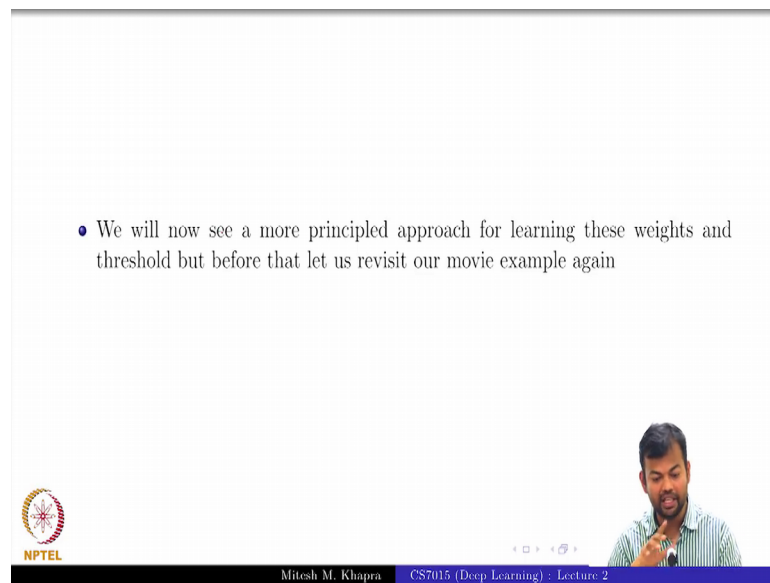


Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 2.5
Lecture - 02
Perceptron Learning Algorithm

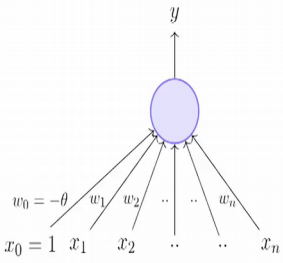
We will now go to the next module obviously Perceptron Learning Algorithm.

(Refer Slide Time: 00:17)



We now see a more principled approach of learning these weights and threshold, but before that we will just again revisit our movie example and make it slightly more complicated.

(Refer Slide Time: 00:25)



The diagram shows a single neuron with an input layer containing nodes $x_0, x_1, x_2, \dots, x_n$ and an output node y . Weights $w_0, w_1, w_2, \dots, w_n$ connect the input nodes to the neuron. The bias weight $w_0 = -\theta$ is associated with $x_0 = 1$.

$w_0 = -\theta$
 $x_0 = 1$ x_1 x_2 \dots x_n
 w_1 w_2 \dots w_n

$x_1 = isActorDamon$
 $x_2 = isGenreThriller$
 $x_3 = isDirectorNolan$
 $x_4 = imdbRating(scaled\ to\ 0\ to\ 1)$
 \dots \dots
 $x_n = criticsRating(scaled\ to\ 0\ to\ 1)$

- Let us reconsider our problem of deciding whether to watch a movie or not
- Suppose we are given a list of m movies and a label (class) associated with each movie indicating whether the user liked this movie or not : binary decision
- Further, suppose we represent each movie with n features (some boolean, some real valued)
- We will assume that the data is linearly separable and we want a perceptron to learn how to make this decision
- In other words, we want the perceptron to find the equation of this separating plane (or find the values of $w_0, w_1, w_2, \dots, w_m$)

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 2 34/69

Now, here what the situation is that we are given a list of movies and a class associated with each movie indicating whether we like the movie or not, right. So, now we have given some data of the past m movies that we have seen and whether we like this movie or not and now instead of these three variables, we have these n different variables based on which we are making decisions. And notice that some of these variables are real, right. They are not Boolean anymore, right. The rating could be any real number between 0 to 1, and now based on this data what do we want is the perceptron to do actually.


