Training a Perceptron in Machine Learning

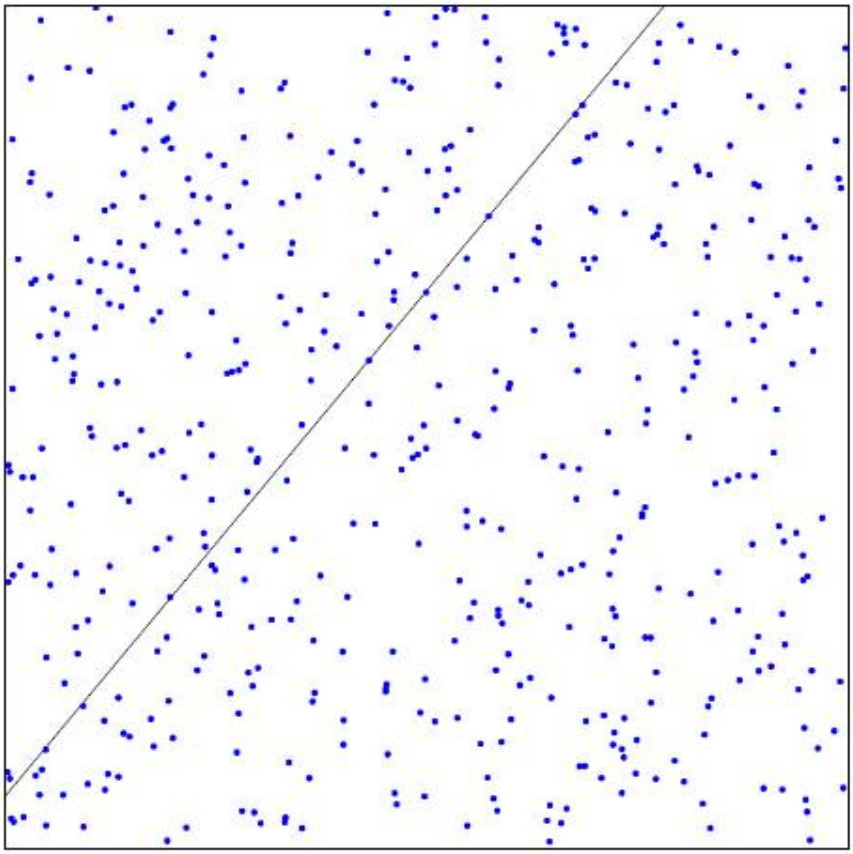
Create a Perceptron object, create a training function, and train the Perceptron based on the correct answers.

1. Create a Perceptron object.
2. Create training functions
3. Perceptron training based on correct answers

Training mission

Imagine a straight line in space with scattered x and y points.

Train a perceptron to classify points that lie on and below that line.



Create a Perceptron object.

Create a Perceptron object. Give it any name you like (for example, Perceptron).

Allow the perceptron to accept two parameters:

1. Number of inputs (no)
2. Learning rate.

Set the default learning speed to 0.00001.

Next, generate random weights from -1 to 1 for each input.

For example:

```
// Perceptron Object function Perceptron(no, learningRate = 0.00001) { // Set In
```

Random weights

The perceptron will start with a random weight for each input.

Learning speed

During Perceptron training, each error will cause the weights to be adjusted by a small amount.

This small part is called the "Perceptron learning rate".

In the Perceptron object, we call it learnc.

Bias

Sometimes, if both inputs are zero, the Perceptron may produce an incorrect output.

To avoid this, we provide the Perceptron with an additional input of 1.

This is called bias.

Add a trigger function

Remember the perceptron algorithm:

1. Multiply each input by the perceptron's weight.
2. Add the results
3. Calculate the result

For example:

```
this.activate = function(inputs) { let sum = 0; for (let i = 0; i  inputs.length
```

The trigger function will output:

1. 1 if the sum is greater than 0
2. 0 if the sum is less than 0

Create a training function

The training function predicts the outcome based on the activation function.

Whenever the prediction is wrong, the perceptron will adjust the weights.

After several predictions and adjustments, the weights will be accurate.

For example:

```
this.train = function(inputs, desired) { inputs.push(this.bias); let guess = this
```

Backpropagation

After each prediction, the perceptron calculates the error of that prediction.

If the prediction is wrong, the perceptron will adjust the bias and weights so that the next prediction will be slightly more accurate.

This type of learning is called backpropagation.

After trying (several thousand times), your perceptron will become quite good at guessing.

Create your own library

Library code:

```
// Perceptron Object function Perceptron(no, learningRate = 0.00001) { // Set Ini
```

Now you can integrate the library into your HTML:

Use the library

For example:

```
// Initiate Values const numPoints = 500; const learningRate = 0.00001; // Creat
```