Cheat sheet for AI web application

Neural Networks

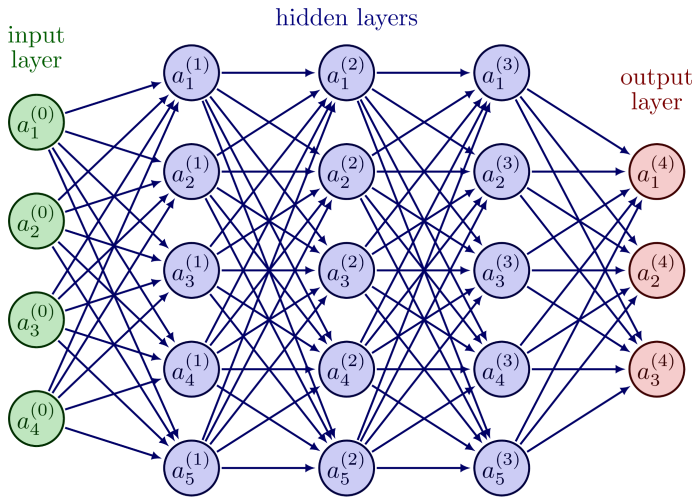
An algorithm that converts matrices of input data into matrices of outputs by running it through a sequence of weights and biases. Weights and biases can be thought of as a series of ‘gradients’ and ‘constants’ in a neural network

Inputs

The set of data that needs to be inputted into the neural network. The inputs need to take the form of matrices to be inputted to the neural network. The inputs can be split into training data and testing data. The training data can be used to train the neural network to change the weights and biases of a model. The testing data can be used to test the accuracy of the model and whether the model is able to predict the outputs correctly

Outputs

The matrix that is outputted by the neural network. The outputs are run through the series of weights and biases to create a matrix of outputs. The outputs also can be split into training data and testing data.



Weights and Biases: The Basis of Neural Networks

The weights can be thought of as a set of ‘gradients’ that connect to the inputs to a set of nodes. During this, the input data is multiplied with the series weights to output a matrix in the nodes. The biases can be thought of as a set of ‘constants’ that are added to the nodes of the neural networks to make the data more accurate. Both the weights and biases are altered in the process of the neural network ‘learning’ as explored in Backpropagation and Gradient Descent.

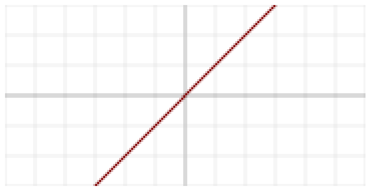
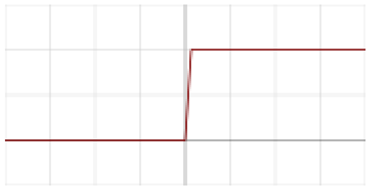
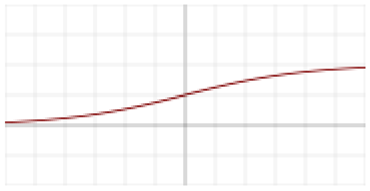
Loss Functions

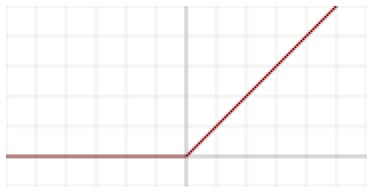
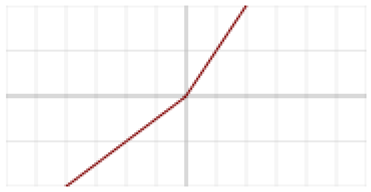
When we compare the values outputted by the neural network and the actual data, there will be a deviation from the predicted data and actual data. We can use a function as a sort of measure of the deviation called a Loss function. After it is passed through this function, this value is called the cost In a neural network, we train the model to minimize this cost function to make the model as accurate as possible. There are many different cost functions that can be used. An example would be the mean sum error loss function, which subtracts the intended output from the actual output and squares the value.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Activation Functions

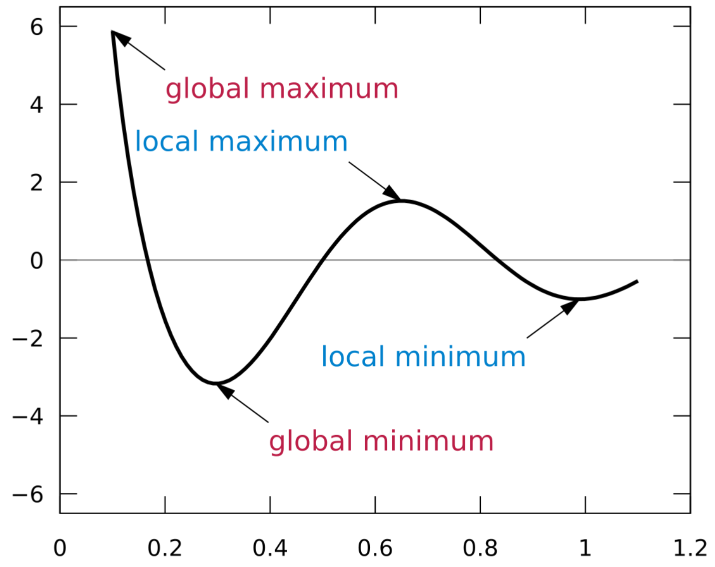
Activation functions are functions applied to the nodes of the neural network to squeeze the values after multiplication between weights between 0 and 1. This is necessary in order to make sure the values of the nodes are applied to a sort of distribution curve to ensure the neural network functions properly. Below are a list of common activation functions that either form the foundation or are activation functions that are used in modern research.

Name	Plot	Function
Identity		g(x) = x
Binary Step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Sigmoid		$\sigma(x) = \dfrac{1}{1+e^{-x}}$

RELU (Rectified Linear Unit)		
Leaky rectified linear unit (Leaky ReLU)		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$

Backpropagation and Gradient Descent

Backpropagation and Gradient descent are the processes through which the machines
The aim of gradient descent is typically for a model to optimise to the local minimum of the cost function to minimise the cost function



Let’s say you are looking at a 2D plot - the local minimum is the most minimised value in a particular domain, in this case possibly from 0.6<x<1.2, the local minimum is -1.

Gradient descent differentiates the cost function with respect to the weights and the activation function with respect to the weights and the weights themselves and compiles the gradients using the Chain Rule in calculus and multiplies a learning rate to the differential equation which will be the size of ‘steps’ the model will take towards the local minima. The smaller the learning rate, the more accurate the steps, but the greater the iterations needed to train and the longer the training time.

Imports and Libraries

There are 2 main libraries that developers use to create machine learning models and neural networks: Pytorch and Tensorflow. These libraries make it convenient and accessible for people to create these models through many functions.

Libraries that may need to be used:

- Pytorch
- Tensorflow
- Keras (under tensorflow)
- Numpy
- Sklearn
- Matplotlib
- Random
- Pandas