Opdracht - Logic Operators met ADALINE Netwerk

Vandaag gaan we een neurale netwerk maken die inputs van logic operations kan uitvoeren. Logic operations die onze neurale netwerk aan het einde van de opdracht kan doen zijn AND, OR en NOT operations. De neurale netwerk die wij gaan maken is een single-layer netwerk, de ADALINE.

Data

Inputs

De input data voor onze logic operations zijn als volgt:

```
inputs = np.array([[+1, -1, -1],
                   [+1, -1, +1],
                   [+1, +1, -1],
                   [+1, +1, +1]])
```

Dit is heel weinig data, maar we dekken alle mogelijke stappen die we kunnen doen met een logic operation. True : True, True : False, False : True en False : False.

Desired Output

De verwachtte output die we aan het netwerk geven, definieert ook gelijk de soort logic operation. Hieronder zien we een OR operation die we het netwerk gaan leren:

```
desiredOutputs = np.array([-1, +1, +1, +1])
```

Als we dit willen veranderen naar een AND, dan zou de volgende array aan verwachtte outputs het geval zijn:

```
desiredOutputs = np.array([-1, -1, -1, +1])
```

Variabelen Definiëren

Bij een ADALINE netwerk moeten we bepaalde variabelen definiëren voordat we het netwerk gaan maken. De variabelen zijn: aantal inputs, de leer tempo, start gewichten, activatie functie en delta rule.

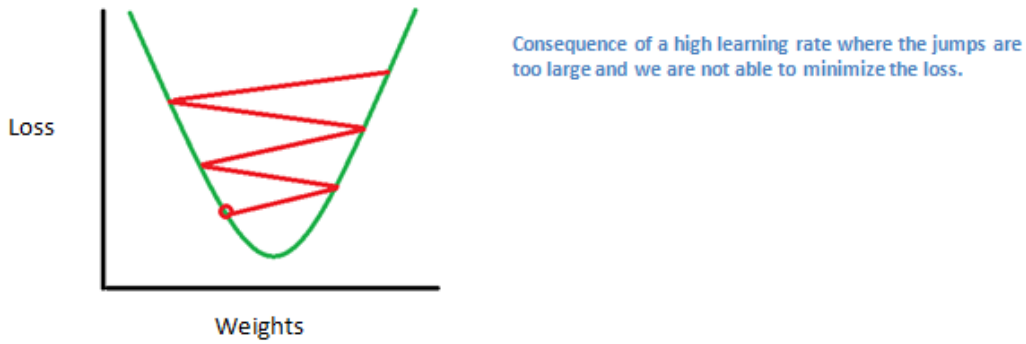
Aantal inputs

De logic operations die wij willen voorspellen met onze neurale netwerk hebben allemaal twee inputs nodig. Ook heeft de ADALINE zelf een input die altijd aanwezig moet zijn met een waarde van +1. Dit komt dus uit op drie inputs. We kunnen inputs het beste definiëren in arrays. Dus als we met een AND gate werken en de inputs die we gaan testen hebben eerst een true dan een false, dan ziet dat bij onze input in de ADALINE eruit als : [+1, +1, -1]. De eerste +1 is van de verplichtte input van een ADALINE, de tweede waarde is onze true en de laatste waarde is de false. Gebruik numpy arrays om dit soort arrays te maken. Vb: (np.array([+1, +1, -1])

Leer tempo

De leer tempo is bedoelt om van elk stuk data een klein beetje te leren. Als we geen leer tempo bepalen, dan leert de neurale netwerk eigenlijk elke keer te snel. Elk volgende stuk data wordt aangenomen als de totale waarheid, terwijl we juist willen dat het netwerk leert door middel van alle vorige ervaringen.

Probeer de leer tempo tussen de 0 en 1 te houden. Hoe hoger deze waarde, hoe sneller het netwerk tot een oplossing komt. Maar een te hoge waarde kan betekenen dat het netwerk elke keer het optimale waarde overschiet, omdat het te snel leert. Hieronder zien we een voorbeeld van een netwerk dat een te hoge leer tempo heeft.



Figuur 1 - Grafiek|Foutmarge als functie van gewicht

Start gewicht

Om te beginnen met het trainen van onze neurale netwerk, hebben we eerst gewichten nodig op de verbindingen tussen neuronen. Als begin gewicht kunnen we random getallen nemen die niet te hoog zijn, dus tussen de 0-1. Ook de gewichten kunnen we bijhouden in een array, gebruik weer een numpy array om gewichten van verbindingen bij te houden.

