UNIT 1 : Linear Classifier and Generalizations

LECTURE 1 : Introduction to Machine Learning

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!!! Whatever 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

COOL DOG

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 : CINDERELLA

Once upon a time there was this chick, called Cinderella who lived with her evil step mother and stepsisters. They treated her like trash. She had to do all the chores. A prince threw a party in the town, in order to score a wifey. Every hottie in the town was invited. Step mom wouldn't let Cinderella go. She sad.

Cinderella cried. A fairy appeared, and told her to shut it. She said bring me a pumpkin and some mice. Apparently Cinderella had mice with her. She did not do a very good job of cleaning the house, I suppose. Fairy turned the pumpkin into a carriage an mice into horses (WTF!!!). She turned Cinderella's raggy old clothes into a beautiful gown. Be back by 12 midnight. Fairy said.

Cinderella went to the party. There was booze and everything. The prince looked at all the hotties, and picked Cinderella to dance. A thing you should know about men is, it dosen't matter what the chick does for a living. If she's hot she's in.

Anyways it was 12, Cinderella didn't account for the time. Maybe because she did too many tequila shots at the open bar earlier . She ran but left her glass shoes behind. Everything was back to normal, the horse and the dress and all.

Next day the prince decided , to search for Cinderella. He said whoever will fit this shoe will be my wifey. The dumb idiot did not have the cognitive capabilities to recognize a girl by seeing her face. So he came up with this brilliant plan. To everyone's suprise no one fit the shoe in town.

When the shoe checker reached Cinderella's home. The stepmother told her real daughters to try the shoe. It didn't fit. One of the sisters was so desperate, that she cut her toes in order to fit the shoe. This is a children's story mind you. Anyways it still didn't fit.

Finally Cinderella tried it and the shoe (which was by now probably all soaked in the stepsister's blood) and viola!! It fit her feet. Somehow she had this unique shoe size in all the town. Cinderella showed a big middle finger her stepmom and stepsisters and married the prince. **THE END**

MORAL : Use some sort of scale or measuring tape to see if your feet will fit a shoe before you go of cutting you toes. No one wants to see that, it's gross.

I hope these notes will help you

Let's get started

▷ Machine learning is everywhere

There are numerous fields where machine learning is applied

> Google interprets your queries using ML
> Netflix recommends movies using ML
> Alexa spies on you using ML
> Tesla is building self driving car using ML
> A program called Alpha Go defeated the Go Champion using ML

The great philosopher Sal Vulcano from the TV show Impractical Jokers is attributed (probably falsely) writing the following.

'AI is like Jesus. It's Everywhere'

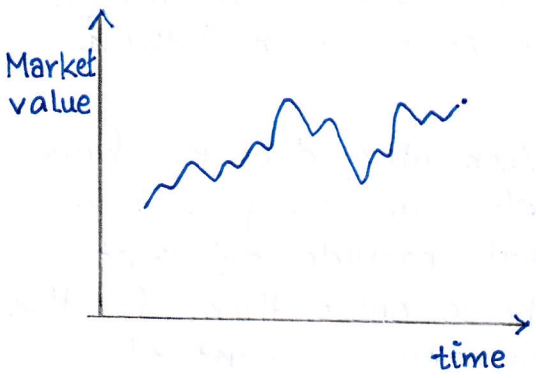
With that being said, lets see what the definition of machine learning is

Machine learning as a discipline aims to design, understand and apply computer programs that learn from experience (i.e., data) for the purpose of modeling, prediction, or control

▷ Prediction Problems

We will start our journey with the problem of prediction. There can be many type of problems that we can tackle. Let's see some

> About future Events



We may want to predict the price of a stock / index / derivative based on the price pattern up till now.

We want to know its price 1 day, 1 week 1 month or a year from now

ML can help us do that

We could also ask whether an operation will be successful, and answer if the disease will reccur in future.

A self driving car can use ML to predict the path of various cars travelling in the road to adjust its driving accordingly.

> About properties we don't know

- Based on the preference of a viewer (from the movies he/she saw) we can train an ML algorithm to decide whether the viewer will prefer watching a particular movie or not.
- We can predict whether a compound will be soluble in water or not
- We can train computers to see an image and tell us what the image is actually of.
- Translation from one language to other is not as straight forward as translating all the words of the sentence from one language to other. Different languages have different grammatical structures. ML can help us properly translate one language to other.

Lets see the basic idea first

▷ Supervised Learning

So for the begining part of the course we will study supervised learning.

Wikipedia says:

> Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input output pairs. It infers a function from labelled training data consisting of a set of training examples.

What this definition means will be clear eventually, but let's loosely look what the idea is

▷ A loose example : supervised Learning

Suppose we want to train a machine in order to classify an image. What this means is we will feed the machine an image and then ask it what is this image of?

Now we must first note that the machine/computer does not have a brain of it's own. We will have to tell it what are the possibilities that the presented image can be of. ie we will provide different categories to the machine and tell it to choose one of them for the picture/image in question. There can be a numerous amount of categories (like 1000). This problem is called image classification problem.

Now the hardcore engineering way would be to provide specific rules for each and every category. Like if we have the image of a mushroom

we can tell the machine that it is thin at the bottom thick at top
Is bright colored at top (usually)
Is dull colored at bottom

But no matter how many rigid rules we specify. A slightly different image of a mushroom will violate all the rules and will be misclassified.

: For eg. here we clearly see them as two mushrooms, but the machine will not be able to do that if it follows rigid rules

Instead what we do is something completely different. We will feed the machine a number of images of mushrooms and the machine will then learn what a mushroom looks like. we will do this for all the categories and then ask the learned machine which category does the new image (that we show it after training) belong to?

