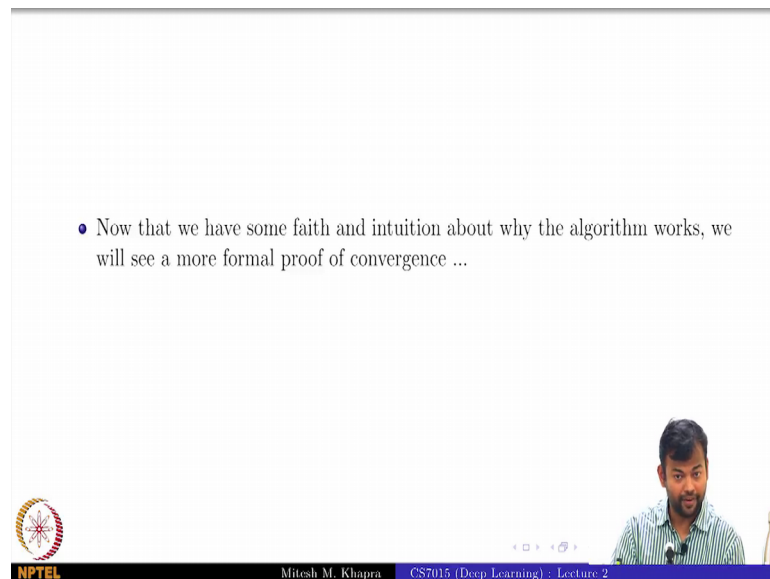


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 2.6
Lecture – 02
Proof of Convergence

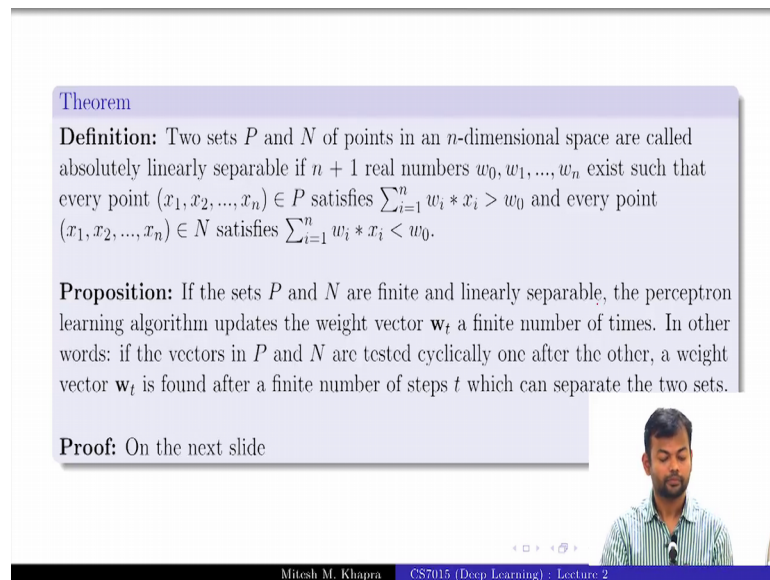
In this module we will talk about the Proof of Convergence for the Perceptron or the Learning Algorithm that we saw in the previous module.

(Refer Slide Time: 00:21)



So, we have some faith and intuition that it actually works, we just need to formally prove it that it actually converges right. So, that is what we are going to do in this module.

(Refer Slide Time: 00:31)



Theorem

Definition: Two sets P and N of points in an n -dimensional space are called absolutely linearly separable if $n + 1$ real numbers w_0, w_1, \dots, w_n exist such that every point $(x_1, x_2, \dots, x_n) \in P$ satisfies $\sum_{i=1}^n w_i * x_i > w_0$ and every point $(x_1, x_2, \dots, x_n) \in N$ satisfies $\sum_{i=1}^n w_i * x_i < w_0$.

Proposition: If the sets P and N are finite and linearly separable, the perceptron learning algorithm updates the weight vector \mathbf{w}_t a finite number of times. In other words: if the vectors in P and N are tested cyclically one after the other, a weight vector \mathbf{w}_t is found after a finite number of steps t which can separate the two sets.

Proof: On the next slide

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

So, before that a very few very simple definitions. So, if you have two sets of points P and N in an n dimensional space and we call say that these points are absolutely linearly separable, if there exists some n plus 1 real numbers which has w_0 to w_n ; such that every point which belongs to P right, P is the case where the output is 1.

