

Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module 2.3
Lecture - 02
Perceptron

Now, let us go to the next module which is Perceptron.

(Refer Slide Time: 00:18)

The story ahead ...

- What about non-boolean (say, real) inputs ?
- Do we always need to hand code the threshold ?
- Are all inputs equal ? What if we want to assign more weight (importance) to some inputs ?
- What about functions which are not linearly separable ?

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2

So, far the story has been about Boolean input, but are all problems that we deal with, we are only dealing with? Do we always only deal with Boolean inputs? So, yeah so, what we spoke about is Boolean functions, right. Now, consider this example. This worked fine for a movie example where we had these as actor so much and his director and so on. But now consider the example where you are trying to decide. You are in oil mining company and you are trying to decide whether you should mine or drill at a particular station or not, right.

Now, this could depend on various factors like what is the pressure on the surface, on the ocean surface at that point, what is the salinity of the water at that point, what is the aquatic marina aquatic life at that point and so on, right. So, these are not really Boolean function, right. The salinity is a real number, density would be a real number, pressure

would be a real number and so on, right and this is a very valid decision problem, right. Companies would be interested in doing this, right. So, in such cases our inputs are going to be real, but so far may call up its neuron only deals with boolean inputs, right. So, we still need to take care of that limitation.

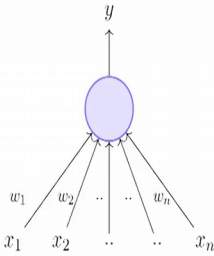
Now, how did we decide the threshold in all these cases? I just asked you, you computed it and you told me right, but that is not going to work out. I mean it does not scale to larger problems where you have many more dimensions and the inputs are not Boolean and so on, right. So, we need a way of learning this threshold.

Now, again returning to the movie example; maybe for me the actor is the only thing that matters and all the other inputs are not so important. Then, what do I need actually? I need some way of weighing these inputs, right. I should be able to say that this input is more important than the others, right. Now, I am treating all of them equal. I am just taking a simple sum.

If that sum causes a threshold, I am fine otherwise I am not fine right, but maybe I want to raise the weight for some of these inputs or lower the weight for some of these inputs, right. So, whether it is raining outside or not maybe does not matter. I have a car, I could go or I could wear a jacket or an umbrella or something, right. So, that input is probably not so important, right.

What about functions which are not linearly separable, right? We have just been dealing with the goody stuff which is all linearly separable, but we will see that even in the restricted Boolean case, there could be some functions which are not linearly separable and if that is the case, how do we deal with it, right. So, these are some questions that we need to answer.

(Refer Slide Time: 02:35)



- Frank Rosenblatt, an American psychologist, proposed the **classical perceptron** model (1958)
- A more general computational model than McCulloch–Pitts neurons
- **Main differences:** Introduction of numerical weights for inputs and a mechanism for learning these weights
- Inputs are no longer limited to boolean values
- Refined and carefully analyzed by Minsky and Papert (1969) - their model is referred to as the **perceptron** model here

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2 22/09

So, first we will start with perceptron which tries to fix some of these things and then, we will move forward from them. So, as we had discussed in the history lecture that this was proposed in 1958 by Frank Rosenblatt and this is what the perceptron looks like. Do you see any difference with the McCulloch Pitts neuron weights, right? You have a weight associated with each of the input otherwise everything seems, right.

So, this is a more general computational model than the McCulloch Pitts neuron. The other interesting thing is that of course we have introduced these weights and you also have a mechanism for learning these weights. So, remember in the earlier case, our only parameter was theta which we are kind of hand setting right, but now with the perceptron, we will have a learning algorithm which will not just help us learn theta, but also these weights for the inputs, right.

How do I know that actor is what matters or director is what matters? Given a lot of past viewing experience, right past given a lot of data about the movies which I have watched in the past, how do I know which are the weights to assign this, right. So, we will see an algorithm which will help us do that, right and the inputs are no longer limited to be Boolean values. They can be real values also, right. So, that is the classical perceptron, but what I am talking about here and the rest of the lecture is the refined version which was proposed by Minsky and Papert which is known as the perceptron model, right. So,

when I say perceptron, I am referring to this model. So, this diagram also corresponds to that.

