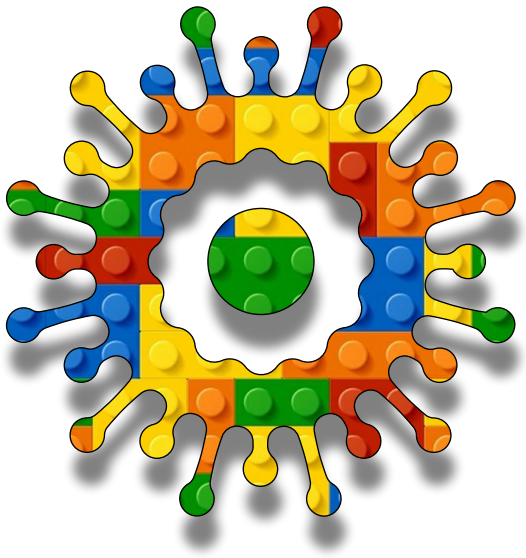
Artificial Intelligence information a brief exploration of AI



Contents

- 🀄 Loss Function
- 🀄 Activation Functions
- 🀄 Pre-trained Models
- 🀄 Overfitting
- 🀄 Underfitting
- 🀄 Normalisation
- 🀄 Gradient Descent
- 🀄 Learning Rate

What is artificial intelligence?

Artificial Intelligence could be described as the ability for a computer to perform tasks usually done by a human, such as learning, problem-solving, and decision-making. The process is to create machines that can simulate human behaviour or intelligence.

This is a very generalised view; you will find that others will have very differing takes on this depending on how one looks at it. The main issue is that we cannot agree on the definition of human intelligence, let alone Artificial Intelligence. Machines don't work like human brains, yet we have an expectation that they do. They work in zeros and ones (binary), our brains also work on electrical impulses and, in that respect, can seem similar.

The human brain has neurons connected to lots of other neurons (86 billion). The thought is that if you can build a big enough machine, then you could start to replicate the processing power of the brain. The rise of Large Language Models (LLMs) like ChatGPT and Gemini has fuelled this notion that we can create a highly intelligent machine capable of competing and surpassing human intelligence.

This has made some people very excited and some people very anxious. The former see it as the solution to the world's problems, like climate change. The latter group sees it as an existential threat to the human race, citing the advent of what is called the singularity. This is when Artificial Intelligence has the capacity to outstrip human cognitive ability and decides to solve the world's problems by removing the main cause of the world's problems, us.

Whatever your views are on this topic, it is here to stay and will only become a bigger part of our lives. The advent of AI is more than hype (although there is certainly a lot of that too), where venture capitalists are investing billions in this industry. Nation-states are realising the power of this technology as a weapon and an opportunity for economic power.

AI or Machine Learning (ML) has so much mystique and mystery about it. Yet it is now easier than ever to use and learn how to develop your own applications from scratch. This is why I have written this tutorial (with the

help of the Coding Train, see references). I have tried to unpack this so that anybody can have an understanding of how a Machine Learning or AI model works.

Although there are some very big players in the AI world, that is only a tiny part of it. It is important that more people are involved in this industry, developing ideas and applications, creating imaginative solutions to problems. Like any new technology, it will attract all sorts of people, for all sorts of reasons, from all sorts of backgrounds, so why not you?

To really understand how most AI/ML models work, we need to take a close look at the algorithm which underpins it. This algorithm is called a Neural Network.

What is a machine learning algorithm?

We use the term Artificial Intelligence (AI) to cover everything from robotics, driverless cars, to facial recognition, or recommending your next Netflix show. A lot of what we call AI is actually a form of Machine Learning (ML). Machine Learning is the backbone of most AI models and is an algorithm designed to look for patterns in data.

An algorithm is nothing special in itself. It is simply coding a set of instructions, similar to a recipe to make a cake. Some algorithms are very short and simple, others are much longer and complex, but at the end of the day, it is usually just a mixture of coding and some basic maths. What some clever people have done is to combine the two, inspired by how the brain processes information.

The most common Machine Learning algorithm is called a Neural Network because of its similarity to the human brain. In reality, the brain does not behave like this, but the success of this algorithm has seen its adoption in most of the ML world. It has a surprisingly simple architecture (structure) but has been tweaked and developed as people research different ways to solve problems more efficiently. The foundation of this algorithm is a basic Neural Network.

Data scientists and software engineers use neural networks to solve a very wide set of problems from, say, robotics to the medical and financial world. Needless to say, it has not gone unnoticed by the military, and this is often where some of the concerns lie.

What is a neural network?