So, I have given you some data. These factors I have also given you the label 1 and 0. So, if the perceptron if I tell you my perceptron has now learnt properly, what would you expect it to do; perfect match, right. So, whenever I feed it, one of these movies it should give me the same label as was there in my data, right and again there are some movies for which I have a label 1 which are positive and some movies which I have a label 0.

So, I am once again looking to separate the positives from the negatives, right. So, it should adjust the weights in such a way that I should be able to separate, right. So, that is the learning problem that we are interested in, ok.

(Refer Slide Time: 01:31)

```
Algorithm: Perceptron Learning Algorithm
P ← inputs with label 1;
N ← inputs with label 0;
Initialize  $\mathbf{w} = [w_1, w_2, \dots, w_N]$  randomly;
while !convergence do
    Pick random  $\mathbf{x} \in P \cup N$  ;
    if  $\mathbf{x} \in P$  and  $\sum_{i=0}^n w_i * x_i < 0$  then
        |  $\mathbf{w} = \mathbf{w} + \mathbf{x}$  ;
    end
    if  $\mathbf{x} \in N$  and  $\sum_{i=0}^n w_i * x_i \geq 0$  then
        |  $\mathbf{w} = \mathbf{w} - \mathbf{x}$  ;
    end
end
//the algorithm converges when all the
inputs are classified correctly
```

- Why would this work ?
- To understand why this works we will have to get into a bit of Linear Algebra and a bit of geometry...



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

So, now with that I will give you the algorithm, ok. This is the Perceptron Learning Algorithm. We have certain positive inputs which had the label 1, we have certain negative inputs which had the label 0 and now, I do not know what the weights are and I have no prior knowledge of what the weights are going to be. I need to learn them from the data. So, what I am going to do is, I am just going to initialize these weights randomly as I am also going to pick up some random values for this. So, this should be small n. So, this should be small n and now, here is the algorithm while not convergence do something.

So, before I tell you what to do, can you tell me what is meant by convergence? When will you say that it has converged? When it is not making any more errors on the training data, right, or its predictions are not changing on the training data. So, that is the definition of convergence, ok. Now, here is the algorithm. I pick up a random from point from my data which could either be positive or negative, right. So, it comes from the union of positive negative. Basically all the data that I have I pick up a random point from there.

If the point is positive, right and this is the condition which happens what does this tell me? If the point was positive, what did I actually want greater than 0, but the condition is less than 0. That means, I have made an error. So, I have made an error, then I will just

add x to w I see a lot of thoughtful nodding and I hope you are understanding what is happening. Let us see. So, what is w actually? A dimensional.

Student: N dimension.

N dimension n plus 1, right because w naught is also inside there. So, actually there should be w naught also here, right and what is x again n dimensional, right and that is why this addition is valid. So, let us understand that w and x both are n dimensional, ok. Now, let us look at the other condition. Can you guess what the other if condition is if x belongs to n and.

Student: Summation.

Summation is greater than equal to 0, then? So, that means you have completely understood how this algorithm works. Well, that is right. So, now consider two vectors w and x . So, remember what we are trying to prove is or get an intuition. Not prove, actually get an intuition for why this works.

(Refer Slide Time: 03:55)

• Consider two vectors w and x

$$w = [w_0, w_1, w_2, \dots, w_n]$$
$$x = [1, x_1, x_2, \dots, x_n]$$
$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

• We are interested in finding the line $w^T x = 0$ which divides the input space into two halves

• Every point (x) on this line satisfies the equation $w^T x = 0$

• We can thus rewrite the perceptron rule as

$$y = 1 \quad \text{if } w^T x \geq 0$$
$$= 0 \quad \text{if } w^T x < 0$$

Handwritten notes: $2 \ 2 \ +$, $x_1 + x_2 = 0$, $-1 \ -2$

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

So, we will consider two vectors w and x and this is what my vectors look like very similar to the case that we are considering w_0 to w_n and 1 to n . So, this again x naught is just 1, right.

