UNIT 1 : Linear Classifier and Generalizations

LECTURE 2 : Linear Classifier and Perceptron

Wooooooooo !!!!!!! We are back baby !!!!! Yeahhhhhh !!!!!

After the successful completion of Fundamentals of Statistics , your pal **Scooby** is back with a brand new set of handwritten lecture notes. I found that many of you liked them and appriciated the work by thanking me personally on my email. Which by the way is :

Scooby's email : **peepeepoopoo873@gmail.com**

Feel free to point out mistakes and message me anyways

For those who do not message me...
I don't care man!!! Whataever man !!!! Screw you guys !!!! Whatever!!!

(I do care, please message me. I was just pretending to be indifferent in the above line so that you can't hurt me emotionally.) Anyways.

I tried to make these notes without any mistakes and cover most of the video material.

But as Yoda has rightly said : ' *Is human, To err* '

I however **DO NOT TAKE** any responsibilty if you fail your exams following my notes. So please check the actual lecture slides and videos if you suspect something is wrong, and mail me too.

The notes **DO NOT CONTAIN** solutions to any homework problems or in-lecture exercises. Only the problems and exercises solved in the videos are included in these notes.

Here is an interesting photo I googled

MARIJUANA MONKEY

I ran out of made up quotes so I will write very short stories instead. These may or may not resemble true events, but will definitely have a message that will help you along the journey, that is life.

STORY TIME : RICH PEOPLE LOVE

There was this dude named Chad. He chugged a lot of bear in parties. He was a trust fund kid. He used to talk about generic s**t like , how much he could bench, or the game last night. Chad frequently used to give life advice to people in his Instagram story, about living life to full, not being afraid of anything, achieving their goals etc. He didn't care about fame, he cared about the lives he touched. Atleast he caimed that.

There was this chick named Iris. Iris's father was a millionaire. She had self diaganosed herself with depression in order to appear cool on her social media. She used to caption her pics with stuff like **'I am a mess, but I am a beautiful mess'**. Obviously the pics were in grayscale tone. Cause she was sad. Maybe she understood life really well.

Chad and Iris met at a rave party. And there were sparks all around. It was as if they understood each other. It was love at first sight.

They talked about how to end world poverty and how the world would be a much better place, if everyone was happy. They went on expensive vacations and were now the power couple that infuenced people on Instagram. They talked about helping those in need, and took a selfie everytime they helped someone. They dismissed their critics as haters, and lived happily ever after.

THE END

MORAL : No matter how screwed up you are, there's always someone equally screwed for you out there. Go make this world a horrible place.

I hope these notes will help you

Let's get started

▷ Review of Basic Concepts

Lets start this lecture by briefly revising concepts from previous Lecture

> Feature vector : X

For each and every case that we had, we had some properties that were inherent to the case. Like for eg. Every movie that we had earlier had a genre, Imdb rating, budget, cast etc.

We for our purposes wanted to predict something on basis of these features. Hence for each case we chose 'd' features that were characteristic of of that case and stacked them in a vector with each entry being a number (because thats what machines understand)

This vector for the ith movie/case was called the feature vector

$$X^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix}_{d\times 1} \in \mathbb{R}^d \quad \text{(Generally)}$$

$x_j^{(i)}$: jth feature of ith case

> Label : $y^{(i)}$

This was the thing that we wanted to predict.

We are talking about supervised learning here, which means we will have Labelled data. (At least train data will be Labelled)

To each case from the train data we will have a label associated & for each case in the test data we want to predict the Label.

For the movie example we are studying each vector could have one of the two labels

$$y^{(i)} = \begin{cases} +1 & \text{: indicating viewer would watch ith movie} \\ -1 & \text{: indicating viewer wouldn't watch ith movie} \end{cases}$$

> Training Set : S_n

Its the set of examples that we feed our algorithm in order to train it. Again we are doing supervised learning which means the training data that we have will be Labelled. And we will have n examples to feed

Therefore mathematically we write

$$S_n = \{(x^{(i)}, y^{(i)}),\ i = 1, 2, 3 \ldots n\}$$

ie set of all pairs (feature vector i, label i) for all n cases

> Classifier : h()

It is simply a function or mapping from $\mathcal{X}$: ie the space of feature vectors to the space of labels. For the movie example we had

$$h : \mathbb{R}^2 \longrightarrow \{-1, +1\}$$

The idea was simple
we will have a function h( ) that will take as input any feature vector X and give us out a corresponding label of +1 or −1. In order to do that we will first need to train the model. This function h( ) we have will infact depend on parameter θ (which can be multidimensional), and using the training data, we will estimate θ's to find an optimal h( )
The we will predict labels using this optimal h( ).

> Training Error $\mathcal{E}_n(h)$

In order to see how our classifier performed wrt training data, we needed some sort of metric. The idea we had was pretty simple : Just look at what fraction of points in the training set are misclassified by h()
So we introduced the double bracket notation

$$[\![\text{Expression}]\!] = \begin{cases} 1 & \text{if Expression is True} \\ 0 & \text{if Expression is False} \end{cases}$$

If for the ith case in training data

$h(x^{(i)}) \neq y^{(i)}$: then h committed an error
$h(x^{(i)}) = y^{(i)}$: then h did not commit an error

So in order to find the fraction of error committed we have

$$\mathcal{E}_n(h) = \frac{1}{n} \sum_{i=1}^{n} [\![h(x^{(i)}) \neq y^{(i)}]\!]$$

> Test Error and Test Set

Now we know what the Test Set is. It is the set of examples or cases for which we want to predict the Labels. The feature vectors are as discussed earlier only a property/characteristic of the examples/cases alone. It does not matter whether the example is from train set or test set, the example will always have a feature vector.

The label is the tricky part

Up until now, we have assumed that the test data we have is unlabelled. This may or may not be true.

Usually what we do while training models is something like this

→ We have a lot of LABELLED data
→ We split it into two parts RANDOMLY

Usually
80 % is the test data
20 % is the train data

→ With this 20 % train data, we train the model
→ Then we check how our trained model performs on the test data

Here we have two things for a case in the TEST SET

For $x^{(j)}$ in TEST SET

The predicted label : $h(x^{(j)})$
The truth : $y^{(j)}$

→ We can then compare these two values to find an error, similiar to our training error

$$\text{Test Error} : \mathcal{E}(h) = \frac{1}{m} \sum_{j=1}^{m} [\![h(x^{(j)}) \neq y^{(j)}]\!]$$

$x^{(j)}$: jth feature vector in TEST SET
m : Total cases in TEST SET
$y^{(j)}$: Label of jth case in TEST SET

Now NOTE that this will only be possible to calculate when we have actual labels for the cases in the TEST DATA SET, which may not always be the case

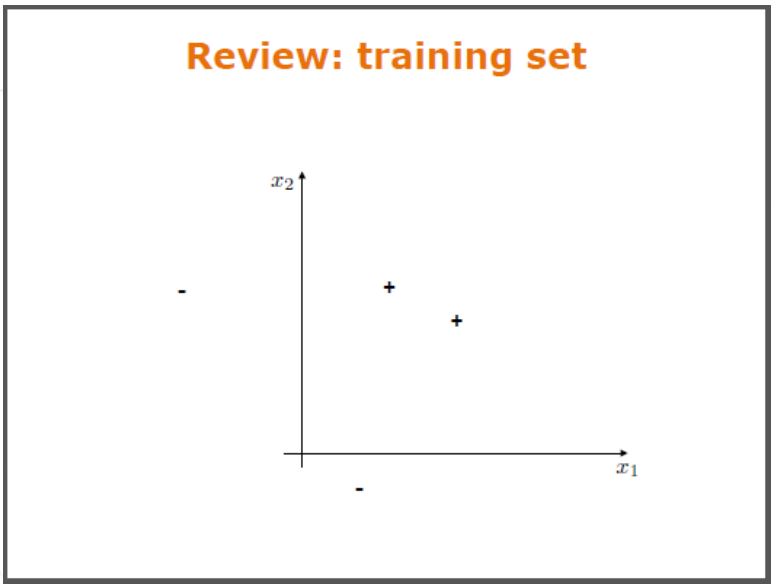
> Set of Classifiers : $\mathcal{H}$

This was the set of choices we had for our function $h()$. ie $h \in \mathcal{H}$
We saw that the more choices we allowed for $h()$, the less generalized the model became. ie although we got perfect classifiers for the train set with $\mathcal{E}_n(h) = 0$; they failed miserably when applied to test data.

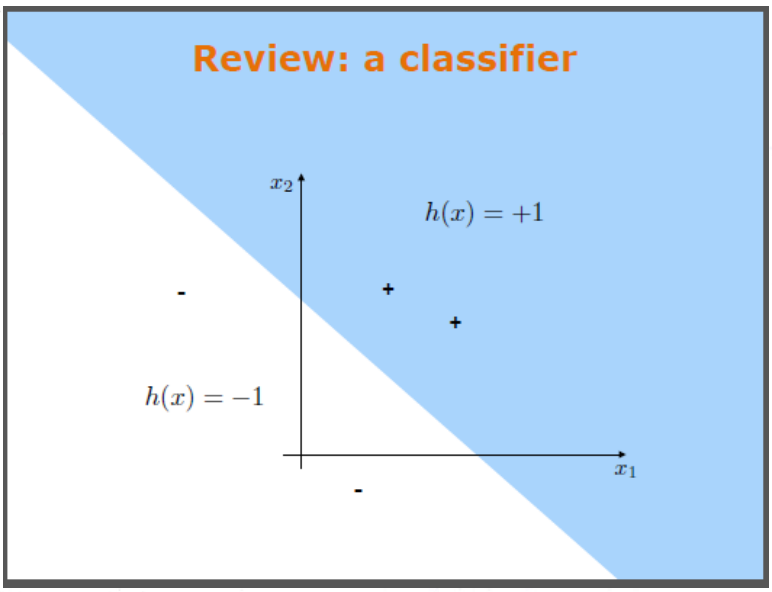
Let's also revise the geometric picture we had in mind while doing all this