Although I consider the best way to learn is by doing, which is what the tutorial is all about, it won't hurt to have a simple overview of what it might look like. However, please don't worry if it seems complicated or you still don't understand; there are so many bits and pieces that make up the whole picture of how a neural network works. What we are going to do is treat it like a jigsaw and look at some of the key pieces in isolation.

The neural network is an algorithm. It is some code and some maths. You cannot actually see anything; it is what people call a black box. Despite the coding and the maths being very well understood, it is not entirely clear what is happening inside. The architecture may be relatively straightforward, but the process of how it is learning is complex and based on initial random values. How it learns is not how we learn, and very often it comes up with solutions that surprise everyone and do something unexpected. This was certainly true when DeepMind's AlphaGo beat the leading Go player; move 37 surprised everyone because a human player wouldn't have made such a move.

This is nothing sinister but a product of the complexity of the neural network. It has lots of nodes (neurons), lots of weights (random variables) connecting these nodes, which it changes incrementally to learn on the training data provided.

When you start to understand how it works, it seems relatively simple and logical. It can find patterns in data that we might miss. However, there is sometimes an assumption that it will always be perfect, which it isn't; it never is and never will be. All it does is make an approximation. This is why it is a real issue to fully rely on AI; it always needs some human oversight.

Nodes and weights

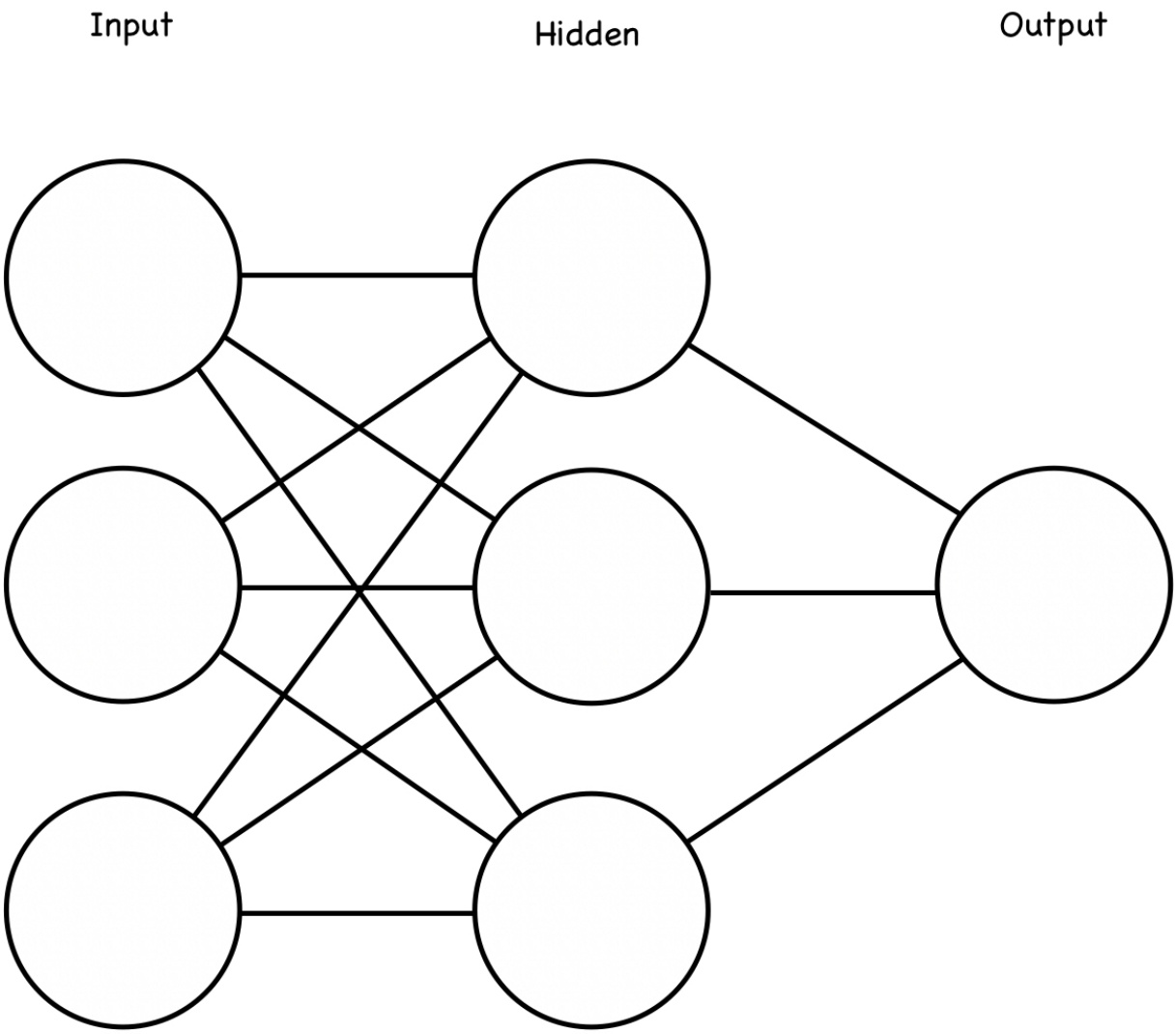
There are the two main aspects of a neural network: nodes (or neurons) and weights. The nodes are like the neurons in the brain, where they fire (or not) depending on the signals coming in. In a neural network, the nodes are connected together to form layers, where the inputs from one layer pass through into the nodes of the next layer. Each node is connected to all the nodes in the next layer; these connections are weighted.

