Support Vector Machines

Introduction to Artificial Intelligence with Mathematics
Lecture Notes

Ganguk Hwang

Department of Mathematical Sciences
KAIST

Linear SVM

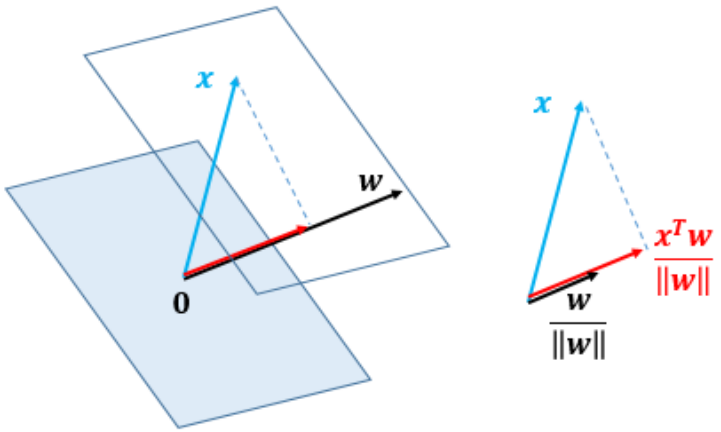
A hyperplane in an n-dimensional feature space can be represented by the following equation:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \sum_{i=1}^{n} x_i w_i + b = 0$$

Dividing by $||\mathbf{w}||$, we get

$$\frac{\mathbf{x}^T\mathbf{w}}{||\mathbf{w}||} = P_{\mathbf{w}}(\mathbf{x}) = -\frac{b}{||\mathbf{w}||}$$

indicating that the projection of any point $\mathbf{x}$ on the plane onto the vector $\mathbf{w}$ always has the length $|b|/||\mathbf{w}||$. Moreover, we see that $\mathbf{w}$ is the normal direction of the plane.



Note that the equation of the hyperplane is not unique. For instance, $c \cdot f(\mathbf{x}) = 0$ represents the same plane for any c.

The n-dimensional space is partitioned into two regions by the plane. Specifically, we define a mapping function $y = sign(f(\mathbf{x})) \in \{1, -1\}$,

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \begin{cases} > 0, & y = sign(f(\mathbf{x})) = 1,\ \mathbf{x} \in P \\ < 0, & y = sign(f(\mathbf{x})) = -1,\ \mathbf{x} \in N \end{cases}$$

A point $\mathbf{x}$ of unknown class will be classified to P if $f(\mathbf{x}) > 0$, or N if $f(\mathbf{x}) < 0$.

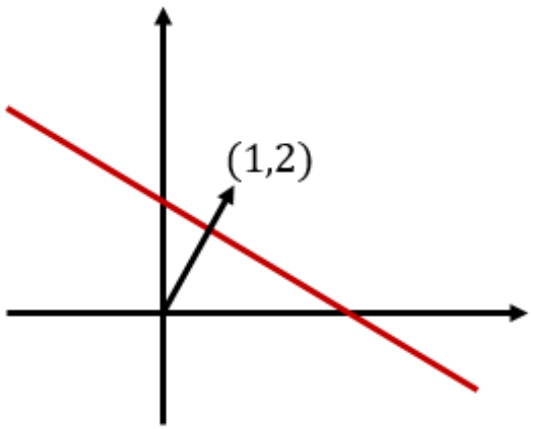
Example:

A straight line in a 2-dimensional space $\mathbf{x} = [x_1, x_2]^T$ described by the following equation:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}+b = [x_1, x_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}+b = [x_1, x_2]\begin{bmatrix} 1 \\ 2 \end{bmatrix}-1 = x_1+2x_2-1 = 0$$

divides the 2-dimensional plane into two halves. The distance between the origin and the line is

$$\frac{|b|}{||\mathbf{w}||} = \frac{1}{\sqrt{w_1^2 + w_2^2}} = \frac{1}{\sqrt{5}} = 0.447$$



Given a set of K training samples from two linearly separable classes P and N:

$$\{(\mathbf{x}_k, y_k), k = 1, \cdots, K\}$$

where $y_k \in \{1, -1\}$ labels $\mathbf{x}_k$ to belong to either of the two classes.

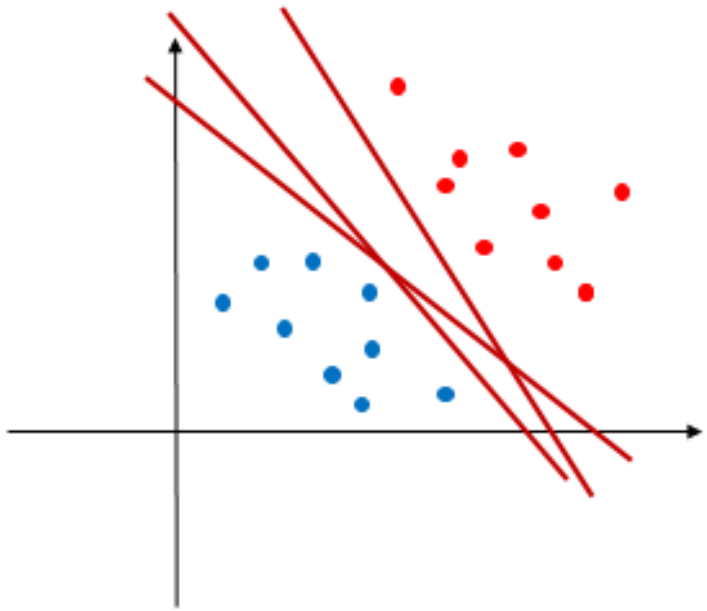
We want to find a hyperplane in terms of $\mathbf{w}$ and b, that linearly separates the two classes.