> Geometric Picture

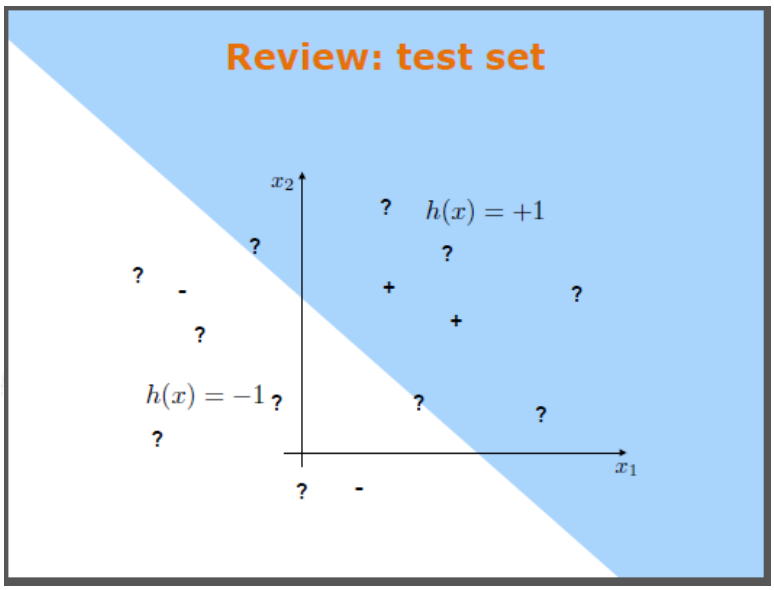
In order to analyse what happened geometrically, we took
$n = 4$ examples in training set
$X \in \mathbb{R}^2$ ie the feature vector was 2 dimensional
$y^{(i)} \in \{+1 \text{ or } -1\}$



This is a scatter plot of the training set



Classifier divided the space into two separate regions



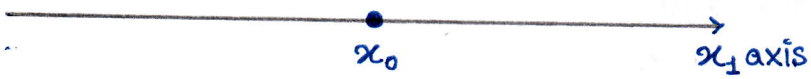
But the real problem was classifying new feature vectors

▷ Linear Classifiers

Now let me remind you something. Although we have talked a lot about how we plan to classify or predict the labels, we haven't yet proposed any mathematical form for the function $h()$ that is our classifier. In our lecture now, we plan to do exactly that, and we will start with the simplest family called the linear classifier. But in order to clearly understand them, we will need a mathematical concept called hyperplane so:

MATHEMATICAL INTERLUDE : HYPERPLANE

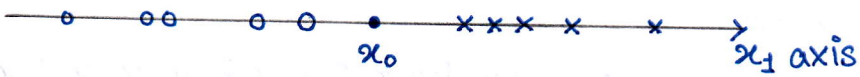
> Lets begin with a simple case



We have a straight line above called o x_1 axis. And we have some point at x_0
What is the equation of this point?
That's a rather silly question but if we have to answer it, the equation would be $x_1 = x_0$

Now note what this point is doing. It's dividing the x_1 axis or line into two regions. Ie

Region 1 : To the right of x_0
Region 2 : To the left of x_0

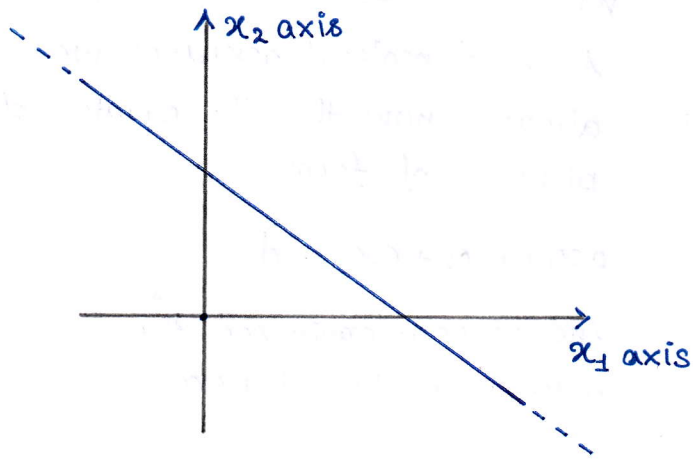


× : points to the right of x_0

o : points to the left of x_0

Good now lets move on to the next case

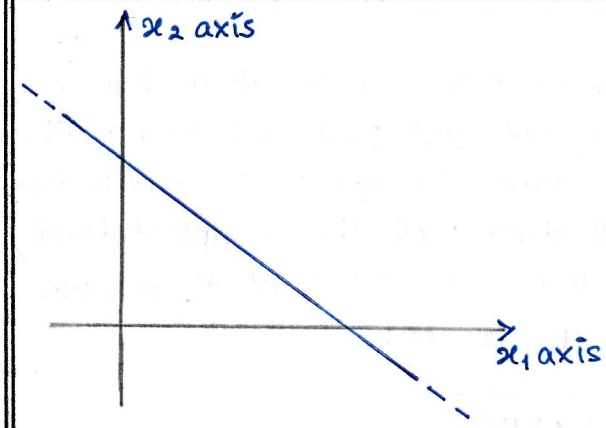
> Lets complicate this a little



Now we have two axes x_1 and x_2
ie we have a 2d plane

We also have a line that passes through this plane

We again ask the question, what is the equation of this line



We already know that the equation of a line looks like

$$x_2 = mx_1 + c$$

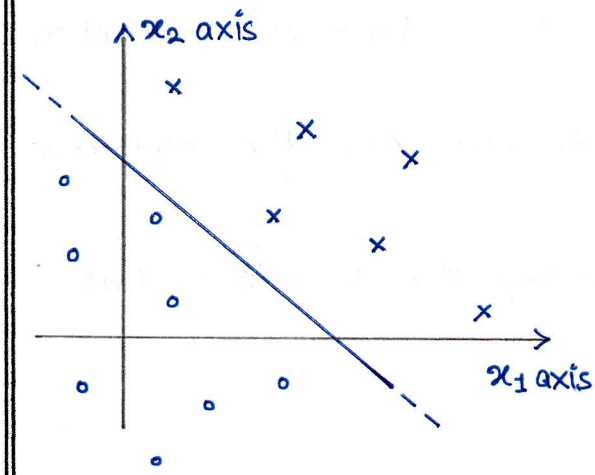
where m is slope
c is intercept

we can write this as

$$x_2 + mx_1 - c = 0$$

But we again note a pattern here. The blue line divides the 2d plane into two regions

Region 1 : Above the line
Region 2 : Below the line

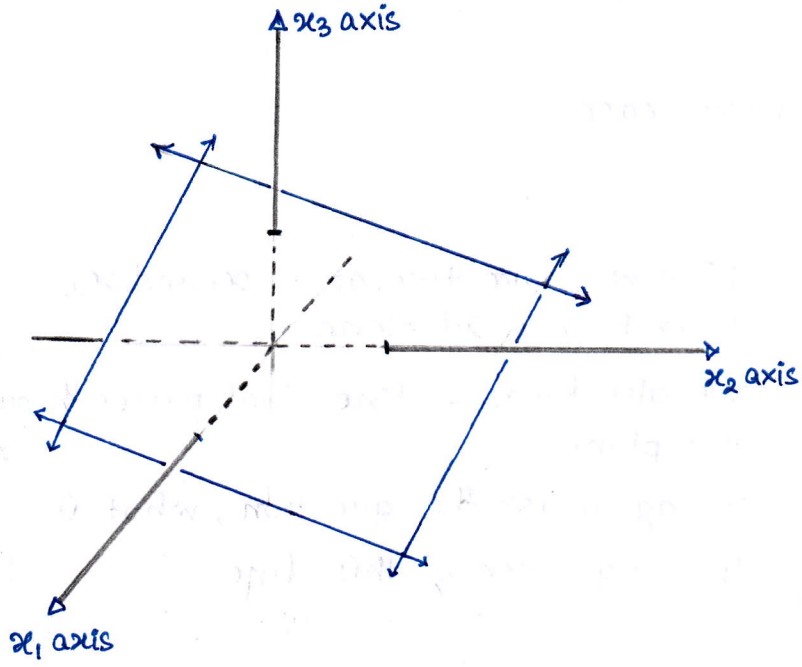


× : points above the line
o : points below the line

Note that there will also be points exactly on the line which satisfy the equation of the line

Good now lets move on to the next case

> Lets complicate this a little more



Now we have 3 axes here x_1, x_2 & x_3
This is what we call a 3d plane
we also have a plane(in blue)
And we again can ask the question what is the equation of this plane
As mathematical geniuses we already know that the equation of plane is of form

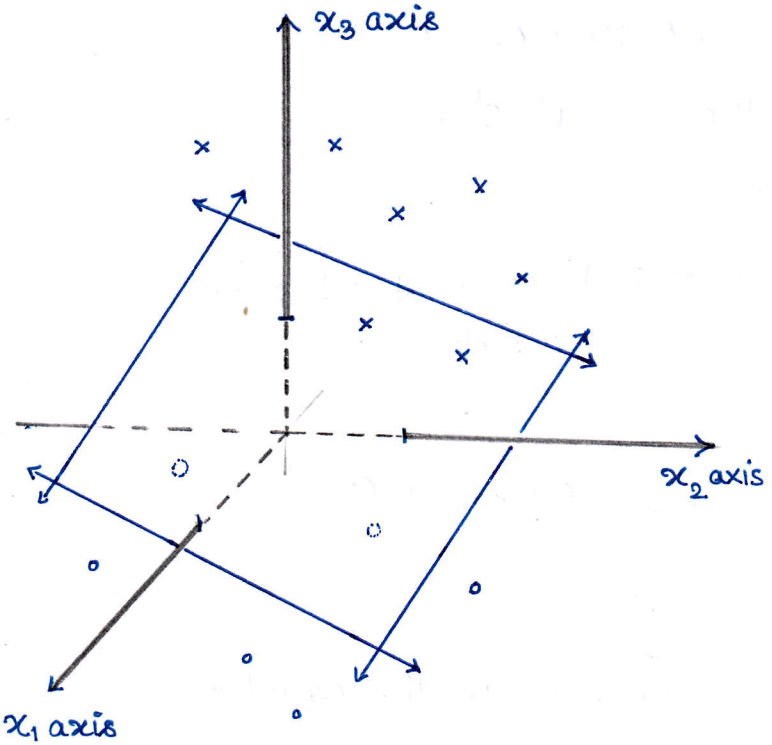
$$ax_1 + bx_2 + cx_3 = d$$

And we can again see the pattern continued here

The plane again divides the 3d region into two parts

Region 1 : Above the plane points

Region 2 : Below the plane points



× : points above the plane

o : points below the plane

Note that there will also be points that lie exactly on the plane & will satisfy plane's equation

> So now we can't go any further visually

Because there's no way we can draw a 4d plot. But we can mathematically translate this idea forward. Before doing that let's look at the similiarities we had for our 3 examples above

Equation of

point in 1dimensions: $x_1 - x_0 = 0$

Line in 2 dimensions: $x_2 - mx_1 - c = 0$

plane in 3dimensions: $ax_3 + bx_2 + cx_1 - d = 0$

So we can say that if we have d dimensional space in general, the equation we are looking for is of the form

$$\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots\cdots + \theta_d x_d + \theta_0 = 0$$

Also note the following

In 1 dimension : dimension of point is 0

In 2 dimension : dimension of line is 1

In 3 dimension : dimension of plane is 2

So if we are in a d dimensional space we will have the above equation as a 'd-1' dimensional thing/object.

