4. Support Vector Machines

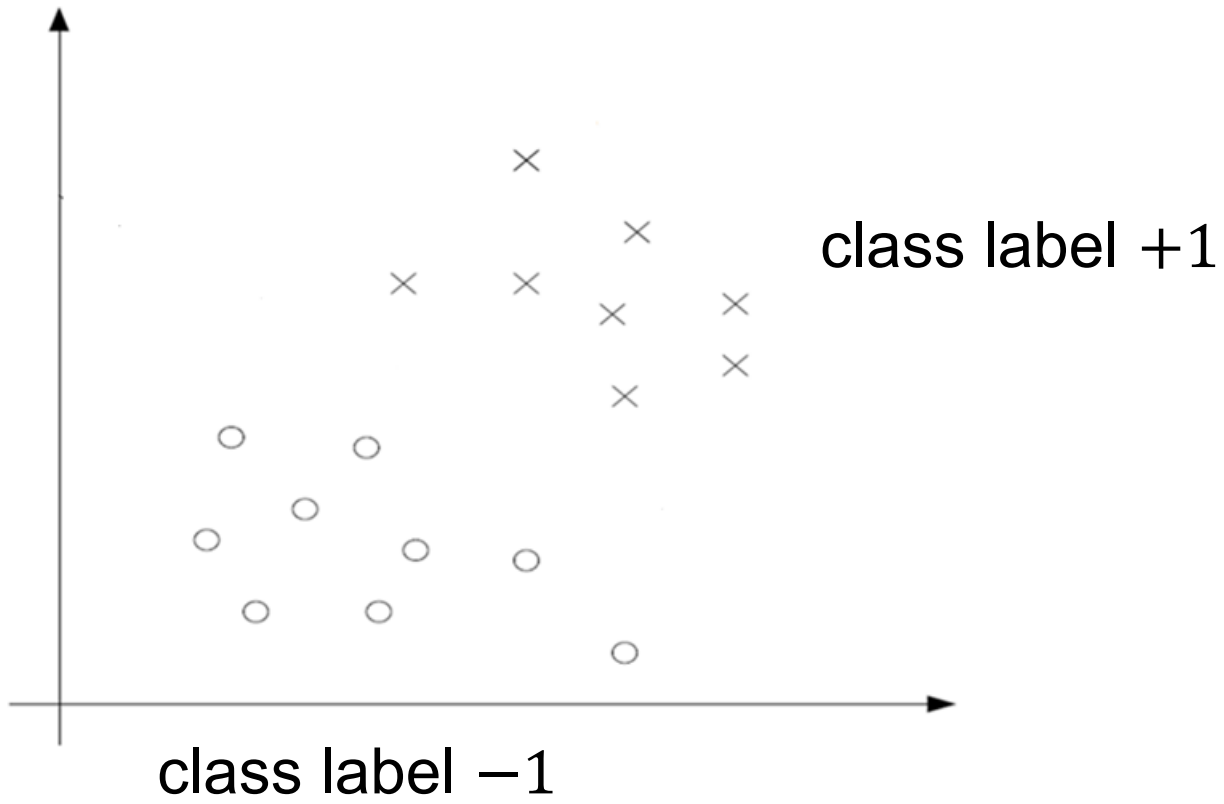
4.1 Introduction

In previous part, radial basis function (RBF) neural network was introduced. In this lecture, we study another type of feedforward neural network, known as support vector machines (SVM). Like RBF neural networks, SVM can be used for pattern classification and regression.

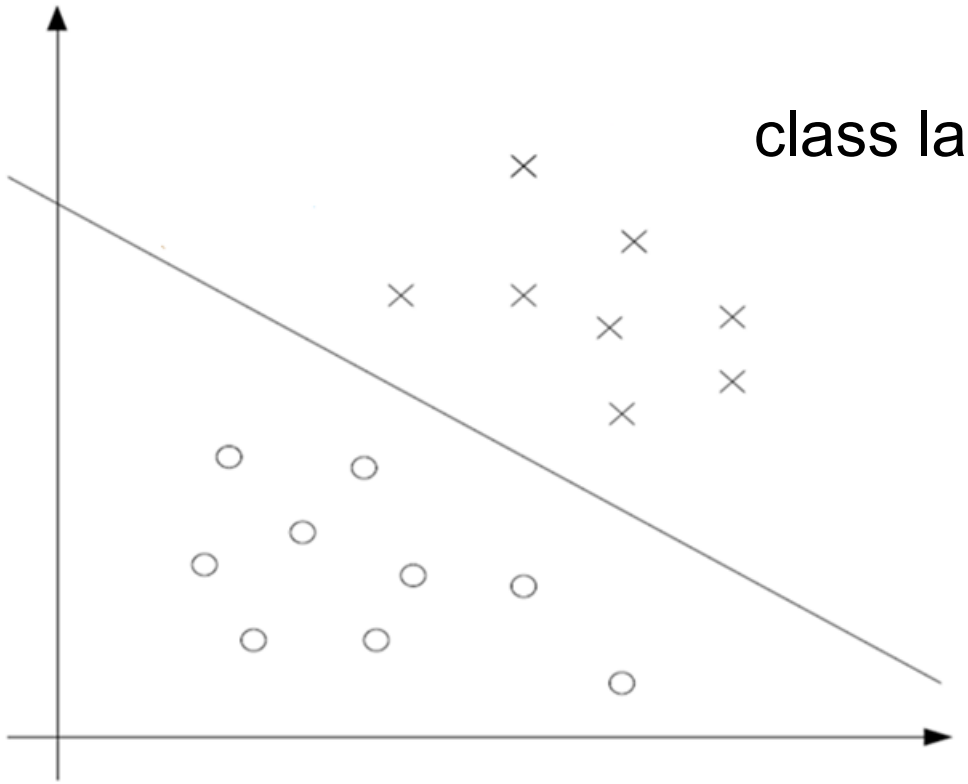
Kernel support vector machine has a very similar architecture to the RBF neural networks, but their learning principles are very different. Kernel SVM is an extension of linear SVM. Next, we first introduce the principle of linear support vector machines (SVM), and then generalize the principle to kernel SVM.

4.2 Motivation

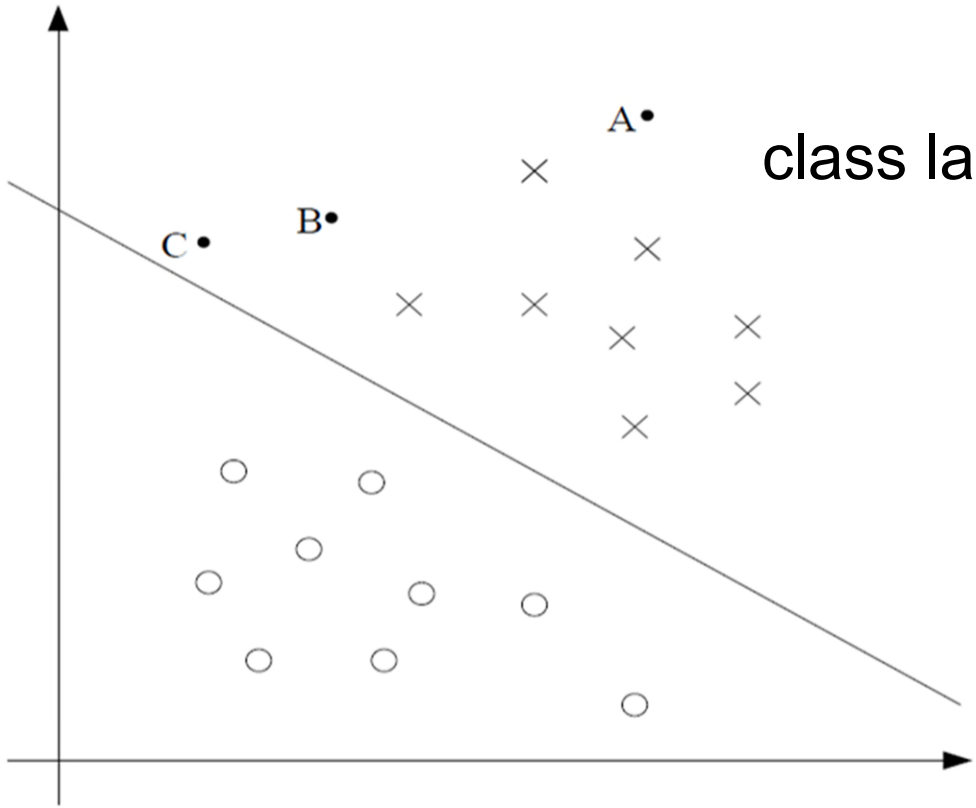
Consider N training samples from two classes, where class label d takes value of $+1$ or -1, respectively



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

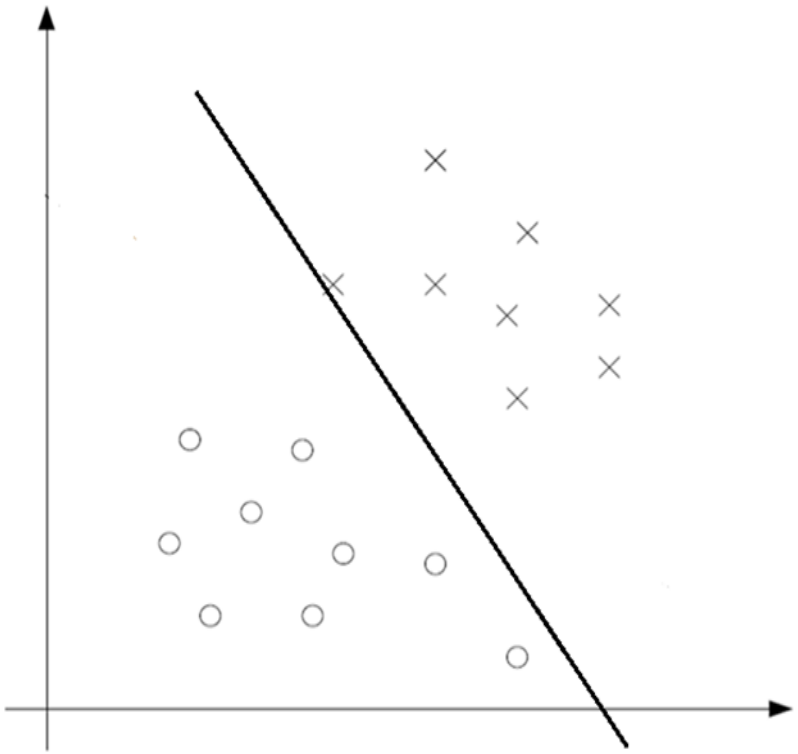
(1) Point A is far from the separating hyperplane. We are quite confident that its label is $+1$.

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

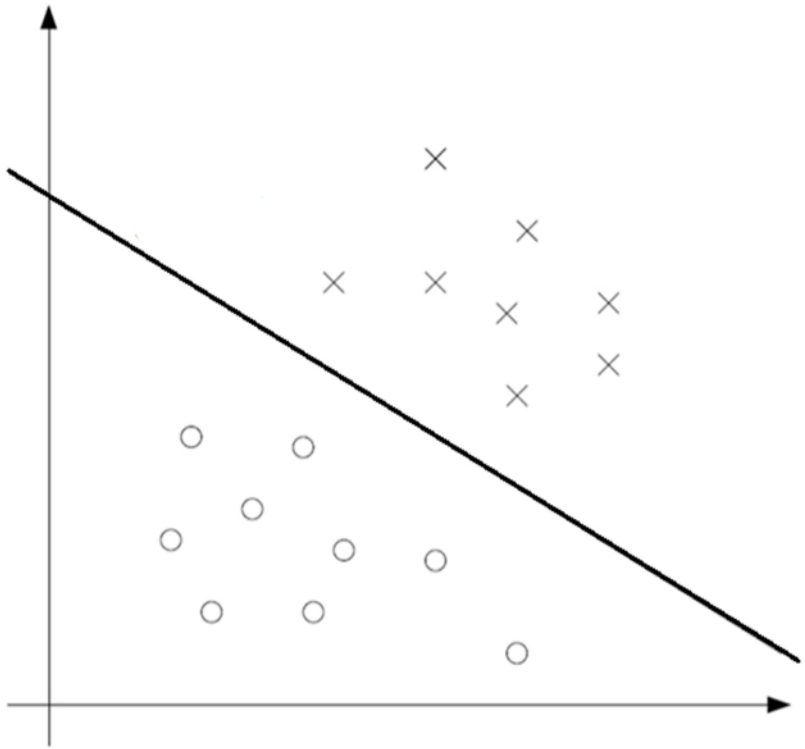
It would be nice if we manage to find a separating plane that allows us to make predictions

(1) correctly, and

(2) confidently (meaning far from the separating hyperplane).



(a)



(b)

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

4.3 Problem formulation

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

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

Where **x** is the feature vector, **w** is the adjustable weight vector, and b is the bias.