How we do this is what we will learn in detail in this course. But the basic idea is this:

- we will have a function called the classifier denoted by $h()$
- The function h will take as input an image and will depend on parameters collectively denoted by θ
- With the already correct examples that we feed into the machine these θ will be tuned.
- The tuned $h(\ ;\theta)$ will then be used to predict category of a new image

Similiarly we can see this for translation

The problem at hand is, we want to properly translate an english sentence into spanish. We will again follow the same idea

- we will form a function that will somehow map english sentences to spanish
- we will train it's parameters θ with many correct examples of proper translations.
- We will then feed our new english sentences to this $h(\ ;\theta)$ and see how good the performance is

Till now we have only talked about doing things in the abstract sense. We will now see how this philosophy translates to the mathematical domain. What precisely are we doing mathematically.

Let's Go.

▷ A concrete example : Movie Recommender

From this example we will see how the mathematical notation are written and how the problem is formulated mathematically.

> The Problem

A user is recommended movies to watch based on the movies he/she has watched earlier. We want a binary answer from our algorithm (machine), ie, we want to know for a given movie, will the particular viewer prefer to watch it (YES) or would not prefer (NO) to watch it

> Training Set

As discussed earlier the machine will learn from correct examples that we give. Here for the sake of simplicity, we have collected data for the following movies (only 4)

-1

-1

+1

+1

You can see each movie has the number +1 or -1 below it

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

We see that to each movie in the above set has +1 or -1 below it This is the label of that particular movie.

+1 : Viewer would watch the movie

-1 : Viewer would not watch the movie

The general label of the ith movie will be denoted by

$$y^{(i)}$$

Seeing from the above example we can say

$$y^{(1)} = -1 \quad ; \quad y^{(2)} = -1 \quad ; \quad y^{(3)} = +1 \quad ; \quad y^{(4)} = +1$$

But where do we get these labels from?

Maybe the viewer did a survey OR

Maybe the viewer watched the +1 labeled movies while scrolled past the -1 labeled movies and Netflix recorded the preferences.

The point is we have labels for movies that reflect viewer's preferences correctly.

> Feature vector : $x^{(i)}$

Now we plan to predict the label of a new movie therefore concluding that the viewer will be recommended the movie or not

But what would our algorithm take as input. It cannot definitely be the poster of the movies as we saw earlier.

We need to mathematically encapsulate the features of any movie that will be fed to the algorithm. Therefore to each movie we associate

feature vector : $x^{(i)}$ of the ith movie

And each element of the $x^{(i)}$ vector will contain information related to the ith movie. It depends on us what kind of features we want to choose for our problem

For example a typical feature vector can look like

$$X = \begin{bmatrix} \text{Comedy ?} \\ \text{Action ?} \\ \text{Romance ?} \\ \vdots \\ \text{Imdb rating} \end{bmatrix}$$

→ The first element will be 1 if it's a comedy 0 otherwise

→ The second element will be 1 if it's a action 0 otherwise

→ The third element will be 1 if it's a romance 0 otherwise

→ The last element will be the numerical Imdb rating of movie

The features will be chosen by us depending on availaibility of data, importance of feature and complexity of the problem

We will denote the number of features we choose by 'd'

ie we will choose d features of each movie

Say for the 2nd movie Gone Girl we collect data from Imdb to find that the genre of the movie is Drama, Mystery and Thriller and the Imdb rating is 8.1/10. The feature vector $x^{(2)}$ will reflect this, the elements for genre Drama Mystery & Thriller will be assigned 1. All other genre like comedy, action, romance, horror, sci-fi will be assigned 0. The Imdb rating will be assigned 8.1

It is to note that depending upon the formulation of our problem we can see that each element can be binary, discrete or continuous

But most generally we write

$$X \in \mathbb{R}^d \quad \text{equivalently } x^{(i)} \in \mathbb{R}^d$$

Note that the feature vector of any movie is it's own property. It has nothing to do with the viewer. So when we will predict the label of a new movie we will have with us the feature vector that we give in as input.

> Training Set: Revisited

Now that we have clarified some terminology we can formally define what we mean by the training set.

First note what training set actually is. It's a set of

n : total number of cases of movies with there respective feature vector

$x^{(i)} \in \mathbb{R}^d$: feature vector of ith movie with each movie labeled correctly

$y^{(i)}$: Label of ith movie $\in \{+1, -1\}$

And with this data we plan to teach/train the machine so mathematically

Training set will be denoted by S_n, where the subscript 'n' tells us the number of cases we have in the set.

The set will contain pairs of the form $(x^{(i)}, y^{(i)})$ ie feature vector and label for the ith movie.

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

For the example here $x^{(i)} \in \mathbb{R}^d$

$y^{(i)} \in \{+1, -1\}$

Let's now move on to first visually see whats happening before we introduce some new terminology

> Visual / Geometric point of view

We now visually want to see what's going on. But human beings in there finite imagination capabilities run into a problem everytime.