Then these set of weights satisfy this condition right and every point which lies in the negative set the set of weights satisfy this condition right. So, nothing very different from what has we have been saying. So, far it is just formally defining it right.

Now, our proposition is that, if the set P and N are finite and there is a fixed number of points in that which was the case in the toy example that we were doing and which will be the case in most examples that, we do and linearly separable right. The perceptron learning algorithm updates the weight vector ok. Before I go there right ok, let me not give you the definition and let me ask you the definition right.

So, now I have given this definition, the first definition and given this part of the proposition. Can you tell me what do I need to prove if I need to prove that the algorithm converges, that is one way of looking at it, but what was happening in that wrong argument which was I was making that it continuously kept toggling right. That means, I am not making a finite number of updates right I have to keep changing again and again and this process continues in a loop right.

So, that is how I am going to define convergence that the perceptron learning algorithm updates a weight vector of finite number of times right, it only needs to update it finite number of times and it will reach a configuration such that now it is able to separate the P from the N ok; that is what the proof of convergence means right.

So, in other words if you are going to pick up these vectors randomly from the set P and N cyclically, as we were doing in the toy example, then a weight vector w is found after a finite number of steps which will separate these two sets, these two sets right. So, that is what we are trying to prove. So, that is the definition of converge, does it make sense? Right. .

(Refer Slide Time: 03:02)

<p>Setup:</p> <ul style="list-style-type: none"> • If $x \in N$ then $-x \in P$ ($\because w^T x < 0 \implies w^T(-x) \geq 0$) • We can thus consider a single set $P' = P \cup N^-$ and for every element $p \in P'$ ensure that $w^T p \geq 0$ • Further we will normalize all the p's so that $\ p\ = 1$ (notice that this does not affect the solution \because if $w^T \frac{p}{\ p\ } \geq 0$ then $w^T p \geq 0$) • Let w^* be the normalized solution vector (we know one exists as the data is linearly separable) 	<p>Algorithm: Perceptron Learning Algorithm</p> <hr/> <pre> P ← inputs with label 1; N ← inputs with label 0; N⁻ contains negations of all points in N; P' ← P ∪ N⁻; Initialize w randomly; while !convergence do Pick random p ∈ P'; p ← p / p (so p now becomes unit norm); if w · p < 0 then w = w + p; end end </pre> <p>//the algorithm converges when all the inputs are classified correctly //notice that we do not need the other if condition because by construction we want all points in P' to lie in the positive half space $w \cdot p \geq 0$</p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

5/69

So, proof is on the next slide and it is going to take me around 5 to 10 minutes to prove it. So, just stay focused all right. So, here is a few set up right. So, I am going to, before I go to the actual proof I am going to make a set up so that it becomes easier for us to prove it right. So, the first thing that I am going to say is that, if there is a point which belongs in negative set then the negative of that point belongs in the positive set and that is very clear, because if the point belongs in the negative set then $w^T x$ is less than 0.

But then $w^T (-x)$ would be greater than equal to 0 right. So, I take the negative of the point, I can just put it in the positive set. So, instead of considering these two different things P and N I am just going to consider one P prime, which is an union

of P and all the N points negative ok, will the set up clear. If this is a setup then what is the condition that I need to ensure for every point in P dash.

Student: (Refer Time: 03:57).

W transpose p should be greater than equal to 0 right. So, I do not care about the negative case, I have just made everything positive now and it is, I am not done anything wrong here, it is just a simple trick. And now this is how the algorithm will look in this setup, these are the inputs with label one inputs with label 0 N minus contains a negation of all the points in N and P prime is a union of these. Now again I start by initializing w randomly, while convergence I will do something, I will pick a random P from P prime. Now what is the, if condition.

Less than 0.

Student: (Refer Time: 04:35).

Do I need the other if condition.

Student: No.

No right, because everything is now positive and the other small thing that I am going to do is, I am going to normalize P . So, that again does not mean, because we are talking in terms of angles and I am not changing the direction of the vector, I am just shrinking it right. So, I am just or maybe scaling it, also I am just making it unit norm. So, that does not change anything right. So, it is still everything still holds.

And in particular you can see here right. So, if this condition was true, this condition will also be true. So, so far just I am done some simple tricks to make things easier for me later on, so now P has been normalized. Now remember that this data is linearly separable; that is what we started the proposition. If P and N are linearly separable then the perceptron learning algorithm will converge right. So, now, if P and N are linearly separable, irrespective of whether we have the perceptron learning algorithm or not what do we know?