Before the classifier is properly trained, the actual output $\hat{y} = sign(f(\mathbf{x}))$ may not be the same as the desired output y. There are four possible cases:

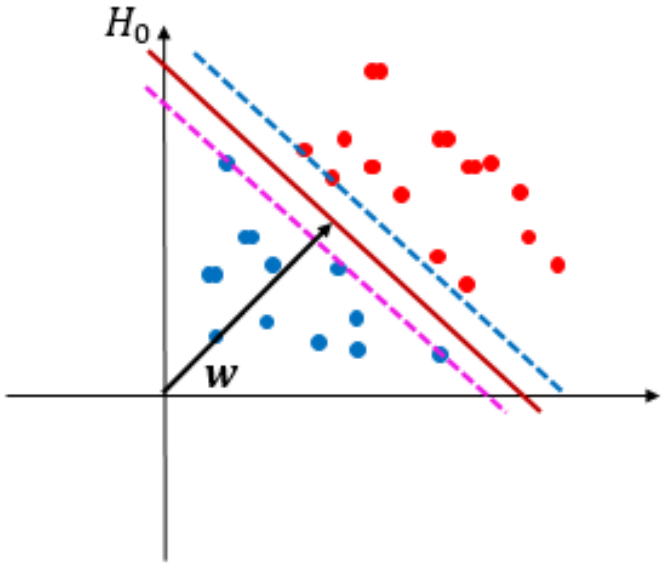
	Input $(\mathbf{x}, y)$	Output $\hat{y} = sign(f(\mathbf{x}))$	result
1	$(\mathbf{x}, y = 1)$	$\hat{y} = 1 = y$	corrrect
2	$(\mathbf{x}, y = -1)$	$\hat{y} = 1 \neq y$	incorrect
3	$(\mathbf{x}, y = 1)$	$\hat{y} = -1 \neq y$	incorrect
4	$(\mathbf{x}, y = -1)$	$\hat{y} = -1 = y$	corrrect

Note that for a correct decision hyperplane it satisfies that

$$y_i(\mathbf{x}_i^T \mathbf{w} + b) \geq 0 \text{ for all } i = 1, 2, \cdots, K.$$



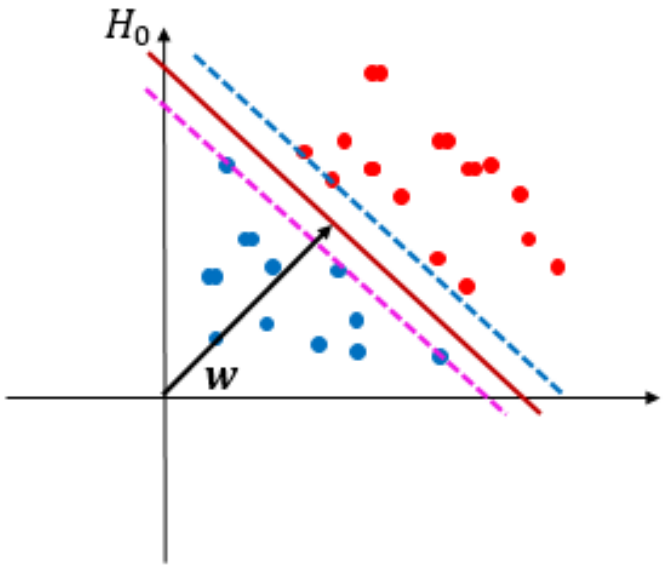
Among all correct decision hyperplanes we want to find the optimal one H_0 that separates the two classes with the maximal margin (the distance between the decision plane and the closest sample points).



Note that the optimal hyperplane should be in the middle of the two classes, so that the distance from the hyperplane to the closest point on either side is the same.

We define two additional hyperplanes H_+ and H_- that are parallel to H_0 and go through the point closest to the hyperplane H_0 on either side. Moreover,

$$\mathbf{x}^T\mathbf{w} + b = 1, \quad \text{and} \quad \mathbf{x}^T\mathbf{w} + b = -1$$



All points $\mathbf{x}_i \in P$ on the positive side should satisfy

$$\mathbf{x}_i^T \mathbf{w} + b \geq 1, \quad y_i = 1$$

and all points $\mathbf{x}_i \in N$ on the negative side should satisfy

$$\mathbf{x}_i^T \mathbf{w} + b \leq -1, \quad y_i = -1$$

These can be combined into one inequality:

