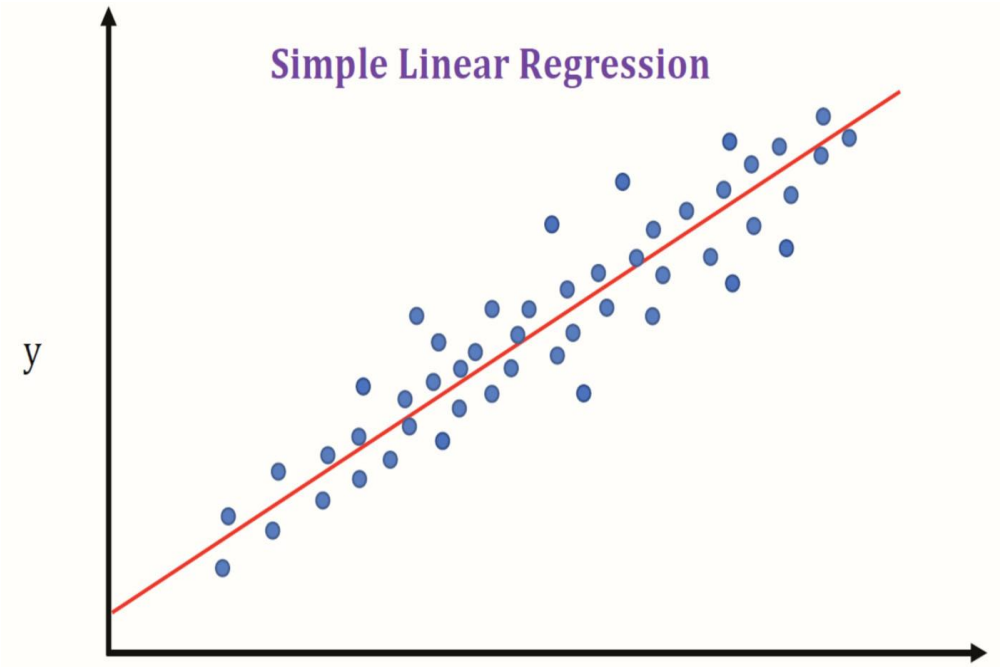
Some Notes on Machine Learning

19 March 2022

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. [1]

It is a sub-field of artificial intelligence (AI) and AI is a sub-field of probability and statistics.



Linear regression [Ref. 2]

Linear regression is a simple example of machine learning.

The equation for the straight line in the figure is y = a.x + b. The parameters 'a' and 'b' can be chosen to minimise, for example, the sum of the square of the difference between the true values of 'y' and those predicted by the straight line. In the language of machine learning, this is called 'training'.

The complete collection of x, y pairs is called the dataset.

If the dataset changes, then the parameters 'a' and 'b' are updated and this can be done programmatically by simply re-training. This is of great advantage if a program had thousands or hundreds of thousands of parameters because there is no need to change them manually.

A dataset can consist of 10s of thousands of x,y pairs.

A more complicated example of a machine learning technique, and one that is in very wide-spread use, is the 'artificial' neural network (ANN) as opposed to the 'biological' neural network. These both consist of interconnected neurons.

