Neural Network implementation

Daan Zimmerman van woesik & Noah titarsole

**Call tree**

Main

….Neural\_network.\_\_init\_\_

….….Input\_Neuron.\_\_init\_\_

….….Neuron.\_\_init\_\_

….….Neuron.\_\_init\_\_

….Neural\_network.connect\_all

….Import\_data\_from\_file

….convert\_names\_to\_correct\_output

….Neural\_network.train

….….Neural\_network.set\_input\_nodes

….….Neural\_network.feed\_forward

….….….Neuron.feed\_forward

….….….….sigmoid

….….Neural\_network.back\_propagation\_last\_layer

….….….Neuron.back\_propagation\_last\_layer

….….….….sigmoid\_derivative

….….….……..sigmoid

….….….……..sigmoid

….….Neural\_network.back\_propagation

….….….Neuron.back\_propagation

….….….….sigmoid\_derivative

….….….……..sigmoid

….….….……..sigmoid

….….Neural\_network.update\_neurons

….….….Neuron.update\_neuron

….….Neural\_network.test\_dataset

….….…. Neural\_network.set\_input\_nodes

….….….Neural\_network.feed\_forward

….….….….Neuron.feed\_forward

….….….….….sigmoid

….Neural\_network.get\_percentage\_correctness

….….Neural\_network.set\_input\_nodes

….….Neural\_network.feed\_forward

….….….Neuron.feed\_forward

….….….….sigmoid

**Documentation**

Omdat ik alle opdrachten van deze week allemaal **tegelijk** uitvoer in een bestand ga ik niet alle functies maar **één keer** helemaal door. Omdat ik de Iris data de laatste opdracht is van de drie opdrachten die je moest maken, zal ik deze doorlopen.

Eerst wordt er een **Neural Network** aangemaakt en worden de input neurons, hidden layer configuratie en outputs gedefinieerd.

In de **init** worden deze getallen gebruikt om de **Neurons** te maken. Eerst wordt er een 2d list aangemaakt die de hoeveelheid van de layers aanmaakt. Daarna worden deze gevuld met Neuronen. Als eerst worden de **Input Neurons** gedefinieerd en aan het begin gezet van deze lijst. Daarna gaat hij de configuratie van de hidden layers af. Hier wordt een lijst meegegeven die aangeeft hoeveel Neuronen per layer zitten. Deze worden aangemaakt voor alle hidden layers. Als laatst worden de **Output Neurons** aangemaakt en ook in list met alle Neuronen aangemaakt.

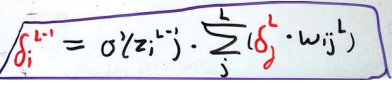
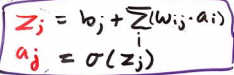
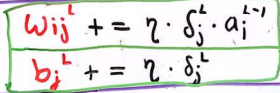
Na de init wordt de **connect\_all** functie toegevoegd. Deze functie is enorm handig en erg simpel. Eerst worden alle neuronen behalve de input neuronen bekeken en krijgen ze de volgende en vorige laag mee. Alle connecties die hier staan worden gebruikt om de berekeningen te maken.

Daarna wordt de **import\_data\_from\_file** functie aangeroepen. Deze functie haalt de data op van een file en wordt de data die wordt opgehaald genormaliseerd. Daarnaast worden ook de namen gesplitst en terug gegeven.

Nu wordt de data van de vorige functie gesplits in de dataset variable en correct\_output variable. De correct variable wordt alleen eerst nog in de **convert\_names\_to\_correct\_output** functie gehaald en zorgt ervoor dat ik van 3 verschillende (bijvoorbeeld) strings een lijst terug krijg met welke index goed is. Bijvoorbeeld als [[‘a’], [‘b’]] in de functie gaat krijg ik [[0,1],[1,0]] terug. Zo kan ik gemakkelijk de code testen.

Hierna komen we bij de belangrijkste functie aan. De **train** functie. Hier worden meerdere parameters aan meegegeven. Eerst wordt de dataset en correct\_output meegegeven. Daarnaast wordt de **learning\_constant** meegegeven. Hierin worden 2 dingen aangegeven. De initial learning\_constant en de **lower\_factor**. Ieder keer wanneer de **Test\_dataset** (leg ik later uit) een hogere waarde teruggeeft dan de vorige iteration dan wordt de learning constant vermenigvuldig met deze factor. Daarnaast heb je de iterations voor hoevaak het neurale netwerk traint. Ten slotte hebben we de show\_info variable die aangeeft of hij elke iteration informatie moet printen.

De functie begint meteen door de learning\_constant te setten naar de initial learning\_constant. Daarna wordt er een for loop gestart voor alle iterations. Daarna wordt de dataset en correct\_output nagelopen volgens de volgorde van Marius.

Als eerst worden de inputs geset naar de waardes van de dataset door **set\_input\_nodes**. Daarna wordt de **feed\_forward** gedaan. Deze functie volgt deze 2 formules die hier rechts staan beschreven. Elke layer wordt dan bekeken op volgorde en wordt daarna de **feed\_forward** uitgevoerd op elke Neuron. De a\_value van elke neuron wordt hiermee geupdate. Hierna wordt de **back\_propagation\_last\_layer** functie uitgevoerd. Hiermee worden de neuronen in de laatste layer hun delta’s berekent. Daarna wordt de **back\_propagation** van de voorgaande neurons berekend. Hier worden de delta’s van de next\_layer gebruikt om de delta van zichzelf te berekenen. Ten slotte worden de weights en biases geupdate door deze functie. Deze zorgt ervoor dat de volgende cycles een betere feed\_forward gedaan wordt.

Dan komen we nu aan bij de **Test\_dataset** functie. Deze functie berekend de sum van alle fouten van alle output nodes. Oftewel er wordt voor elke output node de uitkomst van het neurale netwerk **min**  de correcte output gedaan. Daarna wordt die absoluut gemaakt en al deze getallen gesumt.

Deze waarde wordt ik test\_results gedaan en daar kunnen we eventueel wat data uit halen. Zoals we hierna doen kijken we of de iteration die nu bezig is slechte impact heeft op de test\_dataset functie in vergelijking met de vorige iteration. Als dit zo is dan betekend dit dat de learning\_constant te hoog is. Dan wordt de learning\_constant vermenigvuldigd met de lower\_factor. Wanneer dit gedaan is wordt er wat data geprint als variable show\_info aan staat.

Ten slotte wordt de functie **get\_percentage\_correctness** uitgevoerd. Deze berekend hoe goed het algoritme is op basis van de output neurons. Dit percentage wordt gereturned samen met de raw\_output van de nodes voor verdere evaluatie.