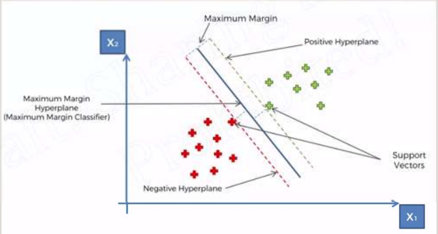
SUPPORT VECTOR MACHINES (SVM)

SUPPORT VECTOR MACHINES (SVM)

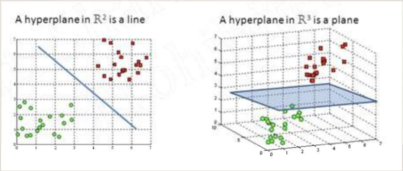
- **Support Vector Machines (SVM)** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- SVMs are based on the idea of finding a **hyperplane** that best divides a dataset into two classes, as shown in the image below.



http://arun-aiml.blogspot.com/2017/07/support-vector-machine-svm.html

WHAT IS A HYPERPLANE?

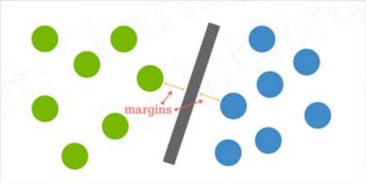
- For a classification task with only two features, you can think of a hyperplane as a line that linearly separates and classifies a set of data. For, three features, it's a plane.
- Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified.
- So when new testing data are added, whatever side of the hyperplane it lands will decide the class that we assign to it.



https://medium.com/swlh/support-vector-machines-and-the-kernel-trick-f946991ebc76

HOW DO WE FIND THE RIGHT HYPERPLANE?

- How do we best segregate the two classes within the data?
- *The distance between the hyperplane and the nearest data point from either set is known as the* ***margin****.*
- The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.



https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

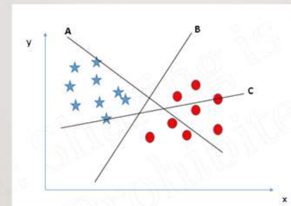
HOW CAN WE IDENTIFY THE RIGHT HYPER-PLANE?

- You need to remember a thumb rule to identify the right hyper-plane:

  "Select the hyper-plane which segregates the two classes better".

THE RIGHT HYPERPLANE

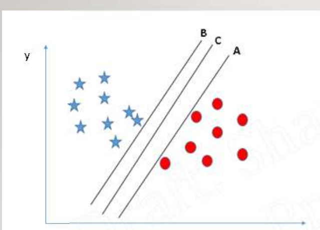
- Here, we have three hyperplanes (A, B and C). Now, identify the right hyperplane to classify star and circle.



- Hyperplane "B" has excellently performed this job.

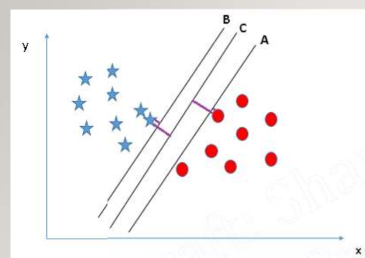
THE RIGHT HYPERPLANE

- Here, we have three hyperplanes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyperplane?



Here, maximizing the distances between nearest data point (either class) and hyperplane will help us to decide the right hyperplane. This distance is called as Margin.

THE RIGHT HYPERPLANE



We can see that the margin for hyperplane C is high as compared to both A and B. Hence, we name the right hyperplane as C. Another lightning reason for selecting the hyperplane with higher margin is robustness. If we select a hyperplane having low margin then there is high chance of misclassification.

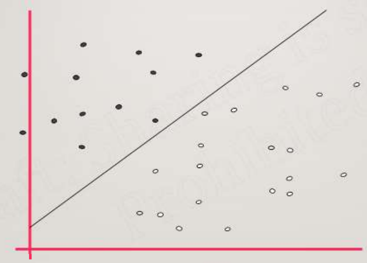
LINEAR CLASSIFIERS

denotes +1
denotes -1

How would you classify this data?

LINEAR CLASSIFIERS

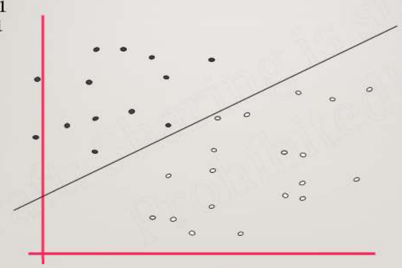
denotes +1
denotes -1



How would you classify this data?

LINEAR CLASSIFIERS

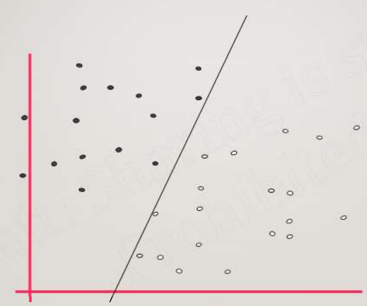
denotes +1
denotes -1



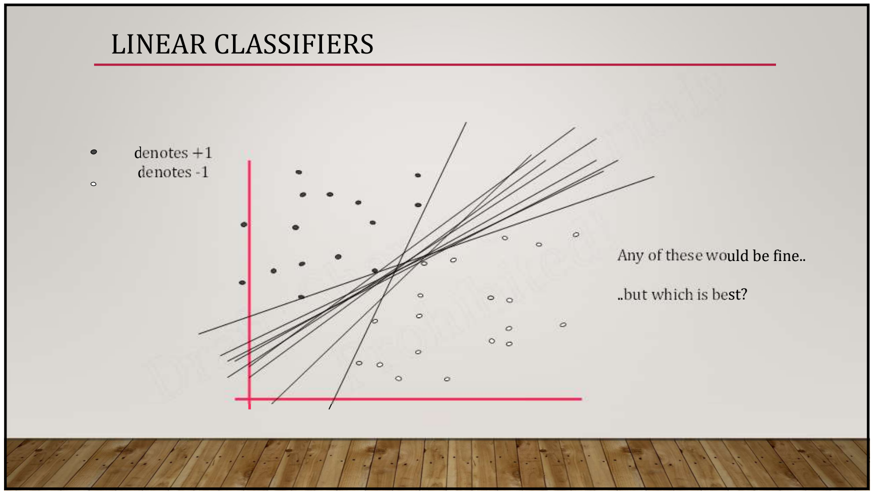
How would you classify this data?

