Machine Learning
ICT-4261

By-
Dr. Jesmin Akhter
Professor
Institute of Information Technology
Jahangirnagar University

Contents

The course will mainly cover the following topics:

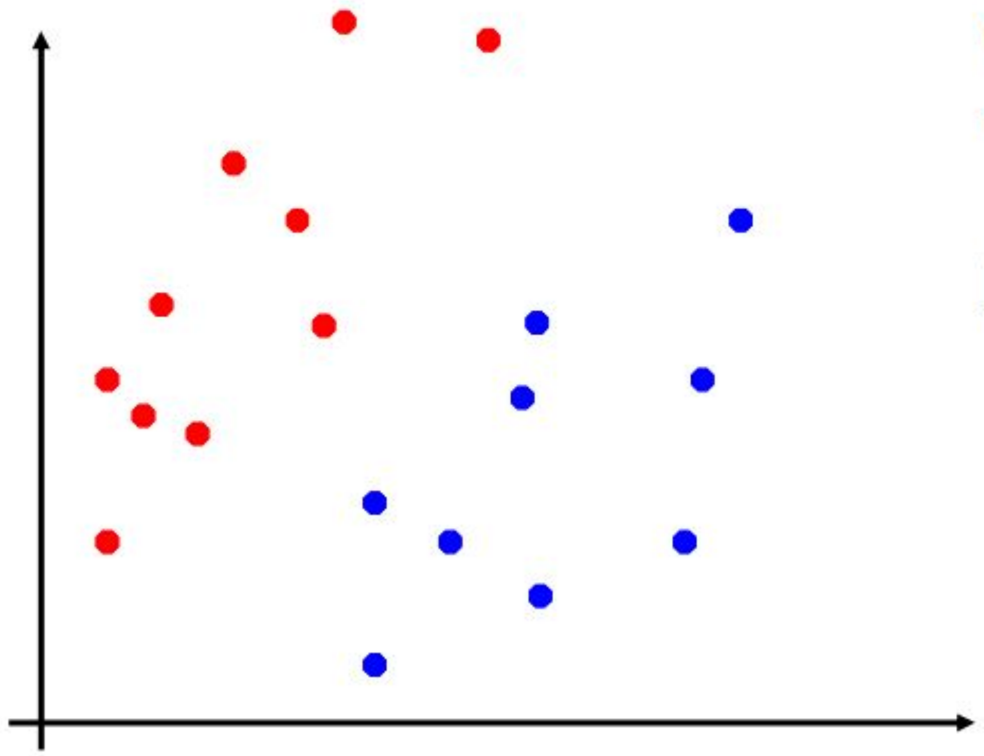
Outline

✔ Support Vector Machines

Support Vector Machines

- ✔ Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks.
- ✔ SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection.
- ✔ **SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships**.
- ✔ SVM algorithms are very effective as we try to find the **maximum separating hyperplane** between the different classes available in the target feature.
- ✔ Sometimes a **dataset can contain extreme values that are outside the range of what is expected and unlike the other data**. These are **called outliers** and often machine learning modeling and model skill in general can be improved by understanding and even removing these outlier values.

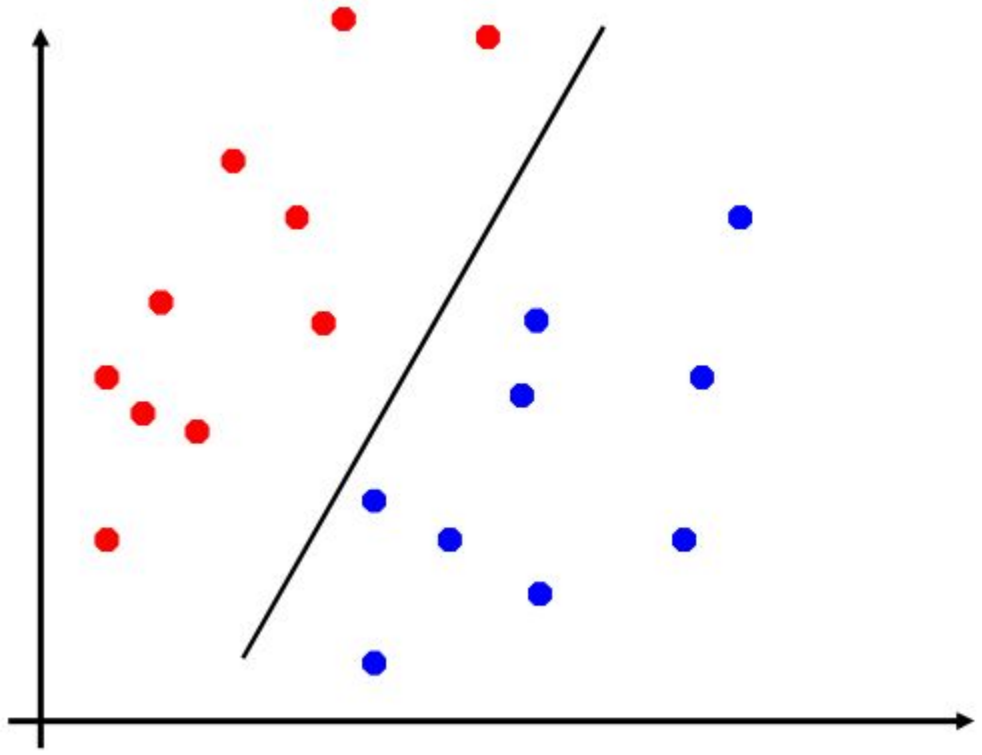
Linear Classifiers

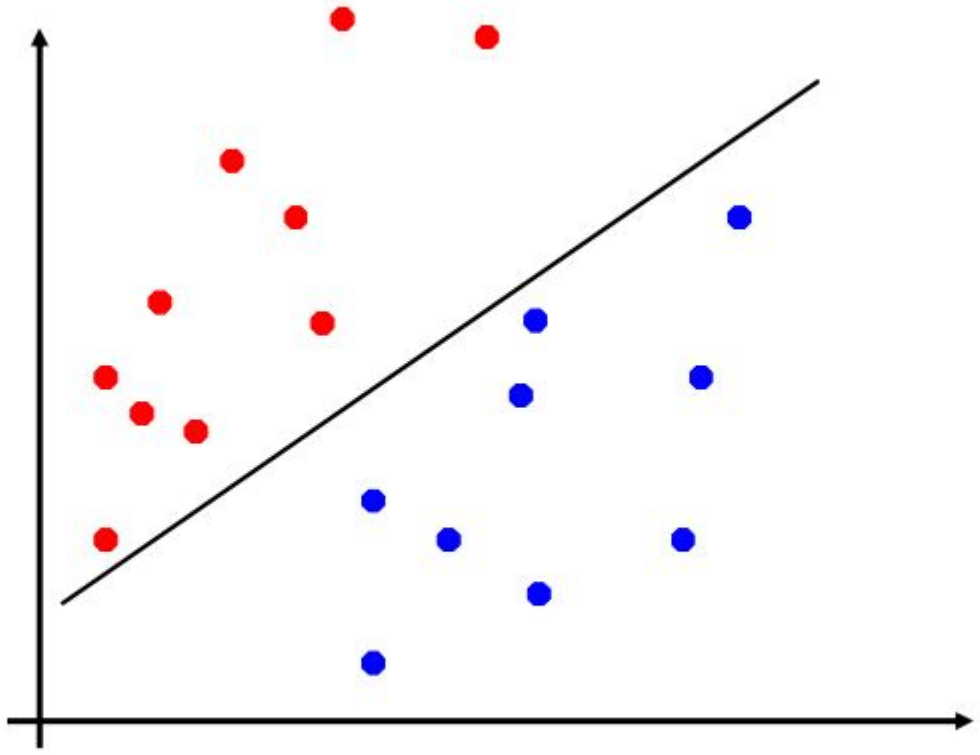


Consider a two dimensional dataset with two classes

How would we classify this dataset?

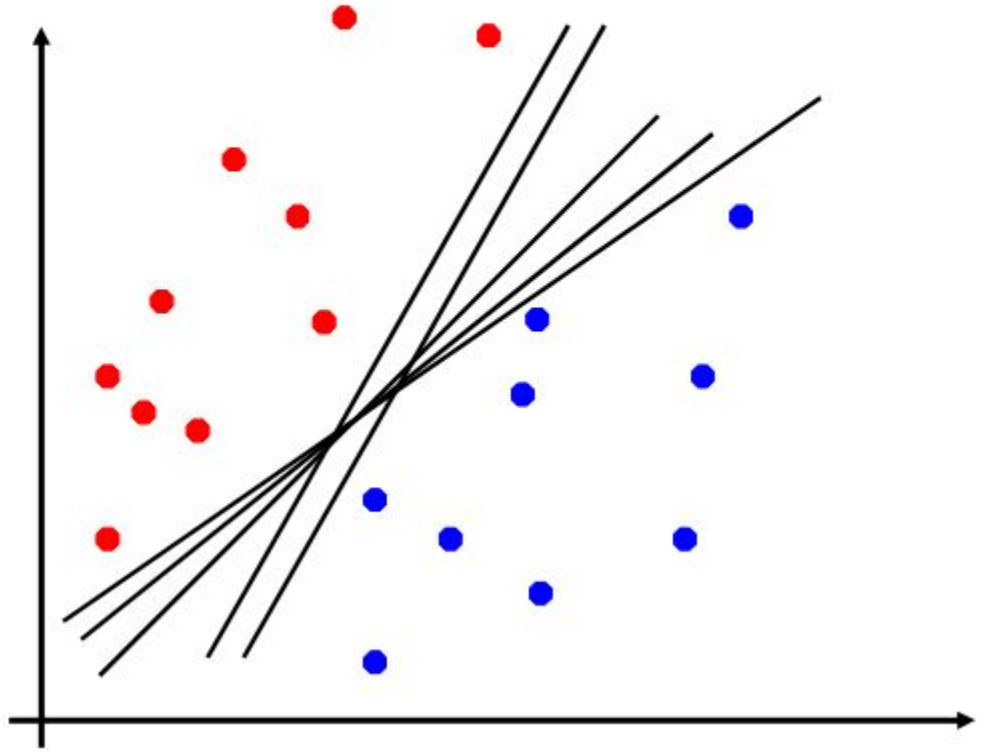
Linear Classifiers





Both of the lines can be linear classifiers.

