(4)

Machine Learning for Data Science

M. H. Rahman

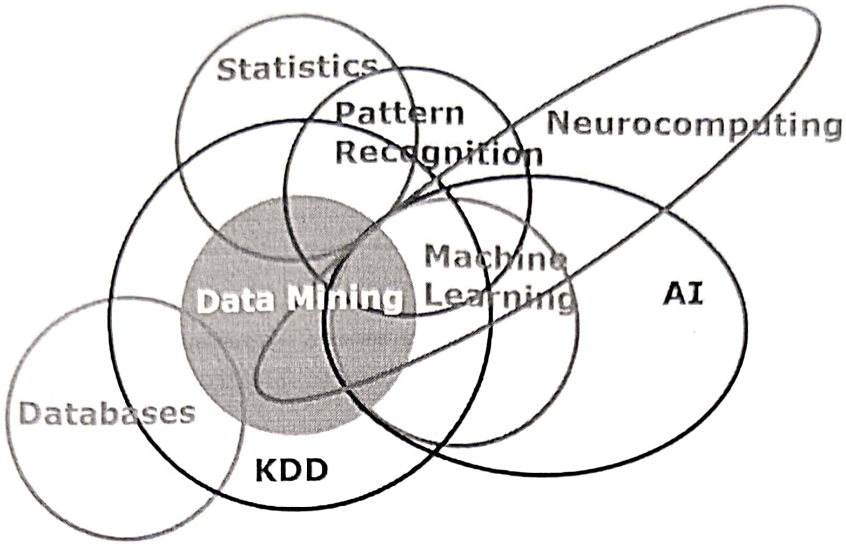
Associate Professor
Department of Statistics and Data Science
Jahangirnagar University
Bangladesh
E-mail: habib.drj@juniv.edu

Twenty-Fifty

Machine Learning for Data Science

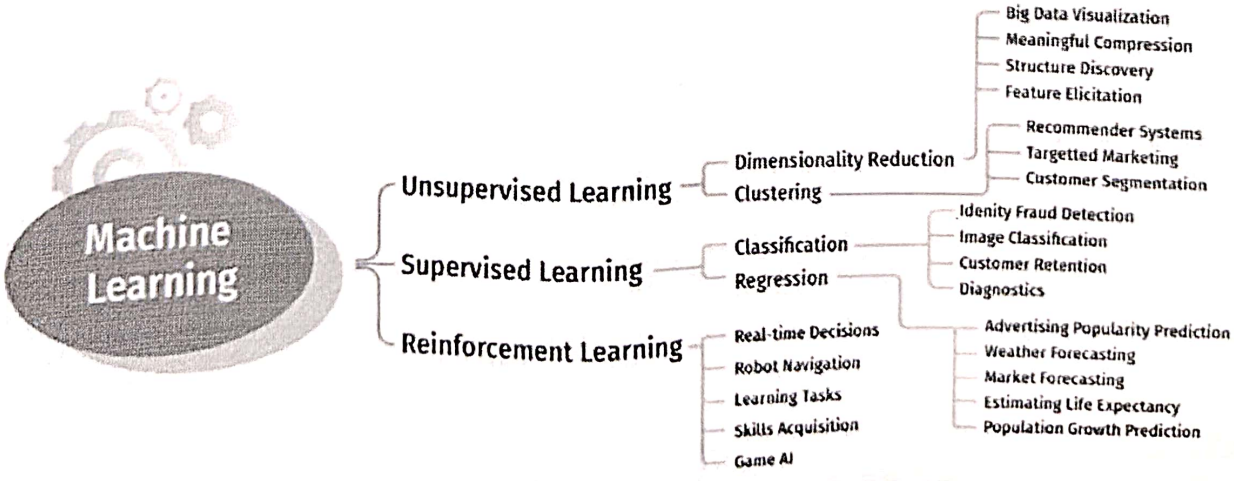
Support Vector Machine (SVM)
or
Support Vector Method (SVM)

Machine Learning for Data Science



The figure is collected from the internet.

Machine Learning for Data Science



The figure is collected from the internet.

Support Vector Machine

A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space.

Vapnik et al. developed SVMs in the 1990s. They published their work in a 1995 paper titled "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing" (Vapnik et al.; 1996).

Support Vector Machine

Support Vector Machine

SVMs are commonly used within classification problems. They distinguish between two classes by finding the optimal hyperplane that maximizes the margin between the closest data points of opposite classes.

The number of features in the input data determines if the hyperplane is a line in a 2-D space or a plane in a n-dimensional space.

Since multiple hyperplanes can be found to differentiate classes, maximizing the margin between points enables the algorithm to find the best decision boundary between classes.

This, in turn, enables it to generalize well to new data and make accurate classification predictions. The lines that are adjacent to the optimal hyperplane are known as support vectors as these vectors run through the data points that determine the maximal margin.

Support Vector Machine

The SVM algorithm is widely used in machine learning as it can handle both linear and nonlinear classification tasks.

However, when the data is not linearly separable, kernel functions are used to transform the data higher-dimensional space to enable linear separation.

This application of kernel functions can be known as the "kernel trick", and the choice of the kernel function, such as linear kernels, polynomial kernels, radial basis function (RBF) kernels, or sigmoid kernels, depends on data characteristics and the specific use case.

Support Vector Machine

Types of SVM classifiers

- Linear SVM
  Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Nonlinear SVM
  Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Types of Kernel

There are a number of different kernel types that can be applied to classify data. Some popular kernel functions include:

- Linear kernel: $K(x, u) = x^T.u$
- Polynomial function: $k(x, u) = (ax^T u + c)^q$, $q > 0$
- Hyperbolic tangent: $K(x, u) = tanh(\beta x^T u + \gamma)$
- Gaussian radial basis function (RBF): $K(x, u) = exp\left(-\frac{||x-u||^2}{\sigma^2}\right)$
- Laplacian radial basis function (RBF): $K(x, u) = exp\left(-\frac{||x-u||}{\sigma}\right)$
- Randomized blocks analysis of variance (ANOVA RB) kernel: $K(x, u) = \sum_{i=1}^{n} exp[-\sigma(x^k - u^k)^2]^d$
- Linear spline kernel in 1D:

$$K(x, u) = 1 + x.u.min(x, u) - \frac{x+u}{2}[min(x, u)^2 + \frac{1}{3}min(x, u)^3]$$

Support Vector Machine Terminology

- Hyperplane: The hyperplane is the decision boundary used to separate data points of different classes in a feature space. For linear classification, this is a linear equation represented as

$$wx + b = 0$$

- Support Vectors: Support vectors are the closest data points to the hyperplane. These points are critical in determining the hyperplane and the margin in Support Vector Machine (SVM).
- Margin: The margin refers to the distance between the support vector and the hyperplane. The primary goal of the SVM algorithm is to maximize this margin, as a wider margin typically results in better classification performance.

Support Vector Machine

Support Vector Machine Terminology...

- Kernel: The kernel is a mathematical function used in SVM to map input data into a higher-dimensional feature space. This allows the SVM to find a hyperplane in cases where data points are not linearly separable in the original space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.
- Hard Margin: A hard margin refers to the maximum-margin hyperplane that perfectly separates the data points of different classes without any misclassifications.
- Soft Margin: When data contains outliers or is not perfectly separable, SVM uses the soft margin technique. This method introduces a slack variable for each data point to allow some misclassifications while balancing between maximizing the margin and minimizing violations.

Support Vector Machine

Support Vector Machine Terminology...

- C: The C parameter in SVM is a regularization term that balances margin maximization and the penalty for misclassifications. A higher C value imposes a stricter penalty for margin violations, leading to a smaller margin but fewer misclassifications.
- Hinge Loss: The hinge loss is a common loss function in SVMs. It penalizes misclassified points or margin violations and is often combined with a regularization term in the objective function.
- Dual Problem: The dual problem in SVM involves solving for the Lagrange multipliers associated with the support vectors. This formulation allows for the use of the kernel trick and facilitates more efficient computation.

Support Vector Machine

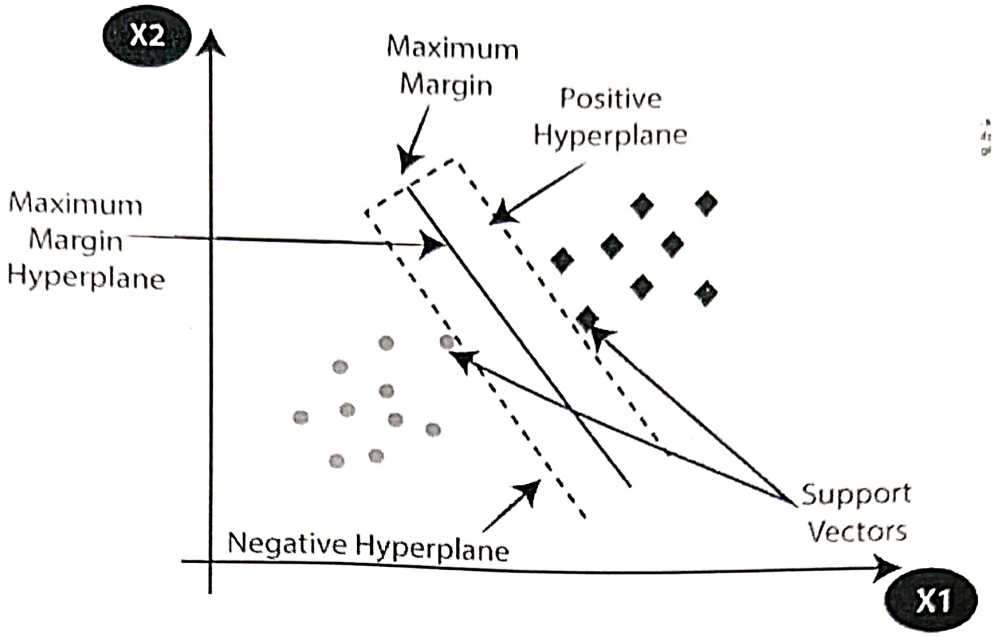


Figure 1: SVM

Support Vector Machine

The SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane.

SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors.

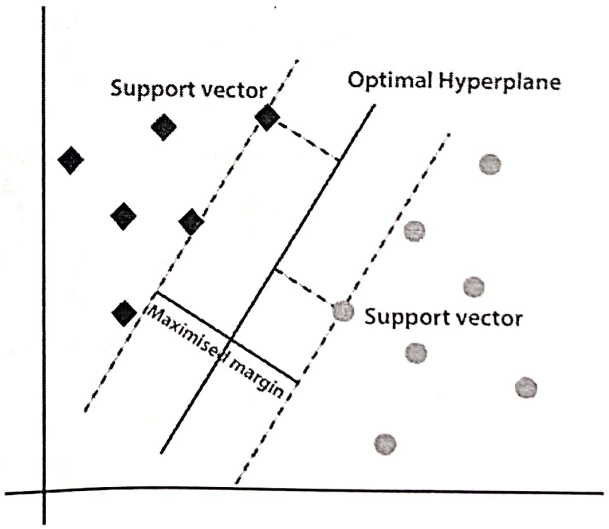
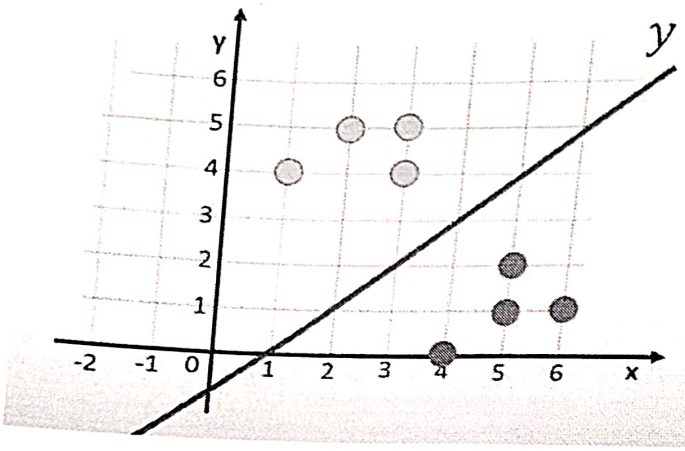


Figure 2: Maximised SVM

The distance between the vectors and the hyperplane is called as margin. And, the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

Support Vector Machine



$Ax + By + C =$

Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

Figure 3: Line for $y = mx + b$

Support Vector Machine

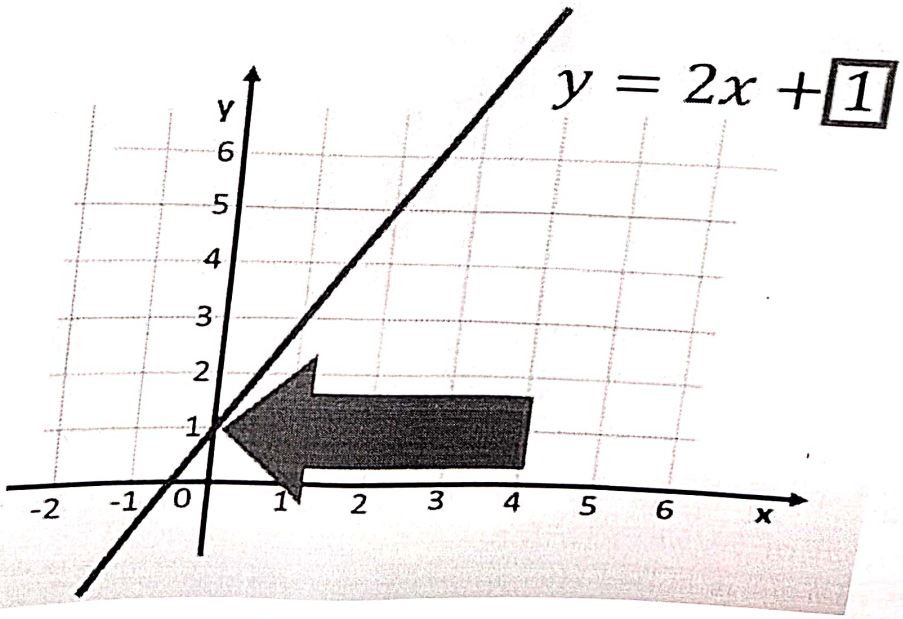


Figure 4: Line indicates the intercept of $y = mx + b$

Support Vector Machine

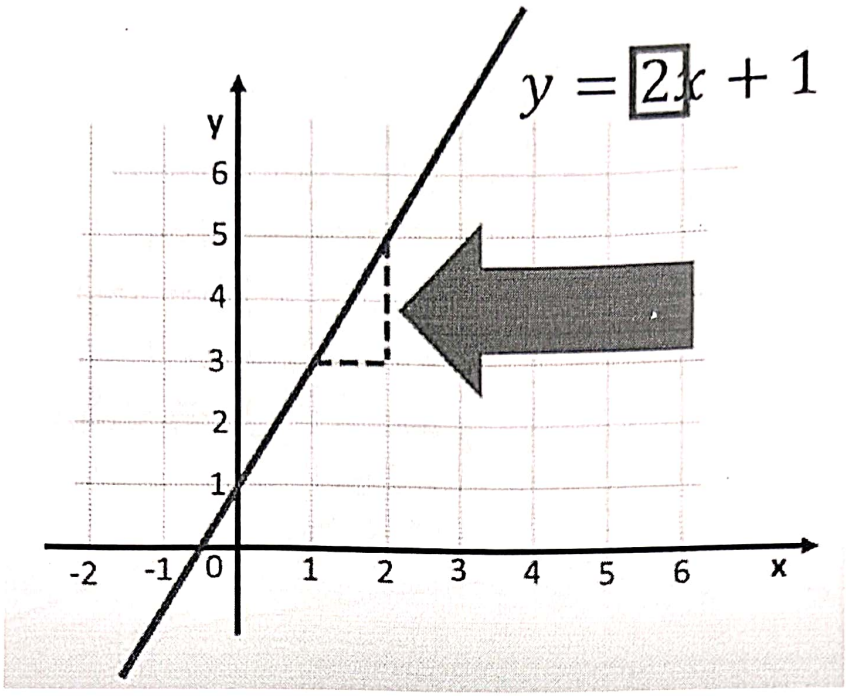


Figure 5: Line indicates the slope of $y = mx + b$

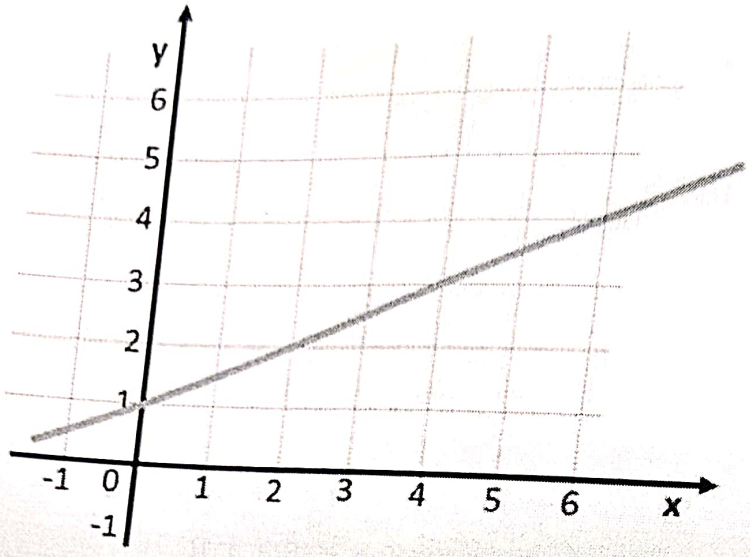
Support Vector Machine

$$\max \frac{2}{\|w\|} \text{ such that } w^T x_i + b \begin{array}{l} \geq 1 \text{ if } Y_i = +1 \\ \leq -1 \text{ if } Y_i = -1 \end{array}$$

