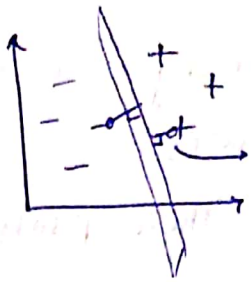
Support Vector Machine:

used for binary classifier

separates into two groups.



these perpendicular distances are the greatest you can come up with.

aim: to find best separating hyperplane or a decision boundary.

hyperplane: separates two sets of points
↳ a surface that acts like a boundary in an n-dim space.

For final classification, we just need to check which side of the boundary does the new image / input falls on.

Use cases:

1. classification
2. regression (Time series prediction)] supervised
3. outlier detection
4. clustering] unsupervised

~~Hyper-parameters:~~

1. Kernel: kernel helps us in finding a hyperplane in higher dimensional space without increasing the computational cost.
2. Hyperplane: decision boundaries that classify data points into their respective classes in multi-dimensional space.

SVM's way to find the best line:

Acc. to SVM algo, we find the points closest to the line from both the classes.

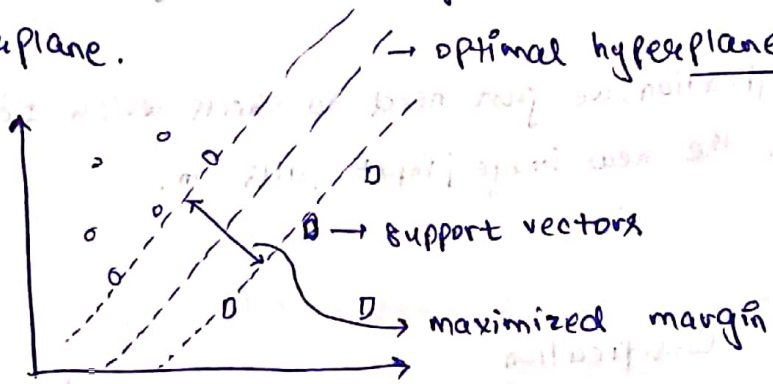
↳ These points are called "support vectors".

↳ we compute the distance between the line and the support vectors.

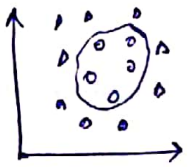
↳ This distance is called "margin".

So, our aim / goal is to maximize the margin.

The hyperplane for which the margin is maximum is called the optimal hyperplane.



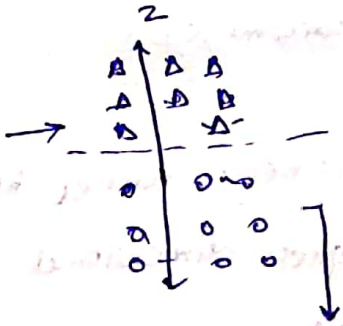
Thus SVM tries to make a decision boundary in such a way that the separation between the two classes is as wide as possible.



non linearly separable.

→ we cannot draw a straight line.

→ but this data can be converted to linearly separable data in higher dimension.



$z = x^2 + y^2$ (square of distance from origin)

↙ new dimension

Thus we can classify data by adding an extra dimension to it so that it becomes linearly separable and then projecting the decision boundary back to original dimensions using mathematical transformation.

Note: finding the correct transformation for any given dataset isn't easy.
we use kernels in sklearn to do that.

A hyperplane in an n-dimensional Euclidean space is a flat (n-1) dimensional subset of that space that divides the space into two disconnected parts.

Eg: line → 1 dim
point → 0 dim → hyperplane for the line.

```
from sklearn.svm import svc
clf = SVC(kernel = 'linear')
clf.fit(X, y)
```

---

optimal hyperplane will be the one with the biggest margin because a larger margin ensures that slight deviations in the data points should not affect the outcome of the model.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.

Maths:

Point: $x \in R^D$

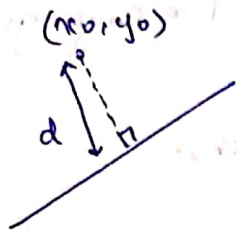
$\phi: R^D \rightarrow R^M$ ← transform into higher dimension.

$\phi(x) \in R^M$.

Decision boundary: $H: w^T \phi(x) + b = 0$

(n-1 dim. separator)

distance measure:



Line: $ax + by + c = 0$

distance of point from line

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

point

$$d_H(\phi(x_0)) = \frac{|w^T \phi(x_0) + b|}{\|w\|_2}$$

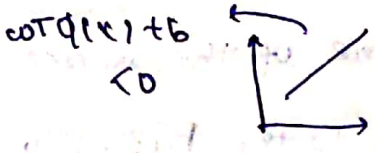
---

what are we optimizing:

case-1: Perfect separation

$$w^* = \arg\max \left[\min d_H(\phi(x_n)) \right]$$

maximize min. distance of from support vector points to our plane.



$$y_n [w^T \phi(x) + b] \rightarrow \begin{cases} >0 & \rightarrow \text{correct} \\ <0 & \rightarrow \text{incorrect} \end{cases}$$

$$w^* = \arg\max \frac{1}{\|w\|_2} \left[\min y_n [w^T \phi(x_n) + b] \right]$$

↳ distance of closest point to H.

Primal form of SVM = $\min_{w} \frac{1}{2} \|w\|_2$

assumptions → data easily separable → 100% accurate classification, → can lead to overfitting.

Case-2: Non-perfect separation.

New primal form : $\min \frac{1}{2} \|w\|_2 + c \cdot \sum_n \xi_n$

Penalty term.

for correct prediction | classification → $\xi = 0$ [zero penalty]

" incorrect " " → $\xi > 1$

$\xi_n \geq 0 \ \forall n$

[convex quadratic optimization]

c = hyperparameter.

$c = 0 \Rightarrow$ less complex boundary. → may lead to underfit.

$c = \infty \Rightarrow$ even a small error is highly penalized.

more strict | complex boundary → may lead to overfitting.

here we need to know ϕ:

use kernels to avoid use of 'ϕ'.

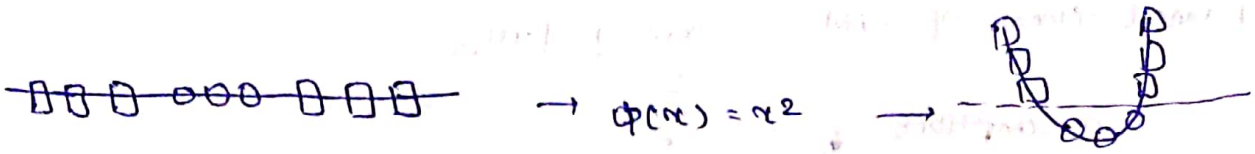
use Lagrange multipliers.

Kernelization:

Function with 2 properties:

① Symmetry

② +ve semi-definite.



linearly non-separable

Kernel Trick:

We have seen how higher dimensional transformations can allow us to separate data in order to make classification predictions.

Not all the function count as a kernel trick. The kernel function has a special property that makes it particularly useful in optimizing non-linear support vector classifiers is called a "kernel trick".

There might be many features in the data and applying transformations that involve many polynomial combination of these features will lead to extremely high computation costs.

Kernel methods represent the data only through a set of pairwise similarity comparisons between the original data (x) instead of explicitely applying the transformation $\phi(x)$.

Here,
data set (X) is represented by an $n \times n$ kernel matrix of pairwise similarity comparisons.

The kernel function acts as a modified dot product