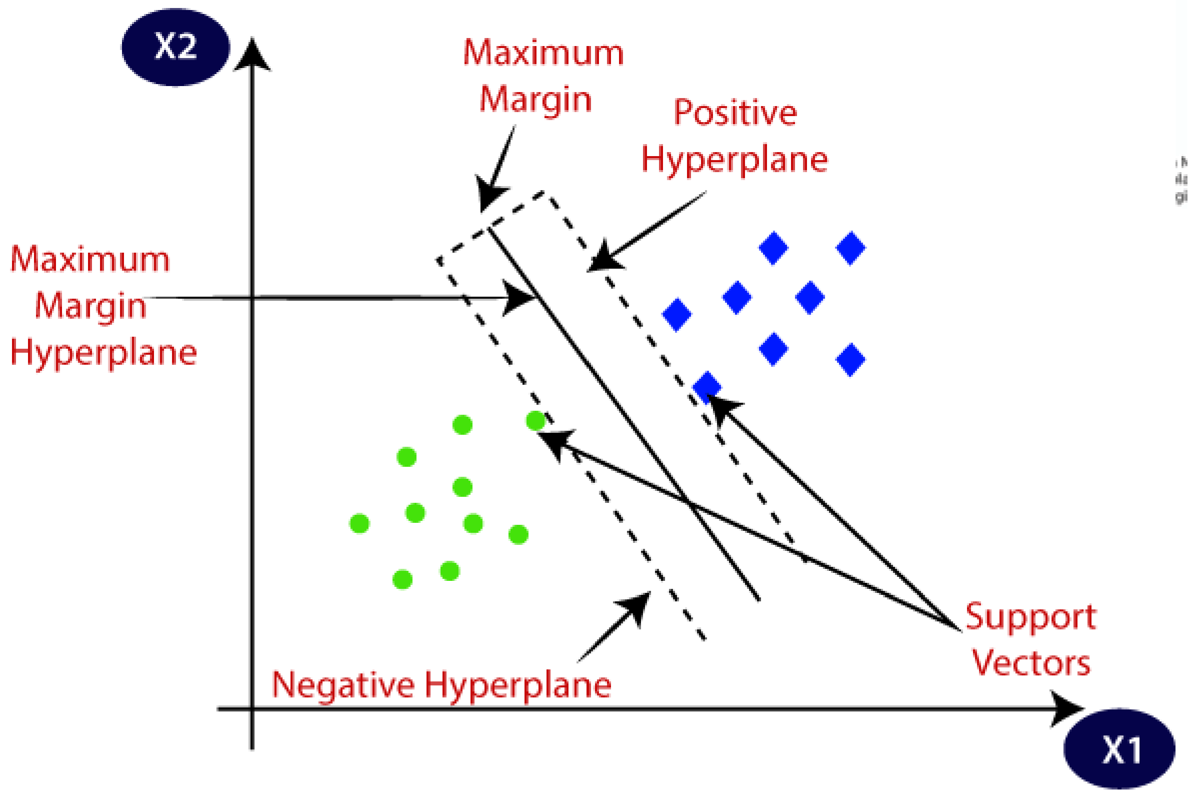
Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Befor going to SVM work first know some genral things:

- 1. Support Vector
- 2. Hyper Plane
- 3. Marginal Distance
- 4. Kernel
- 5. Linear and Non Linear Separable Data

1. 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.
2. 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.
3. Marginal Distance is the two parallel line that create w.r.t most near point of +ve and -ve to plane distance. we always maximise the marginal distance.
4. See the below figure you better understood all these three term.

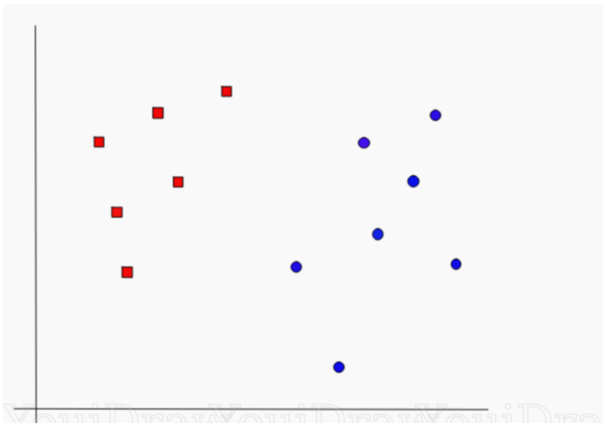


4. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.