These weights are, initially, given random values; think of them as dials or knobs that are tweaked as the neural network learns. The neural network starts off by knowing nothing and makes a wild guess based on no previous knowledge. But as the data is passed through the neural network repeatedly, the neural network changes the weights (remember they are like dials or knobs) until the guess comes closer and closer to the real answer. This is the training phase. The difference between the true answer and the neural network's guess is often called the loss.

This is an oversimplification but is the essence of how a neural network learns, by adjusting the weights based on the error between its guess and the real answer. There can be thousands and thousands of these weights, and that is why it is hard for us to know exactly what has happened inside the neural network because of the random nature of those weights.

We often draw a diagram of a neural network shown below. Where there are three types of layers: they are called the input layer, the hidden layer, and the output layer. The lines connecting them are what we call the weights.

Figure 1: diagram of a simple neural network



Inside a node

Each node has a very simple job; it takes in a single value from all the previous nodes (more on that later) and then passes it through an activation function and pushes out another value. This is similar to a neuron in the brain firing or not firing, except it takes in a value and sends out another value to the next layer of nodes.

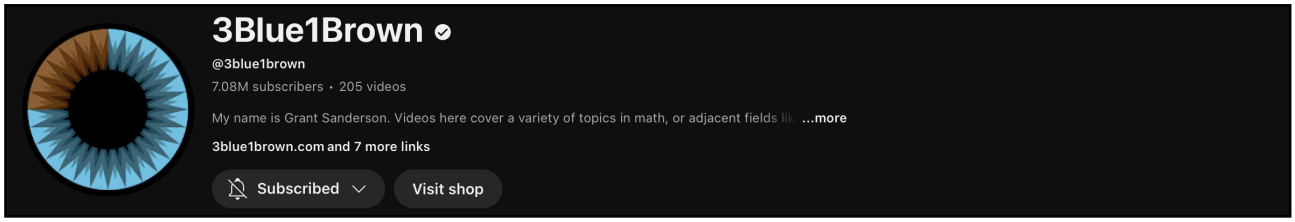
This is really where a neural network differs from how the brain works. Neurons in the brain don't do calculations based on maths; we are still learning how the brain actually does work, which is the fascinating topic of neuroscience. What we have created is an approximation of an artificial version of the brain, which is an artificial neural network, but remember that it does not behave like a human brain. It is, at best, a mathematical and coding algorithm, not a sentient machine, at least not yet!

Returning to our diagram, we have three layers. The data may come from a sensor, historical data, an image, or video into the input layer, and is then passed through to the hidden layer. Each bit of data is multiplied by the weight and then all are added together before being passed into each node in the hidden layer.

That is what weighted sum means; it is the sum of all the weights times the input data. From the nodes in the hidden layer, they produce a value which is also multiplied by a weight and then the receiving node adds them all up before processing the value. If this seems complicated, it isn't really; it just takes time to explain what is going on.

There is a brilliant series on the YouTube channel `3Blue1Brown` on neural networks which graphically explains this better than I can. I will provide a link to those videos in the `Resources` tab.

The best neural network explanation



The data may pass through many hidden layers before reaching the final layer, the output layer. The output layer may use a slightly different function to produce the final answer or output. This first part of the process is called `feedforward`.

There will be a difference between the guess and the real answer. This difference is sent back through the neural network, changing all the weights a tiny bit. This part of the process is called `backpropagation`.

The data is sent through again from the input layer with the newly adjusted weights to see how much error there still is. The backpropagation is repeated, changing the weights a little bit more. This is repeated again

and again until the error is close to zero. The neural network model is then said to be trained and is ready for testing.

The error is often called the loss, and it uses a function to calculate it called the cost function. Even so, there are more than one cost (loss) function. You can see there is a lot to this fascinating topic, and we have barely got started.