We can't visualise points/data in more than 3 dimensions. Thats the limit we have. So in order to see what's happening we will reduce the dimension of our feature vector to $d = 2$ ie our

$x^{(i)}$ will belong to $\mathbb{R}^2$

$x_1^{(i)}$: will be the first feature of the ith movie, which will be a real value

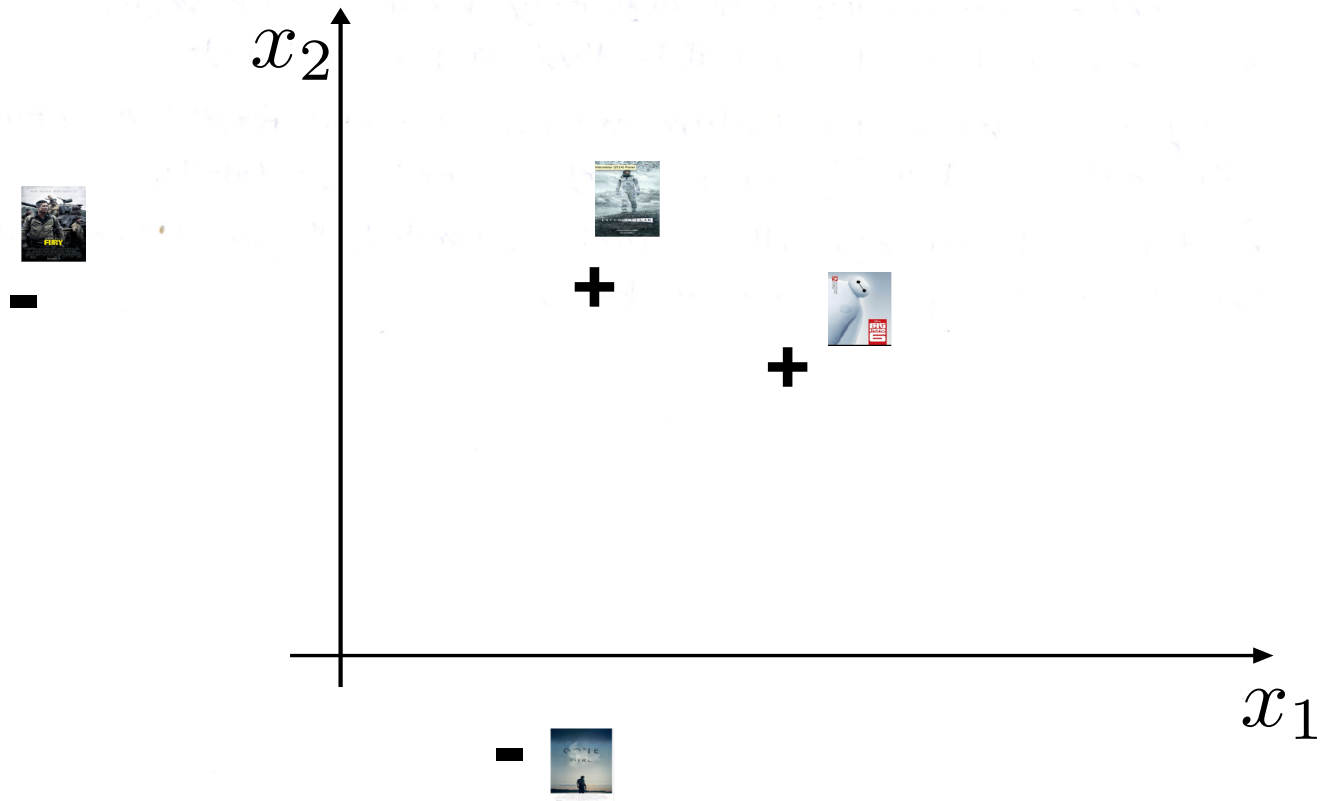
$x_2^{(i)}$: will be the second feature of the ith movie, which will be a real value

These features can be discrete valued But we are taking the most general case here

Each movie will still have the label $y^{(i)}$ which will be denoted by

+ : if label is +1
− : if label is −1

With these arrangements we can now draw some sort of scatter plot to visualise the training set



Now each of the four points above represent the pair

$$(x^{(i)}, y^{(i)})$$

And for our example 4 points are drawn because $n = 4$ ie the number of cases in the training set are 4

One must note that in order to plot a point we do not need the $y^{(i)}$ ie Label information for the ith movie

$x_1^{(i)}$: is coordinate along x_1 axis
$x_2^{(i)}$: is coordinate along x_2 axis

$y^{(i)}$ only comes in play when we decide whether we put '+' or '−' sign at coordinates $(x_1^{(i)}, x_2^{(i)})$