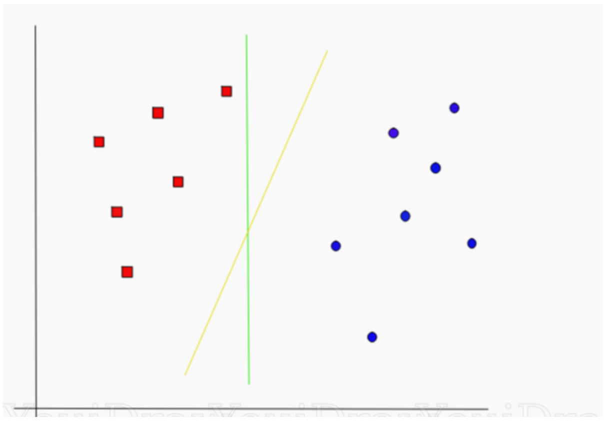
To Know More About Kernel click this:https://data-flair.training/blogs/svm-kernel-functions/#:~:text=SVM%20Kernel%20Functions,it%20into%20the%20required%20form.&text=For%20example%
(RBF)%2C%20and%20sigmoid (https://data-flair.training/blogs/svm-kernel-functions/#:~:text=SVM%20Kernel%20Functions,it%20into%20the%20required%20form.&text=For%20example%
(RBF)%2C%20and%20sigmoid).

- Below i demonstrate linear and non linear separable dataset.

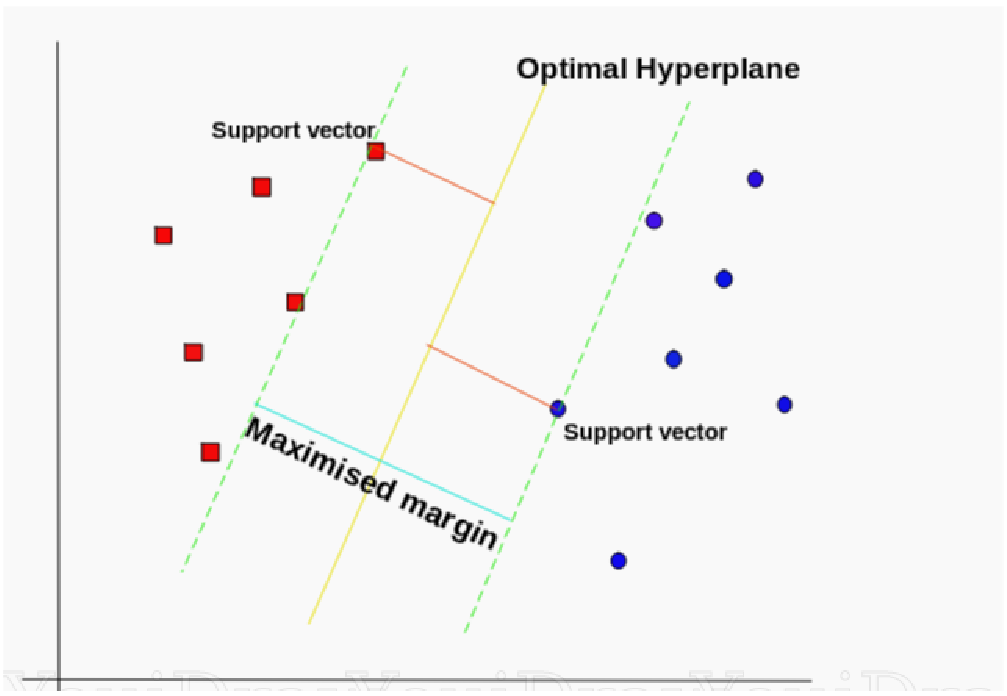
Linear Separable data



- Suppose you have a dataset as shown above and you need to classify the red rectangles from the blue. So your task is to find an ideal line that separates this dataset in two classes.
- It is not so difficult as i learnt before when there is a linear separable data set you just draw a line with the help of linear regression then it will be separable easily.