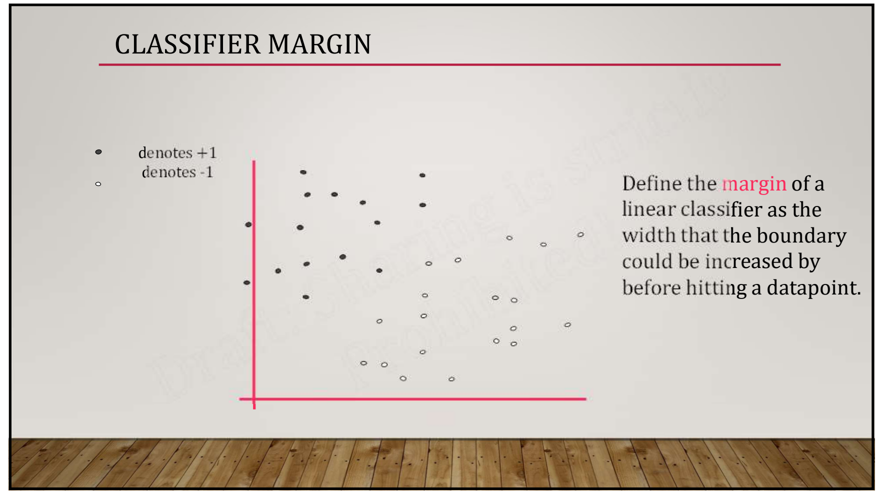
LINEAR CLASSIFIERS

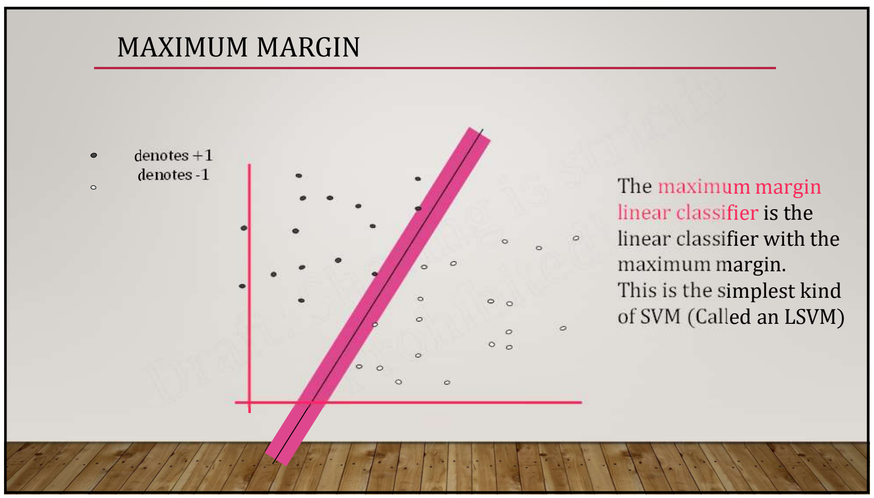
denotes +1
denotes -1

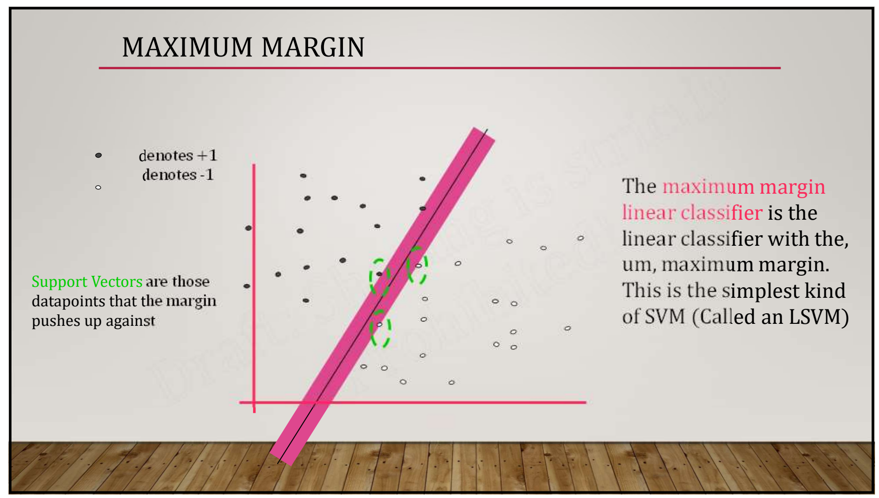


How would you classify this data?


LINEAR CLASSIFIERS
denotes +1
denotes -1
Any of these would be fine..
..but which is best?

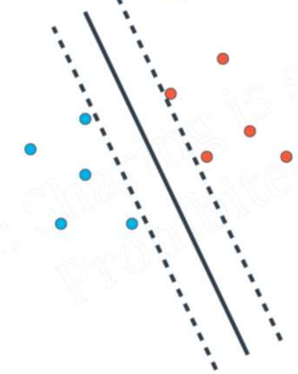

CLASSIFIER MARGIN
denotes +1
denotes -1
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.


MAXIMUM MARGIN
denotes +1
denotes -1
The maximum margin linear classifier is the linear classifier with the maximum margin.
This is the simplest kind of SVM (Called an LSVM)


MAXIMUM MARGIN
denotes +1
denotes -1
Support Vectors are those datapoints that the margin pushes up against
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Error function(s)

Split data - separate lines



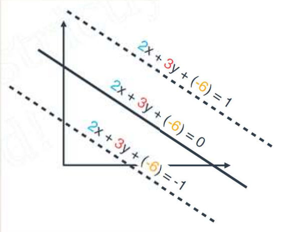
SVM LINE EQUATIONS

One difference between SVMs and other linear classifiers is that SVMs use two parallel lines instead of one.