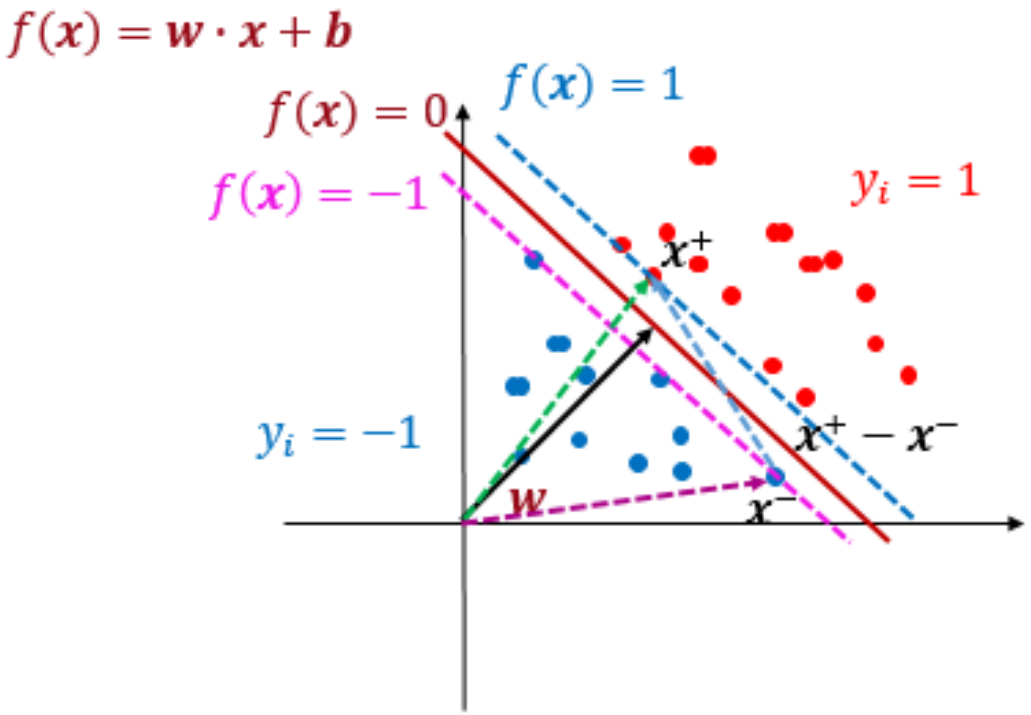
$$y_i(\mathbf{x}_i^T \mathbf{w} + b) \geq 1, \quad (i = 1, \cdots, K)$$

The equality holds for those points on the hyperplane H_+ or H_-. Such points are called *support vectors*, for which

$$\mathbf{x}_i^T \mathbf{w} + b = y_i.$$

Recall that the distances from the origin to the three parallel hyperplanes H_+, H_0 and H_- are $\frac{|b-1|}{||\mathbf{w}||}$, $\frac{|b|}{||\mathbf{w}||}$, and $\frac{|b+1|}{||\mathbf{w}||}$, respectively.

Moreover, the distance between hyperplanes H_- and H_+ is $\frac{2}{||\mathbf{w}||}$.



Our goal is to maximize this distance, equivalently, to minimize the norm $||\mathbf{w}||$ from which we have the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1 \ \ (\text{or} \ \ 1 - y_i(\mathbf{x}_i^T\mathbf{w} + b) \leq 0), \quad i = 1, \cdots, K. \end{aligned}$$

Our next step is to convert into an unconstrained optimization problem by using a penalty term as follows:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \text{ penalty term}$$

where the penalty term is given by

$$\max_{\alpha_i \geq 0} \alpha_i(1 - y_i(\mathbf{x}_i^T\mathbf{w} + b)) = \begin{cases} 0 & \text{if } y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1, \\ \infty & \text{otherwise.} \end{cases}$$

We now have the following unconstrained optimization problem:

$$\min_{\mathbf{w},b}\left(\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{i=1}^{K}\max_{\alpha_i\geq 0}\alpha_i(1-y_i(\mathbf{x}_i^T\mathbf{w}+b))\right)$$

$$=\min_{\mathbf{w},b}\max_{\alpha_i\geq 0}\left(\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{i=1}^{K}\alpha_i(1-y_i(\mathbf{x}_i^T\mathbf{w}+b))\right)$$

$$\geq\max_{\alpha_i\geq 0}\min_{\mathbf{w},b}\left(\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{i=1}^{K}\alpha_i(1-y_i(\mathbf{x}_i^T\mathbf{w}+b))\right)$$

Here, α_i are called Lagrange multipliers and the equality holds in certain conditions.

Let

$$L_p(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{K} \alpha_i(1 - y_i(\mathbf{x}_i^T\mathbf{w} + b)).$$

Solving the minimization problem first by using

$$\nabla_{\mathbf{w}} L_p(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial b} L_p(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

we get

$$\mathbf{w} = \sum_{i=1}^{K} \alpha_i y_i \mathbf{x}_i, \quad \text{and} \quad \sum_{i=1}^{K} \alpha_i y_i = 0.$$

Substituting these two equations back into the expression of $L_P(\mathbf{w}, b, \boldsymbol{\alpha})$, we get the following maximation problem with respect to α_i:

$$\max_{\alpha_i} \left(\sum_{i=1}^{K} \alpha_i - \frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{K} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

$$\text{s.t. } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{K} \alpha_i y_i = 0.$$

c.f. quadratic programming

For the solution $\alpha_i, i = 1, 2, \cdots, K$ of the above problem, we can get the optimal $\mathbf{w}$ by

$$\mathbf{w} = \sum_{i=1}^{K} \alpha_i y_i \mathbf{x}_i$$

To get the constant b, observe that, for a support vector $\mathbf{x}_i$, it satisfies

$$y_i \left(\mathbf{x}_i^T \mathbf{w} + b \right) = 1.$$

Multiplying y_i on both sides yields

$$y_i^2 \left(\mathbf{x}_i^T \mathbf{w} + b \right) = y_i.$$

From the fact that $y_i^2 = 1$, we get

$$\mathbf{x}_i^T \mathbf{w} + b = y_i,$$

i.e.,

$$b = y_i - \mathbf{x}_i^T \mathbf{w}.$$

- Recall that support vectors are those points $\mathbf{x}_i$ on either of the two hyperplanes H_+ and H_- (for which the equality $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ holds).
- We see that $\alpha_i = 0$ for non-support vectors $\mathbf{x}_i$.
- So the training depends only on the support vectors, while all other samples away from the planes H_+ and H_- are not important.