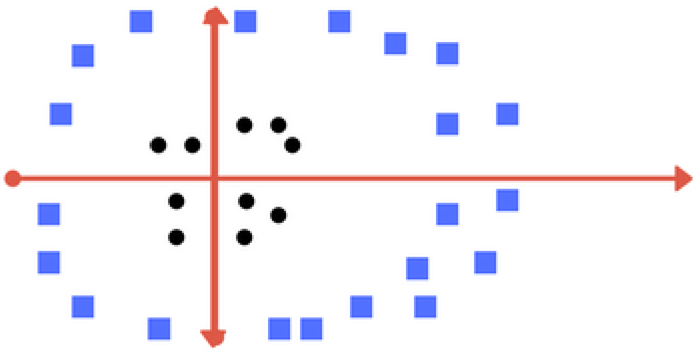


- Two lines here, the green colored line and the yellow colored line. Which line according to you best separates the data?
- If you selected the yellow line then it correct , because thats the line we are looking for. It’s visually quite intuitive in this case that the yellow line classifies better. But, we need something concrete to fix our line.
- Here we see how linear regrssion separate the dataset. Now Times to see How SVM find the best line.
- According to the SVM algorithm that find the points closest to the line from both the classes.These points are called support vectors. Now, we compute the distance between the line and the support vectors.
- This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

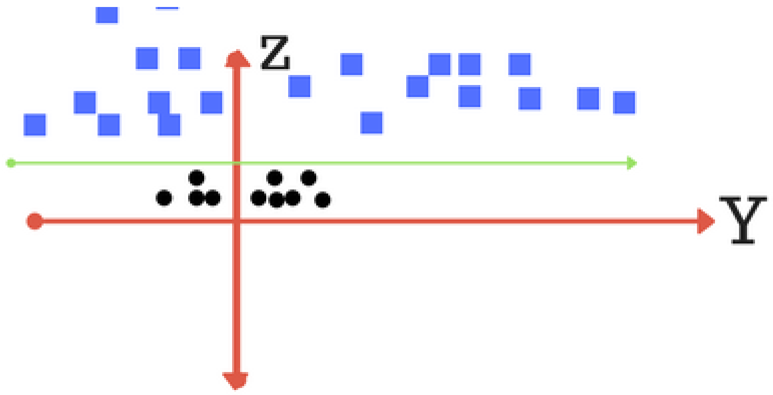


No-Linear Separable Data

- SVM Can work both linear and non- linear dataset. above i dicuss about how svm works in linear data now dicuss about works in non linear data.