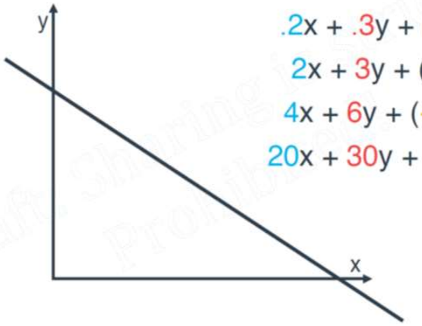
Luckily, two parallel lines have very similar equations, they have the same weights, but a different bias.

We use one central line as a frame of reference L with equation $w_1x_1 + w_2x_2 + b = 0$, and construct two lines, one above it and one below it, with the respective equations

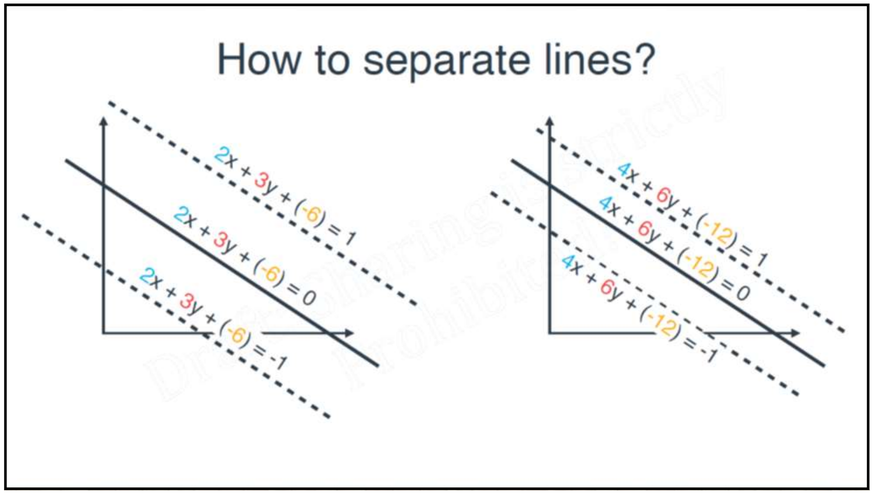
L: $w_1x_1 + w_2x_2 + b = 0$,
L+: $w_1x_1 + w_2x_2 + b = 1$,
L-: $w_1x_1 + w_2x_2 + b = -1$.

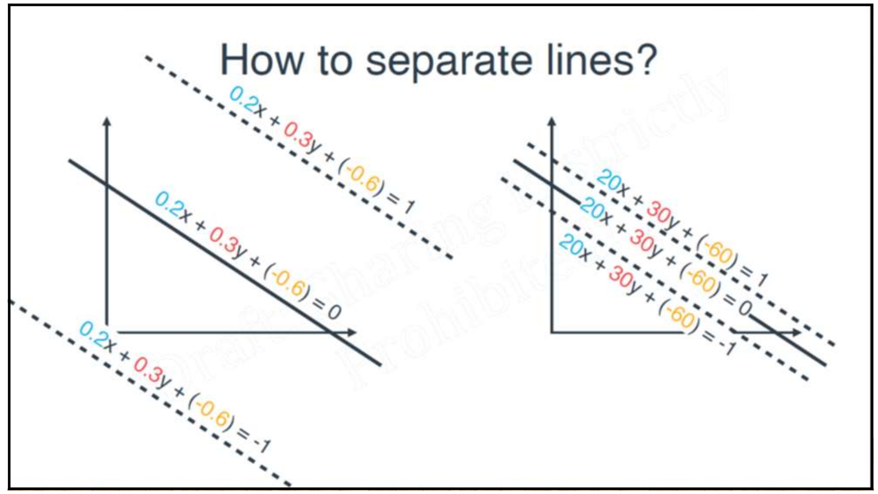


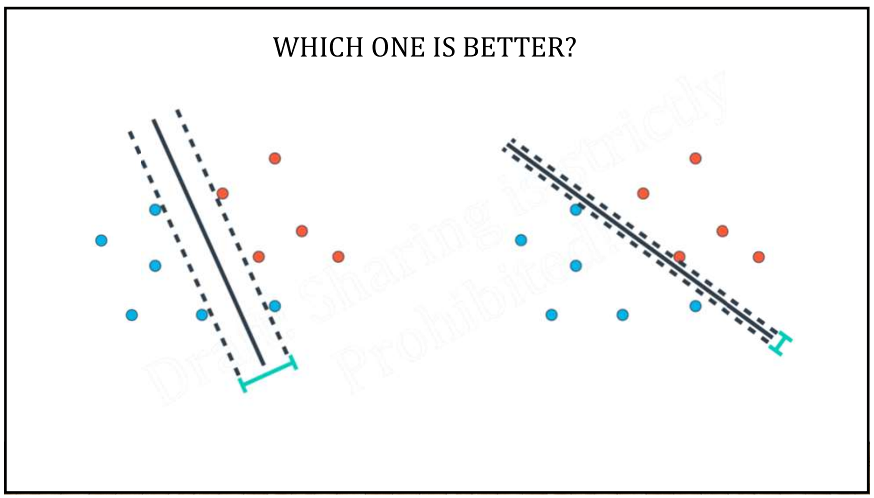
How to separate lines?