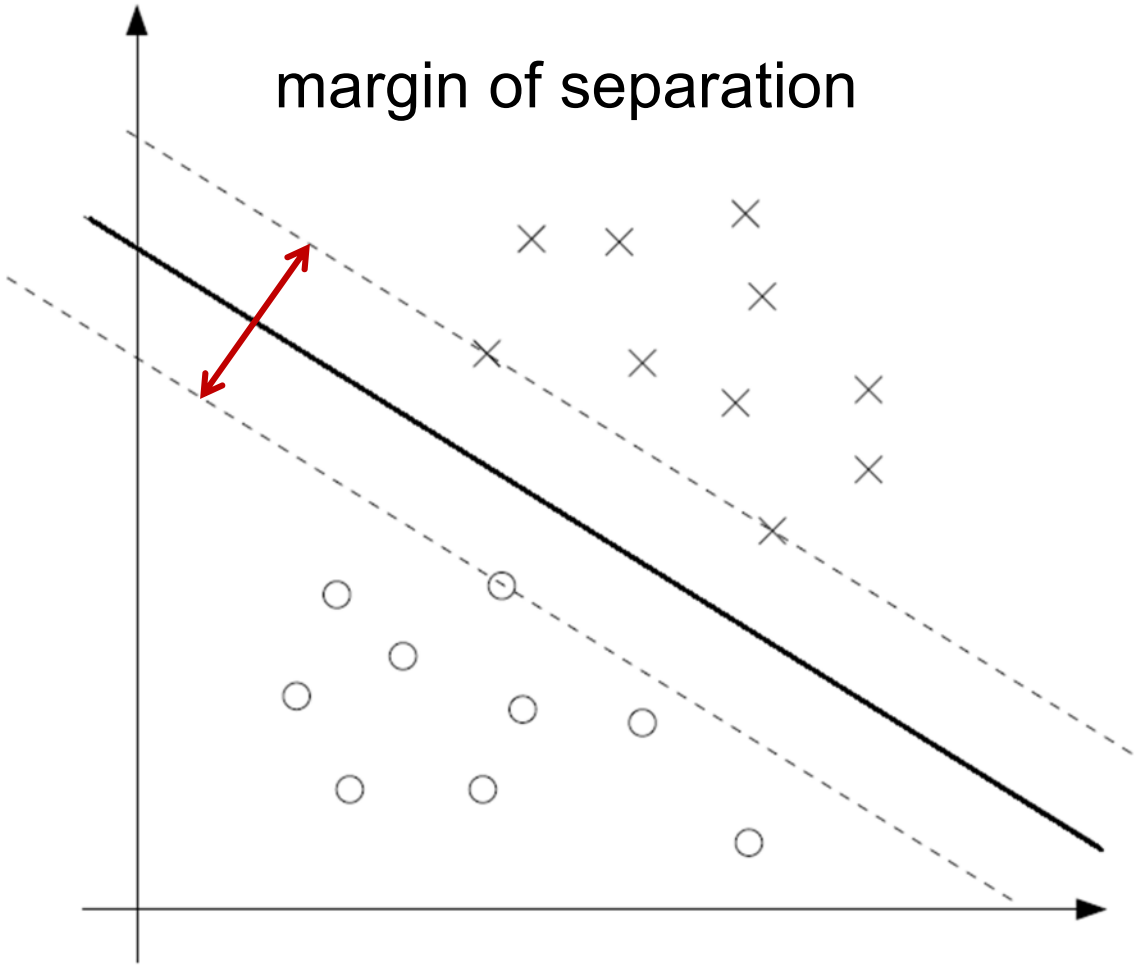
(a) For samples with class label $\mathrm{d} = +1$

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

(b) For samples with class label $\mathrm{d} = -1$

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

The separation between the hyperplane and the closest data points 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.

The discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

gives an algebraic measure of the distance from **x** to the hyperplane.

Let's express **x** as:

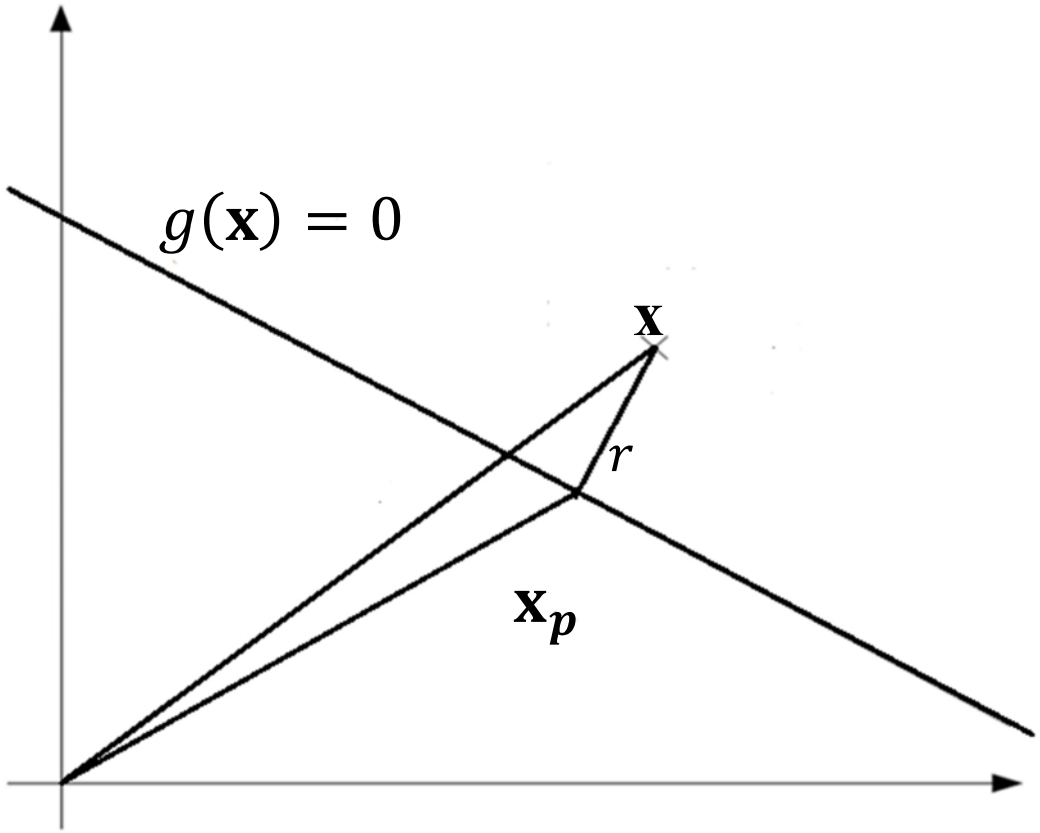
$$\mathbf{x} = \mathbf{x}_p + (\mathbf{x} - \mathbf{x}_p) = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the hyperplane, and r is the distance from **x** to the hyperplane.

r is positive if $\mathbf{x}$ is on the positive side of the optimal hyperplane and is negative if $\mathbf{x}$ is on the negative side.

$\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the hyperplane

r is the distance from $\mathbf{x}$ to the hyperplane



$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

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

$$= \mathbf{w}^T\mathbf{x}_p + b + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}$$

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

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

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

Then we have:

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

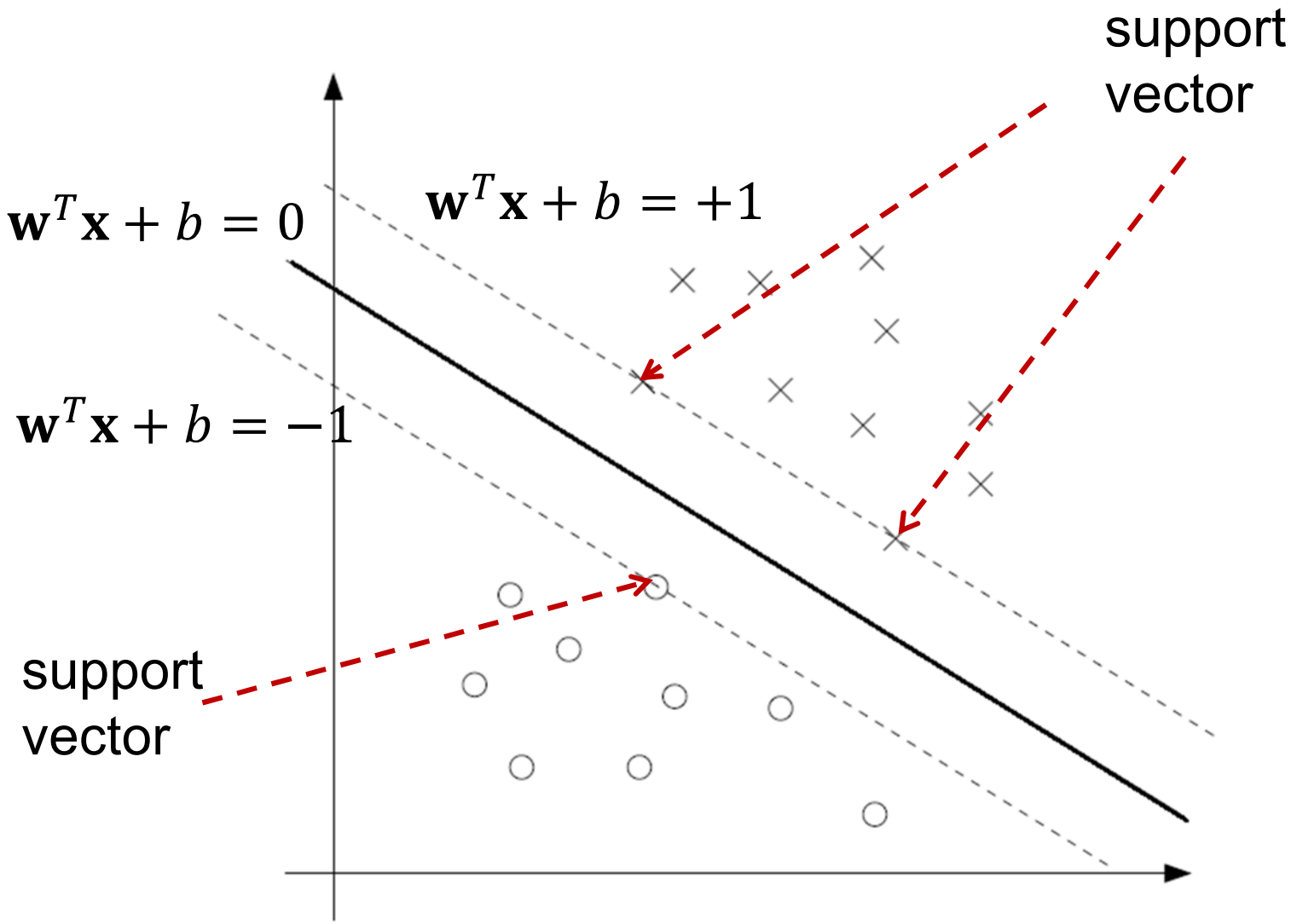
If the data are linearly separable, we can always scale **w** and b so that

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

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

The particular data points for which the above is satisfied with equality sign are called support vectors. The next figure shows the position of the 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.


support
vector
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = +1$
$\mathbf{w}^T\mathbf{x} + b = -1$
support
vector

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

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

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 **w**.

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

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

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

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.

4.3.1 The primal problem

The constrained optimization problem of SVM may be stated as:

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

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

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:

(1) The cost function $J(\mathbf{w})$ is a convex function of $\mathbf{w}$;

(2) The constraints are linear in $\mathbf{w}$.

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

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

where

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

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

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

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

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

From Condition 1:

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

Hence:

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

From Condition 2:

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

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 $J(\boldsymbol{w}, b, \alpha)$:

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

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

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

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

Furthermore, by the optimality condition 1, we have:

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

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

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

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

4.3.2 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:

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

Subject to the following conditions:

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

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

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

After obtaining $\alpha(i)$, we may compute the optimal weight vector **w** and b:

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

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

The bias term can be determined using 1 support vector:

$$(\mathbf{w}^*)^T \mathbf{x}^{(s)} + b^* = d$$

Hence:

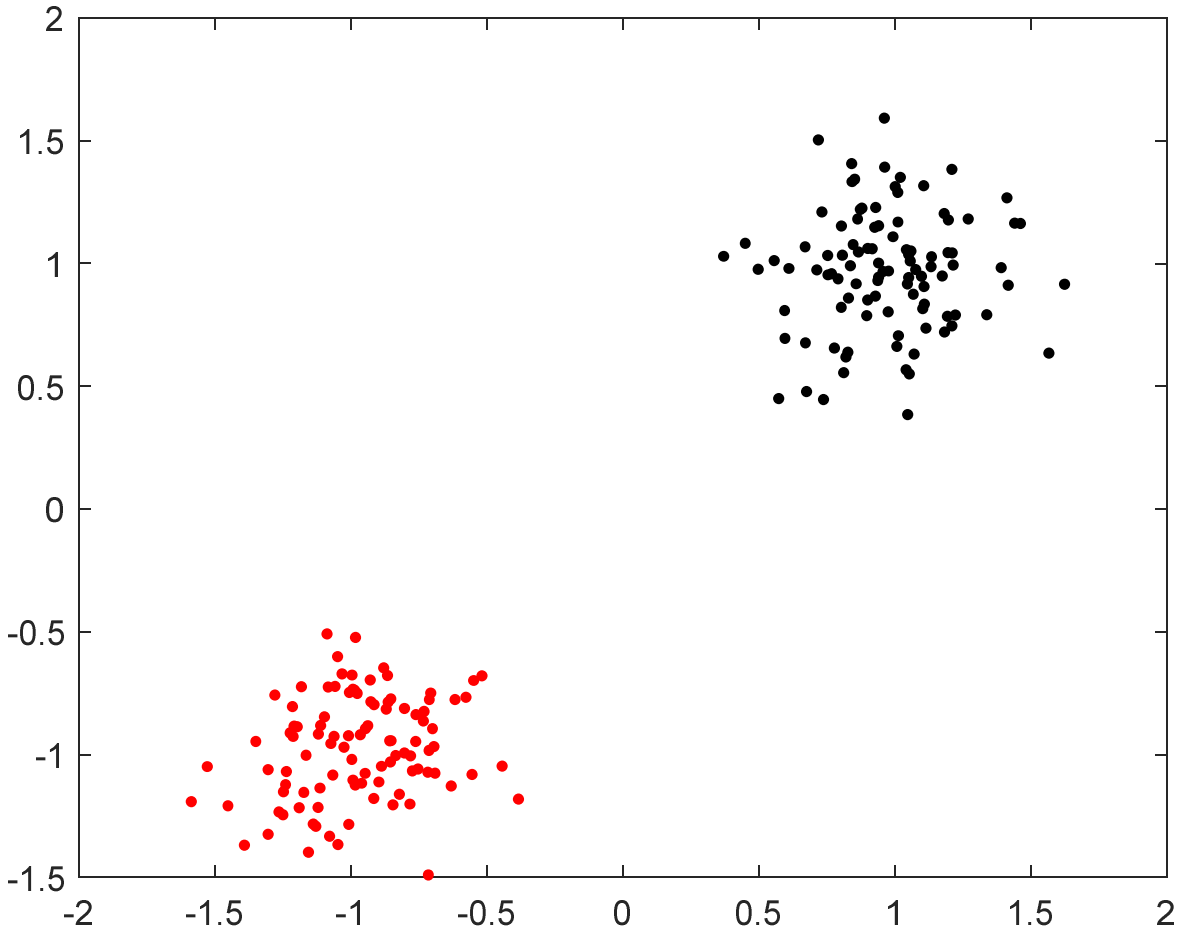
$$b^* = d - (\mathbf{w}^*)^T \mathbf{x}^{(s)}$$

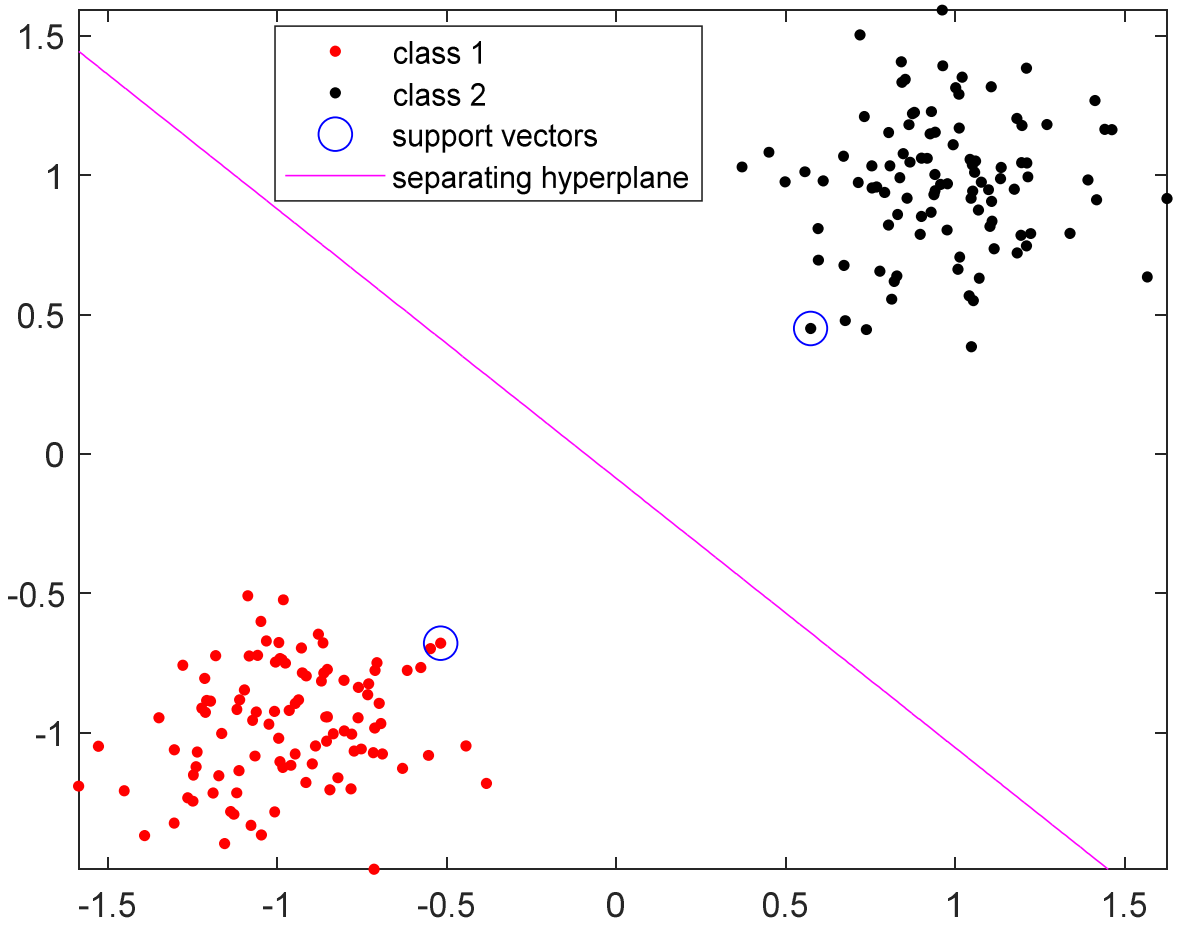
In practice, due to computation errors, we often use each of the support vectors to calculate a value of the bias term and then take the average:

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

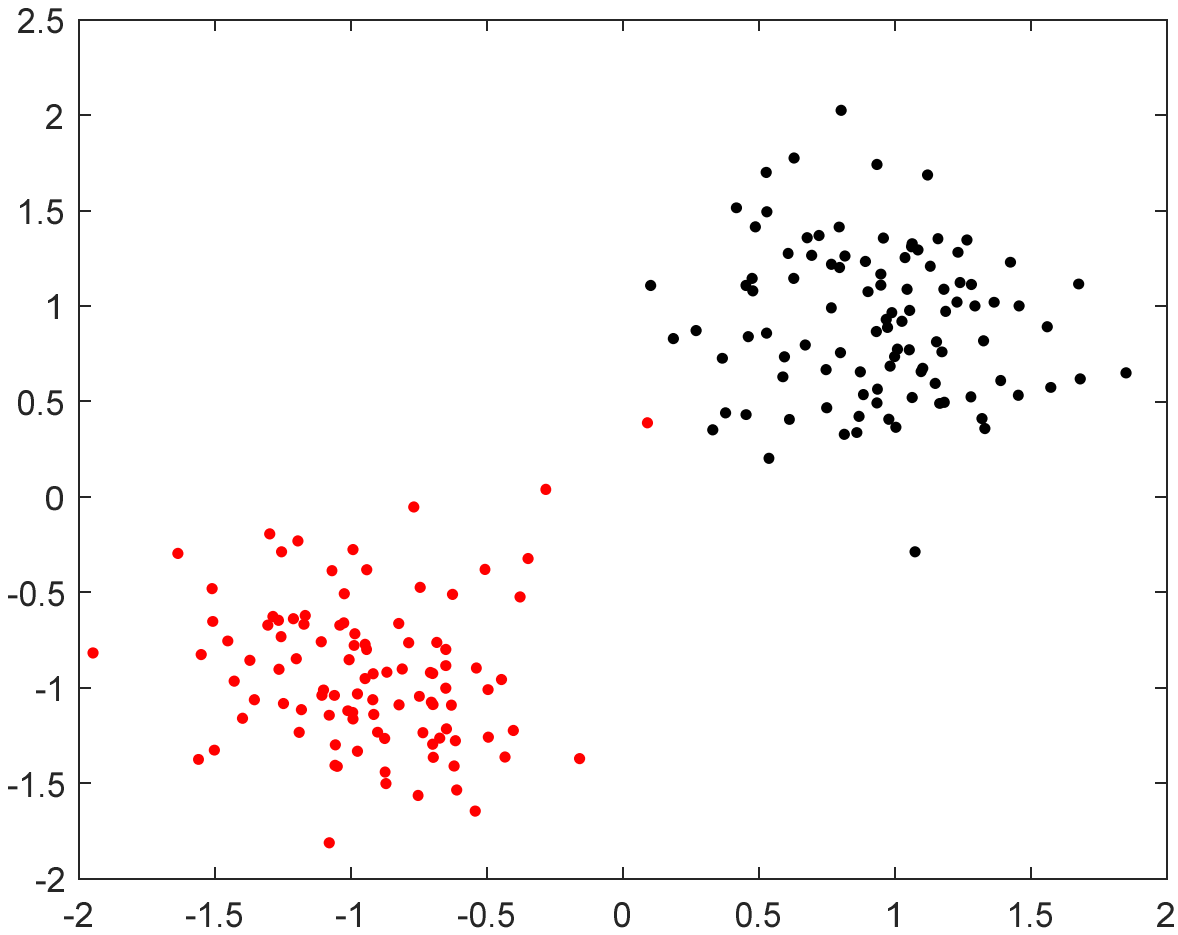
Where N_s is the total number of support vectors.

Example 1 of linearly separable samples




class 1
class 2
support vectors
separating hyperplane
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

Example 2 of linearly separable



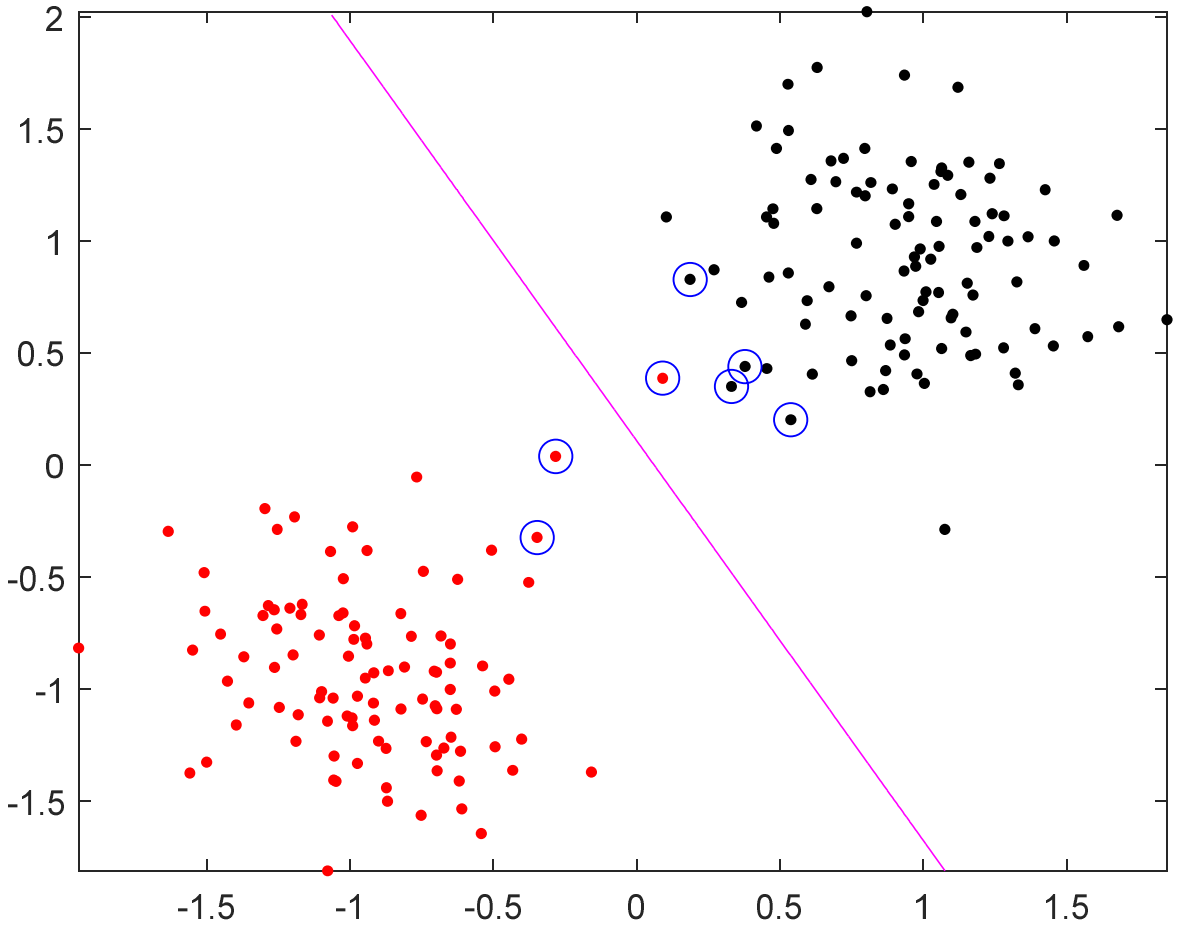
Discussion:

For this data, we have two hyperplanes:

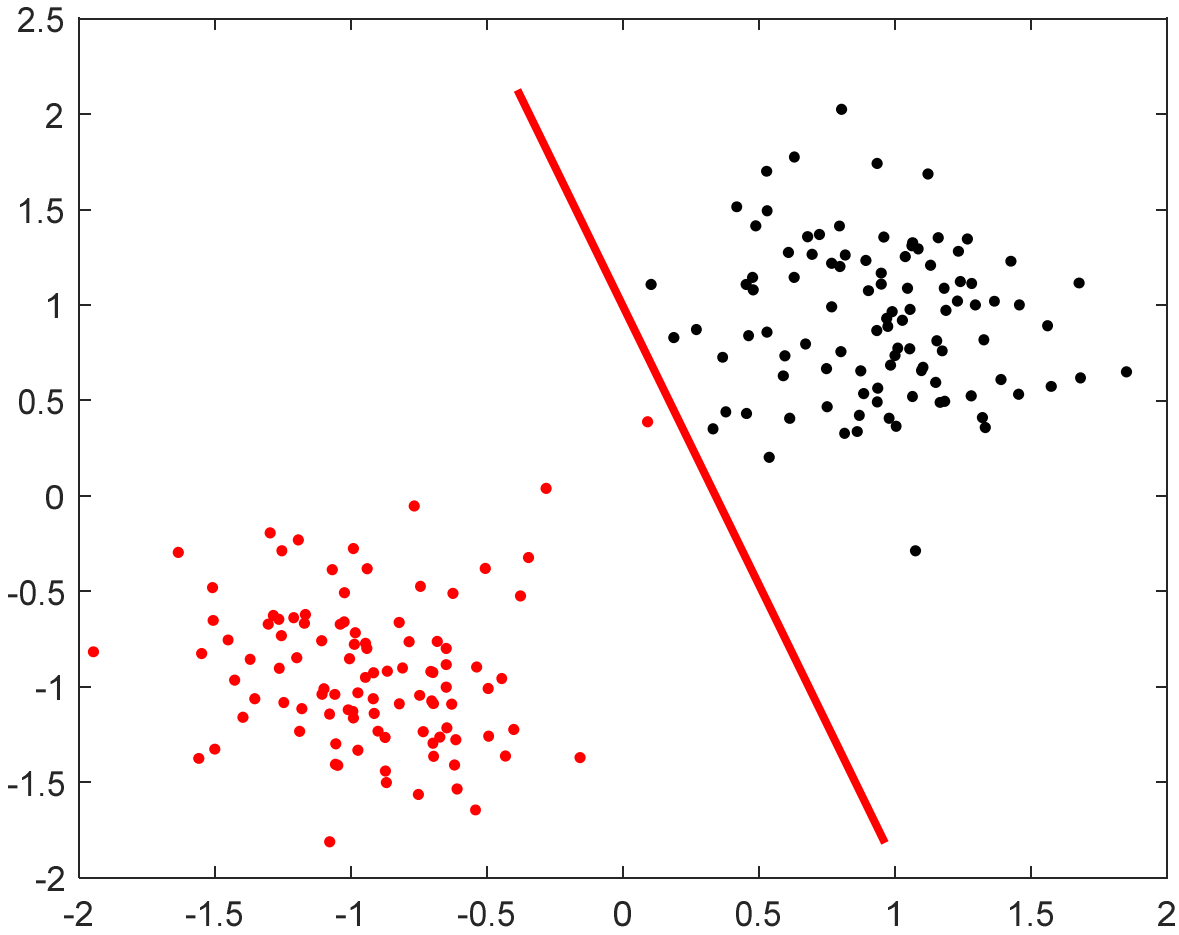
(1) Which one is preferred?

(2) Why?

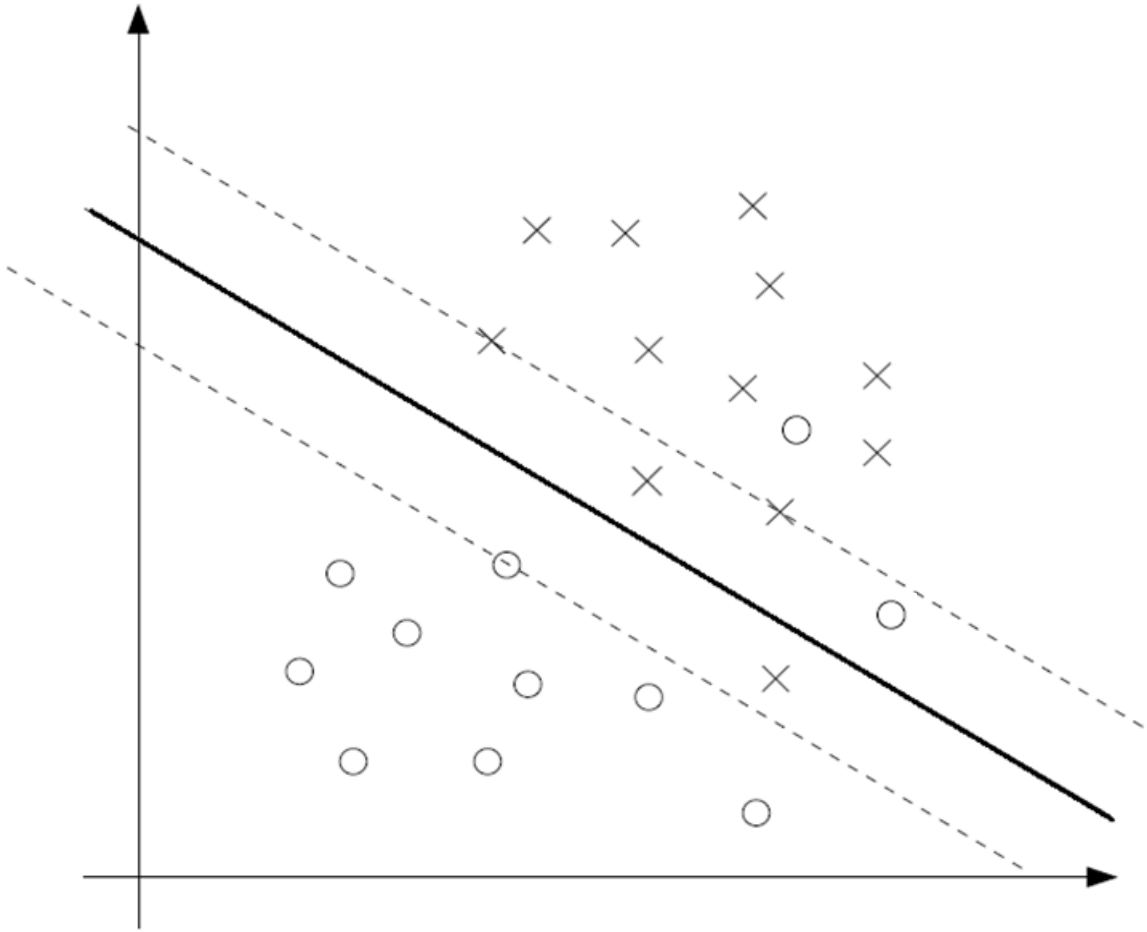
1 error for the training data.



0 error for training data



4.4 Optimal hyperplane for non-separable patterns



For this dataset, it is impossible to construct a separating hyperplane with zero classification error. Nevertheless, we would like to find an optimal hyperplane that minimizes the classification error.

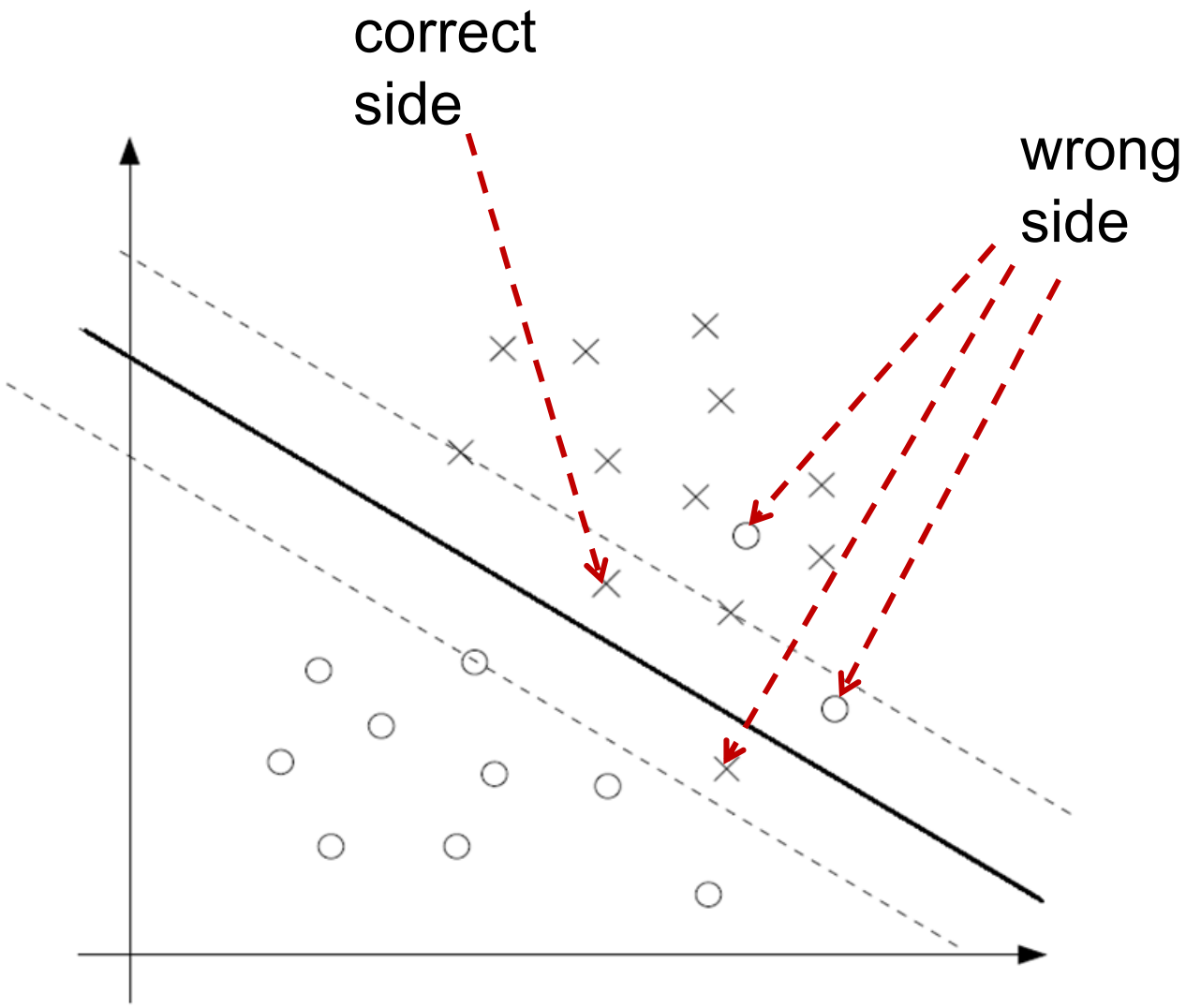
The margin of separation between classes is said to be soft if a data point $\{\mathbf{x}(i), d(i)\}$, $i = 1,2,...N$, violates the following conditions:

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

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.


correct
side
wrong
side

We introduce a new set of nonnegative scalar variables $\xi(i)$, $i = 1,2,\ldots,N$, 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.