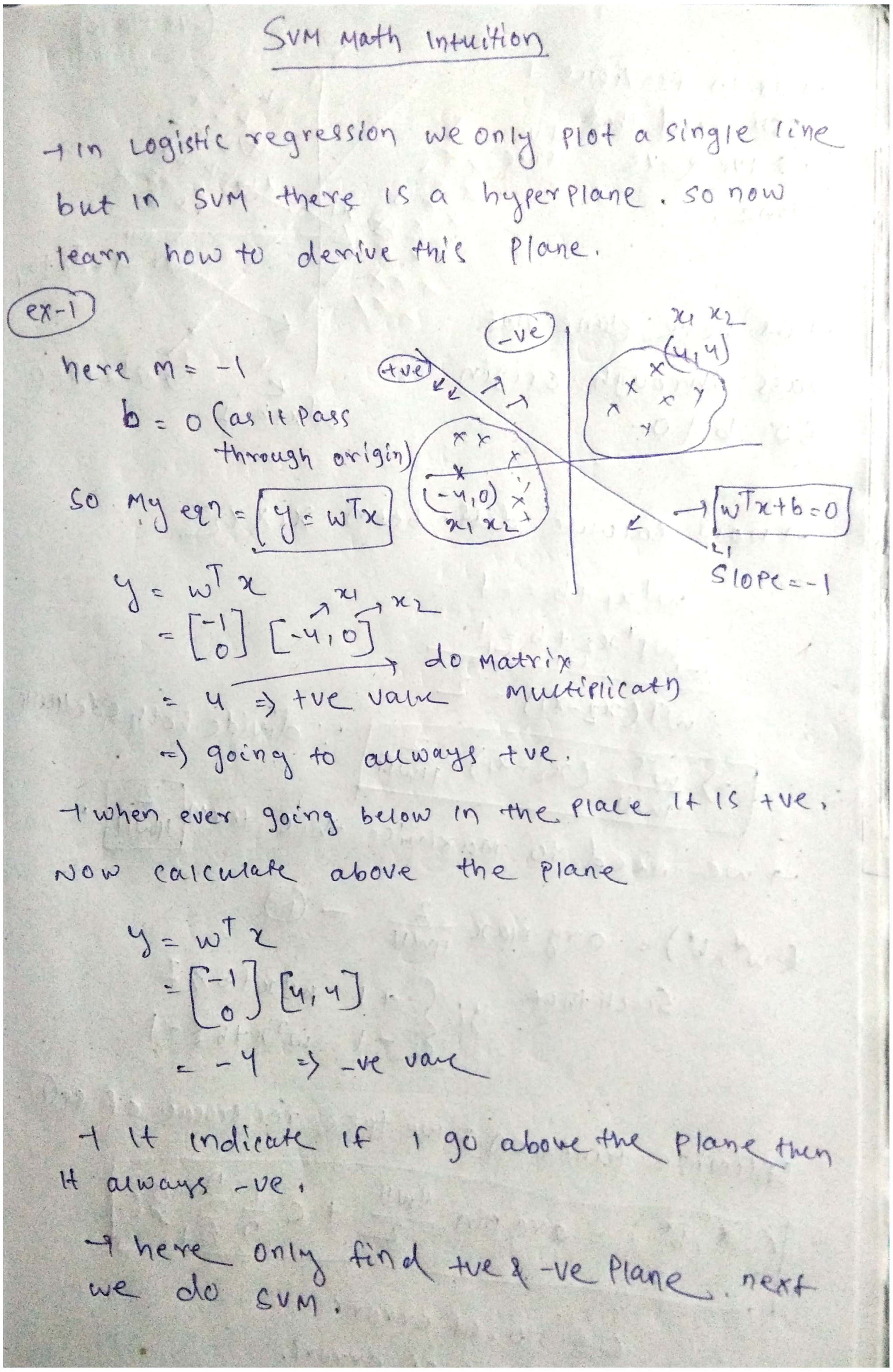


- Now consider what if we had data as shown in image below? This is a type of non linear data. Clearly See, there is no line that can separate the two classes in this x-y plane. So what do we do?
- We apply transformation and add one more dimension as we call it z-axis. Lets assume value of points on z plane, w = $x^2 + y^2$ - So, basically z co-ordinate is the square of distance of the point from origin. Let's plot the data on z-axis..
- After Transform



- These Transform Done By SVM kernels.

Math Intution About SVM



SVM Math Intuition

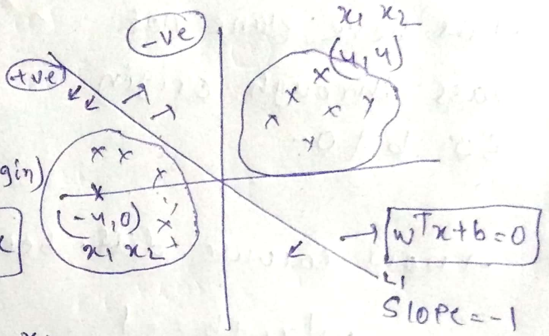
→ In Logistic regression we only plot a single line but in SVM there is a hyperplane . so now learn how to derive this plane.

ex-1

here m = -1

b = 0 (as it pass through origin)

so My eqn = $\boxed{y = w^T x}$



$y = w^T x$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} \overset{x_1}{-4}, \overset{x_2}{0} \end{bmatrix} \xrightarrow{} \text{do Matrix multiplicatn}$

$= 4 \Rightarrow$ +ve value

⇒ going to allways +ve.