For a support vector $\mathbf{x}_i$ (on the H_- or H_+ plane), it satisfies that

$$y_i \left(\mathbf{x}_i^T \mathbf{w} + b \right) = 1 \quad (i \in I)$$

where I denotes a set of all indices of support vectors $\mathbf{x}_i$.

Substituting

$$\mathbf{w} = \sum_{j=1}^{K} \alpha_j y_j \mathbf{x}_j = \sum_{j \in I} \alpha_j y_j \mathbf{x}_j,$$

we get

$$y_i (\sum_{j \in I} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j + b) = 1$$

That is,

$$y_i \sum_{j \in I} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j = 1 - y_i b.$$

For the optimal weight vector $\mathbf{w}$ and the optimal parameter b, we have:

$$\begin{aligned} ||\mathbf{w}||^2 &= \mathbf{w}^T \mathbf{w} = \sum_{i \in I} \alpha_i y_i \mathbf{x}_i^T \sum_{j \in I} \alpha_j y_j \mathbf{x}_j = \sum_{i \in I} \alpha_i y_i \sum_{j \in I} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j \\ &= \sum_{i \in I} \alpha_i (1 - y_i b) = \sum_{i \in I} \alpha_i - b \sum_{i \in I} \alpha_i y_i \\ &= \sum_{i \in I} \alpha_i \end{aligned}$$

where we use the fact that $\sum_{i \in I} \alpha_i y_i = \sum_{i=1}^{K} \alpha_i y_i = 0$.

Recall that the distance between the two margin planes H_+ and H_- is $2/||\mathbf{w}||$ and

$$\frac{2}{||\mathbf{w}||} = \frac{2}{\sqrt{\sum_{i \in I} \alpha_i}}.$$

Moreover, the margin, the distance between H_+ (or H_-) and the optimal decision hyperplane H_0, is

$$\frac{1}{||\mathbf{w}||} = \frac{1}{\sqrt{\sum_{i \in I} \alpha_i}}$$

Comparison between Logistic Regression and Support Vector Machine

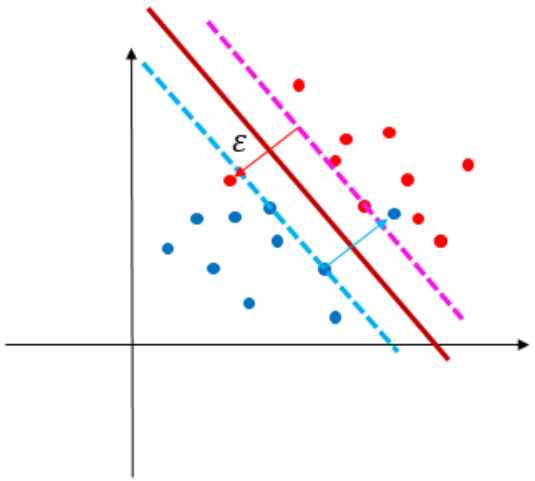
- Logistic regression takes the decision hyperplane that maiximizes the probability of data (we use the likelihood function). So the farther the data lies from the decision hyperplane (on the correct side), the happier the logistic regression is.
- Support vector machine takes the decision hyperplane that maximizes the distance of the closest points (the margine of the support vectors) from the hyperplane. If a point is not a support vector, it doesn't really matter.

Soft Margin SVM

Soft Margin SVM

When the two classes are not linearly separable (e.g., due to noise), the condition for the optimal hyperplane can be relaxed by including an extra term:

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1 - \xi_i, \quad (i = 1, \cdots, K)$$



For minimum error, $\xi_i \geq 0$ should be minimized as well as $||\mathbf{w}||$, and the objective function becomes:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{K} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i, \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, K) \end{aligned}$$

Here C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error.

A small value of C tends to emphasize the margin while ignoring the outliers in the training data, while a large value of C may tend to overfit the training data.

Using Lagrange mutipliers α_i and $\mu_i, 1 \leq i \leq n$, we have the following unconstraned optimization problem:

$$\min L_p(\mathbf{w}, b, \xi_i, \alpha_i, \mu_i)$$

$$\text{s.t. } \alpha_i \geq 0, \mu_i \geq 0, \qquad 1 \leq i \leq n$$

where

$$L_p(\mathbf{w}, b, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i \xi_i$$

Observe that

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \text{yields} \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \text{yields} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = 0 \quad \text{yields} \quad C - \alpha_i - \mu_i = 0$$

Using the above results we have the following optimization problem.

$$\max_{\alpha_i} L_D(\alpha_i) = \max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

for $0 \leq \alpha_i \leq C$.