The previous diagram of the neural network had three input nodes, with three nodes in the hidden layer and one output node. The data decides how many input and output nodes there are, but the number of hidden nodes and hidden layers is entirely up to you. This is where it becomes more art than pure science or maths.

The key elements to a neural network are:

- The number of layers
- The number of nodes
- The weights
- The activation functions
- The loss (error)

Glossary of terms found in a Neural Network

Instead of going through everything in fine detail below, there are some brief descriptions on a number of key topics relating to Machine Learning and Neural Networks. You could write pages, if not books, on each of these, but for the main part, just a working knowledge is required to get started.

These are snippets of information. It is useful to be aware of their meaning and purpose before we start using them. If you are one of those who are happy to read lots before starting, then I will try to include some books, videos, and articles that might be of interest to you in the resources section of the website.

Neural Network

Is a particular type of machine learning model; it usually consists of nodes (neurons) in layers. The neural network can be considered the architecture, and some architectures can be very complex.

Model

A model is the overall process of training on patterns of data, then predicting. You might find I use the term interchangeably with neural network.

Artificial Intelligence

This is a generic term to represent anything that mimics human behaviour, whether it uses a neural network or not.

Machine Learning

This is a subset of artificial intelligence that often refers to the use of neural networks, although not always.

Deep Neural Networks

This is a subset of Machine Learning that has many hidden layers, thus the term deep as opposed to shallow, where there may be just one or two hidden layers. Most neural networks are likely to be Deep Neural Networks (DNNs).

🀄 Hyperparameters

They are the things or parameters you have control over, such as the number of hidden layers, the number of nodes in those hidden layers, the number of epochs, the batch size, type of activation functions, and learning rate, and so on.

🀄 Input Layer

This is predetermined by the dataset; it can be just one, two inputs, or many thousands, such as in an image.

🀄 Hidden Layers

This is a hyperparameter determined by you; it can be just one layer or many layers, which just depends on the complexity of the dataset. More hidden layers don't always equate to better.

🀄 Output Layer

This can be a single node or a number of nodes depending on what the task or problem is. This is also defined by the data you are using.

🀄 Datasets

This is the whole data you are going to use. Usually, it will include all the inputs and the corresponding outputs. The machine learning algorithm will try to establish a relationship between the inputs and outputs through adjusting the `weights`. These need to be as large as possible, error-free, and accurate. You will need to split your dataset into three components: training (80%), validating (10%), and testing (10%); all three sub-datasets need to be representative, completely different, and only used for those tasks of training, validating, and testing.

🀄 Types of Learning

There are three main types of learning:

1. `supervised learning`,
2. `unsupervised learning`, and
3. `reinforcement learning`.

🀄 Supervised Learning

This is a dataset that has inputs and labelled corresponding outputs; it learns to match the inputs to the output through a neural network.

🀄 Unsupervised Learning

This only has the `input data`; it has no output data. In order to classify and learn, it has to find patterns or features in the data and then group them according to those features. For instance, if you show it pictures of cats and dogs (with no labels), it might, hopefully, start to group the animals according to their similar features.

🀄 Reinforcement Learning

This is where it has no dataset at all to work with; it is often just a `reward system` for making certain decisions. It will explore, possibly randomly, to see which actions give the best rewards and learn from that. This is often used in training robotics.

🀄 Tasks

The task or problem you are trying to solve can broadly be categorised into one of two broad headings: `Classification` or `Regression`.

🀄 Classification

This is to classify according to `labels`. For instance, classifying images of cats and dogs after being trained on thousands of labelled images of cats and dogs. It then predicts the likelihood or probability that an image presented to the model is either a cat or a dog.

🀄 Regression

This usually gives you a value, for example, the price of a house after being trained on a dataset that has `features` based on many houses, their sold price plus all other features such as number of bedrooms, garage, size of garden, distance to nearest train station or school, and so on.

🀄 Feedforward

This is the process when all the data is passed forward through the `neural network`; this starts at the input layer, then goes through the hidden layer(s), and finally to the output layer, with the final prediction.

🀄 Backpropagation

Once the feedforward process is complete, the loss is calculated and then that is passed back through the neural network, tweaking all the weights before beginning the whole process again.

🀄 Fully Connected

This means that all the nodes from the previous layer are connected to all the nodes in the next layer. Also called Dense.

🀄 Weights

These are random values, usually between 0 and 1 (or −1 and +1). One way to think of these is as the connections between the nodes in one layer with the nodes in the next layer. These are adjusted each time backpropagation takes place.