$$y = mx + c$$

$$w^T x + b = 0$$

Support Vector Machine

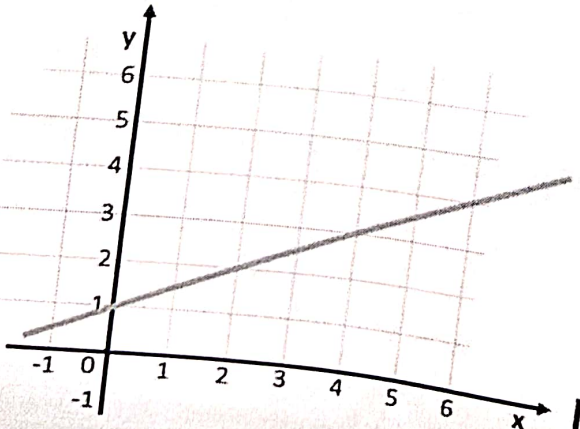


$$y = mx + b$$

$$y = 0.5x + \boxed{1}$$

Figure 6: Line for $y = 0.5x + 1$

Support Vector Machine



$$y = 0.5x + 1$$

$$y = mx + b$$
$$Ax + By + C = 0$$

$$-0.5x + y - 1 = 0$$

$$-2x + 4y - 4 = 0 \quad \times (-1)$$

$$\boxed{2x - 4y + 4 = 0}$$

Figure 7: Line for $Ax + By + C = 0$

Support Vector Machine

Effect of C at $Ax + By + C = 0$

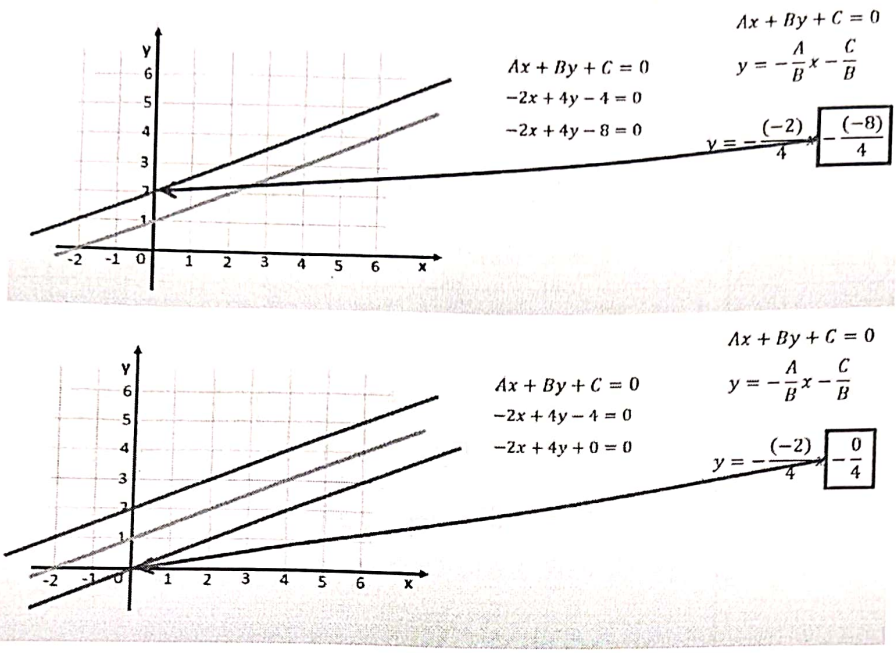


Figure 8: Effect of C at $Ax + By + C = 0$

Support Vector Machine

Effect of B at $Ax + By + C = 0$

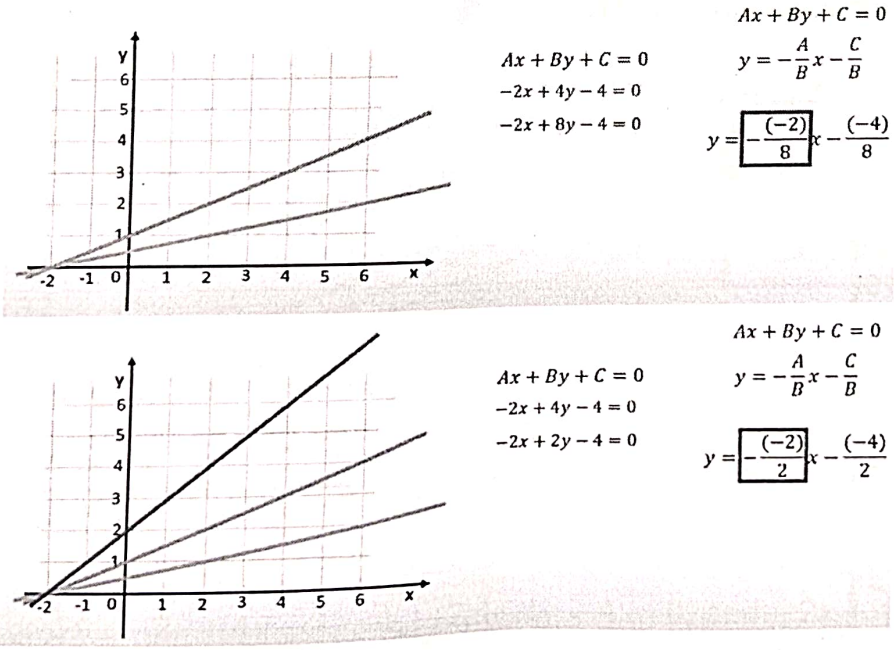
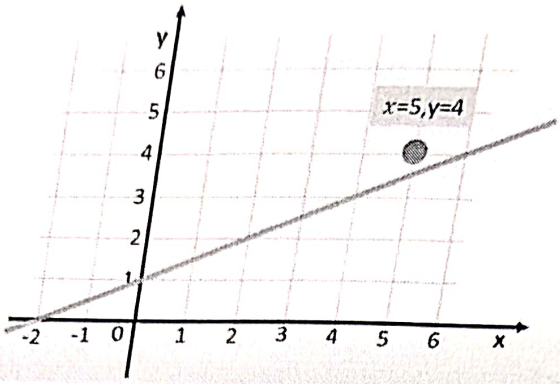
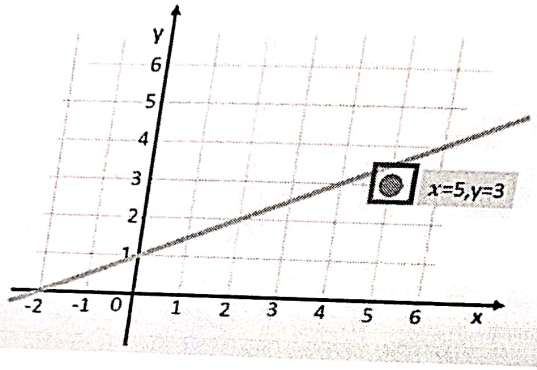


Figure 9: Effect of B at $Ax + By + C = 0$

Support Vector Machine



$$Ax + By + C = 0$$
$$-2x + 4y - 4 = 0$$
$$-2 \cdot 5 + 4 \cdot 4 - 4 = \boxed{2}$$

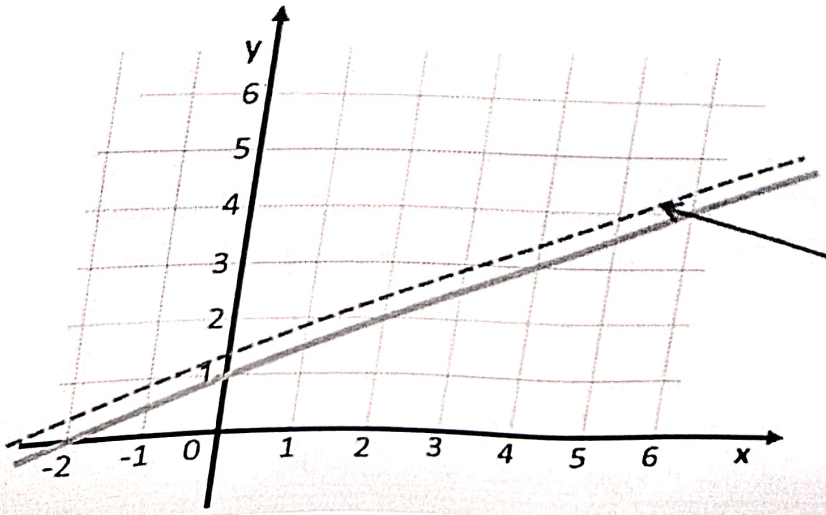


$$Ax + By + C = 0$$
$$-2x + 4y - 4 = 0$$
$$-2 \cdot 5 + 4 \cdot 3 - 4 = \boxed{-2}$$

Figure 10: Consider the points $(x = 5, y = 4)$ and $(x = 5, y = 3)$.

Support Vector Machine

Consider $Ax + By + C = 1$



$$Ax + By + C = 0$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = 1$$
$$4y = 2x + 5$$
$$y = 0.5x + \boxed{1.25}$$