So next we move onto

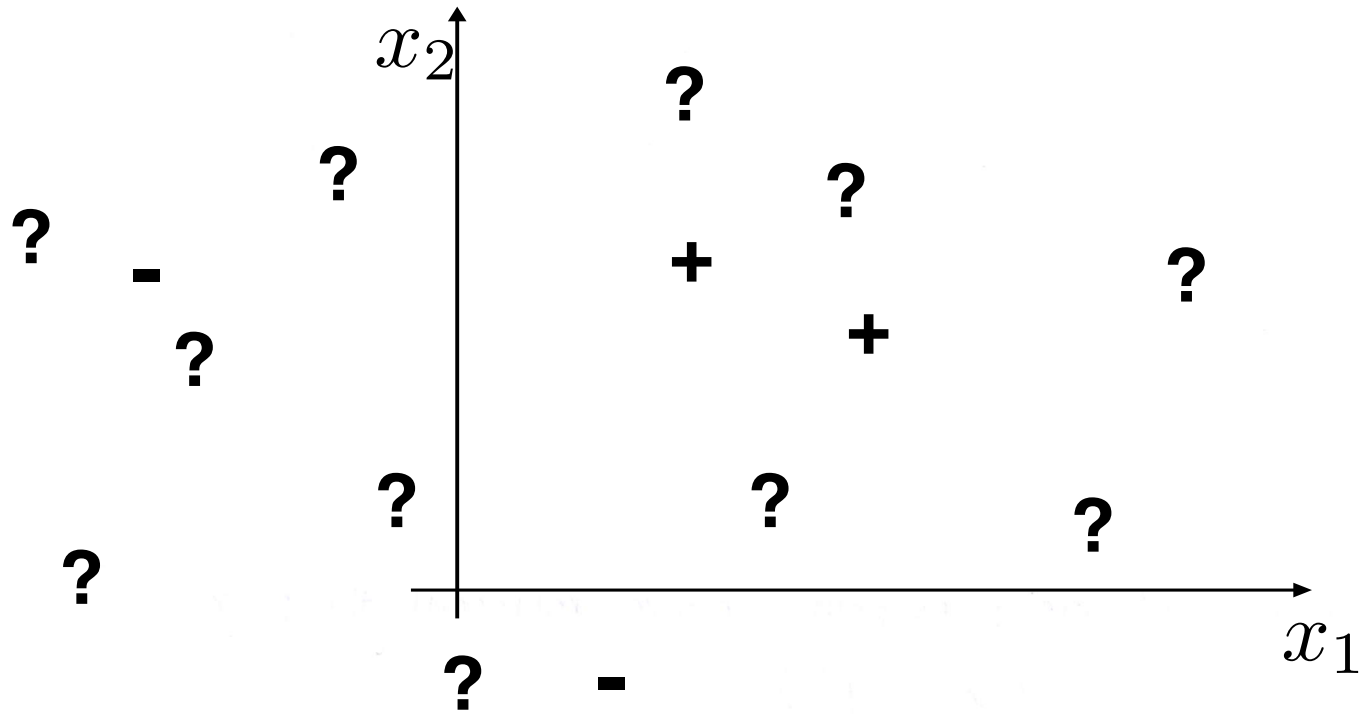
▷ Test Set

Now as decided / discussed earlier we plan to predict whether the viewer will prefer to watch the movie, in which case it will be labelled +1 or prefer not to watch the movie, in which case it will be labelled -1

This will be done on a new set of movies disjoint from our training set As the feature vector of ith movie, is completely a property of the movie itself. This set of new movies will definitely have feature vectors. What we don't know beforehand will be their respective Labels.

But as long as we know the feature vectors. we can draw the points on a scatter plot and put '?' in place of '+' or '-' as labels

This set for which our algorithm/machine predicts the label is called the Test Set. The scatter plot is shown below



▷ Classifier

Now we have been babbling about how we will classify label new movies with +1 or -1 labels. But how do we do this mathematically.

The obvious answer is functions

We will use a function also called a mapping also called classifier (here) called $h()$, it will take as input the feature vector and spit out the label +1 or -1 according to some rule

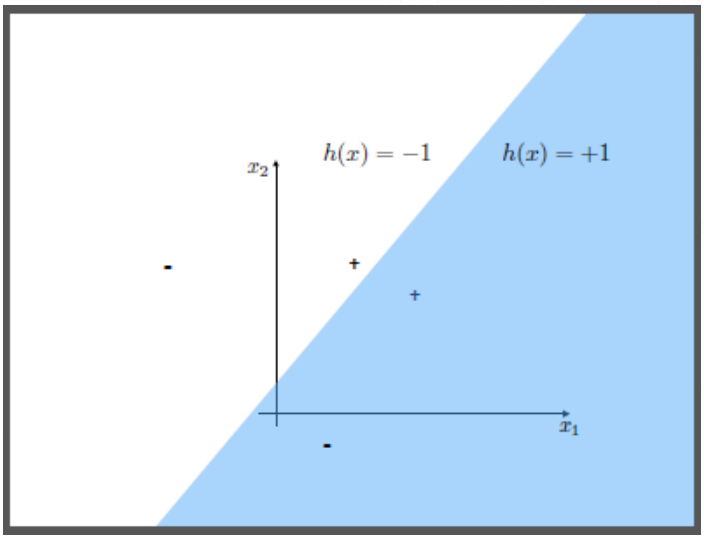
This is the most general thing & most obvious thing we can do.

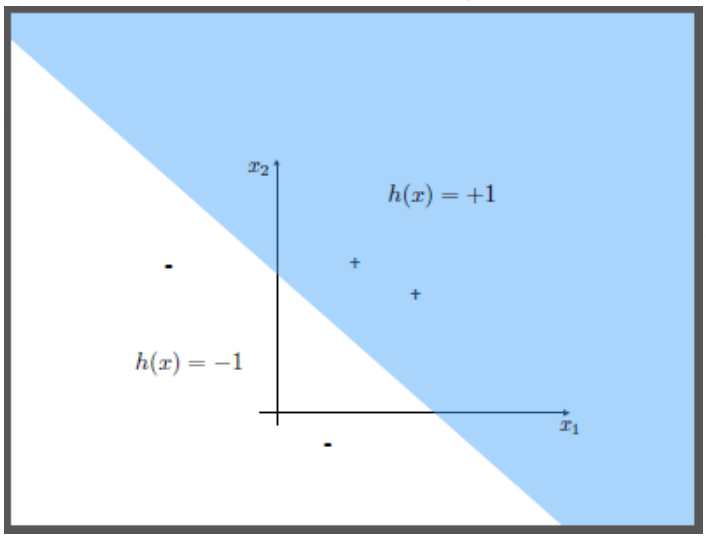
How we choose h will be see later but for now just understand