🀄 Bias

To stop a scenario of nodes or layers becoming zero, which means it cannot be trained, each layer is given an extra node called a bias, which usually has a value of 1 to keep the neural network from collapsing.

🀄 Training

This is done on the bulk of the dataset. It is trained on it until the loss is near to zero. You set the hyperparameters to what you think are most likely going to give the best result, avoiding underfitting and overfitting.

🀄 Validating

This is the dataset you use after you have finished the training phase. It is set apart from the training dataset. This is an opportunity for you to see which hyperparameters need adjusting to get an even better result. This dataset is used to check the accuracy of the training model with previously unseen data. After you have changed the hyperparameters, the model is retrained again, followed by another validating process. Only when the results are acceptable do you move onto the next phase of testing the model.

🀄 Testing

Once you have trained the model on the training dataset, and you have validated it through adjusting the hyperparameters, you save the model and test it. This uses the final sub-dataset and will highlight if the model works well or underfits/overfits. The test dataset should not be used again. The testing phase is the final check before using the model for real. The model won't be perfect, but you will have to decide whether it is good enough to use in the real world.

🀄 Epochs

This is when a complete training dataset has passed through the model once. The number of `epochs` you might pass through the neural network during training depends on the loss; there is no point in training beyond a certain loss point. This is when the loss is so small that any extra time training is pointless. Choosing the number of epochs is one of the main hyperparameters.

🀄 Batch Size

The `batch size` is the size of a chunk of data that passes through the model before the neural network is updated. It is a proportion of the total data, so if an epoch is `1000` bits of data and the batch size is `10`, it will update the model `100` times per epoch. A small batch size may improve accuracy but takes longer, and conversely, a larger batch size is less accurate but quicker. It is a bit of trial and error; also, it is another hyperparameter you have to make decisions about.

🀄 Loss or Cost Function

The words `loss` or `cost` are interchangeable, although there is a subtle but distinct difference. This is the measurement between the prediction (or guess) and the true results, label, or output data. This goes through various mathematical functions and decides how much the weights are adjusted by. The loss function is usually shown as a chart and aims to be as close to zero as possible. There are numerous loss or cost functions; `Mean Squared Error` is one such function, which also depends on the task.

🀄 Activation Functions

Inside each node is an activation function which takes the multiplication of the weights and input data from all the nodes in the previous layer, adds

them together (weighted sums), and passes them through the activation function so that the output from that node is a single value based on the weights, inputs, and activation function. Historically, the Sigmoid function was always used, but now modern neural networks use ReLU, which is much easier computationally, which also means quicker and cheaper, and is also very effective. This is another hyperparameter, where you decide which activation functions to use and where.

🀄 Pre-trained Models

These are models that can be used in applications such as facial recognition and voice control. The hard work (training, validating, and testing) has already been done for you, and you can use the model for whatever purpose seems appropriate.

🀄 Overfitting

This is where the model has learned the data by heart and has not really approximated it. What this means is that when it is presented with data it has not seen before, its accuracy is very poor despite achieving a very low loss value in training. If it seems too good to be true, beware!

🀄 Underfitting

This is usually a model that has been trained on a dataset that is too small. You may get reasonable results in training, but the test data produces poor end results.

🀄 Normalisation

This is applied to the dataset before training begins; it changes the size of the input values, which can vary in size considerably (think of the coordinates in an image), to values between 0 and 1 or −1 and +1. Neural networks work best with values of this magnitude; it avoids the impact of a few large numbers/values having a disproportionate effect on the training process. The values of the weights are also of the same magnitude.

🀄 Gradient Descent

This is where the loss function is calculated. It works out how much to change the weights based on the current loss. The gradient of the curve is the key component. The steeper the curve, the bigger the adjustment.

🀄 Learning Rate

This is a hyperparameter that controls how big a step each learning iteration takes. It stops the model from taking huge steps, which might miss or overshoot the lowest loss point. Without the `Learning Rate`, it may never really settle on the minimum loss. The smaller, the better, but it takes longer and may also not reach the minimum (or stop at a local, not global minimum). If you use a large `Learning Rate`, you might overshoot.

🗒️ Notes

There is a lot to this, and I have glossed over some of the fine detail and oversimplified some of the important features. At this stage, it is an introduction to some of the key elements involved and not an in-depth explanation. It will give you a flavour of this fascinating topic. Everything becomes clearer and more concrete when you start coding, building your own neural networks, and training them in this tutorial.

🌻 Next

In the next section, I want to introduce you to the environment you will code in, the web editor for p5.js. This is where you will code and see the machine learning in action, but first, I want to show you around the workings of that web editor.