Now a final thing to note is

No matter which dimension d, we choose, the equation we wrote before divides the d dimensional space into <u>Two</u> seperate regions

And there are also points that lie on this object that satisfy the equation we wrote earlier ie, $\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d + \theta_0 = 0$

We are now ready to understand the concept of :

> <u>Hyperplane</u>

A hyperplane in n dimensions is a n-1 dimensional subspace.

A hyperplane seperates the space into two sides

The equation of a hyperplane is

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots\cdots + \theta_n x_n = 0$$

θ_0 : is called the offset term

Linear algebra provides us a neat way to write the equation

If we take

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}_{n\times 1} \quad \& \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1}$$

Then we can use the dot product ($\theta^T X$ or $\theta \cdot X$) to write the equation of hyperplane as

$$\theta_0 + \theta \cdot X = 0$$

Now in light of this new formulation, lets look at our old examples

point in 1d : $x_1 - x_0 = 0$; $\theta_1 x_1 + \theta_0 = 0 \Rightarrow \theta_1 = 1, \ \theta_0 = -x_0$

line in 2d : $x_2 - m x_1 - c = 0$; $\theta_2 x_2 + \theta_1 x_1 + \theta_0 = 0 \Rightarrow \theta_1 = -m, \ \theta_2 = 1, \ \theta_0 = -c$

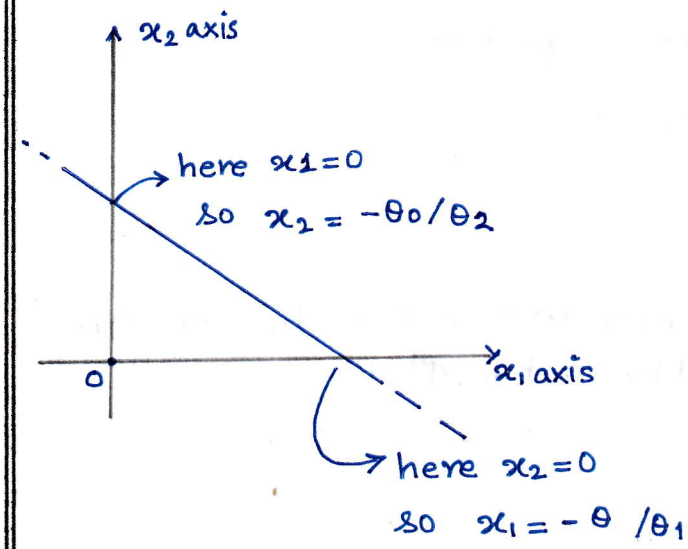
plane in 3d

$a x_3 + b x_2 + c x_1 - d = 0$; $\theta_3 x_3 + \theta_2 x_2 + \theta_1 x_1 + \theta_0 = 0$

ie $\theta_1 = c, \ \theta_2 = b, \ \theta_3 = a, \ \theta_0 = -d$

Now before we end this topic one last thing

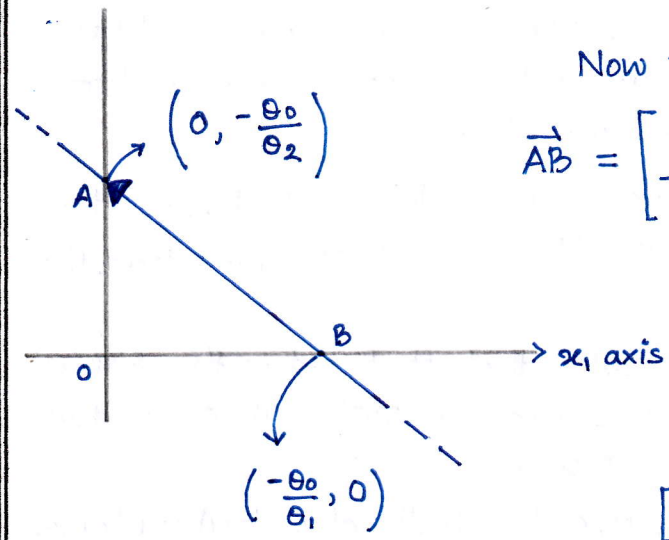
Lets first see the case of line in 2d



the equation of line is

$$\theta_1 x_1 + \theta_2 x_2 + \theta_2 = 0$$

ONE LAST THING



Now look at AB vector

$$\vec{AB} = \begin{bmatrix} 0 \\ -\frac{\theta_0}{\theta_2} \end{bmatrix} - \begin{bmatrix} -\frac{\theta_0}{\theta_1} \\ 0 \end{bmatrix} = \begin{bmatrix} \theta_0/\theta_1 \\ -\theta_0/\theta_2 \end{bmatrix}$$

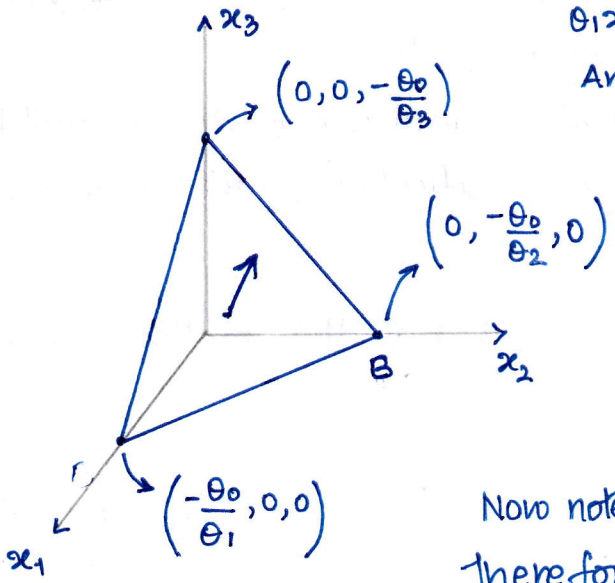
If we find the dot product of θ vetor with $\vec{AB}$ we get

$$\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} \theta_0/\theta_1 \\ -\theta_2/\theta_2 \end{bmatrix} = \theta_0 - \theta_0 = 0$$

which means any vector thats along the direction of $\vec{AB}$ is perpendicular θ vector.

Similiarly for the case of plane in 3d we have

$$\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_0 = 0$$



Any vector on the plane can be written as

$$\vec{V} = \lambda_1 \vec{BA} + \lambda_2 \vec{CA}$$

$$\vec{BA} = \begin{bmatrix} 0 \\ -\theta_0/\theta_2 \\ 0 \end{bmatrix} - \begin{bmatrix} -\theta_0/\theta_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \theta_0/\theta_1 \\ -\theta_0/\theta_2 \\ 0 \end{bmatrix}$$

$$\vec{CA} = \begin{bmatrix} 0 \\ 0 \\ -\theta_0/\theta_3 \end{bmatrix} - \begin{bmatrix} -\theta_0/\theta_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\theta_0/\theta_1 \\ 0 \\ -\theta_0/\theta_3 \end{bmatrix}$$

Now note $\theta \cdot \vec{BA} = 0 = \theta \cdot \vec{CA}$

Therefore $\theta \cdot \vec{V} = 0$ ie θ vector is perpendicular to any vector on the plane.

Therefore we can conclude

For a hyperplane in n dimensions with equation

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = 0$$

$$\text{OR} \quad \theta_0 + \theta \cdot X = 0$$

The vector θ will be orthoganal to any vector that lies on this hyperplane, ie lives in the subspace defined by the hyperplane.

Let's now move on to the lecture

▷ Linear Classifiers

Ok, now we are ready to study our first classifier mathematically. And the problem we had in mind is the movie recommender, where a movie from test data set will be assigned either $+1$ or -1 label.

We already know what a classifier does for our problem. It divides the space where $X \in \mathbb{R}^d$ live into two regions. (For our example, we had the blue region and the white region).

But the study of hyperplanes earlier, revealed that it too does the same thing. A hyperplane in n dimensions is a $n-1$ dimensional subspace that divides the n dimensional space into two regions.

So as our first classifier, why don't we use the help of a hyperplane in order to find our classifier $h()$?

And we will indeed do exactly that. We will now tackle the movie recommender problem, using hyperplane's help in forming a classifier.

Now note that a hyperplane is actually just a line, when we are in 2 dimensional space. We decided earlier to keep things simple, that the feature vector for ith movie $X^{(i)}$ will have two features ie $X^{(i)} \in \mathbb{R}^2$ So our hyperplane will be a line in our example.

