<https://www.manning.com/books/deep-learning-with-python>

A machine learning system is trained rather than explicitly programmed. It’s presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allow the system to come up with rules for automating the task.

Machine learning models are all about finding appropriate representations for their input data – transformations of data that make it more amenable to the task at hand. All machine learning algorithms consist of automatically finding suck transformations that turn data into more useful representations for a given task. Machine learning algorithms aren’t usually creative in coming up with these transformations they are just searching through a predefined set of operations, called a hypothesis space. So: machine learning is searching for useful representations of some input data, within a predefined space of possibilities, using guidance from a feedback signal which measures whether the algorithm is doing a good job (some measure of the distance between the algorithm’s current output and its expected output).

Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. 'Deep' isn't a reference to a deeper understanding obtained by this approach, rather it stands for the idea of successive layers of representation. In deep learning these layered representation are almost always learned via models called neural networks, structured in literal layers stacked on top of each other.

A neural network transforms the original input into representations that are increasingly different from the original image and increasingly informative about the final result. A deep network is a multistage information-distillation operation, where information goes through successive filters and comes out increasingly purified (that is, useful with regard to some task).

In more detail: the specification of what a layer does to its input is stored in the layers’ weights, which is in essence a bunch of numbers. In technical terms, we’d say that the transformation implemented by a layer is parameterised by its weights.

To control the output of a neural network you need to be able to measure how far the output is from what you’d expect – this is the job of the loss (or objective) function of the network. The loss function takes the predictions of the network and the true target (what you wanted the network to output) and computes a distance score, capturing how well the network has done on this specific example. The fundamental trick in deep learning is to use this score as a feedback signal to adjust the value of the weights a little in a direction that will lower the loss score for the current example.

Initially the weights of the network are assigned random values, so the network merely implements a set of random transformations. And so naturally the output is far from what it should ideally be. But with every example the network processes, the weights are adjusted a little in the correct direction and the loss score decreases. This is the training loop which, repeated a sufficient number of times (typically tens of iterations over thousands of examples) yields weight values that minimise the loss function.

**Probabilistic modelling**

This is the application of principles of statistics to data analysis. Naïve Bayes is one of the best known algorithms in this category. It is a machine learning classified based on applying Bayes theorem while assuming that the features in the input data are all independent.

Previous machine learning techniques only involved transforming the input data into one or two successive representation spaces, usually via simple transformations such as high-dimensional non-linear projections, or decision trees. But the refined representations required by complex problems generally can’t be attained using these techniques. Thus, humans had to go to great lengths to make the initial input data more amenable to processing by these methods, and had to manually engineer good layers of representations for their data. This is called feature engineering. However, deep learning automates this step and you learn all features in one pass rather than having to engineer them yourself.

**Neural networks**

The core building block of a neural network is the layer, a data processing module that acts as a filter for data. Some data goes in, and it comes out in a more useful form. Specificall, layers extract representations of the data fed into them – hopefully representations that are more meaningful for the problem at hand.

Tensors

A tensor is a container for data, normally numbers. A matrices is a 2d tensor: a tensor is a generalisation of matrices to an arbitrary number of dimensions.

All of the following are examples of tensors: number, scalar, array, vector, 2d-array, and matrix.

So tensors are multi-dimensional arrays or nd-arrays for short. The reason we say tensor is a generalisation is because we use the word tensor for all values of n. The number of axes is the rank of the tensor.

**What is a hypothesis in machine learning?**

Supervised machine learning is often described as the problem of approximating a target function that maps inputs to outputs (function approximation). This is characterised as searching through and evaluating candidate hypothesis from hypothesis spaces. This can be confusing when hypothesis has a distinct but related meaning in statistics and science.

* A scientific hypothesis is a provisional explanation for observations that is falsifiable
* A statistical hypothesis is an explanation about the relationship between data populations that is interpreted probabilistically
* A machine learning hypothesis is a candidate model that approximates a target function for mapping inputs to outputs

A hypothesis is an explanation for something.

In machine learning, the choice of algorithm and the configuration of the algorithm define the space of possible hypotheses that the model may represent.

