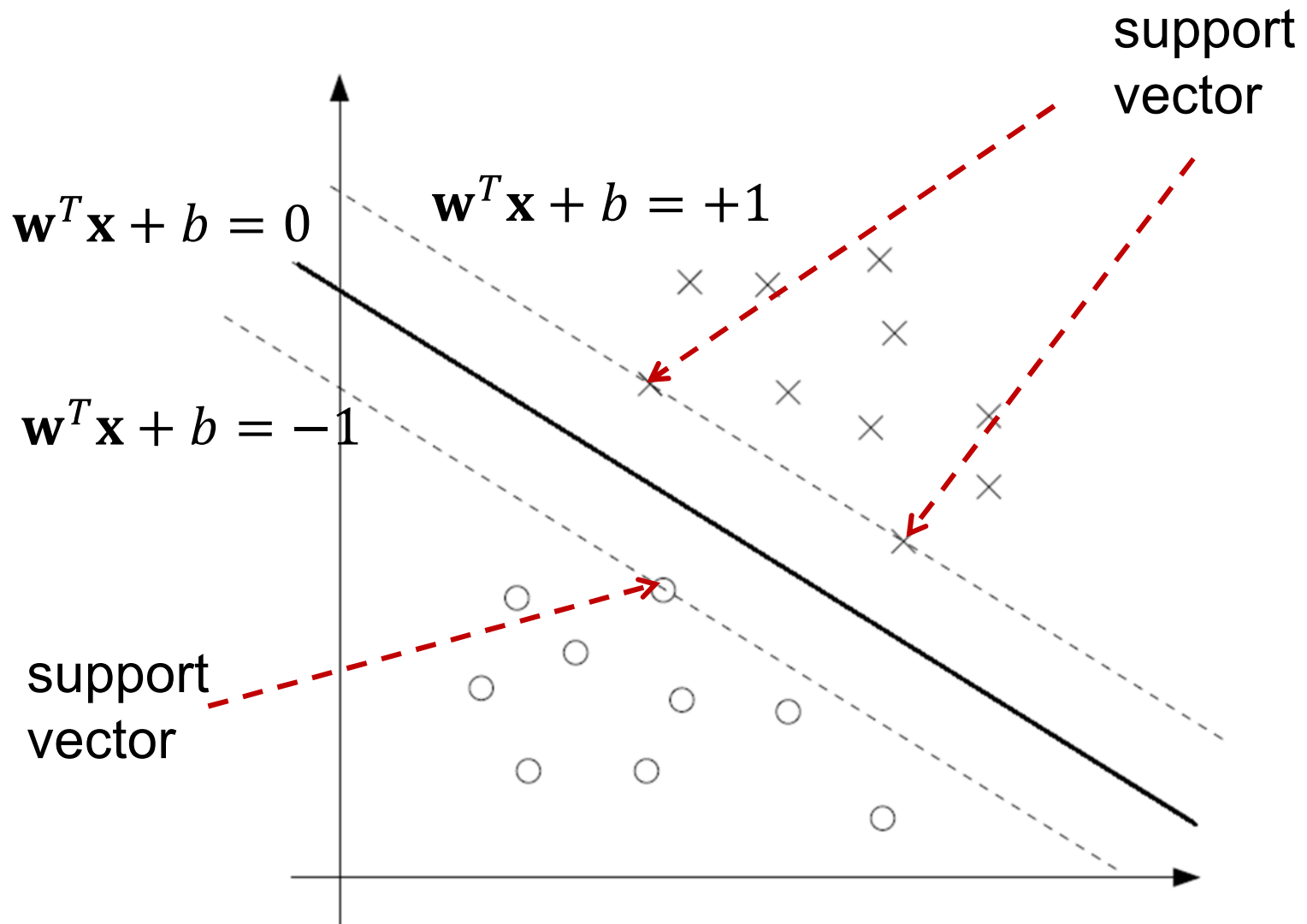
If the data are linearly separable, we can always scale \mathbf{w} and b so that

$$\mathbf{w}^T \mathbf{x} + b \geq 1 \quad \text{for } d = +1$$

$$\mathbf{w}^T \mathbf{x} + b \leq -1 \quad \text{for } d = -1$$

The particular data points for which the above is satisfied with equality sign are called **support vectors**.

The support vectors play a prominent role in the operation of this class of learning machines. In conceptual terms, the support vectors are those data points that **lie closest to the decision surface** and therefore **the most difficult to classify**.



The algebraic distance from support vector $\mathbf{x}^{(s)}$ to the optimal hyperplane is:

$$r = \frac{g(\mathbf{x}^{(s)})}{\|\mathbf{w}\|} = \begin{cases} \frac{1}{\|\mathbf{w}\|} & \text{for } d^{(s)} = +1 \\ \frac{-1}{\|\mathbf{w}\|} & \text{for } d^{(s)} = -1 \end{cases}$$

Where the plus sign indicates that $\mathbf{x}^{(s)}$ lies on the positive side of the optimal hyperplane, and the minus sign indicates that $\mathbf{x}^{(s)}$ lies on the negative side of the optimal hyperplane

The margin of separation between the two classes is given by:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Maximizing the margin of separation ρ is equivalent to minimizing the Euclidean norm of the weight vector \mathbf{w} .

The goal of SVM is to find the optimal hyperplane, subject to the constraint:

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] \geq 1$$

$$i = 1, 2, \dots, N$$

Where $d(i)$ is the class label of sample $\mathbf{x}(i)$. It takes the value of $+1$ or -1 for class 1 and 2, respectively.

5.2.1 The primal problem

The constrained optimization problem of SVM may be stated as:

Given the training samples $\{\mathbf{x}(i), d(i)\}$, $i = 1, 2, \dots, N$, find the optimal values of the weight vector \mathbf{w} and bias b such that they satisfy the constraints

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] \geq 1 \quad \text{for } i = 1, 2, \dots, N$$

and the weight vector minimizes the following cost function:

$$J(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

This constrained optimization problem has the following characteristics:

- ❑ The cost function $J(\mathbf{w})$ is a convex function of \mathbf{w} ;
- ❑ The constraints are linear in \mathbf{w} .

We may solve the constrained optimization problem using the *method of Lagrange multipliers*:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha(i) (d(i) [\mathbf{w}^T \mathbf{x}(i) + b] - 1)$$

where

$$\alpha(i) \geq 0$$

The auxiliary nonnegative variables $\alpha(i)$ are called *Lagrange multipliers*.

Differentiating $L(\mathbf{w}, b, \alpha)$ with respect to \mathbf{w} and b and setting the results to zero, yields two conditions of optimality:

$$\text{Condition 1: } \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0$$

$$\text{Condition 2: } \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0$$

From Condition 1:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} &= \frac{1}{2} \times 2 \times \mathbf{w} - \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}(i) \\ &= \mathbf{w} - \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}(i) = 0 \end{aligned}$$

Hence:

$$\mathbf{w} = \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}(i)$$

From Condition 2:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = - \sum_{i=1}^N \alpha(i) d(i) = 0$$

Hence:

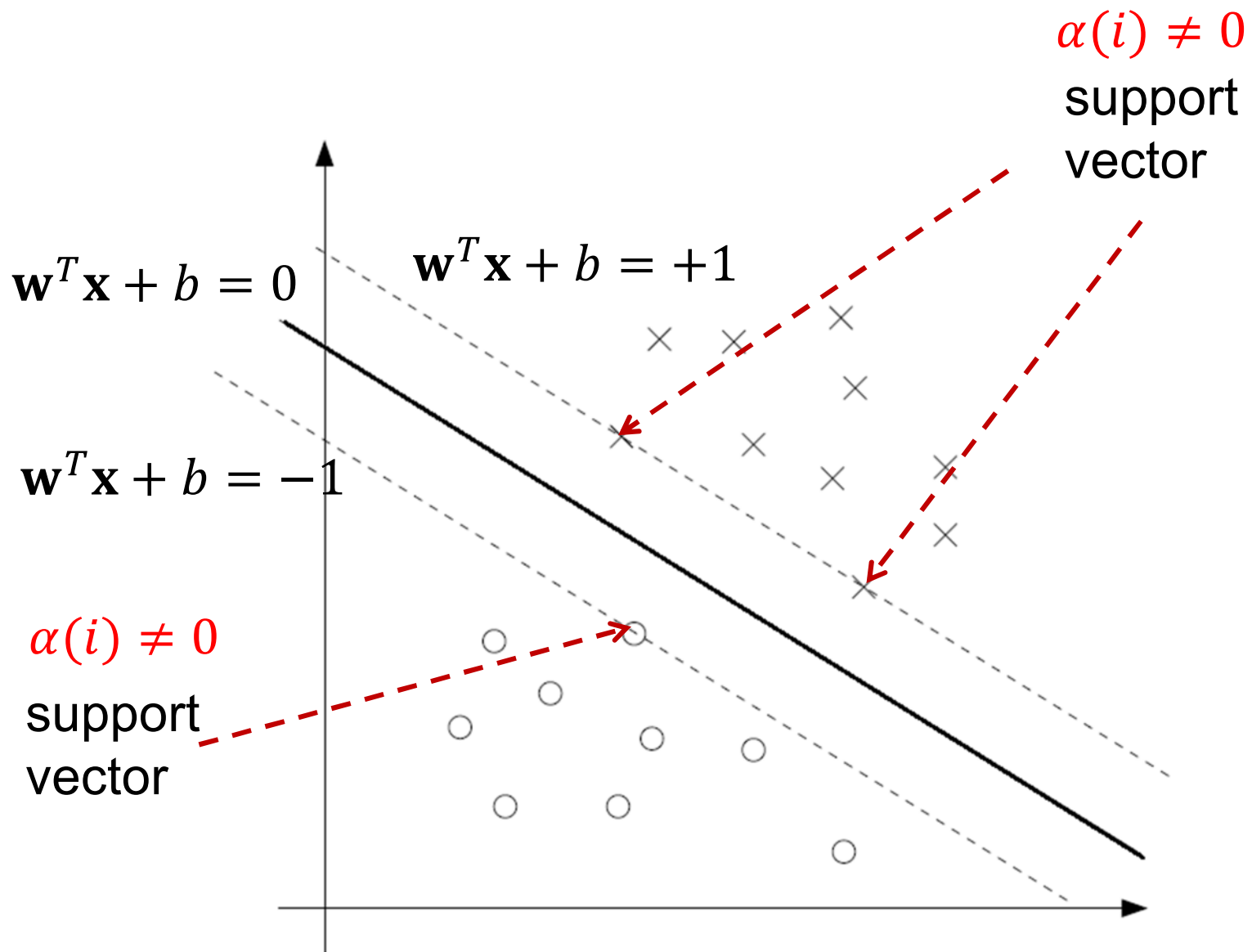
$$\sum_{i=1}^N \alpha(i) d(i) = 0$$

Following Karush-Kuhn-Tucker (KKT) optimization theory (see the appendix), at the optimal solution, we have:

$$\alpha(i) \{d(i) [\mathbf{w}^T \mathbf{x}(i) + b] - 1\} = 0$$

$$\alpha(i) \geq 0$$

which means that $\alpha(i)$ will be nonzero only for points satisfying the equality in the constraint.



We expand $L(\mathbf{w}, b, \alpha)$:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha(i) d(i) \mathbf{w}^T \mathbf{x}(i) - b \sum_{i=1}^N \alpha(i) d(i) + \sum_{i=1}^N \alpha(i)$$

The third term in the above expansion is zero by condition 2. By condition 1, for the second term above, we have:

$$\begin{aligned} \sum_{i=1}^N \alpha(i) d(i) \mathbf{w}^T \mathbf{x}(i) &= \sum_{i=1}^N \alpha(i) d(i) \sum_{j=1}^N \alpha(j) d(j) \mathbf{x}^T(j) \mathbf{x}(i) \\ &= \sum_{i=1}^N \sum_{j=1}^N \alpha(i) d(i) \alpha(j) d(j) \mathbf{x}^T(j) \mathbf{x}(i) \end{aligned}$$

Furthermore, by the optimality condition 1, we have:

$$\begin{aligned}\mathbf{w}^T \mathbf{w} &= \mathbf{w}^T \sum_{j=1}^N \alpha(j) d(j) \mathbf{x}(j) = \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}^T(i) \sum_{j=1}^N \alpha(j) d(j) \mathbf{x}(j) \\ &= \sum_{i=1}^N \sum_{j=1}^N \alpha(i) d(i) \alpha(j) d(j) \mathbf{x}^T(i) \mathbf{x}(j)\end{aligned}$$

Substituting the above into the expansion of $L(\mathbf{w}, b, \alpha)$, yields:

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^N \alpha(i) - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha(i) \alpha(j) d(i) d(j) \mathbf{x}^T(i) \mathbf{x}(j)$$

5.2.2 The dual problem

We may solve the *dual problem*, whose solution is equal to the solution of the original problem (*primal* problem).

The dual problem is stated as follows:

$$\min_{\alpha} Q(\alpha) = \min_{\alpha} \left\{ \sum_{i=1}^N \alpha(i) - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha(i) \alpha(j) d(i) d(j) \mathbf{x}^T(i) \mathbf{x}(j) \right\}$$

Subject to the following conditions:

$$(a) \quad \sum_{i=1}^N \alpha(i) d(i) = 0$$

$$(b) \quad \alpha(i) \geq 0$$

The dual (also primal) problem is a quadratic programming (QP) problem, we can use available QP software to solve it to get $\alpha(i)$.

After having determined the optimum Lagrange multipliers $\alpha(i)$, we may compute the optimal weight vector \mathbf{w} :

$$\mathbf{w}^* = \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}(i) = \sum_{\alpha(i) > 0} \alpha(i) d(i) \mathbf{x}(i)$$

For **support vectors, i.e. samples with $\alpha(i) > 0$**

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] = 1$$

Therefore:

$$b^* = \frac{1}{d(i)} - (\mathbf{w}^*)^T \mathbf{x}(i) = d(i) - (\mathbf{w}^*)^T \mathbf{x}(i)$$

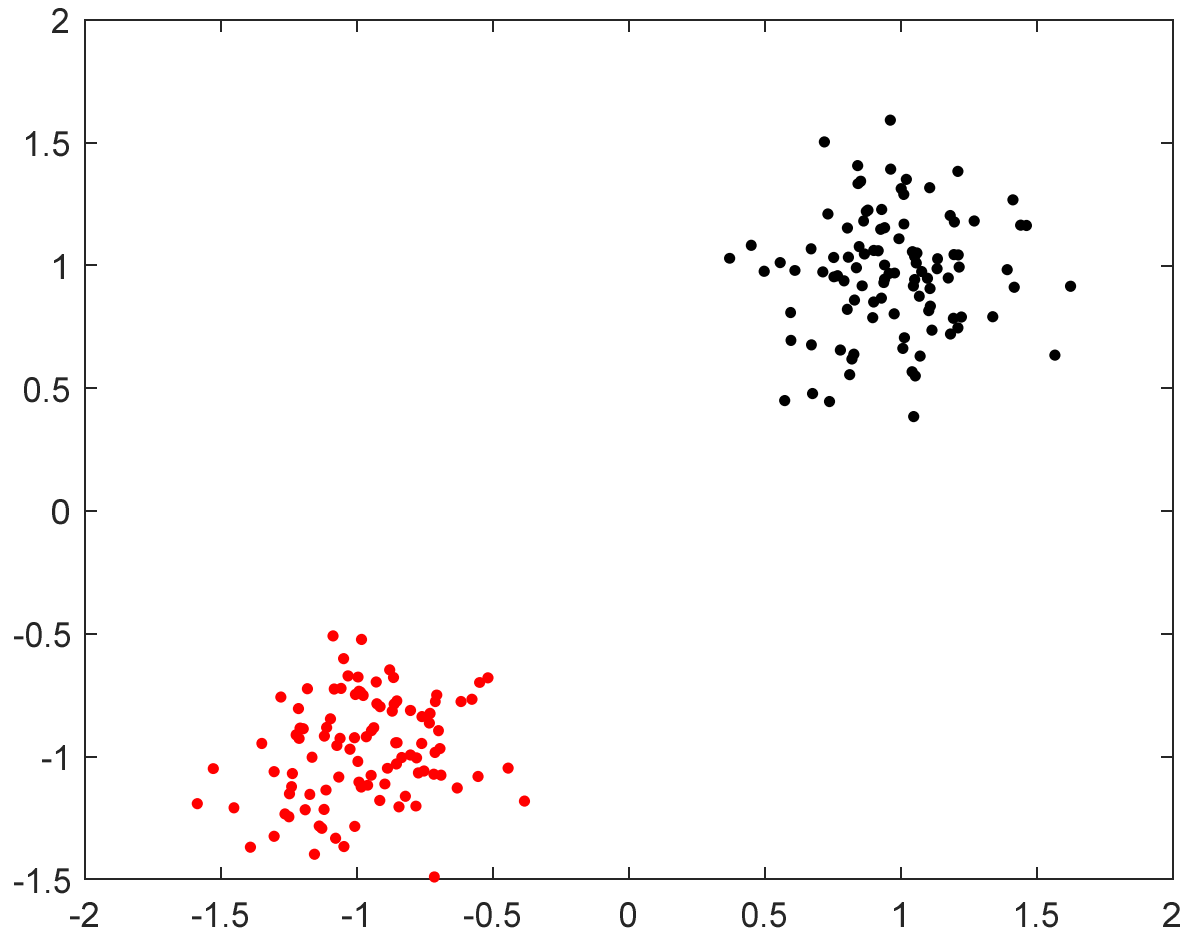
In practice, due to computation errors, we use each of the support vectors to calculate a value of the bias term and then take the average:

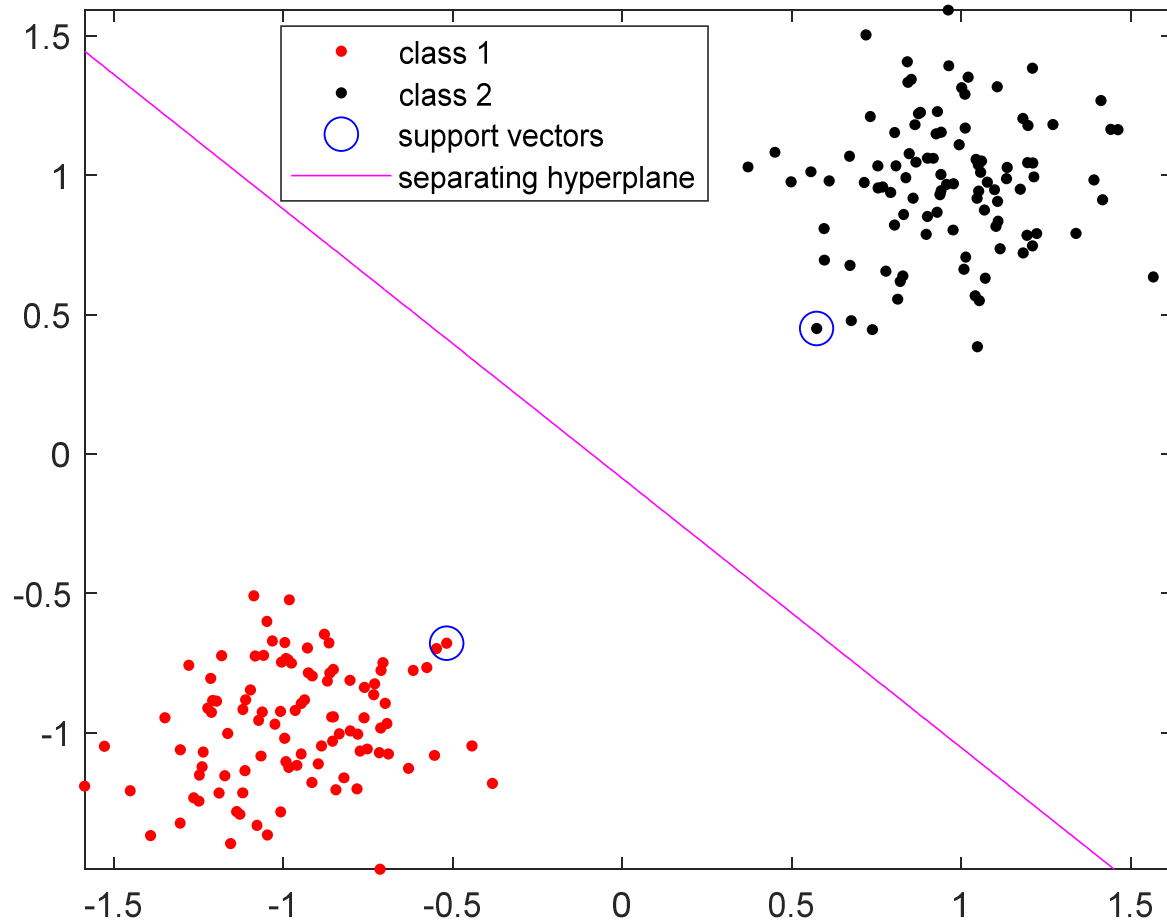
$$b^* = \frac{1}{N_s} \sum_{\alpha(i) > 0} [d(i) - (\mathbf{w}^*)^T \mathbf{x}(i)]$$

Where N_s is the total number of support vectors.

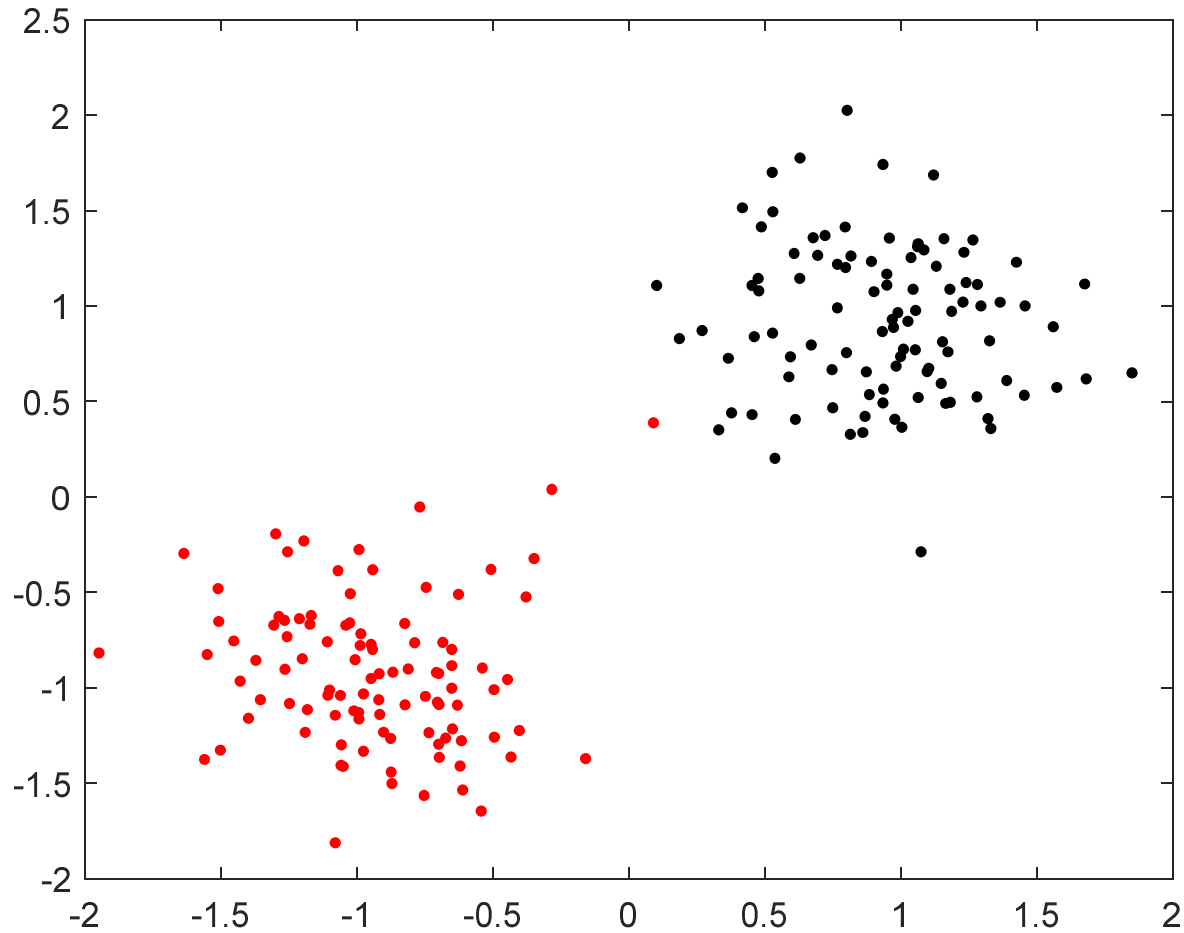
Actually, most of the $\alpha(i)$ are zeros. The **samples $\mathbf{x}(i)$ with nonzero $\alpha(i)$ are the support vectors.** In other words, the **optimal hyperplane is determined by the support vectors only.**

Example 1 of linearly separable samples





Example 2 of linearly separable



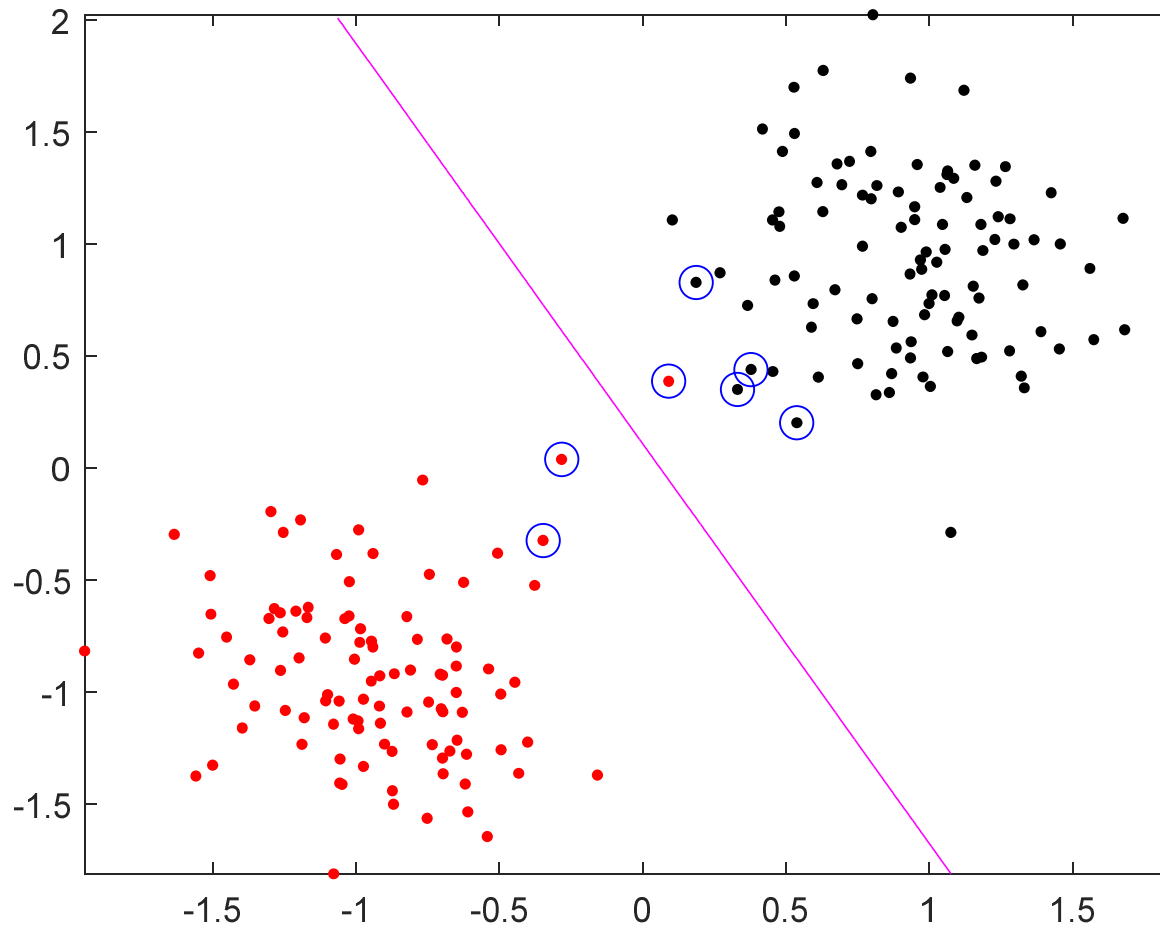
Discussion:

For this data, we have two hyperplanes:

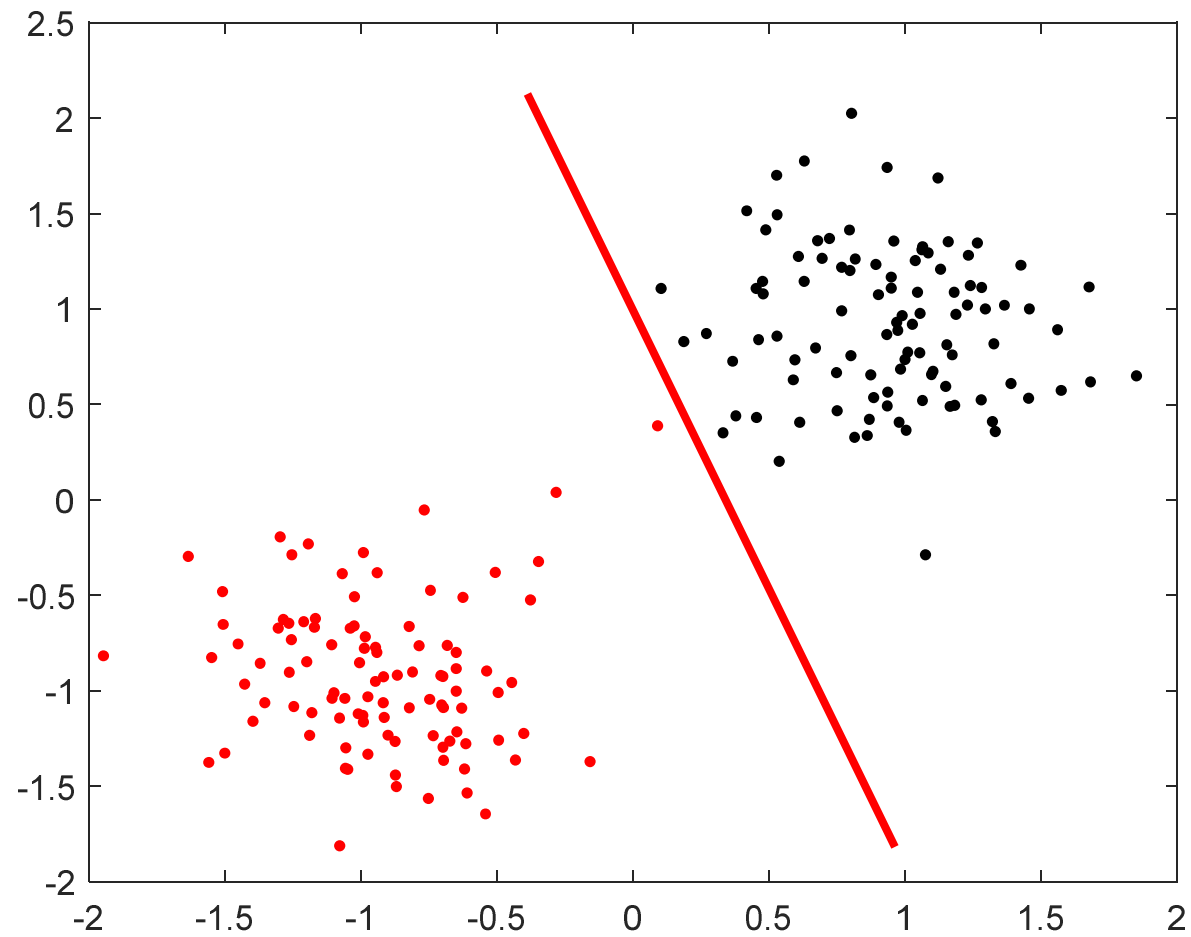
(1) Which one is preferred?

(2) Why?

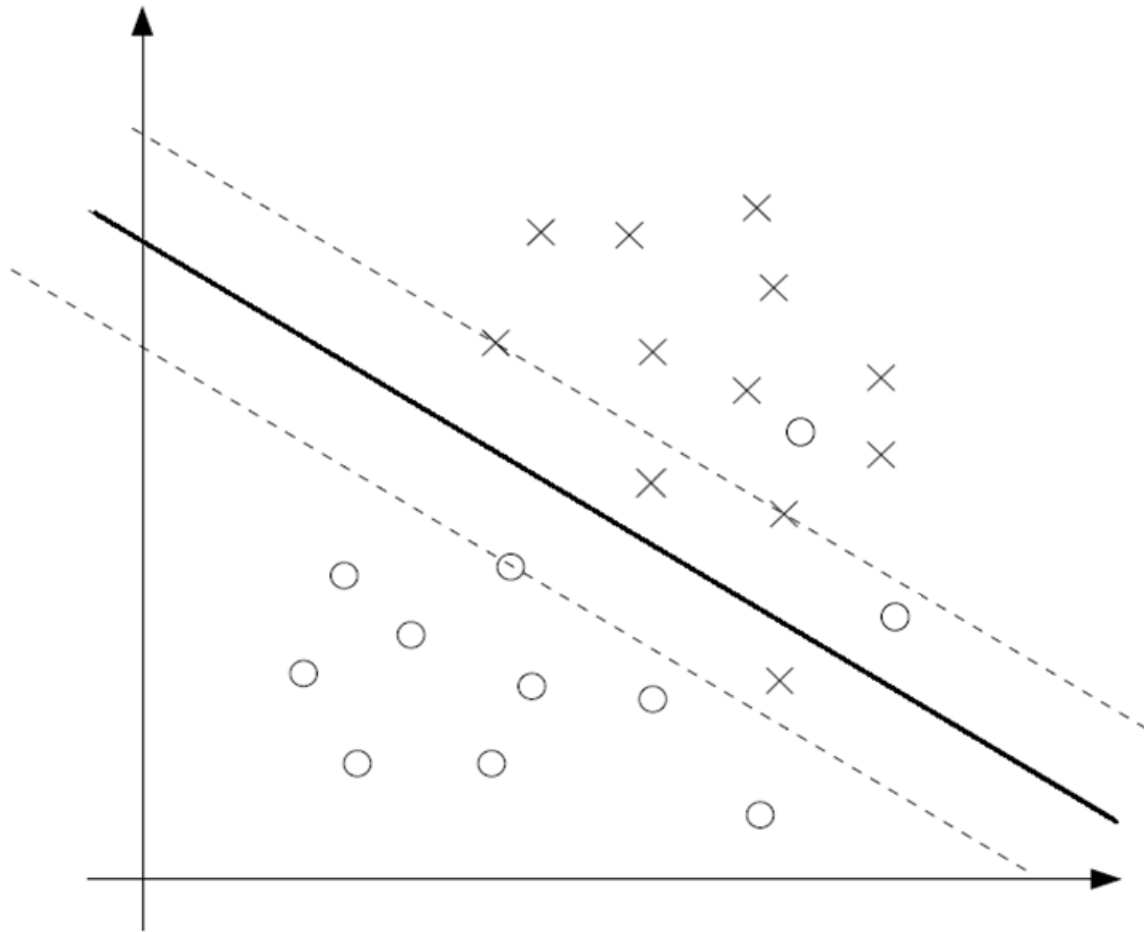
1 error for the training data.