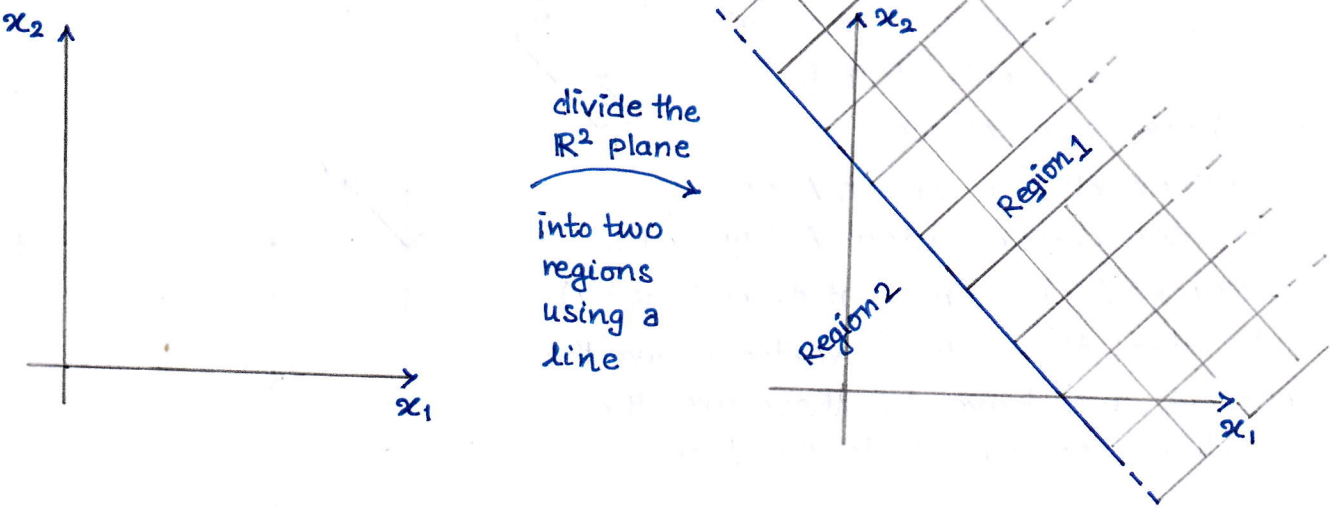
However the mathematical formulation that we will use or derive will be applicable for any problem of two-class classification, with a general feature vector $X \in \mathbb{R}^d$

We may not be able to visualize it when $d > 3$ but the mathematical ideas will remain the same

So buckle up, let's do it

> The idea

We have $x^{(i)} \in \mathbb{R}^2$. Ie the feature vectors are 2 dimensional. So we need to divide the $\mathbb{R}^2$ plane in 2 regions, using a line



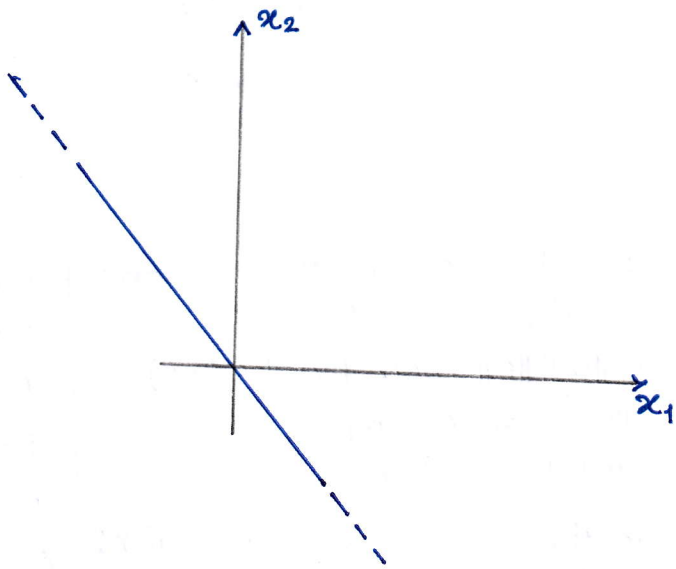
Now the general equation of a line is

$$\theta_1 x_1 + \theta_2 x_2 + \theta_0 = 0$$

This line here is called $\boxed{\text{DECISION BOUNDARY}}$. It's the term machine learning people use. Are you a machine learning people? If yes then use this term. In general d dimensions. The hyperplane will be called the decision boundary

Ok now in order to carry our analysis forward, we will first simplify the math a bit. We will let the decision boundary, pass through origin

We will later see the general case but first lets see this one



If that is the case then point (0,0) should satisfy the equation

ie $\theta_1 \cdot 0 + \theta_2 \cdot 0 + \theta_0 = 0$

$\Rightarrow \quad \theta_0 = 0$

So the offset term θ_0 is 0

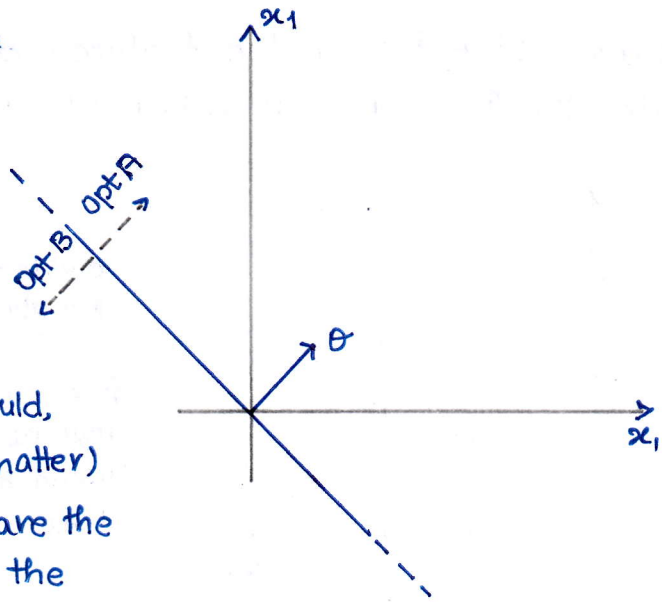
The equation of line becomes

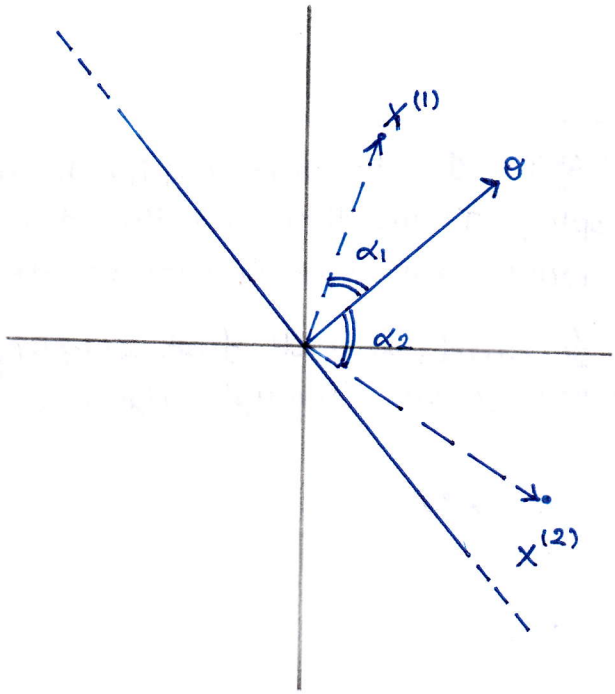
$$\theta_1 x_1 + \theta_2 x_2 = 0$$

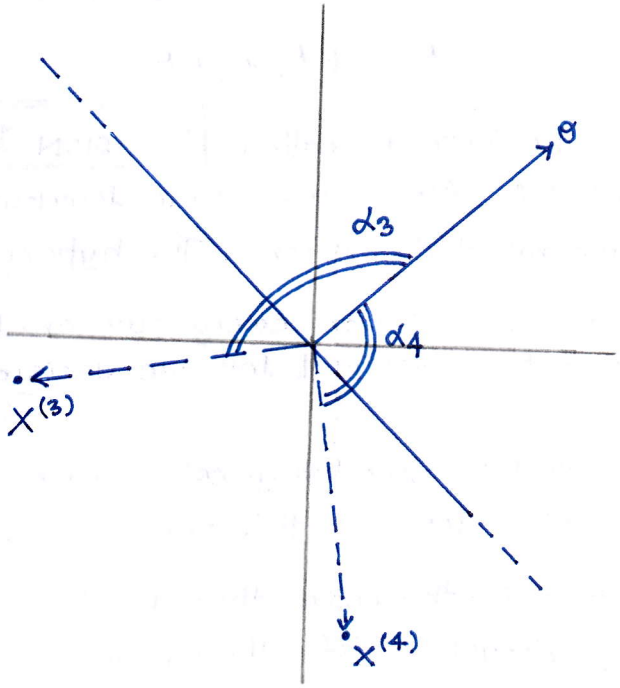
Decision boundary passing through origin

Now let's move a step further

Now we want to look at the vector θ
As we know that vector θ is orthogonal to the hyperplane (line here)
The real direction of θ will become clear when we actually know the numerical values of its components. But for now we have two choices
we can either choose option A or option B. We choose option A (we could, have very well choosen B, it dosen't matter)
But note that these two directions are the only choices we have as these are the only directions orthogonal to the line







Here we see two vectors that lie on Region . By def'n of dot product

$\theta \cdot X = \|\theta\| \|X\| \cos(\angle \text{ bw the two})$

$\theta \cdot X = \|\theta\| \|X^{(1)}\| \cos \alpha_1$

$\theta \cdot X = \|\theta\| \|X^{(2)}\| \cos \alpha_2$

As long as the o x lies in Region
The angle between will be between 0 to $\pi/2$ therefore

$\theta \cdot X > 0$ for all X in Region 1

Here we see two vectors that lie on Region By def'n of dot product

$\theta \cdot X = \|\theta\| \|X\| \cos(\angle \text{ bw the two})$

$\theta \cdot X^{(3)} = \|\theta\| \|X^{(3)}\| \cos \alpha_3$

$\theta \cdot X^{(4)} = \|\theta\| \|X^{(4)}\| \cos \alpha_4$

As long as the vector X lies in Region 2
The angle between, will be between $\pi/2$ to π

