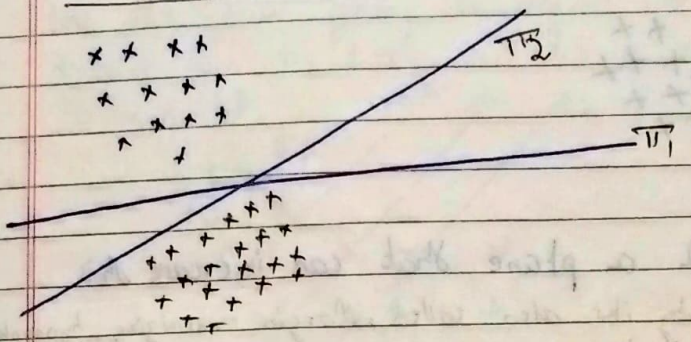
Support Vector Machine

↳ Can keep in the family of logistic Regression

Saathi

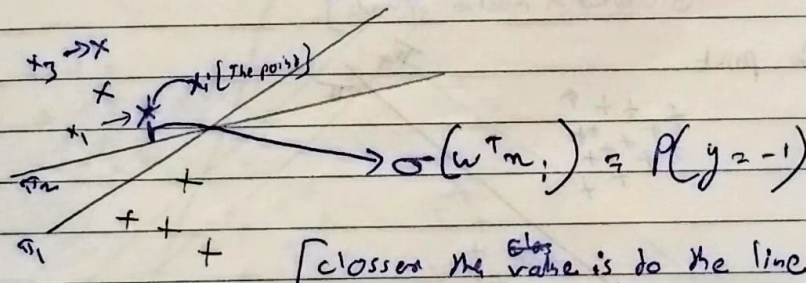
Date ___/___/___

Geometric intuition



Both the line are able to classify Blue & Red. Logistic Regression takes/considereds both equally correct but svm takes the better line/hyperplane to classify Blue & Red. svm would take Π_1 as the classifing line/hyperplane

Now the reason of selecting Π_1.



considering Π_2

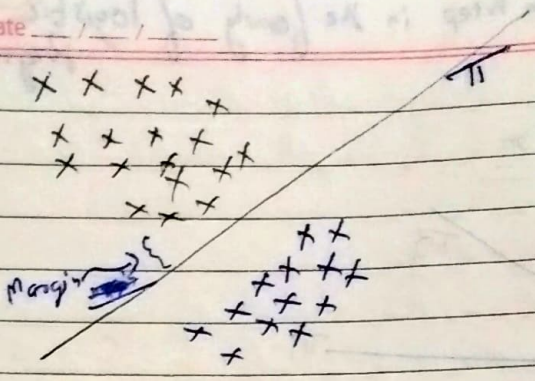
closer the value is to the line $\sigma(w^T x_i) \to 0$

so for x_1 $P(y=-1) \approx 0$

But for x_3 $P(y=-1) \approx 0.99$

SVM tries to classify all the point by selecting that hyperplane who classifies all the points as widly as possible. So that when in future we try to get unseen data Π_1 will perform better than Π_2.

Date ___/___/____



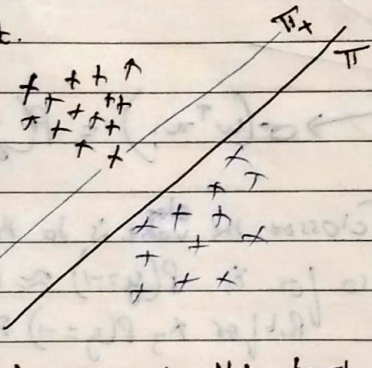
We need to find a plane that can increase this margin & that's why it's also called, Margin maximizing hyperplane. The core idea is to find a hyperplane which gives us a maximum margin. So that our model becomes more generalized and works well for unseen data also.

But how?

First we randomly select a hyperplane. Now in the two direction of the hyperplane until we reach the first

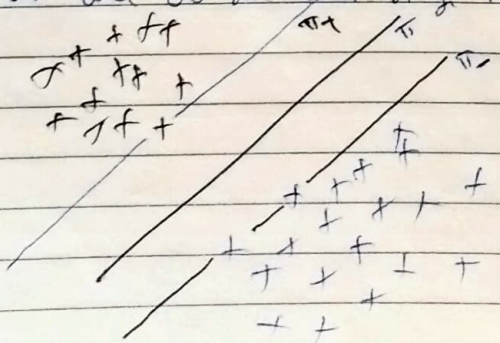
[(-ve) direction matlab ↑ upwards]

black point.



This is // to π and this is the ~~one~~ hyperplane which touches the black point and this is a Red Flag so we stop here.

Now we do the same thing in the oppo. direction.

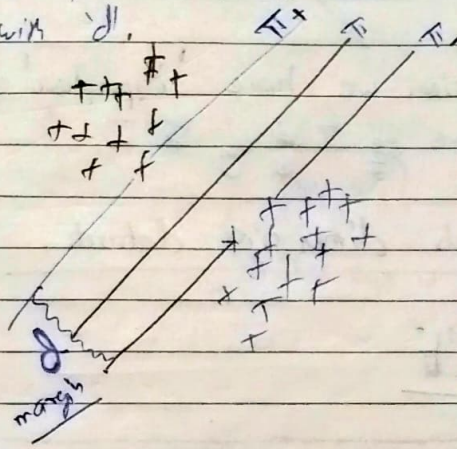


Saathi

Date ___ / ___ / _____

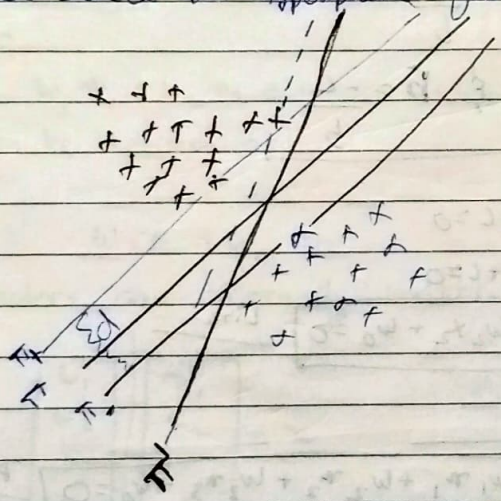
So we stop when we reach the first blue point.

We are now going to define margin. We calc. the distance b/w π_+ & π_- and this is the margin and denote it with 'd'.

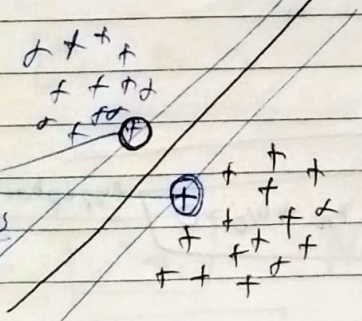


Now do the same steps for a bunch of hyperplanes. Find π_+, π_- for all and evaluate d for all. Then we shall select the hyperplane for which 'd' is maximum.

Eg



Support Vectors

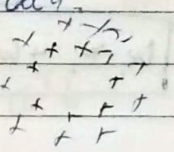


These are called support vectors

SVM

1) Robust to outliers
   ↳ Able to handle outliers

2) Can also work on non-linear data.



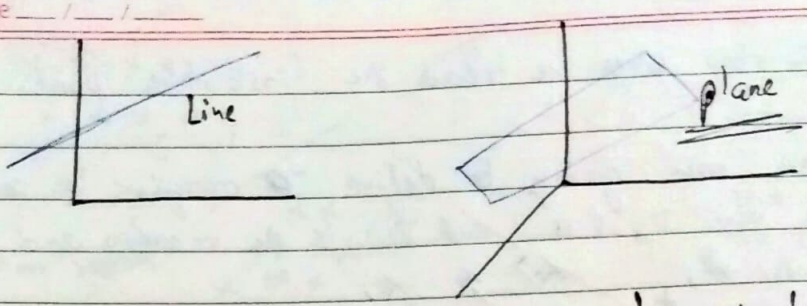
3) Can be implemented to both classification & Regression

Page No.