0 error for training data



5.3 Optimal hyperplane for non-separable samples

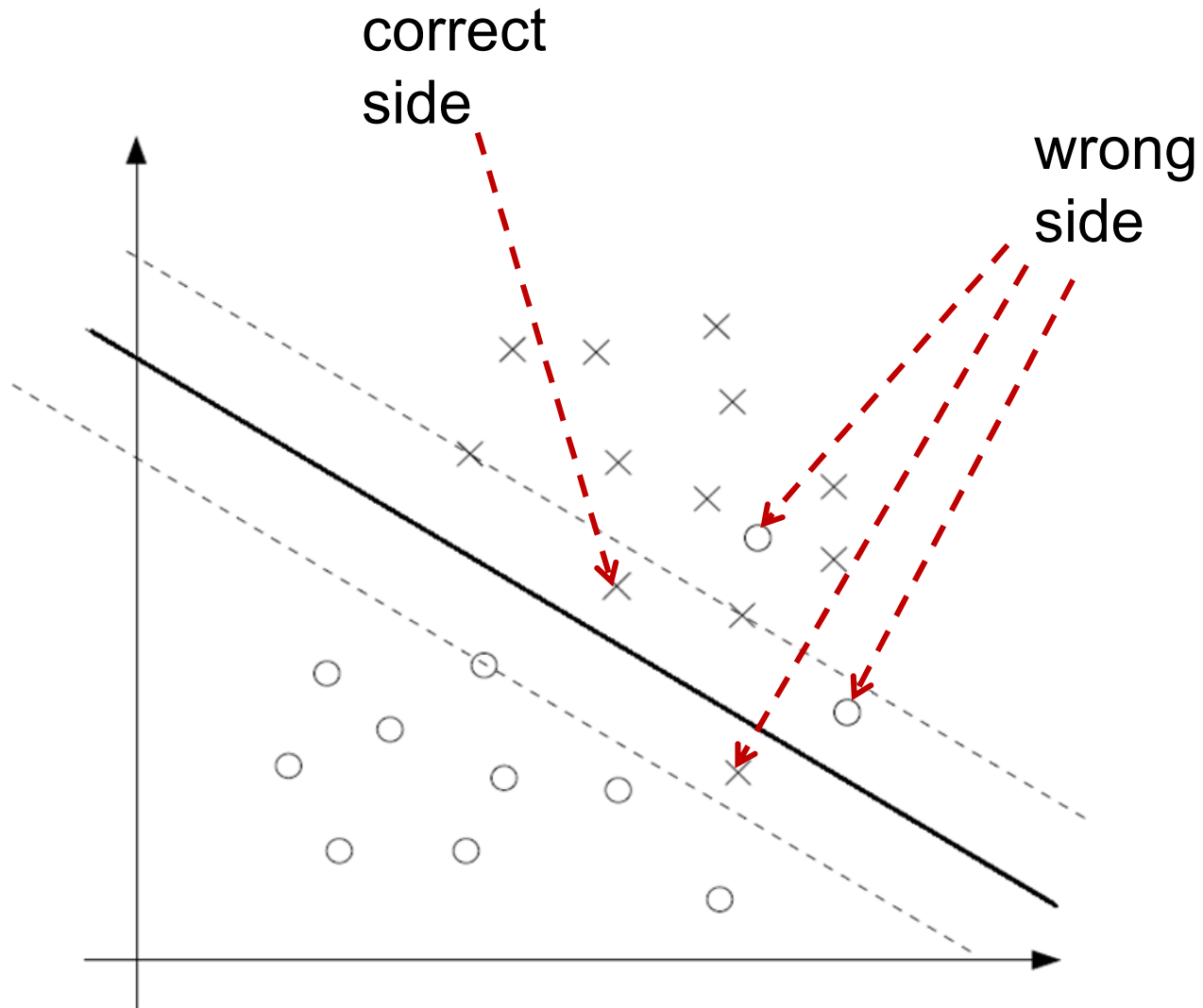


The margin of separation between classes is said to be soft if a data point $\{\mathbf{x}(i), d(i)\}$, $i = 1, 2, \dots, N$, violates the following conditions:

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] \geq 1$$

The violation can arise in the following two ways:

- (a) The data point $\mathbf{x}(i)$ falls inside the region of separation but **on the correct side** of the hyperplane;
- (b) The data point $\mathbf{x}(i)$ falls **on the wrong side** of the hyperplane.



We introduce a new set of **nonnegative** scalar variables $\xi(i)$, $i = 1, 2, \dots, N$, into the definition of the separating hyperplane as shown below:

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] \geq 1 - \xi(i)$$

$\xi(i)$ is called **slack variable**, which measures the deviation of data point $\mathbf{x}(i)$ from the ideal condition of pattern separability.

There are two cases for $\xi(i)$:

- (a) For $0 \leq \xi(i) \leq 1$, the data point falls inside the region of separation but on the **correct side** of the hyperplane.
- (b) For $\xi(i) > 1$, it falls on the **wrong side** of the separating hyperplane.

5.3.1 The primal problem for non-separable samples

We may formulate the linear support vector machines for non-separable samples as the following constrained optimization problem:

$$\min_{\mathbf{w}, b, \xi} \{J(\mathbf{w}, b, \xi)\} = \min_{\mathbf{w}, b, \xi(i)} \left\{ \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi(i) \right\}$$

Subject to conditions:

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] \geq 1 - \xi(i)$$

$$\xi(i) \geq 0$$

Where parameter C is a hyperparameter, which is given by user or determined using validation data.

Introducing Lagrange multipliers $\alpha(i)$ and $\beta(i)$, we obtain:

$$L = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi(i) - \sum_{i=1}^N \beta(i) \xi(i) - \sum_{i=1}^N \alpha(i) \{d(i) [\mathbf{w}^T \mathbf{x}(i) + b] - 1 + \xi(i)\}$$

Following Karush-Kuhn-Tucker (KKT) optimization theory, at the optimal solution, we have:

$$\alpha(i) \{d(i) [\mathbf{w}^T \mathbf{x}(i) + b] - 1 + \xi(i)\} = 0$$

$$\beta(i) \xi(i) = 0$$

$$\alpha(i) \geq 0$$

$$\beta(i) \geq 0$$

5.3.2 The dual problem for non-separable samples

The dual problem for non-separable samples is formulated as follows (deviation is similar to the case of separable samples, details are skipped here)

$$\min_{\alpha} Q(\alpha) = \min_{\alpha} \left\{ \sum_{i=1}^N \alpha(i) - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha(i) \alpha(j) d(i) d(j) \mathbf{x}^T(i) \mathbf{x}(j) \right\}$$

Subject to conditions:

$$\sum_{i=1}^N \alpha(i) d(i) = 0$$

$$0 \leq \alpha(i) \leq C$$

Again, we can solve the QP problem to get $\alpha(i)$. Then the optimal weight vector \mathbf{w} is obtained as :

$$\mathbf{w}^* = \sum_{i=1}^N \alpha(i) d(i) \mathbf{x}(i) = \sum_{\alpha(i) > 0} \alpha(i) d(i) \mathbf{x}(i)$$

Next, we find b^* . For $\beta(i)$, we can find it by solving:

$$\frac{\partial L}{\partial \xi(i)} = C - \beta(i) - \alpha(i) = 0$$

Hence:

$$\beta(i) = C - \alpha(i)$$

According to KKT condition:

$$\beta(i) \xi(i) = [C - \alpha(i)] \xi(i) = 0$$

Thus, for the support vectors with $\alpha(i) > 0$, we have two cases to consider:

(i) $\xi(i) > 0$, then $[C - \alpha(i)] = 0$, hence $\alpha(i) = C$, or

(ii) $C - \alpha(i) > 0$, then $\xi(i) = 0$. These are precisely the **support vectors that are on the margin**.

Using **those support vectors that are on the margin**, that is $0 < \alpha(i) < C$ and $\xi(i) = 0$, we can solve for the bias term:

$$d(i) \times [\mathbf{w}^T \mathbf{x}(i) + b] = 1$$

Therefore,

$$b^* = \frac{1}{d(i)} - (\mathbf{w}^*)^T \mathbf{x}(i) = d(i) - (\mathbf{w}^*)^T \mathbf{x}(i)$$