Figure 11: Consider $-2x + 4y - 4 = 1$

Support Vector Machine

Consider $Ax + By + C = -1$

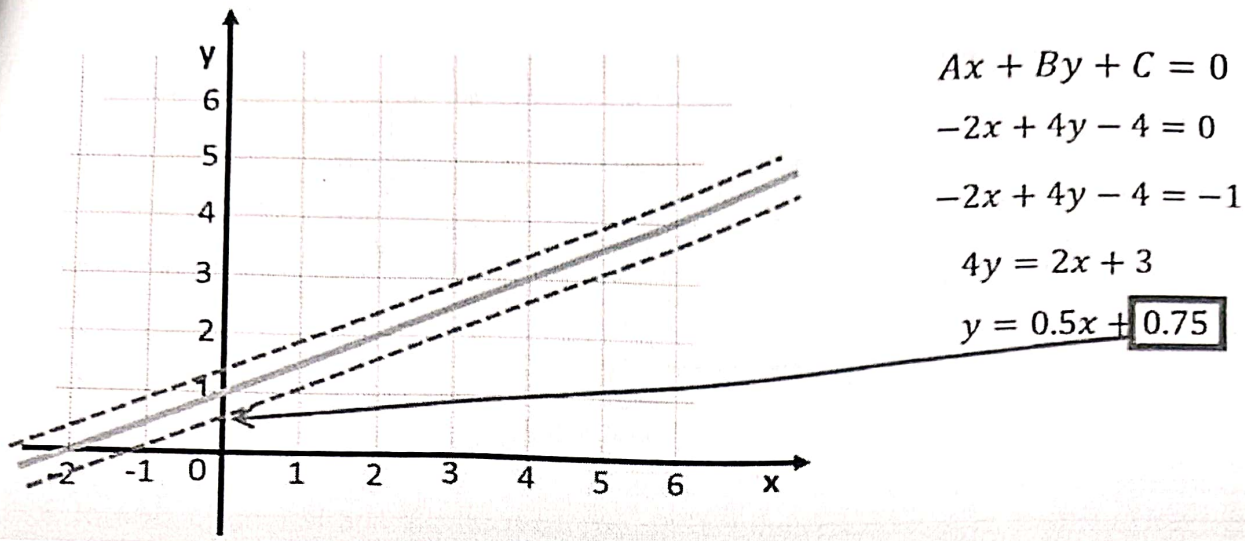


Figure 12: Consider $-2x + 4y - 4 = -1$

Support Vector Machine

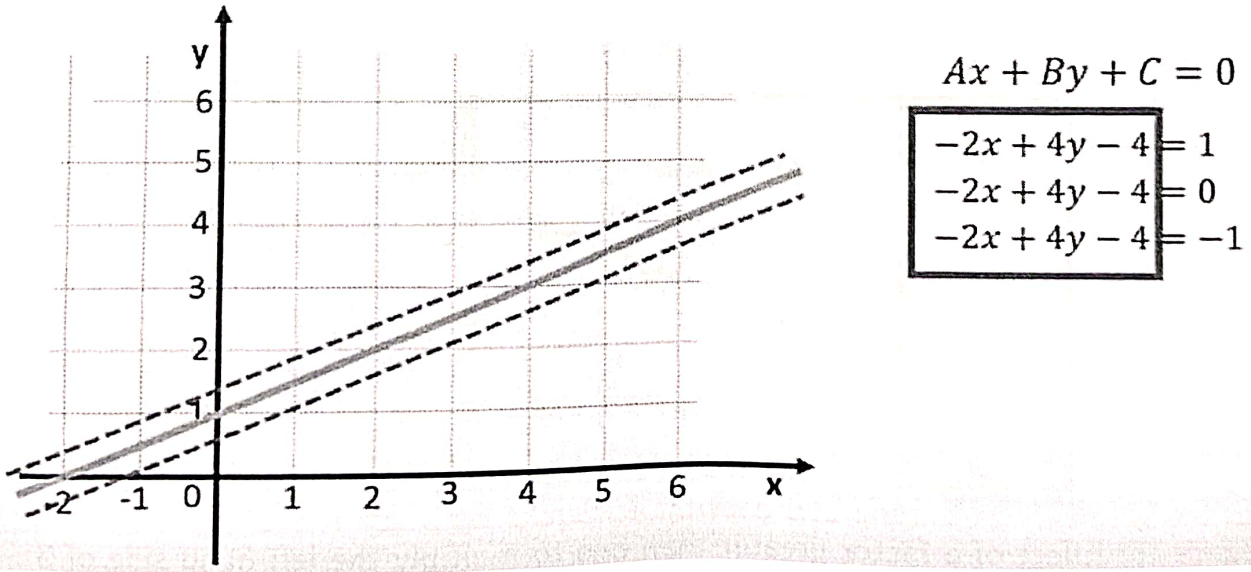
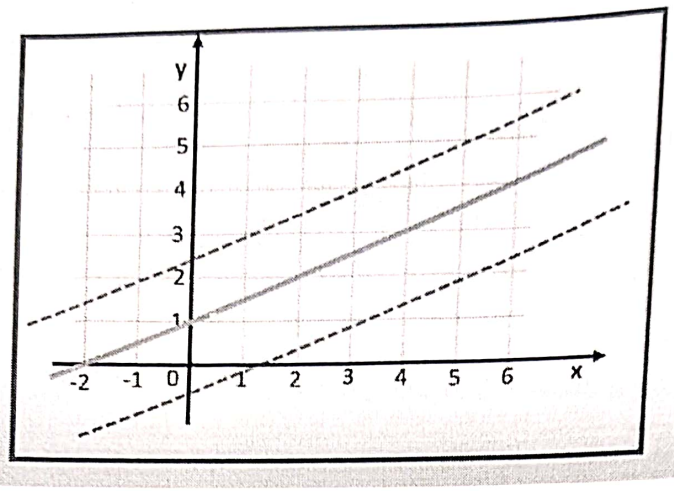


Figure 13: Consider $-2x + 4y - 4 = 1$, $-2x + 4y - 4 = 0$, and $-2x + 4y - 4 = -1$

Support Vector Machine



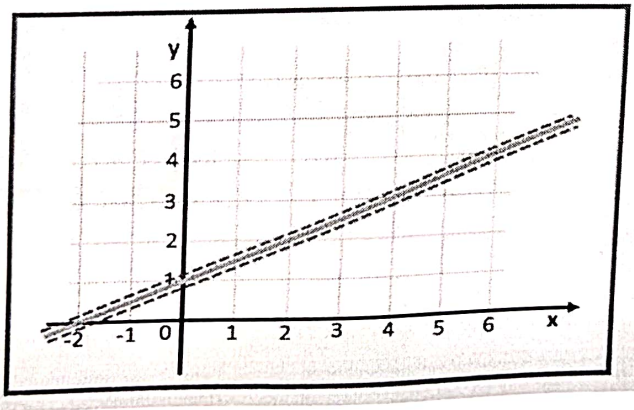
$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

Figure 14: Effect of a factor smaller than one to multiply the left-hand side of 3 equations.

If we would multiply the terms on the left-hand side with these three equations with a factor smaller than one the green lines would move away from the original line which stays in the same position.

Support Vector Machine



$$Ax + By + C = 0$$

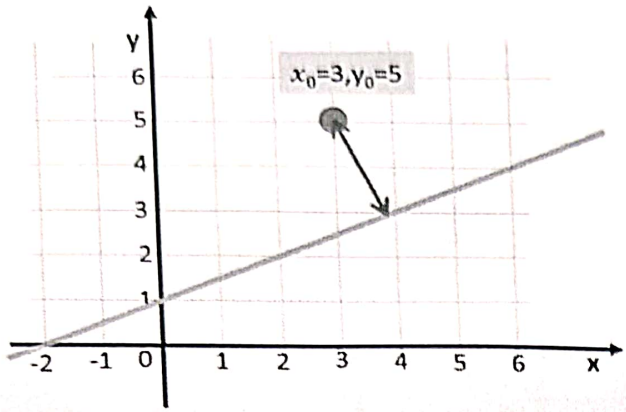
$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

Figure 15: Effect of a factor greater than one to multiply the left-hand side of 3 equations.

If we would multiply the terms on the left-hand side with these three equations with a factor greater than one the green lines would move away from the original line which stays in the same position.
This method is used in support vector machines to increase or decrease the so-called margin.

Support Vector Machine

To calculate the distance between a data point and a line the shortest distance between a data point and a line can be calculated by the following formula.



$$Ax + By + C = 0$$
$$-2x + 4y - 4 = 0$$

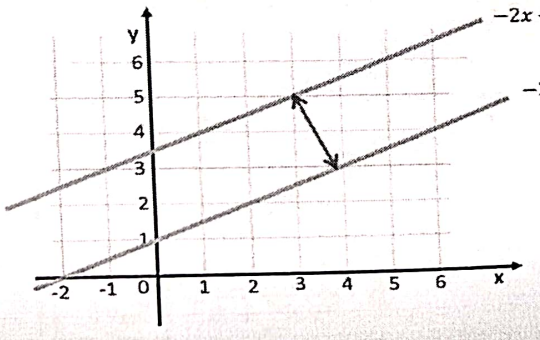
$$d = \frac{|Ax_0 + By_o + C|}{\sqrt{A^2 + B^2}}$$

$$d = \frac{|(-2)\cdot 3 + 4\cdot 5 + (-4)|}{\sqrt{(-2)^2 + 4^2}} = \frac{10}{\sqrt{20}} = \boxed{2.236}$$

Figure 16: Distance between a data point and a line

Support Vector Machine

Similarly, the distance between two parallel lines can be calculated by the formula.



$$Ax + By + C = 0$$
$$-2x + 4y - 14 = 0$$
$$-2x + 4y - 4 = 0$$

$$d = \frac{|C_2 - C_1|}{\sqrt{A^2 + B^2}}$$

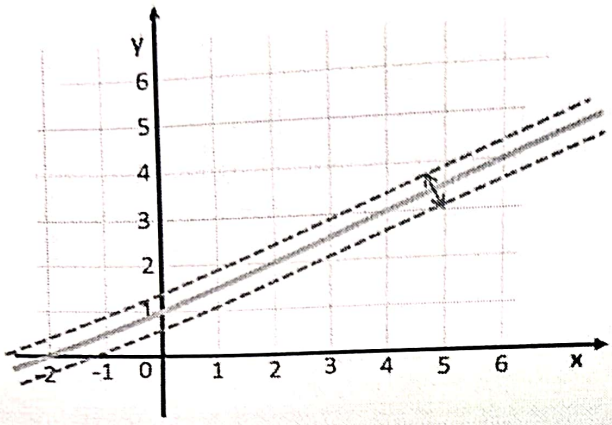
$$d = \frac{|(-4) - (-14)|}{\sqrt{(-2)^2 + 4^2}} = \frac{10}{\sqrt{20}} = \boxed{2.236}$$

Figure 17: Distance between two lines

If you plug in the coefficients of the equations of the two parallel lines and the constants, we see that the shortest distance between these two lines is about 2.24.