→ when ever going below in the place it is +ve,

Now calculate above the plane

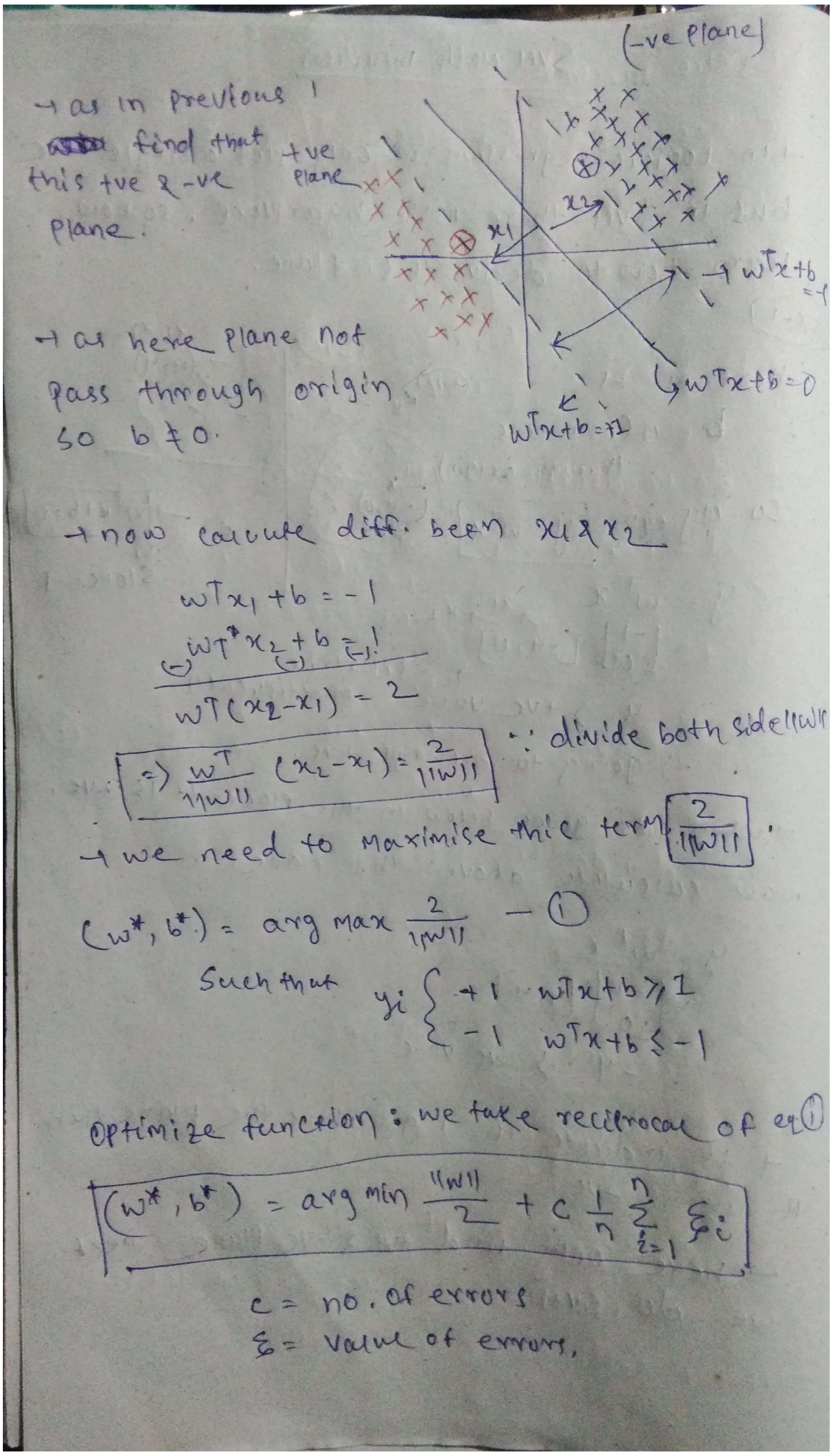
$y = w^T x$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4, 4]$

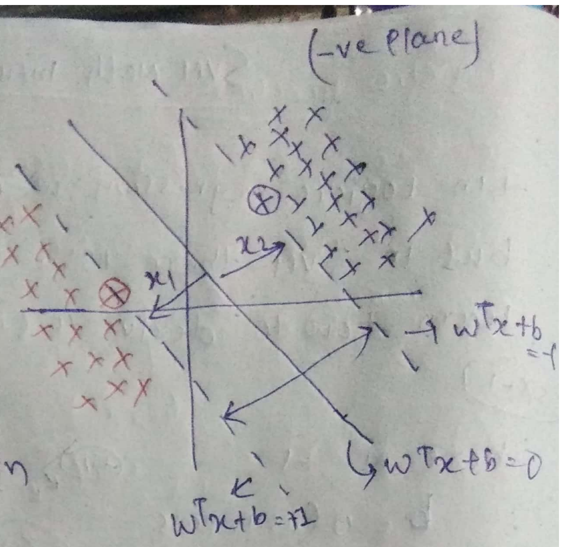
$= -4 \Rightarrow$ -ve vaue

→ It indicate if I go above the plane then it always -ve.

→ here only find +ve & -ve Plane. next we do SVM.



→ as in previous I ~~[illegible]~~ find that this +ve & -ve plane.



→ as here plane not pass through origin so $b \neq 0$.

→ now calcute diff. betn x_1 & x_2

$$w^Tx_1 + b = -1$$
$$\underset{(-)}{w^T} x_2 \underset{(-)}{+ b} = \underset{(-)}{1}$$
$$w^T(x_2 - x_1) = 2$$

$$\boxed{\Rightarrow \frac{w^T}{\|w\|}(x_2 - x_1) = \frac{2}{\|w\|}}$$ ∵ divide both side $\|w\|$

→ we need to maximise this term $\boxed{\frac{2}{\|w\|}}$.

$$(w^*, b^*) = \arg\max \frac{2}{\|w\|} \quad — ①$$

Such that $y_i \begin{cases} +1 & w^Tx+b \geq 1 \\ -1 & w^Tx+b \leq -1 \end{cases}$

optimize function : we take recilrocal of eq ①

$$\boxed{(w^*, b^*) = \arg\min \frac{\|w\|}{2} + c\frac{1}{n}\sum_{i=1}^{n} \xi_i}$$

c = no. of errors
ξ = value of errors.

