UnCodemy

Neural Networks Basics PDF

A neural network is a model made of layers of simple units called neurons. Each neuron multiplies inputs by weights, adds a bias and applies an activation function. The network learns by comparing its output with the right answer using a loss, and adjusting weights with gradient descent through backpropagation. This page explains each step with a small example.

Key takeaways

- A neuron computes a weighted sum of inputs plus a bias, then applies an activation function.
- Layers stack neurons: input, hidden and output layers.
- Training repeats: forward pass, loss, backpropagation and weight update.
- Activation functions add non-linearity so the network can learn complex patterns.
- Overfitting is handled with more data, regularisation, dropout and early stopping.

What is neural networks?

A neural network is a machine learning model made of connected layers of artificial neurons that learns to map inputs to outputs by adjusting numerical weights to reduce error on training data.

What is a neuron?

An artificial neuron takes several numbers as input, multiplies each by a weight, adds them together with a bias and passes the result through an activation function. In symbols: output = activation(w1·x1 + w2·x2 + … + b). The weights decide how much each input matters, and the bias shifts the result. Learning means finding good weights and biases.

How do layers work?

Neurons are arranged in layers. The **input layer** receives the data. **Hidden layers** transform it step by step. The **output layer** gives the prediction: one number for regression, or class scores for classification. A network with several hidden layers is called deep. Each layer's output becomes the next layer's input.

Why do we need activation functions?

Without an activation function, stacked layers collapse into one linear calculation, which cannot learn curved patterns. Common choices:

Function	Behaviour	Typical use
ReLU	0 for negatives, the value itself for positives	Hidden layers
Sigmoid	Squashes values between 0 and 1	Binary output
Tanh	Squashes values between -1 and 1	Some hidden layers
Softmax	Turns scores into probabilities that sum to 1	Multi-class output

How does a network make a prediction?

The data flows forward through the layers, called the **forward pass**. Example: an input of two features goes through a hidden layer of three neurons with ReLU, then an output neuron with sigmoid that gives a probability, such as 0.82 for "will buy". The numbers shown in this example are for illustration only.

What is the loss function?

The loss measures how wrong a prediction is. For regression, mean squared error is common. For classification, cross-entropy is common. A lower loss means better predictions on the training data. Training aims to minimise it.

How does learning happen?

Gradient descent moves each weight a small step in the direction that reduces the loss. The step size is the **learning rate**: too large and training becomes unstable, too small and it is slow. **Backpropagation** is the method that efficiently calculates how much each weight contributed to the error, working backward from the output using the chain rule. One training loop is:

1. Take a batch of examples.
2. Forward pass to get predictions.
3. Compute the loss.
4. Backpropagate to get gradients.
5. Update the weights with an optimiser.
6. Repeat for many batches and epochs (full passes over the data).

What is a simple training walkthrough?

```
data = training examples (inputs X, targets y)
initialise weights with small random numbers
for each epoch:
    for each batch in data:
        predictions = model(batch.X)
        loss = loss_function(predictions, batch.y)
        gradients = backpropagate(loss)
        weights = weights - learning_rate * gradients
    check loss on validation data
stop when validation loss stops improving
```

Libraries such as PyTorch and TensorFlow do the gradient steps for you.

What are batch size, epoch and optimiser?

A **batch** is a small group of examples processed together. An **epoch** is one pass through all training data. An **optimiser** such as SGD or Adam decides how to use the gradients, often with extra tricks that speed up learning.

How do I prevent overfitting?

Use more data, keep the model as small as needed, add **dropout** (randomly switching off neurons during training), use **weight decay** (a penalty on large weights), apply **early stopping** when validation loss rises and augment data where possible. Always keep a separate test set.

When should I use a neural network?

They shine on images, audio, text and other complex data when you have enough examples. For small tabular data, simpler models such as linear models and tree ensembles are often as good, faster to train and easier to explain.

What mistakes do beginners make?

Not scaling inputs, using a learning rate that is too high, judging the model on training data only, training too long and not setting a random seed for repeatable results.

How do I practise as a beginner?

Train a tiny network on a small dataset such as handwritten digits or a simple tabular task with a library. Change one setting at a time, such as the number of hidden units, the learning rate or the activation function, and record how the training and validation loss change. Plot both curves; a gap that grows between them is the visual sign of overfitting. Writing down what each change did gives you real intuition.

What should I learn next?

After the basics, study convolutional networks for images, recurrent and transformer models for sequences and text, and ways to reuse pretrained models. Learn how to evaluate models with the right metrics and how to explain results to others. Read the documentation of a deep learning library and follow a few small projects from start to finish.

Frequently asked questions

What is a neural network?

It is a machine learning model of connected layers of artificial neurons that learns to map inputs to outputs by adjusting weights to reduce error.

What does an activation function do?

It adds non-linearity so that stacked layers can learn complex, curved patterns instead of only linear ones.

What is backpropagation?

It is the method that efficiently calculates how much each weight contributed to the error so gradient descent can update the weights.

What is an epoch?

An epoch is one complete pass through the whole training dataset, during which the weights are updated many times in batches.

When should I avoid neural networks?

For small tabular datasets, simpler models are often as accurate, faster to train and easier to explain.

Sources and further reading

- PyTorch documentation
- TensorFlow API documentation
- scikit-learn documentation

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550