Linear Classifiers

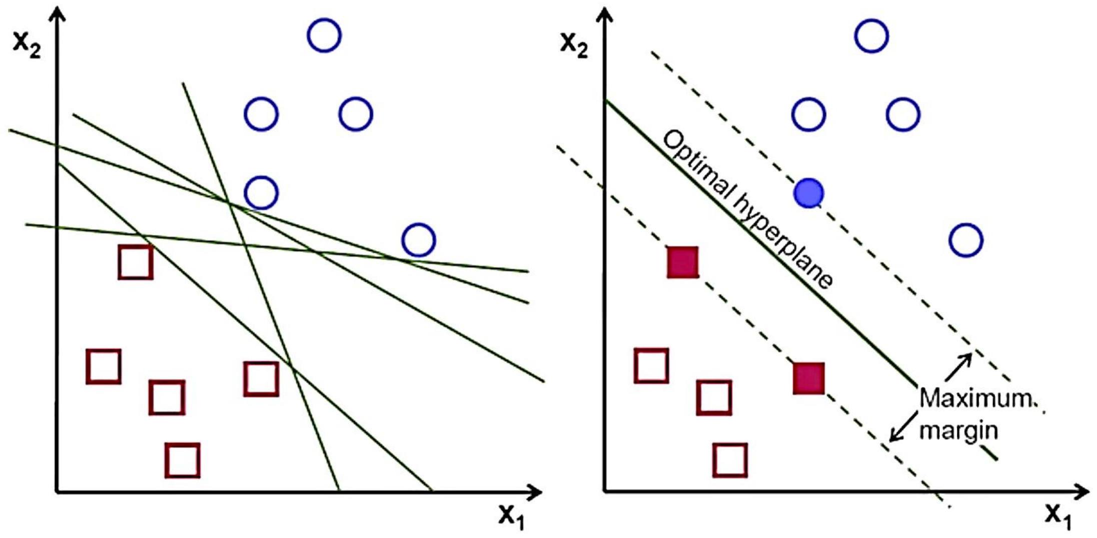


There are many lines that can be linear classifiers.

Which one is the optimal classifier.

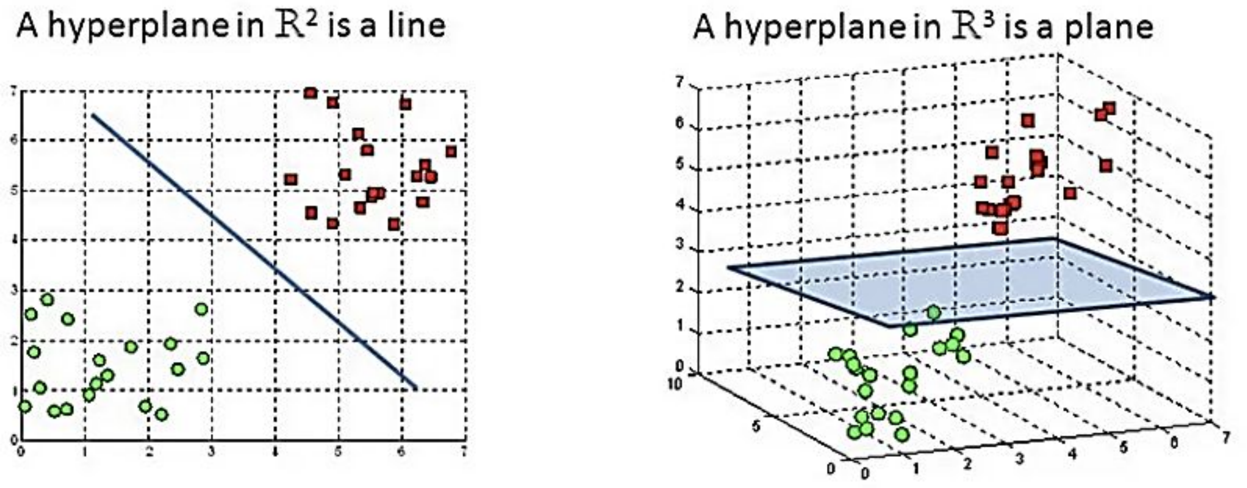
Objective-Support Vector Machines

- ✔ The main objective of the SVM algorithm is to **find the optimal hyperplane** in an N-dimensional space that can separate the data points in different classes in the feature space.
- ✔ The hyperplane tries that the **margin between the closest points of different classes should be as maximum as possible.**



Possible hyperplanes

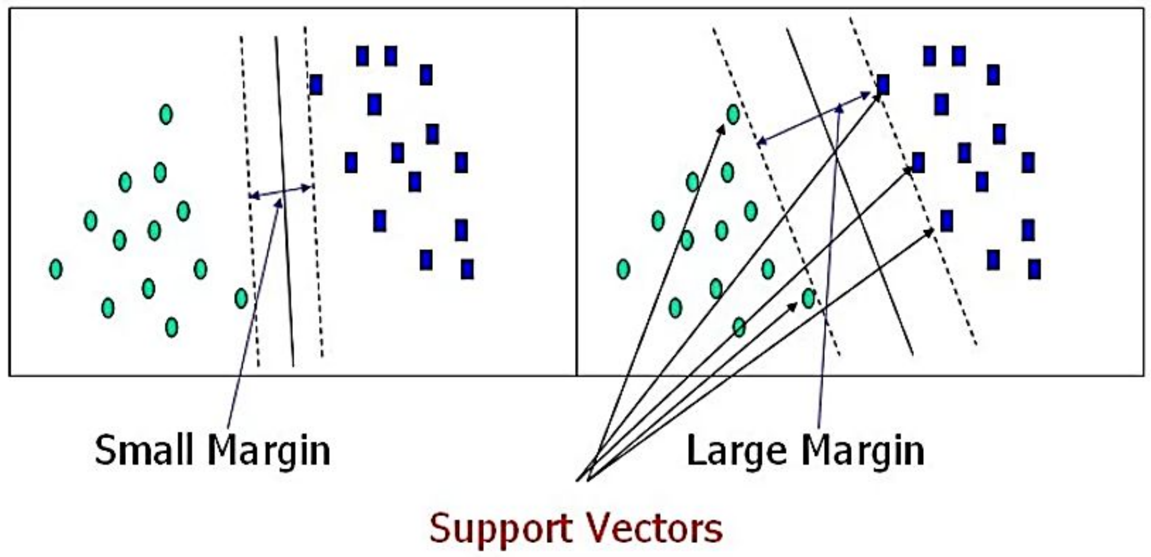
Hyperplanes and Support Vectors



Hyperplanes in 2D and 3D feature space

✔ Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Hyperplanes and Support Vectors



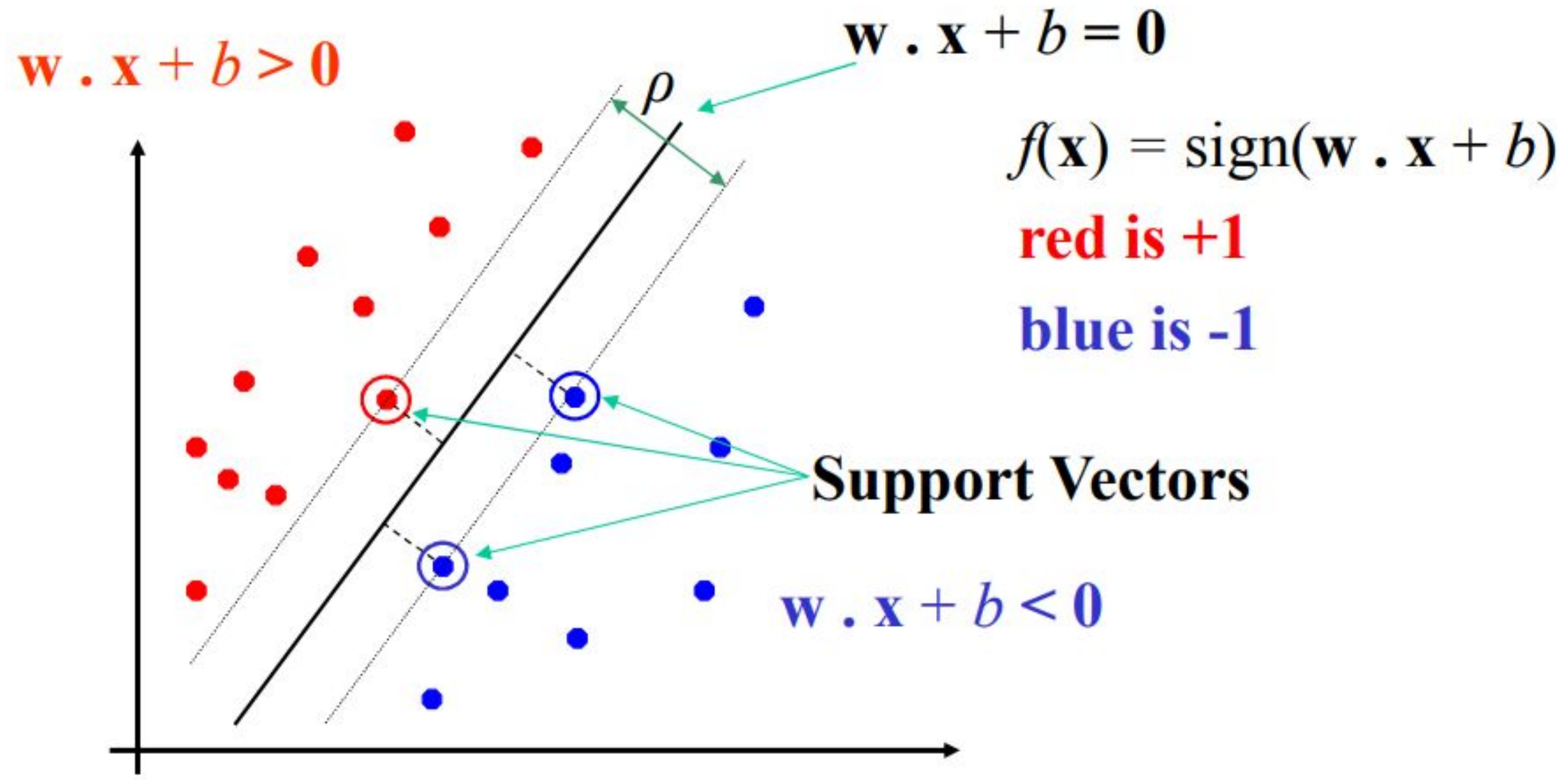
Support Vectors

✔ Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

Hyperplanes and Support Vectors

✔ A separating line will be defined with the help of **Support Vectors**.

- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



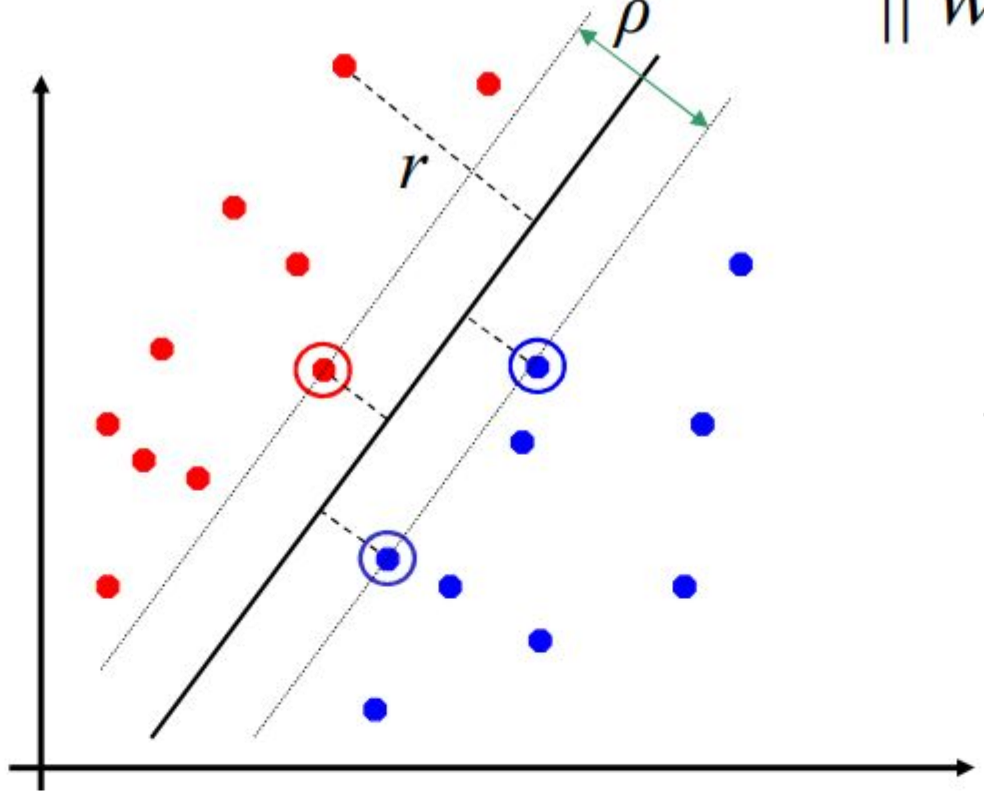
$\mathbf{w} \cdot \mathbf{x} + b > 0$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
ρ
$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$
red is +1
blue is -1
Support Vectors
$\mathbf{w} \cdot \mathbf{x} + b < 0$

Hyperplanes and Support Vectors

- Distance from example $\mathbf{x}_i$ to the separator is

$$r = \frac{\mathbf{w}.\mathbf{x}_i + b}{\|\mathbf{w}\|}$$

$$\| w \| \; is \; \sqrt{w_1^2 + \ldots + w_n^2}$$

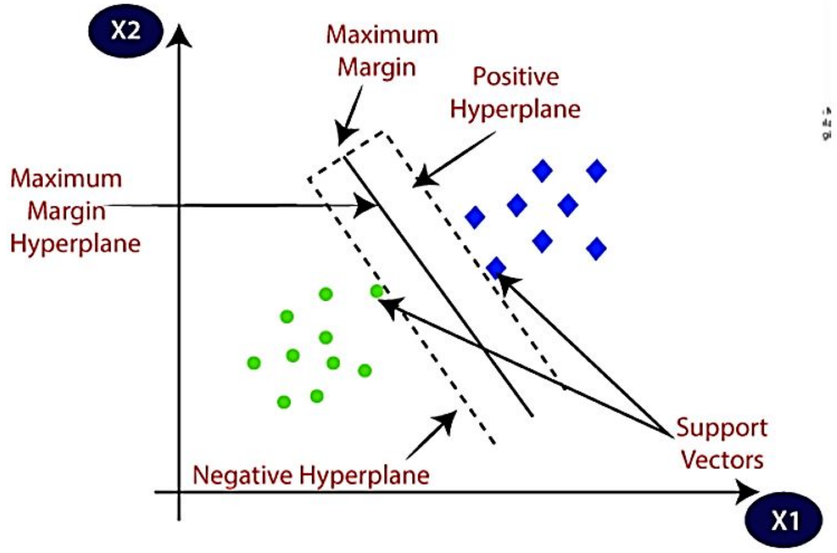


d = |Ax$_1$ + By$_1$ + C|]/ √(A^2 + B^2)

Margin

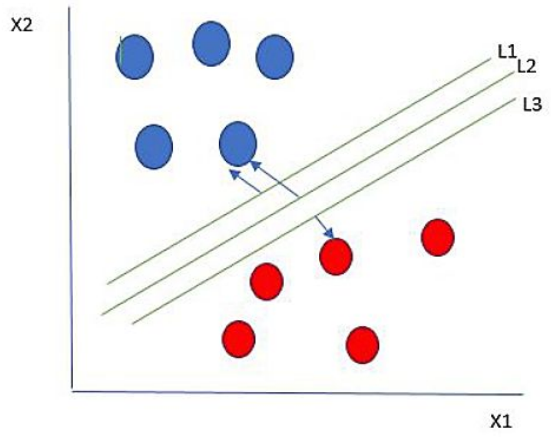
Margin:

- ✔ it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins **hard margin** and **soft margin.**
- ✔ **The Role of Margin in SVMs**
  When the data is linearly separable, and we don't want to have any misclassifications, we use SVM with a hard margin. However, when a linear boundary is not feasible, or we want to allow some misclassifications in the hope of achieving better generality, we can opt for a soft margin for our classifier.



How does SVM work?

- ✔ One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

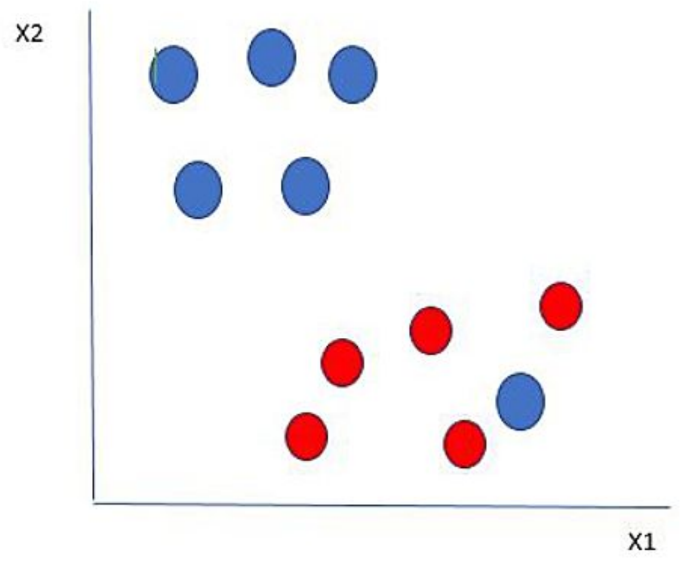


Multiple hyperplanes separate the data from two classes

- ✔ So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2

How does SVM work?

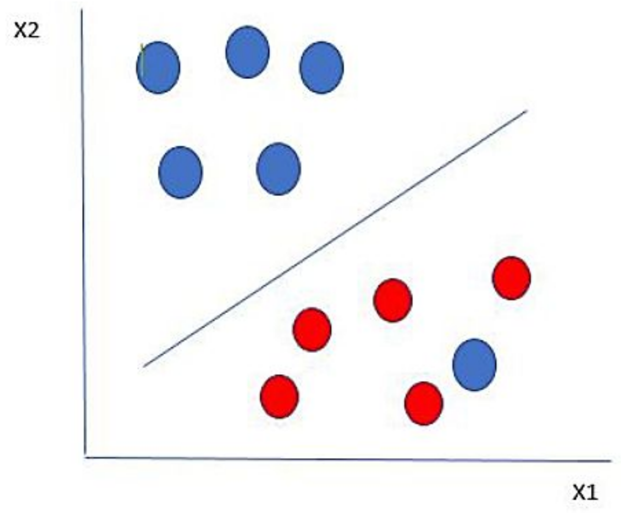
- ✔ Let's consider a scenario like shown below



Selecting hyperplane for data with outlier

- ✔ Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

How does SVM work?

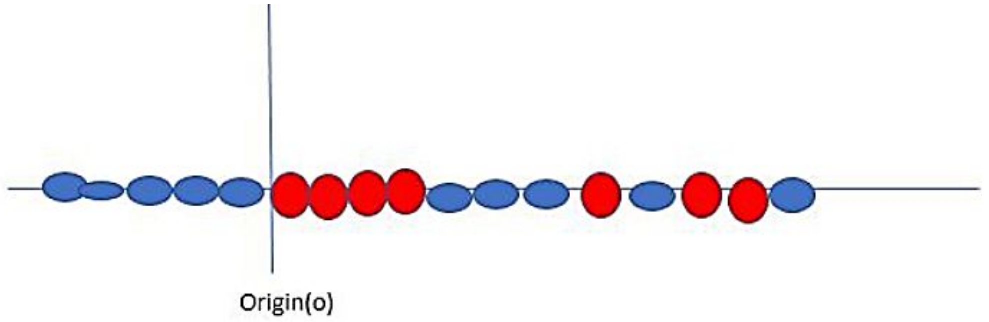


Hyperplane which is the most optimized one

- ✔ So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**.

How does SVM work?

- ✔ Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?

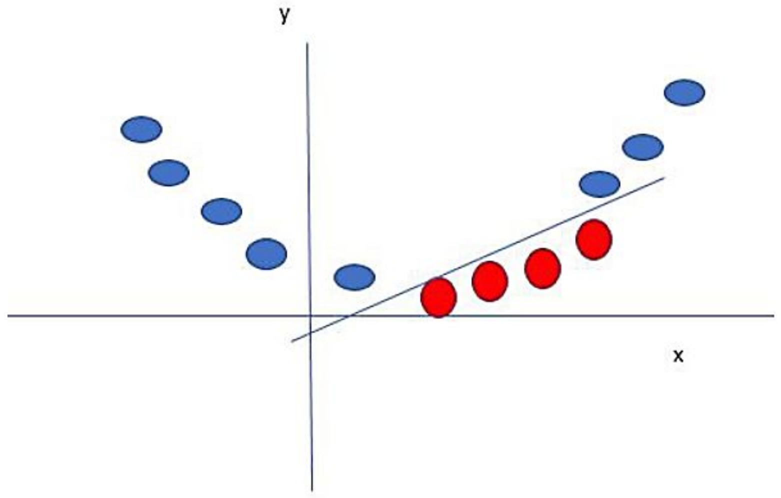


Original 1D dataset for classification

- ✔ Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**.

How does SVM work?

