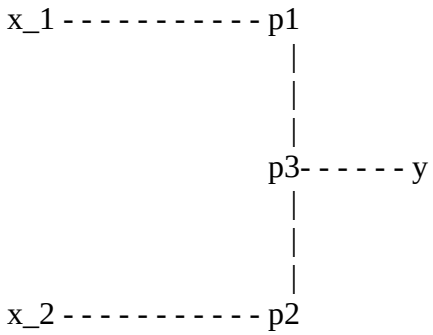
NN.MLP.01: Perzeptron-Netze (2P)

Auffälligkeit des Graphen +1 ist klassifiziert bei x_1 > 2 und x_2 > 3.
unsere 3 Perzeptrons sind also
p1 bedingung x_1 > 2
p2 bedingung x_2 > 3
p3 ODER verknüpfung p1 oder p2

Skizze des Netzes :



p1 :

$z1 = 0 \cdot x1 + 1 \cdot x2 - 3$
$h1 = \text{sign}(z1)$

Gewichte und Bias

- $w1 = (0, 1)$
- $b1 = -3$

p2 :

$z2 = 1 \cdot x1 + 0 \cdot x2 - 2$
$h2 = \text{sign}(z2)$

Gewichte und Bias

- $w2 = (1, 0)$
- $b2 = -2$

p3 :

$z3 = 1 \cdot h1 + 1 \cdot h2 + 0$
$y = \text{sign}(z3)$

Gewichte und Bias

- $w3 = (1, 1)$
- $b3 = 0$

NN.MLP.02: Vorwärtslauf im MLP (2P)

Dimensionen der Gewichtsmatrizen und Bias-Vektoren

Gewichtsdimensionen:

- W[1]: 64×25
- W[2]: 32×64
- W[3]: 4×32

Bias-Dimensionen:

- b[1]:64
- b[2]:32
- b[3]:4

Mögliche Problemlösung :

Das Netztwerk könnte beispielsweise zur interpretation von Sensordaten benutzt werden. Wir haben 25 Sensoren die ein System überwachen (bspw. Photovoltaikanlage) und aus diesen 25 Sensordaten könnte man dann 4 resultierende Werte vorhersagen z.B.

- aktuelle Leistung (tatsächlich erzeugte Watt)
- Temperatur der Anlage (Effizienzverlust bei zu stark erhitzen bauteilen)
- Vorhersage der Erzeugten Leistung (vergleich aktuelle leistung mit berechnet möglicher Leistung)
- Verschleiß der Anlage (bei differenz zwischen tatsächlich erzeugtem und Vorhergesagten

NN.MLP.03: Tensorflow Playground (6P)

1. keine ahnung was hier hin soll. Hab ich gemacht

2. Habe eine Menge mit der Seite rumgespielt und ausprobiert was so möglich ist und muss sagen es wäre eine unendliche Schreibarbeit das alles festzuhalten deswegen bearbeite ich die aufgabe anhand von einem Beispiel.

l = layer,

n=neuronen,

learning rate = 0.001,

a = aktivierungsfunktion,

Vergleich nach 1 Epoch

te = testloss,

tr = trainingloss,

train to test ratio = 20%

bester von 5 durchläufen

Circle

l = 1, n = 2,

a = ReLU , te = 0.033, tr = 0.019

a = Tanh, te = 0.073, tr = 0.045

a = Sigmoid, te = 0.313, tr = 0.291

Von meinem Experimentieren wirkt es so als wenn ReLU sehr schnell auf ein annährend genaues ergebnis kommt aber dafür auch schneller als der Rest auf einem Plateau stecken bleibt bei der Genauigkeit und Sigmoid ist das genaue gegenteil, Sigmoid braucht relativ lange um zu einem genaueren Ergebniss zu kommen aber wird dafür seeeeehr lange besser und plateaud dementsprechend sehr langsam. Tanh ist quasi eine Mittellösung zwischen den anderen nicht so präzise wie Sigmoid und nicht so schnell wie ReLU aber dafür besser als die beiden in deren Schwachpunkten.

Bei den unterschiedlichen Testbedingungen gibt es natürlich noch sehr viele andere dinge die einem dann auffallen. Bspw. Sigmoid braucht EWIG (bei learning rate 0.01 ca. 3,000 E bis überhaupt etwas passiert) um überhaupt irgendwas zu klassifizieren bei 4 Layern mit 7 Neuronen, aber sobald dann irgendwas passiert wird es rasant besser, hat also starke Startschwierigkeiten. Auch die Anderen Aktivierungsfunktionen verhalten sich ähnlich wie beim kleineren beispiel nur die unterschiede treten noch stärker auf.

Circle, Noise = 15

l = 1, n = 2,

a = ReLU , te = 0.081, tr = 0.077

a = Tanh, te = 0.084, tr = 0.069

a = Sigmoid, te = 0.188, tr = 0.181