Neurale Netwerk Opdracht van week 10

Opdracht: Bouw een neuraal netwerk met generatieve AI

Doel: Het doel van deze opdracht is om een basisbegrip van neurale netwerken te ontwikkelen door een eenvoudig neuraal netwerk te implementeren, zonder gebruik te maken van backpropagation en gradient descent. Er dient voor het trainen van het model een simpel algoritme gebruikt te worden. Je moet de code zelf goed kan uitleggen.

Instructies:

- Gebruik de requirements verderop in dit document om een LLM (bijvoorbeeld chat-gpt, Gemini van google etc.) te instrueren om een NN (Neural Network) te bouwen.
- Bestudeer de code die je van de LLM terugkrijgt. Gebruik de bullet points uit het hoofdstuk 'Stappen' verderop in dit document om de code te begrijpen.
- Begrijp de code en controleer of de code klopt. Let op! De LLM's genereren niet altijd correcte code. Gebruik de theorie uit de hoorcolleges en google om meer te weten te komen over NN.
- Wees creatief met het gebruik van de LLM's. Vraag bijvoorbeeld:
    - Of het stukken code kan uitleggen als je het zelf niet begrijpt
    - Voeg de code toe in een nieuwe chat sessie en vraag of deze code klopt!
    - Vraag of specifieke regels code kloppen als je vermoed dat het niet klopt of niet conform de theorie is uit de slides
    - **Samenvattend, het doel is om zoveel mogelijk van de code te begrijpen**

Requirements (verwerk in je LLM prompt):

Het instrueren wat de LLM moet opleveren kan je zelf bedenken. Zorg in ieder geval dat je de volgende requirements meegeeft. Niet klakkeloos copy-pasten, want niet alle zinnen zijn als geheel nuttig voor je prompt.

1. NN heeft 4 input nodes, 1 hidden layer met een door de student gekozen aantal nodes en 1 output node.
2. Gebruik 1 tot 5 input datapunten met bijbehorende output (antwoorden)
3. Maak gebruik van arrays
4. Laat de LLM code opleveren in C# of Java

5. We raden het niet aan maar voor de Python die-hards, mag het ook in Python. Echter, zorg dat je of Numpy gebruikt als library of, een andere structuur die arrays vervangt. Zie de opmerking hieronder voor meer info hierover.
6. Gebruik geen externe bibliotheken voor het bouwen van het neurale netwerk, met uitzondering van #5 als je daarvoor kiest.
7. Instrueer de LLM dat het geen backpropagation mag gebruiken en ook niet Gradient Descent algoritme. Vraag of het een simpele manier kan gebruiken op basis van de 'error' om de gewichten te trainen. Zie hoofdstuk Trainen verder in het document

- Opmerking: NN worden vooral met python gemaakt, maar omdat er geen kennis is opgedaan tijdens de hoorcolleges over deze speciale libraries (zoals tensorflow, keras etc.), houden we het bij C# of Java om de basisconcepten aan te leren.

Stappen:

De volgende algemene stappen zou je terug moeten kunnen vinden of herkennen in de gegenereerde code van je NN. Dit is een hulpmiddel voor je om de code te begrijpen. Het is niet erg als jou gegenereerde code hier iets van afwijkt.

1. Definieer de structuur van het neurale netwerk, inclusief het aantal input nodes, het aantal nodes in de hidden layer en het aantal output nodes.
2. Initialiseer de gewichten van het netwerk willekeurig.
3. Implementeer de feedforward-methode om de input door het netwerk te sturen en de output te berekenen.
4. Bereken de error of fout tussen de voorspelde output en de werkelijke output.
5. Pas de gewichten niet aan met behulp van backpropagation en gradient descent. In plaats daarvan kunnen de studenten ervoor kiezen om de gewichten met een eenvoudige regel aan te passen.
6. Train het netwerk met behulp van de gegeven training samples en evalueer de prestaties ervan.

Formeel Datapunt (FDP-3)

- Bij de volgende FDP-3 kan er door de docent o.a. vragen gesteld worden over de code die je hebt laten generen. (zorg dat je een versie hebt zonder commentaar erbij)

Trainen

Het kan zijn dat de LLM alsnog, direct of indirect, de backpropagation en gradient descent trainingsalgoritme gebruikt. Andere termen die hier direct te maken mee hebben zijn de ‘afgeleide’ (in het engels de derivative).
Dat zie je als je bijvoorbeeld het volgende ziet in de code die de LLM genereerd:

- inputToHiddenWeights[i, j] += error * input[i] * hiddenOutput[j] * (1 - hiddenOutput[j]);

Metname als het **“… (1 - …)”** gedeelte.

Je wil het liefst code zien dat er als volgt uit ziet:

- inputToHiddenWeights[i, j] += error * input[i] * hiddenOutput[j];