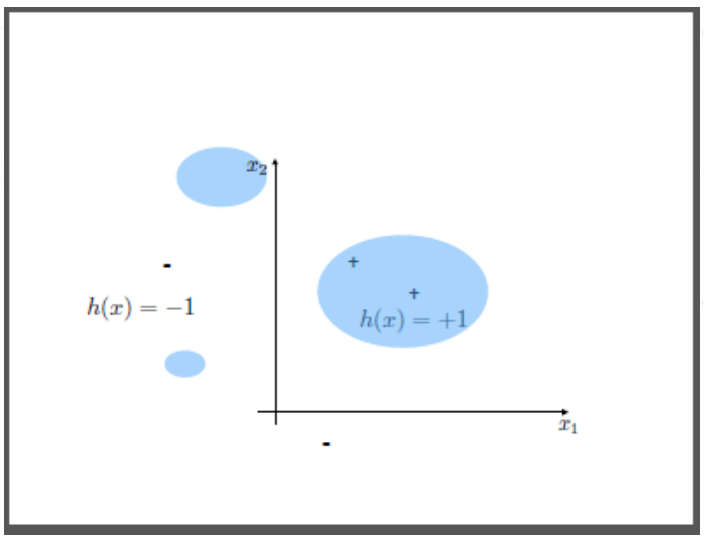
A classifier is a function / mapping that takes elements (vectors) from the space in which feature vectors live, which we denote here by $\mathcal{X}$ and maps it into the categories or label space , here it's $\{-1, 1\}$

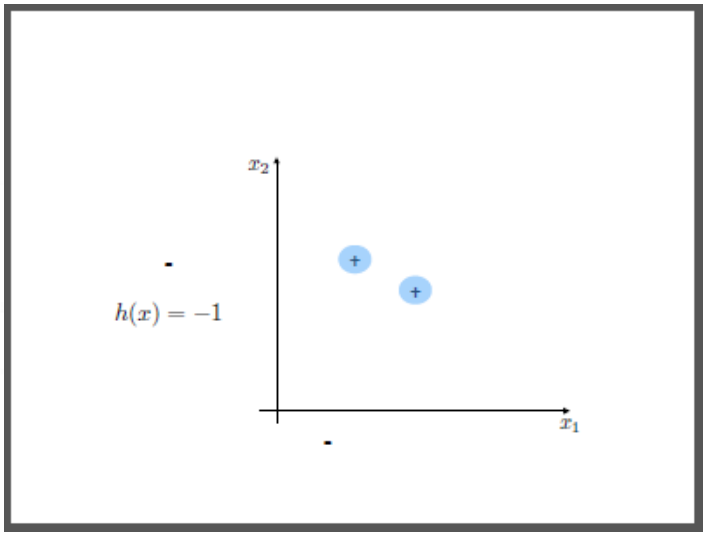
$$h : \mathcal{X} \longrightarrow \{-1, +1\}$$

Now note that we have not constrained our classifier in any way it can be any function for now, no matter how complicated.
What a classifier (for our example) will definitely do is divide the region $\mathcal{X}$ into two parts. Because each vector will either be mapped to $+1$ or -1. We can see some examples of this phenomenon visually.
The blue region is where our classifier labels the x vectors inside the region, as $+1$ ie $h(x) = +1$ in blue region. And the white region is where vectors are labelled -1 ie $h(x) = -1$ in the white region. Let's see some examples below

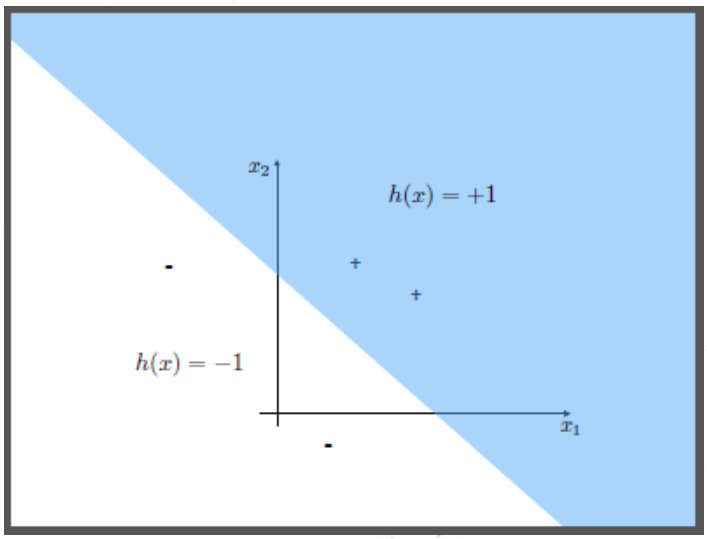


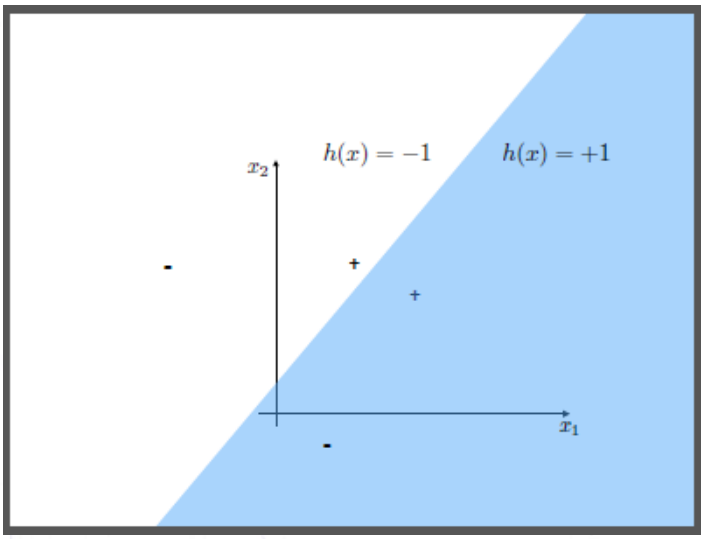






Now look at the following two classifiers





we get a feeling that something is wrong with the right one, and something is right with the Left one. (Pun definitely intended)

And the reason is pretty clear. If you look carefully at the right one a movie with '−' label lies in the blue region & a movie with '+' label lies in the white region.