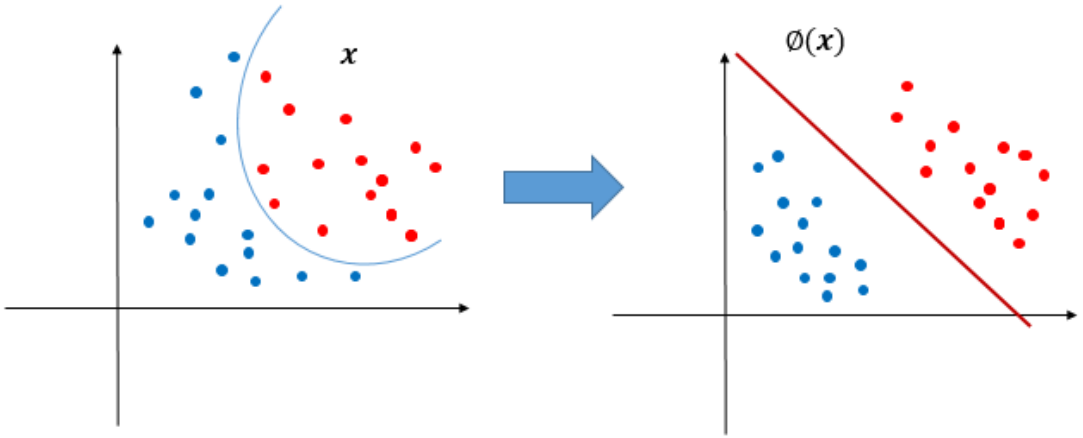
Kernel Trick

Kernel Mapping

The algorithm above converges only for linearly separable data. If the data set is not linearly separable, we can map the samples $\mathbf{x}$ into a feature space of higher dimensions:

$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

in which the classes can be linearly separated.



The decision function in the new space becomes:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} + b = \sum_{i=1}^{K} \alpha_j y_j (\phi(\mathbf{x})^T \phi(\mathbf{x}_j)) + b$$

where

$$\mathbf{w} = \sum_{j=1}^{K} \alpha_j y_j \phi(\mathbf{x}_j)$$

and b are the parameters of the decision plane in the new space.

- As the vectors $\mathbf{x}_i$ appear only in inner products in both the decision function and the learning law, the mapping function $\phi(\mathbf{x})$ does not need to be explicitly specified.
- Instead, all we need is the inner product of the vectors $\phi(\mathbf{x})$ in the new space.
- The function $\phi(\mathbf{x})$ is a kernel-induced *implicit* mapping.

Definition: A kernel is a function that takes two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ as arguments and returns the value of the inner product of their images $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$$

- As only the inner product of the two vectors in the new space is returned, the dimensionality of the new space is not important.

The learning algorithm in the kernel space can be obtained by replacing all inner products in the learning algorithm in the original space with the kernels:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} + b = \sum_{j=1}^{K} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b$$

The parameter b can be found from any support vectors $\mathbf{x}_i$:

$$\begin{aligned} b &= y_i - \phi(\mathbf{x}_i)^T \mathbf{w} \\ &= y_i - \sum_{j=1}^{K} \alpha_j y_j (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)) \\ &= y_i - \sum_{j=1}^{K} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

Example 1. linear kernel
Assume $\mathbf{x} = [x_1, \cdots, x_n]^T$, $\mathbf{z} = [z_1, \cdots, z_n]^T$,

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z} = \sum_{i=1}^{n} x_i z_i$$

Example 2. polynomial kernels
Assume $\mathbf{x} = [x_1, x_2]^T$, $\mathbf{z} = [z_1, z_2]^T$,

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2 &= (x_1 z_1 + x_2 z_2)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 \\ &= < (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) >= \phi(\mathbf{x})^T \phi(\mathbf{z}) \end{aligned}$$

This is a mapping from a 2-dimensional space to a 3-dimensional space.

Example 3.

$$K(\mathbf{x}, \mathbf{z}) = e^{-||\mathbf{x}-\mathbf{z}||^2/2\sigma^2}$$

Example 4.

$$K_2(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_1(\mathbf{x}, \mathbf{x})^{-1/2}K_1(\mathbf{z}, \mathbf{z})^{-1/2}$$

Note that Kernels measure **similarity**.

Similarly as in the linear SVM, we solve the following maximization problem:

$$\max_{\alpha_i \geq 0} \left(\sum_{i=1}^{K} \alpha_i - \frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{K} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$

Unlike the linear SVM, we cannot express $\mathbf{w}$ as a linear combination of support vectors in this case.

Even though kernels allow very flexible models, we have to determine the kernel parameters which is one of drawbacks.

Example 1. Hard margin support vector machine - linearly seperable case

Consider a classification problem in a 2D space, which is **linearly separable**. Here, the goal is to find a **linear** decision boundary.