Let's Do an Example

In [1]:

```
## Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Importing the dataset

In [2]:

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

In [3]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
```

Feature Scaling

In [4]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the SVM model on the Training set

In [5]:

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[5]:

```
SVC(kernel='linear', random_state=0)
```

Predicting the Test set results

In [6]:

```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

In [7]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

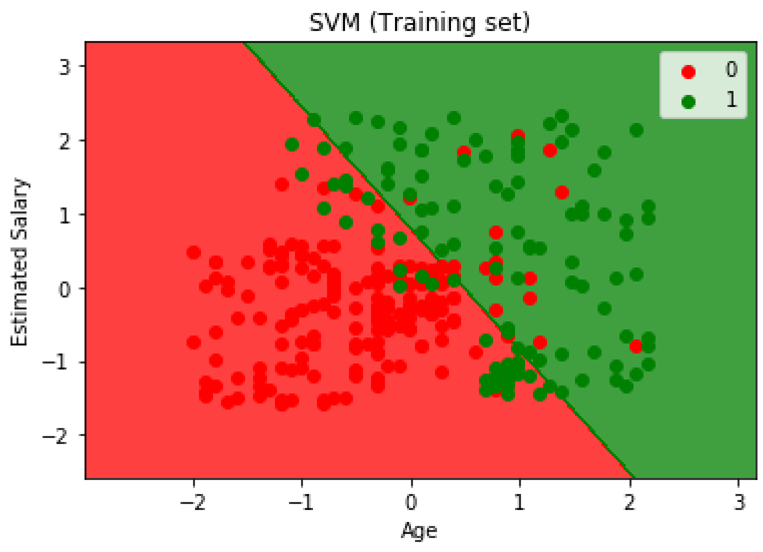
```
[[66  2]
 [ 8 24]]
```

Visualising the Training set results

In [8]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1,
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1,
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.sh
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should
be avoided as value-mapping will have precedence in case its length matches
with 'x' & 'y'.  Please use a 2-D array with a single row if you really want
to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should
be avoided as value-mapping will have precedence in case its length matches
with 'x' & 'y'.  Please use a 2-D array with a single row if you really want
to specify the same RGB or RGBA value for all points.
```

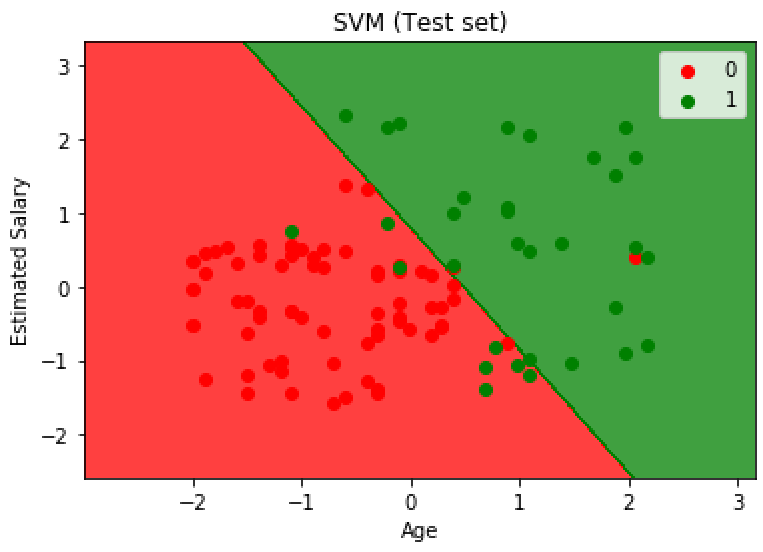


Visualising the Test set results

In [9]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1,
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1,
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.sh
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should
be avoided as value-mapping will have precedence in case its length matches
with 'x' & 'y'.  Please use a 2-D array with a single row if you really want
to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should
be avoided as value-mapping will have precedence in case its length matches
with 'x' & 'y'.  Please use a 2-D array with a single row if you really want
to specify the same RGB or RGBA value for all points.
```


SVM (Test set)
● 0
● 1
Estimated Salary
Age