# Neural networks: Learning Perceptron

Alexandre Bergel
http://bergel.eu
07/09/2020

# Computational Universality

So far, we have seen that perceptrons can be used to model any computation

Good news: a network of perceptrons is as *expressive as any computational model*

# Learning Algorithm

In our previous examples, we have *provided values of weights and bias*

*A learning algorithm can automatically tune these values*

This tuning happens in response to *external stimuli*

Making perceptrons / neurons learn is the real difference from logical gates

# Learning strategies

## Supervised learning

You need examples from which patterns are extracted from

E.g., Classical techniques in deep learning, natural language processing, image processing

## Unsupervised learning

Patterns are discovered without example

# Supervised learning

Involve a teacher that is smarter than the perceptron / neuron / network itself

Facial recognition example:

The teacher shows the network many person faces with the name of that person

Ask the network to make a guess of a person name by only giving the face

The teacher compares the guess with the correct name, and make adjustments according to the error

# Unsupervised Learning

When there is not data examples with *known answers*

Searching for a hidden pattern in a data set

Examples:

Clustering: dividing a set of elements into groups according to some unknown pattern

Self-driving cars, Video games, Robotic

We will focus on unsupervising learning when we will talk about neuroevolution (combination of neural network and genetic algorithm)

# Perceptron learning algorithm (1/2)

During the training period, a series of inputs are presented to the perceptron

Each input follows x = (x1, x2, …, xN)

For each input set, there is a *desired output (0 or 1)*

The actual output is determined by w*x + b

If the actual output is wrong, then some learning has to happen

# Perceptron learning algorithm (2/2)

```
diff = desiredOutput - realOutput

lr = 0.1

For all N:

    weightN = weightN + (lr * inputN * diff)

bias = bias + (lr * diff)

lr is called the learning rate

Weights and bias is randomly initialized
(within a range of -2.0 to +2.0)
```

# Perceptron in action

We stated that neural networks are often used for pattern recognition, including facial recognition

Even simple perceptrons can demonstrate the basics of *classification*

Consider a line in a two-dimensional space

Points in that space can be classified as living on either one side of the line or the other

# Perceptron in action

Our perception does not know about the dividing line, however, it has to guess on the side of the line live a given point

This example is very simplistic, however, it *shows how a perceptron can be trained* to recognize points on one side versus another

# Perceptron in action

Nothing prevent a perceptron from taking as input any arbitrary numerical values

In this example, we will consider inputs as a set of arbitrary numbers, and not only 0 and 1

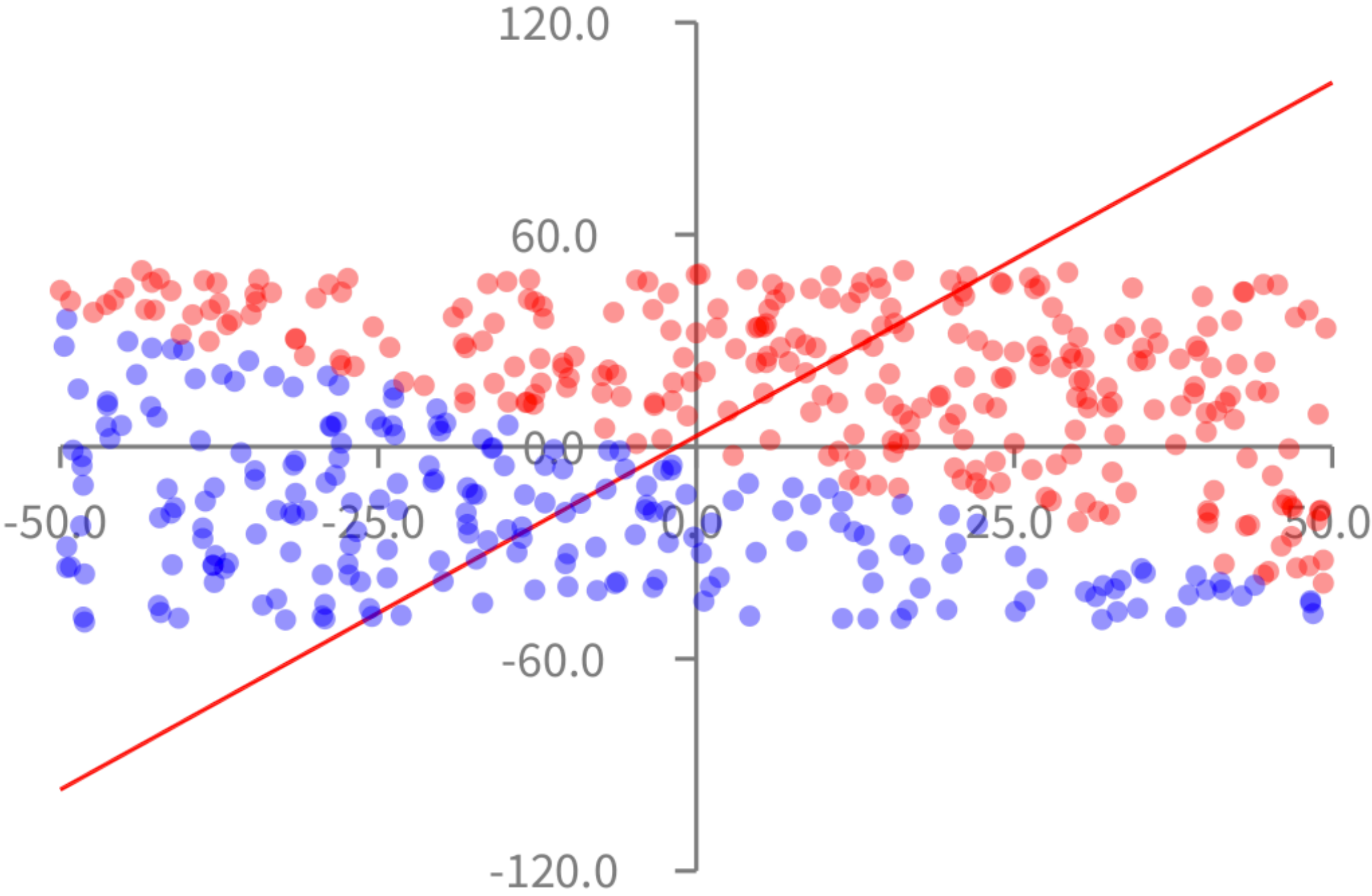
Let's take a perceptron, with two inputs