Support Vector Machine

Remember that we previously added a one and a negative one on the right hand side to obtain the two green lines. The distance between these two green lines is calculated like this:



$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

$$d = \frac{|C_2 - C_1|}{\sqrt{A^2 + B^2}}$$

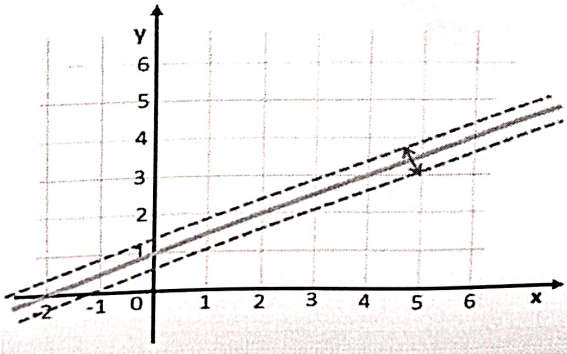
$$d = \frac{|(-3) - (-5)|}{\sqrt{(-2)^2 + 4^2}} = \frac{\boxed{2}}{\sqrt{20}} = 0.447$$

Figure 18: Distance between two lines $-2x + 4y - 4 = 1$ and $-2x + 4y - 4 = -1$

We have a two in the numerator because the absolute difference between the constants is always two for such lines.

Support Vector Machine

For such lines, we can simplify the equation to this or like this



$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

$$d = \frac{2}{\sqrt{A^2 + B^2}} = \boxed{\frac{2}{||w||}}$$

Figure 19: Distance between two lines $-2x + 4y - 4 = 1$ and $-2x + 4y - 4 = -1$ as maximum margin

$$w = [-2, 4]$$

$$||w|| = \sqrt{(-2)^2 + (4)^2} = \sqrt{20}$$

Support Vector Machine

We now look at the real data set where we will use a support vector machine to generate a line separating the two groups as well as possible. The yellow points belong to group "A" which for example could represent four individuals with a certain disease, whereas these green points belong to group "B".

Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

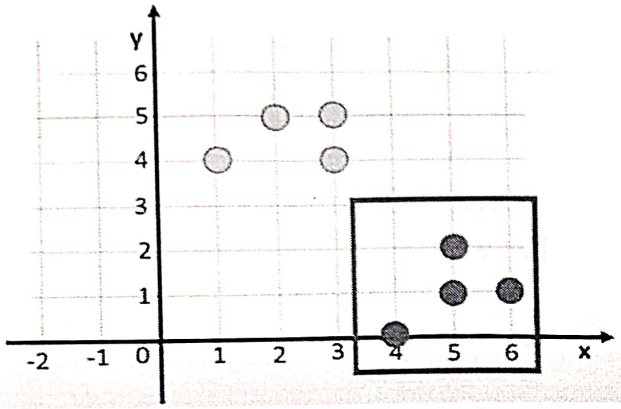


Figure 20: Plot for data points of two groups

Support Vector Machine

The following line is the best line to separate the two grooves based on the training data.

Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

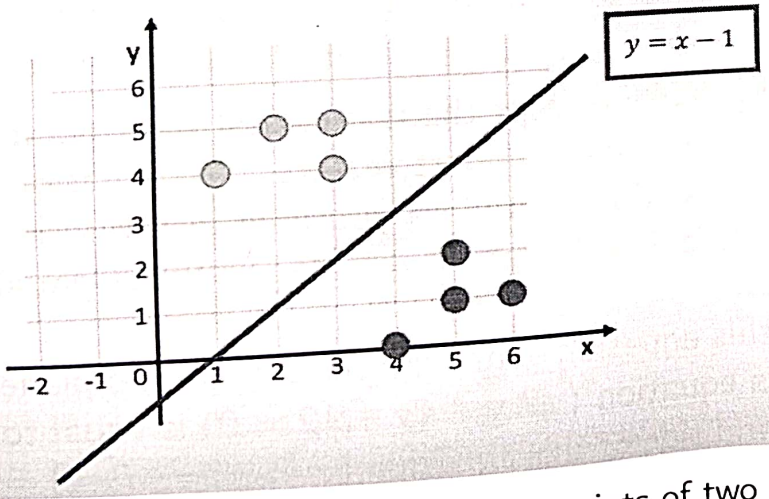
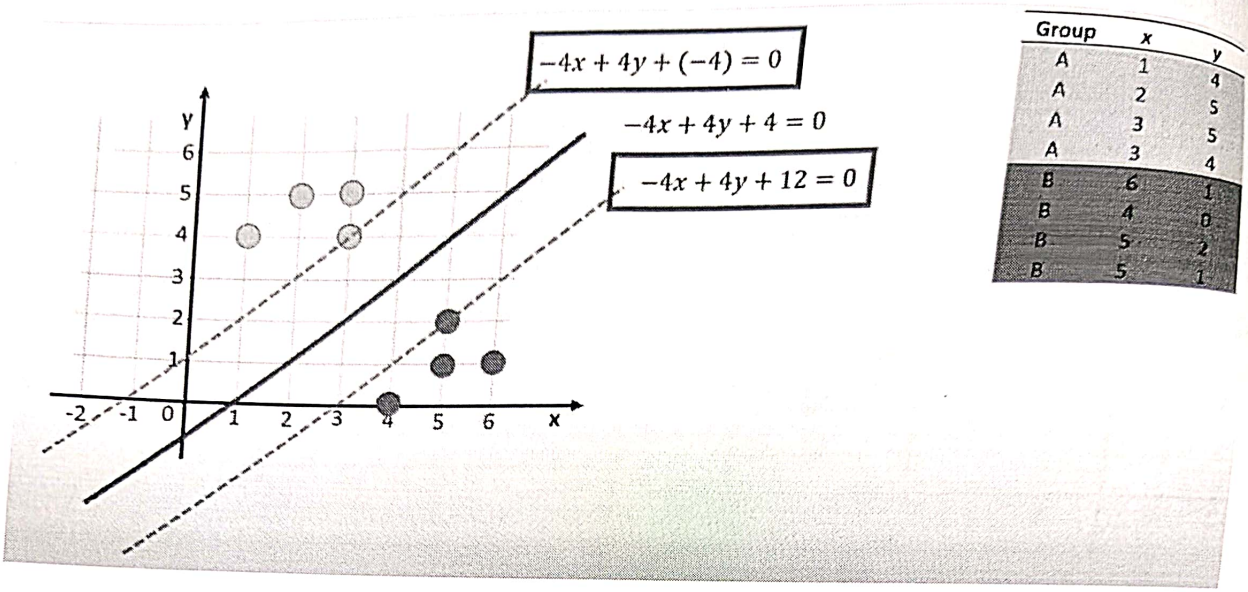


Figure 21: Plot for data points of two groups separated by line

$-x + y + 1 = 0$ that is, $-4x + 4y + 4 = 0$

Support Vector Machine



Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

Figure 22: Plot for data points of two groups separated by a line, positive and negative hyperplane

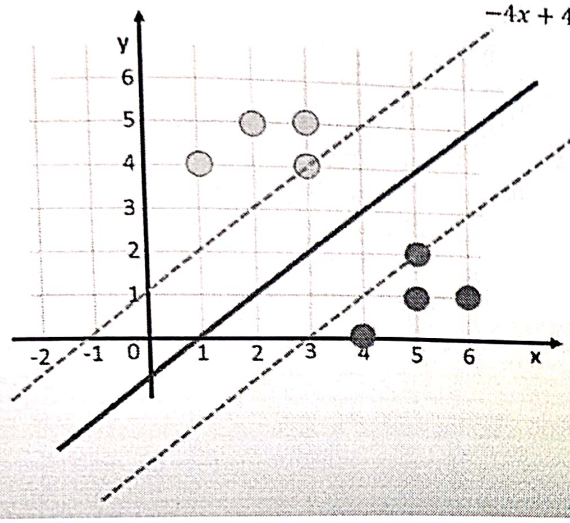
Support Vector Machine

We'll now draw two parallel lines that intercept the two closest data points to the hyperplane such data points are called support vectors. The equations of these two blue lines look like this. For example if you plug in the x and y coordinates of a point on this blue line the left-hand side should be equal to zero which is true in this case.

So can we somehow normalize this equation that represents the hyperplane so that the left-hand side of this equation $(-4x + 4y + 12 = 0)$ is equal to negative one and that the left-hand side of this equation $(-4x+4y+(-4) = 0)$ is equal to positive one given that all three equations have the same constant term on the left-hand side as the equation of the hyperplane

Support Vector Machine

Given that all three equations have the same constant term on the left-hand side as the equation of the hyperplane. In other words, can we find the value of k so that the left-hand side is equal to one or negative one. Let's plug in the x and y coordinates of this support vector like this.



$$-4x + 4y + (-4) = 0$$

$$-4x + 4y + 4 = 0$$

$$-4x + 4y + 12 = 0$$

Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

$$k(-4x + 4y + 4) = 1$$

$$k(-4 \cdot 3 + 4 \cdot 4 + 4) = 1$$

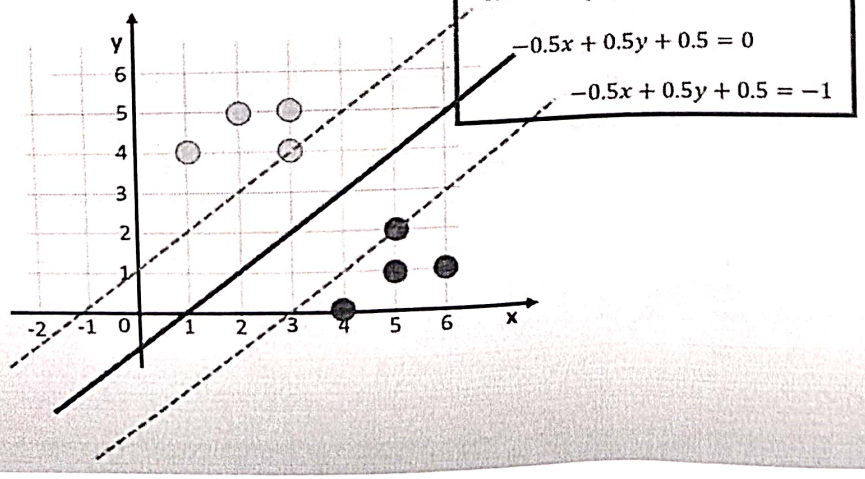
$$k(-12 + 16 + 4) = 1$$

$$k \cdot 8 = 1$$

$$\boxed{k = 1/8}$$

Support Vector Machine

If we solve this equation we see that we should multiply the terms on left-hand side by one over eight. After dividing the terms by eight the equations look like this which is the standard form of the equations in support vector machines where the right hand side is equal to positive one a negative one of the two blue lines.