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:

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

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:

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

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

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

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

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 **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 try to 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 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,

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

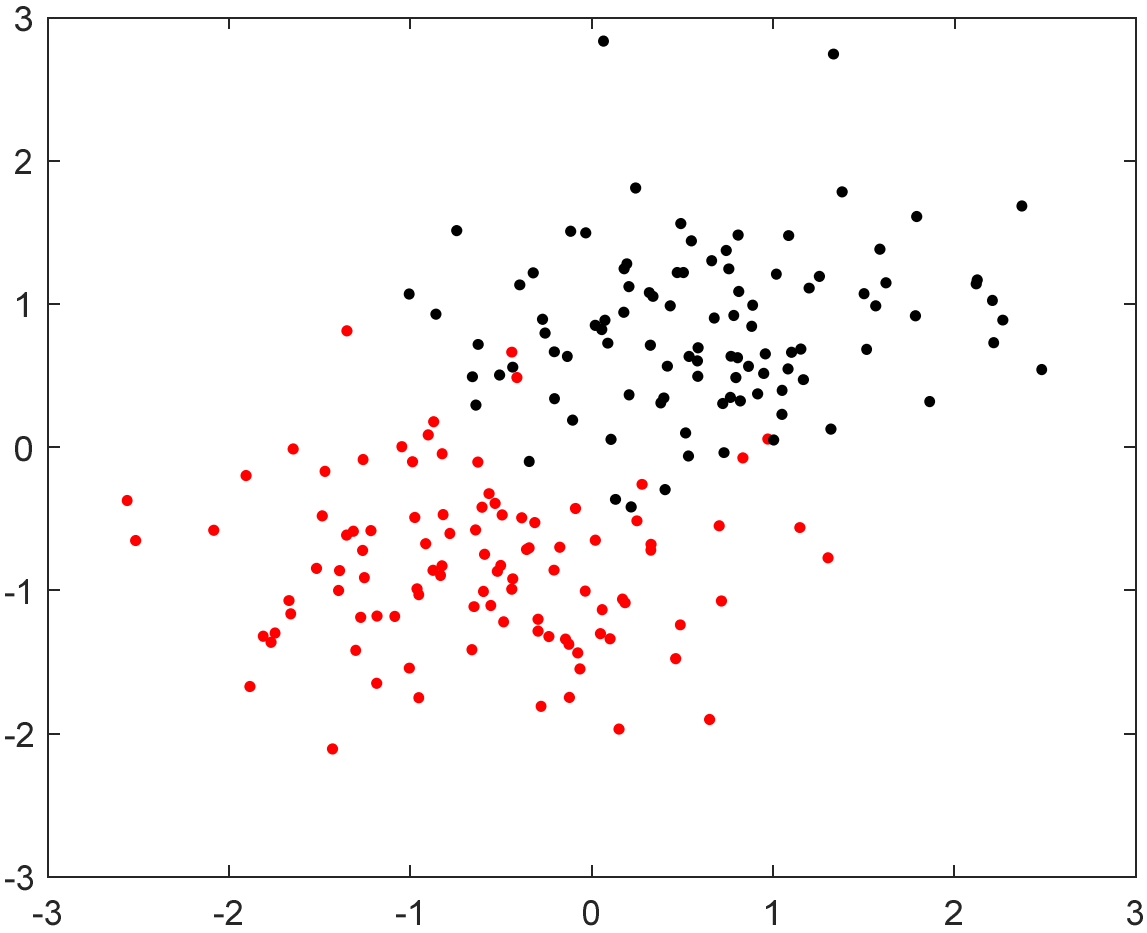
Note: $\mathbf{x}(i)$ is a support vector with $0 < \alpha(i) < C$.

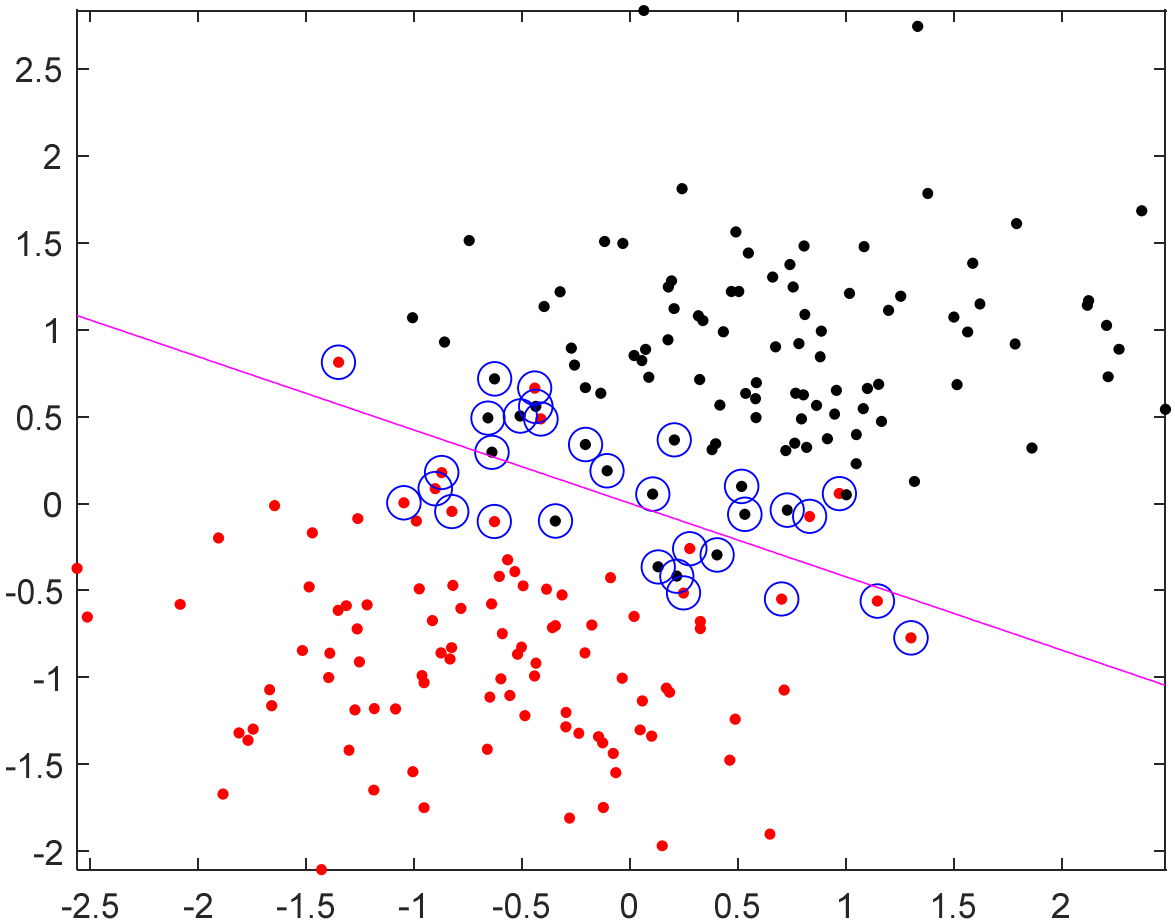
In practice, due to computation errors, we use each of the support vectors with $0 < \alpha(i) < C$ to calculate a value of the bias term and then take the average:

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

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

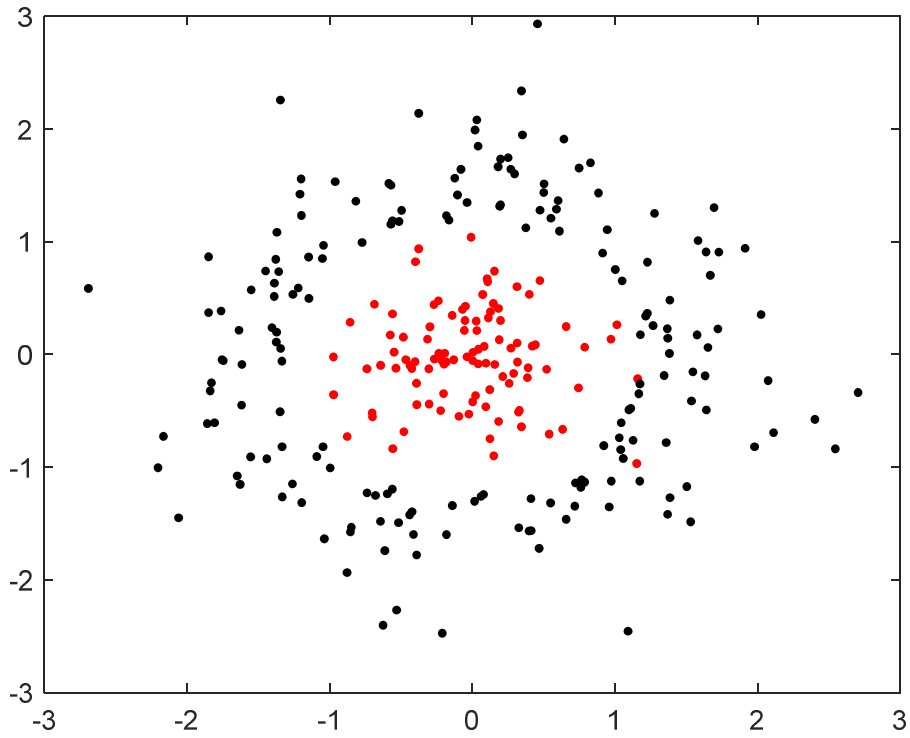
Example of linearly non-separable patterns



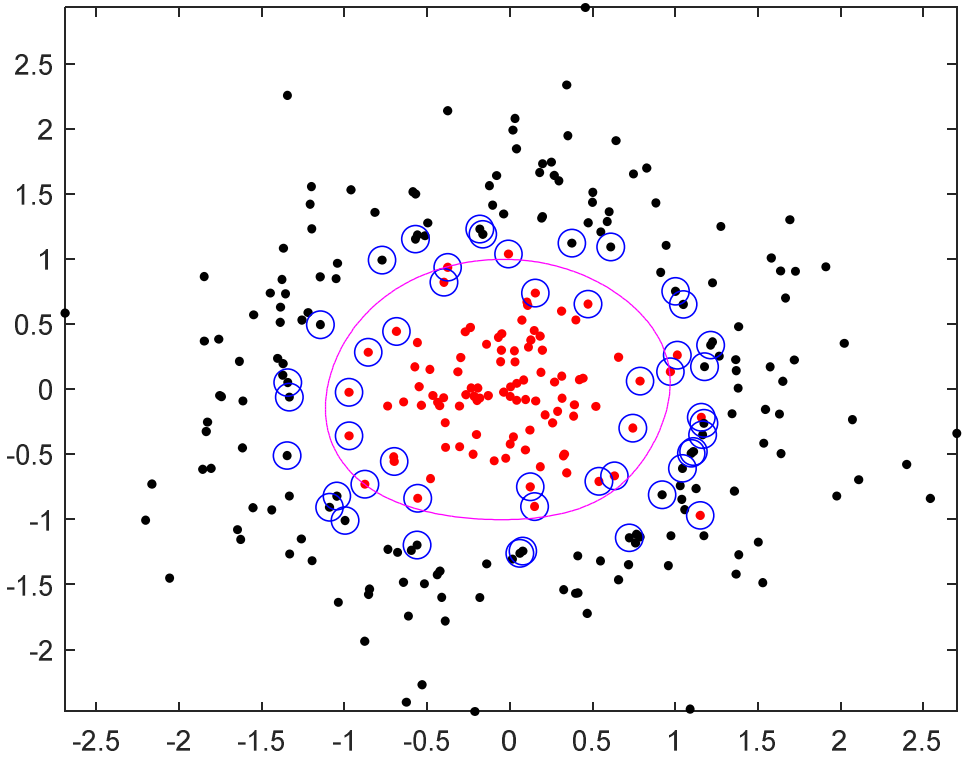

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

4.5 Kernel Support Vector Machines

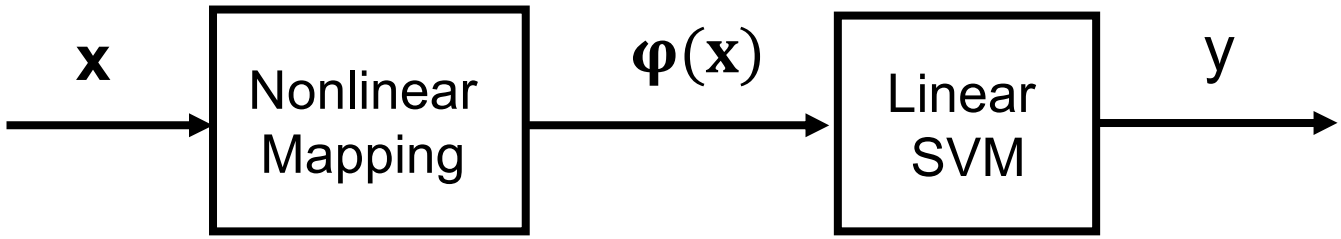
In the above, linear SVM is presented. If the data is linearly inseparable as shown below, how to construct a classifier to classify the data?



The solution is kernel support vector machines.



Kernel SVM can be interpreted as follows:

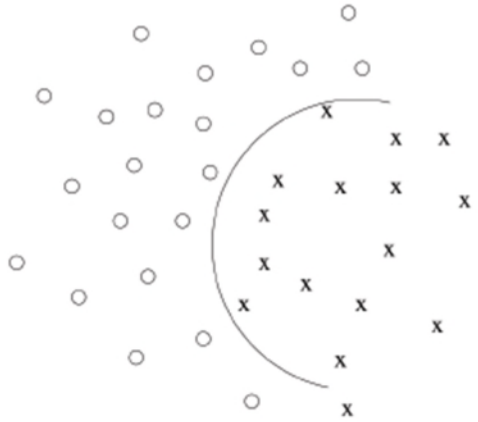


Step 1: Nonlinear mapping of an input vector into a high-dimensional feature space:

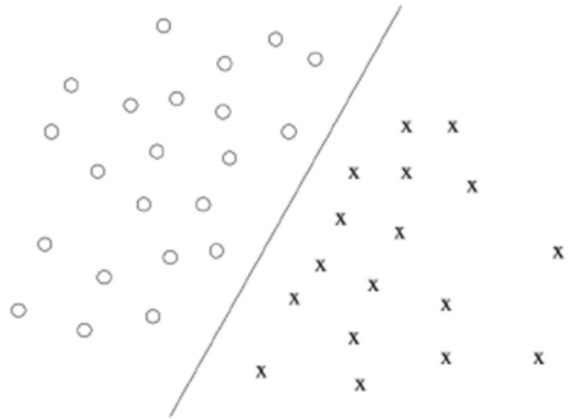
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

Step 2: Construction of an optimal hyperplane for separating the data in the high-dimensional feature space:

$$\boldsymbol{\varphi}(\mathbf{x}) \rightarrow y$$



Non-linear separator in the original $\mathbf{x}$-space



Linear separator in the feature ϕ-space

The first step operation is in accordance with *Cover's theorem* on the separability of patterns. Consider an input space made up of nonlinearly separable patterns. *Cover's theorem states that such a multi-dimensional space may be transformed into a new feature space where the patterns are linearly separable with high probability*, provided two conditions are satisfied:

(a) The transformation is nonlinear.
(b) The dimensionality of the feature space is high enough.