Now, this condition that I have been talking about is nothing, but the dot product. How many of you have gone through the prerequisites for today's lecture? Ok good. So, it is just a dot product, ok. Now, we can just read write the perceptron rule as this instead of the dot product. I mean instead of using that summation thing, we can just say that it is a dot product.

Now, we are interested in finding the line $\mathbf{w}^T \mathbf{x} = 0$. So, that is our decision boundary, right which divides the input into two halves, ok. Now, every point on this line satisfies the equation $\mathbf{w}^T \mathbf{x} = 0$. What does that mean actually?

So, just a simple example is that if I have the line $x_1 + x_2 = 0$, then all the points which lie on the line satisfy this equation, right. So, you could have 1 minus 1, 2 minus 2 and so on, but 2, 2 cannot be a point on this line. At every point lying on this line satisfies this equation. So, every point lying on this line actually satisfies the equation $\mathbf{w}^T \mathbf{x} = 0$.

(Refer Slide Time: 05:14)

The slide contains the following content:

- Consider two vectors \mathbf{w} and \mathbf{x}

$$\mathbf{w} = [w_0, w_1, w_2, \dots, w_n]$$

$$\mathbf{x} = [1, x_1, x_2, \dots, x_n]$$

$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T \mathbf{x} = \sum_{i=0}^n w_i * x_i$$
- We can thus rewrite the perceptron rule as

$$y = 1 \quad \text{if} \quad \mathbf{w}^T \mathbf{x} \geq 0$$

$$= 0 \quad \text{if} \quad \mathbf{w}^T \mathbf{x} < 0$$
- We are interested in finding the line $\mathbf{w}^T \mathbf{x} = 0$ which divides the input space into two halves
- Every point (\mathbf{x}) on this line satisfies the equation $\mathbf{w}^T \mathbf{x} = 0$
- What can you tell about the angle (α) between \mathbf{w} and any point (\mathbf{x}) which lies on this line ?
- The angle is 90° ($\because \cos \alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|} = 0$)
- Since the vector \mathbf{w} is perpendicular to every point on the line it is actually perpendicular to the line itself

Navigation icons and footer: 36/60, Mitesh M. Khapra, CS7015 (Deep Learning) : Lecture 2

So, can you tell me what is the angle between \mathbf{w} and any point on this line? How many say how many of you say perpendicular? Why?

Dot product is 0, right. So, if the dot product is 0, they are orthogonal right. So, that means if I take this line, then my vector \mathbf{w} is orthogonal to this. It is orthogonal to this point or this point to this point to every point on the line which is just the same as saying

that the vector is perpendicular to the line itself, right as simple as that. So, the angle is 90 degrees because the dot product gives you the cos alpha and that is 0, right and since it is perpendicular as I said to every point of the line it is just perpendicular to the line itself, right.

(Refer Slide Time: 05:58)

- Consider some points (vectors) which lie in the positive half space of this line (i.e., $\mathbf{w}^T \mathbf{x} \geq 0$)
- What will be the angle between any such vector and \mathbf{w} ? Obviously, less than 90°
- What about points (vectors) which lie in the negative half space of this line (i.e., $\mathbf{w}^T \mathbf{x} < 0$)
- What will be the angle between any such vector and \mathbf{w} ? Obviously, greater than 90°
- Of course, this also follows from the formula ($\cos \alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$)
- Keeping this picture in mind let us revisit the algorithm

37/00

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

So, this is what the geometric interpretation looks like. This is our decision boundary $w^T x$ and the vector w is actually orthogonal to this line and that is exactly the intuition that we have built so far.

Now, let us consider some points which are supposed to lie in the positive half space of this line, right. That means these are the points for which the output is actually 1, ok. Now, can you tell me what is the angle between any of these points and w or you guys are actually trying to tell me the angle we have got some measuring stuff, no. So, I will give you three options i.e. equal to 90, greater than 90 and less than 90.

Student: Less than 90.