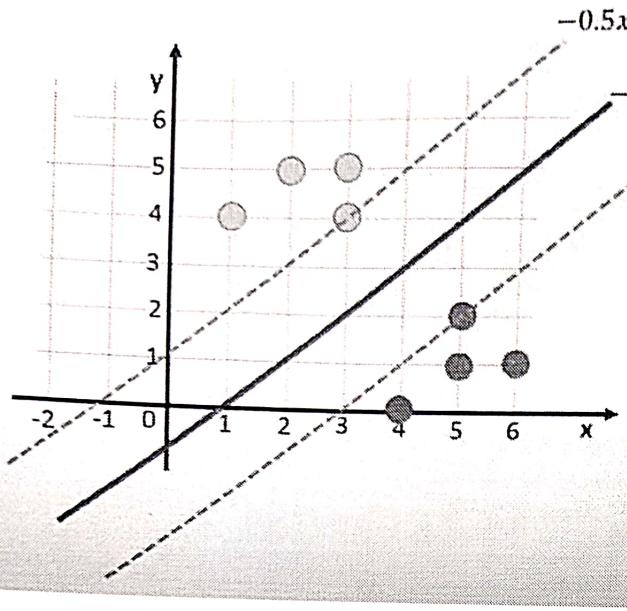
$$-0.5x + 0.5y + 0.5 = 1$$

$$-0.5x + 0.5y + 0.5 = 0$$

$$-0.5x + 0.5y + 0.5 = -1$$

Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

Support Vector Machine



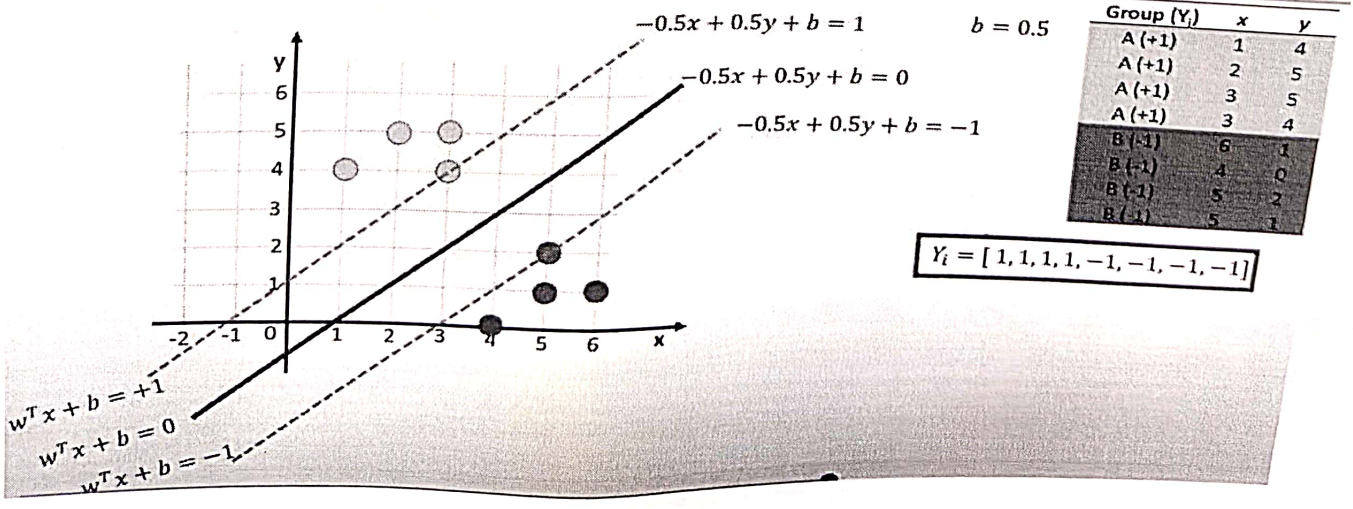
Group	x	y
A	1	4
A	2	5
A	3	5
A	3	4
B	6	1
B	4	0
B	5	2
B	5	1

$$w^T x + b = 0$$

$$[-0.5 \quad 0.5] \cdot \begin{bmatrix} x \\ y \end{bmatrix} + b = 0$$

The equation of the hyperplane is usually expressed like this in support vector machines this represents a vector that holds our coefficients and this is a vector that includes the variables x and y in our example T represents that we should transpose the vector w.

Support Vector Machine

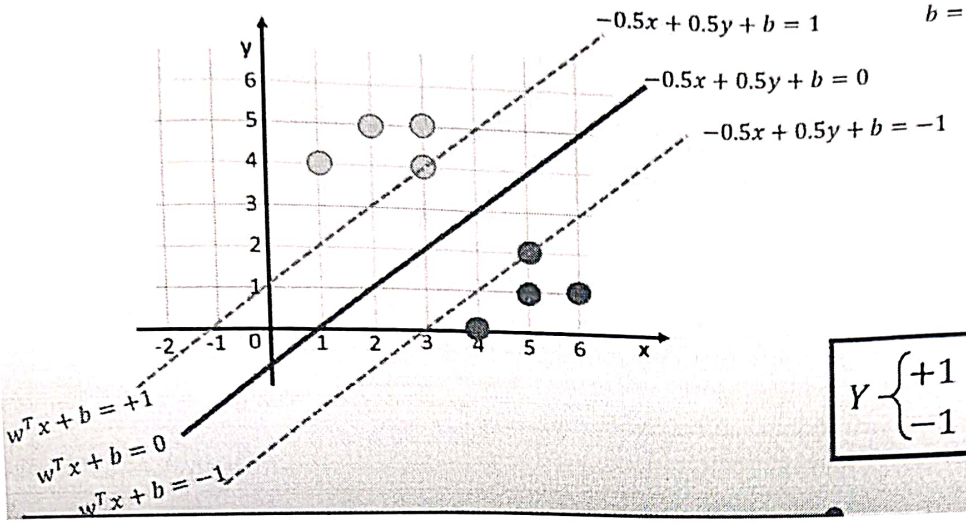


Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$$Y_i = [1, 1, 1, 1, -1, -1, -1, -1]$$

Support Vector Machine

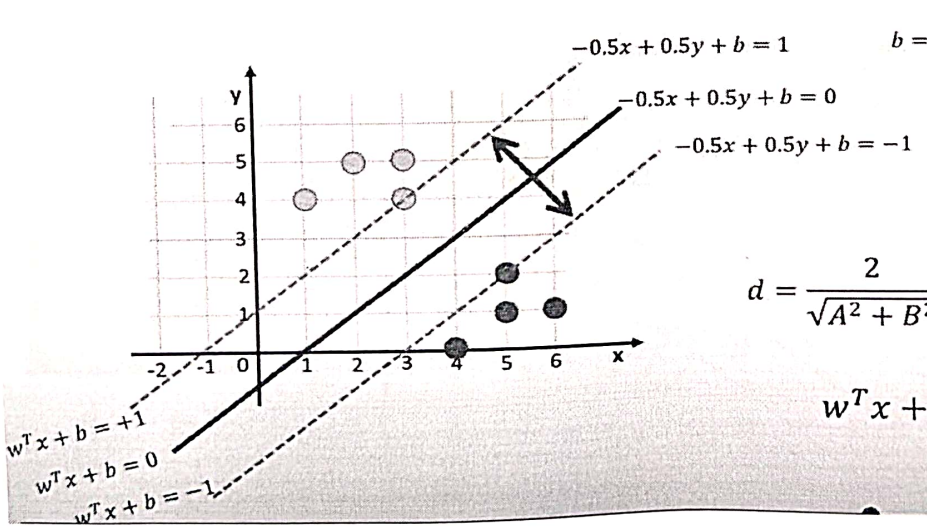
The classification in support vector machines is usually defined like this



Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$$Y \begin{cases} +1 \text{ if } w^T x + b \geq 0 \\ -1 \text{ if } w^T x + b < 0 \end{cases}$$