Eqn of a Hyperplane (π)

Saathi

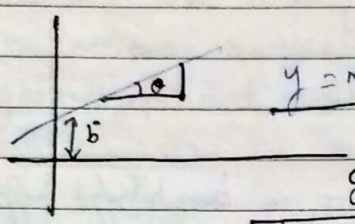
Date ___ / ___ / _____



For 4D, 5D.... dimension we have 'Hyperplane' equivalent to line of 2D & plane of 3D.

We will be getting high dimension dataset.

2D

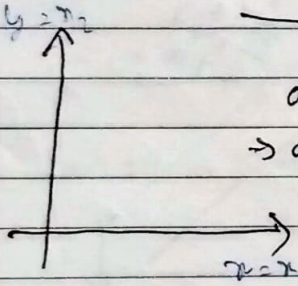


general form

$$ax + by + c = 0$$

$$\Rightarrow y = -\frac{a}{b}x - \frac{c}{b}$$

$$\because m = -\frac{a}{b} \text{ & } b = -\frac{c}{b}$$

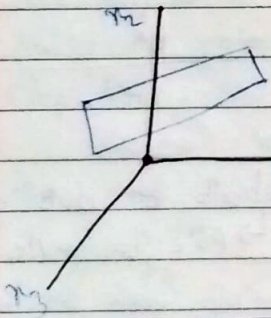


$$ax + by + c = 0$$

$$\rightarrow ax_1 + bx_2 + c = 0$$

$$w_1x_1 + w_2x_2 + w_0 = 0$$ Line

3D



$$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$$ plane

nD

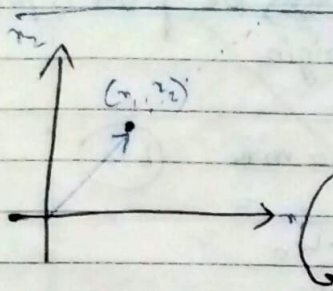
$$w_1x_1 + w_2x_2 + \dots + w_nx_n + w_0 = 0$$ hyperplane

$w_1 x_1 + w_2 x_2 + \dots + w_n x_0 + w_0 = 0$

we can compact it further.

TT

20



$\{x_1, x_2\}$ is taken as a vector.

$w_1 x_1 + w_2 x_2 = 0$

$\{w_1, w_2\}$ is taken as a vector

So This eqn is simply a dot product.

$w \cdot x = w_1 x_1 + w_2 x_2$

$\vec{w} = [w_1 \; w_2 \; w_3 \; \dots \; w_n]$

$\vec{x} = [x_1 \; x_2 \; x_3 \; \dots \; x_n]$

so

$w_1 x_1 + w_2 x_2 + \dots + w_0 x_n + w_0 = 0$

can be written as

$w \cdot x + w_0 = 0$

Remember vectors are normally written in ~~vertical~~ column format

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x$$

So to get the desired eqn we need.

$$\{w_1 \; w_2 \; w_3 \; \dots \; w_n\} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

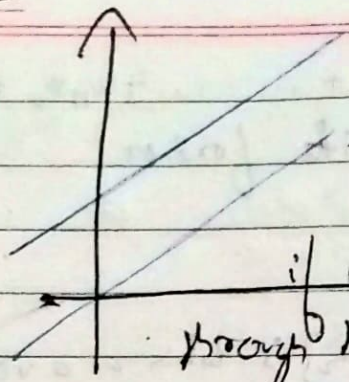
row vector

So $\boxed{w^T x}$ NOTE: $a \cdot b = a^T b$

$\boxed{w^T x + w_0 = 0}$ In any dimension

Now what is w_0?

Trying to derive it from 2D



$y = mx + b$

$\Rightarrow w_1 x_1 + w_2 x_2 + w_0 2D$

if we say the line is passing through the origin.

Then

$y = mx$ (b = 0)

hence for any plane passing through the origin

$w^T x = 0$

$\Rightarrow$ $w^T x = 0$ On the assumption that plane is passing from the origin.

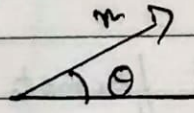
And $b = -\frac{w_0}{w_2}$

$\therefore$ w_0 need to be 0

$w^T x = 0$

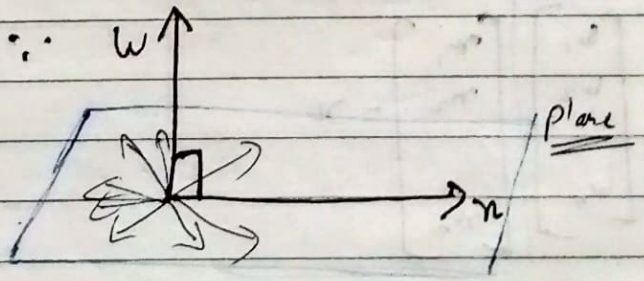
what is w?

$w^T x = w \cdot x = ||w|| \; ||x|| \cos\theta = 0$



possible only when $\theta = 90$

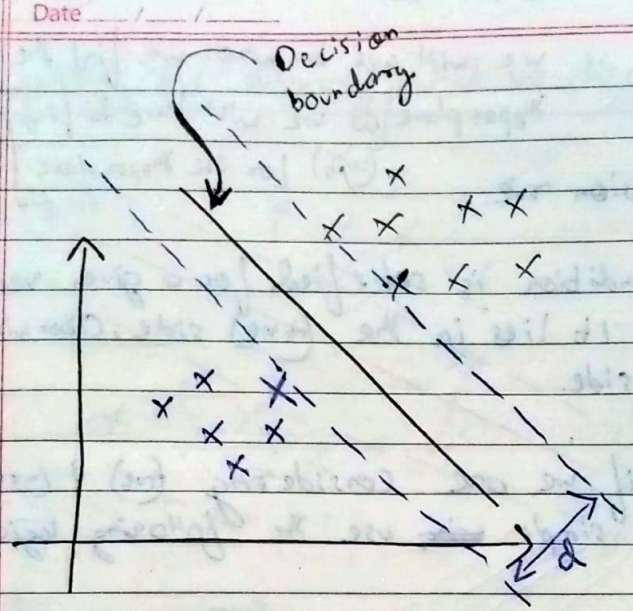
$\therefore$ w



plane

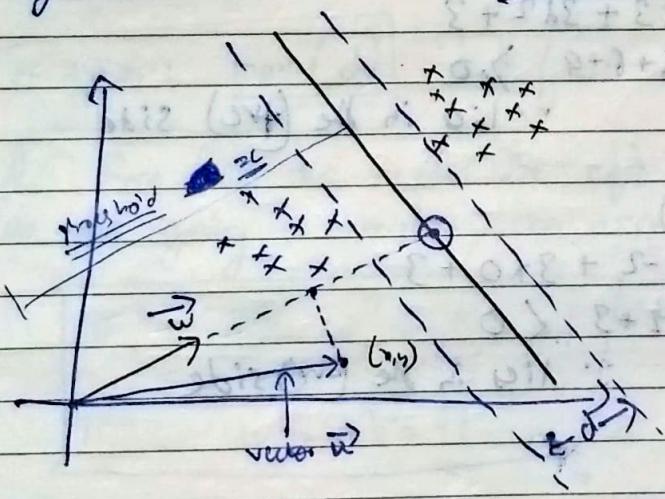
Maths of SVM | Hard Margin

Date ___/___/_____



Basic thing is to find ~~d~~ decision boundary such that d is max.