Less than 90 it is obvious from the figure, right. Now, if I take any point which lies in the negative half space, what is the angle going to be between them? It is greater than 90, right. Again obvious and it also follows from the fact that $\cos \alpha$ is $w^T x$ by something and we know that for the positive points $w^T x$ is greater than equal to 0, right. That means, $\cos \alpha$ would be greater than equal to 0, that means the angle

alpha would be less than 90 degrees and for the negative points w transpose x is actually less than 0. That means, cos alpha would be less than 0 that means alpha would be greater than 90 degrees, right.

So, it actually follows from the formula itself, but it is also clear from the figure. So, keeping this picture in mind let us revisit the algorithm, ok. So, this is the algorithm, ok.

(Refer Slide Time: 07:32)

Algorithm: Perceptron Learning Algorithm

```

P ← inputs with label 1;
N ← inputs with label 0;
Initialize w randomly;
while !convergence do
    Pick random x ∈ P ∪ N;
    if x ∈ P and w · x < 0 then
        | w = w + x;
    end
    if x ∈ N and w · x ≥ 0 then
        | w = w - x;
    end
end
//the algorithm converges when all the
inputs are classified correctly

```

$$\cos \alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$$

- For $\mathbf{x} \in P$ if $\mathbf{w} \cdot \mathbf{x} < 0$ then it means that the angle (α) between this \mathbf{x} and the current \mathbf{w} is greater than 90° (but we want α to be less than 90°)
- What happens to the new angle (α_{new}) when $\mathbf{w}_{new} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned} \cos(\alpha_{new}) &\propto \mathbf{w}_{new}^T \mathbf{x} \\ &\propto (\mathbf{w} + \mathbf{x})^T \mathbf{x} \\ &\propto \mathbf{w}^T \mathbf{x} + \mathbf{x}^T \mathbf{x} \\ &\propto \cos \alpha + \mathbf{x}^T \mathbf{x} \end{aligned}$$

$$\cos(\alpha_{new}) > \cos \alpha$$
- Thus α_{new} will be less than α and this is exactly what we want

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

Now, let us look at the first condition which was this. Now, if x belongs to p and w transpose x is less than 0, then means that the angle between x and the current w is actually greater than 90 degrees, but what do we want it to be? Less than 90 degrees and our solution to do this is, but we still do not know why this works? Now, anyone knows why this works? So, let us see why this works.

So, what is the new cos alpha going to be? It is going to be proportional to this, right. It is going to be proportional to this. I will just substitute what w nu is, fine. That means, if cos alpha nu is going to be greater than cos alpha, what is alpha nu going to be? It will be less than and that is exactly what we wanted. This angle was actually greater than 90 degrees. So, you want to slowly move it such that it becomes less than 90 degrees. It is not going to get solved in one iteration and that is why till convergence.

So, we will keep doing this. I will keep picking xs again and again till it reaches convergence. That means, till we are satisfied with that condition, right.

(Refer Slide Time: 08:47)