Support Vector Machine

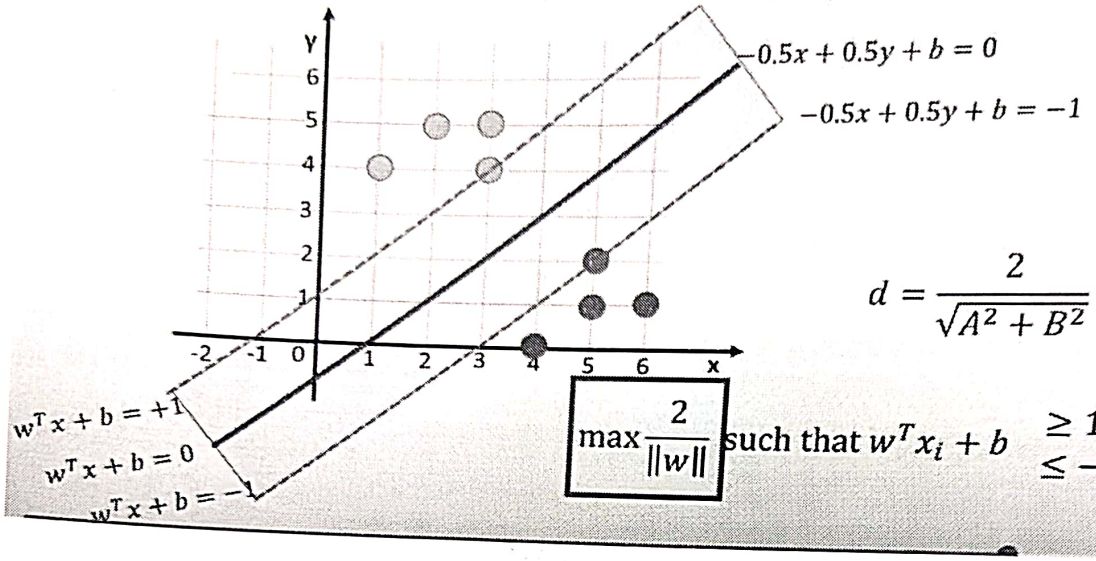


Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$$d = \frac{2}{\sqrt{A^2 + B^2}} = \boxed{\frac{2}{\|w\|}}$$

$$w^T x + b = 0$$

Support Vector Machine

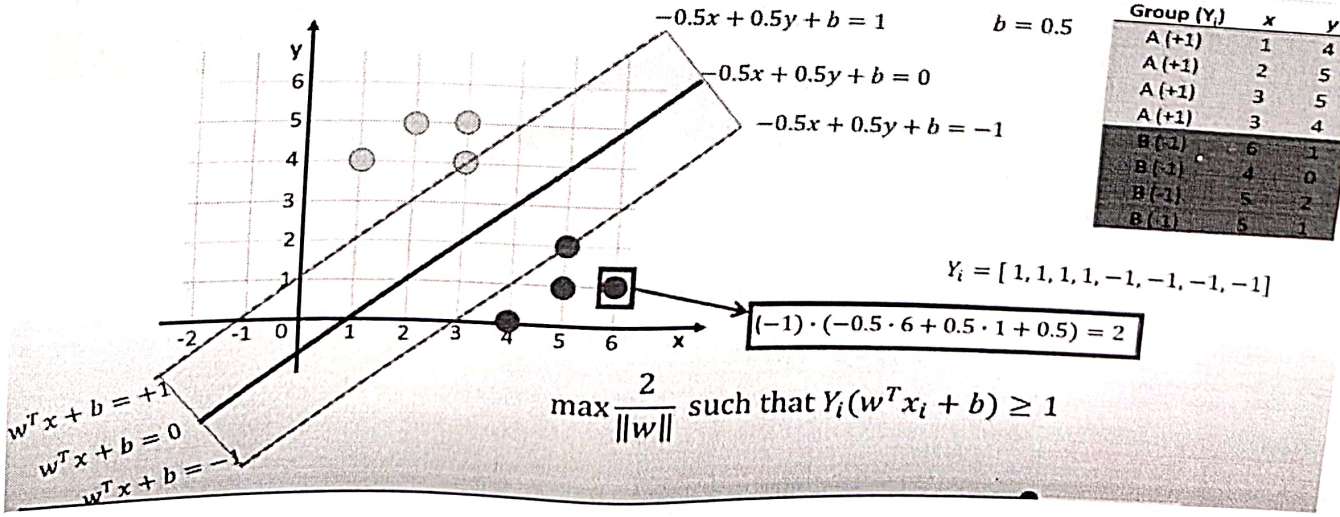


Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$$d = \frac{2}{\sqrt{A^2 + B^2}} = \frac{2}{\|w\|}$$

$$\max \frac{2}{\|w\|} \text{ such that } w^T x_i + b \begin{cases} \geq 1 & \text{if } Y_i = +1 \\ \leq -1 & \text{if } Y_i = -1 \end{cases}$$

Support Vector Machine



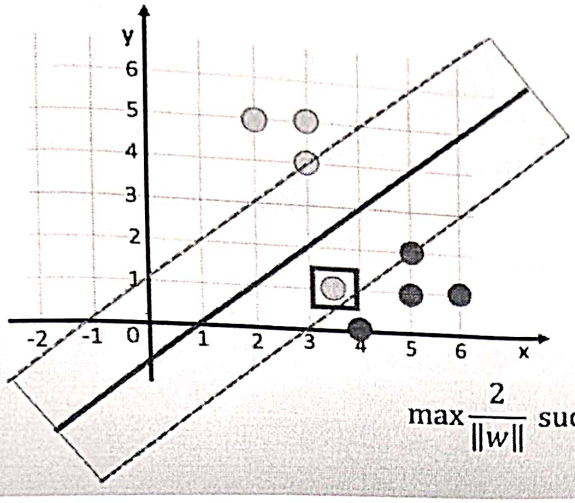
Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$Y_i = [1, 1, 1, 1, -1, -1, -1, -1]$

$$(-1) \cdot (-0.5 \cdot 6 + 0.5 \cdot 1 + 0.5) = 2$$

$$\max \frac{2}{\|w\|} \text{ such that } Y_i(w^T x_i + b) \geq 1$$

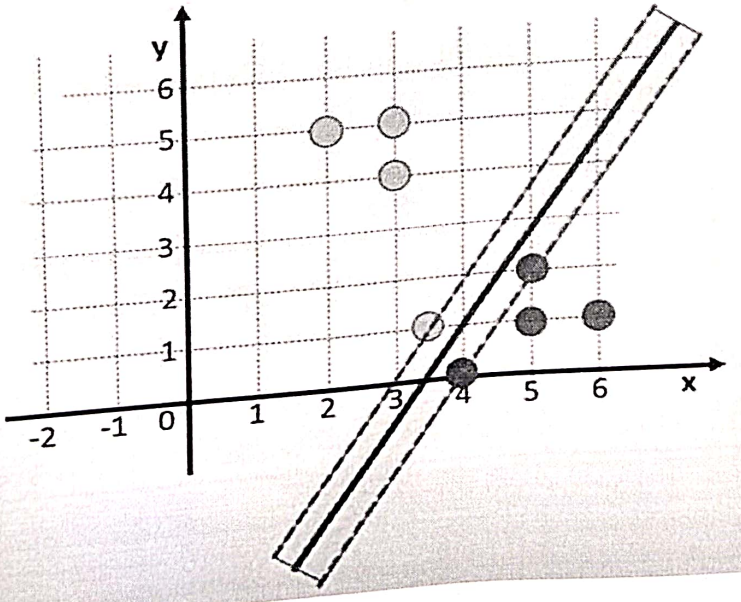
Support Vector Machine



$$\max \frac{2}{\|w\|} \text{ such that } Y_i(w^T x_i + b) \geq 1$$

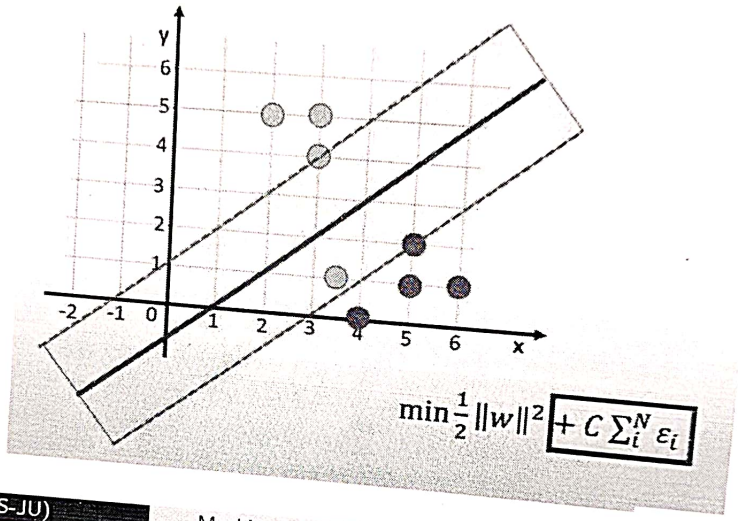
So what should we do if a yellow data point is down here or here we can either accept that the data point will be incorrectly classified. We'll change the hyperplane so that we correctly classify all data points but with the cost of a much smaller margin.

Support Vector Machine



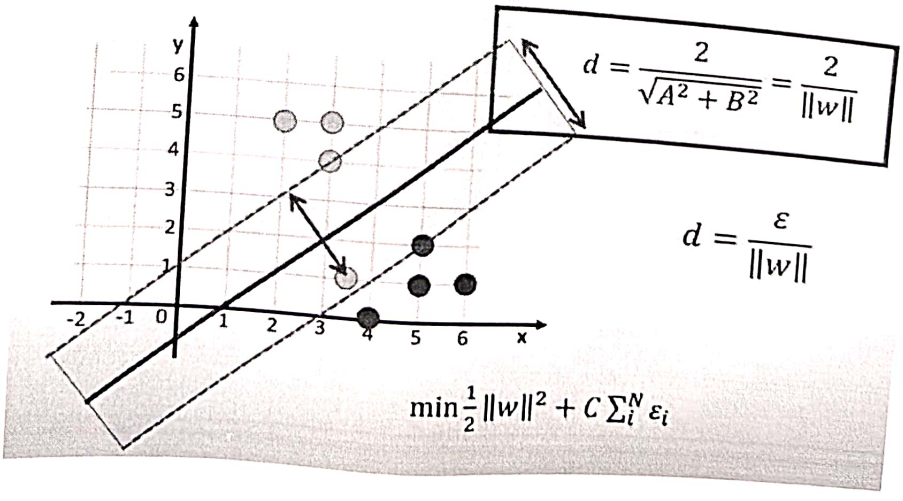
Support Vector Machine

To allow for misclassification, we can add this term where epsilon is a d
tance measure of the data points from their corresponding blue line. This
called a slack variable in support vector machines. In this example, there i
only one data point that is incorrectly classified because it is in the wrong
side of the hyperplane.