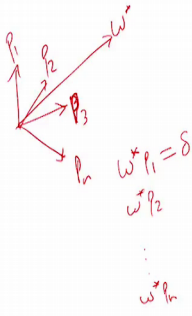

Student: (Refer Time: 05:34).

That there exists.

Student: Line (Refer Time: 05:37).

There exists a w^* which is the solution vector right, there exists at least one w^* which is the solution vector right; such that it will separate the P points from the N points. So, this vector which we do not know, but we just know that it exists, so you can refer to it. So, we will call this w^* fine. Now we start the proof.

(Refer Slide Time: 05:56)

Observations:	Proof:
<ul style="list-style-type: none">w^* is some optimal solution which exists but we don't know what it is	<ul style="list-style-type: none">Now suppose at time step t we inspected the point p_i and found that $w_t^T \cdot p_i \leq 0$We make a correction $w_{t+1} = w_t + p_i$Let β be the angle between w^* and w_{t+1}
	$\cos \beta = \frac{w^* \cdot w_{t+1}}{\ w_{t+1}\ }$ $\text{Numerator} = w^* \cdot w_{t+1} = w^* \cdot (w_t + p_i)$ $= w^* \cdot w_t + \underline{\underline{w^* \cdot p_i}}$
	
<p>Mitesh M. Khapra CS7015 (Deep Learning) - Lecture 2</p>	

So, w^* is some optimal solution which we know exists.

But we do not know what it is right. Now suppose you had a time step t . So, remember that this algorithm is going on while convergence. So, you have time step 1 2 3 you are picking up points. So, we are at a time step t , at which you pick up a random point p_i and you find that the condition is actually violated. So, this should actually be less than 0, if I know the condition is violated. So, now, what will you have to do?

Student: (Refer Time: 06:26).

w is equal to.

Student: (Refer Time: 06:29).

w_{t+1} . So, I will just call it the new w w_{t+1} is equal to the old w plus p_i . Now what I am going to do is I am going to consider the angle β between w^* and w_{t+1} . I do not know what w^* is, but we can still assume it exists and make some calculations

based on that right. So, what is the angle between w^* and w^{t+1} , its β and what is the cost of that angle this

Student: (Refer Time: 06:54).

And remember that we do not have w^* here, because we had assumed that it is the normalized vector right. So, we do not need that bit, this is actually equal to 1. So, now, if I just take the numerator, w^* in dot product w^{t+1} now I am going to expand w^* w^{t+1} p_i fair; that is exactly what I did on the previous step, is it fine.

Now, now what is p_i actually, it is. So, what you had is you had these $P_1 P_2 P_3$. My hand writing is really horrible and up to P_n right. So, I have just picked one of these p_i 's. Now what I am going to define is. Now suppose this is my, these are my p_i 's right. So, these are all the vectors that I have. Now suppose I have this w^* , suppose this was the w^* that I am interested.

Now, for each of these I could compute $w^* P_1 w^* P_2$ and so on up to $w^* P_n$ right and I could sort them. Now what I am doing is that for whichever of these points $w^* p_i$ is the minimum ok, I am going to call that value as δ . Suppose $w^* P_1$ is the smallest quantity out of $w^* P_1 w^* P_2 w^* P_n$ right, and I am calling that quantity δ .

So, I have this quantity here and my δ is the minimum of all the possible values that it can take. It can make $w^* P_1 P_2$ up to P_n , so δ is the minimum quantity. So, here I have an equality.

(Refer Slide Time: 08:45)

Observations:	Proof:
<ul style="list-style-type: none">w^* is some optimal solution which exists but we don't know what it isWe do not make a correction at every time-stepWe make a correction only if $w^T \cdot p_i \leq 0$ at that time stepSo at time-step t we would have made only k ($\leq t$) correctionsEvery time we make a correction a quantity δ gets added to the numeratorSo by time-step t, a quantity $k\delta$ gets added to the numerator	<ul style="list-style-type: none">Now suppose at time step t we inspected the point p_i and found that $w^T \cdot p_i \leq 0$We make a correction $w_{t+1} = w_t + p_i$Let β be the angle between w^* and w_{t+1} $\cos\beta = \frac{w^* \cdot w_{t+1}}{\ w_{t+1}\ }$ $\begin{aligned} \text{Numerator} &= w^* \cdot w_{t+1} = w^* \cdot (w_t + p_i) \\ &= w^* \cdot w_t + w^* \cdot p_i \\ &\geq w^* \cdot w_t + \delta \quad (\delta = \min\{w^* \cdot p_i \forall i\}) \\ &\geq w^* \cdot (w_{t-1} + p_j) + \delta \\ &\geq w^* \cdot w_{t-1} + w^* \cdot p_j + \delta \\ &\geq w^* \cdot w_{t-1} + 2\delta \\ &\geq w^* \cdot w_0 + (k)\delta \quad (\text{By induction}) \end{aligned}$