Lets now ~~focus~~ focus on the thing that how is the new item is going to be classified. Therefore we need set Decision Rule. Now we take vector $\vec{w}$ ⊥ to all the three lines. We know the direction but not the magnitude. Let now assume a point.



projection of $\vec{u}$ on $\vec{w}$ = $\vec{w} \cdot \vec{u}$

The vector $\vec{u}$ to be (+ve) on black side, it needs to pass the threshold. But we don't know the threshold point. Let's say threshold distance is c.

Therefore

$\vec{w} \cdot \vec{u} \geq c$

b will be

$\vec{w} \cdot \vec{u} - c \geq 0$

we will get b when we find the line/hyperplane (as we will have to find (w,b) for the hyperplane)

$\boxed{\vec{w} \cdot \vec{u} + b \geq 0}$

Decision rule

If the above condition is satisfied for a given vector $\vec{u}$ then it means it lies in the (+ve) side. Otherwise it lies on the (-ve) side.

Now, Note that if we are considering (+ve) & (-ve) of a line we can simple use the following logic

Eg: Let the line be

$\boxed{2x + 3y + 3 = 0}$

writing in vector form

$w_1 x_1 + w_2 x_2 + w_0 = 0$

where $w = (3, 3)$ $\quad w_0 = 3$

$U = (x_1, x_2)$

Let

$U = (3, 2)$

so $2 \times 3 + 3 \times 2 + 3$

$\Rightarrow 6 + 6 + 9 \geq 0$

$\therefore$ lies in the (+ve) side

let $U = (-2, 0)$

so $2 \times -2 + 3 \times 0 + 3$

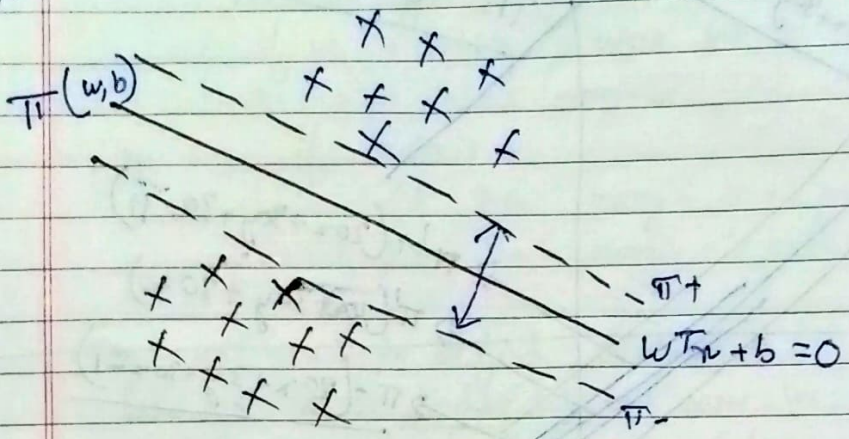
$\Rightarrow -4 + 3 < 0$

$\therefore$ lies in the (-ve) side

Saathi

Date ___/___/_____

Putting the summary what we discussed.



for any $\vec{x}_i$;

$$\hat{y} = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 0 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + b < 0 \end{cases}$$

To find the line

condition: $\max(d)$ but first we need an eq^n for d. For this we need the eq^n of π^+ & π^- We are going to make an assumption.

π^+ : $w^T x + b = 1$

π^- : $w^T x + b = -1$

But why?

To understand lets take an example in 2D

π : $2x + 3y + 3 = 0$ → π^+ : $2x + 3y + 3 = +1$

→ π^- : $2x + 3y + 3 = -1$

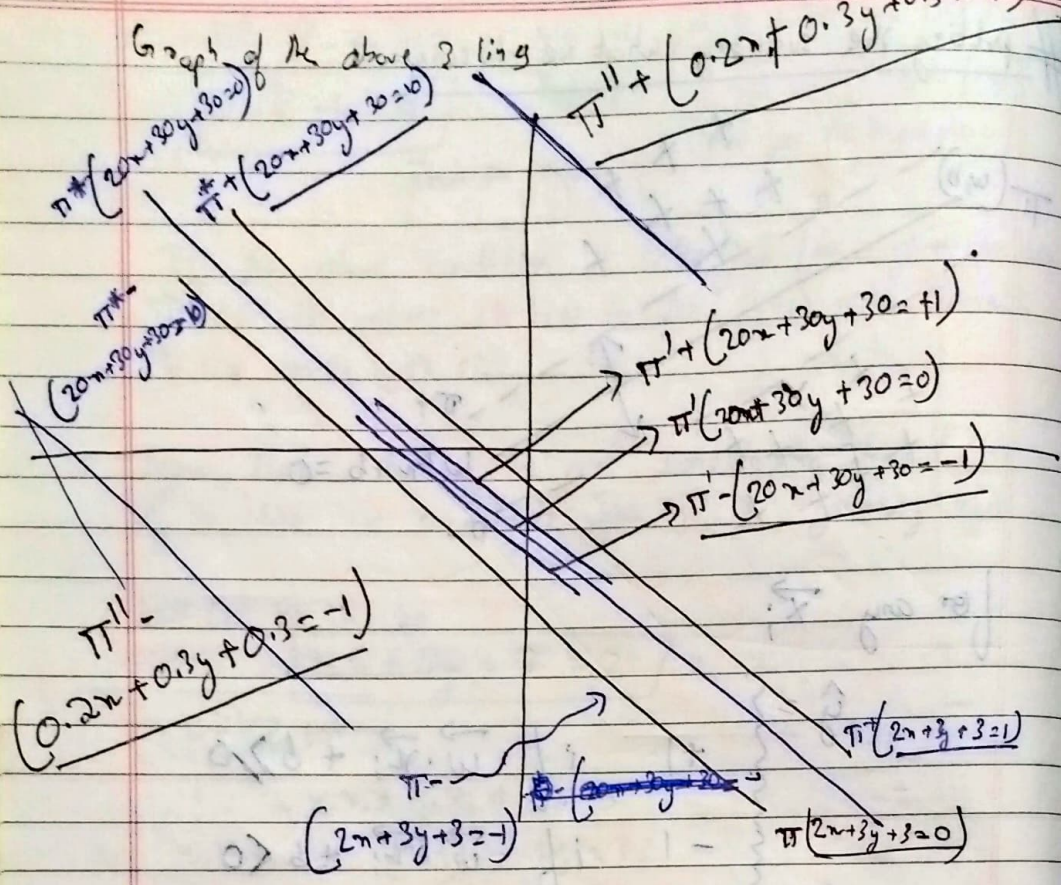
Page No.

$\pi^+ : 2x+3y+3=+1$

$\pi : 2x+3y+3=0$

$\pi^- : 2x+3y+3=-1$

Graph of the above 3 lines



If we multiply 10 to π that is

$\pi^*: 20x+30y+30=0$

we will get the same line.

NOTE for $[\pi'^+, \pi', \pi'^-] \& [\pi^+, \pi, \pi^-]$

- if we multiply just the LHS of $[\pi^+, \pi, \pi^-]$ with 10, we see π^+ & π^- lines get closer to π.

now for $[\pi''^+, \pi'', \pi''^-] \& [\pi^+, \pi, \pi^-]$

if we multiply just the LHS of $[\pi^+, \pi, \pi^-]$ with 0.1, we see π^+ & π^- lines move further away from π