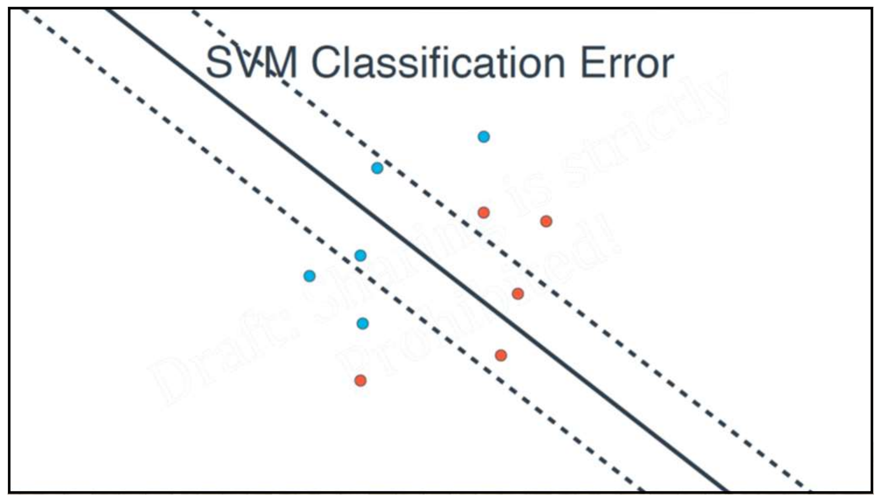


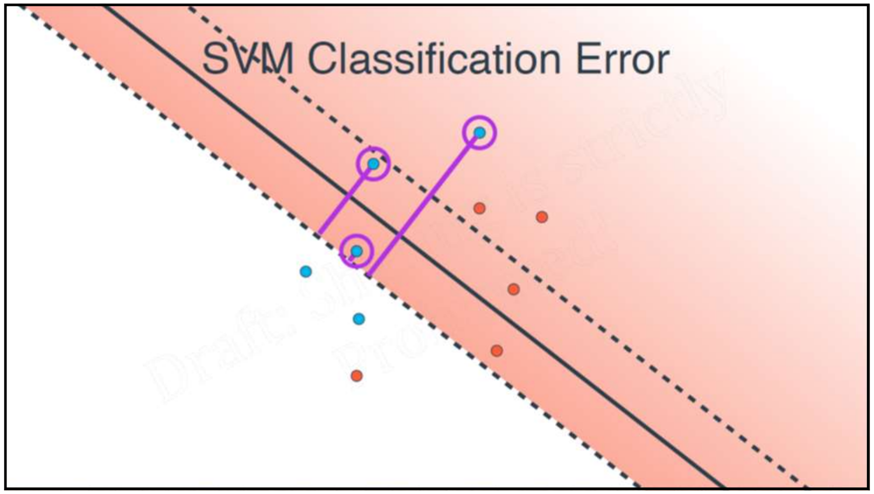
$.2x + .3y + (-.6) = 0$
$2x + 3y + (-6) = 0$
$4x + 6y + (-12) = 0$
$20x + 30y + (-60) = 0$

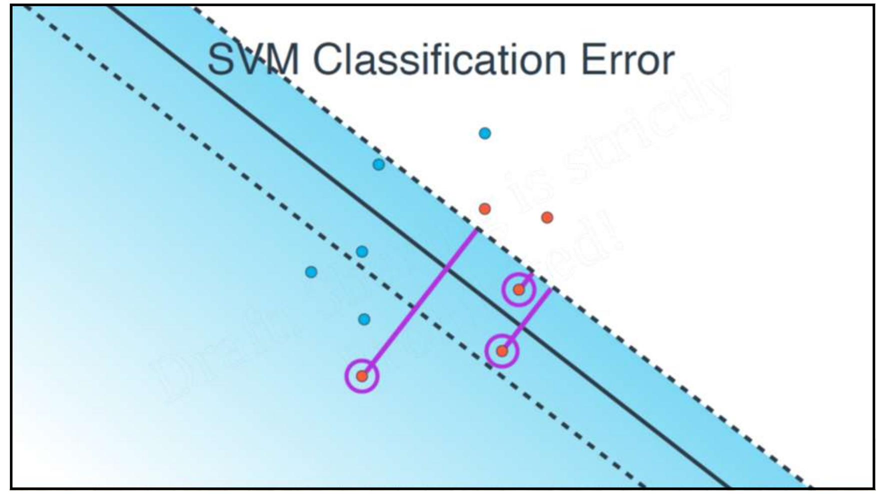

How to separate lines?
2x + 3y + (-6) = 1
2x + 3y + (-6) = 0
2x + 3y + (-6) = -1
4x + 6y + (-12) = 1
4x + 6y + (-12) = 0
4x + 6y + (-12) = -1

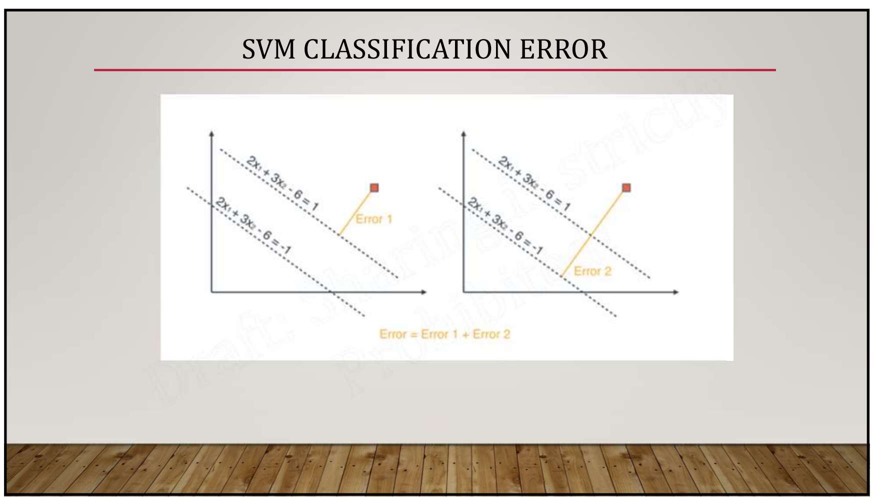

How to separate lines?
0.2x + 0.3y + (-0.6) = 1
0.2x + 0.3y + (-0.6) = 0
0.2x + 0.3y + (-0.6) = -1
20x + 30y + (-60) = 1
20x + 30y + (-60) = 0
20x + 30y + (-60) = -1

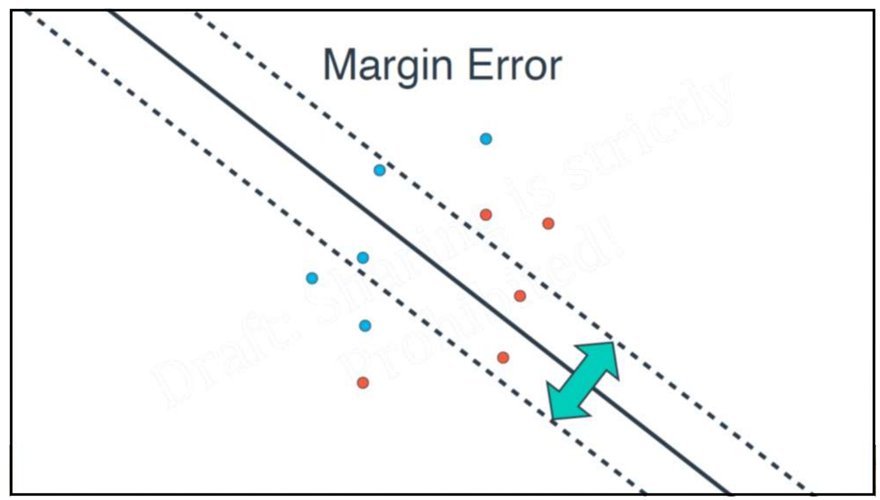

WHICH ONE IS BETTER?


