1

00:00:00,540 --> 00:00:04,650

[Auto-generated transcript. Edits may have been applied for clarity.]

Neural networks begin with the basic unit known as a neuron or unit.

2

00:00:05,100 --> 00:00:10,200

This concept is inspired by biological neurons in the human brain, where neurons,

3

00:00:10,230 --> 00:00:16,170

or in this case units, are interconnected and can be either active or inactive.

4

00:00:16,560 --> 00:00:22,080

Let's model this behavior mathematically to understand how an artificial neuron works.

5

00:00:22,710 --> 00:00:31,110

Each neuron performs a mathematical operation. Graphically, this can be depicted as a diagram showing the operation inside the computer.

6

00:00:31,380 --> 00:00:38,550

The neuron holds or outputs a value called the activation, which indicates whether the unit is active or inactive.

7

00:00:38,940 --> 00:00:44,370

Here's how it works. The neuron receives inputs, for example x1 and x2.

8

00:00:44,790 --> 00:00:49,410

Each input is multiplied by a corresponding weight w1 and w2.

9

00:00:49,740 --> 00:00:52,680

A bias term w0 is added to the sum.

10

00:00:53,220 --> 00:01:03,330

The result of this operation, x1 w1 plus x2 w2 plus w0 is passed through a function called the activation function or decision function.

11

00:01:03,720 --> 00:01:12,060

A common activation function is the step function, which outputs zero for values below zero, and one for values at or above zero.

12

00:01:12,420 --> 00:01:18,900

This is the basic building block of a neural network. We're still talking about a single neuron, not yet a full network.

13

00:01:19,110 --> 00:01:23,250

Let's illustrate how this simple neuron can learn and make predictions.

14

00:01:23,670 --> 00:01:28,740

Consider a data set where you're allowed to play if either mom or dad authorizes you.

15

00:01:29,160 --> 00:01:38,970

If both say no, you can't play. The data set has two input columns one for mom's authorization, one for yes, zero for no, and one for dads.

16

00:01:39,480 --> 00:01:43,050

The third column is the target output whether you can play.

17

00:01:43,410 --> 00:01:48,150

This is known as the or function to train a neuron to learn this rule.

18

00:01:48,240 --> 00:01:55,860

Let's assign weights w1 equals one, w2 equals one and w zero equals negative one.

19

00:01:56,430 --> 00:02:00,540

Now let's test the input where both parents say no zero zero.

20

00:02:01,080 --> 00:02:06,060

The calculation is zero times one plus zero times one equals negative one.

21

00:02:06,330 --> 00:02:12,090

The activation function g negative one equals zero which correctly predicts no play.

22

00:02:12,810 --> 00:02:18,150

Now try 010 times one plus one times one equals zero.

23

00:02:18,330 --> 00:02:22,050

G zero equals one. Meaning you can play.

24

00:02:22,470 --> 00:02:28,140

Try 111 times one plus one times one minus one equals one.

25

00:02:28,370 --> 00:02:31,710

G1 equals one. Again predicting correctly.

26

00:02:32,070 --> 00:02:36,120

During training the network learns these weights automatically from the data.

27

00:02:36,690 --> 00:02:40,020

Now let's try a different function. The end function.

28

00:02:40,320 --> 00:02:44,010

You can only play if mom authorizes and it's not raining.

29

00:02:44,280 --> 00:02:48,690

Again two inputs mom's authorization and whether it's raining.

30

00:02:49,050 --> 00:02:57,030

The correct weights for this are w1 equals one, w2 equals one and w zero equals negative two.

31

00:02:57,150 --> 00:03:02,460

Try zero zero plus zero zero minus two equals negative two.

32

00:03:02,730 --> 00:03:08,820

G equals zero. Try one one plus one plus one equals zero.

33

00:03:09,120 --> 00:03:13,530

G equals one. This neuron correctly models the and function.

34

00:03:14,160 --> 00:03:20,430

Each weight contributes to the neurons activation. If the output is zero the neuron is inactive.

35

00:03:20,640 --> 00:03:25,560

If it's one, it's active. The bias term adjusts the activation threshold.

36

00:03:25,890 --> 00:03:34,890

This basic model is called a perceptron. It generalizes to any number of inputs using a weighted sum followed by an activation function.

37

00:03:35,370 --> 00:03:41,310

For example, to predict a person's height, you could input arm length, biological sex, and athleticism.

38

00:03:41,550 --> 00:03:45,870

The perceptron weights these inputs and adjusts the activation using the bias.

39

00:03:46,260 --> 00:03:49,320

While the step function is the simplest activation function.

40

00:03:49,560 --> 00:03:53,700

In practice we use differentiable functions like the sigmoid function.

