1

00:00:00,540 --> 00:00:04,980

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

The learning algorithm used to train deep learning models is called gradient descent.

2

00:00:05,160 --> 00:00:10,620

So let's talk about gradient descent and its implementation in neural networks known as backpropagation.

3

00:00:11,100 --> 00:00:16,830

Gradient descent is an optimization algorithm that iteratively adjusts the model's parameters,

4

00:00:17,040 --> 00:00:24,000

its weights and biases to minimize the error between the predicted and actual outputs from the training data set.

5

00:00:24,660 --> 00:00:27,770

Suppose you're training a neural network to recognize a bird.

6

00:00:27,780 --> 00:00:38,010

Initially, the weights are random if the network should output one for a bird, but instead outputs 0.3, the error is one -0.3 equals 0.7.

7

00:00:38,460 --> 00:00:44,700

We want to reduce the mean squared error. So we square that difference one -0.3 squared.

8

00:00:45,180 --> 00:00:51,060

The algorithm then adjusts the weights slightly. After the adjustment the error becomes smaller.

9

00:00:51,420 --> 00:00:58,920

If we show another sample of a bird, the error might go from one -0.3 squared to one -0.6 squared.

10

00:00:59,370 --> 00:01:02,700

This process is repeated gradually reducing the error.

11

00:01:03,420 --> 00:01:08,850

Gradient descent can be visualized as a hiker descending a valley to find the lowest point,

12

00:01:09,240 --> 00:01:13,560

or a ball rolling downhill to reach the minimum of the loss function.

13

00:01:14,160 --> 00:01:18,960

Each step of the training algorithm moves in the direction of the steepest descent.

14

00:01:19,470 --> 00:01:21,810

That's why it's called gradient descent.

15

00:01:22,080 --> 00:01:29,490

The gradient is a mathematical entity that shows how to tweak the weights to minimize the loss in the most optimized way.

16

00:01:29,850 --> 00:01:32,450

Each training step is called an epoch.

17

00:01:32,460 --> 00:01:40,110

An epoch means starting the network with random values, showing the examples, tweaking the weights, and repeating the process.

18

00:01:40,620 --> 00:01:43,650

After each epoch, the error should decrease.

19

00:01:43,950 --> 00:01:51,390

However, neural networks can get stuck in what's called a local minimum and fail to reach the optimal value.

20

00:01:51,840 --> 00:01:58,410

To mitigate this, training is often repeated multiple times with different random initializations.

21

00:01:58,770 --> 00:02:03,690

Starting from different points increases the chances of reaching the global minimum,

22

00:02:04,110 --> 00:02:08,130

resulting in an optimal model with the best parameters for prediction.

23

00:02:08,730 --> 00:02:14,040

The specific implementation of gradient descent in neural networks is called backpropagation.

24

00:02:14,730 --> 00:02:19,350

Here's how it works. Run the neural network with the current weights and compute the error.

25

00:02:19,770 --> 00:02:24,600

For example, suppose you're trying to classify lung cancer using various parameters.

26

00:02:25,050 --> 00:02:30,840

You run the network, compute the error, and use the gradient to adjust the weights that contributed to the error.

27

00:02:31,500 --> 00:02:37,800

You start with the output layer, adjusting its weights, and then move backward, updating each set of weights.

28

00:02:38,130 --> 00:02:41,490

This step by step reverse update is called backpropagation.

29

00:02:42,180 --> 00:02:51,600

We've discussed multilayer perceptrons MLPs, a type of deep learning model with a feedforward architecture where data flows from input to output.

30

00:02:51,960 --> 00:02:55,110

Each neuron activates the next non in the next layer.

31

00:02:55,440 --> 00:02:59,780

Other important architectures include convolutional neural networks.

32

00:02:59,820 --> 00:03:05,010

CNNs CNNs are widely used for image classification and processing.

33

00:03:05,400 --> 00:03:09,000

They are more effective than traditional MLPs for image tasks.

34

00:03:09,540 --> 00:03:18,540

For example, in bird classification, CNNs use filters or convolutions to detect patterns such as edges, corners, and textures.

35

00:03:18,600 --> 00:03:22,170

CNNs are now the standard for visual recognition tasks.

36

00:03:22,650 --> 00:03:25,110

Recurrent neural networks are RNNs.

37

00:03:25,350 --> 00:03:33,510

RNNs are designed for sequential or time series data, such as stock price forecasting, word prediction, and other trend based tasks.

38

00:03:33,750 --> 00:03:39,690

In RNNs, the state of a neuron at a previous time step influences the current activation.

39

00:03:40,080 --> 00:03:45,180

This creates a form of memory that allows the network to model temporal dependencies.

40

00:03:45,870 --> 00:03:52,860

For example, suppose you want to forecast stock prices over time x1, x2, x3 and so on.

41

00:03:53,250 --> 00:03:59,190

From x1 you predict x2, from x2, you predict x3, and so on.

42

00:04:00,030 --> 00:04:06,000

The architecture of an RNN includes additional connections between neurons across time steps.

43

00:04:06,630 --> 00:04:12,210

When you input x1 to predict x2 and then input x2 to predict x3,

44

00:04:12,450 --> 00:04:17,850

the activation of the neuron from the x1 step still influences the current prediction.

45

00:04:18,300 --> 00:04:24,840

This creates memory in the network. Each prediction uses both the current input and the past internal state.

46

00:04:25,020 --> 00:04:29,130

Making RNNs very suitable for time series forecasting.

47

00:04:29,460 --> 00:04:37,680

Now that we've seen some models, let's continue exploring how deep learning architectures evolve and how they're applied in real world tasks.