(Refer Slide Time: 04:06)

y

$w_0 = -\theta$ w_1 w_2 \dots w_n

$x_0 = 1$ x_1 x_2 \dots x_n

y

$$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i \geq \theta$$

$$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i < \theta$$

Rewriting the above,

A more accepted convention,

$$y = 1 \quad \text{if } \sum_{i=0}^n w_i * x_i \geq 0$$

$$= 0 \quad \text{if } \sum_{i=0}^n w_i * x_i < 0$$

where, $x_0 = 1$ and $w_0 = -\theta$

$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta \geq 0$

$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta < 0$

23/00

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2

So, now let us see what the perceptron does. This is how it operates. It will give an output of 1 if the weighted sum of the inputs is greater than a threshold, right. So, remember that in the m p neuron we did not have these weights, but now we have these weighted sum of the inputs and the output is going to be 0 if this weighted sum is less than threshold, right. It is not very different from the m p neuron, right.

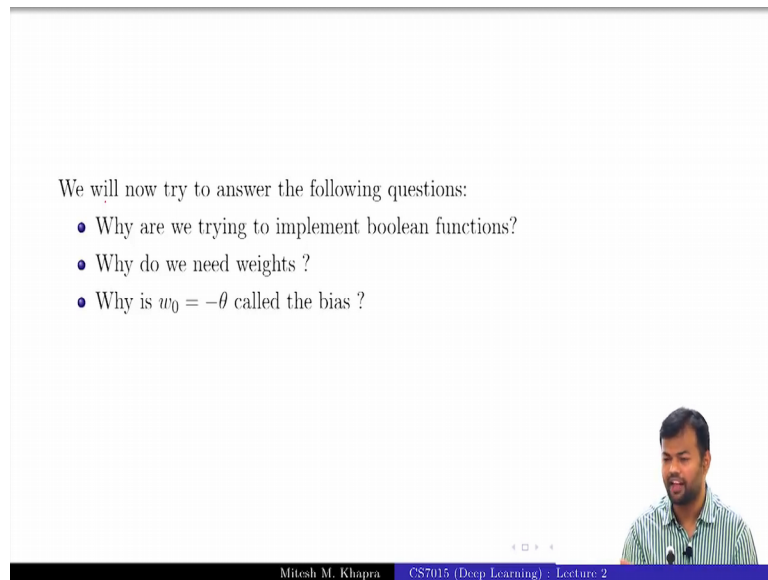
Now, I am just going to do some trickery and try to get it to a better notation or a better form, right. So, is this? I have just taken the theta on this side. Now is this ok? Notice this here the indices were 1 to n. Now, I have made it 0 to n and the theta is suddenly disappeared. So, what has happened.

Student: w_0 is.

Minus theta, right and x_0 is 1. Does anyone not get this right. If I just start it from 1 to n, then it would be summation i equal to 1 to n $w_i * x_i$ plus $w_0 * x_0$, but I am just saying w_0 is equal to minus theta and x_0 is equal to 1 which exactly gives me back this, right; So, very simple x_0 equal to 1 and w_0 is equal to minus theta. So, in effect what I am assuming is that instead of having this threshold as a separate quantity, I just think that that is one of my inputs which is always on and the weight of that input is minus theta.

So, now the job of all these other inputs and their weights is to make sure that their sum is greater than this input which we have, right; does not make sense, fine. So, this is how this is the more accepted convention for writing the perceptron equation, right. So, it fires when this summation is greater than equal to 0, otherwise it does not fire, ok.

