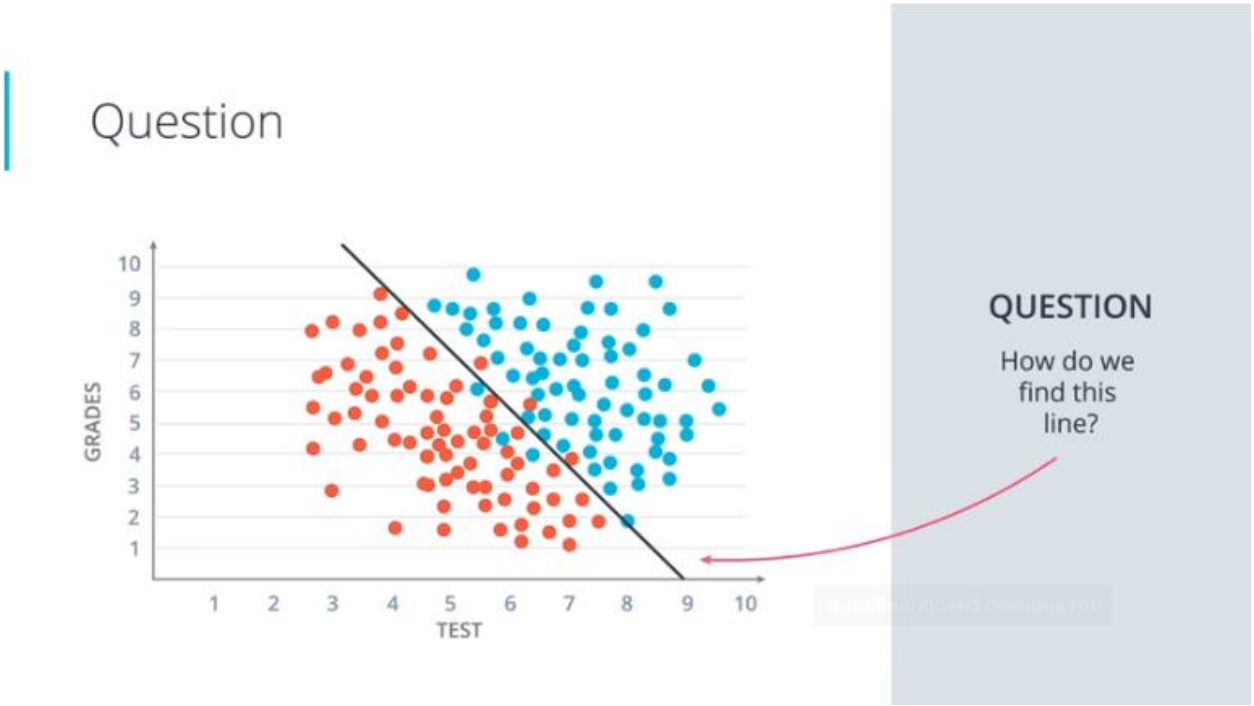
## Perceptron Trick

In the last section you used your logic and your mathematical knowledge to create perceptrons for some of the most common logical operators. In real life, though, we can't be building these perceptrons ourselves. The idea is that we give them the result, and they build themselves. For this, here's a pretty neat trick that will help us.

So we had a question we're trying to answer and the question is, how do we find this line that separates

the blue points from the red points in the best possible way?

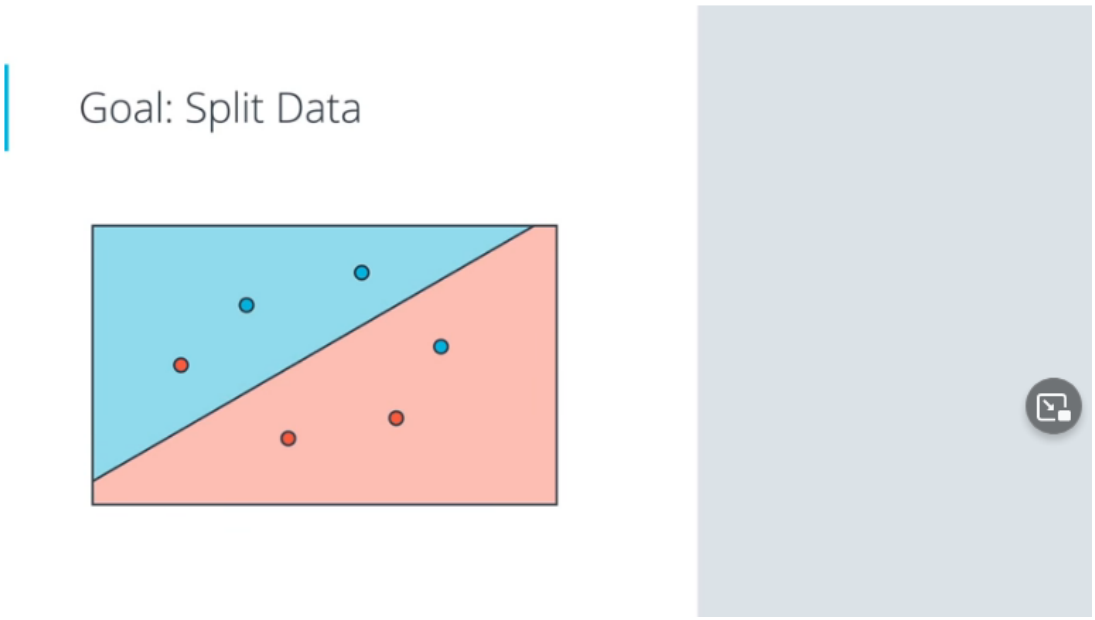



Let's answer this question by first looking at a small example with three blue points and three red points. And we're going to describe an algorithm that will find

the line that splits these points properly. So the computer doesn't know where to start.