The second step operation exploits the ideas of building an optimal separating hyperplane linear SVM, but with a fundamental difference: the hyperplane is now defined as a linear function of vectors in the feature space rather than the original input space.

Let $\varphi_j(\mathbf{x}), j = 1,2,\ldots,m,$ denotes a set of nonlinear transformations from the input space to the feature space, where m is the dimension of the feature space. It is assumed that $\varphi_j(\mathbf{x})$ is defined a *priori* for all j. Given such a set of nonlinear transformations, we may define a hyperplane decision surface as follows:

$$\sum_{j=1}^{m} w_j \varphi_j(\mathbf{x}) + b = 0$$

The above hyperplane in the feature space can also be written as:

$$\sum_{j=0}^{m} w_j \varphi_j(\mathbf{x}) = 0$$

where w_0 is the bias term b, and $\varphi_0(\mathbf{x}) = 1$ for any $\mathbf{x}$.

Define the vector:

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}) \quad \varphi_1(\mathbf{x}) \quad \cdots \quad \varphi_m(\mathbf{x})]^T$$

Then the decision surface is:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0$$

Adapting the optimality condition of linear SVM to the present situation involving a feature space where we now seek linear separability of features, we may write:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha(i)\, d(i) \boldsymbol{\varphi}[\mathbf{x}(i)]$$

Thus, the decision surface in the feature space is:

$$\sum_{i=1}^{N} \alpha(i)\, d(i) \boldsymbol{\varphi}[\mathbf{x}(i)]^T \boldsymbol{\varphi}(\mathbf{x}) = 0$$

The term $\boldsymbol{\varphi}[\mathbf{x}(i)]^T\boldsymbol{\varphi}(\mathbf{x})$ represents the inner product of two vectors in the feature space. We introduce the inner-product kernel denoted by $K[\mathbf{x}, \mathbf{x}(\mathrm{i})]$ and defined by:

$$K[\mathbf{x}, \mathbf{x}(i)] = \boldsymbol{\varphi}(\mathbf{x})^T\boldsymbol{\varphi}[\mathbf{x}(i)]$$
$$= \sum_{j=0}^{m} \varphi_j(\mathbf{x})\varphi_j[\mathbf{x}(i)]$$

From this definition, we immediately see that the inner product kernel is a symmetric function of its arguments:

$$K[\mathbf{x}, \mathbf{x}(i)] = K[\mathbf{x}(i), \mathbf{x}]$$

Most importantly, we may use the inner-product kernel to construct the optimal hyperplane in the feature space without having to consider the feature space itself in explicit form:

$$\sum_{i=1}^{N} \alpha(i)\, d(i) K[\mathbf{x}(i), \mathbf{x}] = 0$$

Let's see an example $K[\mathbf{x}, \mathbf{z}] = (\mathbf{x}^T \mathbf{z})^2$, which can also be written as:

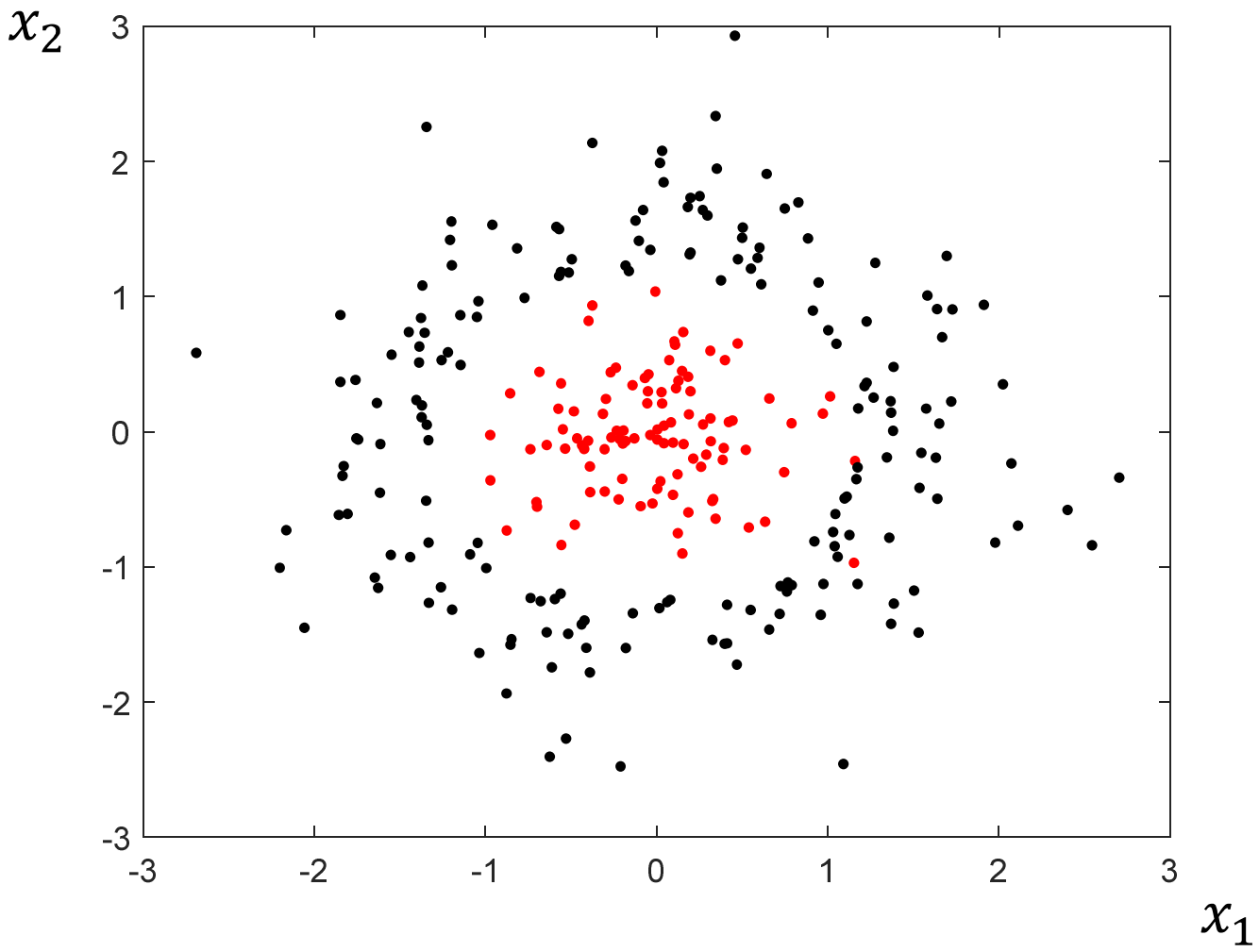
$$K[\mathbf{x}, \mathbf{z}] = \left(\sum_{i=1}^{2} x_i z_i \right) \left(\sum_{j=1}^{2} x_j z_j \right)$$

$$= \sum_{i=1}^{2} \sum_{j=1}^{2} x_i z_i x_j z_j = \sum_{i=1}^{2} \sum_{j=1}^{2} (x_i x_j)(z_i z_j)$$

If $\boldsymbol{\varphi}(\mathbf{x}) = [x_1^2 \quad x_1 x_2 \quad x_2 x_1 \quad x_2^2]^T$

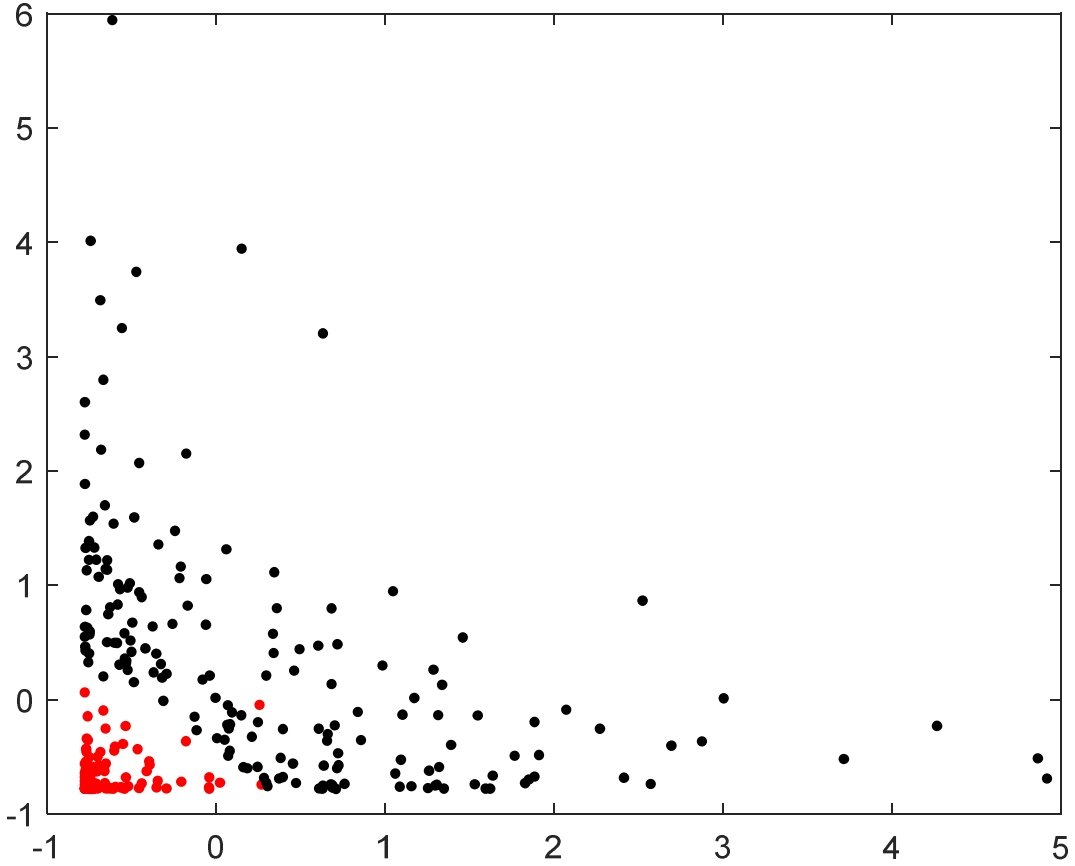
Then $K[\mathbf{x}, \mathbf{z}] = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{z})$

Samples in input space



Samples in feature space ($\boldsymbol{\varphi}$ space)

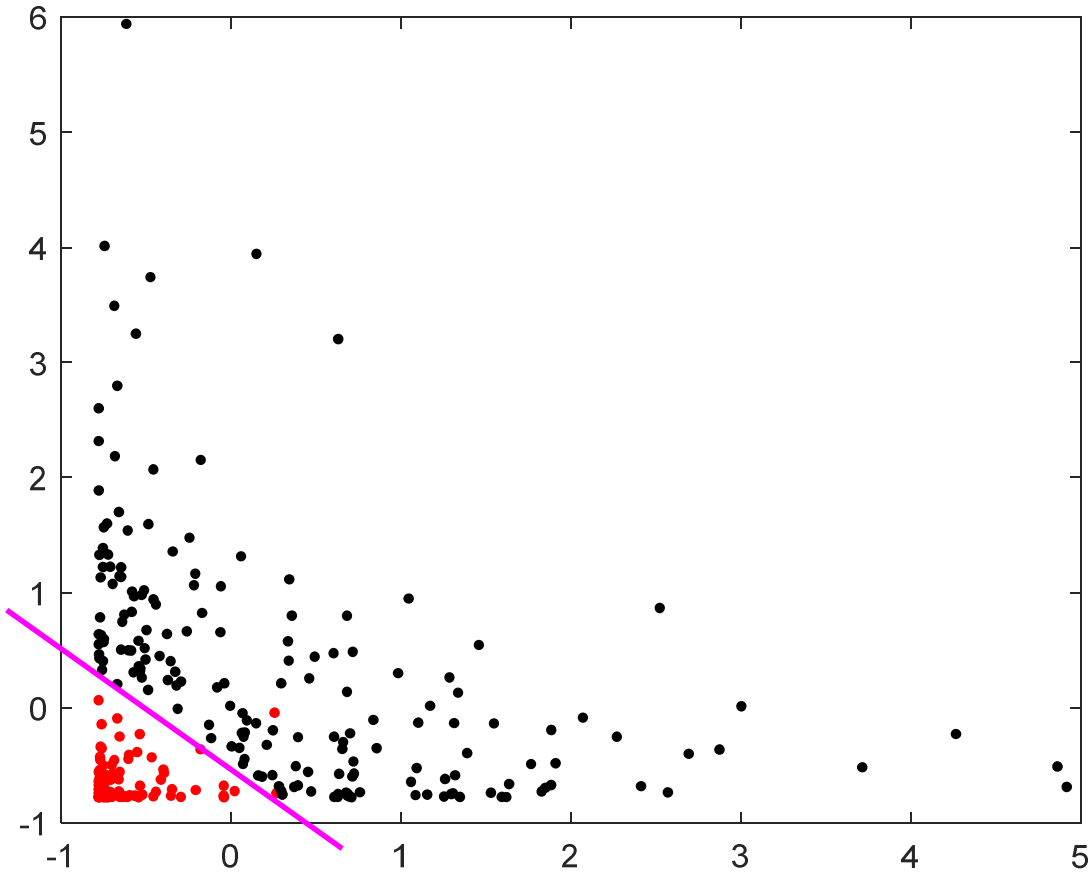
$$\varphi_2(\mathbf{x}) = x_2^2$$



$$\varphi_1(\mathbf{x}) = x_1^2$$

Linearly separable in feature space ($\boldsymbol{\varphi}$ space)

$$\varphi_2(\mathbf{x}) = x_2^2$$



$$\varphi_1(\mathbf{x}) = x_1^2$$

Now we introduce kernel matrix **K**, whose element at row i and column j is defined as:

$$k(i,j) = K[\mathbf{x}(i), \mathbf{x}(j)]$$

Obviously, kernel matrix **K** is symmetric. In addition, for any vector **z**, we have:

$$\begin{aligned}
\mathbf{z}^T\mathbf{K}\mathbf{z} &= \sum_i \sum_j z_i \, k(i,j) z_j \\
&= \sum_i \sum_j z_i \, \boldsymbol{\varphi}[\mathbf{x}(i)]^T \boldsymbol{\varphi}[\mathbf{x}(j)] z_j \\
&= \sum_i \sum_j z_i \sum_k \varphi_k[\mathbf{x}(i)] \varphi_k[\mathbf{x}(j)] \, z_j \\
&= \sum_k \sum_i \sum_j z_i \varphi_k[\mathbf{x}(i)] \varphi_k[\mathbf{x}(j)] \, z_j \\
&= \sum_k \left(\sum_i z_i \varphi_k[\mathbf{x}(i)] \right)^2 \geq 0
\end{aligned}$$