$\theta \cdot X < 0$ for all X in Region 2

So now we have a new insight

we had the decision boundary $\theta \cdot X = 0$ passing through origin

And we know that it divides the space into two regions

The vectors that lie on the side towards which θ points will have a positive dot product ie $\theta \cdot X > 0$ for X along θ's side

The vectors that lie on the side opposite to which θ points will have a negative dot product ie $\theta \cdot X < 0$ for X opposite to θ's direction.

with this insight we can develope a classifier as follows

→ we need a classifier h such that

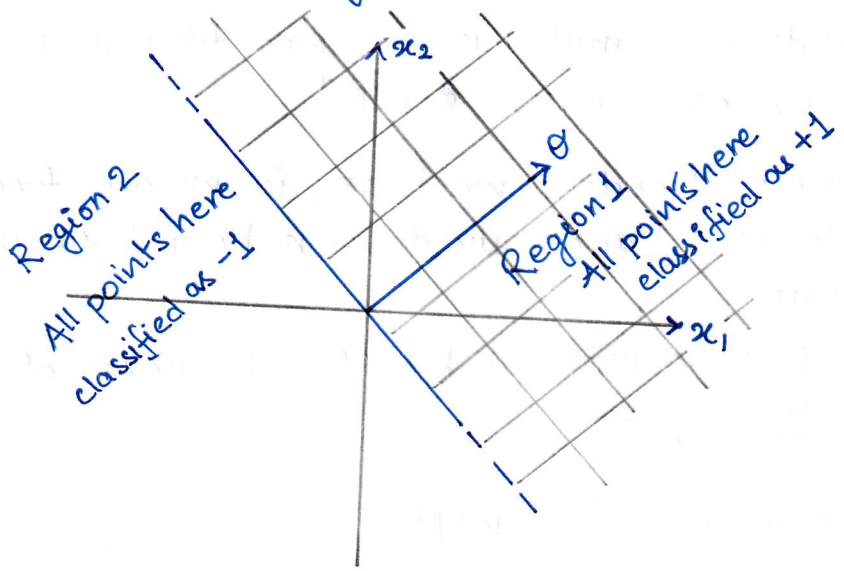
$$h : \mathbb{R}^2 \longmapsto \{+1, -1\}$$

→ Ie it should give output $+1$ or -1

→ why don't we just take the sign of the innerproduct $\theta \cdot X$ for a vector X lying in $\mathbb{R}^2$

$$h(X; \theta) = \text{Sign}(\theta \cdot X)$$

→ This is equivalent to saying that h classifies all vectors in Region 1 as $+1$ and Region 2 as -1



So to conclude

Decision Boundary $\theta \cdot X = 0$

Classifier : $h(X; \theta) = \text{Sign}(\theta \cdot X)$

Now here we will see a degree of freedom that we still have.

This will later help us

> A degree of freedom unused

Now look at the expression of the decision boundary

$$\theta \cdot X = [\theta_1 \quad \theta_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \theta_1 x_1 + \theta_2 x_2 = 0$$

What happens if i change θ to say 25θ, lets see

$$\theta \cdot X = [25\theta_1 \quad 25\theta_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 25(\theta_1 x_1 + \theta_2 x_2) = 0$$

$$\Rightarrow \theta_1 x_1 + \theta_2 x_2 = 0$$

which means the equation of decision boundry is still the same
And this is not true only for 25θ but any $c\theta$ where $c \in \mathbb{R}$
The point here is as long as θ is replaced by some multiple of θ, the decision boundary remains the same.

This is also true for the classifier

$$h(X;\theta) = \text{sign}(\theta \cdot X)$$

If c is some real positive number then

$$h(X; c\theta) = \text{sign}(c\theta \cdot X) = \text{sign}(\theta \cdot X) = h(X;\theta)$$

So all mathematical formulation, we saw till now won't get affected if we use θ or $c\theta$ where $c \in \mathbb{R}^+$.

This is a degree of freedom we have, ie we are free to choose any positive multiple of θ, and our decision boundary and classifier will remain the same

But note that if θ is replaced by $c\theta$. The norm of vector change

ie $\|\theta\| = \sqrt{\theta_1^2 + \theta_2^2}$

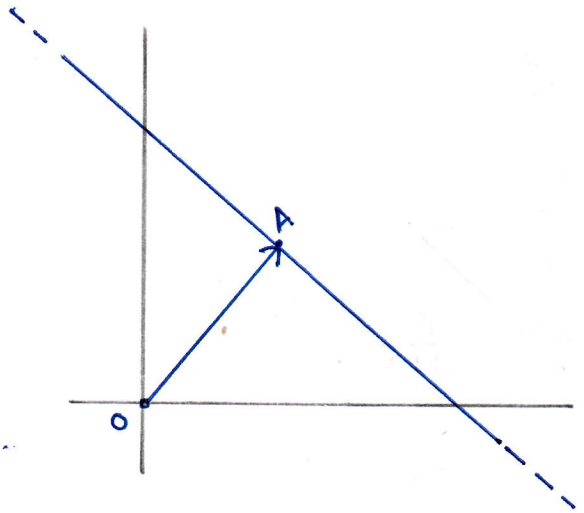
$$\|c\theta\| = \sqrt{c^2\theta_1^2 + c^2\theta_2^2} = |c|\,\|\theta\|$$

So the additional degree of freedom we have relates to the norm of the θ vector, which we haven't fixed up untill now.
We will later see in Lecture 3 that this norm will help us to form what are called large margin classifiers. But that's later.

First lets generalise for the case of any line and not just line passing through origin.

> The general case $\theta_0 \neq 0$

If you don't wanna be bothered with some tedious calculation you can skip this part and, assume yourself that the classifier for this case will be similiarly defined but if you are curious heres what's happening



> The equation of line is

$\theta_1 x_1 + \theta_2 x_2 + \theta_0 = 0$

> we for some reason want to find the distance of origin to the line

> we draw a vector starting from origin and ending at the line, orthogonal to the line

> If its ortoogonal to the line this can be written as $\lambda\vec{\theta}$

> So Point A or $\vec{\ } = \lambda\vec{\theta}$ where $\lambda \in \mathbb{R}$

> This means $\vec{OA} = \lambda\vec{\theta} = [\lambda\theta_1 \quad \lambda\theta_2]^T$

> If point A lies on the line then it must satisfy its equation

therefore $\theta_1(\lambda\theta_1) + \theta_2(\lambda\theta_2) + \theta_0 = 0$

> From here we find $\lambda = -\theta_0 / \theta_1^2 + \theta_2^2 = -\dfrac{\theta_0}{\|\vec{\theta}\|^2}$ ——— (A)

> So the perpendicular distance of origin (0,0) from line is

$$\|\vec{OA}\| = \|\lambda\vec{\theta}\| = \|\vec{\theta}\| = \frac{|\theta_0|}{\|\vec{\theta}\|} \quad \text{——— (B)}$$

> Now note due to our construction θ_0 is negative because $\lambda\vec{\theta}$ can only be along $\vec{\theta}$'s direction if λ is positive, otherwise it will be to the opposite direction. And we found

$$\lambda = \frac{-\theta_0}{\|\vec{\theta}\|^2} \implies \theta_0 \text{ is negative}$$

> Now every thing is ready. Lets look at

$$\vec{\theta} \cdot \vec{X} + \theta_0$$

We know $\vec{\theta} \cdot \vec{X} = \|\vec{\theta}\| \|\vec{X}\| \cos\alpha$, so above equation becomes

$\|\vec{\theta}\| \|\vec{X}\| \cos\alpha + \theta_0$; Lets take $\|\vec{\theta}\|$ outside

$$\|\vec{\theta}\| \left(\|\vec{X}\| \cos\alpha + \frac{\theta_0}{\|\vec{\theta}\|} \right)$$

We also know θ_0 is negative due to our construction so we can write it as $-|\theta_0|$

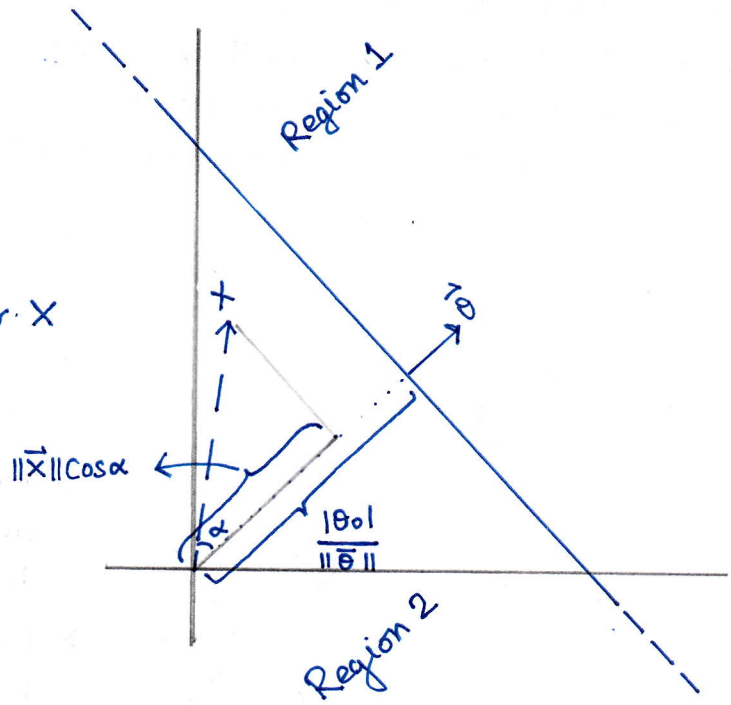
> So $\vec{\theta}\cdot\vec{X} + \theta_0$ finally becomes

$$\|\vec{\theta}\| \left(\|\vec{X}\| \cos\alpha - \frac{|\theta_0|}{\|\vec{\theta}\|} \right)$$

So this is

$\|\vec{X}\| \cos\alpha$: Projection of any vector X along $\vec{\theta}$'s direction

$\frac{|\theta_0|}{\|\theta\|}$: Perpendicular distance of origin to the line derived before eq. (B)