SVM Classification Error

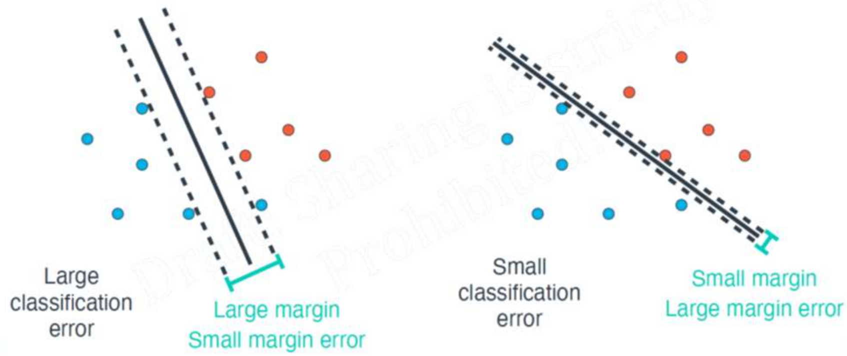

SVM Classification Error


SVM Classification Error

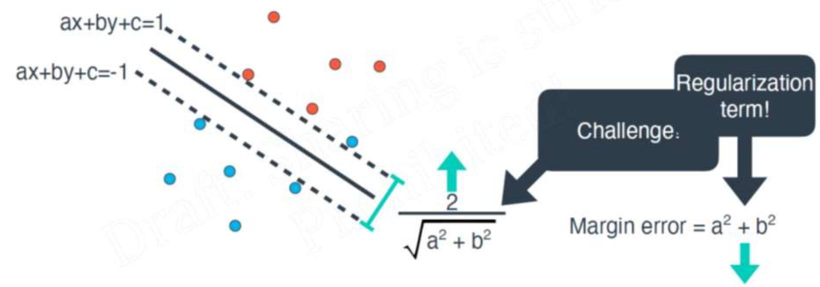

SVM CLASSIFICATION ERROR
2x1 + 3x2 - 6 = 1
2x1 + 3x2 - 6 = -1
Error 1
2x1 + 3x2 - 6 = 1
2x1 + 3x2 - 6 = -1
Error 2
Error = Error 1 + Error 2


Margin Error

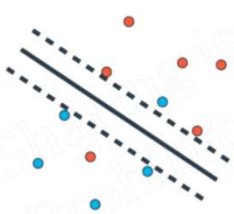
Margin Error



Margin Error

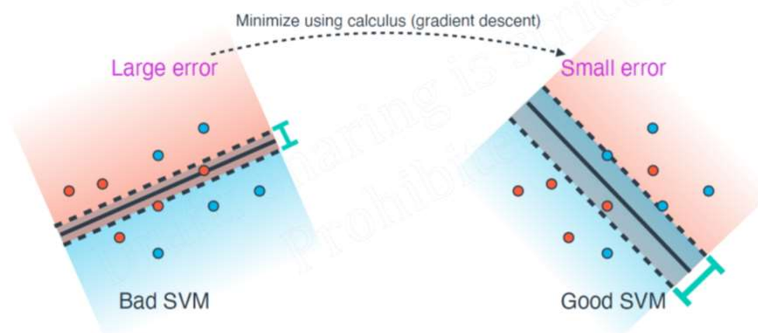


SVM Error

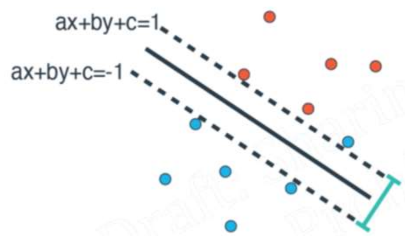


Gradient Descent

Same as the SVM trick!



Challenge - Gradient Descent



Margin error = $a^2 + b^2$

expanding factor!

dError/da = 2a

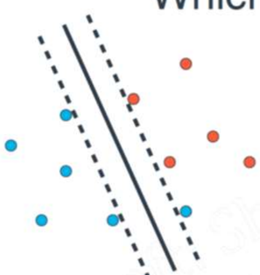
dError/db = 2b

$$a \longrightarrow a - \eta\, 2a = a\boxed{(1 - 2\eta)}$$

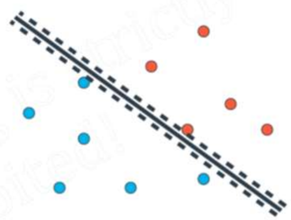
$$b \longrightarrow b - \eta\, 2b = b\boxed{(1 - 2\eta)}$$

The C Parameter

Which line is better?