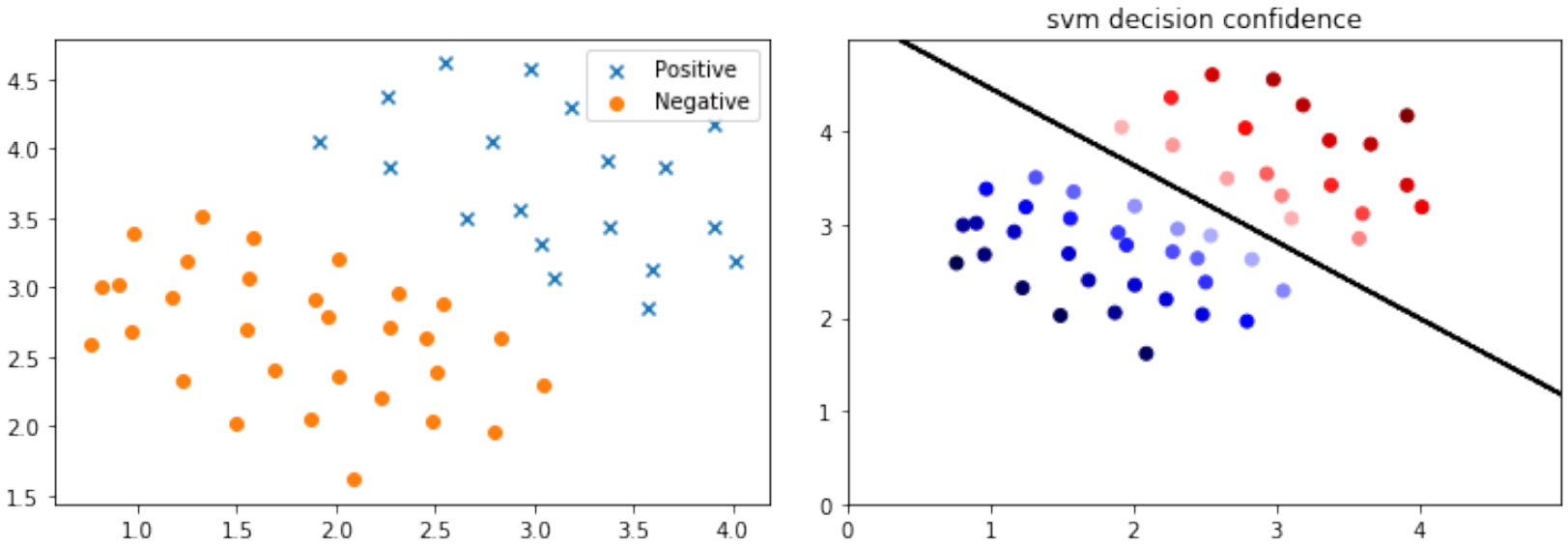


Figure: Linearly separable training data

Here, the black contour is the linear decision boundary which comes from a hard margin support vector machine.

Example 2. Soft margin support vector machine

Now, by adding one outlier to the previous data set, we make the data to be **non-linearly separable**. In this case, if we want to find a linear decision bound, we should use a soft margin support vector machine.

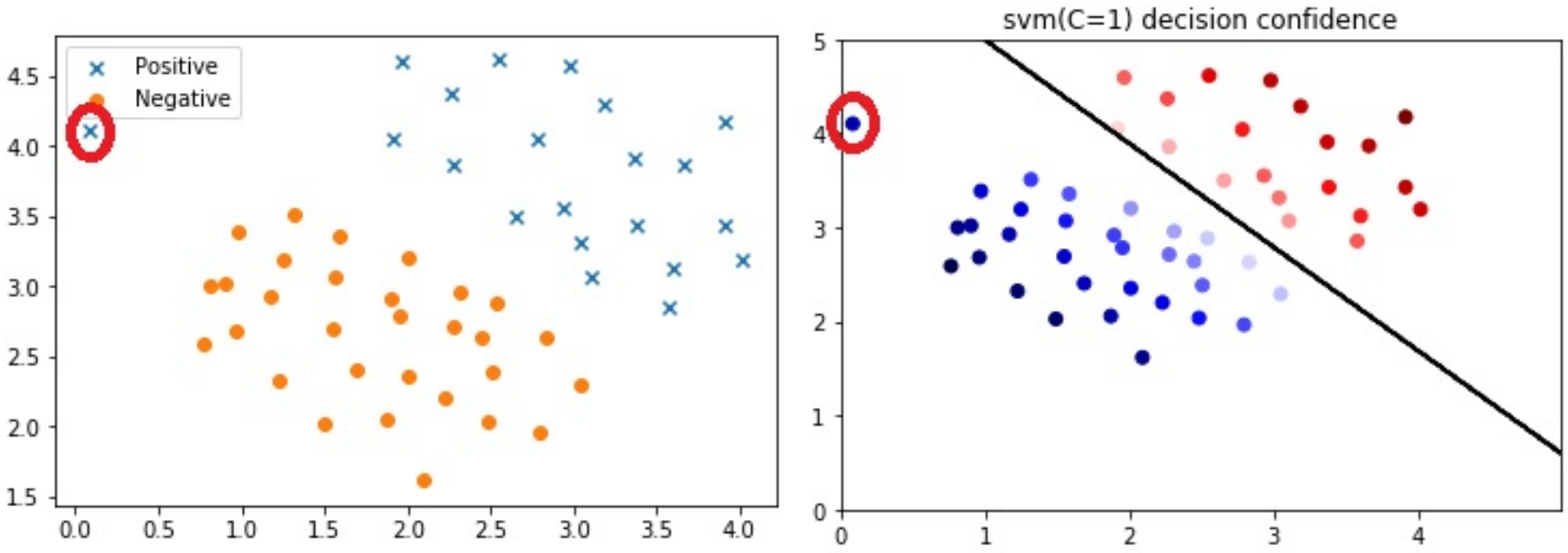


Figure: Results with a soft margin SVM

In this case, the soft margin SVM model suggests that mis-classifying the positive outlier is the best.

Example 3. Kernelized support vector machine

We now consider a classification problem with a **highly non-linearly separable** data set which needs a **non-linear decision boundary** as shown in the following two cases.

