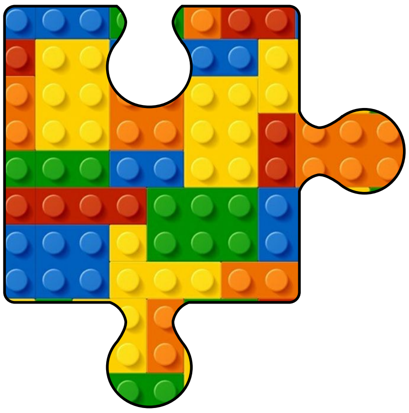
The Joy of Coding

Artificial Intelligence

Intro to Artificial Neural Networks



Content

What is meant by Artificial Intelligence

Artificial Intelligence is a general term to describe a machine, usually a computer doing some work that a human could do. It can be called machine learning because in essence that is what it is doing, learning.

There is nothing sinister about AI, it is simply an algorithm that runs some code based on some data and produces an outcome. What is clever about it is that it can learn things without being explicitly told how to do it.

To do this it needs data, lots of data e.g. images or information. The classic example is the dog v cat scenario. You show it lots of pictures of dogs and tell it that these are dogs, then you show it lots of pictures of cats and say these are cats.

It is then shown a picture of a cat or dog it hasn't seen before and if the training is successful it should correctly identify whether it is a cat or a dog with a great deal of certainty. I mean how hard can it be?

The answer is, surprisingly, very hard indeed. The computer has no concept what a dog or a cat is, a two year old child could guess it right 99% of the time but a computer can really struggle.

Yet if there is a huge amount of data e.g. weather for many years it can sit there and process it very fast and for as long as it takes without getting tired or needing a break.

There are things that AI is good at and things it is not so good at. But it is a useful tool and one that is getting better and better especially with the advent of faster and more powerful graphics processors. They were developed for the gaming industry but it meant that machine learning algorithms could now work even faster than ever before.

What we will be using

To explore AI we are using a machine learning library called TensorFlow. It has often been used in another coding language called python. TensorFlow.js is doing all the heavy lifting for you, in the recent past you would have to understand how that library worked and was beyond the scope of the average programmer.

Now you have other libraries that sit on top of it. In python you have one called Keras but we will use on called ml5.js. The beauty of ml5.js works seamlessly with p5.js. This makes life so much easier. There is so much going on behind the scenes which in due time you will need to know about to continue to explore this fascinating topic.

For the moment it will suffice to know there is a neural network, that there are layers in the network. Also what kind of network it is, whether it is classification or regression. So the next section is worth a read through before you start just to give the most basic introduction to the concepts used.

I have kept it deliberately brief and over the course of the next few sections will cover more aspects as they become necessary. What I mean is I want to drip feed it to you as and when it (hopefully) makes more sense.

You can head over to the web page where there is a lot more information and many examples.

www.ml5js.org

A Neural Network

A single perceptron, although useful can only solve linear problems. To solve more complex and non-linear problems you need a network of nodes, called an Artificial Neural Network, this term was coined originally because it was inspired by how the brain connected all these neurons.

Even though people think that an Artificial Neural Network is similar to how the brain works because of the regional terminology it doesn't but the name stuck we just use the name Neural Network but don't think there are too many links with how the brain works.

Below is a diagram of a simple Neural Network. It now has five layers (technically only four layers because the input layer isn't actually a layer just the inputs).

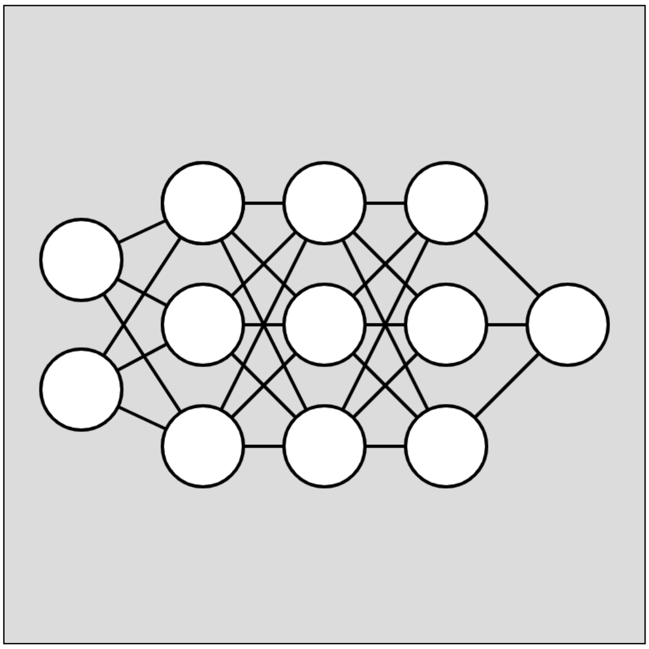
There are three types,

input (not really a layer!),

hidden layer(s) and

output layer.