Which mean the right (aboveplotted) classifier will misclassify these two movies which we beforehand know the Label of. These movies belong to the training set. We haven't even talked about the test set yet.

Why would we want such a classifier that Labels 2 out of 4 movies wrongly when we already know their correct Labels. Wonder what it will do the test set?

The left classifier (plotted above) at Least classifies all the examples in the training set correctly, so intuitively it's a much better classifier than the right one.

So here we see an issue, which tells us that any classifier needs to be evaluated on the basis of how it interacts with the training set

We need a quantity that can measure how much a classifier makes an error with respect to the training set and that is what we will study next.

> Training Error : $E_n(h)$

The idea is pretty simple. We want to see what fraction of points in the training set does our classifier h, classifies incorrectly. We are taking points from the training set so we

→ have each point/vector with a label $y^{(i)}$

→ We will compute what our classifier predicts for $x^{(i)}$, ie we find $h(x^{(i)})$

→ If $h(x^{(i)}) \neq y^{(i)}$, then the classifier committed an error

We will use the double bracket in order to assign 1 or 0

$$[\![\text{Expression}]\!] \begin{cases} \rightarrow 1 \text{ if Expression TRUE} \\ \rightarrow 0 \text{ if Expression FALSE} \end{cases}$$

therefore

$$[\![h(x^{(i)}) \neq y^{(i)}]\!] \begin{cases} \rightarrow 1 \text{ if error is committed} \\ \rightarrow 0 \text{ if error is not committed} \end{cases}$$

→ we will then sum all these evaluations for each point/vector in trainig data set

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

→ and finally in order to find the total fraction we divide by n

so we have

$$\text{Training Error : } E_n(h) = \frac{1}{n} \sum_{i=1}^{n} [\![h(x^{(i)}) \neq y^{(i)}]\!]$$

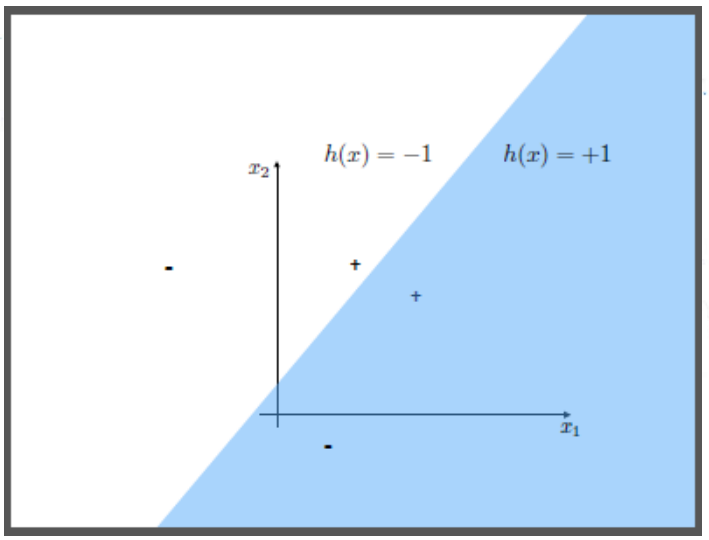
This will give us an idea of the classifiers performance wrt the training data set.

Lets see some examples of these calculations for our examples presented previously. The formula we stated above is for general purposes. In our example here we dont need to keep track of notations that carefully as we deal with $n=4$ case. We will just go to the the 4 points and see how many of them are misclassified by h.

'+' in white region is misclassified

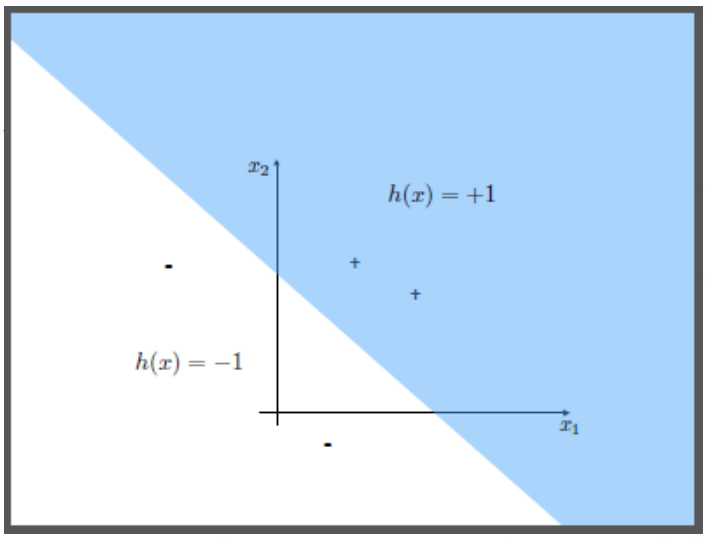
'−' in blue region is misclassified

Then we divide them by 4. Let's do it.



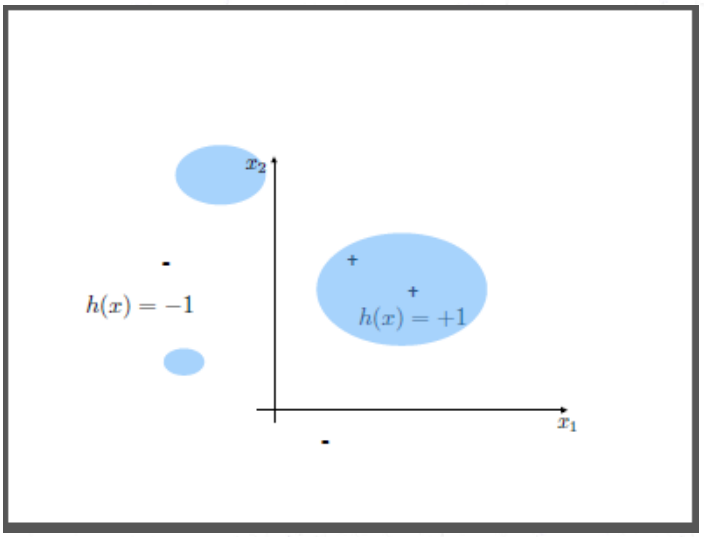
one ' + ' is in white region
one ' - ' is in blue region