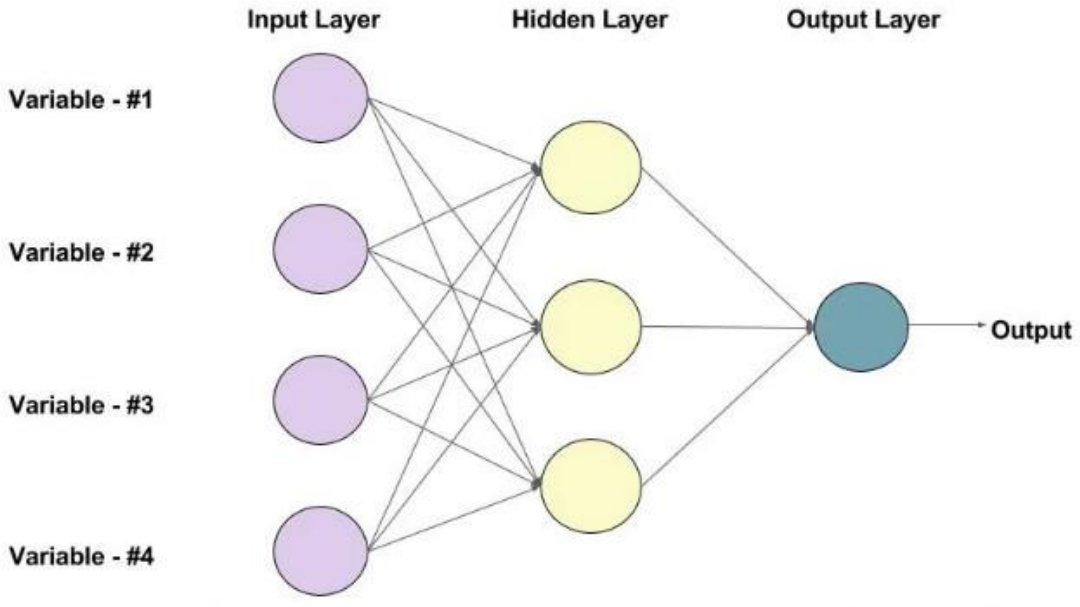
Biological neural networks are part of just about every living creature. A worm has a few hundred neurons, a frog has about 400 million neurons, which is about the number used in a modern very large neural network running on a supercomputer, and the human brain has about 86 billion neurons.

A structure of a very simple ANN is shown below and represents the most basic form of a biological neural network.

Ignore the two circles with the '1's for now. The remainder of the circles are the neurons, nodes or more properly, perceptrons, as they are artificial representations of neurons.

The biases are used to introduce non-linearity into the network. This topic is beyond my current understanding but I have found by experimentation that sometimes they can be excluded.

Whether or not they are included does not depend on the complexity of a problem but whether or not the problem is linear. If the problem is non-linear then biases are required. The idea of linearly of a problem in terms of a neural network is not yet clear to me.

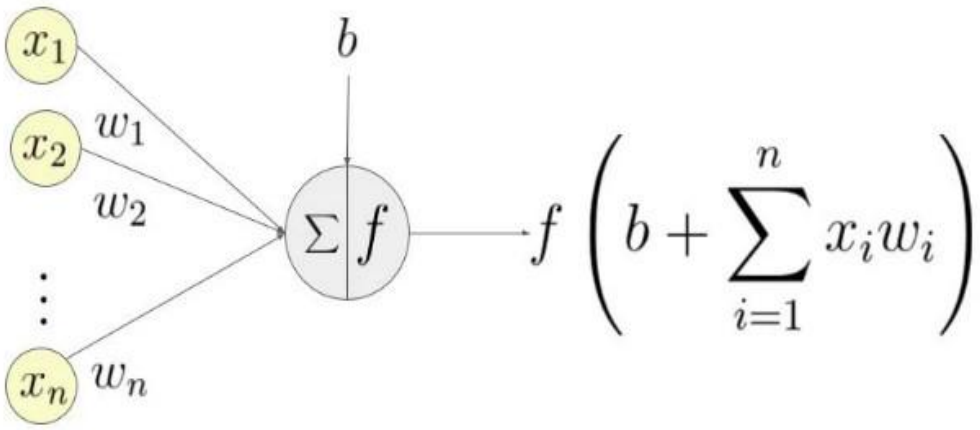


A simple neural network [Ref. 3]

The inputs are analogous to the values of 'x' and the output(s) are analogous to the values of 'y' for the case of linear regression. The 'W' and 'B' values are equivalent to the values of 'a' and 'b' for the case of linear regression.

There can be more than one hidden layer. Typical vision systems that I have experimented with at home have about 100 hidden layers.

Each neuron, except for each input neuron, is modelled as a summation of weighted input signals multiplied by an activation function just like with biological neural networks.