- ✔ We call a point x_i on the line and we create a new variable y_i as a function of distance from origin o. So if we plot this we get something like as shown below
- ✔



Mapping 1D data to 2D to become able to separate the two classes

- ✔ In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

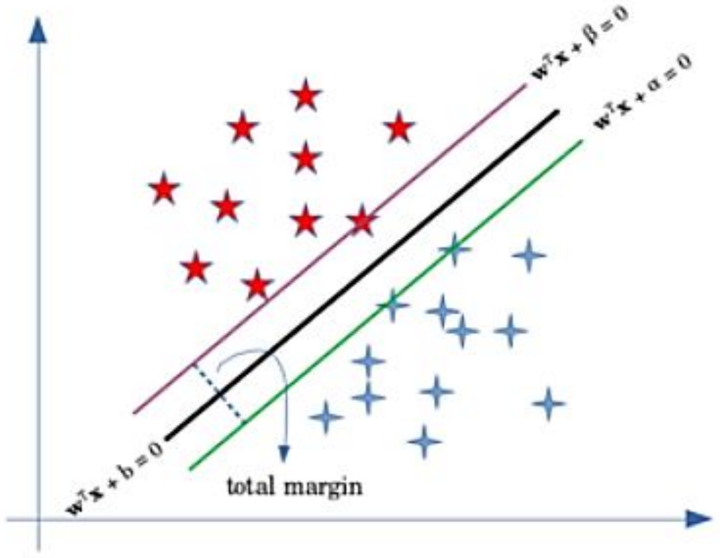
SVM with a Hard Margin

- ✔ assume that the hyperplane separating our two classes is defined as

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

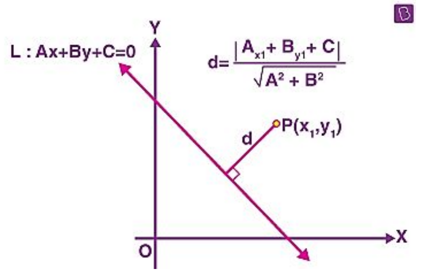
- ✔ Then, we can define the margin by two parallel hyperplanes:

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \alpha = 0,$$
$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \beta = 0$$



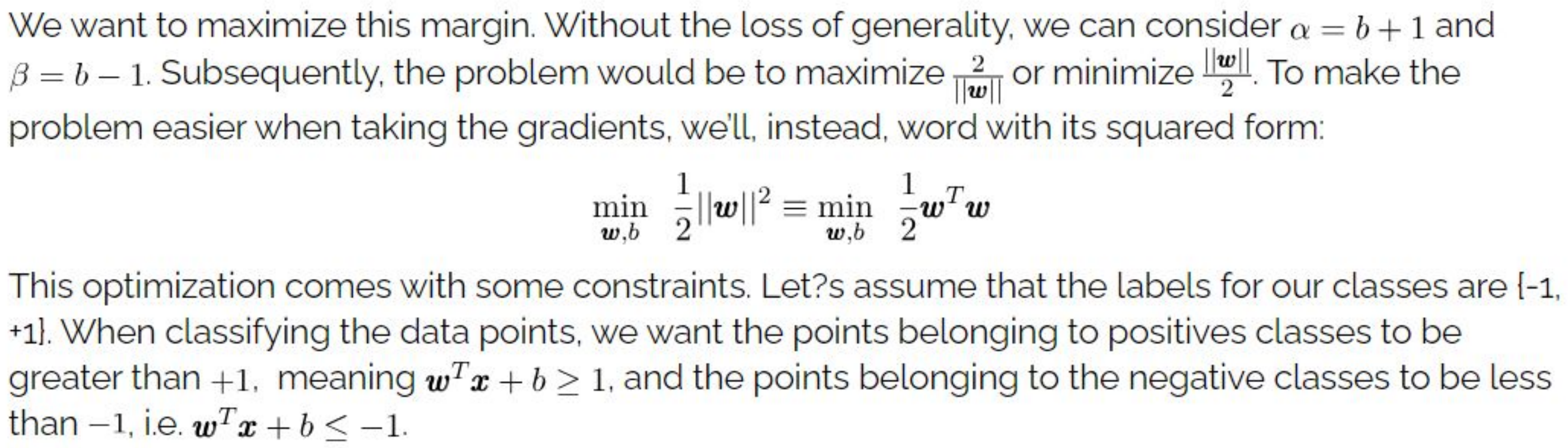
- ✔ They are the green and purple lines in the above figure. Without allowing any misclassifications in the hard margin SVM, we want to maximize the distance between the two hyperplanes.
- ✔ To find this distance, we can use the formula for the distance of a point from a plane. So the distance of the blue points and the red point from the black line would respectively be:

$$\frac{|\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \alpha|}{||\boldsymbol{w}||} \text{ and } \frac{|\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \beta|}{||\boldsymbol{w}||}$$



SVM with a Hard Margin

- ✓ Distance d between two parallel lines y = mx + α and y = mx + β is given by
  d = $\frac{|\alpha-\beta|}{||w||}$
- ✓ As a result total margin become $\frac{|\alpha-\beta|}{||w||}$



We want to maximize this margin. Without the loss of generality, we can consider $\alpha = b + 1$ and $\beta = b - 1$. Subsequently, the problem would be to maximize $\frac{2}{||\boldsymbol{w}||}$ or minimize $\frac{||\boldsymbol{w}||}{2}$. To make the problem easier when taking the gradients, we'll, instead, word with its squared form:

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}||\boldsymbol{w}||^2 \equiv \min_{\boldsymbol{w},b} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

This optimization comes with some constraints. Let?s assume that the labels for our classes are {-1, +1}. When classifying the data points, we want the points belonging to positives classes to be greater than $+1$, meaning $\boldsymbol{w}^T\boldsymbol{x} + b \geq 1$, and the points belonging to the negative classes to be less than -1, i.e. $\boldsymbol{w}^T\boldsymbol{x} + b \leq -1$.

SVM with a Hard Margin

We can combine these two constraints and express them as: $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$. Therefore our optimization problem would become:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\textbf{s.t.} \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$

$$y_i(wx_i + b) - 1 \geq 0$$

This optimization is called the primal problem and is guaranteed to have a global minimum. We can solve this by introducing Lagrange multipliers (α_i) and converting it to the dual problem:

For this we multiply all of the constraints by Lagrange multipliers (alphas) and subtract from our objective function:

$$L(\boldsymbol{w}, b, \alpha_i) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1)$$

This is called the Lagrangian function of the SVM which is differentiable with respect to $\boldsymbol{w}$ and b.

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \alpha) = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\nabla_b L(\boldsymbol{w}, b, \alpha) = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

SVM with a Hard Margin

✔ By substituting them in the second term of the Lagrangian function, we will get the dual problem of SVM:

$$\max_{\alpha} L_{dual} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

✔ The dual problem is easier to solve since it has only the Lagrange multipliers. Also, the fact that the dual problem depends on the inner products of the training data which is convenient when extending linear SVM to learn non-linear boundaries.

SVM with a Hard Margin

- ✔ the quadratic optimization problem is a well studied problem. A quadratic objective function with linear inequality constraints must satisfy some conditions at the solution. These conditions are known as KKT conditions and are named after Karush, Kuhn and Tucker who independently derived them.
- ✔ The KKT conditions are:

1) $\nabla_w L(w, b, \alpha) = 0 \quad \& \quad \dfrac{\partial L(w, b, \alpha)}{\partial b} = 0$

2) $\alpha_i \geq 0$

3) $\alpha_i \left(y_i (w x_i + b) - 1 \right) = 0$

4) $y_i (w x_i + b) - 1 \geq 0$

SVM with a Hard Margin

- ✔ First lets expand the Lagrangian to isolate the w and b terms:

$$L(w,b,\alpha)=\frac{1}{2}w\cdot w-\sum_{i=1}^{N}\alpha_i\left(y_i\left(wx_i+b\right)-1\right)$$

$$L(w,b,\alpha)=\frac{1}{2}w\cdot w-\sum_{i=1}^{N}\alpha_i\left(y_i wx_i+y_i b-1\right)$$

$$L(w,b,\alpha)=\frac{1}{2}w\cdot w-\sum_{i=1}^{N}\alpha_i y_i wx_i-\sum_{i=1}^{N}\alpha_i y_i b+\sum_{i=1}^{N}\alpha_i$$

- ✔ Now we can take the derivative of our Lagrange function with respect to the vector w to get:

$$\nabla_w = w-\sum_{i=1}^{N}\alpha_i y_i x_i$$

- ✔ And by setting the result equal to zero and solving for w we get:

$$w=\sum_{i=1}^{N}\alpha_i y_i x_i$$

SVM with a Hard Margin

- ✔ Now we can move on to the derivative of the Lagrange formula with respect to b. We have:

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

- ✔ And we again set the result equal to zero:

$$-\sum_{i=1}^{N} \alpha_i y_i = 0$$

SVM with a Hard Margin

✔ Now let's substitute these results into the Lagrange function:

$$L(w,b,\alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{N} \alpha_i \left(y_i (w x_i + b) - 1 \right)$$

$$L(w,b,\alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{N} \alpha_i y_i w x_i - b \underbrace{\sum_{i=1}^{N} \alpha_i y_i}_{0} + \sum_{i=1}^{N} \alpha_i$$

$$L(w,b,\alpha) = \frac{1}{2} \sum_{i=1}^{N} \alpha_i y_i x_i \cdot \sum_{i=1}^{N} \alpha_i y_i x_i - \sum_{i=1}^{N} \alpha_i y_i x_i \cdot \sum_{i=1}^{N} \alpha_i y_i x_i + \sum_{i=1}^{N} \alpha_i$$

$$L(w,b,\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \alpha_i y_i x_i \cdot \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$L(w,b,\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j \cdot$$

$$\max_{\alpha} L_{dual} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

SVM with a Hard Margin

- ✔ It is important to note that we are now trying to maximize this Lagrange function since it is only a function of alphas. This is the so-called Dual formulation of the optimization problem.

$$\max_{\alpha} L_{dual} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

SVM with a Hard Margin

- ✔ Once we solve for alphas we need to be able to use our classifier. That means we need the coefficients of our hyperplane. To get back our coefficients w we can use below equation that we derived earlier:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

- ✔ Once we have our w we can use the fact that for any of the support vectors we have:

$$y_i(wx_i + b) = 1$$

- ✔ And we can solve this for our b coefficient:

$$b = \frac{1}{y_i} - wx_i$$

SVM with a Hard Margin

- ✔ Now that we have our hyperplane coefficients we can use our classifier to assign any observation xi by plugging it into below equation:

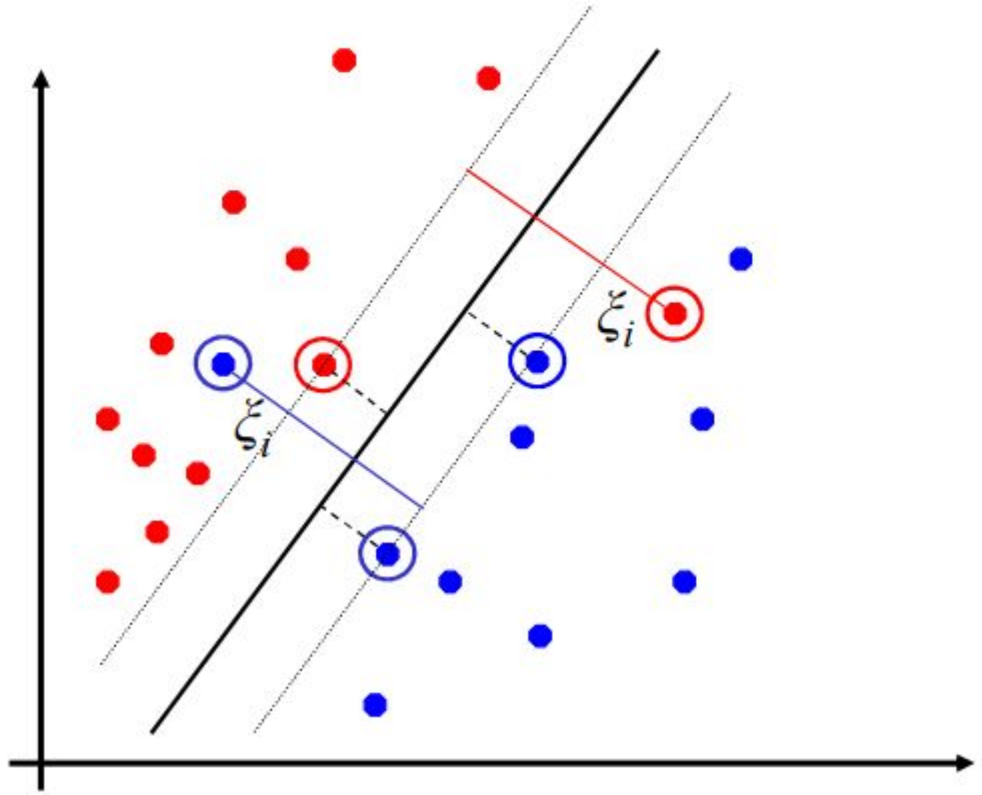
$$y_i = sign(wx_i + b)$$

SVM with a Hard Margin

- ✔ In a hard margin SVM, the goal is to find the hyperplane that can perfectly separate the data into two classes without any misclassification. However, this is not always possible when the data is not linearly separable or contains outliers. In such cases, the hard margin SVM will fail to find a hyperplane that can perfectly separate the data, and the optimization problem will have no solution.