<https://machinelearningmastery.com/applied-machine-learning-as-a-search-problem/>

The goal of the learning system is to learn a generalised mapping between input and output data such that skilful predictions can be made for new instances from the domain where the output variable is unknown. The learned mapping will always be imperfect.

**What is an activation function? (or transfer function)**

This defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

Simply put it calculates a weighted sum of its input, adds a bias and then decides whether it should be fired or not. Every activated function passes information on to the next layer.

Activation function decides whether a neuron should be activated or not. This means that it will decide whether the neuron’s input to the network is important or not in the process of prediction.

The purpose of the activation function is to add non-linearity to the neural network

Purpose of the activation function is to introduce non-linearity into the network.

**Elements of a neural network**

Input layer – accepts input features. It provides information from the outside world to the network, no computation is performed, nodes here just pass on the information to the hidden layer

Hidden layer – nodes of this layer are not exposed to the outer world, they are just part of the abstraction provided by any neural network. Hidden layers perform all sorts of computation on the features entered through the input layer and transfer the results to the output layer

Output layer – This layer brings the information learned by the network to the outer world

Layers consist of small individual units called neurons.

<https://machine-learning.paperspace.com/wiki/weights-and-biases#:~:text=A%20neuron,have%20the%20value%20of%201>.

Weights = control the signal between two neurons. In other words, a weight decides how much influence the input will have on the output

Biases- which are constant, are an additional input into the next layer that will always have a value of 1. It is a bit like intercept in a linear equation. It is an additional parameter in the neural network which is used to adjust the output along with the weighted sum of the inputs to the neuron. Therefore bias is a constant which helps the model in a way that it can fit best for the given data.

**output = sum (weights \* inputs) + bias**

Bias allows you to move the line down and up fitting the prediction with the data better. If the constant c is absent then the line will pass through the origin (0, 0) and you will get a poorer fit.

The bias node in a neural network is always on.

<https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98>

Two issues with machine learning models:

* Model over fits data (overfitting is like learning by heart – good at answering questions its already been asked, but when you ask it something out of the box it fails)
* Model cannot learn patterns from data

<https://www.youtube.com/watch?v=aircAruvnKk>

Neuron -> thing that holds a number between 0 and 1. Can actually think of each neuron as a function, which takes in as inputs all values from the previous layer and spits out a number as an output. Really, the entire network is a function (a complicated one!)

<https://www.guru99.com/backpropogation-neural-network.html>

Back propagation in neural networks is a short term for backward propagation of errors

<https://www.guru99.com/deep-learning-tutorial.html>

Deep learning occurs in two phases:

* The first phase consists of applying a nonlinear transformation of the input and creating a statistical model as the output
* The second phase aims at improving the model with a mathematical model known as a derivative

The neural network repeats these 2 phases hundreds to thousands of times until it has reached a tolerable level of accuracy. The repeat of this two phase is called an iteration.

Types of neural network:

* Feed-forward neural network: the simplest kind. Information flows in one direction only, forward. The information starts at the input layer, goes to the hidden layers and ends at the output layer. The network does not have a loop
* Recurrent neural network: a multi-layered neural network that can store information in context nodes, allowing it to learn data sequences and output a number or another sequence. The network includes loops.

<https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9?gi=794f014041f5>

RNNs are good at processing sequence data for predictions. How? Using sequential memory

<https://builtin.com/data-science/recurrent-neural-networks-and-lstm>

Feed-forward neural networks have no memory of the input they receive and are bad at predicting what’s coming next. In RNNs the information cycles through a loop and when it makes a decision it considers the current input and also what it has learned from the previous inputs.

With a normal feed-forward network if you give it the word ‘neuron’ and it processes it character by character, then by the time it reaches character ‘r’ it will have already forgotten the previous characters and so be incapable of predicting which character would come next. A RNN is able to remember those characters, however, because of its internal memory. An RNN has two inputs: the present and the recent past.

With backpropogation you basically try to tweak the weights of your model whilst training it.

A gradient is a partial derivative with respect to its inputs. A gradient measures how much the output of a function changes if you change the inputs a little bit. The higher the gradient the steeper the slope and the faster the model can learn. But if the slope is 0 then the model stops learning. Vanishing gradients are when the values of the gradients are too small and the model stops learning or takes way too long as a result. This was solved through long short-term memory.