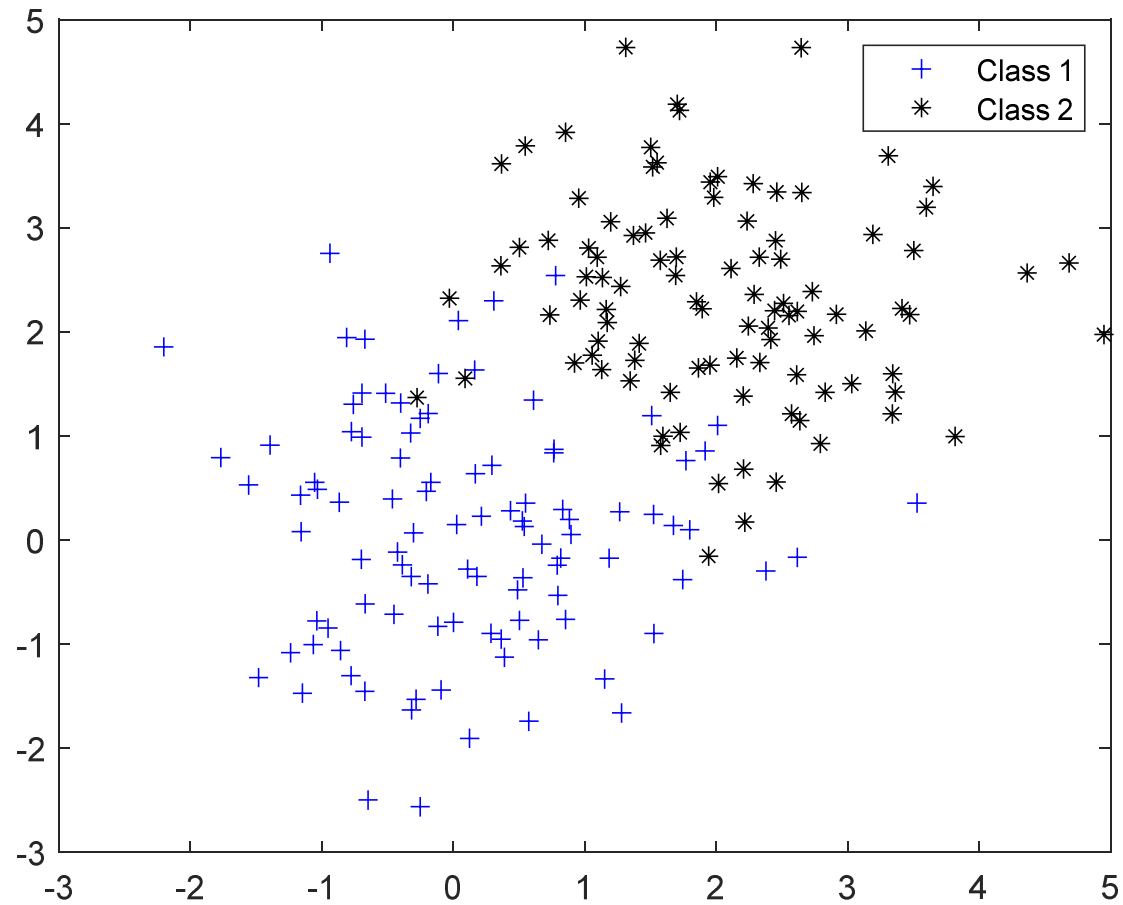
Note: $\mathbf{x}(i)$ is a support vector with $0 < \alpha(i) < C$.

In practice, due to computation errors, we use each of the support vectors with $0 < \alpha(i) < C$ to calculate a value of the bias term and then take the average:

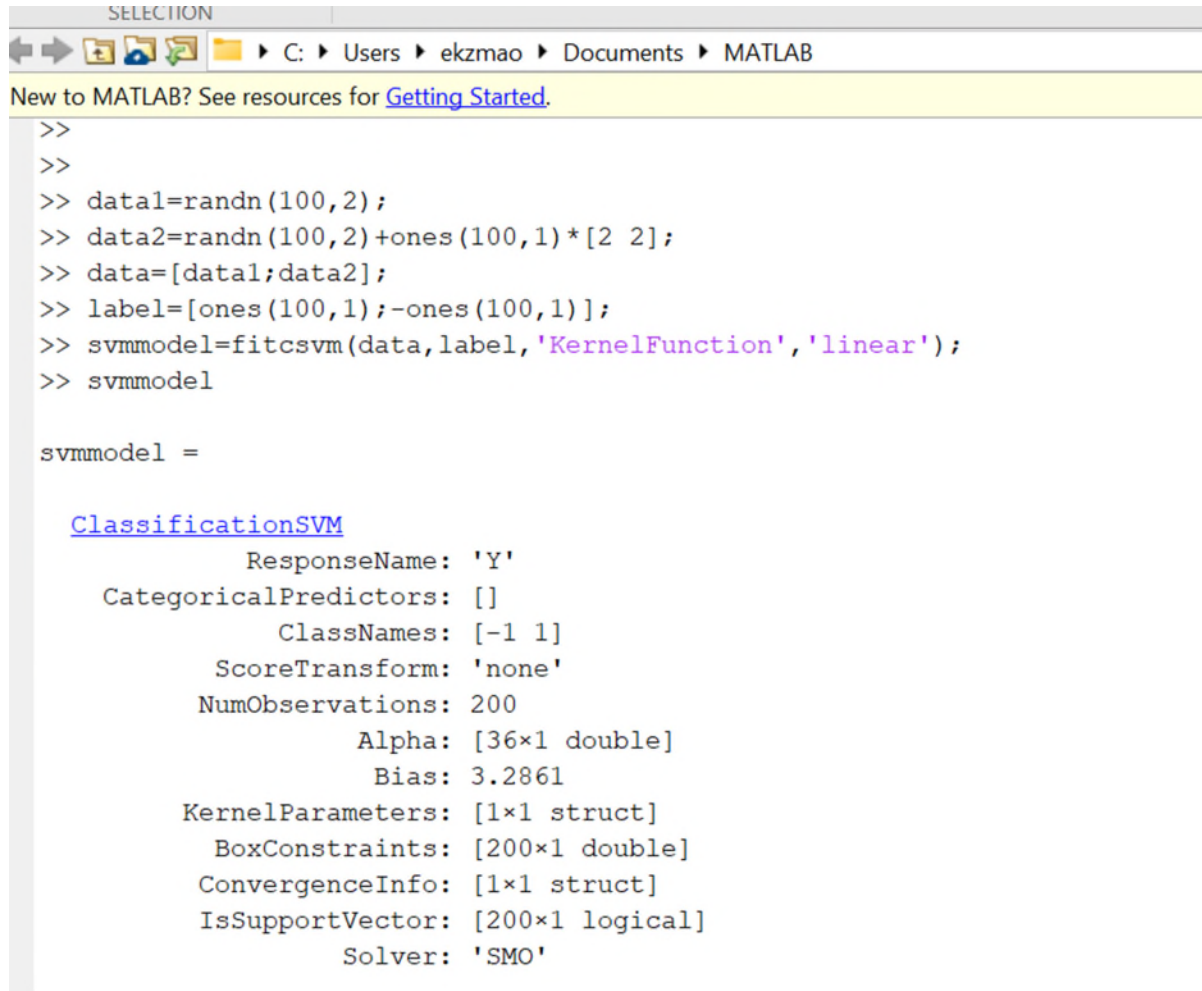
$$b^* = \frac{1}{N_s} \sum_{0 < \alpha(i) < C} [d(i) - (\mathbf{w}^*)^T \mathbf{x}(i)]$$

Where N_s is the total number of support vectors with Lagrange multiplier $0 < \alpha(i) < C$.

Example of linearly non-separable samples



Next, Matlab is used as an example to illustrate some properties of SVM (can be skipped)



```
SELECTION
C:\Users\ekzmao\Documents\MATLAB
New to MATLAB? See resources for Getting Started.

>>
>>
>> data1=randn(100,2);
>> data2=randn(100,2)+ones(100,1)*[2 2];
>> data=[data1;data2];
>> label=[ones(100,1);-ones(100,1)];
>> svmmodel=fitcsvm(data,label,'KernelFunction','linear');
>> svmmodel

svmmodel =

ClassificationSVM
    ResponseName: 'Y'
    CategoricalPredictors: []
        ClassNames: [-1 1]
    ScoreTransform: 'none'
    NumObservations: 200
           Alpha: [36x1 double]
           Bias: 3.2861
    KernelParameters: [1x1 struct]
    BoxConstraints: [200x1 double]
    ConvergenceInfo: [1x1 struct]
    IsSupportVector: [200x1 logical]
           Solver: 'SMO'
```

```
svmmodel =
```

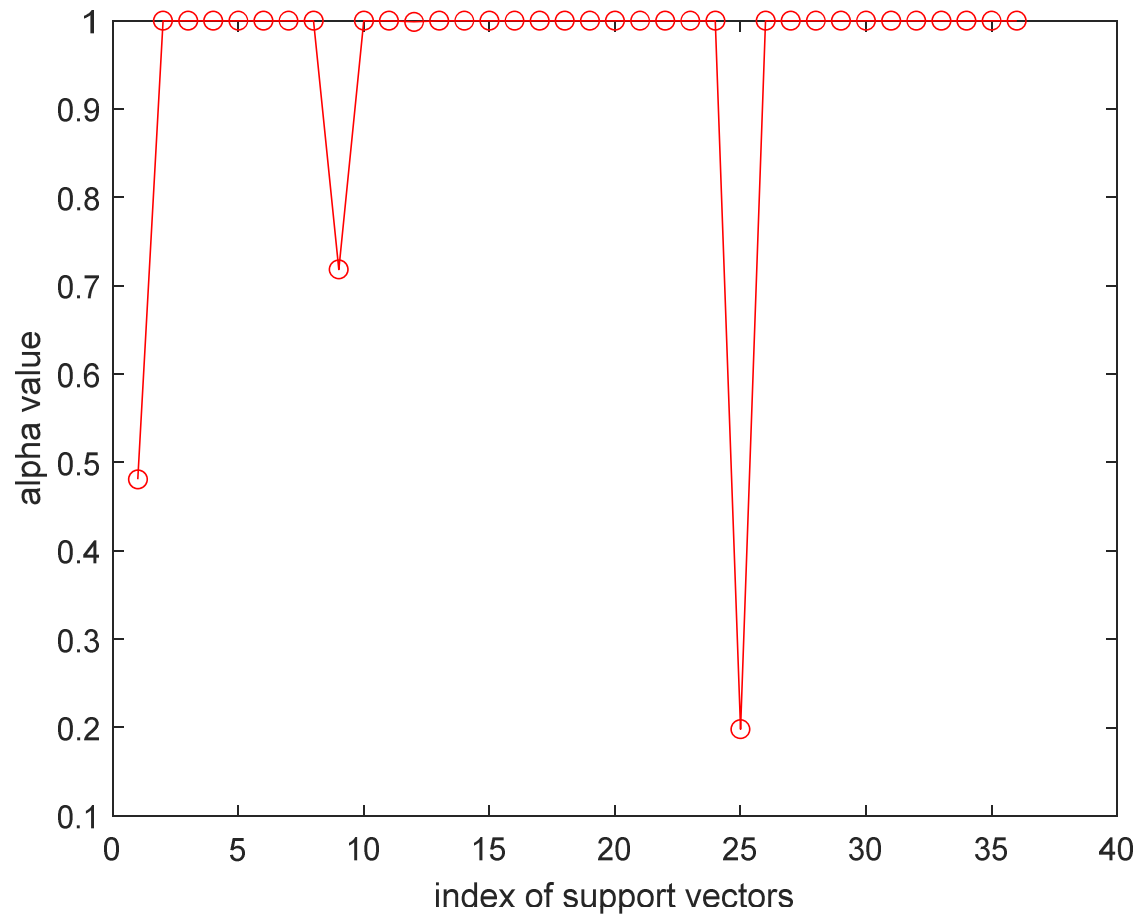
ClassificationSVM

```
    ResponseName: 'Y'  
    CategoricalPredictors: []  
        ClassNames: [-1 1]  
    ScoreTransform: 'none'  
    NumObservations: 200  
        Alpha: [36×1 double]  
        Bias: 3.2861  
    KernelParameters: [1×1 struct]  
        BoxConstraints: [200×1 double]  
    ConvergenceInfo: [1×1 struct]  
    IsSupportVector: [200×1 logical]  
        Solver: 'SMO'
```