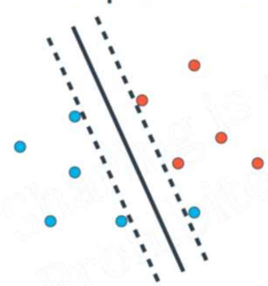


Classification Error + Margin Error

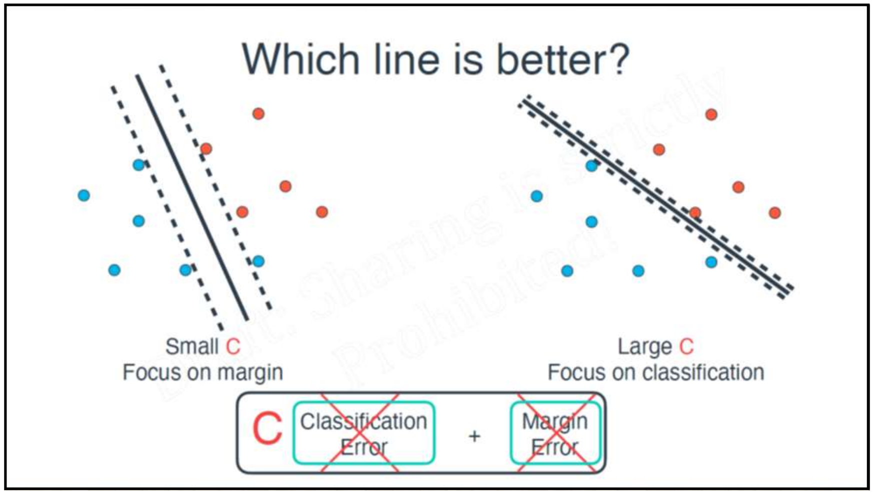


Classification Error + Margin Error

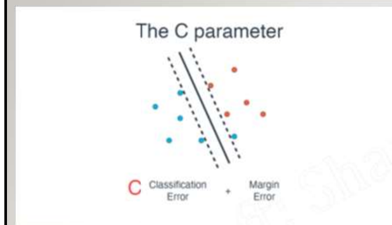
The C parameter

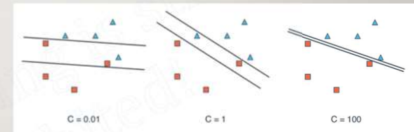


C Classification Error + Margin Error

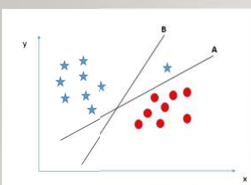


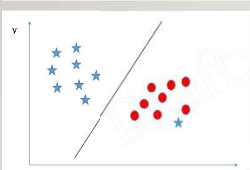
THE 'C' PARAMETER





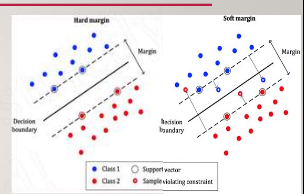
IDENTIFY THE RIGHT HYPERPLANE

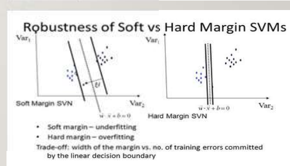




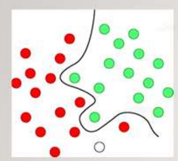
Hard Margin Classifier or Maximal Margin Classifier:
No data points are allowed in the margin areas.

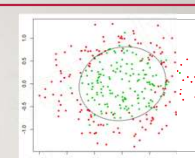
Soft Margin Classifier or Support Vector Classifier:
Data points are allowed in the margin areas.





SUPPORT VECTOR MACHINE (SVM)



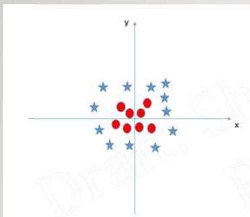


Support Vector Machines:
When non-linear boundary is used for the classification purpose, it is called Support Vector Machines (SVMs). It is an extension of the Support Vector Classifier and done by enlarging the feature space in a specific way, using *kernels*.

https://intellipaat.com/blog/tutorial/machine-learning-tutorial/svm-algorithm-in-python/

FIND THE HYPERPLANE

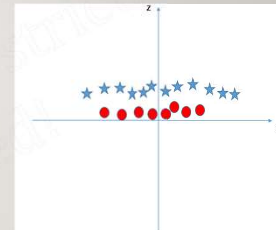
- In the scenario below, we can't have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyperplane.



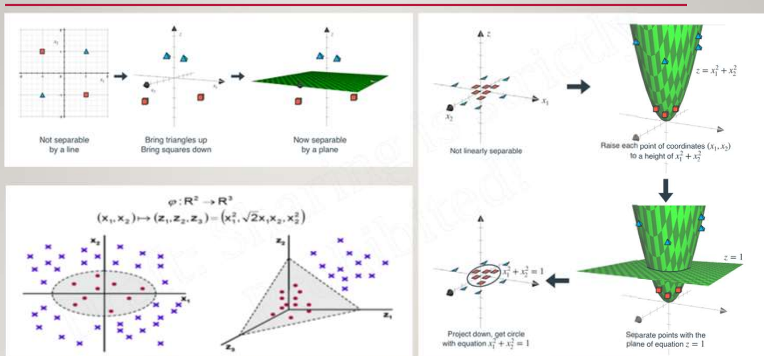
SVM can solve this problem. It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$.

FIND THE HYPERPLANE

- Now, let's plot the data points on axis x and z:
- In this plot, points to consider are:
  - All values for z would be positive always because z is the squared sum of both x and y
  - In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

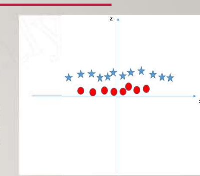


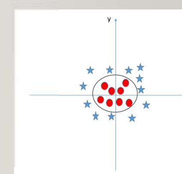
KERNEL FUNCTİONS



KERNEL TRICK

- In SVM, it is easy to have a linear hyperplane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyperplane.
- No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts non separable problem to separable problem, these functions are called **kernels**.
- It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.
- When we look at the hyperplane in original input space it looks like a circle:





KERNEL FUNCTİONS

There are different kernels. The most popular ones are the polynomial kernel and the radial basis function (RBF) kernel.

"Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these" (Wikipedia), just like the example above. With n original features and d degrees of polynomial, the polynomial kernel yields n ^ d expanded features.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

The format of polynomial kernel

The RBF kernel is also called the Gaussian kernel. There is an infinite number of dimensions in the feature space because it can be expanded by the Taylor Series. In the format below, The γ parameter defines how much influence a single training example has. The larger it is, the closer other examples must be to be affected (sklearn documentation).