For any vector X in Region 1

$\|\vec{X}\| \cos\alpha > |\theta_0| / \|\vec{\theta}\|$ therefore $\text{sign}(\vec{\theta}\cdot\vec{X} + \theta_0) > 0$

For any vector X in Region 2

$\|\vec{X}\| \cos\alpha < |\theta_0| / \|\vec{\theta}\|$ therefore $\text{sign}(\vec{\theta}\cdot\vec{X} + \theta_0) < 0$

Therefore we can generalise our classifier

Decision Boundary :	$\theta\cdot X + \theta_0 = 0$
Classifier	$h(X; \theta, \theta_0) = \text{sign}(\theta\cdot X + \theta_0)$

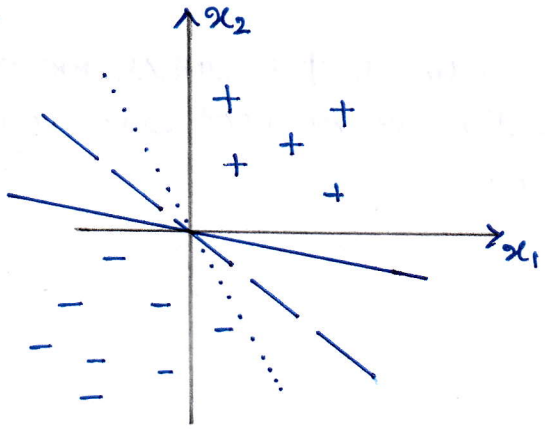
Now we have our first classifier. Congratulations !!!

▷ Constraints of our classifier

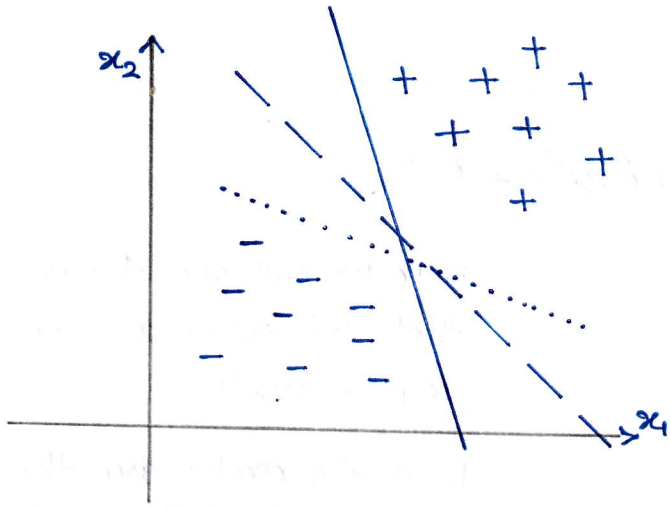
Although we have a classifier, this does not mean we can classify properly. What I mean to say is, the linear classifier that we made just now is nothing more than a line in 2d. Why would it, solve every problem of two-category classification that we throw at it?

The point here is there is a small category of problems that our linear classifier above could tackle properly.

If we have our training data, there's no guarantee that all points would be classified properly. Lets see this visually.

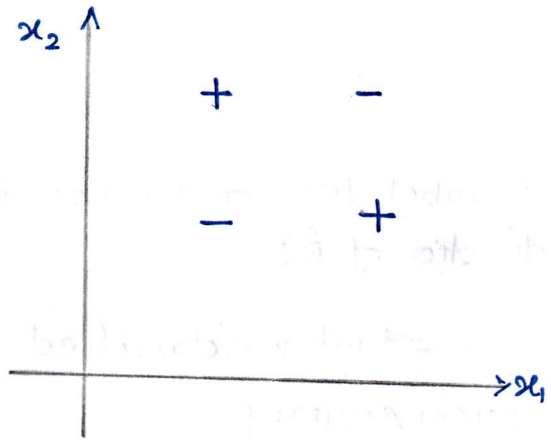


Here we have some training data
we see there can be many lines
through origin that can classify
each point properly
Some are drawn

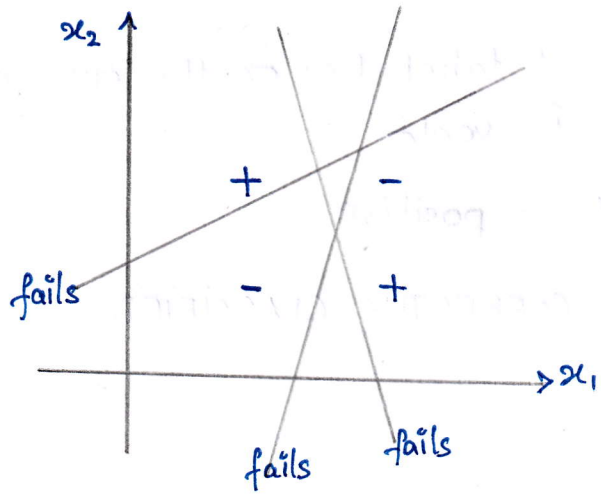


Here we see some training data
we see there can be no line through
origin that correctly classifies
all the points.
But there still can be many lines
ie linear classifier that can do
our job
Some are drawn

Now look at the example below. We run into a problem here



NO MATTER how hard we try
we will never be able to find a
line ie a classifier, that
classifies ALL the training
points correctly. And we are just
using 4 examples in 2 dimensions
here.



Every line we try fails. It's not
possible

And this brings us to the topic
of linear separability of a set
that we will study next

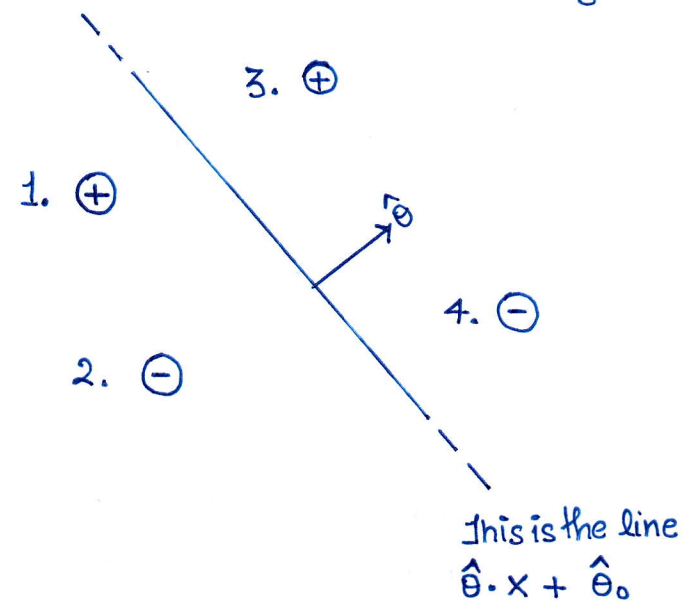
▷ Linear Seperation

So we saw earlier, that in order to find a classifier that is LINEAR, and classifies ALL the points in training set correctly, we will need some condition to be satisfied and that condition is following:

Definition:
Training examples $S_n = \{(x^{(i)}, y^{(i)}\}), i = 1, \ldots, n\}$ are *linearly separable* if there exists a parameter vector $\hat{\theta}$ and offset parameter $\hat{\theta}_0$ such that $y^{(i)}(\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) > 0$ for all $i = 1, \ldots, n$.

Lets understand this properly

First understand this term : $y^{(i)}(\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0)$



Here we can see 4 cases that will cover all our require ments

Inside the circles are the actual labels of the vectors

Let's see them one by one

1. ⊕ : when feature vector with +1 label lies on the opposite side of decision boundary wrt $\hat{\theta}$ vector

Then : $y^{(i)} = +1$; $\hat{\theta} \cdot x^{(i)} + \theta_0 =$ negative

Therefore: $y^{(i)}(\hat{\theta} \cdot x^{(i)} + \theta_0) < 0$: MISCLASSIFIED

2. ⊖ : when feature vector with −1 label lies on the opposite side of decision boundary wrt $\hat{\theta}$ vector

Then : $y^{(i)} = -1$; $\hat{\theta} \cdot x^{(i)} + \theta_0 =$ negative

Therefore : $y^{(i)}(\hat{\theta} \cdot x^{(i)} + \theta_0) > 0$: CORRECTLY CLASSIFIED

3. ⊕ : When feature vector with +1 label, lays on the same side of the decision boundary wrt $\hat{\theta}$ vector

Then: $y^{(i)} = +1$; $\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0 =$ positive

Therefore: $y^{(i)} (\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) > 0$: CORRECTLY CLASSIFIED

4. ⊖ : when the feature vector with −1 label, lays on the same side of decision boundary wrt $\hat{\theta}$ vector

Then: $y^{(i)} = -1$; $\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0 =$ positive

Therefore: $y^{(i)} (\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) < 0$: MISCLASSIFIED

So we finally get to see the following

When a vector is MISCLASSIFIED : $y^{(i)} (\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) < 0$

When a vector is CORRECTLY CLASSIFIED : $y^{(i)} (\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) > 0$

So when in the definition of linear separability we said.

$$y^{(i)} (\hat{\theta} \cdot x^{(i)} + \hat{\theta}_0) > 0 \quad \text{for all } i = 1, 2, 3 \ldots n$$

we are just saying that the line (or hyperplane) classifies all the points in the training set correctly so the definition can be boiled down to:

Defn

A set $S_n = \{ (x^{(i)}, y^{(i)}), i = 1, 2, 3 \ldots n \}$ are linearly seperable if we can find $\hat{\theta}$ and θ_0 ie parameter vector and offset parameter that seperate them. ie if we can find a line that seperates all points correctly

That is redundant

And if we look at it closely, it absolutely says nothing at all. It just says you can correctly classify all points using a linear classifier if you can find a linear classifier that can do that

But this definition is needed for the next algorithm that we are about to see called the perceptron. Which relies on the fact that a solution will be found for $\hat{\theta}$ vector and θ_0 offset parameter only if S_n is linearly seperable. But first lets calculate training error in this case.