The above result means the kernel matrix **K** is positive semi-definite. To judge whether a *K* is a valid kernel, we have the following Mercer's Theorem:

Mercer's Theorem

K is a valid kernel, it is necessary and sufficient that for any samples $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)$, *the corresponding kernel matrix* ***K*** *is symmetric and positive semi-definite.*

For the example kernel discussed above:

$$K[\mathbf{x}, \mathbf{z}] = (\mathbf{x}^T \mathbf{z})^2$$

φ(**x**) has a computation complexity of $O(n^2)$, while K has a complexity of $O(n)$, which is linear in the dimension of the input space. The computational complexity reduction by using kernel is significant if the input space is of very high dimensional.

Kernel Functions in SVM

The requirement on kernel $K(\mathbf{x}, \mathbf{z})$ is to satisfy Mercer's Theorem. Within this requirement, there is some freedom in how it is chosen. Two inner-product kernels are often used:

- Polynomial kernel

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^p$$

where p is a positive integer, usually set to 2 or 3.

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right)$$

where σ is the width of the Gaussian kernel.

Design of a kernel support vector machine

The expansion of the inner-product kernel permits us to construct a decision surface that is nonlinear in the input space but is linear in the feature space. We may state the dual form for kernel support vector machine as follows:

Find the Lagrange multipliers that maximize the following objective function:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha(i) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha(i)\alpha(j)d(i)\,d(j)K(\mathbf{x}(i), \mathbf{x}(j))$$

Subject to conditions:

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

(2) $$0 \leq \alpha(i) \leq C$$

The dual problem of kernel SVM is of the same form as that of linear SVM for nonseparable patterns, except for the fact that the inner product $\mathbf{x}^T(i)\mathbf{x}(j)$ used in linear SVM is replaced by the inner-product kernel $K[\mathbf{x}(i), \mathbf{x}(j)]$.

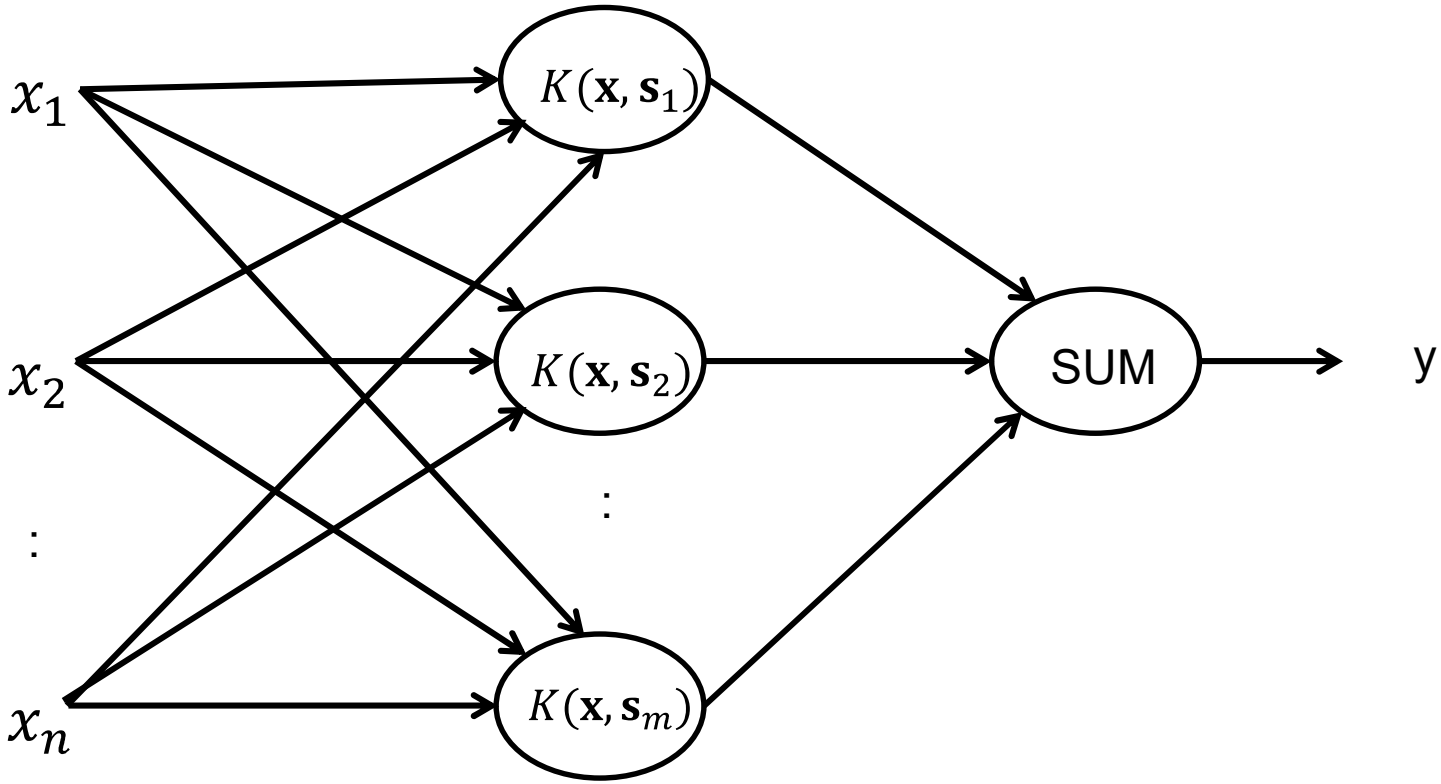
After having found the optimum values of the Lagrange multipliers, denoted by $\alpha(i)$, we may compute the decision value of a given input vector $\mathbf{x}$:

$$\begin{aligned} y &= \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= \sum_{i=1}^{N} \alpha(i) d(i) K[\mathbf{x}(i), \mathbf{x}] \end{aligned}$$

We do not compute the weight vector $\mathbf{w}$ explicitly because the feature space is hidden, and $\boldsymbol{\varphi}[\mathbf{x}(i)]$ is unavailable:

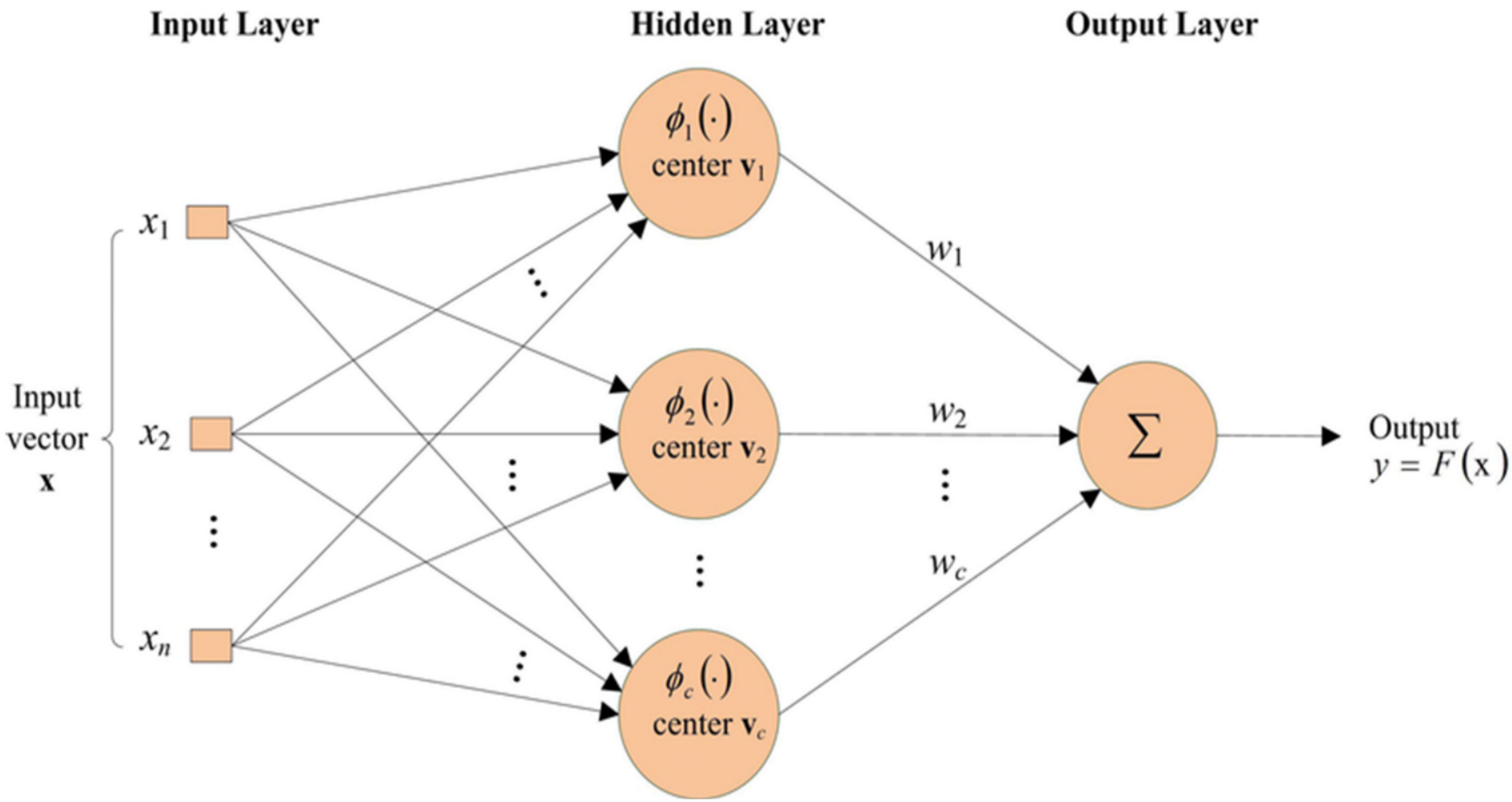
$$\mathbf{w} = \sum_{i=1}^{N} \alpha(i)\, d(i) \boldsymbol{\varphi}[\mathbf{x}(i)]$$