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}, \gamma > 0$$

Keep this in mind, we will talk about it later

KERNEL FUNCTİONS

Let's look at an example:

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Here x and y are two data points in 3 dimensions. Let's assume that we need to map x and y to 9-dimensional space. We need to do the following calculations to get the final result, which is just a scalar. The computational complexity, in this case, is $O(n^2)$.

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

KERNEL FUNCTİONS

However, if we use the kernel function, which is denoted as k(x, y), instead of doing the complicated computations in the 9-dimensional space, we reach the same result within the 3-dimensional space by calculating the dot product of x -transpose and y. The computational complexity, in this case, is O(n).

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^T\mathbf{y})^2 \\ &= (x_1y_1 + x_2y_2 + x_3y_3)^2 \\ &= \sum_{i,j=1}^{3} x_i x_j y_i y_j \end{aligned}$$

In essence, what the kernel trick does for us is to offer a more efficient and less expensive way to transform data into higher dimensions. With that saying, the application of the kernel trick is not limited to the SVM algorithm. Any computations involving the dot products (x, y) can utilize the kernel trick.

EXAMPLES

For example let x and y be defined as **x = (x1,x2,x3)** and **y = (y1,y2,y3).**

The mapping to 9 dimensions would be
f(x) = (x1x1,x1x2,x1x3,x2x1,x2x2,x2x3,x3x1,x3x2,x3x3)

We can define a kernel which would be equivalent to the above equation
K(x,y) = dot(x,y) = x.y = (x^Ty)^2

Let.
x = (1, 2, 3)
y = (4,5,6).

Then:
f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)
f(y) = (16,20,24,20,25,30,24,30,36)

Calculating <f(x), f(y)> , gives us
16 + 40 + 72 + 40 + 100+ 180 + 72 + 180 + 324 = 1024

Instead of doing so many calculations, if we apply the kernel instead:
K(x,y) = (4 + 10 + 18) ^2 = 32² = 1024

KERNEL FUNCTİONS

- Consider polynomials of degree q:

$$K(\mathbf{x},\mathbf{y}) = \left(\mathbf{x}^T\mathbf{y}+1\right)^2$$
$$= \left(x_1y_1 + x_2y_2 + 1\right)^2$$
$$= 1 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2 + x_1^2y_1^2 + x_2^2y_2^2$$
$$\phi(\mathbf{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2\right]^T$$

Find $\varphi(\mathbf{Y})$ same way

Notice that φ(X) & φ(Y) are in 6 dimensional space. But, you don't need to convert X & Y to six dimensions. You just apply kernel function K(x, y) and get six dimensional dot product calculations of φ(X).φ(Y) in two dimensions.

(Cherkassky and Mulier, 1998)

KERNEL FUNCTİONS

- Let's look at the previous example in a bit more detail

$$\mathbf{x} \to \phi(\mathbf{x}) = [x_1^2 \;\; x_2^2 \;\; \sqrt{2}x_1x_2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2 \;\; 1]$$

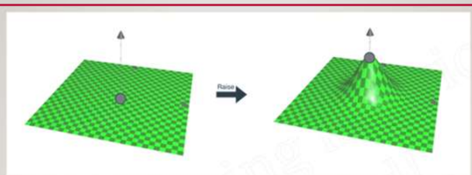
- The SVM classifier deals only with inner products of examples (or feature vectors). In this example,

$$\begin{aligned}\phi(\mathbf{x})^T\phi(\mathbf{x}') &= x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1x_1' + 2x_2x_2' + 1 \\ &= (1 + x_1x_1' + x_2x_2')^2 \\ &= (1 + (\mathbf{x}^T\mathbf{x}'))^2\end{aligned}$$

so the inner products can be evaluated without ever explicitly constructing the feature vectors $\phi(\mathbf{x})$!

- $K(\mathbf{x},\mathbf{x}') = (1 + (\mathbf{x}^T\mathbf{x}'))^2$ is a *kernel function* (inner product in the feature space)

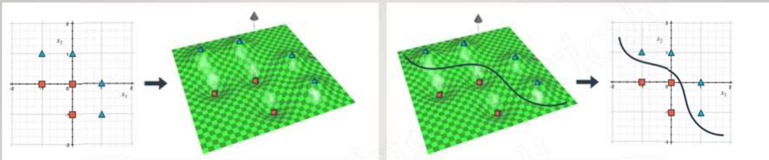
RADIAL BASIS FUNCTION (RBF) KERNEL



Imagine if you had a point in the plane, and the plane was like a blanket. Then you pinch the blanket at that point, and raise it. This is how a *radial basis function* looks like.

We can raise the blanket at any point we like, and that gives us one different radial basis function. The *radial basis function kernel* (also called rbf kernel) is precisely the set of all radial basis functions for every point in the plane.

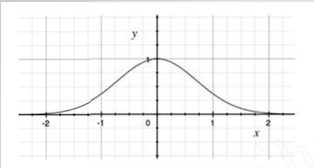
RADIAL BASIS FUNCTION (RBF) KERNEL



Lift the plane at every triangle, and push it down at every square. Then simply draw a plane at height 0, and intersect it with our surface. This is the same as looking at the curve formed by the points at height 0.

Imagine if there is a landscape with mountains and with the sea. The curve will correspond to the coastline, namely, where the water and the land meet. This gives us the curves (when we project everything back to the plane), and we obtain our desired classifier.

RADIAL BASIS FUNCTION (RBF) KERNEL



One of the simplest radial basis function has the formula:

$$y = e^{-x^2}$$

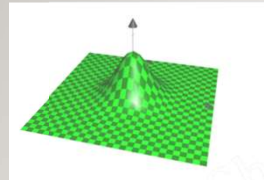
Notice that this bump happens at 0. If we wanted it to appear at any different point, say p, we simply translate the formula, and get

$$y = e^{-(x-p)^2}$$

Thus, if I want to obtain the radial basis function centered at the point 5 is precisely

$$y = e^{-(x-5)^2}$$

RADIAL BASIS FUNCTION (RBF) KERNEL



And as usual, this n-dimensional bump is centered at zero, but if we wanted it centered at the point (p_1, …, p_n), the formula is:

$$y = e^{-[(x_1-p_1)^2+\cdots+(x_n-p_n)^2]}$$

For two variables, the formula for the most basic radial basis function is:

$$z = e^{-(x^2+y^2)}$$

Again, this bump happens exactly at the point (0,0).
If we wanted it to appear at any different point, say (p, q), we simply translate the formula, and get