41

00:03:54,090 --> 00:03:57,270

These are smoother and allow for efficient optimization.

42

00:03:57,660 --> 00:04:01,110

They also provide a probabilistic interpretation of the output.

43

00:04:01,650 --> 00:04:07,560

Neurons are organized into layers. Each neuron in one layer connects to all neurons in the next.

44

00:04:07,860 --> 00:04:11,520

This structure is called a multilayer perceptron MLP.

45

00:04:11,880 --> 00:04:15,660

The weights and biases determine how each neuron is activated.

46

00:04:15,930 --> 00:04:21,450

An MLP can learn more complex nonlinear decision boundaries than a single neuron.

47

00:04:21,840 --> 00:04:26,910

Deep learning models can have thousands, millions, or even billions of parameters.

48

00:04:27,150 --> 00:04:32,640

Neural networks can also learn from images, for example in handwritten digit classification.

49

00:04:32,790 --> 00:04:39,030

0 to 9 traditional models like logistic regression require carefully selected features.

50

00:04:39,570 --> 00:04:44,070

If someone writes the digit one in a different position, the model might fail.

51

00:04:44,400 --> 00:04:50,370

Neural networks, by contrast, can take raw pixel values and pass them through layers of neurons.

52

00:04:50,910 --> 00:04:55,110

The final layer outputs a prediction without needing manual feature extraction.

53

00:04:55,260 --> 00:04:59,970

This is a major advantage of deep learning. It reduces or eliminates.

54

00:05:00,010 --> 00:05:03,880

The need for manual feature engineering. In earlier models,

55

00:05:04,060 --> 00:05:13,480

experts manually extracted features from x rays such as angles and gradients to train models like logistic regression with deep learning.

56

00:05:13,570 --> 00:05:19,270

Raw pixels can be fed directly into the network which learns relevant features during training.

57

00:05:19,630 --> 00:05:24,610

So what is deep learning? It refers to large neural networks with many hidden layers.

58

00:05:25,030 --> 00:05:33,730

These networks have many parameters and biases. A deep neural network is essentially a multilayer perceptron with several hidden layers.

59

00:05:34,000 --> 00:05:37,780

Neural networks can also perform multi-class classification.

60

00:05:38,050 --> 00:05:44,860

For example, in digit classification, the output layer has one neuron per class 0 to 9.

61

00:05:45,220 --> 00:05:49,120

The neuron with the highest activation determines the predicted label.

62

00:05:49,570 --> 00:05:54,700

If the outputs are 0.1, 0.3 and 0.79.

63

00:05:54,820 --> 00:05:58,630

The prediction is the class corresponding to 0.79.

64

00:05:59,140 --> 00:06:03,490

MLPs can also perform regression using a single output neuron.

65

00:06:03,850 --> 00:06:10,750

For example, given inputs like arm length, biological sex, and athleticism, the network can predict height.

66

00:06:11,200 --> 00:06:17,860

The output activation function in this case should be linear, allowing the neuron to output any continuous value.

67

00:06:18,490 --> 00:06:24,729

To understand neural networks heuristically, consider an example from Geoffrey Hinton, a pioneer in.

68

00:06:24,730 --> 00:06:29,470

I suppose you want to classify whether an image contains a bird.

69

00:06:29,590 --> 00:06:38,320

A 100 by 100 pixel image has 10,000 pixels, and with RGB channels, that's 30,000 input values.

70

00:06:38,890 --> 00:06:44,830

You need to map these to a single output. Hinton described wiring the network by hand.

71

00:06:45,400 --> 00:06:47,950

The first layer detects features like edges.

72

00:06:48,310 --> 00:06:56,650

For example, a horizontal edge detector might have strong positive weights from one row with pixels and strong negative weights from the row below.

73

00:06:57,160 --> 00:07:02,950

If one row is bright and the other is dark, the neuron activates, detecting an edge.

74

00:07:03,340 --> 00:07:10,330

You can have many such detectors across the image. The next layer might detect combinations of edges like a beak.

75

00:07:10,870 --> 00:07:18,150

Further layers detect more complex features eyes, wings, feet until the final layer predicts bird.

76

00:07:18,160 --> 00:07:23,080

Instead of wiring everything by hand, we initialize the network with random weights.

77

00:07:23,470 --> 00:07:27,010

When we input a bird image, the prediction might be wrong.

78

00:07:27,220 --> 00:07:35,590

Say 0.3 instead of one. We compute the error and send it backward through the network using calculus to adjust the weights.

79

00:07:36,100 --> 00:07:41,260

Repeating this process with many examples, the network learns to classify accurately.

80

00:07:41,740 --> 00:07:45,250

Next, we'll talk about how to tweak the weights in more detail.