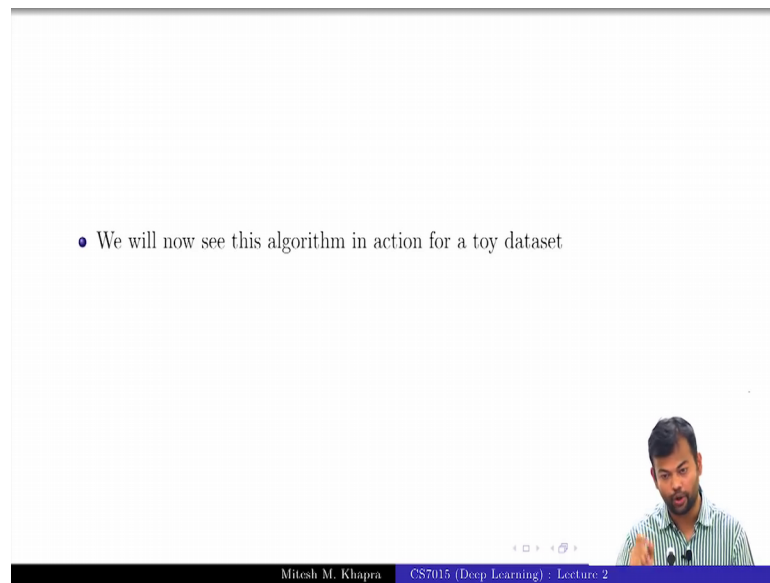
<p>Algorithm: Perceptron Learning Algorithm</p> <p>$P \leftarrow$ inputs with label 1; $N \leftarrow$ inputs with label 0; Initialize \mathbf{w} randomly; while !convergence do Pick random $\mathbf{x} \in P \cup N$; if $\mathbf{x} \in P$ and $\mathbf{w} \cdot \mathbf{x} < 0$ then $\mathbf{w} = \mathbf{w} + \mathbf{x}$; end if $\mathbf{x} \in N$ and $\mathbf{w} \cdot \mathbf{x} \geq 0$ then $\mathbf{w} = \mathbf{w} - \mathbf{x}$; end end //the algorithm converges when all the inputs are classified correctly</p> <hr/> $\cos \alpha = \frac{\mathbf{w}^T \mathbf{x}}{\ \mathbf{w}\ \ \mathbf{x}\ }$	<ul style="list-style-type: none"> • For $\mathbf{x} \in N$ if $\mathbf{w} \cdot \mathbf{x} \geq 0$ then it means that the angle (α) between this \mathbf{x} and the current \mathbf{w} is less than 90° (but we want α to be greater than 90°) • What happens to the new angle (α_{new}) when $\mathbf{w}_{new} = \mathbf{w} - \mathbf{x}$ $\begin{aligned} \cos(\alpha_{new}) &\propto \mathbf{w}_{new}^T \mathbf{x} \\ &\propto (\mathbf{w} - \mathbf{x})^T \mathbf{x} \\ &\propto \mathbf{w}^T \mathbf{x} - \mathbf{x}^T \mathbf{x} \\ &\propto \cos \alpha - \mathbf{x}^T \mathbf{x} \\ \cos(\alpha_{new}) &< \cos \alpha \end{aligned}$ <ul style="list-style-type: none"> • Thus α_{new} will be greater than α and this is exactly what we want
--	---

39/69

Let us look at the other condition \mathbf{x} belongs to n and $\mathbf{w}^T \mathbf{x}$ was greater than equal to 0, then it means that the angle α is actually less than 90 degrees and we want it to be the opposite, ok. I will just quickly skim over this \mathbf{w} minus this \mathbf{x} , right. Ok I forgot to mention that this is actually a positive quantity, right. I mean that is why that result holds, ok. That means, $\cos \alpha_{new}$ is going to be less than $\cos \alpha$ and this slight bit of mathematical in correctness, I am doing here, but that does not affect the final result.

So, I will just gloss over that and you can go home and figure it out, but still it does not take away from the final intuition and interpretation, right. So, now the new $\cos \alpha$ is going to be less than the original $\cos \alpha$; that means, the angle is going to be greater and that exactly what we wanted, ok.

(Refer Slide Time: 09:40)