Saathi

Date ___/___/_____

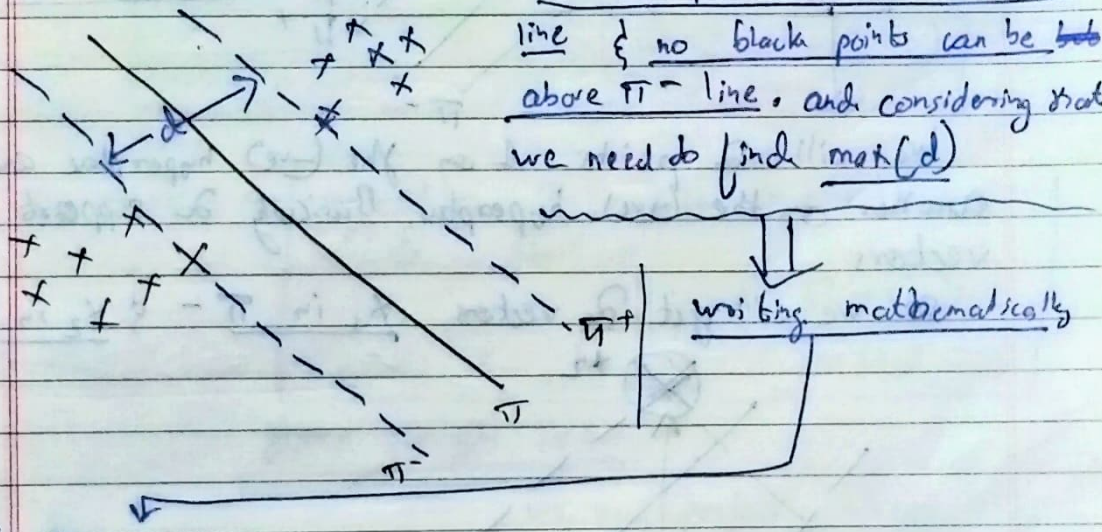
In short

For these family of lines/eq^2
if we multiply a scalar with the $\vec{w}$ & b we see the changes in the margin.
if scalar is
< 1 then margin is expanding
& > 1 then margin is shrinking

---

now we need do find distance d.
But we cannot find d in all cases. we can use in some specific cases itself. There are constraints.

The constrain being ⟹ no blue points can be below π^+ line & no black points can be above π^- line. and considering that we need do find max(d)



writing mathematically

For blue points
$\vec{w} \cdot \vec{x} + b \geq 1$ where $\vec{x}$ = any blue point

For black points
$\vec{w} \cdot \vec{x} + b \leq -1$ where $\vec{x}$ = any black point

Assumption: label of all (+ve) points = $+1 = y_+$
label of all (−ve) points = $-1 = y_-$

converting into 1 eq^n

$$\therefore \boxed{y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1}$$

y_+ / y_-

for all support vectors

$$\boxed{y_i(\vec{w} \cdot \vec{x}_i + b) = 1}$$

Page No.

Date ___/___/_____

Now, what we have,

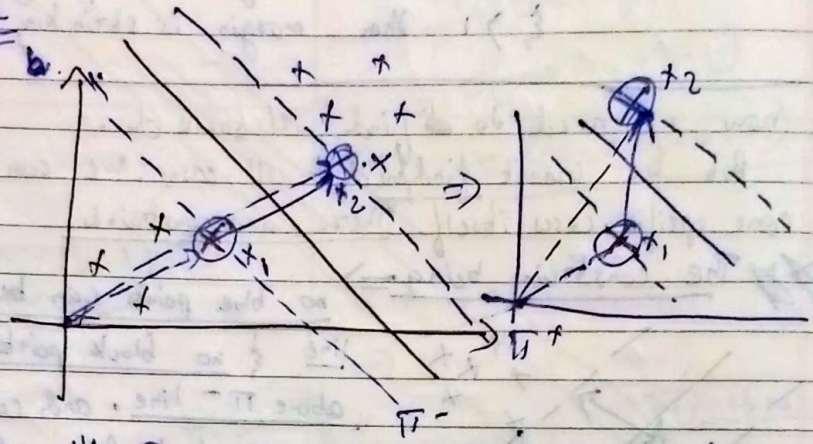
Constraint

$$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

for support vectors

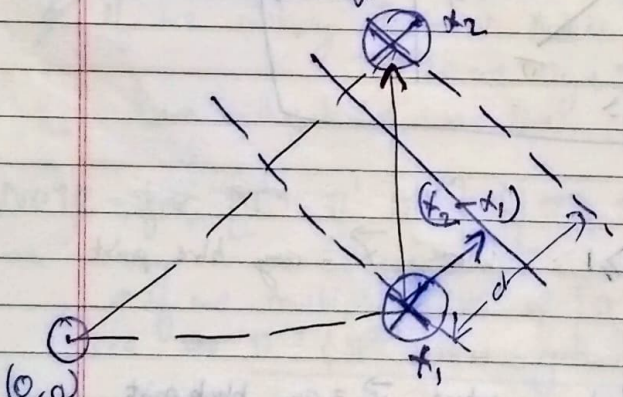
$$y_i(\vec{w}\cdot\vec{x}_i + b) = 1$$

For d



We will 2 points 1 on the (-ve) hyperplane and another on the (+ve) hyperplane. Basically 2 support vectors

so we will get 2 vectors x_1 in Π^- & x_2 in Π^+



[⊥ of $(x_2 - x_1)$ vector]

if we draw a ⊥ from x_1, ~~and proj it back~~, now remember we derived $\vec{w}$ to be ⊥ to all 3 lines so we just need to multiply the unit vector of $\vec{w}$ and $(x_2 - x_1)$

$$\therefore d = (\vec{x}_2 - \vec{x}_1) \cdot \frac{\vec{w}}{\|w\|}$$

Page No.

$$= \frac{\vec{x}_2 \cdot \vec{w} - \vec{x}_1 \cdot \vec{w}}{||w||}$$

$$= \frac{(1-b) - -(1+b)}{||w||}$$

$$= \frac{(1-b+1+b)}{||w||}$$

$$\boxed{d = \frac{2}{||w||}}$$

Expression for distance

x_2 is a support vector for π^+

so for s.v.

$y_i(\vec{w}\vec{x}_i + b) = 1$

$\Rightarrow 1(\vec{w}\vec{x}_2 + b) = 1$

$\boxed{\vec{w} \cdot \vec{x}_2 = 1 - b}$

x_1 is a support vector for π^-

so

for sv

$y_i(\vec{w}\vec{x}_i + b) = 1$

$= -1(\vec{w}\vec{x}_i + b) = 1$

$\Rightarrow \vec{w}\vec{x}_i + b = -1$

$\Rightarrow \boxed{\vec{w}\vec{x}_i = -(1+b)}$

so we need do $\max_{(w^*, b^*)} (d) = \frac{2}{||w||}$ such that

$$y_i(\vec{w}\vec{x}_i + b) \geq 1$$

Hard margin SVM

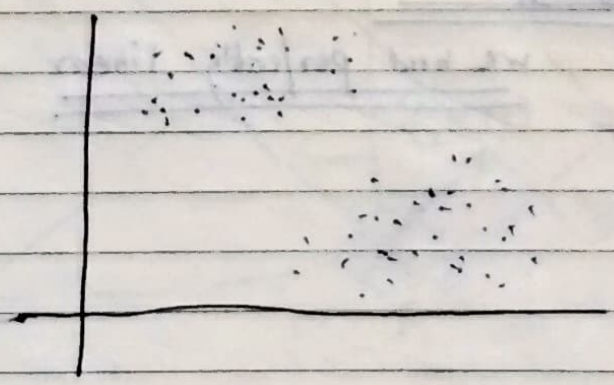
we need perfectly linear

Code Sample

Saathi

Date ___/___/___

```
# importing libraries

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# use seaborn plotting default

import seaborn as sns
sns.set()

# we are working with perfectly linear dataset

from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=50, centers=2,
                  random_state=0, cluster_std=0.60)

plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='winter')
```



Saathi

Date ___/___/_____

```
from sklearn.svm import SVC # Support vector Classifier
model = SVC(kernel='linear', C=1)
model.fit(X, y)
```

> SVC(C=1, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, ~~gamma='auto_deprecated'~~, gamma='auto_deprecated', kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.01, verbose=False)

```
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot a decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary & margins
    ax.contour(X, Y, P, colors='k', levels=[-1, 0, 1],
               alpha=0.5, linestyles=['--', '-', '--'])
```