So E_n (h) = 2/4 = 0.5



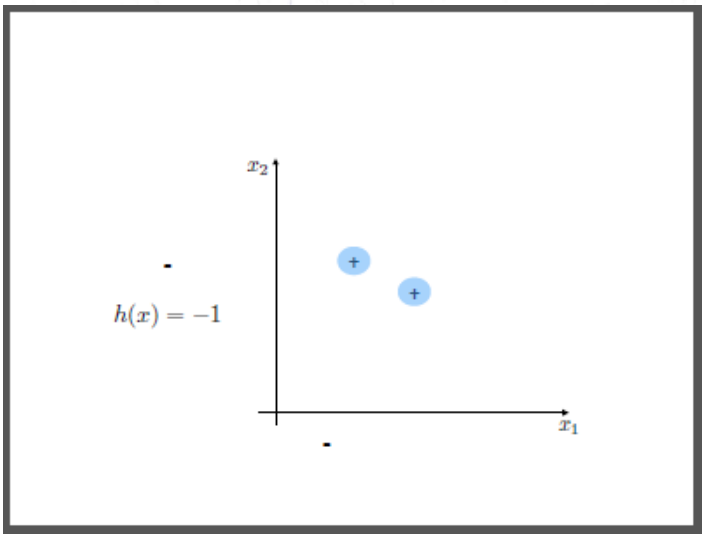
No ' + ' or ' - ' is in wrong region

So En (h) = 0/4 = 0



No ' + ' or ' - ' is in wrong region

So En (h) = 0/4 = 0

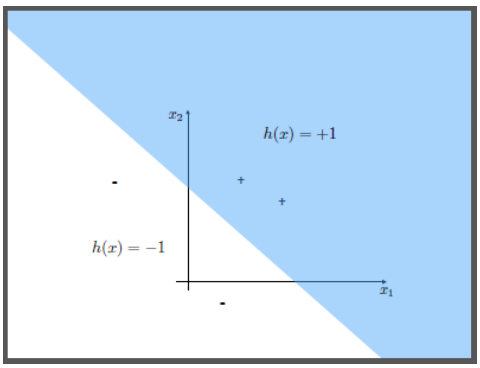


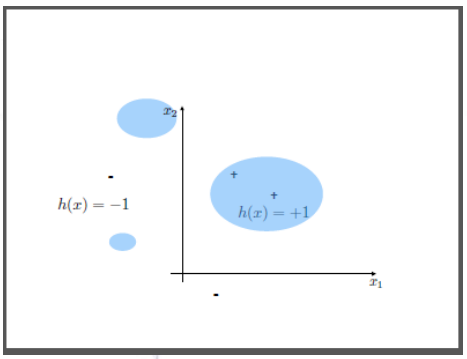
No ' + ' or ' - ' is in wrong region

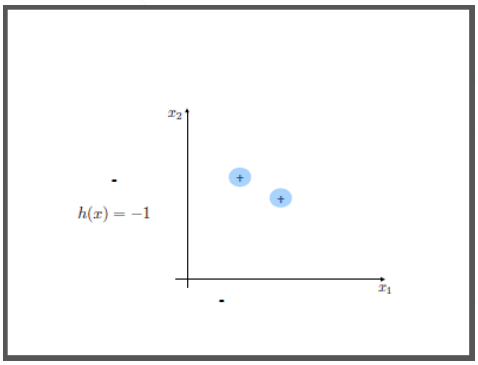
So En (h) = 0/4 = 0

> Generalization

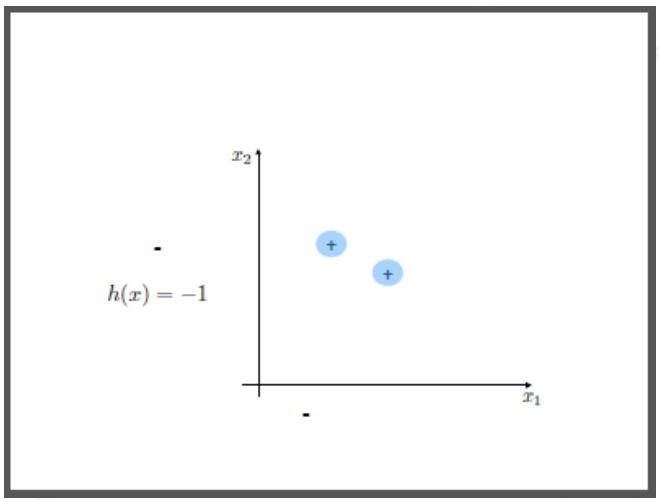
Now look at the following three classifiers

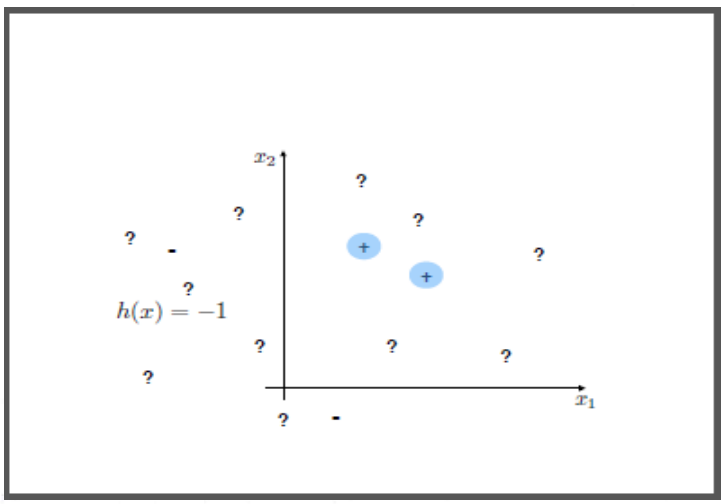






All three of them has training error equal to 0. But we have a problem here. To see that lets look at the 3rd case.