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2 46/69

Now, are you with this? This is the minimum quantity right. So, any p_i that I put in here it is always going to be greater than or worst case equal to delta fair ok, fine.

Now, again this w_{t+1} itself I could write it as $w_t + p_j$, because that also would have come up from some update in the previous step. Again this is there which I could call it as delta and still retain the greater than equal to here, fine. So, let us see where are we heading with this right.

Now, notice that we do not make a correction at every time step, when I was running that toy algorithm I was not making a correction at every time step. We were only making a correction at those time steps for which the condition was violated. So, now, if I am at this time step, maybe I have made only k which is less than or equal to t corrections. At max I would have made t corrections, but it could have been less than that also.

So, now every time we make a correction we are adding a value delta to this right. So, at the time step t what would happen, I had started off from w_0 I have reached time t and I have made a case that, I have not made t updates I have made k less than equal to t updates right. So, how many deltas would get added?.

Student: $k\delta$.

$k\delta$. So, I could say that with respect to w_0 where I had started from, this is what this quantity is, is that fine, anyone has a problem with this ok.


(Refer Slide Time: 10:19)

Proof (continued:)

So far we have, $w^T \cdot p_i \leq 0$ (and hence we made the correction)

$$\cos\beta = \frac{w^* \cdot w_{t+1}}{\|w_{t+1}\|} \quad (\text{by definition})$$

Numerator $\geq w^* \cdot w_0 + k\delta$ (proved by induction)

$$\begin{aligned} \text{Denominator}^2 &= \|w_{t+1}\|^2 \\ &= (w_t + p_i) \cdot (w_t + p_i) \\ &= \|w_t\|^2 + 2w_t \cdot p_i + \|p_i\|^2 \\ &\leq \|w_t\|^2 + \|p_i\|^2 \quad (\because w_t \cdot p_i \leq 0) \\ &\leq \|w_t\|^2 + 1 \quad (\because \|p_i\|^2 = 1) \\ &\leq (\|w_{t-1}\|^2 + 1) + 1 \\ &\leq \|w_{t-1}\|^2 + 2 \\ &\leq \|w_0\|^2 + (k) \quad (\text{By same observation th}) \end{aligned}$$


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

So, So, far what are we shown right, we started with this, this condition was true again not less than equal to and hence we made the correction and this was the point that we picked up at the t th step and thence we made that correction.

And we also showed that the numerator is actually greater than equal to this quantity right, we showed it by induction fine. Now let us look at the denominator and particularly let us look at the denominator squared, is a step right.

This is actually w_t plus 1 dot product w_t plus 1, but w_t plus 1 can be written as w_t plus p_i , is this. This bracket needs to disappear right, is that fine. Now what is, what is this quantity.

Student: (Refer Time: 11:15).

No.

Student: (Refer Time: 11:17).

That is less than equal to 0. So, now, can you guess what is the next thing that I am going to write right.

Student: (Refer Time: 11:30).

That is correct, yeah it is a negative quantity. So, that is going to be less than equal to this. So, that is fine and what about p_i square or this term.

Student: (Refer Time: 11:44).

Because this is less than right that is why.

Student: yeah (Refer Time: 11:52)

Correct is this fine ok. Now what is p_i square?

Student: 1 (Refer Time: 12:00).

One. Now can you guess what I am going to do by induction?

Student: k.

K; that is. So, what is w_t square again right. Just this w_t plus 1 square was w_t square plus 1, w_t square is going to be w_t minus 1 square plus 1 right and how many such ones will get added k of those right, starting from w naught.