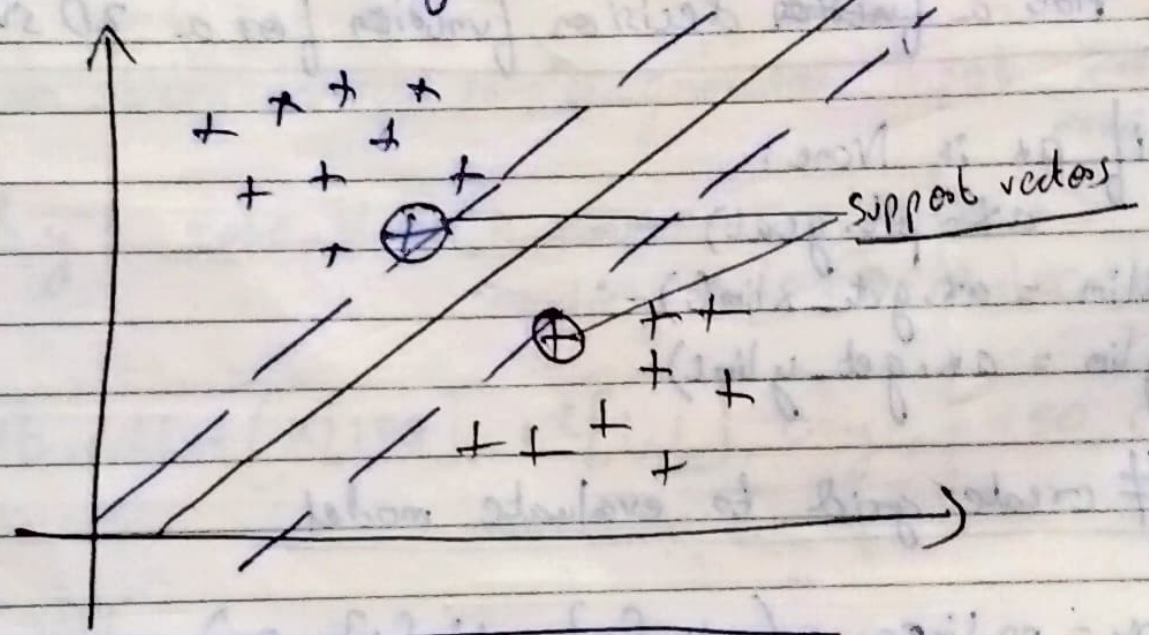
Saath

Date ___/___/_____

#plot_support vetors

```
if plot_support:
    ax.scatter(model.support_vectors_[:,0],
               model.support_vectors_[:,1],
               s=300, linewidth=1, facecolors='none');

ax.set_xlim(xlim)
ax.set_ylim(ylim)
```

```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='winter')
plot_svc_decision_function(model)
```

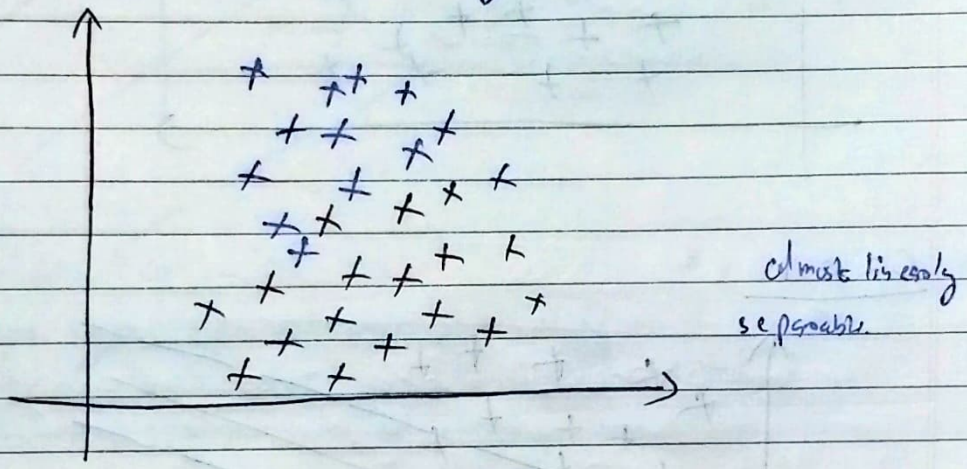


Saathi

Date ___/___/_____

Working with Almost linearly Separable datasets.

```
X, y = make_blobs(n_samples=100, centers=2,
                  random_state=0, cluster_std=1.2)

plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='winter')
```

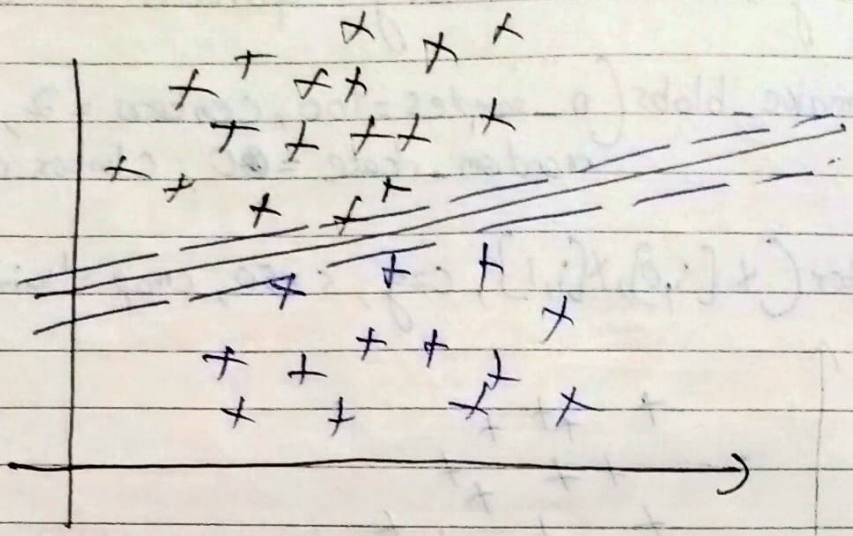


We will train svm twice with C=10 & 0.1. In SVM classifier hyperparameter there are 2 errors that define the whole svc. C handles the classifier error.

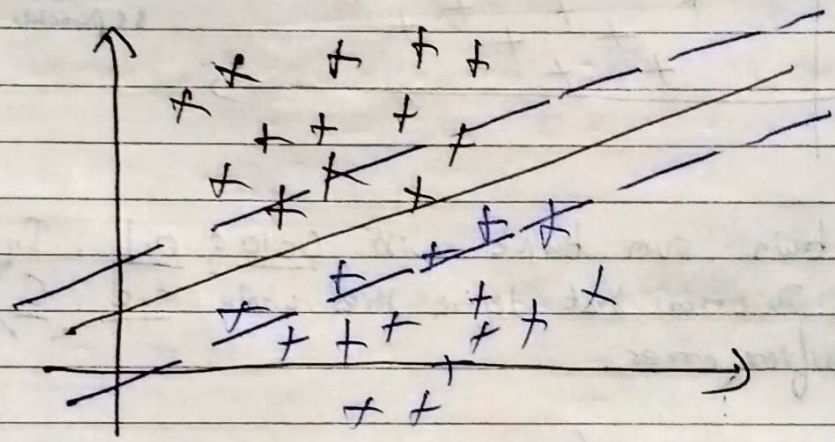
```
X, y = make_blobs(n_samples=100, centers=2,
                  random_state=0, cluster_std=0.8)

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)

for axi, C in zip(ax, [10.0, 0.1, 100, 0.01]):
    model = SVC(kernel='linear', C=C).fit(X, y)
    axi.scatter(X[:,0], X[:,1], c=y, s=50, cmap='winter')
    plot_svc_decision_function(model, axi)
    axi.scatter(model.support_vectors_[:,0],
                model.support_vectors_[:,1],
                s=300, lw=1, facecolors='none')
    axi.set_title('C={0:0.1f}'.format(C), size=14)
```