(Refer Slide Time: 06:07)



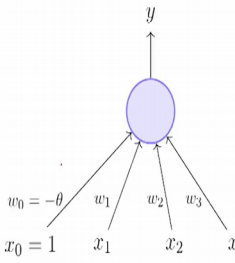
We will now try to answer the following questions:

- Why are we trying to implement boolean functions?
- Why do we need weights ?
- Why is $w_0 = -\theta$ called the bias ?

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2


Now, let me ask a few questions, right. So, why are we trying to implement Boolean functions. I have already answered this, but I will keep repeating this question, so that it really gets drill in. Why do we need weights? Again we briefly touched upon that and why is w_0 which is negative of theta often called the bias?

(Refer Slide Time: 06:25)



- Consider the task of predicting whether we would like a movie or not
- Suppose, we base our decision on 3 inputs (binary, for simplicity)
- Based on our past viewing experience (**data**), we may give a high weight to *isDirectorNolan* as compared to the other inputs
- Specifically, even if the actor is not *Matt Damon* and the genre is not *thriller* we would still want to cross the threshold θ by assigning a high weight to *isDirectorNolan*

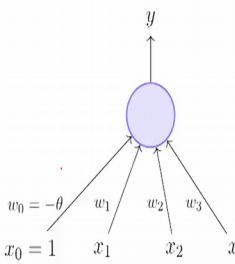
$x_1 = \text{isActorDamon}$
 $x_2 = \text{isGenreThriller}$
 $x_3 = \text{isDirectorNolan}$



Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2


So, again let us return back to the task of predicting whether you would like to watch a movie or not and suppose we base our decisions on three simple inputs; actor genre and director, right. Now, based on our past viewing experience, we may give a high weight to Nolan as compared to the other inputs. So, what does that mean? It means that as long as the director is Christopher Nolan, I am going to watch this movie irrespective of who the actor is or what the genre of the movie, right. So, that is exactly what we want and that is the reason why we want these weights.

(Refer Slide Time: 06:58)



- w_0 is called the bias as it represents the prior (prejudice)
- A movie buff may have a very low threshold and may watch any movie irrespective of the genre, actor, director [$\theta = 0$]
- On the other hand, a selective viewer may only watch thrillers starring Matt Damon and directed by Nolan [$\theta = 3$]
- The weights (w_1, w_2, \dots, w_n) and the bias (w_0) will depend on the data (viewer history in this case)

$x_1 = \text{isActorDamon}$
 $x_2 = \text{isGenreThriller}$
 $x_3 = \text{isDirectorNolan}$

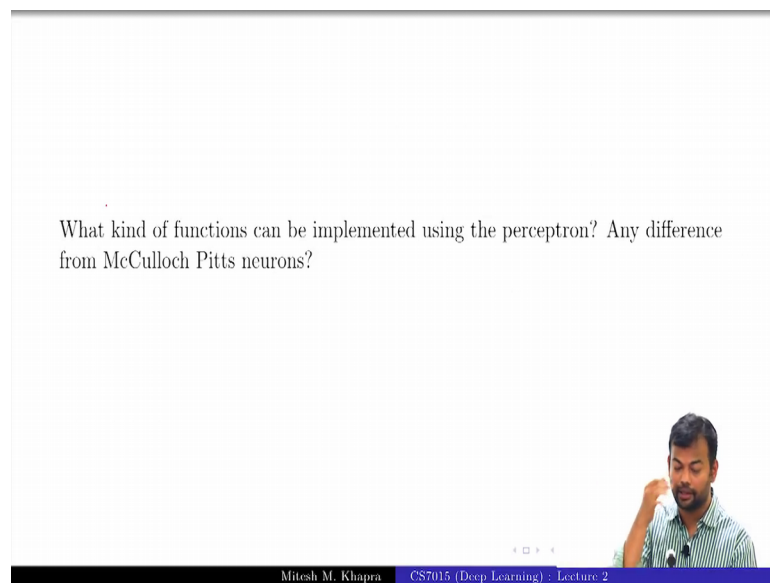


Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2

Now, w_0 is often called the bias as it represents the prior. So, now let me ask a very simple question. Suppose you are a movie buff. What would θ be? 0 right. I mean you will watch any movie irrespective of who the actor, director and genre, right. Now, suppose you are a very niche movie watcher who only watches those movies which are which the genre is thriller, the director was Christopher Nolan and the actor was Damon, then what would your threshold be? 3 right.