(Refer Slide Time: 12:36)

Proof (continued:)

So far we have, $w^T \cdot p_i \leq 0$ (and hence we made the correction)


$$\cos\beta = \frac{w^* \cdot w_{t+1}}{\|w_{t+1}\|} \quad (\text{by definition})$$

Numerator $\geq w^* \cdot w_0 + k\delta$ (proved by induction)

Denominator² $\leq \|w_0\|^2 + k$ (By same observation that we made about δ)

$$\cos\beta \geq \frac{w^* \cdot w_0 + k\delta}{\sqrt{\|w_0\|^2 + k}}$$

- $\cos\beta$ thus grows proportional to \sqrt{k}
- As k (number of corrections) increases $\cos\beta$ can become arbitrarily large
- But since $\cos\beta \leq 1$, k must be bounded by a maximum number
- Thus, there can only be a finite number of corrections (k) to w and the algorithm will converge!



48/60

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

So, what have we shown, the numerator is greater than equal to this, the denominator is less than this. Now if I put them together I actually get that $\cos\beta$ is going to be greater

than equal to the numerator over the denominator. Now what is this quantity proportional to, k , k square, k cube square root of k , k by 2?

Student: Square root of k .

Square root of k right, you have, I mean roughly speaking you have a k here, you have a square root of k here. So, I could roughly speaking say that it is proportional to square root of k . So, as k grows what will happen to $\cos \beta$, it will grow and that is fine right, it can keep growing.

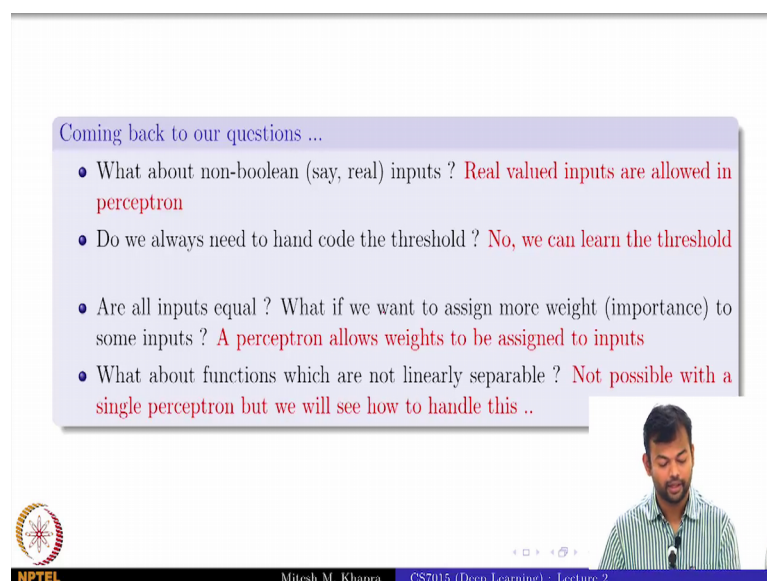
Student: (Refer Time: 13:31).

Only until one right. So, $\cos \beta$ is going to be proportional to k what is k ? The number of updates that you make. Now if I were to take that degenerate case which you guys were hinting at, where that it will keep changing again and again, what will happen to k ? It will keep going to infinity can that happen.

Student: No

No, because $\cos \beta$ will blow up right and that is not allowed. So, k has to be finite, so that $\cos \beta$ stays within its limits right. Hence are we done, how many if you think we are done, how many if you are satisfy that we are done, it is not a trick question that we are done. Please we are done.

(Refer Slide Time: 14:29)



Coming back to our questions ...

- What about non-boolean (say, real) inputs? **Real valued inputs are allowed in perceptron**
- Do we always need to hand code the threshold? **No, we can learn the threshold**
- Are all inputs equal? What if we want to assign more weight (importance) to some inputs? **A perceptron allows weights to be assigned to inputs**
- What about functions which are not linearly separable? **Not possible with a single perceptron but we will see how to handle this ..**

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

So, yeah. So, this says that we can only have a finite number of such k updates that we make and after that the algorithm will converge, is that ok. So, we have a proof of convergence. Now coming back to our questions this is where we had started at one point, what about non billion inputs, so perceptron allows that right. We took I M D B rating and critics rating as an input. Do we always need to hand code the threshold?

Student: No.

No, in our perceptron learning algorithm are all inputs equal, no we now assign weights to input. What about functions which are not linearly separable? We still do not know right. So, that is where we are headed now, not possible with a single perceptron, but we will see how to handle this ok. So, far the story is clear to everyone ok. So, we will end this module here.