Properties, Methods

```
>> alpha=svmmodel.Alpha;  
>> plot(alpha, 'ro-')
```

Value of $\alpha > 0$. Three alpha is in the range of $0 < \alpha < C$, where $C = 1$.



```
svmmodel =
```

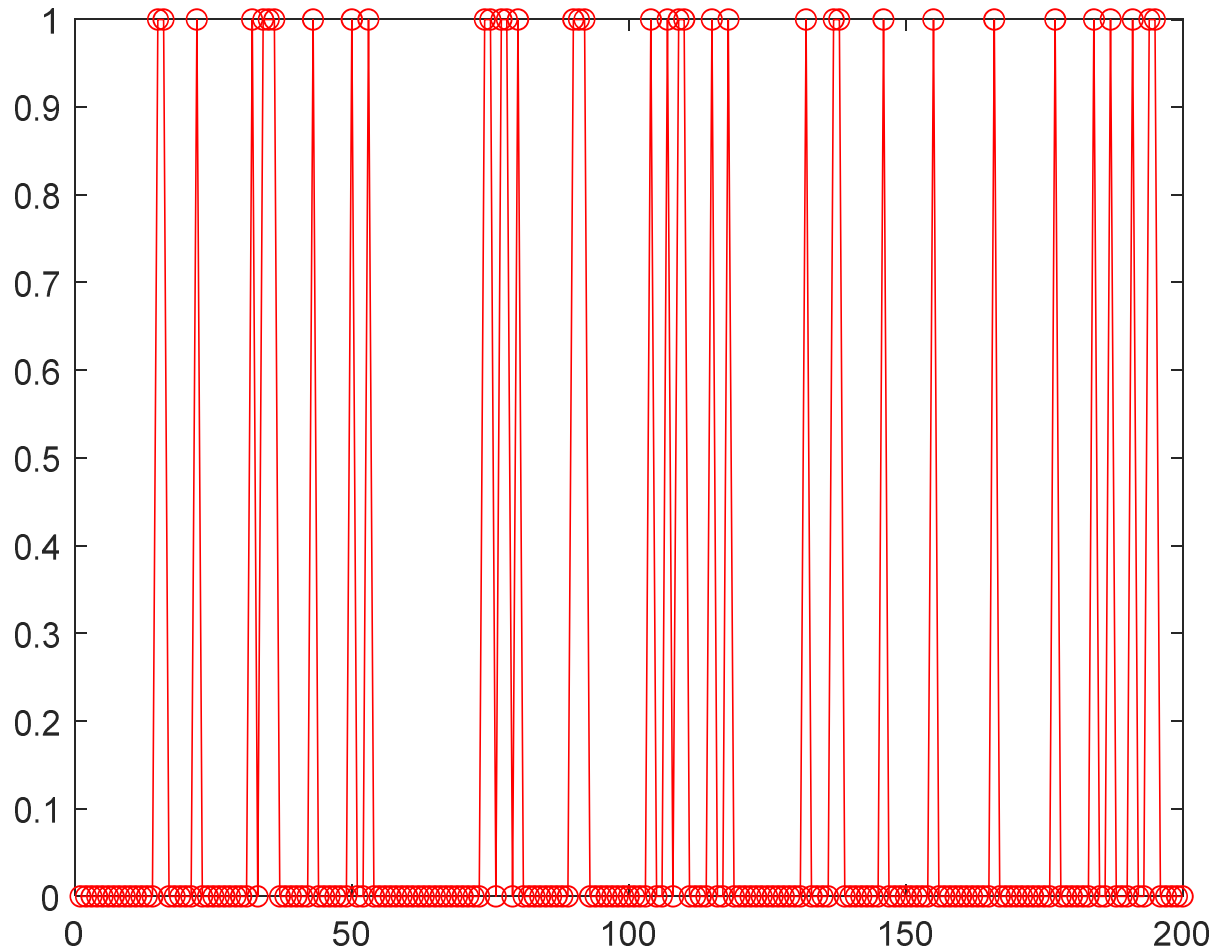
ClassificationSVM

```
    ResponseName: 'Y'  
    CategoricalPredictors: []  
        ClassNames: [-1 1]  
    ScoreTransform: 'none'  
    NumObservations: 200  
        Alpha: [36×1 double]  
        Bias: 3.2861  
    KernelParameters: [1×1 struct]  
        BoxConstraints: [200×1 double]  
    ConvergenceInfo: [1×1 struct]  
    IsSupportVector: [200×1 logical]  
        Solver: 'SMO'
```

Properties, Methods

```
>> figure(3)  
>> plot(svmmodel.IsSupportVector, 'ro-')
```

1 indicates the sample is a support vector ($\alpha > 0$)
0 indicates the sample is not a support vector ($\alpha = 0$)



```
svmmodel =
```