High in this case; I always ask this question do you know of any such movie always takes a while. Interstellar so, the weights and the bias will depend on the data which in this case is the viewer history, right. So, that is the whole setup, right. That is why you want these weights and that is why you want these biases and that is why we want to learn them,.

(Refer Slide Time: 07:48)



Now, before we see whether or how we can learn these weights and biases, one question that we need to ask is what kind of functions can be implemented using the perceptron and are these function any different from the McCulloch Pitts neuron? So, before I go to the next slide, any guesses? I am hearing some interesting answers which are at least partly correct.

(Refer Slide Time: 08:11)

McCulloch Pitts Neuron
(assuming no inhibitory inputs)

$$y = 1 \quad \text{if } \sum_{i=0}^n x_i \geq \theta$$
$$= 0 \quad \text{if } \sum_{i=0}^n x_i < \theta$$

Perceptron

$$y = 1 \quad \text{if } \sum_{i=0}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if } \sum_{i=0}^n w_i * x_i < \theta$$

- From the equations it should be clear that even a perceptron separates the input space into two halves
- All inputs which produce a 1 lie on one side and all inputs which produce a 0 lie on the other side
- In other words, a single perceptron can only be used to implement linearly separable functions
- Then what is the difference? The weights (including threshold) can be learned and the inputs can be real valued
- We will first revisit some boolean functions and then see the perceptron learning algorithm (for learning weights)

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2 27/69

So, this is what a McCulloch Pitts neuron looks like and this is what a perceptron looks like. The only difference is this red part which is weights which have count on it, right. So, it is again clear that what the perceptron also does is, it divides the input space into two halves where all the points for which the output has to be one, would lie on one side of this plane and all the points where which the output should be 0 would lie on the other side of this plane, right. So, it is not doing anything different from what the perceptron was doing. So, then what is the difference?

You have these weights and you have a mechanism for learning these weights as well as a threshold. We are not going to hand code them. So, we will first revisit some Boolean functions and then, see the perceptron learning algorithm, ok.

(Refer Slide Time: 08:55)

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

- One possible solution to this set of inequalities is $w_0 = -1, w_1 = 1.1, w_2 = 1.1$ (and various other solutions are possible)
- Note that we can come up with a similar set of inequalities and find the value of θ for a McCulloch Pitts neuron also (Try it!)

$-1 + 1.1x_1 + 1.1x_2 = 0$

(0,1) (1,1)
(0,0) (1,0)

NPTEL Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2 28/60

So, now let us see what the first condition does, right. This condition if I actually expand it out, then this is what it turns out to be, right and what is that condition telling me actually w_0 should be less than 0, clear. So, now based on these, what do you have here? Actually what is this? A system of linear inequalities, right and you know you could solve this, right. You have algorithms for solving this not always, but you could find some solution, right and one possible solution which I have given you here is w_0 is equal to minus 1 w_1 equal to 1.1 and w_2 equal to 1.1.

So, just let us just draw that line, ok. So, what is the line? It is $1.1x_1 + 1.1x_2$ is equal to 1, right. That is the line and this is the line and you see it satisfies the conditions that I have is this the only solution possible. No, right I could have this also as a valid line. If I could draw properly, right all of these are valid solutions, right. So, it is result in different w_1 w_0 s, ok. So, all of these are possible solutions.

In fact, I have been telling you that you had to set the threshold by hand for the McCulloch Pitts neuron, but that is not true because you could have written similar equations there and then, decided what the value of theta should be, right. So, you could try this out for the McCulloch Pitts neuron. Also, you will get a similar set of conditions or I mean similar set of inequalities and you can just say what is the value of theta, that you could set to solve that right. So, that ends that module.