Date ___ / ___ / _____

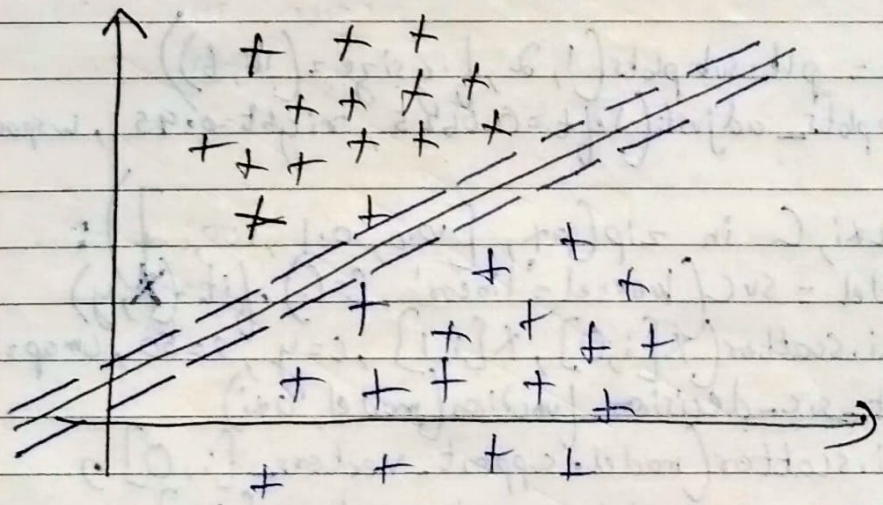
C = 10.0



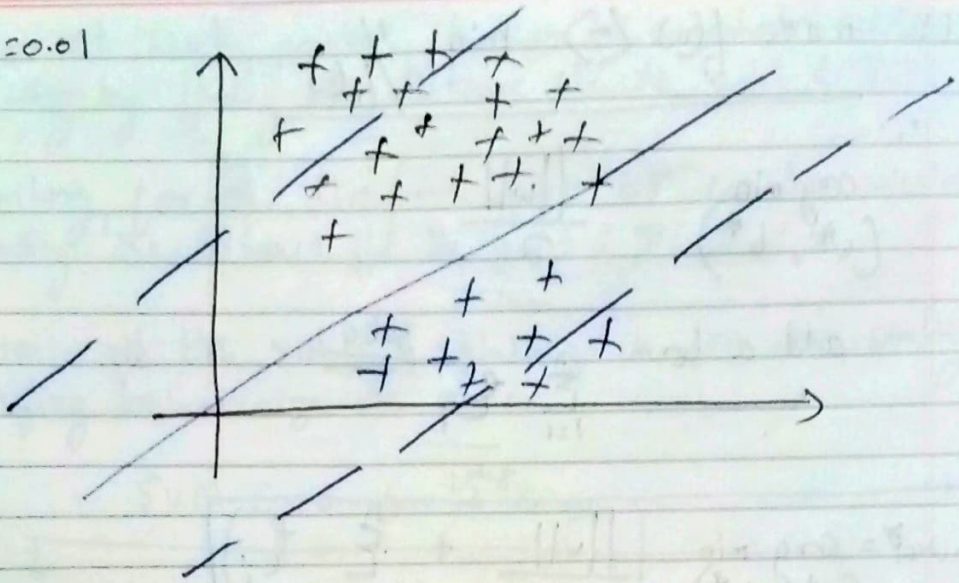
C = 0.1



C = 100



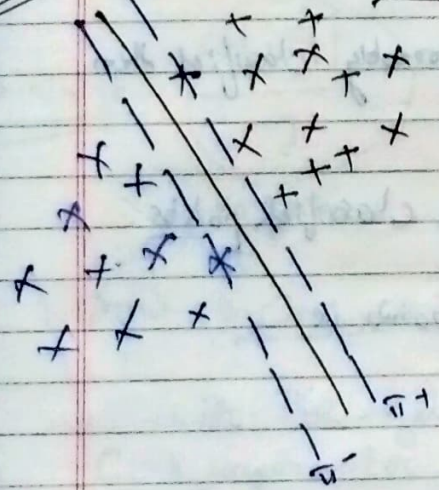
C=0.01



I hope you can observe the difference in all 4 graphs.

Theory

Summary of Hard Margin

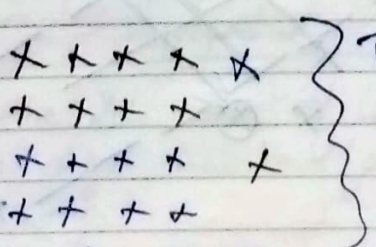


$$\underset{(w^*, b^*)}{\arg\max} \frac{2}{||w||}$$

s.t. $y_i (w^T x_i + b) \geq 1$

means all positive points should lie above π^+ & all (-ve) points should lie below π^-

But in real world we will not get a perfectly linearly separable dataset. At max we can see almost linear.



This will not be handled by Hard margin.

So in order to handle all this we have another logic called soft margin.

Soft margin SVC

Date

NOTE: $\max f(x) \Leftrightarrow \min \frac{1}{f(x)}$

$\therefore$

$$\underset{(w^*, b^*)}{\arg\min} \frac{||w||}{2}$$

add a term $\sum_{i=1}^{n} \xi_i$ ← zeta

ξ_i : zeta$_i$

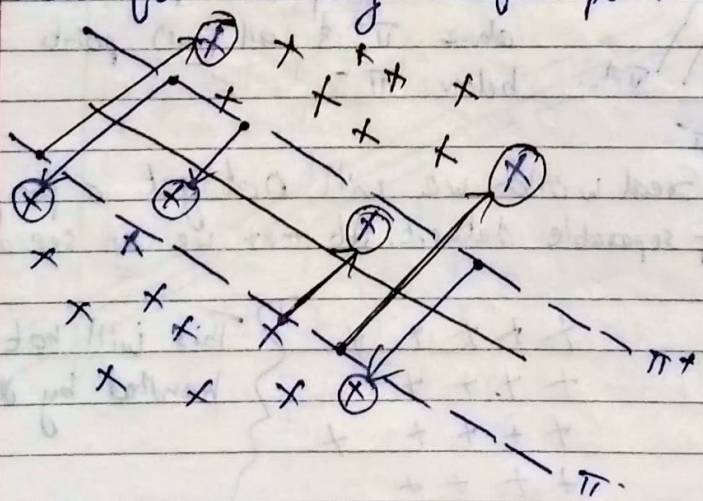
$$\text{loss}_{\text{function}} = \underset{(w^*, b^*)}{\arg\min} \left(\frac{||w||}{2} + \sum_{i=1}^{n} \xi_i \right)$$

we will be calculating zeta for every point

What is zeta?

if the points are correctly classified then $\xi_i = 0$ / zeta$_i$ = 0 i.e. zeta$_i$ $\xi_i = 0$ for correctly classified points

Now for incorrectly classified points i.e.



Saathi

Date ___/___/___

(black means +ve)

for all black points ξ_i can be ~~calculated~~ calculated by finding the distance b/w the point & π^+,

Similarly, for all blue/-ve points ξ_i can be calculated by finding the distance b/w the point & π^-

Then all the values & minimize it. We are simply trying to minimize the error

SVM Error has 2 things

Margin Error

To increase the width

if margin is big then margin error is small & vice versa

Classification Error

$\varepsilon \xi_i$

minimize $\frac{\|w\|}{2}$ + $\varepsilon \xi_i$

$\frac{\|w\|}{2}$ margin error

$\varepsilon \xi_i$ classification error.