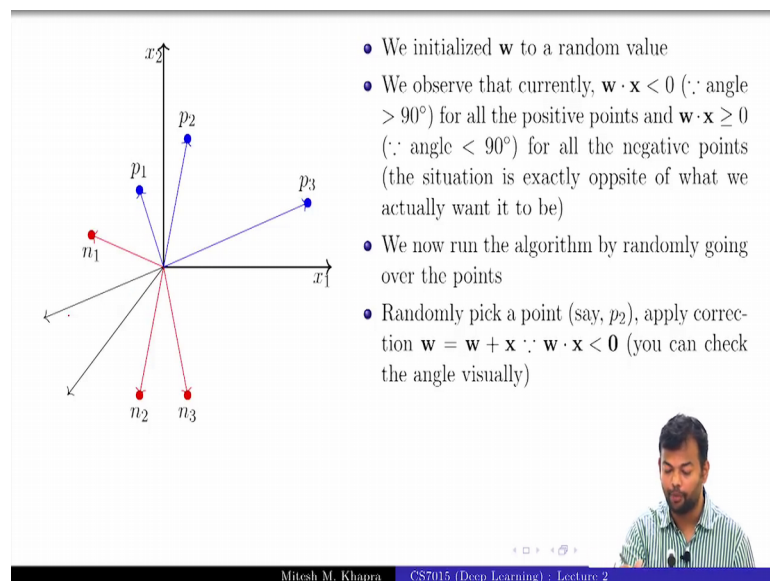


- We will now see this algorithm in action for a toy dataset

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

So, we will now see this algorithm in action for a toy data set.

(Refer Slide Time: 09:44)



- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, p_2), apply correction $w = w + x$ $\because w \cdot x < 0$ (you can check the angle visually)

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

So, this is the toy data set we have and we have initialized w to a random value and that turns out to be this, right. I just picked up some random value for w and ended up with this particular configuration for w .

Now, we observe that currently w transpose x is less than 0 for all the positive points and it is actually greater than equal to 0 for all the negative points. If you do not understand w transpose x , it is just that the all the positive angle points actually have a greater than

90 degree angle and all the negative points actually have a less than 90 degree angle. So, this is exactly opposite of the situation that we want and now from here on, we want to actually run the perceptron algorithm, right and try to fix this w . How does it work? Remember we randomly pick a point, ok. So, say we pick the point p_1 , do we need to apply a correction?

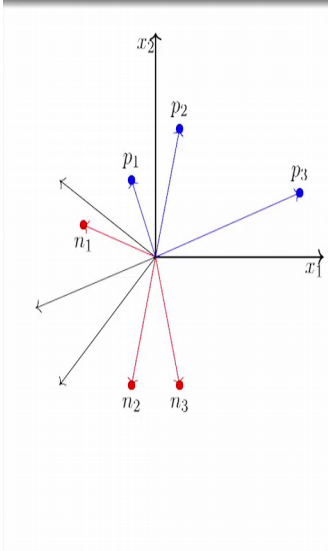
Student: Yes.

Yes. Why, because it is a positive point and the condition is violated, right. So, now we add w equal to w plus x and we get this new w . So, notice that we have a new w , ok. We again repeat this, we again pick a new point and this time we have picked p_2 . Do we need a correction?

Student: Yes.

At least from the figure it looks like the angle is greater than 90, right. So, we will again do a correction. We will add w is equal to w plus p , right. This x is actually, sorry p_2 and this is where we end up, ok.

(Refer Slide Time: 11:12)

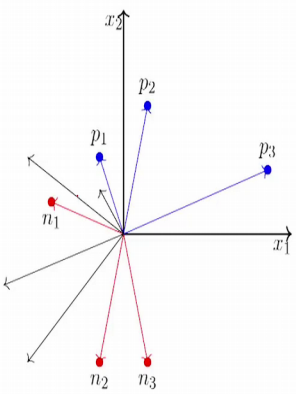


- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, n_1), apply correction $w = w - x \because w \cdot x \geq 0$ (you can check the angle visually)

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

Now, again we pick a point randomly n_1 . Do we need a correction? So, this is what our w is. This line here and n_1 , right. So, we need a correction, ok. Now, what is the correction going to be? It will be minus and then, the w changes, ok.