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



SVM with a Soft Margin

- ✔ First, we allow misclassifications to happen.
- ✔ Second, to minimize the error, we should define a loss function. **A common loss function used for soft margin is the hinge loss.**
- ✔ The loss of a misclassified point is called a slack variable and is added to the primal problem that we had for hard margin SVM. So the primal problem for the soft margin becomes

$$\min \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} \zeta_i$$

$$\textbf{s.t.} \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \zeta_i \quad \forall i = 1, ..., n, \ \zeta_i \geq 0$$

- ✔ C is the control parameter
- ✔ We would like to minimize the sum total of the penalties *ξi* over all i

SVM with a Soft Margin

- ✔ The introduction of the **slack variables allows us to create a flexible margin** so that some vectors belonging to a class can also be found in the opposite part of the hyperspace and can be included in the model training. The strength of this flexibility can be set using the parameter C.
- ✔ Small values (close to zero) bring about very hard margins, while values greater than or equal to 1 allow more and more flexibility (also increasing the misclassification rate).
- ✔ Ideally, we are interested in the number of non zero ξi , as that is the count of errors made by our classifier on the set of training examples.
- ✔ We take the lagrangian in the usual manner:

$$\underbrace{L(w,\xi,b,\alpha) = \frac{1}{2}w^T w}_{\text{original objective}} + \underbrace{C\sum_{i=1}^{m}\xi_i}_{\text{slack}} + \underbrace{\sum_{i=1}^{m}\alpha_i\left[1-\xi_i-y_i(w^T x_i+b)\right] + \sum_{i=1}^{m}\lambda_i(0-\xi_i)}_{\text{converted constraints}}$$

SVM with a Soft Margin

To find the dual form of the problem, we first need to minimize $\mathcal{L}(w, \xi, b, \alpha)$ with respect to w, ξ, and b (for fixed α), to get Θ_D.

$$\underset{w,\xi,b}{min} \ \mathcal{L}(w, \xi, b, \alpha), \quad \xi_i \geq 0 \tag{5}$$

Since the Lagrangian function is linear in α, we cannot set the gradient with respect to α to zero. We obtain the following dual optimization problem:

$$\max_{\alpha} L_{dual} = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left\langle \phi(x^{(i)}), \phi(x^{(j)}) \right\rangle . \tag{6}$$

. $$0 \leq \alpha_i \leq C \tag{7}$$

$$\tag{8}$$

SVM with a Soft Margin

take partial derivative w.r.t w, b and ξ_i to turn this to dual Lagrangian.

$$\delta_b L = \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\delta_{\xi_i} L = C - \alpha_i - \lambda_i = 0$$

$$\delta_w L = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$w = \sum_{i=1}^{m} \alpha_i x_i^T y_i$$

Plugging these to the Lagrangian we get,

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

We obtain the following dual optimization problem after applying kernel technique:

$$\max_{\alpha} L_{dual} = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left\langle \phi(x^{(i)}), \phi(x^{(j)}) \right\rangle .$$

$$0 \le \alpha_i \le C$$

SVM with a Soft Margin
Detail calculation

✔ L should be minimized w.r.t β and b, and should be maximized w.r.t αi. Hence, instead of minimizing the Primal Problem, we can now maximize the Dual Problem

$$\textbf{Objective Function: } \max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\textbf{Linear Constraints: } \alpha_i \geq 0, \forall i \in D, \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

SVM with a Soft Margin
Detail calculation

Duality: Soft Margin Classifier

Similar to the Hard Margin Classifier, we will derive the Dual Problem for Soft Margin Classifier.

Primal Objective function,

Objective Function: $\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i)^k$

$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$ [The value of k could be set to either 1 (Hinge Loss) or 2 (Quadratic Loss)]

s.t Linear Constant for soft margin: $y_i(w^T x_i + b) >= 1 - \xi_i$ where $\xi_i >= 0$

SVM with a Soft Margin
Detail calculation

Hinge Loss

We can compute the Lagrangian by introducing two Lagrange multipliers λ_i and α_i ; as we have two inequality constraints.

$$\alpha_i \, (1 - \xi_i - y_i \, (w^T x_i + b) = 0, \text{ and } \alpha_i \geq 0$$
$$\lambda_i \, (0 - \xi_i) = 0, \text{ and } \lambda_i \geq 0$$

The Lagrangian can be then defined as,

$$L(w, \zeta, b, \alpha) = \frac{1}{2} w \, w^T + C \sum_{i=1}^{m} \zeta_i + \sum_{i=1}^{m} \lambda_i (0 - \zeta_i) + \sum_{i=1}^{m} \alpha_i [1 - \zeta_i - y_i (x_i^T w + b)]$$

$$= \frac{1}{2} w \, w^T + C \sum_{i=1}^{m} \zeta_i - \sum_{i=1}^{m} \lambda_i \, \zeta_i - \sum_{i=1}^{m} \alpha_i (y_i (w^T x_i + b) - 1 + \zeta_i)$$

SVM with a Soft Margin Detail calculation

We will take partial derivative w.r.t ß, b and §; to turn this to dual Lagrangian

$$S_b L = \sum \alpha_i y_i = 0$$

$$\delta_{\zeta_i} L = C - \alpha_i - \lambda_i = 0$$

$$\delta_w L = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$\delta_{\xi_i} L$ does not have the summation ($\sum$) term, as we are taking partial derivative against ξ_i and not ξ.

SVM with a Soft Margin
Detail calculation

Plugging these to the Lagrangian we get,

$$L_{dual} = \frac{1}{2} w\, w^T + C \sum_{i=1}^{m} \zeta_i - \sum_{i=1}^{m} \lambda_i\, \zeta_i - \sum_{i=1}^{m} \alpha_i (y_i (w^T x_i + b) - 1 + \zeta_i)$$

$$= \frac{1}{2} w\, w^T - \sum_{i=1}^{m} (C - \alpha_i - \lambda_i) \zeta_i - b \sum_{i=1}^{m} \alpha_i\, y_i + \sum_{i=1}^{m} \alpha_i - w^T \left(\sum_{i=1}^{m} \alpha_i\, y_i x_i \right)$$

$$= \frac{1}{2} w\, w^T - \sum_{i=1}^{m} (0) \zeta_i - b \cdot 0 + \sum_{i=1}^{m} \alpha_i - w^T w$$

$$= -\frac{1}{2} w\, w^T + \sum_{i=1}^{m} \alpha_i$$

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=i}^{m} \alpha_i\, \alpha_j y_i y_j x_i^T x_j$$

SVM with a Soft Margin Detail calculation

The Dual Objective can be written as,

$$\textbf{Objective Function: } \max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\textbf{Linear Constraints: } 0 \leq \alpha_i \leq 0, \forall i \in D, \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Notice the objective function is same as the Hard Margin Classifier, however the inequality constraint is different.

Since $\boldsymbol{\alpha_i + \lambda_i = C}$ and $\boldsymbol{\alpha_i \geq 0, \lambda_i \geq 0}$, we can say $\boldsymbol{0 \leq \alpha_i \leq 0}$. There is no constraint on $\boldsymbol{\lambda}$ as its not part of the final equation.

Hard Margin vs. Soft Margin

- ✔ **The difference between a hard margin and a soft margin in SVMs lies in the separability of the data. If our data is linearly separable, we go for a hard margin.** However, if this is not the case, it won’t be feasible to do that. In the presence of the data points that make it impossible to find a linear classifier, we would have to be more lenient and let some of the data points be misclassified. In this case, a soft margin SVM is appropriate.
- ✔ **Sometimes, the data is linearly separable, but the margin is so small that the model becomes prone to overfitting or being too sensitive to outliers. Also, in this case, we can opt for a larger margin by using soft margin SVM in order to help the model generalize better.**

Hard Margin vs. Soft Margin

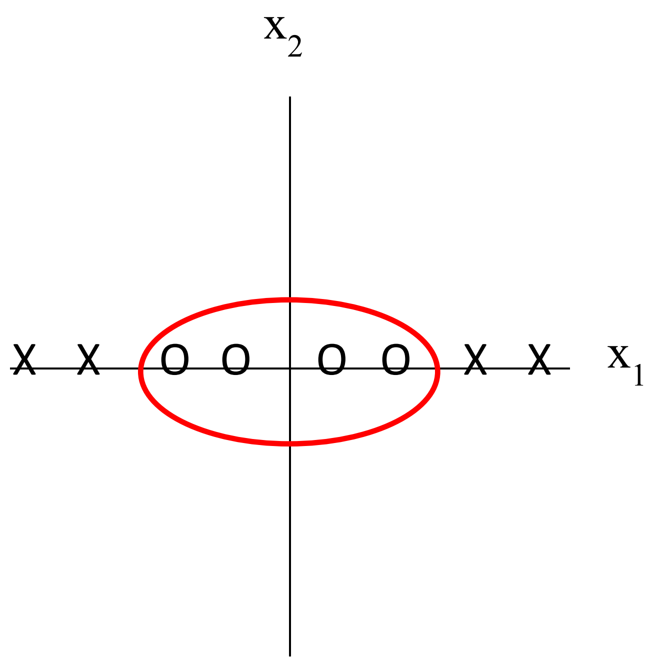
- ✔ Geometrically, the soft margin SVM introduces a penalty for the data points that lie on the wrong side of the margin or even on the wrong side of the hyperplane. The slack variables ξi allow these data points to be within the margin or on the wrong side of the hyperplane, but they incur a penalty in the objective function. The optimization problem in a soft margin SVM finds the hyperplane that maximizes the margin while minimizing the penalty for the misclassified data points.
- ✔ hard margin SVM aims to find a hyperplane that perfectly separates the data into two classes without any misclassification, while soft margin SVM allows some misclassification by introducing slack variables. Geometrically, the hard margin SVM looks for the widest margin possible, while the soft margin SVM finds the hyperplane that maximizes the margin while minimizing the penalty for misclassification.