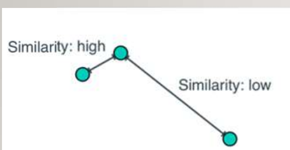
$$y = e^{-[(x-p)^2+(y-q)^2]}$$

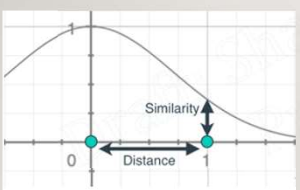
Thus, if I want to obtain the radial basis function centered at the point (2, -3), the formula is precisely

$$y = e^{-[(x-2)^2+(y+3)^2]}$$

- Text and Image Courtesy: Grokking Machine Learning by Luis Serrano

RBF KERNEL: SIMILARITY





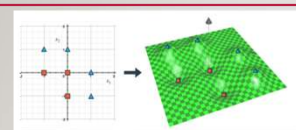
In order to build an SVM using the RBF kernel, we need one notion: the notion of similarity.

We say that two points are similar if they are close by, and not similar if they are far away.

$$similarity(p, q) = e^{-distance(p,q)^2}$$

- Text and Image Courtesy: Grokking Machine Learning by Luis Serrano

RBF KERNEL EXAMPLE



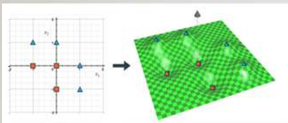
$$distance(point\ 1, point\ 6) = \sqrt{(0+1)^2+(0-1)^2} = \sqrt{2}.$$

$$similarity(point\ 1, point\ 2) = e^{-distance(q,p)^2} = e^{-2} = 0.135.$$

Point	X_1	X_2	y
1	0	0	0
2	-1	0	0
3	0	-1	0
4	0	1	1
5	1	0	1
6	-1	1	1
7	1	-1	1

- Text and Image Courtesy: Grokking Machine Learning by Luis Serrano

RBF KERNEL EXAMPLE

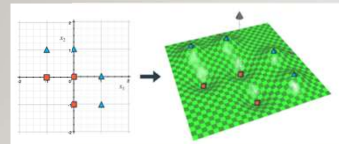


If we define the weights of x_1 and x_2 as w_1 and w_2. and The weight of Sim p is v_p, for p = 1, 2, ..., 7. and the bias is b.

A classifier that works well is: $w_1 = 0$, $w_2 = 0$, $v_1 = -1$, $v_2 = -1$, $v_3 = -1$, $v_4 = 1$, $v_5 = 1$, $v_6 = 1$, $v_7 = 1$, $b = 0$

Point	X_1	X_2	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5	Sim 6	Sim 7	y
1	0	0	1	0.135	0.135	0.135	0.135	0.368	0.368	0
2	-1	0	0.135	1	0.368	0.368	0.018	0.135	0.018	0
3	0	-1	0.135	0.368	1	0.018	0.135	0.007	0.368	0
4	0	1	0.135	0.368	0.018	1	0.135	0.368	0.007	1
5	1	0	0.135	0.018	0.135	0.135	1	0.007	0.368	1
6	-1	1	0.368	0.135	0.007	0.368	0.007	1	0	1
7	1	-1	0.368	0.018	0.368	0.007	0.368	0	1	1

RBF KERNEL EXAMPLE



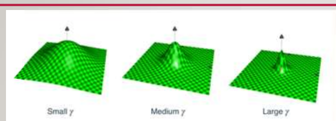
If we define the weights of x_1 and x_2 as w_1 and w_2. and The weight of Sim p is v_p, for p = 1, 2, ..., 7. and the bias is b.

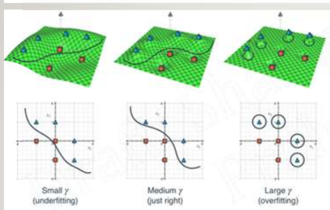
A classifier that works well is: $w_1 = 0$, $w_2 = 0$, $v_1 = -1$, $v_2 = -1$, $v_3 = -1$, $v_4 = 1$, $v_5 = 1$, $v_6 = 1$, $v_7 = 1$, $b = 0$

Point	X_1	X_2	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5	Sim 6	Sim 7	y
1	0	0	1	0.135	0.135	0.135	0.135	0.368	0.368	0
2	-1	0	0.135	1	0.368	0.368	0.018	0.135	0.018	0
3	0	-1	0.135	0.368	1	0.018	0.135	0.007	0.368	0
4	0	1	0.135	0.368	0.018	1	0.135	0.368	0.007	1
5	1	0	0.135	0.018	0.135	0.135	1	0.007	0.368	1
6	-1	1	0.368	0.135	0.007	0.368	0.007	1	0	1
7	1	-1	0.368	0.018	0.368	0.007	0.368	0	1	1

To check that this classifier works, simply add the values of the columns Sim 4, Sim 5, Sim 6, and Sim 7, then subtract the values of the columns Sim 1, Sim 2 and Sim 3. You'll notice that you get a negative number in the first three rows, and a positive one in the last four rows. This means the classifier works.

RBF KERNEL: GAMMA PARAMETER





When the gamma parameter is small, the curve formed is very wide, and when gamma is large, the curve is very narrow.

Can you Recall?

The RBF kernel is also called the Gaussian kernel. There is an infinite number of dimensions in the feature space because it can be expanded by the Taylor Series. In the format below, The γ parameter defines how much influence a single training example has. The larger it is, the closer other examples must be to be affected (sklearn documentation).

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}, \gamma > 0$$

Keep this in mind, we will talk about it later

SVM: REFERENCES & ACKNOWLEDGEMENTS

- http://www.svms.org/tutorials/Berwick2003.pdf
- http://blog.aylien.com/support-vector-machines-for-dummies-a-simple/
- https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/
- https://data-flair.training/blogs/svm-support-vector-machine-tutorial/
- https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners
- https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r/
- https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d
- https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/
- https://www.youtube.com/c/LuisSerrano

THANK YOU