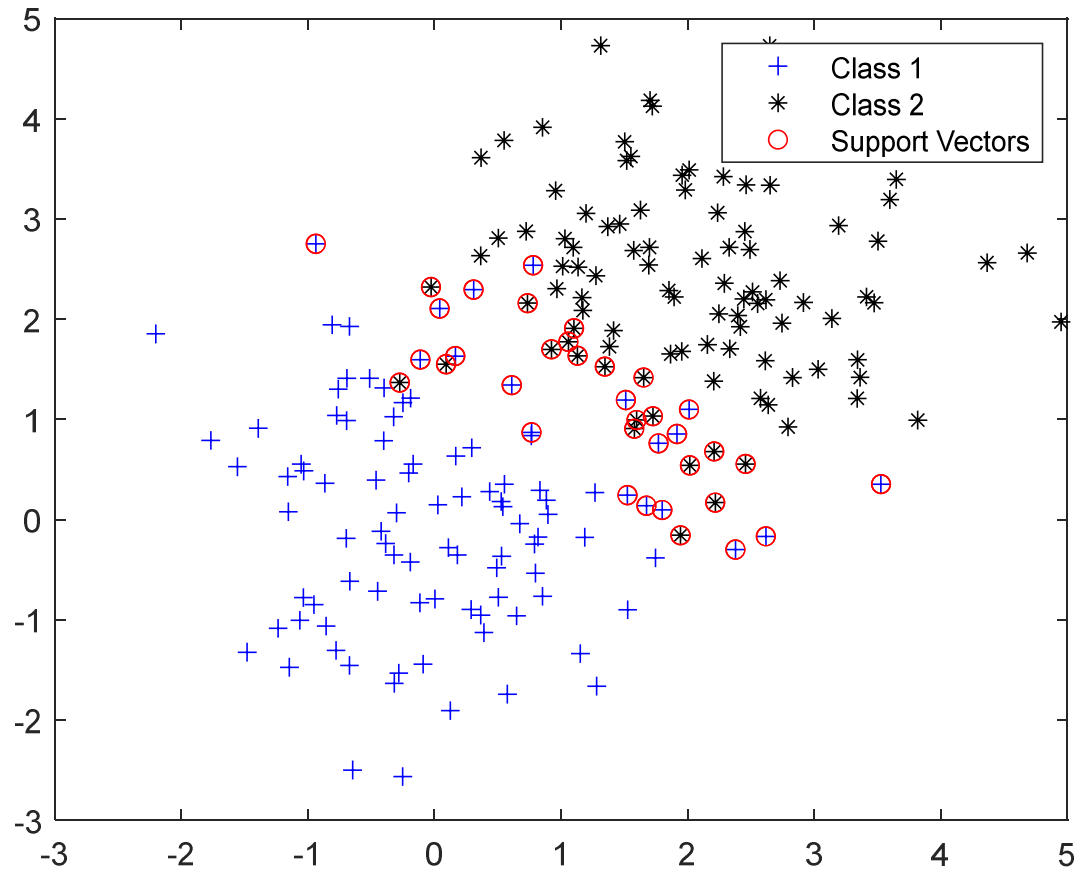
ClassificationSVM

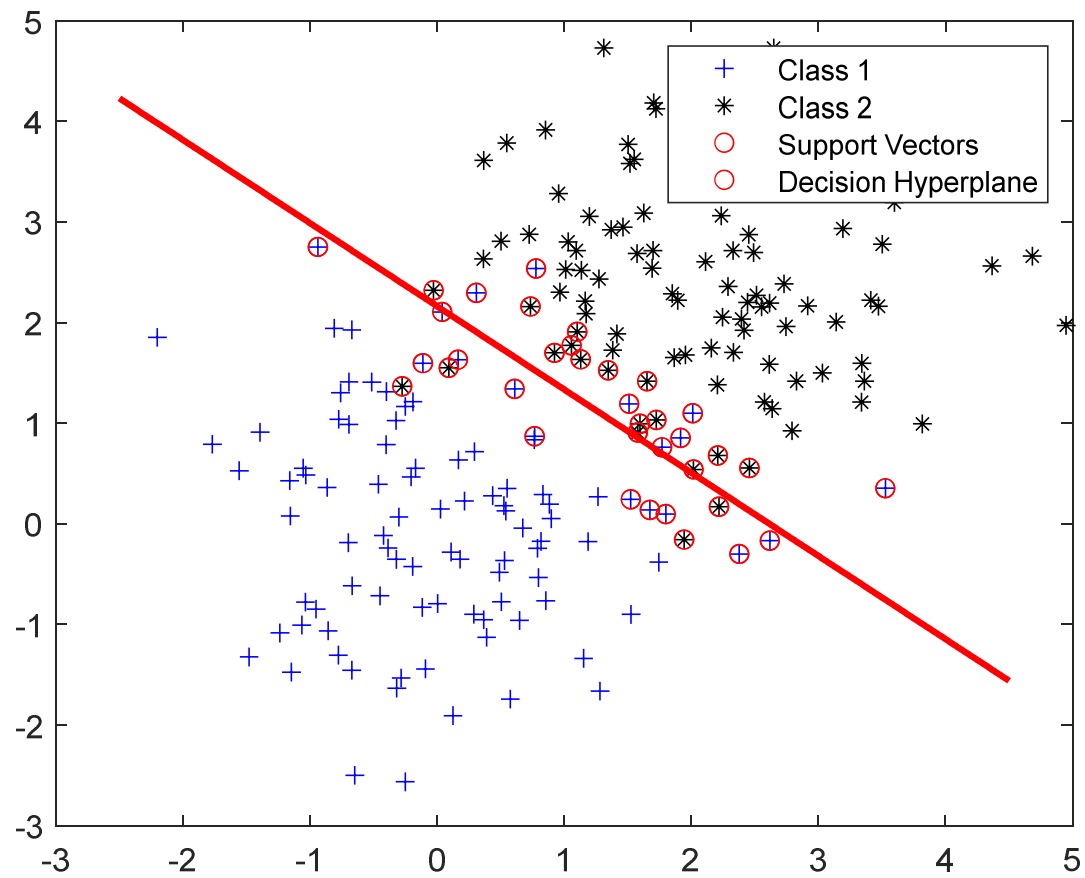
```
    ResponseName: 'Y'  
    CategoricalPredictors: []  
        ClassNames: [-1 1]  
    ScoreTransform: 'none'  
    NumObservations: 200  
        Alpha: [36×1 double]  
        Bias: 3.2861  
    KernelParameters: [1×1 struct]  
        BoxConstraints: [200×1 double]  
    ConvergenceInfo: [1×1 struct]  
    IsSupportVector: [200×1 logical]  
        Solver: 'SMO'
```

Properties, Methods

```
>> I=find(svmmodel.IsSupportVector==1);  
>> SV=data(I,:);  
>> plot(SV(:,1),SV(:,2),'ro')  
>> legend('Class 1','Class 2','Support Vectors')
```

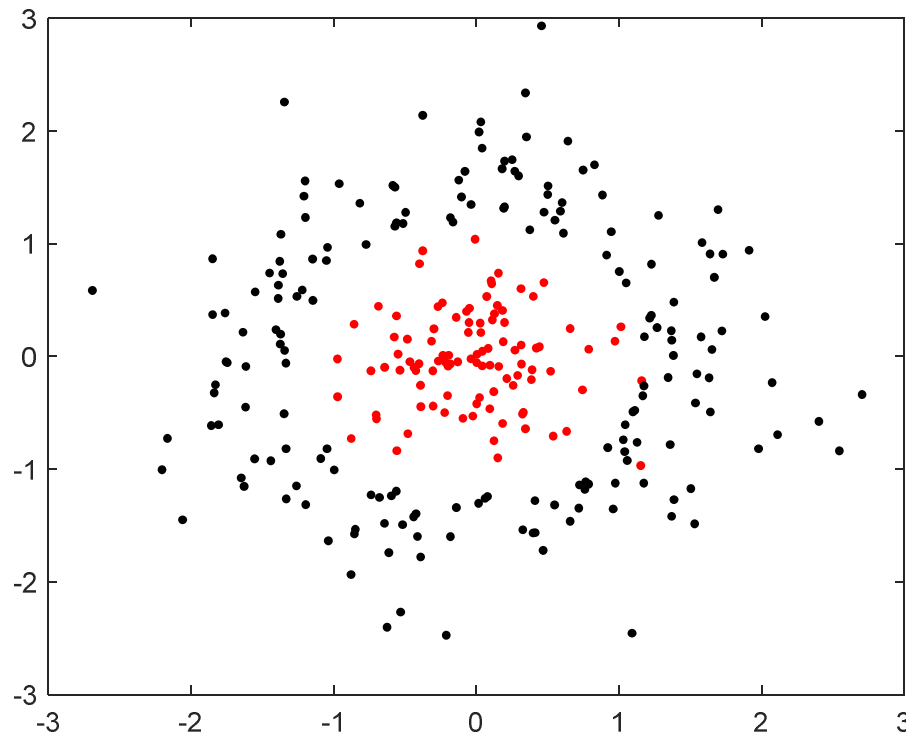
Support vectors



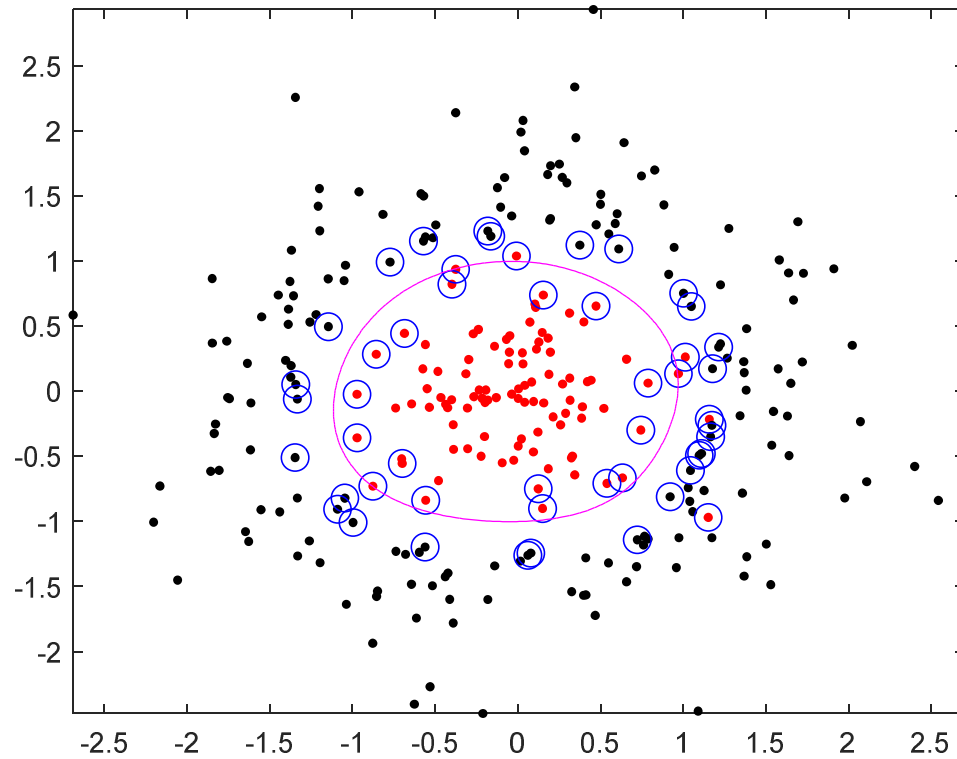


Kernel Support Vector Machines

In the above, linear SVM is presented. If the data is linearly inseparable as shown below, how to construct a classifier to classify the data?



The solution is kernel support vector machines.



Kernel support vector machines is introduced in

EE7207 Neural Networks and Deep Learning