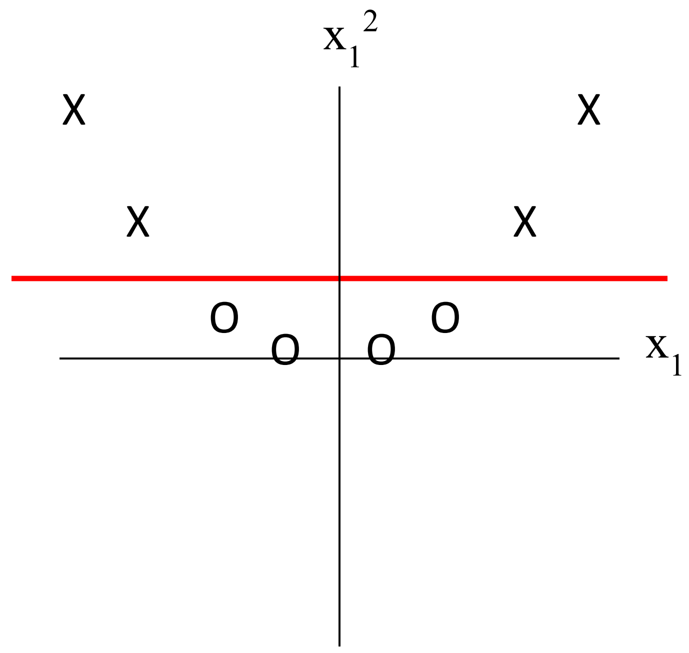
Kernel Methods

Making the Non-Linear Linear

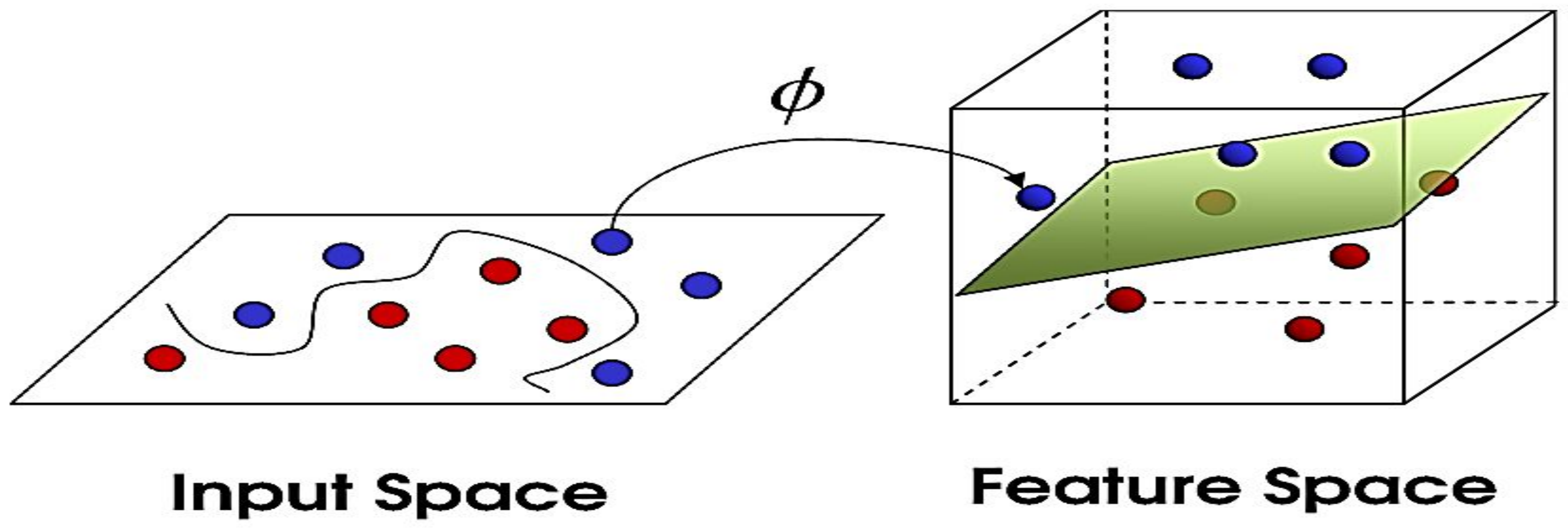
When Linear Separators Fail

In machine learning applications, the data can be text, image, or video. So, there is a need to extract features from these data prior to classification. Hence, in the real world, many classification models are complex and mostly require non-linear hyperplanes.





Mapping into a New Feature Space



$$\Phi : x \Box X = \Phi(x)$$

$$\Phi(x_1,x_2) = (x_1,x_2,x_1{}^2,x_2{}^2,x_1x_2)$$

- ✔ Rather than run SVM on x_i, run it on $\Phi(x_i)$
- ✔ Find non-linear separator in input space
- ✔ What if $\Phi(x_i)$ is really big?
- ✔ Use kernels to compute it implicitly!

Kernel Trick in Support Vector Machine

- For example, one mapping function $\emptyset : \boldsymbol{R}^2 \rightarrow \boldsymbol{R}^3$ used to transform a 2D data to 3D data is given as follows:

$$\emptyset(x, y) = (x^2, \sqrt{2}xy, y^2)$$

- Consider a point (2, 3) in 2D space, if you apply above mapping function we can convert it into 3D space and it looks like this,
- Here $x = 2 \; and \; y = 3$,
- Hence point in 3D space is:

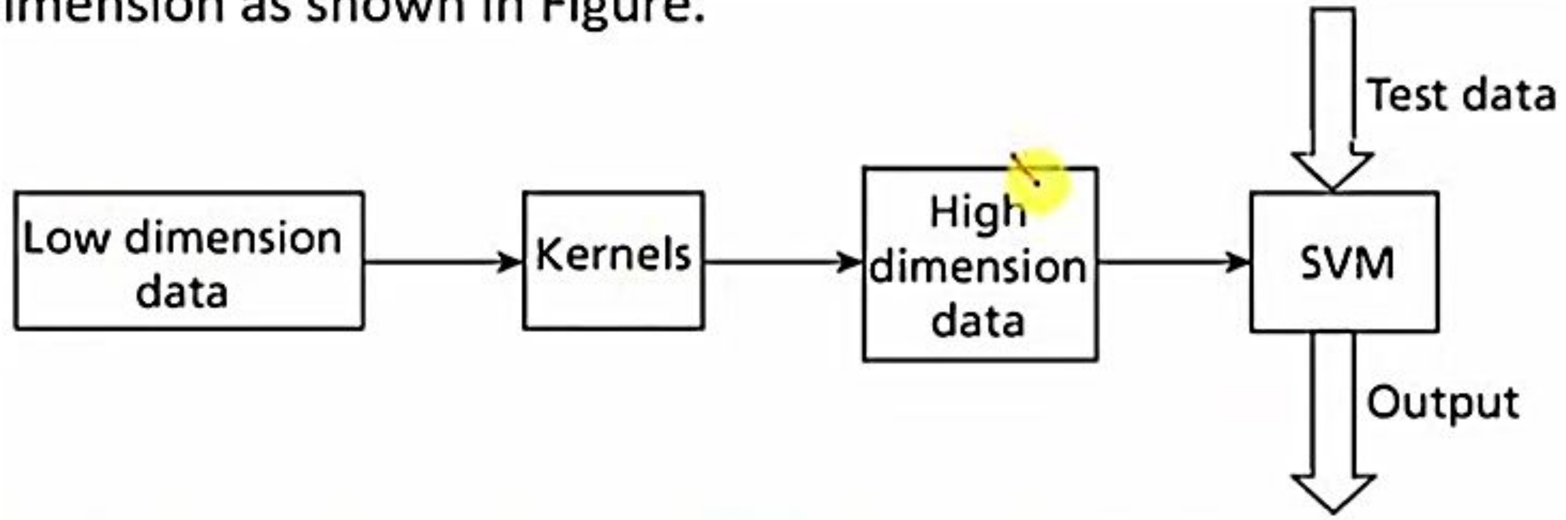
$$(2^2, \sqrt{2} * 2 * 3, 3^2) = (4, 6\sqrt{2}, 9)$$

Kernel Trick in Support Vector Machine

- While mapping functions play ✓ an important role, there are many disadvantages, as mapping involves more computations and learning costs. D_1 D_2
- Also, the disadvantages of transformations are that there is no generalized thumb rule available describing what transformations should be applied and if the data is large, mapping process takes huge amount of time.
- In real applications, there might be many features in the data and applying transformations that involve many polynomial combinations of these features will lead to extremely high and impractical computational costs.
- In this context, only kernels are useful.
- Kernels are used to compute the value without transforming the data.

Kernel Trick in Support Vector Machine

- **What is a Kernel?**
- Kernels are a set of functions used to transform data from lower dimension to higher dimension and to manipulate data using dot product at higher dimensions.
- The use of kernels is to apply transformation to data and perform classification at the higher dimension as shown in Figure.



Kernel Trick in Support Vector Machine

Types of Kernels

- Linear Kernel
- Polynomial Kernel
- Exponential Kernel
- Homogeneous Kernel
- Inhomogeneous Kernel
- Gaussian Kernel
- Hyperbolic or the Sigmoid Kernel
- Radial-basis function kernel
- Etc…

Types of Kernel in SVM

- Linear Kernel
- Polynomial Kernel
- Exponential Kernel
- Homogeneous Kernel

- Inhomogeneous Kernel
- Gaussian Kernel
- Sigmoid Kernel
- Radial-basis function kernel

Types of Kernel in Support Vector Machine

Linear Kernel

- Linear kernels are of the type

$$\underline{k(x, y) =} \; x^T . y$$

- where x and y are two vectors.
- Therefore $k(x, y) = \underline{\emptyset(x)}. \emptyset(y) = x^T . y.$

Types of Kernel in Support Vector Machine

Polynomial Kernel

- Polynomial kernels are of the type:

$$\underline{k(x,y) = (x^T y)^q}$$

- This is called homogeneous kernel.
- Here, q is the degree of the polynomial.
- If q = 2 then it is called **quadratic kernel**.

Types of Kernel in Support Vector Machine

Polynomial Kernel

- For inhomogeneous kernels, this is given as:

$$k(x, y) = (c + x^T y)^q$$

- Here c is a constant and q is the degree of the polynomial.
- If c is zero and degree is one, the polynomial kernel is reduced to a linear kernel.
- The value of degree q should be optimal as more degree may lead to overfitting.

Types of Kernel in Support Vector Machine

- Consider two data points $x = \begin{pmatrix}1\\2\end{pmatrix}$ and $y = (2,3)$ with c =1. Apply linear, homogeneous and inhomogeneous kernels.
- **Solution:**
- The kernel is given by $k(x,y) = (x^T y)^q$
- If q = 1 the it is called linear kernel
- $k(x,y) = \left(\begin{pmatrix}1\\2\end{pmatrix}^T (2,3)\right)^1 = 8$

Types of Kernel in Support Vector Machine

- Consider two data points $x = \begin{pmatrix}1\\2\end{pmatrix}$ and $y = (2, 3)$ with c =1. Apply linear, homogeneous and inhomogeneous kernels.
- **Solution:**
- The kernel is given by $k(x, y) = (x^T y)^q$
- If q = 2 the it is called homogeneous or quadratic kernel
- $k(x, y) = \left(\underline{\begin{pmatrix}1\\2\end{pmatrix}^T (2,3)}\right)^2 = 8^2 = 64$

Types of Kernel in Support Vector Machine

- Consider two data points $x = \binom{1}{2}$ and $y = (2, 3)$ with c =1. Apply linear, homogeneous and inhomogeneous kernels.
- **Solution:**
- The kernel is given by $k(x, y) = (x^T y)^q$
- If q = 2 and c =1 the it is called inhomogeneous kernel
- $k(x, y) = (\underline{c} + x^T y)^q = \left(1 + \binom{1}{2}^T (2,3)\right)^2 = (1 + 8)^2 = 81$

Types of Kernel in Support Vector Machine

Gaussian Kernel

- Radial Basis Functions (RBFs) or Gaussian kernels are extremely useful in SVM.
- The RBF function is shown as below:

$$k(x, y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

- Here, y is an important parameter. If y is small, then the RBF is similar to linear SVM and if y is large, then the kernel is influenced by more support vectors.
- The RBF performs the dot product in R^{∞}, and therefore, it is highly effective in separating the classes and is often used.