(Refer Slide Time: 11:32)

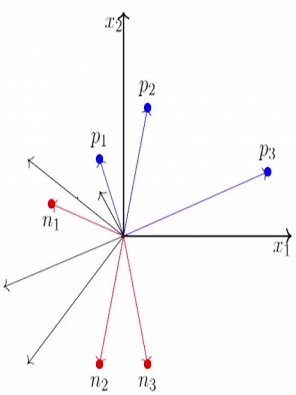


- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, n_3), no correction needed $\because w \cdot x < 0$ (you can check the angle visually)

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

Now, we pick another point n_3 . Do we need a correction? No at least on the figure it seems like the angle is greater than 90° and we continue this, right.

(Refer Slide Time: 11:39)

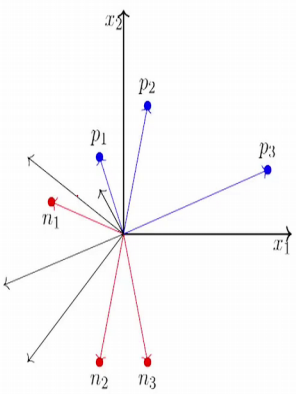


- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, n_2), no correction needed $\because w \cdot x < 0$ (you can check the angle visually)

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

For n_2 , we do not need a correction. Now, for p_3 again we do not need a correction.

(Refer Slide Time: 11:43)

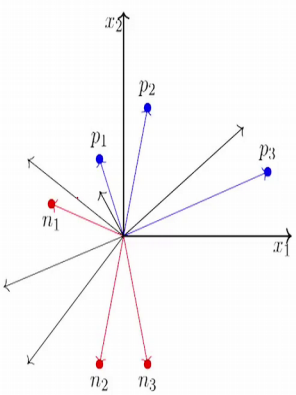


- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, p_3), apply correction $w = w + x \because w \cdot x < 0$ (you can check the angle visually)

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

The angle looks less than 90. Sorry actually it is we need a correction. The angle is slightly greater than 90 and this is our correction, and now we keep cycling.

(Refer Slide Time: 11:52)

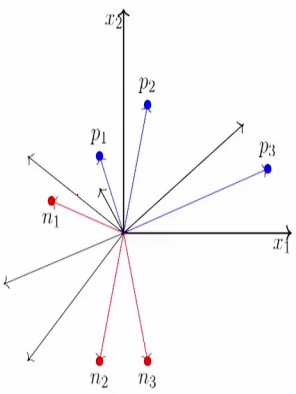


- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly opposite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, p_1), no correction needed $\because w \cdot x \geq 0$ (you can check the angle visually)

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

Now, as I keep cycling over the points, I realize that I no longer need any correction.

(Refer Slide Time: 11:54)



- We initialized w to a random value
- We observe that currently, $w \cdot x < 0$ (\because angle $> 90^\circ$) for all the positive points and $w \cdot x \geq 0$ (\because angle $< 90^\circ$) for all the negative points (the situation is exactly oppsite of what we actually want it to be)
- We now run the algorithm by randomly going over the points
- Randomly pick a point (say, n_1), no correction needed $\because w \cdot x < 0$ (you can check the angle visually)

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

It should be obvious from the figure that for this particular value of w , now all my positive points are making an angle less than 90° and all my negative points are actually making an angle greater than 90° . That means, by definition now my algorithm has converged, right. So, I can just stop it. So, I can just make one pass over the data. If nothing changes, I will just say it has converged, ok. Now, does anyone see a problem with this?

Student: Never converge.

It will never converge in some cases. So, can someone tell me why? We are considering only cases where the data is linearly separable, right. That we already assumed. So, what you are trying to tell me is that you are going over these points cyclically. So, let me just rephrase and put words in your mouth that what you are trying to tell me actually is that I take a point, I adjust w , but now for the next point I maybe go back to the same w because that point asked me to move it again and I keep doing this again and again and basically, end up nowhere, right. That is why this will never converge. That is exactly what you are trying to tell me, right.

Now, that is exactly what I am forcing you to tell me. So, that is not the case, right. This algorithm will converge.