> Training Error for Linear Classifier

We earlier defined training error as the fraction of points in the training set that were misclassified. Here for the case of linear classifier we can say that

$$h(x^{(i)}) = \theta \cdot x^{(i)} + \theta_0$$

We also figured out that irrespective of the label $\{+1 \text{ or } -1\}$ if

$y^{(i)}(\theta \cdot x^{(i)} + \theta_0) > 0$ then point is correctly classified

$y^{(i)}(\theta \cdot x^{(i)} + \theta_0) < 0$ the point is misclassified

There is a case when feature vector will lie on the hyperplane (line here) in that case

$$\theta \cdot x^{(i)} + \theta_0 = 0 \implies y^{(i)}(\theta \cdot x^{(i)} + \theta_0) = 0$$

We will also consider this case as an error, because we really don't know how to classify it

If you remember the $[\![\ ]\!]$ double bracket notation, we had

$$[\![\text{Expression}]\!] = \begin{cases} 1 & \text{if Expression true} \\ 0 & \text{if Expression false} \end{cases}$$

For all errors

$$y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0 \quad \text{will be true}$$

Therefore we can write

$$\mathcal{E}_n(h) \equiv \mathcal{E}_n(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} [\![y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0]\!]$$

Now we will finally see the Perceptron Algorithm, that finds a linear classifier, if there exists one.

Ie it will find a classifier if the training set S_n that we have is linearly seperable. This will not be a unique classifier as there may be many other LINEAR classifier that can accomplish this job.

Let's see it

▷ PERCEPTRON

Instead of laying the algorithm on you, we would rather see how it comes us geometrically. Now note that we are here in search of a line, but it is totally decided, once we have the vector θ.

Also note that, we will see the case of line passing through origin, just for simplicity.

Also in order for this algorithm to be successful, we need linearly seperable data ie we will need points that can be divided/classified by a line

The plane/line/decision boundary will be perpendicular to the θ vector always.

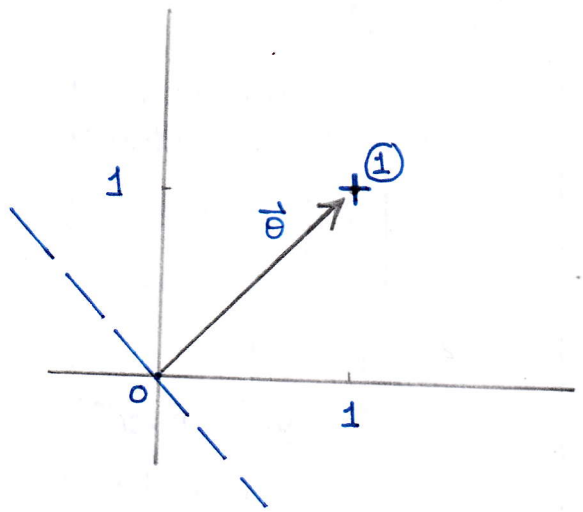
Let us begin.

> Lets first say that $\vec{\theta}$ is $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

1st Point

suppose our algorithm encounters the 1st vector

$x^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ label $= +1$



we tell our algorithm to draw a vector from origin to this point and make it the new

> $\vec{\theta} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

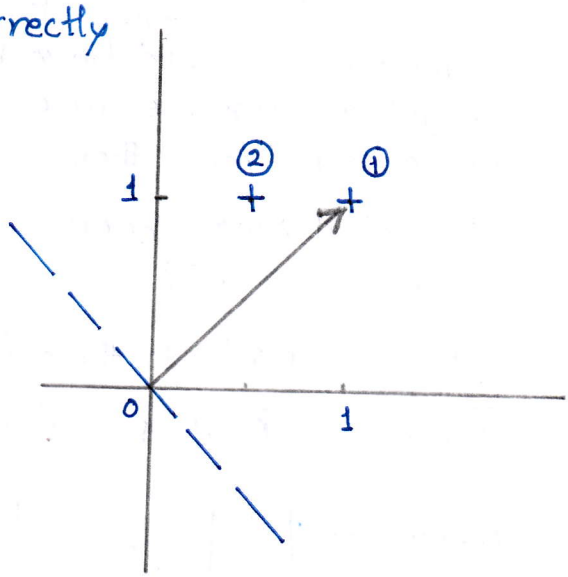
The idea is simple, its a point with +1 Label

If θ that I make is pointed towards the one point that we have then, by default this point is correctly classified

2nd Point

suppose the next point is $x^{(2)} = \begin{bmatrix} .5 \\ 1 \end{bmatrix}$ with label $+1$

Now i ask my algorithm. Is the new point correctly classified according to the θ we have



Here we can see visually that both points are to the right side of the line (in blue)

Mathematically we ask

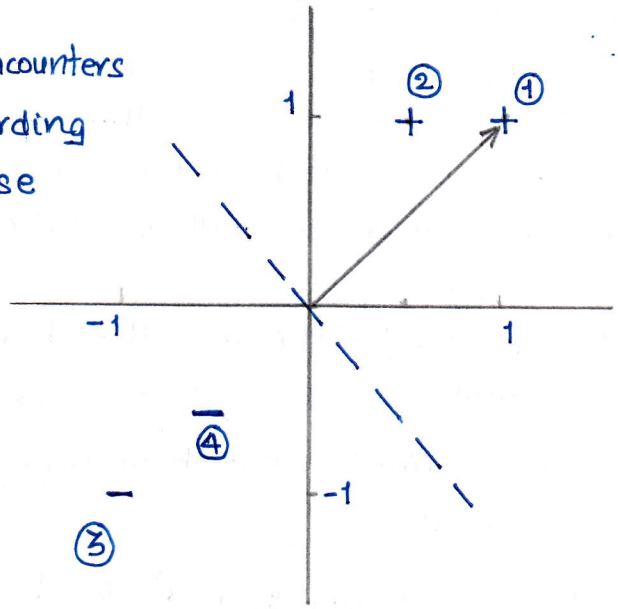
> check if $y^{(2)}(\vec{\theta} \cdot x^{(2)}) > 0$

↳ YES because $(+1)\left[[1,1]\begin{bmatrix} .5 \\ 1 \end{bmatrix}\right] > 0$

So move on to the next point

$\boxed{\text{3rd point}}$ & $\boxed{\text{4th point}}$

The point is as long as the algorithms encounters new points that are correctly classified according to our θ, we don't need to worry, because we have a correct classifier



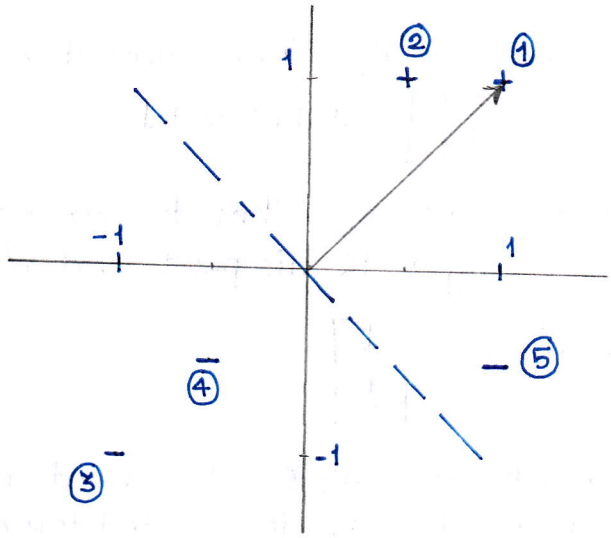
Here we have two new points

$x^{(3)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ label -1 $\vec{\theta}$ remain same

$x^{(4)} = \begin{bmatrix} -.5 \\ -.5 \end{bmatrix}$ label -1 $\vec{\theta}$ remains same

$\boxed{\text{5th point}}$

Now comes the tricky part, Let's say the 5th point encountered by algorithm is



$x^{(5)} = \begin{bmatrix} 1 \\ -.5 \end{bmatrix}$ label -1

when we check if

> check if $y^{(5)}(\vec{\theta} \cdot x^{(5)}) > 0$

$\quad$ L $\quad -1\left([1, 1]\begin{bmatrix} 1 \\ -.5 \end{bmatrix}\right)$ is not > 0

So false

we can also see that the new point $x^{(5)}$ is not correctly classified according to the current $\vec{\theta}$. So we will have to update it.

Now just forget for one second everything uptill now. and answer this.

What would $\vec{\theta}$ be if only point $x^{(5)}$ existed.

The answer is it would have been a vector from origin to $x^{(5)}$ but with a negative sign because we want feature vectors with label -1 to point away from them so

$\vec{\theta}$ would have been $y^{(i)} \cdot x^{(i)}$ or $y^{(5)} \cdot x^{(5)}$ if there was only one point $x^{(5)}$

Now we come back to the problem. The algorithm says

$$\vec{\theta}_{new} = \vec{\theta}_{old} + y^{(5)} x^{(5)}$$

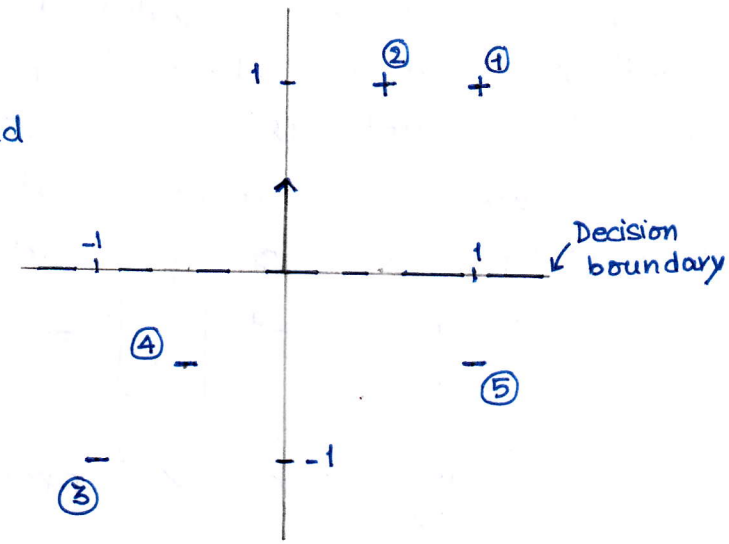
$$\vec{\theta}_{new} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 1 \cdot \begin{bmatrix} 1 \\ -.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}$$

Ie the algorithm says to find the new vector

New θ vector = Old theta vector that classified previous points correctly + New θ vector that classifies the new point correctly, if the new vector was alone.