Support Vector Machine

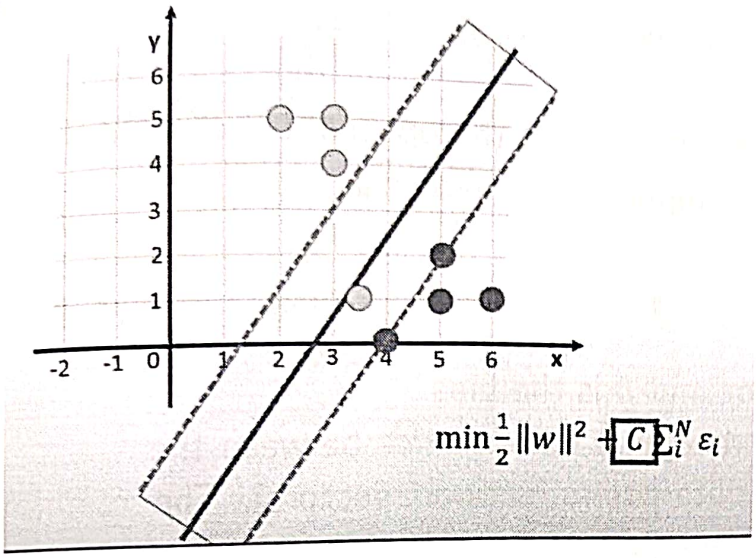
The distance between this data point and its corresponding blue line can be described by the following equation.



Remember that the width of the margin can be calculated by the following formula.

Support Vector Machine

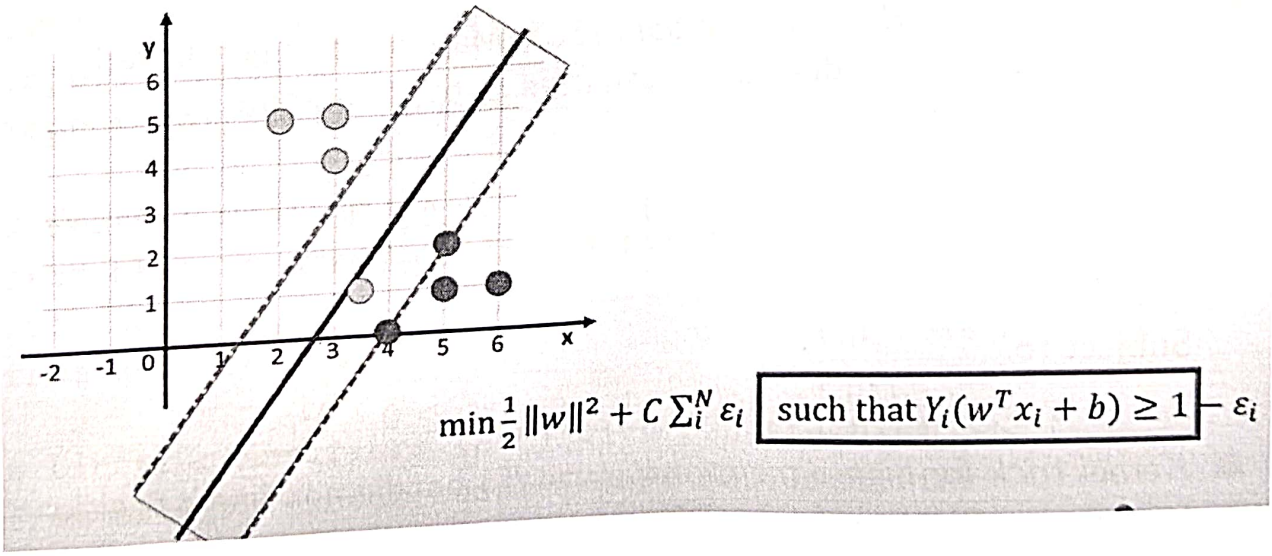
C is a tuning parameter that controls how much weight we should put on the data points that are incorrectly classified. If C is small we'll get a large margin where we put little weight on data points that are misclassified.



The value of C can be optimized by using for example cross-validation, where you pick the value of C that results in the best performance according to the cross validation.

Support Vector Machine

Similarly as before optimizing the support vector machine is constrained by



Support Vector Machine

The steps followed by SVM are:

- *Input the data:* Begin with a dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the feature vector, and $y_i \in \{-1, +1\}$ represents the class labels.
- *Define the hyperplane:* SVM aims to find the optimal hyperplane that separates the data points belonging to different classes. The hyperplane is represented as:

$$w^T x + b = 0$$

where w is the weight vector and b is the bias.

- *Maximize the margin:* The margin is the distance between the hyperplane and the closest data points (support vectors). The optimization problem is formulated to maximize this margin:

$$\text{Minimize: } \frac{1}{2}\|w\|^2$$

$$\text{Subject to: } y_i(w^T x_i + b) \geq 1, \quad \forall i$$

Support Vector Machine

- *Handle non-linearly separable data (Soft Margin):* For non-linearly separable data, introduce slack variables ξ_i to allow some misclassification:

$$\text{Minimize: } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

Subject to:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

- *Kernel trick for high-dimensional data:* To handle non-linear decision boundaries, SVM uses kernel functions to map the data to a higher-dimensional space:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Support Vector Machine

- *Solve the optimization problem:* Use the Lagrange multiplier method to solve the optimization problem, leading to the dual form:

$$\text{Maximize: } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i$$

- *Make predictions:* The decision function for a new input x is:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j) + b \right)$$

where α_i are the Lagrange multipliers, and b is the bias term.

Support Vector Machine

SVM for Mymensingh Data

```
"""#===Support Vector Machine Classifier===#"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Feature Scaling
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

# Linear classifiers (SVM) with SGD training.
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

Support Vector Machine

SVM for Mymensingh Data

```
import statsmodels.tools.tools as stattools
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import graphviz

D=pd.read_csv('D:/Mymensingh.csv')
D.dropna(how='any',axis=0, inplace=True)
D.shape
D.head()

DD=D.drop(['ID','Station','Year','Month','T_RAN','A_RAIN'], axis=1)
X=DD.drop(['RAN'], axis=1)
Y=DD['RAN']
```


Support Vector Machine

SVM for Mymensingh Data

```
np.random.seed(104729)
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,
                        test_size=0.25,
                        random_state=31)

SV = make_pipeline(StandardScaler(),
                   SGDClassifier(max_iter=1000000,
                   loss='hinge',
                       tol=1e-3,fit_intercept=True))
SV.fit(X_train, Y_train)
```

upport Vector Machine

SVM for Mymensingh Data

```
# Prerdict for Training Data
SPR=SV.predict(X_train)
accuracy_score(Y_train, SPR)
print(classification_report(Y_train,SPR))
```

	precision	recall	f1-score	support
LTR	0.67	0.63	0.65	196
MHR	0.66	0.80	0.72	115
NRT	0.83	0.77	0.80	179
accuracy			0.72	490
macro avg	0.72	0.73	0.72	490
weighted avg	0.72	0.72	0.72	490

Support Vector Machine

SVM for Mymensingh Data

```
# Prerdict for Test Data
SPS=SV.predict(X_test)
accuracy_score(Y_test, SPS)

print(classification_report(Y_test,SPS))
```

	precision	recall	f1-score	support
LTR	0.61	0.72	0.66	54
MHR	0.71	0.79	0.75	43
NRT	0.96	0.75	0.84	67
accuracy			0.75	164
macro avg	0.76	0.75	0.75	164
weighted avg	0.78	0.75	0.76	164

Support Vector Machine

SVM for Mymensingh Data

```
'''==Cohen Kappa=='''
from sklearn.metrics import cohen_kappa_score
cohen_kappa_score(Y_train, SPR)

cohen_kappa_score(Y_test, SPS)


SV.get_params(deep=True)
```


Support Vector Machine

SVM for Mymensingh Data

```
'''===K-fold Cross Validation==='''
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
K=[5,10]
p=-1
S=np.zeros((len(K),10))

for i in K:
    scoring_01=['accuracy','precision_macro',
                'recall_macro','f1_macro']
    SV1 = make_pipeline(StandardScaler(),
                        SGDClassifier(max_iter=1000000,
                                loss='hinge',
                                tol=1e-3,fit_intercept=True))
    scores_01 =cross_validate(SV1, X, Y,  cv=i,
                              scoring=scoring_01,
                              return_train_score=True)
    p=p+1
    QW=pd.DataFrame.mean(pd.DataFrame(scores_01), axis=0)
    S[p,]=np.array(QW)
```

Support Vector Machine

SVM for Mymensingh Data

```
#sorted(scores_01.keys())
scores_01.keys()

S1=pd.DataFrame(S)
S1.columns =['fit_time', 'score_time', 'test_accuracy',
             'train_accuracy', 'test_precision_macro',
             'train_precision_macro', 'test_recall_macro',
             'train_recall_macro', 'test_f1_macro',
             'train_f1_macro']

# Using DataFrame.insert() to add a column
S1.insert(0, "K-fold", K, True)
np.round(S1,4)
```

	K-fold	fit_time	...	test_f1_macro	train_f1_macro
0	5	0.0094	...	0.6309	0.6774
1	10	0.0055	...	0.6879	0.6741

Look for Details

Find Detalis: Han et al. (2022), Hastie et al. (2017), Awad and R (2015), Cortes (1995)