Types of Kernel in Support Vector Machine

- Consider two data points x = (1, 2) and y = (2, 3) with $\sigma = 1$.
- Apply RBF kernel and find the value of RBF kernel for these points.

$$k(x,y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

- Substitute the value of x and y in RBF kernel.
- The squared distance between the points (1, 2) and (2, 3) is given as:

$$(1 - 2)^2 + (2 - 3)^2 = 2$$

- If $\sigma = 1$, then $k(x,y) = e^{\left\{-\frac{2}{2}\right\}} = e^{-1} = 0.3679.$

Types of Kernel in Support Vector Machine

Sigmoid Kernel

- The sigmoid kernel is given as:

$$\underline{k(x_i, y_i)} = \tanh(k x_i y_j - \sigma)$$

3.4 Sigmoid Kernel

The sigmoid kernel is used for data that is not linearly separable, but it has a sigmoid-shaped decision boundary. We can use it as the proxy for neural networks. The equation for the sigmoid kernel is:

$$f(x1, x2) = \tanh(\alpha x^T y + x)$$

where α is a scaling factor, *x* is a constant, and tanh is the hyperbolic tangent function.
It is just taking your input, mapping them to a value of 0 and 1 so that they can be separated by a simple straight line.

Kernel Trick in Support Vector Machine

- Kernel Trick for 2nd degree Polynomial Mapping $\emptyset(x, y) = (x^2, \sqrt{2}xy, y^2)$

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = \begin{pmatrix} a_1{}^2 \\ \sqrt{2}\, a_1 a_2 \\ a_2{}^2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1{}^2 \\ \sqrt{2}\, b_1 b_2 \\ b_2{}^2 \end{pmatrix} = a_1{}^2 b_1{}^2 + 2a_1 b_1 a_2 b_2 + a_2{}^2 b_2{}^2$$

$$= (a_1 b_1 + a_2 b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 = \left(\mathbf{a}^T \cdot \mathbf{b}\right)^2$$

The Polynomial Kernel

- ✔ $K(x_1,x_2) = < x_1, x_2 >^2$
  - $x_1 = (x_{11}, x_{12})$
  - $x_2 = (x_{21}, x_{22})$
- ✔ $< x_1, x_2 > = (x_{11}x_{21} + x_{12}x_{22})$
- ✔ $< x_1, x_2 >^2 = (x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11} x_{12} x_{21} x_{22})$
- ✔ $\Phi(x_1) = (x_{11}^2, x_{12}^2, \sqrt{2}x_{11} x_{12})$
- ✔ $\Phi(x_2) = (x_{21}^2, x_{22}^2, \sqrt{2}x_{21} x_{22})$
- ✔ $K(x_1,x_2) = < \Phi(x_1), \Phi(x_2) >$
- ✔ Variation - $K(x_1,x_2) = (< x_1, x_2 > + 1)^2$(Inhomogeneous)

Using a Different Kernel in the Dual Optimization Problem

✔ Maximize over α

– $W(\alpha) = \Sigma_i \alpha_i - 1/2\ \Sigma_{i,j}\ \alpha_i\ \alpha_j\ y_i\ y_j <x_i, x_j>$

✔ Subject to

– $\alpha_i \geq 0$

– $\Sigma_i\ \alpha_i\ y_i = 0$

✔ $K(x_1,x_2) = < x_1, x_2 >^2$

– $x_1 = (x_{11}, x_{12})$

– $x_2 = (x_{21}, x_{22})$

✔ $< x_1, x_2 > = (x_{11}x_{21} + x_{12}x_{22})$

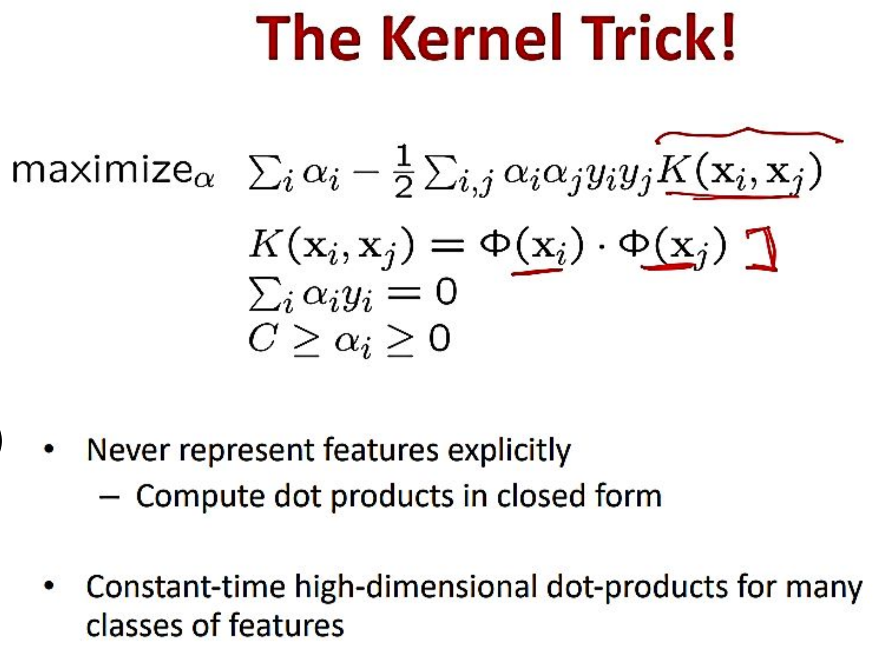
✔ $< x_1, x_2 >^2 = (x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}\ x_{12}\ x_{21}\ x_{22})$

✔ $\Phi(x_1) = (x_{11}^2, x_{12}^2, \sqrt{2}x_{11}\ x_{12})$

✔ $\Phi(x_2) = (x_{21}^2, x_{22}^2, \sqrt{2}x_{21}\ x_{22})$

✔ $K(x_1,x_2) = < \Phi(x_1), \Phi(x_2) >$

✔ Variation - $K(x_1,x_2) = (< x_1, x_2 > + 1)^2$(Inhomogeneous)



The Kernel Trick!

maximize$_\alpha$ $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \overbrace{\underline{K(\mathbf{x}_i,\mathbf{x}_j)}}$

$K(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\underline{\mathbf{x}_i}) \cdot \Phi(\underline{\mathbf{x}_j})$

$\sum_i \alpha_i y_i = 0$

$C \geq \alpha_i \geq 0$

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features

✔ For example, using the polynomial kernel with d = 4 (including lower-order terms).

$(<x_i, x_j> + 1)^4$

Quadratic Basis Functions

$$
\mathbf{\Phi}(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}
$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

Number of terms (assuming m input dimensions)

Quadratic Dot Products

$$\mathbf{\Phi(a)} \bullet \mathbf{\Phi(b)} = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

$$\}\ 1$$

$$+$$

$$\}\ \sum_{i=1}^{m} 2a_ib_i$$

$$+$$

$$\}\ \sum_{i=1}^{m} a_i^2b_i^2$$

$$+$$

$$\}\ \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Quadratic Dot Products

$$\boldsymbol{\Phi}(\mathbf{a}) \bullet \boldsymbol{\Phi}(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

Types of Support Vector Machine

- ✔ Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:
- ✔ **Linear SVM**
- ✔ **Non-Linear SVM**

Linear SVM

- ✔ **Linear SVMs** use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
- ✔ Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).

SVM - Linearly Separable

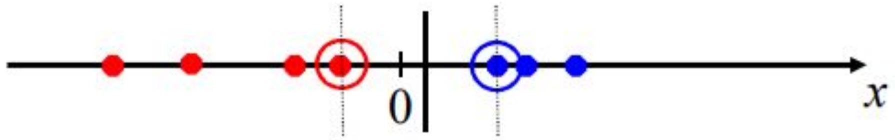
- A separating hyperplane can be written as
  - $\mathbf{W} \bullet \mathbf{X} + b = 0$
  - where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as
  - $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:
  - H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and
  - H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

Non-Linear SVM

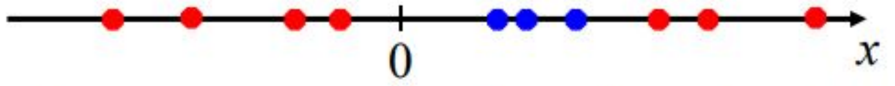
- ✔ **Non-Linear SVM** can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.
- ✔ The kernel function computes the similarity between data points, allowing SVM to capture complex patterns and nonlinear relationships between features. This enables nonlinear SVM to handle intricate data distributions, such as curved or circular decision boundaries. By leveraging the kernel trick, nonlinear SVM provides a powerful tool for solving classification problems where linear separation is insufficient, extending its applicability to a wide range of real-world scenarios.

Non-linear SVMs

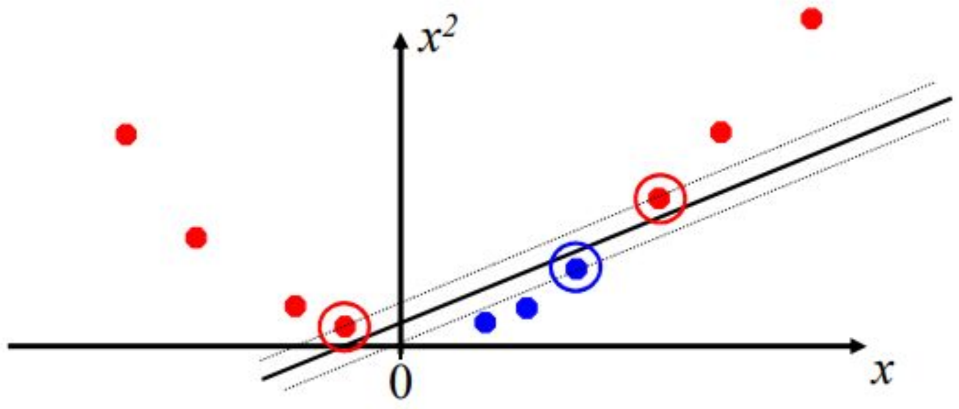
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