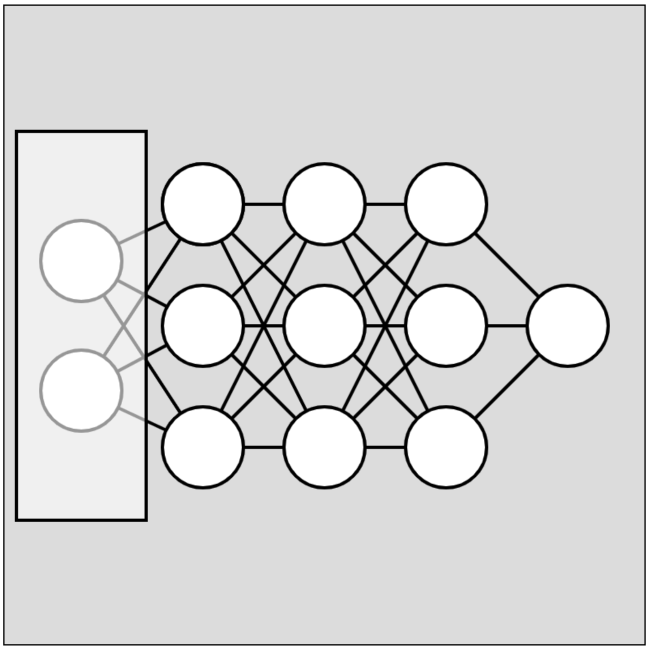
In this case there is an input layer, three hidden layers and an output layer, let's look at these in a bit more detail



Input layer

Highlighted below is the input layer, it can have any number of input nodes. For instance if it was an image it could have thousands. Technically this is not a layer because it doesn't have nodes as such just input data.

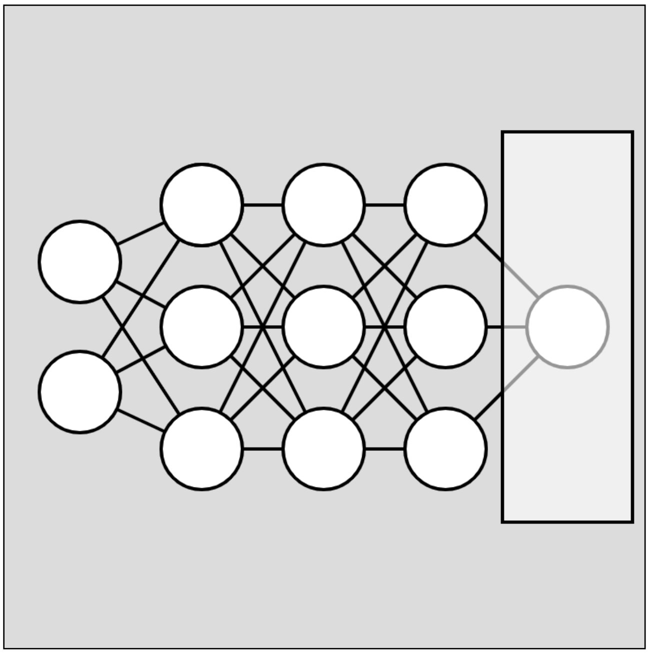
Input Layer



Output Layer

This layer again can have many nodes. In this case just one. For a classifying whether the image is of a cat or dog it will have two, one for dog and one for cat.

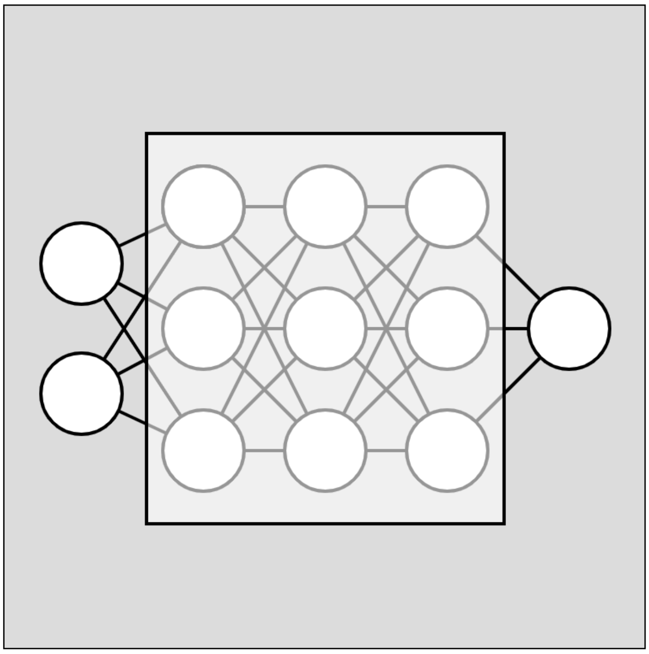
Output Layer



Hidden layers

Thesis are the layers in-between the output and input layers. There can be just one layer or many. In this case there are three hidden layers with each layer having three nodes. The layers don't have to have the same number of nodes and usually don't.

Hidden Layers



Types of Networks

There are two types of networks based on the type of outcome.

