2024

Programming NNs

IMPLEMENTATIE VAN EEN NEURAAL NETWERK VOOR DE IRIS-DATASET

AHMET SERDAR ÇANAK EN ROEL STIERUM

1. Inleiding

Dit verslag beschrijft de implementatie van een neuraal netwerk voor de classificatie van de Iris-dataset. De opdracht is uitgevoerd door Ahmet Serdar Çanak en Roel Stierum in samenwerking. Het bestand is genoemd als "Programming NNs_AhmetSerdarÇ_RoelS.pdf".

2. Uitleg van de Code

2.1. Call-Tree

__main__
……..load_dataset()
……..normalize()
……..serialize()
……..split_by_percentage()
……..split_by_percentage()
……..perf_counter()
……..neural_network()
……..neural_network__init__()
……..neural_network.train()
……..show_progress()
……..predict()
……..hidden_neuron.forward_propagation()
……..output_neuron.forward_propagation()
……..back_propagation()
……..sigmoid_derivative()
……..sum()
……..sigmoid_derivative()
……..update_weights()
……..hidden_neuron.update()
……..output_neuron.update()
……..perf_counter()
……..neural_network.predict()
……..neural_network.predict()

2.2. Totaal-uitleg

De `**data_loader**` class wordt gebruikt om de Iris-dataset te laden(**load_dataset**), normaliseren(**normalize**) en splitsen(**split_by_percentage**) in trainings- en testgegevens. De uitvoertypen worden omgezet naar binaire waarde voor classificatie (**serialize**).

Er wordt een **neural_network** object gemaakt. Dat object is een netwerk opgebouwd uit lagen met neuronen. Tijdens het initialiseren van de lagen worden **neuron** objecten gemaakt. Met dit netwerk kunnen classificaties gemaakt worden.

Om correcte classificaties te maken met het **neural_network** object moet het object eerst getraind worden. Dit kan doormiddel van de **train** functie. De **train** functie voert als eerst forward_propagatie (**predict** en **forward_propagation**) toe per neuron. Vervolgens voert het **back_propagation** uit per

neuron en tot slot worden alle gewichten geüpdatet (**update_weights** en **update)**. Deze stappen worden uitgevoerd voor het aantal epochs wat gedefinieerd is

Nadat er getraind is kunnen we beginnen met classificaties. We testen nu de nauwkeurigheid van het model wat getraind is met de **train_data**. De classificatie tests worden uitgevoerd op de aan de kant gezette **test_data**.

3. Reproduceerbare Resultaten

1. **Trainen van het neurale netwerk**

   - In dit voorbeeld is het neurale netwerk getraind met 85% van de genormaliseerde Iris-data.

   - Het heeft een inputlaag van 4, een verborgen laag van 8 neuronen en een uitvoerlaag van 3 neuronen.

   - Het leerpercentage is ingesteld op 0.1 en het trainingsproces duurt ~1 seconde.

2. **Evaluatie van nauwkeurigheid**

   - De nauwkeurigheid van het getrainde netwerk op de volledige dataset en de test-set berekend.

   - Het netwerk is 100% correct in de classificatie van de Iris-bloemen in de test-set bij 100 epochs en een learning rate van 0,1.

   - Het netwerk is 72.7% correct in de classificatie van de Iris-bloemen in de volledige dataset bij 100 epochs en een learning rate van 0,1.

   - Het netwerk is 87% correct in de classificatie van de iris-bloemen in de test-set bij 500 epochs en een learning rate van 0,1.

   - Het netwerk is 96.7% correct in de classificatie van de Iris-bloemen in de volledige dataset bij 500 epochs en een learning rate van 0,1.

3. **Individuele Voorspelling**

   - Voor een specifieke index (index 50) is de echte klasse en de voorspelde percentages voor elke klasse weergegeven.

4. Conclusie

De implementatie van het neurale netwerk voor de Iris-dataset is succesvol uitgevoerd. Het netwerk behaalde een bevredigende nauwkeurigheid en was in staat om individuele voorspellingen te doen met 96,7% nauwkeurigheid bij 500 epochs.