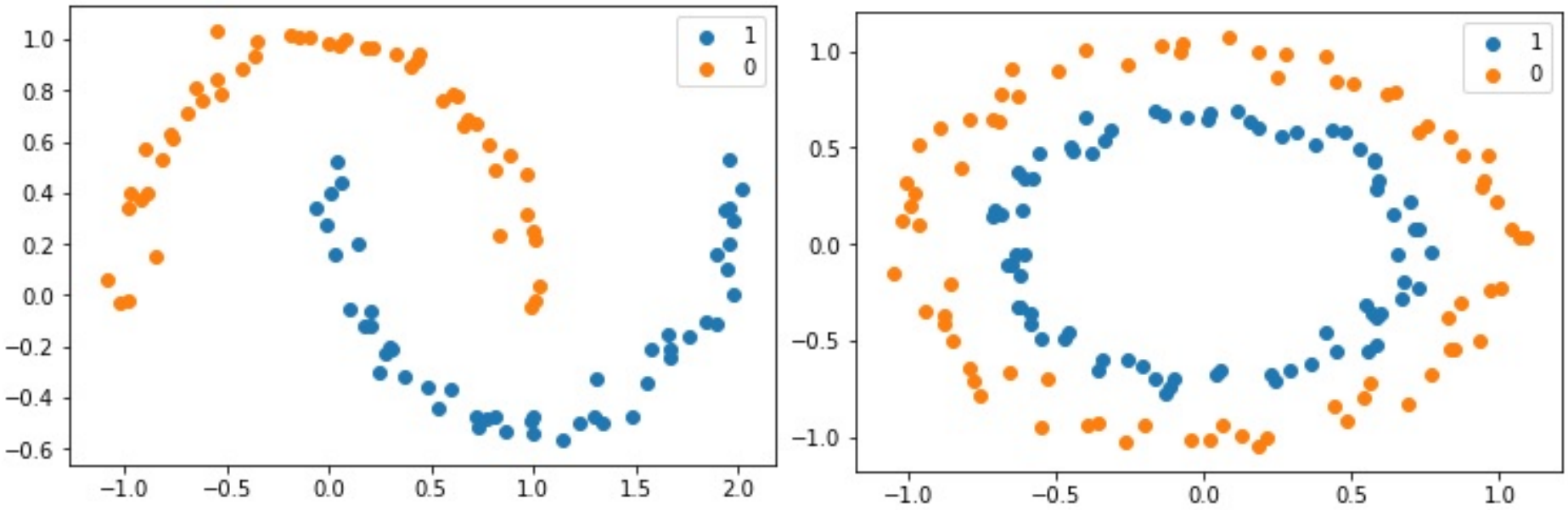


Figure: Examples of highly non-linearly separable data

To get a non-linear decision boundary, we use a kernel trick in support vector machine learning.

For two cases, we use the following polynomial kernel of degree 3.

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^3 = (x_1 z_1 + x_2 z_2)^3$$

We then get the following results.