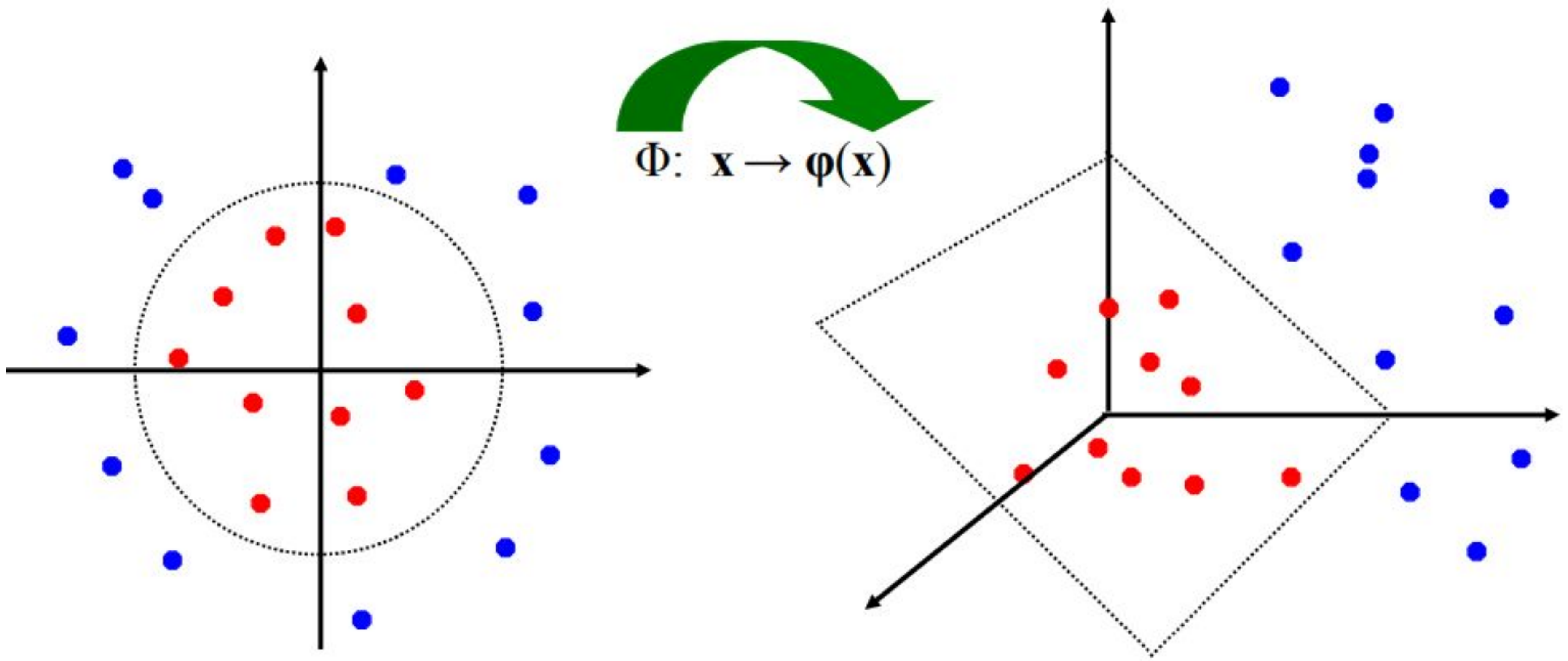


- How about… mapping data to a higher-dimensional space



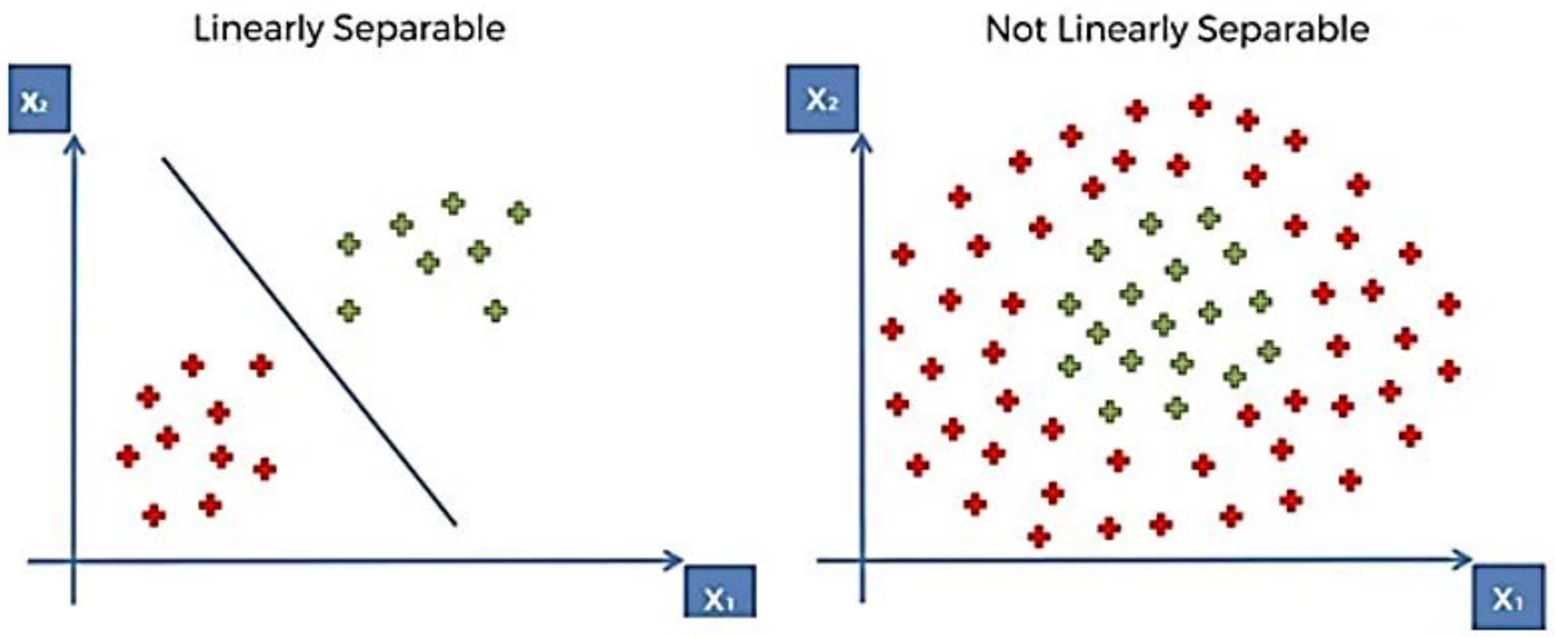
Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



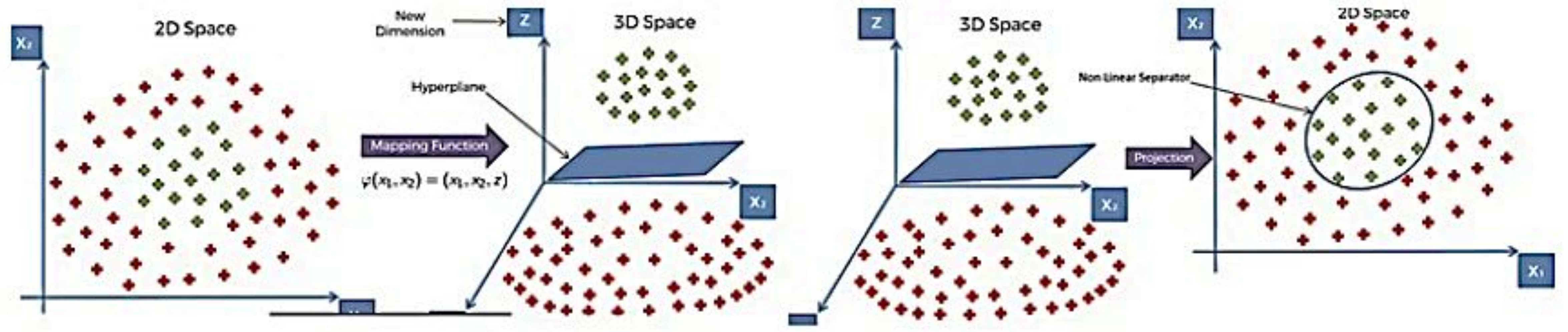
Non-Linear SVM

- ✔ Select any kernel function like RBF(gaussian), polynomial, sigmoid and etc instead of a linear and 1-degree model. Kernel function takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space and this is called kernel function.
- ✔ Non-linear transformation is to make a dataset higher-dimensional space **(Mapping a higher dimension)**. And it is also the fundamental of a non-linear system. The below graph reveals a non-linear dataset and how it can not be used Linear kernel rather than the Gaussian kernel.



Non-Linear SVM

- ✔ In geometry, a **hyperplane** is a subspace whose dimension is one less than that of its ambient space. If space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if space is 2-dimensional, its hyperplanes are the 1-dimensional lines. This notion can be used in any general space in which the concept of the dimension of a subspace is defined.



Mapping Function makes 2D to 3D and Projection returns the 3D to 2D

Thank You

Duality: Hard Margin Classifier

We will now change our Hard Margin Classifier's Objective Function from **Primal Problem** to **Dual Problem** using **Lagrange Multiplier** and **KKT Conditions**.

Recall the optimization problem,

$$\textbf{Objective Function}: \min_{\beta,b}\left\{\frac{||\beta^2||}{2}\right\}$$

$$\textbf{s.t Linear Constraint}: y_i\left(\beta^T x_i + b\right) \geq 1 \,, \forall x_i \in D$$

The constraint can be redefined as following (This is required for the 2^{nd} KKT Condition $g_i\left(x\right) \leq 0$),

$$\textbf{s.t Linear Constraint}: 1 - y_i\left(\beta^T x_i + b\right) \leq 0 \,, \forall x_i \in D$$

The **Lagrangian** can defined as following,

$$\begin{aligned}\min L &= \frac{||\beta^2||}{2} + \sum_{i=1}^{n}\alpha_i(1 - y_i(\beta^T x_i + b)) \\ &= \frac{||\beta^2||}{2} - \sum_{i=1}^{n}\alpha_i(y_i(\beta^T x_i + b) - 1)\end{aligned}$$

Taking derivatives w.r.t β and **b**,

$$\delta_\beta L = \beta - \sum_{i=1}^{n}\alpha_i y_i x_i = 0$$

$$\beta = \sum_{i=1}^{n}\alpha_i y_i x_i$$

$$\text{and } \delta_b L = \sum_{i=1}^{n}\alpha_i y_i = 0$$

Plugging these we get the **Dual Lagrangian Objective Function**,

$$\begin{aligned}L_{dual} &= \frac{||\beta^2||}{2} - \sum_{i=1}^{n}\alpha_i(y_i(\beta^T x_i + b) - 1) \\ &= \frac{1}{2}\beta^T\beta - \sum_{i=1}^{n}\alpha_i y_i \beta^T x_i - \sum_{i=1}^{n}\alpha_i y_i b + \sum_{i=1}^{n}\alpha_i \\ &= \frac{1}{2}\beta^T\beta - \beta^T\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right) - b\left(\sum_{i=1}^{n}\alpha_i y_i\right) + \sum_{i=1}^{n}\alpha_i \\ &= \frac{1}{2}\beta^T\beta - \beta^T\left(\beta\right) - b\left(0\right) + \sum_{i=1}^{n}\alpha_i \\ &= -\frac{1}{2}\beta^T\beta + \sum_{i=1}^{n}\alpha_i \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\end{aligned}$$