Now One more thing

with the zeta function, we multiply $\varepsilon \xi_i$ with C hyperparameter. So the loss function would look like

$$\text{loss function} = \frac{\|w\|}{2} + C \cdot \sum_{i=1}^{n} \text{zeta}_i \; ; \; C > 0$$

(C: hyperparameter)

if we ~~increase~~ increase the value of C to a large value we are telling the algo not to focus ~~on~~ margin in maximizing the margin

Loss function = $\underset{(w^*, b^*)}{\text{Argmin}} \frac{\|w\|}{2}$ + $C \sum_{i=1}^{n} \xi_i$

Regularization ↓ hinge loss

Kernel Trick SVM For nonlinear data

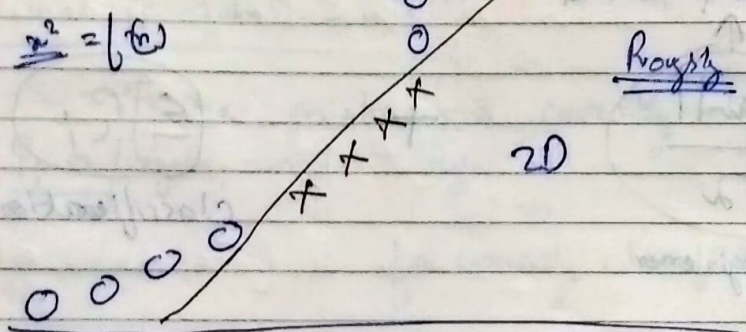
1D

O O O O X X X X O O O O

we cannot seperate X & O using 2 points

Inorder to classify using SVM, we convert the whole 1D into 2D as follows.

Lets say $x^2 = f(x)$



1D

O O O O X X X X O O O O

using $f(x) = x^2$ we could seprate the 1D nonlinear data set by converting into a 2D. Therefore $f(x)$ is the Kernel

$\therefore f(x)$ a) kernel function

3 types of kernel function

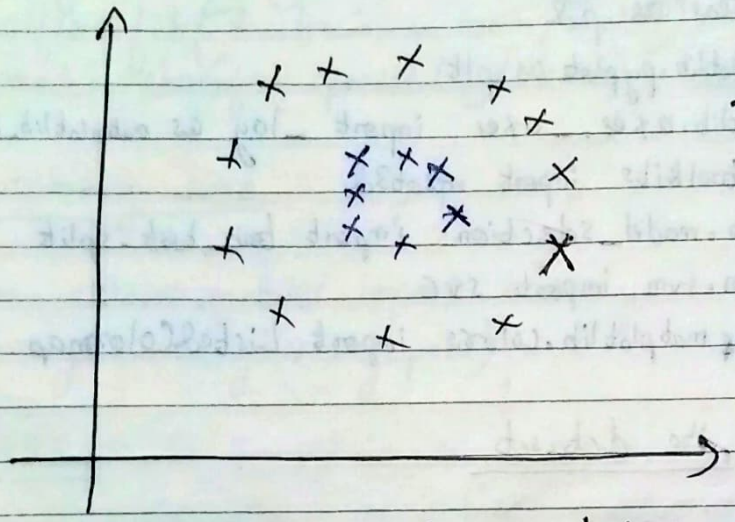
- RBF
- Polynomial $\{x^2, x^3 \dots\}$
- Sigmoid

Saathi

Date ___ / ___ / _____

The transformation from 1D to 2D is called kernel's transformation.

Another Ex:

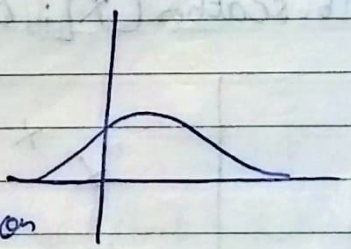


No 1 line can separate black X & blue X.

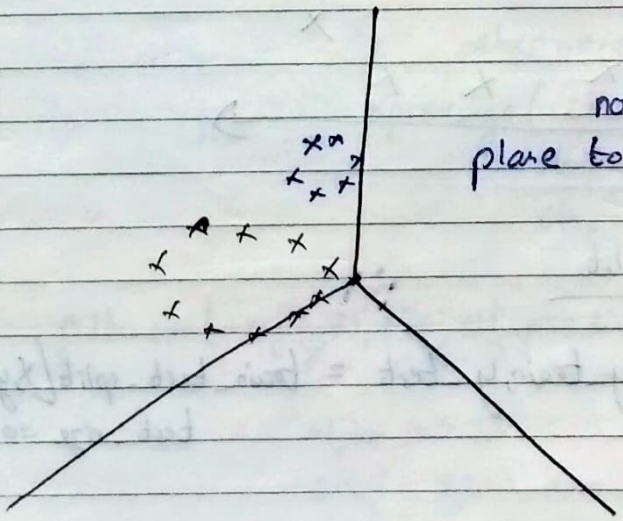
Let's say we are applying the function

$f(x) = e^{-x^2}$ → RBF

↳ The function lifts up the center



So the 2D after transformation will look like in 3D as follows



now we can find a plane to separate both of them.

Code

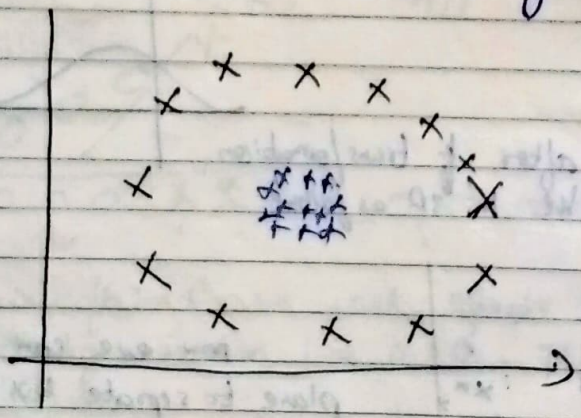
Date

Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.axes._axes import _log as matplotlib_axes_logger
from mpl_toolkits import mplot3d
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from matplotlib matplotlib.colors import ListedColormap.
```

making the dataset

```
from sklearn.datasets.samples_generator import make_circles
X,y = make_circles(100, factor=0.1, noise=0.1)
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='bwr')
```



train, test split

```
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                   test_size=0.20)
```

Date / /

appling SVC [linear]

```
classifier = SVC(kernel = "linear")
classifier.fit(x_train, y_train.ravel())
y_pred = classifier.predict(x_test)
```

accuracy score

```
from sklearn.metrics import accuracy_score
accuracy_score(ytest, y_pred)
```

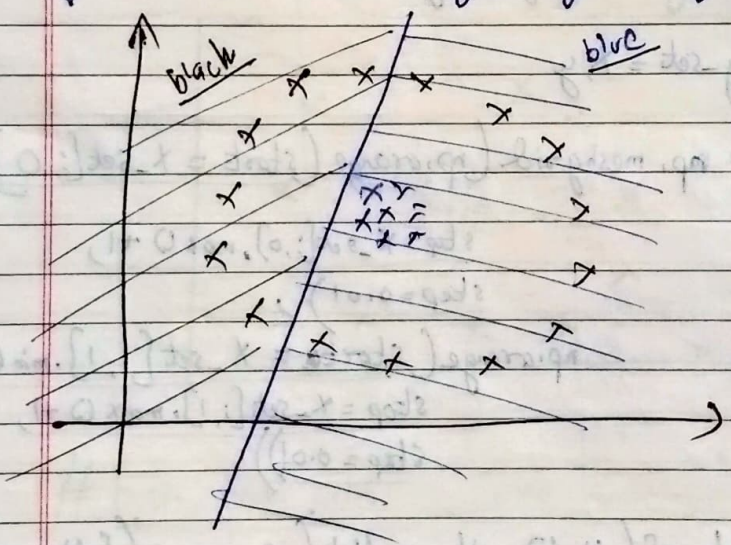
> 0.55

plotting