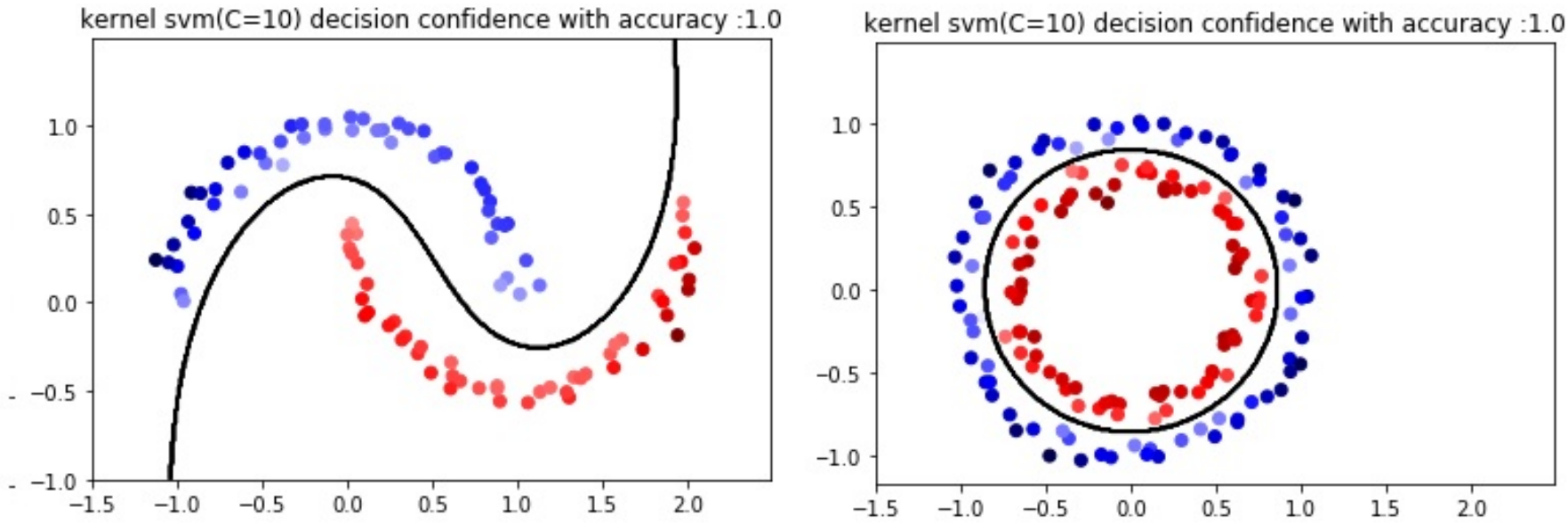


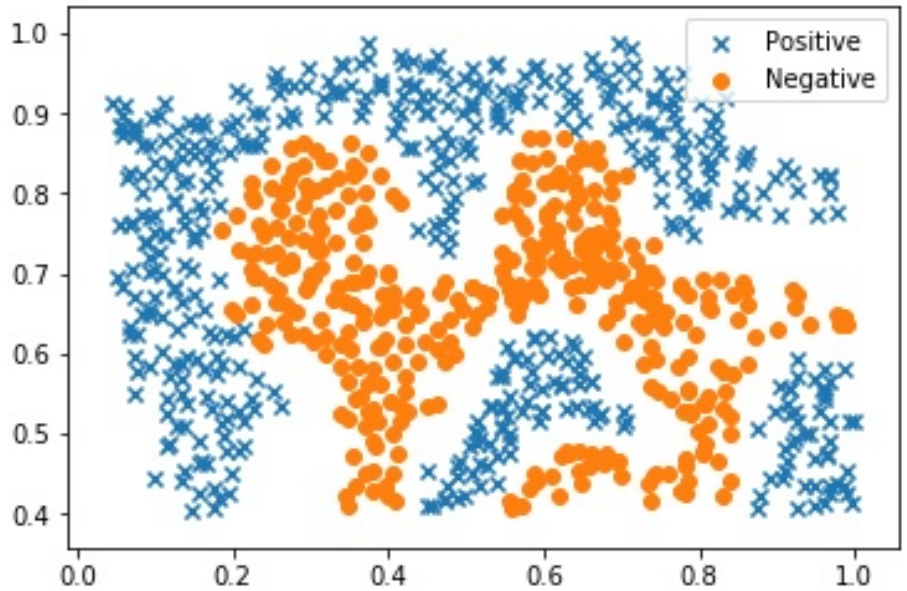
Figure: Results with a polynomial kernel of degree 3

Example 4. Kernelized SVM

In the kernelized svm, not only choosing a proper kernel, but also selecting a valid hyperparameter is also important. Consider the following radial basis function, which is a widely used kernel.

$$K(\mathbf{x}, \mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2/2\gamma^2}$$

For a soft margin kernelized svm with the radial basis function, choosing γ and the regularization term C is very important as shown in the following example.



By varying the values of C and γ, we get different decision boundaries as follows.

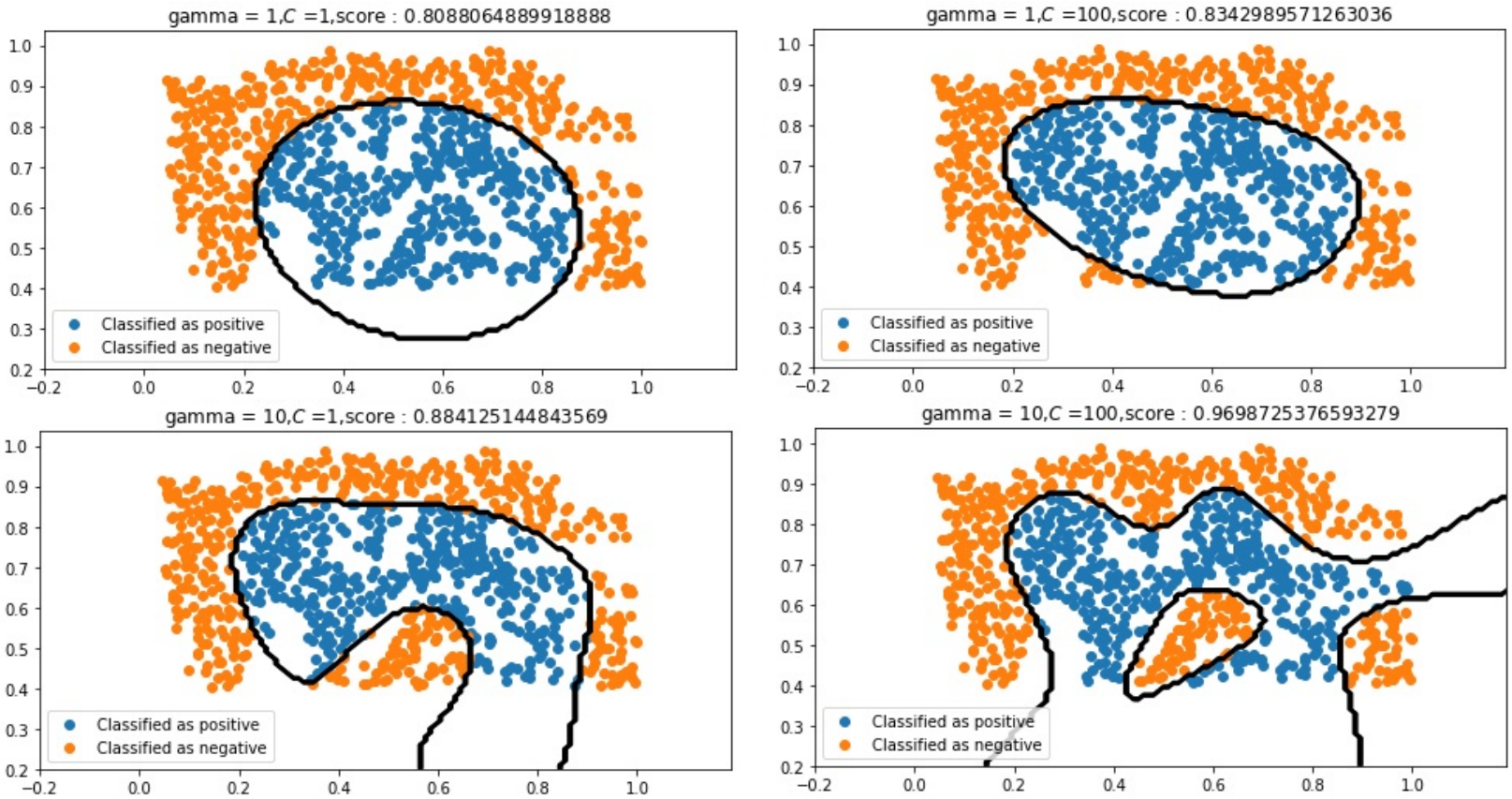


Figure: Results with a RBF kernel with different C and γ

Here, the score means the accuracy of the decision boundary for training data.