1. Classification (e.g. where the picture is either a cat or dog)
2. Regression (e.g. where the outcome is predicting house prices)

Classification

This is where you define what kind of output you want and what your out put actually is. You might be classifying images of dogs and cats. So you will send in lots of pictures of dogs and label them 'dogs' and the lots of pictures of cats and label them 'cats'.

The model will then run this data through lots of times and when you show it a picture of a dog or cat it hasn't seen before it should give a confidence score (probability) which will indicate whether it is in fact a dog or a cat.

You may think that it should easily identify one from the other but you would be surprised what is can and can't do easily. For us telling a dog from a cat is easy, we could do it with 100% accuracy when just a few years old. However a computer doesn't actually know what a dog or cat is.

All it has to gone on is a bunch of pixels arranged in such a way that we have labelled 'dog' or 'cat'. This is so very challenging for the network and is where much is currently being invested. Time and money.

You might also use it to categorise numbers, letter, words, speech, movement etc

Regression

This is different to classification because the output is a value not a classification like a dog or cat. You might want to predict the weather for tomorrow based on a lot of environmental data. Predict stock market, house prices or even the national lottery.

This is still quite similar to classification in every other way. You might have images of your arm in different positions and use that data to control a game or a musical instrument depending on how high or low you move your arm.

Types of Learning

Learning can be based on different types of data sets
1 Supervised Learning (data with labels)
2 Unsupervised Learning (data without labels)
3 Reinforcement Learning (no data, no labels just rewards)

Supervised Learning

This is one of serval (possibly the most common) ways the network will learn, it can be for classification or regression. All it means is that all the data has a label attached and it just learns to match the input data with the output data (or labels).

It is said to be supervised. Obviously there is unsupervised learning where the data is presented with no labels and it looks for patterns and groups things according to those patterns. E.g it may work out that all dogs have similar features and so it classifies all the dogs and all the cats separately.

Equally well it may not so most data has labels attached and this is a human intensive job. This means someone has to sit down and label each image of a cat or a dog. Bear in mind you might need 10s of thousands of images, that is a very labour intensive and costly exercise.

Unsupervised Learning

This is where the network is presented with lots of data but no outcomes (labels). It looks at patterns in the data and creates its own labels. You could show it lots of cats, dogs and horses and it looks to see how it can group them according to the features in the images.

Reinforcement Learning

This is where much of the research is done, and in my opinion is real AI. In this situation it has some data which it has to make sense of. Remember this a computer that has no concepts of what the world is like outside its own little brain of '1's and '0's.

An example would be playing chess. You could tell it all the rules and hope it makes good decisions. Yet it doesn't actually know what the point of the game is (it doesn't even know it is a game). So you give it rewards for making good moves and remove points for making bad ones. I have simplified this to make the point.

It then learns what are good moves and is encouraged to make more good moves to maximise the total score. Taking this a step further, you don't even tell it the rules so it starts off blindly trying different things to see what happens. Bit by bit (and it can take a long time) it starts to work out the best strategy to get the most points (highest score). You could say it is learning without being explicitly programmed to play the game.

This was brilliantly demonstrated by a company called 'DeepMind' who created an AI that could play the game of 'Go'. It was called **AlphaGo** and it beat on the best players in the world. This was shocking because no-one thought it was possible for a computer to learn to play such a complex strategy game so well. There is a documentary on this that you can find on YouTube, I would highly recommend that.

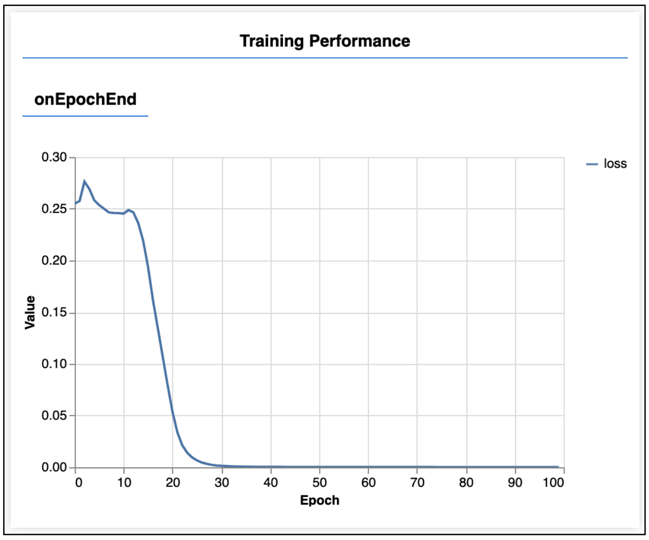
The most recent developments to help science and medicine in particular is something called **AlphaFold** which is going to have a huge impact on medicine and related technologies.

Loss

We need to talk about loss. The loss is the difference between the actual (target) and the guess. We want to train the model with its guesses using the weights to get that loss as close to zero as possible. That is the art of training the Neural Network and some of it is trial and error.

We can see the loss as it progresses through the network model. Remember it is training itself to minimise the loss the difference between the actual and its guess. So that it has a model that works well on any data not just the data you have given it. We can see this from a chart shown below...

Loss chart


Training Performance
onEpochEnd
Value
Epoch
loss