Architecture of a kernel support vector machine



$\mathbf{s}_i$ are the support vectors

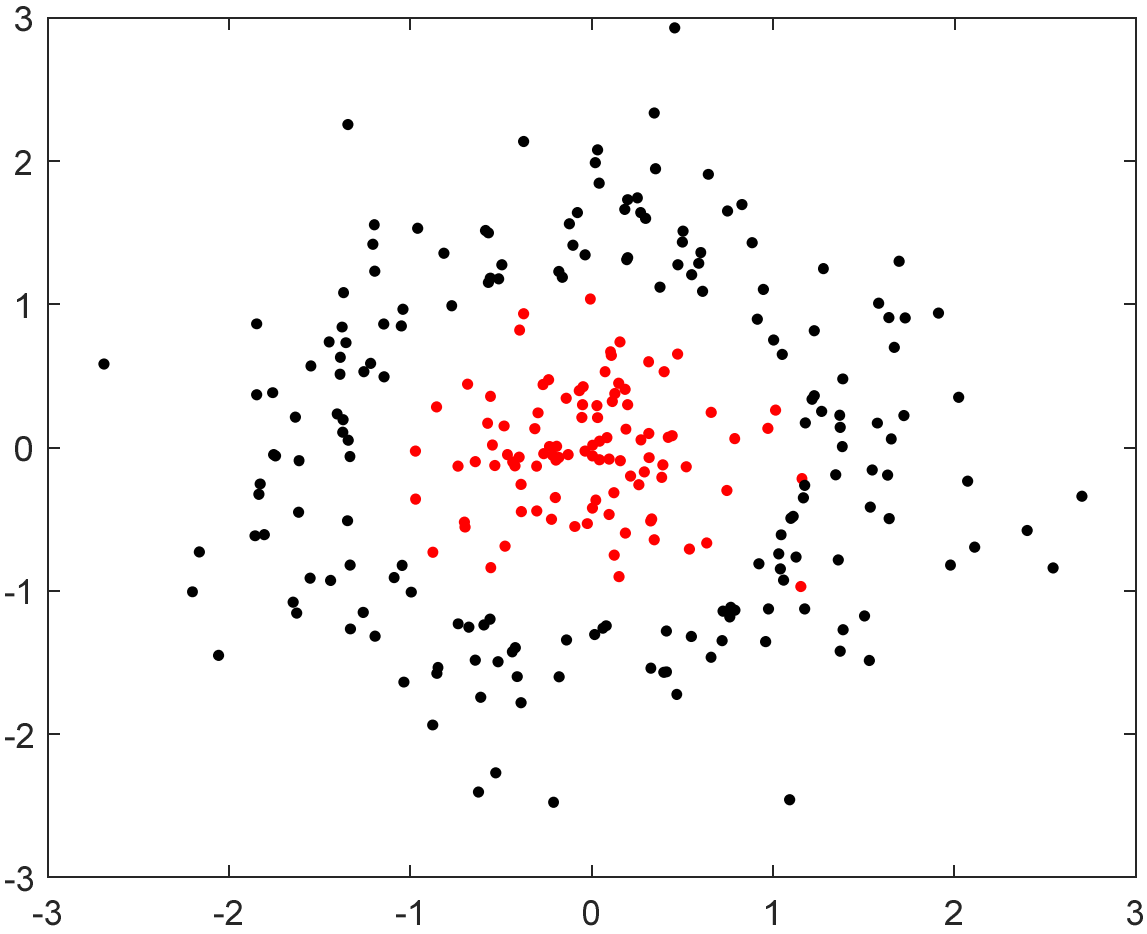
Architecture of RBF Neural Network



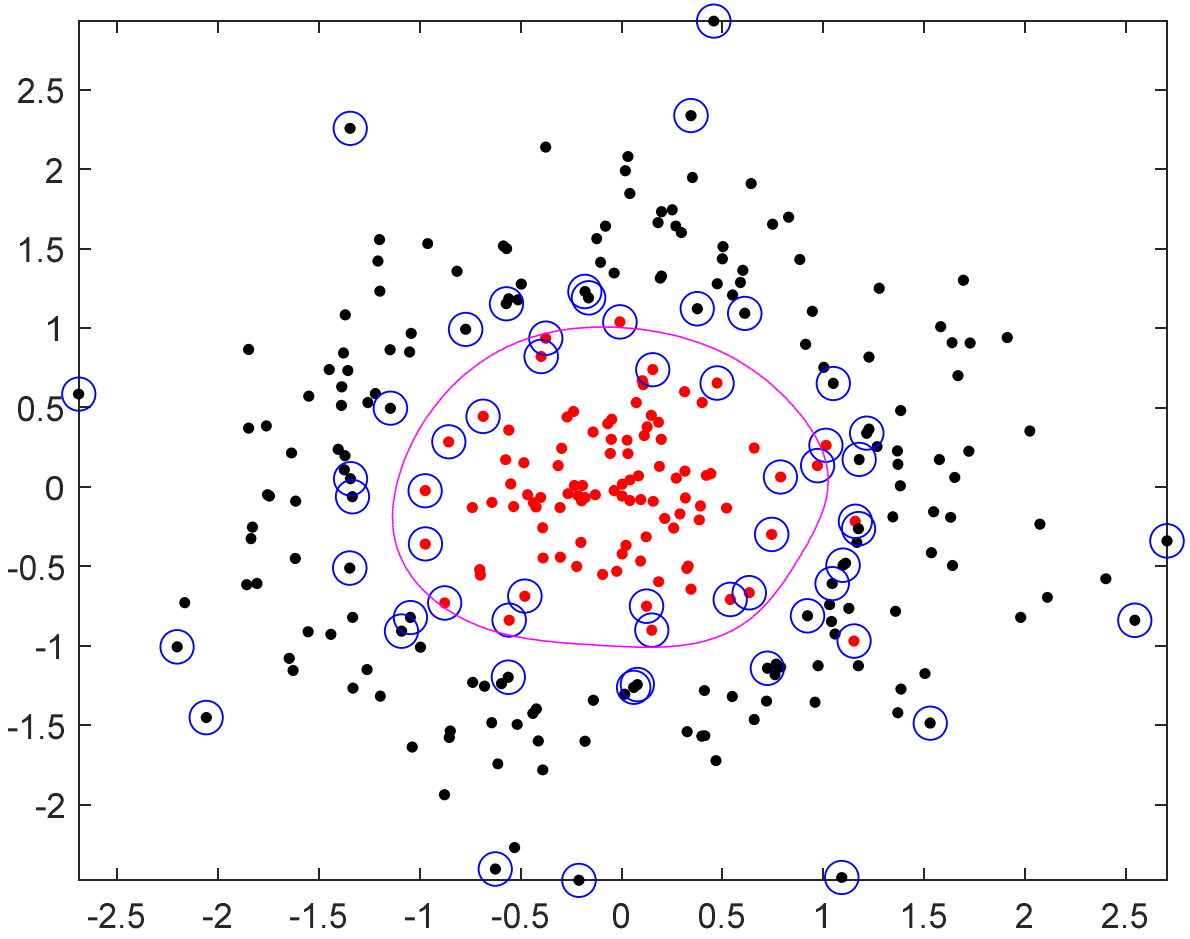
The following points are noteworthy:

- The kernel SVM has a similar architecture with RBF neural networks.
- In Gaussian kernel SVM, the number of the neurons and their center vectors (i.e. support vectors) are determined automatically, avoiding the need for heuristics often used in the design of RBF neural networks

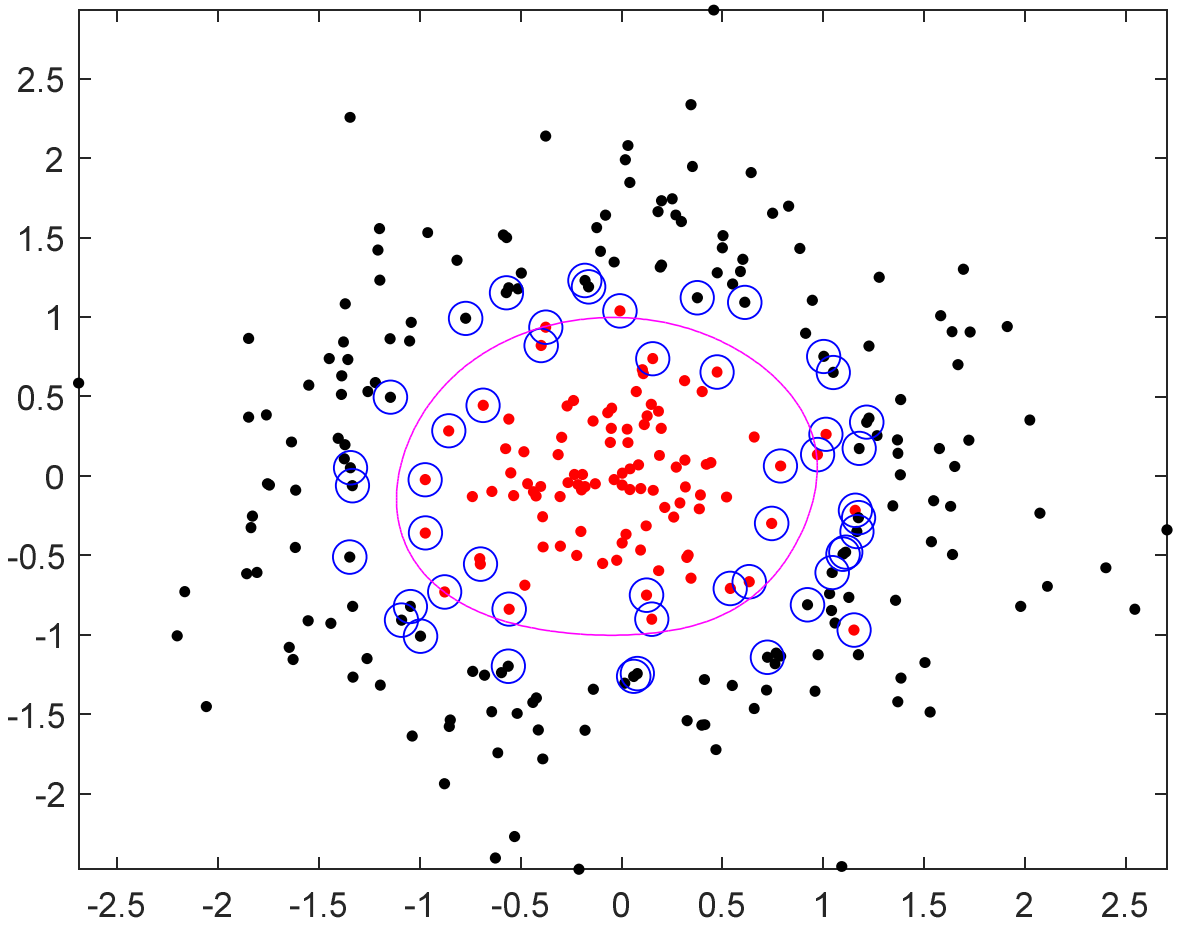
Example of kernel SVM



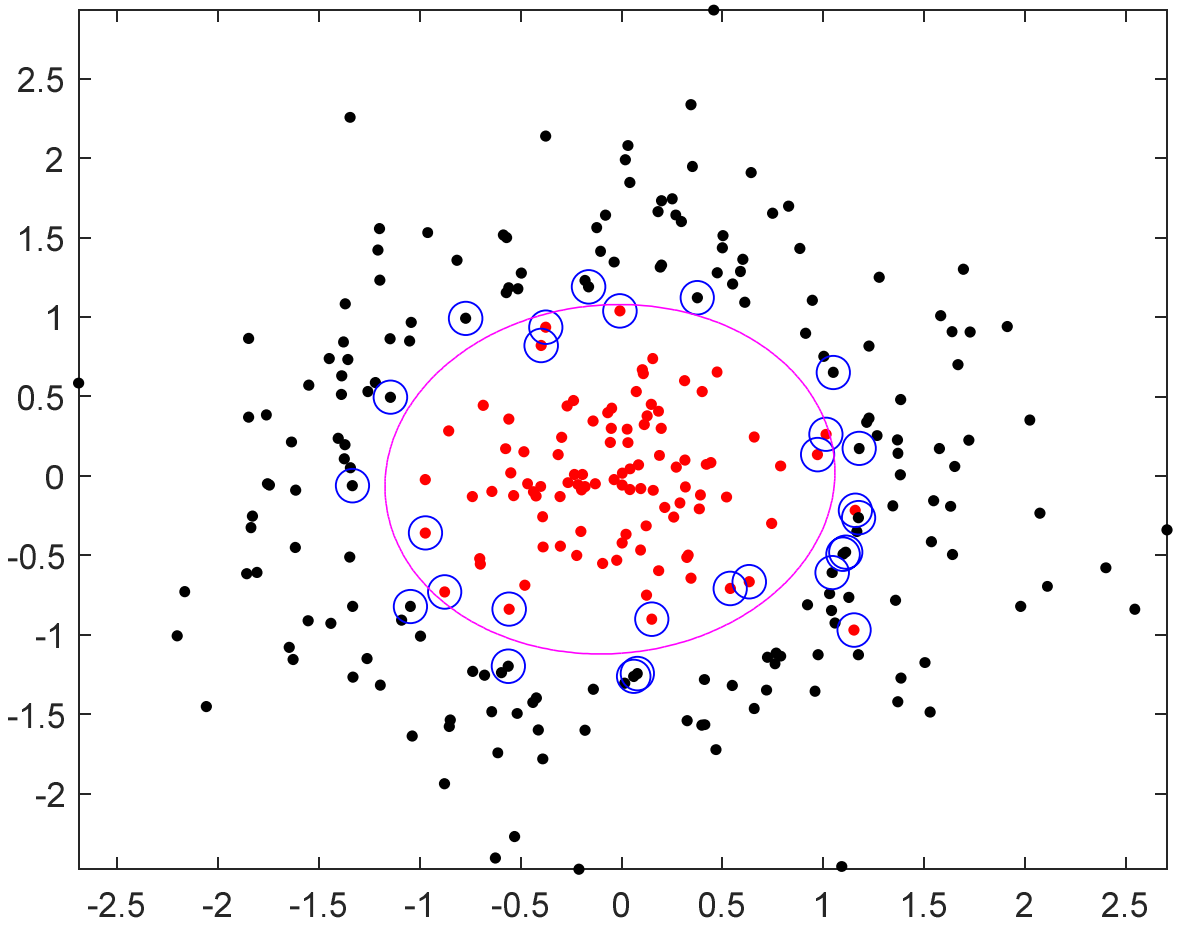
Gaussian kernel ($\sigma = 1$)



Gaussian kernel ($\sigma = 0.6767$)



Polynomial kernel ($p = 2$)



Polynomial kernel ($p = 3$)