Lets see how our graph looks like with $\vec{\theta} = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}$

we see that the x_1 axis itself has now become the decision boundary and all the point as correctly classified



Now just as the final nail in the coffin we will add another point

$\boxed{\text{6th point}}$

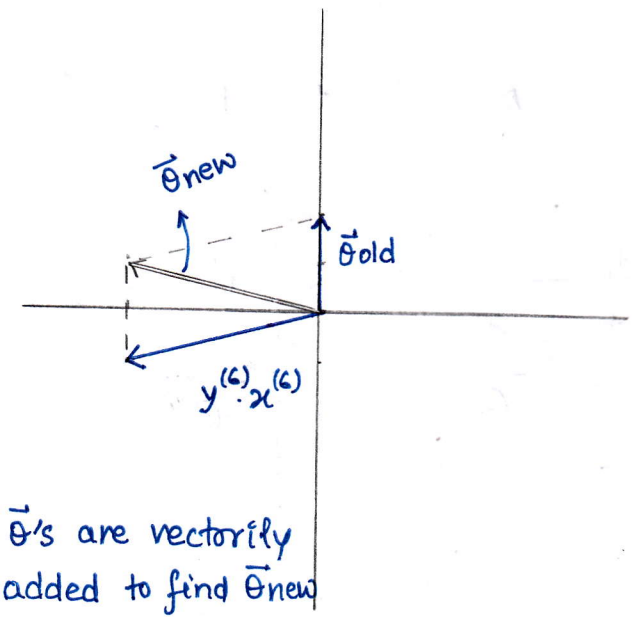
$x^{(6)} = \begin{bmatrix} -2 \\ -.25 \end{bmatrix}$ label = +1

the point will not be properly classified because $y^{(6)}(\vec{\theta} \cdot x^{(6)})$

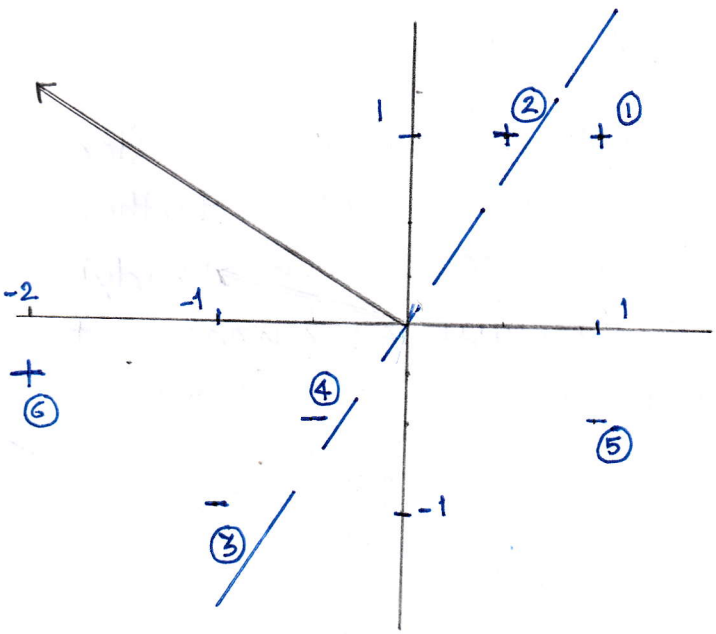
$= +1\left([0, .5]\begin{bmatrix} -2 \\ -.25 \end{bmatrix}\right) = -.125 < 0$

the new $\vec{\theta}_{new} = \vec{\theta}_{old} + y^{(6)} x^{(6)}$

$= \begin{bmatrix} 0 \\ 1.5 \end{bmatrix} + 1 \cdot \begin{bmatrix} -2 \\ -.25 \end{bmatrix} = \begin{bmatrix} -2 \\ .25 \end{bmatrix}$



$\vec{\theta}$'s are vectorily added to find $\vec{\theta}_{new}$



OOPSIEeeeee.

Just when you thought, we were about to end. The new vector $\vec{\theta}$ does not classify all the points properly. ① ③ & ④ are on the wrong side

How do you feel? Do you feel like an idiot now?

Well you should, because you are one. You suck!!! Boooooo!!!

Now Now don't be sad. I will tell you a trick now

When we are provided the data set. We won't be provided the points one after the other. We will be provided all the n (here 6) points together. So we can choose any point to be the 1st, 2nd, 3rd and nth. Its upto us. SO WE START AGAIN

So now me, who is a mathematical genius tells you to do the following

Take 1st point to be $x^{(5)}$ previously.

we get $\vec{\theta} = -1 \cdot \begin{bmatrix} 1 \\ -.5 \end{bmatrix} = \begin{bmatrix} -1 \\ .5 \end{bmatrix}$

This won't classify still believe me

Take 2nd point to be $x^{(4)}$ previously

$$y \cdot (\vec{\theta} \cdot x) = -1 \cdot \left(\begin{bmatrix} -1 & .5 \end{bmatrix} \begin{bmatrix} -.5 \\ -.5 \end{bmatrix} \right) = -(.5 - .25) = -.25 < 0$$

So

$$\vec{\theta}_{new} = \vec{\theta}_{old} + y^{(2)} x^{(2)}$$

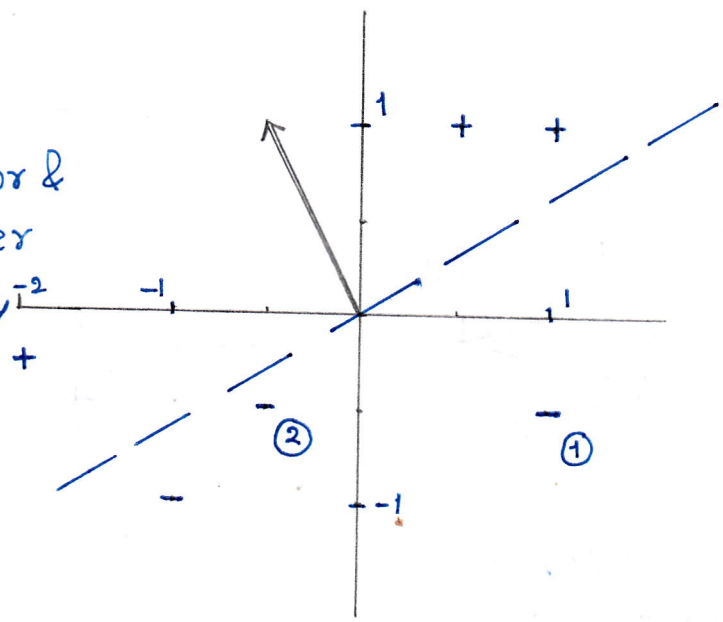
$$= \begin{bmatrix} -1 \\ .5 \end{bmatrix} + (-1) \begin{bmatrix} -.5 \\ -.5 \end{bmatrix} = \begin{bmatrix} -.5 \\ 1 \end{bmatrix}$$

(2) is the new second point when we started again

Lets look what happens now

VIOLA!!!

We now have a linear seperator & we don't need to look at further points because they are already correctly classified & wont change $\vec{\theta}$



So the Algorithm is something like this

Initialize $\vec{\theta} = \vec{0}$

Choose a sequence from 1 to n to iterate over

for loop in i

Check if $y^{(i)}(\vec{\theta} \cdot x^{(i)}) \leq 0$

↳ YES $\vec{\theta} = \vec{\theta} + y^{(i)} x^{(i)}$

↳ NO Dont do anything

Go to next i

After end of for loop

if $\vec{\theta}$ classifies all points correctly

we found the classifier . END CODE

if not ⟶ shuffle the indices

If you do this for a long time with enough faith in your lord and saviour Jesus you will find a solution

The actual algorithm looks like this

procedure PERCEPTRON($\{(x^{(i)}, y^{(i)}), i = 1, \ldots, n\}$, T)

$\theta = 0$ (vector)

for $t = 1, \ldots, T$ **do**

for $i = 1, \ldots, n$ **do**

if $y^{(i)}(\theta \cdot x^{(i)}) \leq 0$ **then**

$\theta = \theta + y^{(i)} x^{(i)}$

return θ

Note that perceptron find <u>A</u> solution, there can be many more solutions for the same training set

Now we will see how we can generalize this when there is an offset parameter

> Perceptron : General Case

we already know to test classification we can do

$$y^{(i)}(\vec{\theta}\cdot x^{(i)} + \theta_0) > 0 \quad \text{or} \quad \leq 0$$

the real trick is in updating θ_0. For this we think of our problem as finding a new vector $\vec{\theta}'$ that will have an additional last entry of θ_0 ie

$$\vec{\theta}' = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \\ \theta_0 \end{bmatrix}_{d+1 \times 1}$$

we will add an additional 1 to the last element of each feature vector.

$$x' = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}_{d+1 \times 1}$$

And update just like before

$$\vec{\theta}'_{new} = \vec{\theta}'_{old} + y^{(i)} x'^{(i)} = \vec{\theta}'_{old} + y^{(i)} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$

this will give us θ_0 new $=$ θ_0 old $+ y^{(i)}$

The algorithm looks like

procedure PERCEPTRON($\{(x^{(i)}, y^{(i)}), i = 1, \ldots, n\}, T$)
  $\theta = 0$ (vector), $\theta_0 = 0$ (scalar)
  for $t = 1, \ldots, T$ **do**
    for $i = 1, \ldots, n$ **do**
      if $y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0$ **then**
        $\theta = \theta + y^{(i)} x^{(i)}$
        $\theta_0 = \theta_0 + y^{(i)}$
  return θ, θ_0

This brings us to the end of the lecture

Check out my code **l2_perceptron_visualization.ipynb** in the shared folder to visualize a perceptron.

FIN.