- x and y coordinate of a point

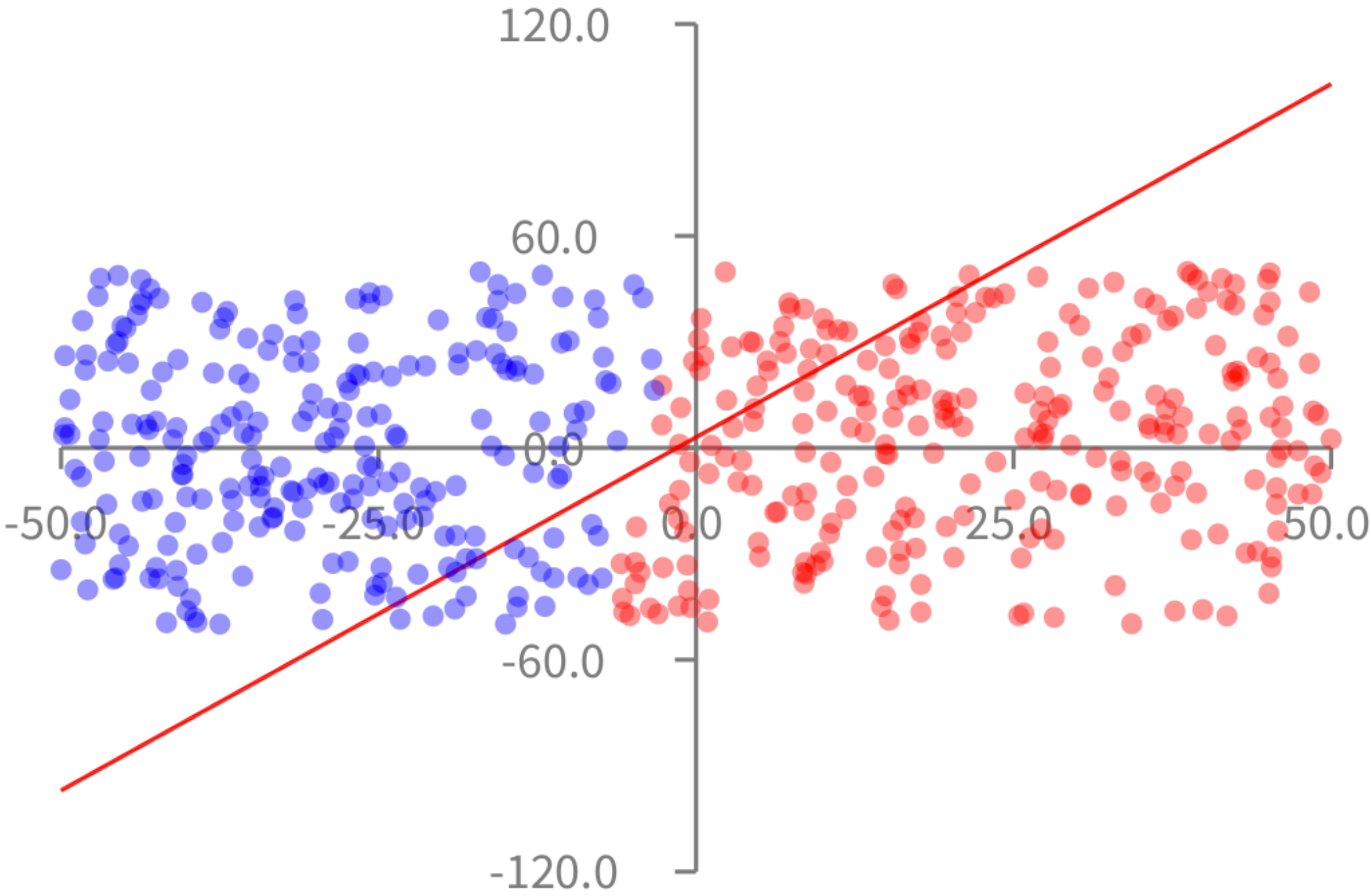
The output will be either 0 or 1

- 0 indicating the point is below the line
- 1 indicates the point is above the line

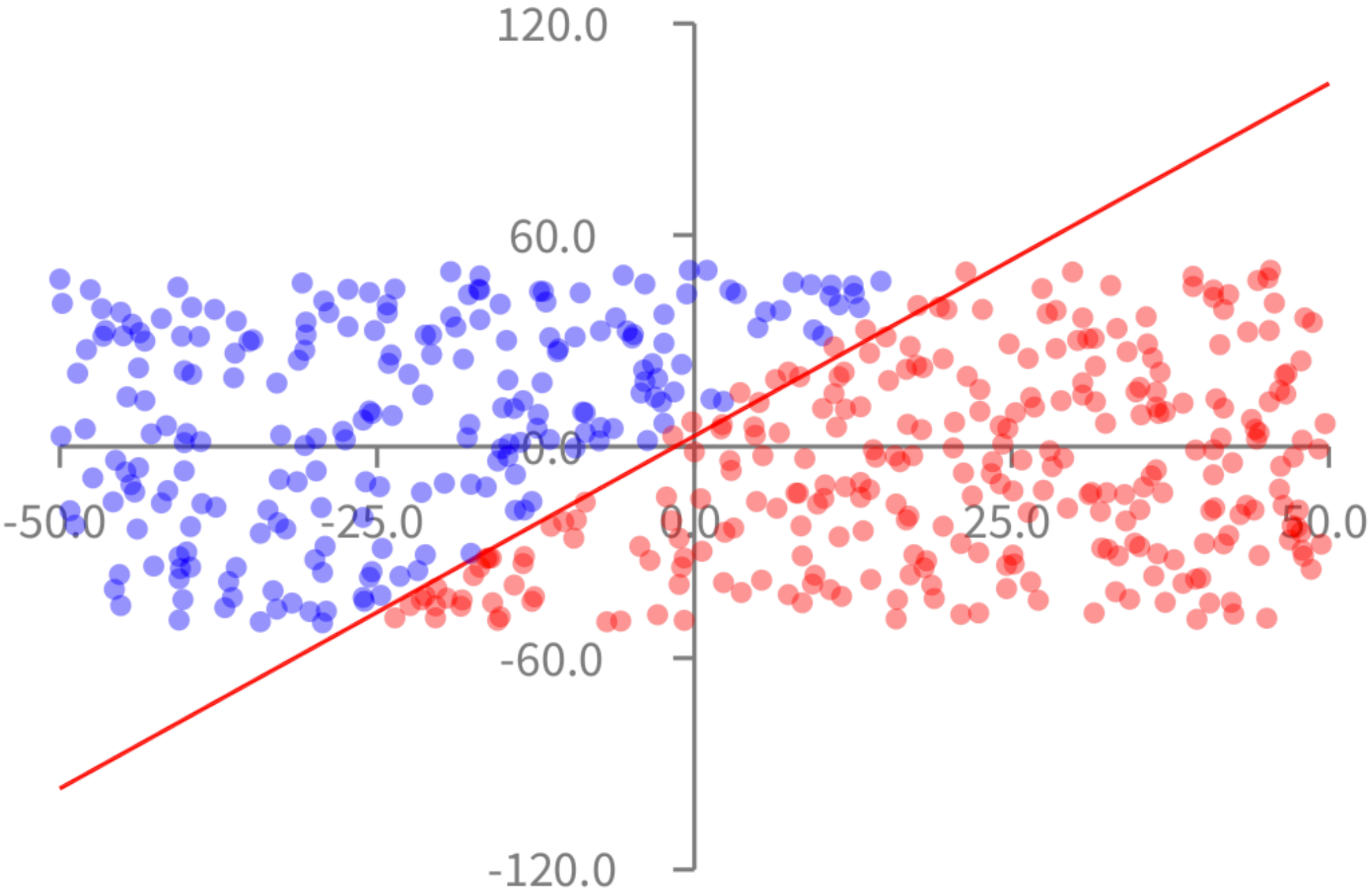
# After 2 trainings



# After 20 trainings



# After 50 trainings



# Exercise

Implement a learning perceptron with the space position problem we have just sen

Try learning rate different values

Show the curve precision vs number of training