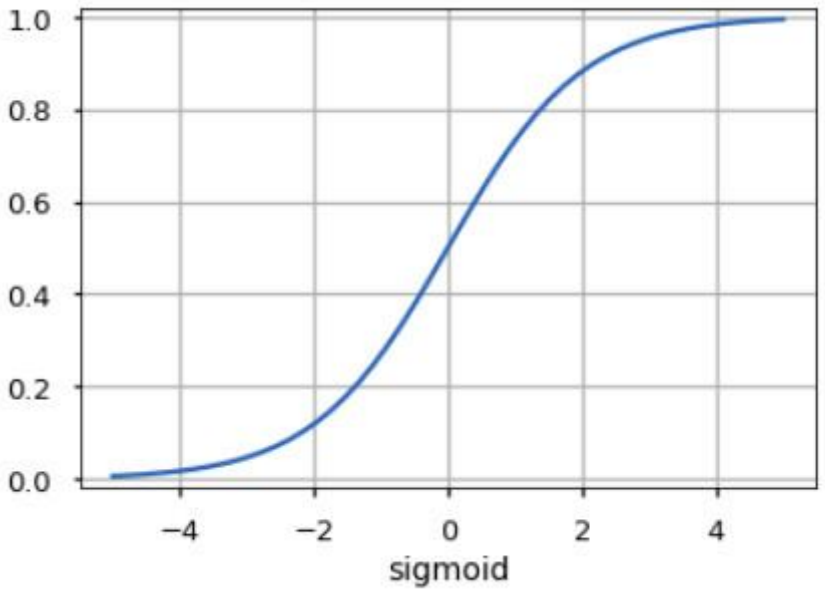


An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

A neuron (perceptron) [Ref. 3]

There are many types of activation functions and much research goes into which type is best for the type of problem being solved.

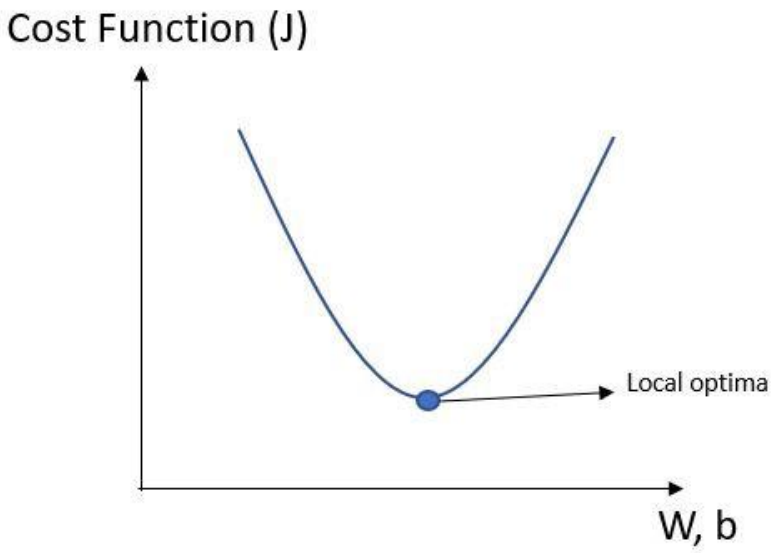
A very common activation function used with ANNs is the sigmoid function.



Sigmoid activation function [Ref. 3]

Given these details of the structure of an ANN, weights and biases are calculated in order to minimise a "cost" function which is also known as a "loss" function.

This function takes many forms. For example it could be the sum of the square of the difference between the predicted output and the actual output as used in the linear regression case.



Cost function [Ref. 6]

The process has these steps:

1) Initialization of all weights and biases with random numbers between -1 and 1

2) Forward propagation

3) Backwards propagation

4) Repeated application of steps 2 and 3

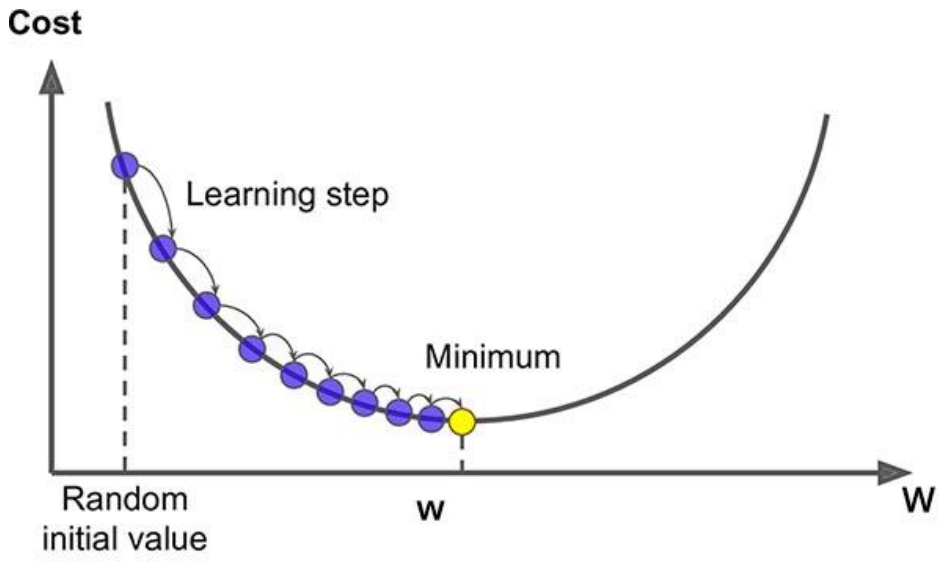
Steps 2 and 3 are repeated until the cost function is minimized. Each repetition is called an **epoch** which is referred to as a hyperparameter. It is important to note that the complete dataset is processed during each epoch.

Forward propagation is simply applying the input and calculating all downstream values to calculate a predicted output.

Backwards propagation is more complicated. The Calculus chain rule is used in conjunction with the error between the known output and predicted output to update the weights and biases.

During each application of backwards propagation, the error moves towards the minimum. This is known as **gradient decent.**

Another important hyperparameter is called the **learning rate** which determines the maximum change of weights and biases during each backwards propagation. If it is too small then then calculation time is unnecessarily extended. If it is too large then the minimum can be overshot and calculations become unstable.



Learning rate [Ref. 5]

Training is very time consuming and requires a lot of computing power. My most recent project was an ANN to detect and read car licence plates and is a real-world application. My home computers just froze so I resorted to Google Colab which is a free cloud computing service. Training took 3 hours and 15 minutes.

Once a network is trained then predicted values are estimated with application of a single forward propagation step. This is called **inference**. This process is much easier and quicker taking fractions of a second.

Inference is a good word to use because results predicted by a neural network are provided in terms of a probability. In the 1990s probabilities were measured in the range of 60% and now 95% + seems typical.

Some concluding remarks:

The idea of artificial neural networks has been around since the late 1890s. As late as the 1980s artificial neural networks were in an academic "dead zone". It wasn't until the late 1990s that applications moved from academic study to real-world application. Within the last 5 years, home experimentation with real-world applications has become possible. Hardware is in the region of $100s and software is free.

Careers in artificial intelligence are no longer limited to academics and now include opportunities in, for example, engineering and business. Applications are wide ranging from agriculture to medicine including detection of cancerous cells.

If a dataset is available, a neural network can be trained. In some cases even if the dataset only includes the input, a neural network can be trained ("unsupervised" learning as opposed to "supervised" learning).

For students interested in learning neural networks, vision systems, in particular object detection applied to robotics, is a good place to start.

Once a robot can 'see' it can be programmed on what to do. For example, self-driving cars can detect stops signs and thus be programmed to stop.

A good place to start learning details is a book called, "Make Your Own Neural Network" by Tariq Rashid. I read it twice when I started learning how to use neural networks to control machinery in April 2021.

I hope to demonstrate one or two examples from this book that can be run for free on Google Colab.

From there, I'm hoping to help students incorporate neural networks into their robotics programming.

I've just started to investigate object detection with $64 of hardware in a package just a little bigger than a stick of gum.

[1] Wikipedia

[2] https://medium.datadriveninvestor.com/machine-learning-101-part-1-24835333d38a

[3] https://learnopencv.com/understanding-feedforward-neural-networks/

[4] https://www.jeremyjordan.me/nn-learning-rate/

[5] https://www.machinelearningman.com/post/best-explanation-of-gradient-descent-for-beginners

[6]https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de