It (the classifier) works fine on the training data. But as soon as we see / plot our test cases on the scatter plot, we begin to see the problem. The blue region of our classifier above, is so narrow that only the feature vectors that are very close (in the near vicinity) of the training examples will be labelled +1, all other will fall in the white region and therefore labelled -1.

For our example above the '?' represents points / vectors from the test set. None of them fall in the blue region, due to its narrowness. We don't have the labels of these examples beforehand, so its a fair bet to assume that half of them will be +1 (label) in reality and half of them -1 (label) in reality

If we go by this assumption we get a ball park idea of what the Test error would be. ie what fraction of test examples will the classifier h(), misclassify

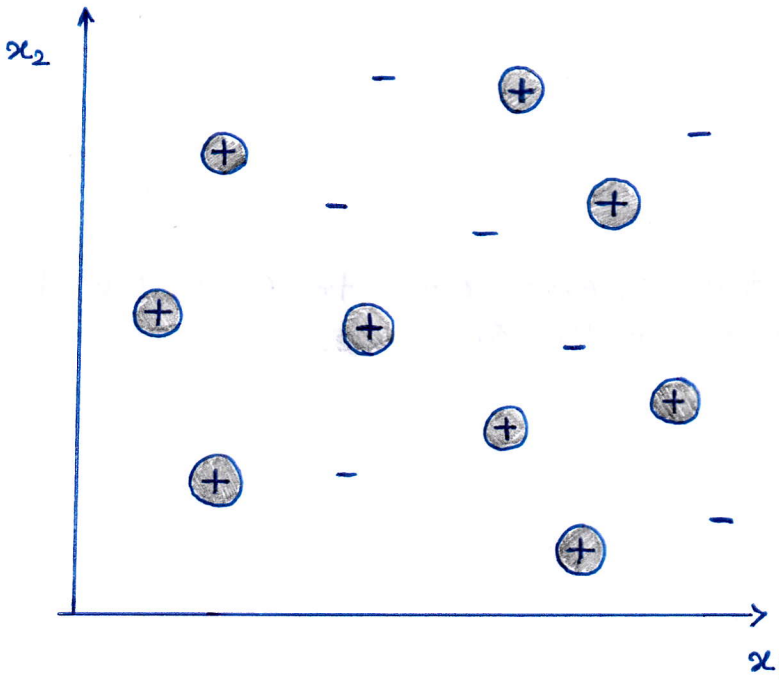
And the answer is around 1/2 ie 50%.

So this classifier is no good then a coin toss when it comes to predicting labels for new data. Although it's test error was 0

The issue we are facing here is generalization. We must address the question about how well the classifier that we train on the training set, generalize or apply correctly, similiarly to the test examples.

And we want models that can generalize from training set to test set properly. But there is a mathematical issue here

Look at the following classifier



No matter how many points we have in our training set. I can always find a mapping $h()$ that will wrap a small circle around my +1 vectors for every +1 Labeled vector in train set. This will then give me a classifier $h()$ which will have training error $\varepsilon_n(h) = 0$ always. But this will again lead us to a coin flip prediction for test data.

This is because we were free to choose <u>any</u> classifier that we liked.

Define

$\mathcal{H}$: Hypothesis space : The set of all possible classifiers

And the problem was we had this huge $\mathcal{H}$ that allowed us to choose any function $h()$ of our liking.

So we improve generalization when we constraint $\mathcal{H}$ ie we only allow our classifier to be from a limited set $\mathcal{H}$ and not anything we want

Put another way the more complex set of classifiers we consider, the less likely we are to generalize.

▷ Supervised Learning : Revisited

Now in the next lecture we will continue our journey to explore the classifier and see how we constraint our classifier to a family of linear function. But before that we see that this idea of function h() can be used to solve many problems like

- Multi-way classification (e.g., three-way classification)

$$h\left(\text{  }\right) = \text{politics} \qquad h : \mathcal{X} \to \{\text{politics}, \text{sports}, \text{other}\}$$

- Regression

$$h\left(\text{  }\right) = \$1{,}349{,}000 \qquad h : \mathcal{X} \to \mathbb{R}$$

- Structured prediction

$$h\left(\text{  }\right) = \begin{array}{l}\text{A group of people}\\ \text{shopping at an}\\ \text{outdoor market}\end{array} \qquad h : \mathcal{X} \to \{\text{English sentences}\}$$

The type of h() we choose will depend on the type of problem we want to tackle, some of which are listed above.

Finally to end this lecture we see some other king of machine learning problems. Some of these will be studied later in the course.

Types of machine learning

- Supervised learning
  - prediction based on examples of correct behavior
- Unsupervised learning
  - no explicit target, only data, goal to model/discover
- Semi-supervised learning
  - supplement limited annotations with unsupervised learning
- Active learning
  - learn to query the examples actually needed for learning
- Transfer learning
  - how to apply what you have learned from A to B
- Reinforcement learning
  - learning to act, not just predict; goal to optimize the consequences of actions
- Etc.