Activatie functie

De activatie functie staat simpelweg voor de beslissing die we maken aan het eind. Bij het voorspellen willen we een definitief antwoord hebben op onze vraag. Geven de ingevoerde waardes een true of een false bij de output. De activatie functie die wij gaan gebruiken is heel simpel. We weten de net waarde al, dit is de optelling van alle inputs vermenigvuldigd met de gewichten. Als de net waarde boven de 0 zit, dan geven we een true, oftewel 1. Als de net waarde gelijk is aan 0 of eronder ligt, dan geven een een false, oftewel -1.

Delta rule

De delta rule houdt in, het berekenen van de foutmarge en hiermee gewichten binnen het netwerk aanpassen. De foutmarge bij de ADALINE is het verschil tussen de verwachtte waarde en uitkomst van de net waarde.

Net is hier de optelling van alle inputs die vermenigvuldigd zijn met de gewichten. De verwachtte waarde is een +1 of -1. Bij een AND operation verwachten we een +1 als beide inputs een true zijn en een -1 in alle andere gevallen.

Nadat we de foutmarge hebben berekent moeten we per verbinding het gewicht aanpassen. Met de volgende formule zien we de verandering delta W die we moeten aanpassen bij de gewichten.

$$\Delta w = \frac{n}{P} \sum_{p=1}^{P} \delta_p x_{ip}$$

Figuur 2 - Formule Verandering in Gewicht

De N staat voor de leer tempo die we gedeclareerd hebben. De grote P staat voor aantal data stukken waarvan we de foutmarge willen berekenen, voordat we het gaan doorrekenen in de gewichten. Dus als we de neurale netwerk per data stuk willen trainen, dan wordt de grote P een 1. De δ staat voor de fout marge die we gerekend hebben. De Xi staat voor de input waarvan we het gewicht gaan veranderen en de p staat voor het data stuk waar we zijn. Als we de neurale netwerk per stuk data willen veranderen, dan kunnen we de formule ook opschrijven als :

$$\Delta w = n * \delta_p x_{ip}$$

Figuur 3 - Formule Verandering in Gewicht per Input Data

Voorspellen

Voorspellen met de ADALINE is heel simpel. We hebben een array aan gewichten, deze gewichten zijn aan het einde van het train proces geconvergeerd, op een optimale oplossing gekomen. Tijdens het train proces moeten we ook voorspellen met het netwerk. Dit doen we dan met gewichten die nog niet kloppen. De array aan gewichten vermenigvuldigen we met de inputs die gegeven worden. Als de inputs ook arrays zijn, dan is de taak die we uitvoeren alleen maar een vermenigvuldiging van arrays.

Vb: Gewichten zijn [0.2, 0.5, 1] en de inputs zijn true en false [+1, +1, -1]. De voorspelling wordt dan 0.2 * 1 + 0.5 * 1 + 1 * -1 = -0.3. Met onze activatie functie zien we dat -0.3 kleiner is dan 0, dus geven we een False bij onze output.

Tip: de numpy functie [numpyArray].dot([numpyArray]) doet dit al heel snel voor je.

Leren

Elke training die we doen moet de eerder gedefinieerde gewichten veranderen. Dit doen we door eerst een voorspelling te maken met de test data. Met de uitkomst van de voorspelling (de net waarde, niet de output) gaan we kijken wat we nou eigenlijk hadden verwacht bij de output. Vervolgens voeren we de delta rule uit. Eerst berekenen we de foutmarge en vervolgens voeren we de formule uit die we eerder hebben gezien. De uitkomst van de formule is dan de verandering die doorgevoerd moet worden naar het gewicht van de verbinding.

Vb: net waarde is -0.3. We verwachtte dat de output een false zou zijn, dus een -1. De foutmarge is dan -1 - - 0.3 = -1 + 0.3 = -0.7. De foutmarge gebruiken we vervolgens in de formule. Leertempo * foutMarge * gegevenInput. Als we een leerTempo hadden aangenomen van 0.1 en de gegevenInput van de verbinding -1 was, dan is de verandering het volgende: 0.1 * -0.7 * -1 = 0.07. Het gewicht wordt dus veranderd met een waarde van 0.07.