```
zero_one_colormap = ListedColormap(('blue', 'red'))

def plot_decision_boundary(X, y, clf):
    X_set, y_set = X, y
    X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                                   stop = X_set[:, 0].max() + 1,
                                   step = 0.01),
                         np.arange(start = X_set[:, 1].min() - 1,
                                   stop = X_set[:, 1].max() + 1,
                                   step = 0.01))
    plt.contourf(X1, X2, clf.predict(np.array([X1.ravel(),
                 X2.ravel()]).T).reshape(X1.shape),
                 alpha = 0.75
                 cmap = zero_one_colormap)
```

```
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 = (zero_one_colormap)(i),
                label = j)
plt.title("SVM. Decision Boundary")
plt.xlabel("X1")
plt.ylabel("X2")
plt.legend
return plt.show()
```

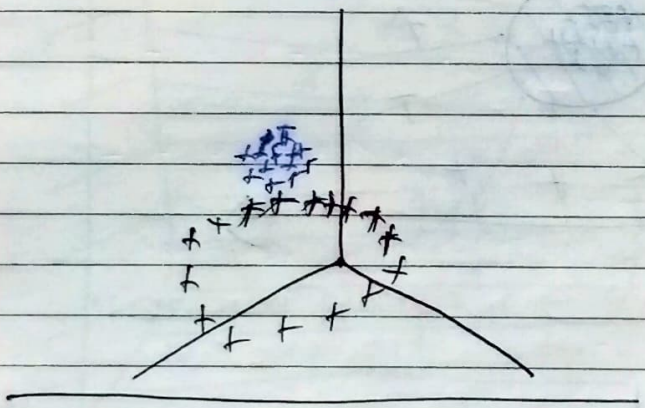
plot_decision_boundary(X, y, classifier)



Date ___ / ___ / ____

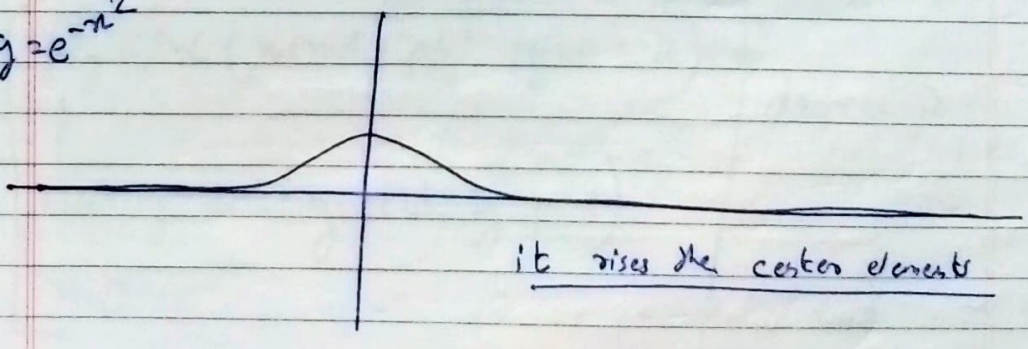
~~plot~~ transforming from 2D to 3D

```
def plot_3d_plot(X, y):
    r = np.exp(-(X**2).sum(1))        # radial basis function
    ax = plt.subplot(projection='3d')
    ax.scatter3D(X[:,0], X[:,1], r, c=y, s=100, cmap='bwr')
    ax.set_xlabel('x1')
    ax.set_ylabel('x2')
    ax.set_zlabel('y')
    return ax

plot_3d_plot(X, y)
```



Radial Basial function (e^{-r^2})

$g = e^{-r^2}$



it rises the center elements

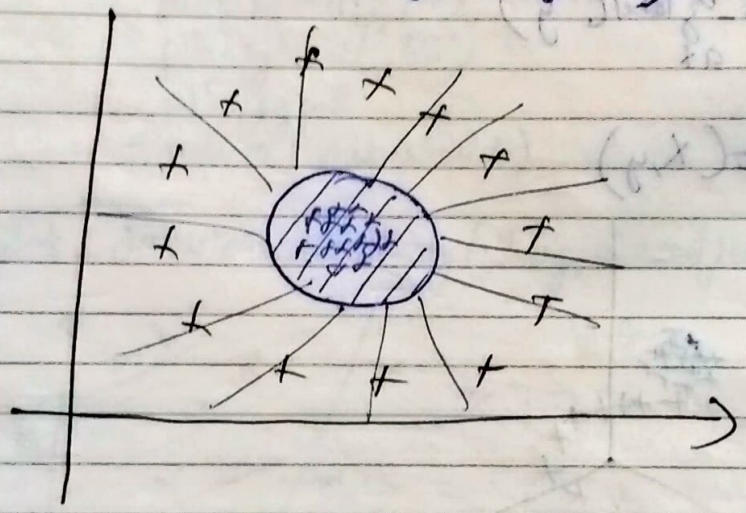
Date / /

SVM kernel's tricky model [RBF]

```
rbf = svc(kernel='rbf')
rbf.fit(x_train, y_train)
y_pred = rbf.predict(x_test)
accuracy_score(y_test, y_pred)
```

> 1.0

```
plot_decision_boundary(x, y, rbf)
```



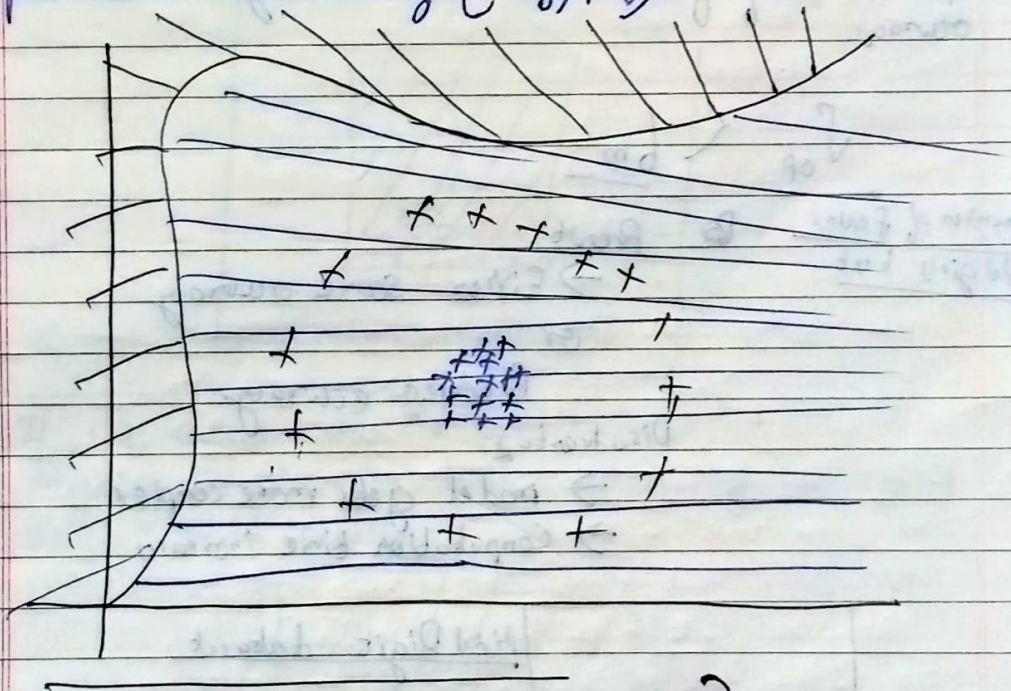
Best model

Saathi

Date ___/___/_____

SVM kernel's trick model [Poly]

```
poly = svc(kernel='poly') # degree=3(default)
poly.fit(x_train, y_train)
y_pred = poly.predict(x_test)
accuracy_score(y_test, y_pred)
>0.45
plot_decision_boundary(x, y, poly)
```



```
poly = svc(kernel='poly', degree=2)
:
accuracy_score(y_test, y_pred)
>1.0
```

degree=2