It might as well start at a random place by picking a random linear equation.

This equation will define a line and a positive and negative area given in blue and red respectively.




What we're going to do is to look at how badly this line is doing and then move it around to try to get better and better.

Now the question is, how do we find how badly this line is doing?

So let's ask all the points. Here we have four points that are correctly classified.

They are these two blue points in the blue area and these two red points in the red area. And these points are correctly classified, so they say, "I'm good." And then we have these two points that are incorrectly classified.

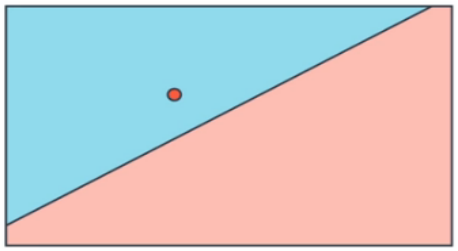
That's this red point in the blue area and this blue point in the red area.

We want to get as much information from them so we want them to tell us something so that we can improve this line. So what is it that they can tell us?

So here we have a misclassified point, this red point in the blue area.

Now think about this. If you were this point, what would you tell the line to do?

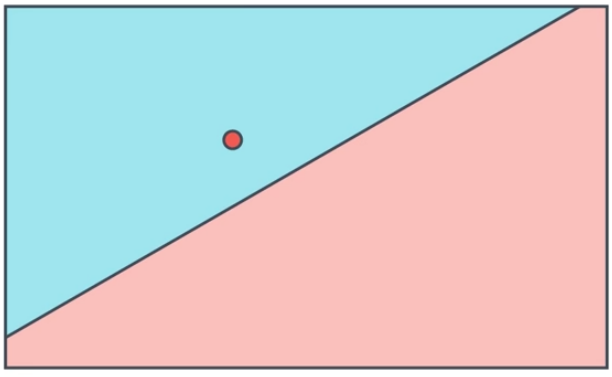
Goal: Split Data



Would you like it to come closer to you or farther from you?

That's our quiz.

Will the misclassified point want the line to come closer to it or farther from it?



QUIZ

Does the misclassified point want the line to be close or farther?

- Closer
- Farther

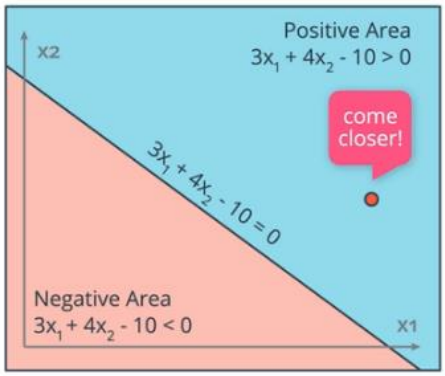
**Question 1 of 2**

Does the misclassified point want the line to be closer or farther?

- Closer

## Time for some math!

Now that we've learned that the points that are misclassified, want the line to move closer to them, let's do some math. The following video shows a mathematical trick that modifies the equation of the line, so that it comes closer to a particular point.




**LINE:** $3x_1 + 4x_2 - 10 = 0$
**POINT:** (4,5)
**LEARNING RATE:** 0.1

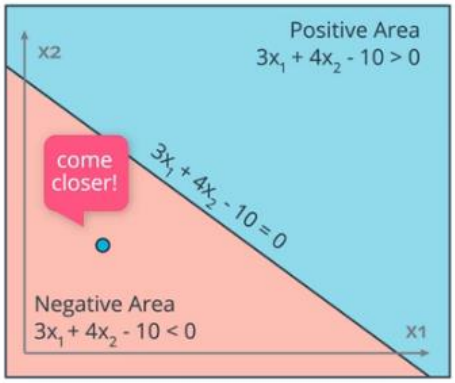
| | | | |
|---|---|---|---|
| − | 3 | 4 | -10 |
| | 0.4 | 0.5 | 0.1 |
| | 2.6 | 3.5 | -10.1 |

**NEW LINE**
$2.6x_1 + 3.5x_2 - 10.1 = 0$

To make the line come closer to red point (place red point to negative region) we will use the point coordinates, bias (1) and the learning rate to make some movement to the line, like follows

$P(4,5) \times 0.1 \rightarrow 0.4,\ 0.5$ , we will take the modified coordinate and subtract them from the line equation's coefficient, as in the image above.

In the same way if we have a blue point in the red area (negative area) $P(1,1)$ , we will do the same but this time we'll add the modified coordinates by learning rate to the line's parameters as in the image below




**LINE:** $3x_1 + 4x_2 - 10 = 0$
**POINT:** (1,1)
**LEARNING RATE:** 0.1

| | | | |
|---|---|---|---|
| + | 3 | 4 | -10 |
| | 0.1 | 0.1 | 0.1 |
| | 3.1 | 4.1 | -9.9 |

**NEW LINE**
$3.1x_1 + 4.1x_2 - 9.9 = 0$

For the second example, where the line is described by $3x_1 + 4x_2 - 10 = 0$, if the learning rate was set to 0.1, how many times would you have to apply the perceptron trick to move the line to a position where the blue point, at (1, 1), is correctly classified?

Answer: the smallest number of times to move the line is: 10